TAROS 2008
Towards Autonomous Robotic Systems 2008

The University of Edinburgh
September 1\textsuperscript{st} – 3\textsuperscript{rd} 2008

Subramanian Ramamoorthy and Gillian M. Hayes (editors)
The articles in this volume are copyright ©2008 by their respective authors and all opinions expressed therein are those of the authors. No part of this book may be reproduced in any form without permission in writing from the respective authors.

ISBN 978 1 906849 00 9 (softbound)
978 1 906849 01 6 (CD)

Published by:
School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB
Scotland, UK

s.ramamoorthy@ed.ac.uk, g.hayes@ed.ac.uk

Front cover image credits (clockwise from upper left):
3D Bonn marketplace (courtesy of Research Institute for Applied Sciences FGAN, Wachtberg, Germany)
Oculomotor workspace (courtesy of Department of Computer Science, Aberystwyth University)
Informatics Forum spiral staircase (courtesy of School of Informatics, University of Edinburgh)
Titan four-wheel drive outdoor robot (courtesy of School of Computer Science and Software Engineering, University of Wollongong)
Foreword

We are happy to welcome you to the beautiful city of Edinburgh to take part in “Towards Autonomous Robotic Systems 2008”. TAROS 2008 is the ninth in this series of conferences, which started in Manchester in 1997. Since then they have become a regular fixture in the calendar, providing the opportunity for robotics researchers from the UK and further afield to discuss the latest advances in robotics. In 2008 we are very pleased to welcome the TAROS community to Edinburgh and, in particular, to the University of Edinburgh’s brand-new School of Informatics building, the Informatics Forum.

The TAROS series started life as TIMR – “Towards Intelligent Mobile Robots” – with the first conference in Manchester in 1997 and subsequent conferences taking place biannually. In 2003 the conference became an annual event and the name was changed to TAROS, in recognition of the wide range of aspects of autonomy in robotics in addition to mobiles. This year papers on robot learning, active vision, touch perception, artificial muscles and lightweight novel motors as well as on mobiles (wheeled, legged and aerial, singly and in groups) give some indication of the breadth of topics covered. We are also delighted to have papers from some 10 countries: TAROS has truly become international.

This year, we devote our first session to the topic of Personal and Service Robotics (PSR) and we host the annual meeting of the Virtual Research Centre in Personal Robotics, an EPSRC-funded project that was introduced last year to the TAROS audience. We are also privileged to have two distinguished visiting speakers: Roland Siegwart from the ETH Zürich, speaking on “Robots for real world environments” in the context of the PSR session, and Kasper Støy, from the University of Southern Denmark, speaking on “Modular robots in action”.

We would like to acknowledge with thanks the support provided by the School of Informatics of the University of Edinburgh. TAROS is taking place in the Informatics Forum, the fantastic new home of the School of Informatics, and is one of the first conferences to be hosted here – so we are very grateful to participants for road-testing the facilities. TAROS coincides with the official opening of the Forum, for which many activities are planned.

Finally, it is our great pleasure to acknowledge all the support of the many people involved in the organisation of our conference: Monika Lekuse, Dyane Goodchild, Geraldine Debard and Irene Madison for local organising, Matt Whitaker and the volunteer PhD students and research assistants for help before, during and (no doubt) after the conference. We would especially like to thank the programme committee and reviewers for their time and feedback to authors, and Myra Wilson and Fred Labrosse from TAROS 2007 for invaluable advice. Most of all, we thank the authors and participants for providing the core of the conference: without you there would be no TAROS.

Gillian Hayes, Subramanian Ramamoorthy, Ulrich Nehmzow, Chris Melhuish, Mark Witkowski
Edinburgh, August 2008
Organising Committee

Conference co-chairs:  Professor Ulrich Nehmzow (University of Ulster)
Professor Chris Melhuish (University of Bristol and the West of England)
Dr Mark Witkowski (Imperial College London)

Host organisation chair:  Dr Gillian Hayes (University of Edinburgh)
Programme chair:  Dr Subramanian Ramamoorthy (University of Edinburgh)

Programme Committee

Dr Guido Bugmann  University of Plymouth
Dr Jeremy Burn  University of Bristol
Prof Brian Davies  Imperial College London
Prof Alex Ellery  Carleton University
Dr Dongbing Gu  University of Essex
Prof John Hallam  University of Southern Denmark
Dr Roberto Iglesias  University of Santiago de Compostela
Prof Maarja Kruusmaa  Tartu University, Estonia
Dr Theo Kyriacou  University of Keele
Dr Frdric Labrosse  University of Wales, Aberystwyth
Dr Paul Newman  University of Oxford
Dr Ferdinando Rodriguez y Baena  Imperial College London
Prof Sandor Veres  University of Southampton
Professor Gurvinder Virk  Massey University
Dr Myra Wilson  University of Wales, Aberystwyth
Contents

Session 1: Personal and Service Robotics

Monte Carlo Localization for a Guide Mobile Robot in a Crowded Environment Based on Omnivision
Cristina Gamallo, Carlos V. Regueiro, Pablo Quintía and Manuel Mucientes ................................................................. 1

Gaussian Progress Regression for Facial Behaviour Mapping
Peter Jaeckel, Neill Campbell and Chris Melhuish .................................................. 9

Robot Learning from Environment Interaction and Observation of Human Behaviour
Pablo Quintía, Roberto Iglesias, Carlos V. Regueiro, Eva Cernadas and Miguel Rodríguez................................................................. 17

Towards Binocular Active Vision in a Robot Head
Haitham Fattah, Gerardo Aragon-Camarasa and J. Paul Siebert .............. 25

Invited Speaker 1

Robots for Real World Environments
Roland Siegwart ................................................................. 33

Invited Speaker 2

Modular Robots in Action
Kasper Stoy ................................................................. 34

Session 2: Localization and System Analysis

Robot Localization Using Seismic Signals
Theocharis Kyriacou, Peter Styles and Sam Toon .................................................. 35

A Proposal of a Methodology for the Analysis of Robot environment Interaction Through System Identification
Otar Akanyeti, Iñaki Rañó, Ulrich Nehmzow and Steve Billings .............. 43

Improving 3D Scan Registration for SLAM with Clustering and Deterministic Annealing
Timo Röhling and Dirk Schulz ................................................................. 50
Session 3: Learning I

Application of Unsupervised Clustering to Complex Robot Training Tasks
Ulrich Nehmzow, Otar Akanyeti and Steve Billings ................................. 57

Comparative Experiments on the Emergence of Safe Behaviours
Yuri Gavshin and Maarja Kruusmaa ......................................................... 65

Session 4: Learning II

A Developmental Algorithm for Ocular-motor Coordination
Fei Chao, Mark H. Lee and Joseph J. Lee .................................................. 72

Touch Perception with SOM, Growing Cell Structures and Growing Grids
Magnus Johnsson, David Gil Mendez and Christian Balkenius...................... 79

Session 5: Mobile Robotics

Distributed Control of Multi-robot Systems Using Bifurcating Potential Fields
Derek Bennet and Colin McInnes ............................................................... 86

Combining Coordinated Navigation and Reactive Collision Avoidance for GPS-based Convoying
Frank Hoeller, Timo Röhling and Achim Königs ....................................... 93

Cooperative Target Tracking of Mobile Robots
Zongyao Wang and Dongbing Gu ........................................................... 101

Sliding Mode Control for Agents and Humans
Sandor M. Veres and Nicholas K. Lincoln ............................................... 107

Session 6: Novel Actuators and Their Uses

A New Thin Membrane Rotary Motor for Robotic Applications Based on the Dielectric Elastomer Actuator
Iain A. Anderson, Emilio P. Calius, Todd Gisby, Thomas McKay, Benjamin O’Brien and Scott Walbran ................................................................. 118

Associating SOM Representations of Haptic Submodalities
Magnus Johnsson and Christian Balkenius ............................................. 124

Poster Session I

Fabric Manipulation: an Eye Tracking Experiment
Peter Gibbons, Phil Culverhouse and Guido Bugmann .................................. 130

Monocular Omnidirectional Vision Based Robot Localisation and Mapping
Chris Barbridge, Libor Spacek, Joan Condell and Ulrich Nehmzow ............... 135

A Co-evolutionary Approach Toward Face Localization in Color Images
Farshid Hajati and Soheila Gheisari ......................................................... 142
Internal Agent States: Experiments Using the Swarm Leader Concept
Mohamed H. Mabrouk, Craig W. Murray, Kevin Johnstone and Colin R. McInnes .......................................................... 149

A Mathematical Model, Implementation and Study of a Swarm Conglomerate and Its Formation Control
Blesson Varghese and Gerard T. McKee ................................. 156

Error Surface Generation Techniques for Appearance-based Stabilisation of an Intelligent Kite Aerial Photography Platform (iKAPP)
Heidar Hosseini, Mark Neal and Frédéric Labrosse ..................... 163

Induced Policy Segmentation in Multi-agent Reinforcement Learning with Communicative Actions
Matthew B. Whitaker and Gillian Hayes .................................. 171

Simplification of Artificial Neural Networks Using Neural Complexity Measures
Thomas D. Jorgensen, Barry Haynes and Charlotte Norlund ........... 178

Poster Session II

A Software Architecture for Mobile Robot Navigation
Phillip McKerrow, Shérine Antoun and Patricia Worth ................ 185

Modelling the Lizard Ear: Directness of Phonotaxis with Noise Distractor
Lei Zhang, John Hallam and Jakob Christensen-Dalsgaard ............ 193

Relating Textural Concepts to Tactile Sensors
James Edwards, Jonathan Lawry, Jonathan Rossiter and Chris Melhuish . 201

Microbial-powered Artificial Muscles for Autonomous Robots
Ioannis Ieropoulos, Iain Anderson, Todd Gisby, Cheng Hung Wang and Jonathan Rossiter .................................................. 209

Modelling Soil Traction for More Effective Control of Walking Planetary Rovers
Gregory P. Scott, Chakravarthini M. Saaj and Eddie Moxey ........... 217

Towards Temporal Inference for Shape Recognition from Whiskers
Charles Fox, Mat Evans and Tony Prescott ............................... 226

Programmable Differential Brakes for Passive Haptics
Yaroslav Tenzer, Munish Patel, Brian L. Davies and Ferdinando Rodriguez y Baena ................................................................. 234

A Biologically Inspired Fingertip Design for Compliance and Strength
Craig Chorley, Chris Melhuish, Tony Pipe, Jonathan Rossiter and Graham Whiteley .............................................................. 239

Author Index ............................................................................. 245
Monte Carlo localization for a guide mobile robot in a crowded environment based on omnivision

Cristina Gamallo, Carlos V. Regueiro, Pablo Quintía and Manuel Mucientes

Abstract—This work presents a localization system for a robot guide on a crowded environment based on omnidirectional vision and a map of ceiling landmarks. The developed approach uses a Monte Carlo particle filter to manage uncertainty, both on observations and control, in order to track the position of the robot. We describe how landmarks are detected on each image and how the problem of landmark association is posed. To demonstrate the robustness and reliability of our system, we present experiments carried out in a real environment, the Domus Museum in A Coruña (Spain). Results show that the proposed localization system can run on real time and along middle-long trajectories.

I. INTRODUCTION

Robot localization is one of the most important tasks in autonomous mobile robotic. Most of the action a robot has to perform require the knowledge of its position. Determining the location of a mobile robot is estimating the Cartesian coordinates and angular orientation relative to an external reference frame. It requires to be reliable, robust and executable on real time.

Our system has been developed for a guide robot (Fig. 1) at the Domus Museum (Fig. 2 and 3) located in A Coruña (Spain). This environment is highly populated and, therefore, typical range sensors like laser or ultrasonic devices do not work accurately in normal conditions, we are not allowed to modify the environment introducing artificial landmarks to facilitate localization. Thus we have decided to use artificial vision and natural landmarks of the environment to estimate the position of the robot.

A landmark can be any distinctive and recognizable object on the environment. This work uses the lights placed on the ceiling of the museum (Fig. 2) which are easily detectable, repetitive and usually visible along large trajectories. On the other hand, any building has lights, so there is no need for prior preparation of the environment in order to use our localization method. The main problem is the difficulty to distinguish among different landmarks, as they are usually equal.

The camera is pointing to the ceiling and fixed on the robot over 1.5 m above it (Fig. 1), so that their movements are restricted by the degrees of freedom of the robot (a Pioneer IIAT) and the noise or occlusions generated by moving people is minimized. As our camera is an omnidirectional camera (Fig. 4), it means a very wide field of vision (FOV around 185°) which covers the half space of the environment and obtains a lot of information about it in each acquisition. It is noteworthy that the floor of the environment is very irregular and produces swinging in the camera support. This increases the noise both in the observations (images) and in the estimation of the robot’s position with the odometry system.

A Monte Carlo localization algorithm [15] has been used to solve the tracking position problem. The key-point of the
algorithm is to manage the uncertainty in robot perception and action by means of Probability Density Functions (PDF). In the case of Monte Carlo localization, the PDF is estimated with a particle filter.

Section II exposes an overview of related works. The next three sections describe the vision system and landmark detection on the images, the omnidirectional camera and the Monte Carlo localization process. Section VI presents the experimental results in a real environment, and finally, the last section points out conclusions and future work.

II. RELATED WORK

There has been extensive research in the literature to solve the localization problem using vision. Most of them use landmarks in the environment as a reference to obtain the robot positions. The works that use artificial landmarks were not addressed here since they can not be applied in our environment because of its peculiarities, that we have described in the last section.

Concerning omnidirectional vision to locate a mobile robot, the first work was published in 1986 by Cao et at [4]. Although few related studies were published before the end of the nineties. Nowadays such systems, for example [13], [3], [10], [1], [9] and [12], have become popular due to their low cost in addition to the benefit of having a very wide field of vision.

The probabilistic approach is the most used in recent publications. One example is [14]. In which, a Monte Carlo localization algorithm is suggested to solve global localization problem using a camera. They used a visual map of the ceiling, obtained by mosaicing, and localized the robot using a simple scalar brightness measurement as the sensor input. The camera pointed to the ceiling just the same settings our system has. But unlike the present work, this system is sensitive to bumps and as a result of the small FOV of the camera, there are instants that hardly any lights can be seen. This causes more uncertainty in the system. Another similar approach is [12], which use an omnivision camera oriented to the ceiling too, but it is based on information theory to get the global trajectory. The main problem of this work is the high cost of computing.

[1] and [10] use Monte Carlo filtered too but they create a database with images of every route and their positions. The robot can be localized by correlation between the captured images and the database images in real time. These systems have the drawback that they can not work in other routes on the environment.

Menegatti et at. [9] have developed a system which uses a chromatic map of the floor to compute the robot pose. They obtained similar results but their system is limited a environments with natural color transitions and it is also light-sensitive.

Another popular probabilistic algorithm is the Kalman filter and it was considered in [7], [6], [8] and [11] to implement SLAM using a single camera. A limitation of this system is it need a large number of distinguishable features to perform accurately. On the other hand the number of features

---

Fig. 3. Map and landmarks (lights) used to localize the guide robot in the Domus Museum. Lights (circles) with the same height (shown in parenthesis) are classified in three regions.

Fig. 4. (a) Omnidirectional lens (185° FOV). (b) Omnidirectional image acquired with our omnidirectional camera.
III. VISION SYSTEM

The vision system (Fig. 4) consists of a color digital camera MDCS2, equipped with an omnidirectional lens (fish-eye) FE185CO46HA-1 and an infrared baseband filter (IRP), model HOYA IR85. The camera is mounted on the robot with its optical axis perpendicular to the plane of the ground and pointing to the ceiling (Fig. 1).

The use of a high resolution omnidirectional lens, with a very wide field of vision (FOV), reduces the number of landmarks needed, as they can be seen from more different points of the environment.

The infrared filter baseband (IRP) attenuates the components of visible light and only lets the close range pass IR (Fig. 5).

A. Landmark detection

The process of detecting landmarks consists of 5 phases: acquisition, preprocess, segmentation, recognition and features extraction. The output of the system is an array of features for each landmark. The first four stages are implemented using the OpenCV library. The images are grayscale and size 640x480 (Fig. ).

In the preprocess phase the image (Fig. 6(a)) is transformed to facilitate the processing in the next stages. The techniques that have been used are binary thresholding (Fig. 6(b)) and morphological filtering (‘closure operator’) (Fig. 6(c)).

As segmentation techniques a Canny filter (Fig. 6(d)) and contour extraction (Fig. 6(e)). The next step is to extract the characteristics of each region:

- **Ratio**: number of pixels around the perimeter.
- **Centroid**: coordinates of the center of gravity.
- **Radius**: centroid distance to the center of the image.
- **Azimuth**: orientation of an object in the image with respect to axis X, φ on Fig. 8.

If a ceiling light points directly to the camera, then the acquired image will be saturated. In such cases, one big blob can be detected and the image have to be preprocessed again using a higher threshold. This situation is very frequent in the region labelled as B (Fig. 3) because lights can be very close to the camera.

IV. MAP PROJECTIONS BASED ON OMNIVISON CAMERA MODEL

In this section we present the model of our omnivision camera and how it can be used to project the objects of the environment to form an image.
A. Camera Model

The camera model describes how a 3D scene is transformed into a 2D image (Fig. 8). The standard model is the Pin-Hole, which projects the scene on a flat retina (Fig. 7), but it is limited to cameras with FOV $<< 180^\circ$.

The model that best fits our system was developed by Pajdla and Bakstein [2] based on a spherical retina (Fig. 7) where the image is formed on a curved surface. In our case the radial symmetric function is:

$$r = a \cdot \tan \frac{\theta}{b} + c \cdot \sin \frac{\theta}{d},$$

where $a$, $b$, $c$, and $d$ are the adjustment parameters of the model, $r$ is the distance in the image between the projection point of $B$ ($(u_B, v_B)$) and the image center ($(u_0, v_0)$), and $\theta$ is the elevation of $B$ with respect to the optical axis of camera (see Fig. 8).

This function makes it possible to calculate the coordinates of the image $(u,v)$ depending on the azimuth ($\phi$) and the elevation ($\theta$) (Fig. 8):

$$u = u_0 + r \cdot \cos \phi$$
$$v = \beta \cdot (v_0 + r \cdot \sin \phi)$$

(2)

where $\beta$ is the relationship between the width and height of a pixel.

B. A Beacon Projection

If we have the coordinates of landmark $i$ ($B^W_i$) and the coordinates of the camera $P$, both with respect to the environment reference system $(W)$, we can calculate the projection line of the landmark $B^P = B$ in Fig. 8) relative to the camera:

$$B^P_i = \text{Transf}_P(B^W_i) = R_P \cdot B^W_i - P$$

(3)

where $R_P$ is the rotation matrix of $P$ relative to $W$, i.e., the position and orientation of the camera in the environment.

To get the projection of a landmark $B^W_i$, $\text{Proj}(B^P_i)$, in image coordinates $(u_B, v_B$ in Fig. 8), we apply Eqs. 1 and 2:

$$\text{Proj}(B^P_i) = (u_B, v_B)$$

(4)

where Euclidean transformations were used to obtain the elevation ($\theta$) and the azimuth ($\phi$) angles of $B^P_i$ (Fig. 8).

C. Ceiling Map Projection

We have named ceiling map projection, $\text{Map}(P)$, to the set of theoretical positions that each mapped landmark in the environment ($B^W_i$) would have in the image $(u,v)$, i.e. the pixel where it would be if the robot was at the position $P$.

$$\text{Map}(P) = \{ \text{Proj}(B^P_i) \}$$

(5)

The algorithm is summarized in Alg. 1. A graphical example is shown in Fig. 9.

V. MONTE CARLO LOCALIZATION

Monte Carlo Localization (MCL) algorithm [15], [5] is a particle filter algorithm combined with probabilistic models of robot perception and motion. The current pose
Fig. 9. Example of Ceiling Map Projection, $Map(P)$: (a) Original image; (b) Beacons projected when camera is at position $P$ (gray enumeration) and landmarks detected on the image (black enumeration). Beacons in the shaded region are not considered because they are in the horizon.

Algorithm 1 Calculate $Map(P)$ for one position $P$.

for all Beacons $B_t^W$ on the map (environment reference system, $W$) do
  $B_t^P = Transf_p(B_t^W)$ applying Eq.3
  $Proj(B_t^P) = (u_{B_t^P}, v_{B_t^P})$ applying Eqs. 1 and 2
end for

Algorithm 2 MCL Algorithm

for all $m$ do
  $x_t^m = motion_model(u_t, x_{t-1}^m)$
  $w_t^m = measurement_model(z_t, x_t^m, Map)$
  $X_t = X_t + (z_t, x_t^m, Map)$
end for

$X_t = resample_model(X_t)$

return $X_t$

of the robot, also called belief, which model a probability density function over the space of all locations. The belief about the pose space is represented with a set of discrete points in the robot’s environment called particles $X_t = \{x_t^1, \ldots, x_t^N\}$. This type of algorithms proceeds recursively:

- A temporary particle set $X_t$ is computed from the last particle set $X_{t-1}$ and the last control action $u_t$ (motion model).
- A weight factor $w_t^m$ is assigned to each particle. It is computed based on the new sensor data at time $t$ (last observation $z_t$). $w_t^m$ is proportional to the probability that the robot is located in $x_t^m$ (measurement model).
- Then a new sample set $X_t$ is calculated (resample model).

A. Motion Model

The motion model computes the probability that the robot is in state $x_t$, if it was previously in state $x_{t-1}$ and control action $u_t$ is $\mathbb{P}(x_t|u_t, x_{t-1})$.

We have applied the odometry motion model [15], where the odometry measurements are used for calculating the robot’s motion over time. As we need to sample form $\mathbb{P}(x_t|u_t, x_{t-1})$, the algorithm accepts as input the previous state, $x_{t-1}$, an the control $u_t = (\overline{x}_{t-1}, \overline{x}_t)$ and returns a state $x_t$ according to $p(x_t|x_{t-1}, u_t)$ as output.

$$x_t = (x', y', \theta')$$

$$x' = x + \Delta x + N(\Delta x)$$

$$y' = y + \Delta y + N(\Delta y)$$

$$\theta' = \theta + \Delta \theta + N(\Delta \theta)$$

where $\Delta x, \Delta y, \Delta \theta$ are the differences between the two odometry values $(\overline{x}_{t-1}, \overline{x}_t)$ and $N(\overline{x}), N(\overline{y})$ and $N(\overline{\theta})$ are the random noise term. In this paper we modeled it with Gaussian zero-centered random variables with standard deviations $\sigma_x = 25$ cm, $\sigma_y = 25$ cm, $\sigma_\theta = 20^\circ$ respectively.

B. Measurement Model

The measurement model describes the probability of having a certain sensor measurement in given pose. Normally the measurements model is defined as a conditional probability distribution $p(z_t|x_t, Map)$ where $x_t$ is the robot pose, $z_t$ is the measurement (observation) at time $t$ and $Map$ is the map of the environment.

The model depends on the sensor and data used. In our case we have an omnivision camera and a map of landmarks (lights). So that we use a Feature Based Measurement Model.

In this model $z_t$ is the set of features extracted from the sensor measurement:

$$z_t = f(Map(P)) = \{f^1_t, \ldots, f^N_t\}$$

where $f^m_t = (p_{x^m}, p_{y^m})$ is the position on the image (in pixels) for each identified feature (associated landmark). The number of identified features can be different at each image (Fig. 9 and 11).
Algorithm 3 Calculate $N_P$ and $\varepsilon_P$

$$Map(P) \text{(Alg. 1)}$$

for all $B^M_i$ in the map do

- $f^t_n \leftrightarrow Proj(B^P_i)$ (Fig. 11)
- $\varepsilon(B^P_i) = ||Proj(B^P_i) - f^t_n||$
- if $\varepsilon(B^P_i) < \text{THRESHOLD}$ then
  - $\varepsilon_P = \varepsilon_P + \varepsilon(B^P_i)$
  - $N_P = N_P + 1$ (Number of associated landmarks)
- else
  - $\varepsilon_P = \varepsilon_P + \text{THRESHOLD}$
- end if

end for

To calculate $w^m_t$ we need to know the expected landmarks ($Map(x^m_t)$) for each particle $x^m_t$ and their likelihood with the detected landmarks ($z^t$). In order to do that we define a Merit function $M(x^m_t)$ defined as (to let a clarity reading of the next equations $x^m_t$ is denoted as $P$):

$$M(P) = \frac{1}{N_P} * \varepsilon_P$$

where $N_P$ is the number of identified landmarks or ‘matched’ landmarks and $\varepsilon_P$ is the accumulated error (distance in pixels between detected landmarks and projected landmarks on position $P$, $Map(P)$ in Fig. 9). Both are calculated using Alg. 3. A graphical example of the matching process is illustrated in Fig. 11.

The best data association for each particle is the one that has the largest number of identified landmarks $N_P$ and the smallest error $\varepsilon_P$ in matching process.

VI. EXPERIMENTAL VALIDATION

The experimental validation of our system has been carried out in a exposition hall at the Domus Museum located in A Coruña (Spain). It has about $24 \times 7m^2$. The experiments were performed as off-line validation tests on different sequences of images acquired in the museum. All images were labeled with the corresponding position obtained from a laser sensor. One image per second was acquired.

The results of one of this experiments can be seen on Fig. 12. The two trajectories executed by the robot are displayed on Fig. 12(a); only 2D positions ($X,Y$) are shown. The distance travelled in this experiment is more than 48 meters and 120 images were acquired and processed. Two consecutive images were separated at 0.40 meters on average but angular displacements can be very large. As reference, at the end of the first trajectory the robot uses only 3 steps (images from 60 to 63) to turn 180 degrees.

We use as reference the position calculated by the laser...
sensor when there is no people in the environment (that is an unusual situation in the museum). The maximum position error is 1.07 m but the average position error has been 0.41 m (Fig 12(b)). The maximum orientation error is 20 degrees, but only 3 degrees on average (Fig 12(c)). This angular precision is one of the main advantages of using an omnidirectional camera for localization. The irregular floor of the museum affects negatively to the position error. On the other hand, the positions of the mapped landmarks on the museum were very difficult to obtain and the errors are not negligible. Nevertheless, our system calculates correctly the position of a mobile robot on a very complex environment and over long trajectories. Its precision is enough for navigation tasks.

The time for running the algorithm can be seen in Fig. 12(d). We have used a laptop Intel Pentium 4 CPU 3.06GHz. The time for processing each image (vision time) is almost constant, except when an image is saturated. In such cases, a second landmark detection process is launched with a higher threshold (see Sect. III-A). On the other hand, localization time depends on the number of particles checked in the Monte Carlo filter. In this experiments 200 particles were checked in each step. The time required for processing each image is only 40 ms on average, so our algorithm can be executed in real time.

The proposed localization system is very robust. For example, only 10 landmarks of the 29 mapped lights are identified on each image on average. In several steps, only 4 or 5 landmarks can be used for localization. The reasons are that many elements in the environment can occlude the lights to the camera and that lights are oriented (i.e. can not be detected on any position, see Fig. 2). In this sense, using an omnidirectional camera minimizes all these problems.

One final problem arises when there are positions in the environment from which the views of the ceiling are similar. This symmetries can mislead the system. Nevertheless, as the robot moves more information is obtained from the environment and the ambiguity can be reduced, or even eliminated, applying the Monte Carlo filter.

VII. CONCLUSIONS AND FUTURE WORK

The solution adopted in this paper to locate a guide robot on a museum is based on particle filters and a map of lights (landmarks) in the environment. The main difficulties for the localization of the robot are that the Domus museum is highly populated, the irregularity of the ground floor (swinging movement of the camera, increased odometry errors), and the height of the ceiling (measurement errors are proportional to that height).

The results of the experiments confirm the accuracy and robustness of our omnivision localization system. The position and angular errors are 0.4 m and 3 degrees on average, respectively. In spite of the irregular floor of the museum and the imprecision on the mapped landmarks.

The proposed algorithm can be executed very efficiently. On one hand, using a IR filter simplifies the process of landmark detection on the images. On the other hand, the Monte Carlo filter reduces the computations needed to integrate observations and control information over time.

We want to highlight that our system was designed to cope with occlusions. The very wide visual field guarantees that enough landmarks will be viewed to estimate a good localization. In our experiments only 31 lights are considered and the robot can be located in a space of 168 m².

Currently our studies are focussed on the implementation of an omnivison SLAM.

VIII. ACKNOWLEDGMENTS

Authors wish to especially thank the collaboration of Tomas Pajdla and Hynek Bakstein in the calibration of the camera. This work has been financed by the Galician regional government and the Spanish Government through grants PGIDIT-TIC200611PR, TIN2005-03844.

REFERENCES

Fig. 12. Experiments on the Domus Museum: (a) Omnivision localization (gray) plotted on the grid map created with laser data (real trajectory is marked with dark circles). Position (b) and orientation error (c) between laser pose estimation and the omnivision localization. (d) Time for processing each image.
Gaussian Progress Regression for Facial Behaviour Mapping

Peter Jaeckel and Neill Campbell* and Chris Melhuish

Abstract—This paper presents a mapping of observed motion data of facial behaviour footage to a robotic face using Gaussian process regression. The model is built from a training set that consists of examples of input expressions in form of 2-dimensional feature point locations and their appropriate robot pose in terms of servo values. Unlike the linear mapping suggested in previous work, this approach does not suffer from noise in the training and data and the model can handle ‘multi-emotional’ information. It is robust to inherent and hard-to-avoid errors in the training set. The mapping still performs well in cases of input corruption. The probabilistic characteristics of the Gaussian process regression also gives a measure of quality or confidence about an output.

I. INTRODUCTION

The work presented here is part of a research project that is inspired by the idea of creating artificial empathy by means of a robot head. Mirroring back a narrator’s emotional state by generating appropriate dynamic facial expressions shall create the feeling of being listened to sympathetically. One essential requirement is the perception and recognition of narrators’ emotional states from their facial expressions, gaze and head movements. As autonomous robotic systems advance, they will be designed for interacting with humans in order to exchange and retrieve required information for accomplishing their tasks. This requires a vast number of human-like skills such as the ability of interpreting and emulating human facial behaviour. An affective or empathetic agent must be capable of recognising and mirroring or mimicking emotional content [1].

This paper focuses on mapping facial motion data to movements in a robotic face which emulates the majority of the human facial muscles. An Active Appearance Model (AAM) [2] based facial feature tracker supplies information on facial movement in form of 2-dimensional point data. This data is then fed into the proposed model and used to predict appropriate servo positions. This results in the robot mimicking a persons face tracked in a video or camera stream. The model is formed according to a training set containing input - output examples of input faces and robot pose, selected by an animator.

Antecedent work [3] reports advances made, building a control system which aids synthesis of artificial, dynamic, facial behaviour in a humanoid robot head. It has been shown that a small number of 2D feature points, tracked in video footage, are sufficient to generate control commands that result in appropriate facial expressions in a robot head. Firstly, since the acquisition of information concerning human subjects’ facial expressions is crucial, motion capture allows driving on-screen agents in a very realistic manner. We have investigated a linear mapping of facial behaviour and facial movements from video footage into a humanoid robotic face. The approach is limited to a small number of training samples involving jaw, head, lip and eye brow movements. The linear model was only capable of mapping facial behaviour of one particular emotion. Additionally the model was able to map non-emotional facial displays, such as eye blinks and eye brow movements. The results were impaired by limitations of the robotic hardware. The speed and also range of some servos was very restricted. Also non-linearities causing jerky movements make it hard to achieve facial dynamics that are convincing.

Most projects in artificial intelligence and social robotics rely on static and fixed routines of rather exaggerated facial expressions [4], [5]. However, state of the art recognition of exaggerated, stereotypical static facial expressions will not be sufficient. Literature in human psychology indicates that the behaviour of a narrator and listener is dominantly calm and stereotypical facial displays occur very infrequently [6]. As Autonomous Robotic Systems advance, they will also be designed for interacting with humans in order to exchange and retrieve essential information required for accomplishing their tasks. This requires a vast number of human-like skills such as the ability of interpreting and emulating human facial behaviour. An affective or empathetic agent must be capable of recognising and mirroring or mimicking emotional content [1].

II. BACKGROUND

Robot behaviour and physical appearance need to be well matched in order to avoid discomfort during interaction [7]. Researchers have started paying attention to the importance of eye and head movements and their temporal interconnection with facial displays [8]. Also the temporal differences of facial motions during posed and spontaneous smiles have been investigated and proven significant [9], [10]. Synthesising and modelling facial behaviour has shown that sequences of various emotional states do overlap. This again suggests that fixed ‘snapshots’ do not necessarily reveal the true emotional state [11].

It is highly desirable that artificial companions in entertainment and education are trustworthy, helpful, reliable and engaging [12]. There are a number of reasons that recommend and justify the use of physically present robots rather than computer generated on-screen agents. Kidd and
Breazeal have investigated the effects of robots on user perception [12]. Their experiments involved humans interacting with a real human, an animated character and a physically present robot. Evaluation of the interaction in terms of engagement and participants’ perception yields that the robot was more engaging and perceived more enjoyable, credible and more informative than the animated character. Their findings also verified experiments conducted by Reeves et al. [13]. In addition, physical presence makes subjects remember the interaction in more detail [14]. Hence, characteristics, such as life-likeness and physical presence are essential for high quality and effective human-machine interaction. Furthermore, researchers argue that tools used for the simulation and study of human behaviour must be as natural as possible in appearance and behaviour [15].

There have been attempts to map and synthesise human behaviour and movements: The work reported in [16] is based on recognition of stereotypical facial expressions of emotion. Input images are projected into a seven-dimensional emotion space, where each emotion is one of the six basic ones plus neutral. In a second step, the recognised emotional state (location in 7D emotion space) is fed into a character animation engine which produces facial expressions of emotion based on the 7D emotion code. Hence, in terms of mirroring capabilities this system is limited to stereotypical and exaggerated facial expressions of emotion.

In [17], Shon, et. al propose a probabilistic mapping of human subjects pose to a robot’s pose. They used a shared Gaussian Process Latent Variable Model (GP-LVM) which is an extension Lawrence’s work in [18]. GP-LVMs allow low-dimensional descriptions in a latent space of hi-dimensional observations. An explicit mapping is only available from latent to observation space - this allows non-smooth mappings. To find the most likely latent location for a given observation, a search is necessary. In Shon’s work two GP-LVM’s are used by joining their latent space. However, they did not make use of the non-smooth mapping capabilities. Instead they have used inverse (smooth) mappings from the observation space to the latent space.

In [3], the application of a linear modelling and mapping approach, Partial Least Squares, has been proposed for the mapping of emotional, dynamic facial behaviour from 2D-feature point location data to position commands that drive the artificial muscles of a robotic face. The results show that it is a simple and computationally inexpensive approach for creating a facial mapping using a limited and small number of training samples. The linear model does not allow modelling of more than one emotional state. Also an increased number of training samples tends to dramatically impair mapping quality.

The previously proposed model is also very sensitive to errors in the training data. Two or more very similar training samples, which may contain random deviation due to tracking errors, can impair the overall mapping. Hence the mapping is very sensitive to wrongly set servos. This tends to happen for those servo channels that are responsible for subtle features. This would not be a problem if correlations between individual features were not modelled. However, the model does rely on correlations between features that are coupled in the training set. Hence wrongly set subtle features in the training data are correlated with other less subtle features. Hence its likely for small errors to affect the overall robot pose and impair quality of the mapping. Another problem consist of extrapolation beyond the range of training data. The mapping of previously unseen, or ambiguous facial displays may result in unacceptable and unrealistic robot poses. Extension of the model so that it consists of several groups of fundamentally different facial displays (i.e. groups of different FEOE (facial expression of emotion)), is one way to make sure that novel and observation data lies within training range. In cases of new observations a measure of confidence about the output could prove useful and it should further avoid the above mentioned problems of unrealistic, extrapolated facial behaviour.

III. THE PREVIOUS LINEAR REGRESSION MODEL

In this section we review the previously applied Partial Least Squares (PLS) model. The aim is also to familiarise the reader with the basic concept of the previous employed technique as it shall be used as a benchmark for the model proposed in this paper.

The model input consisted of 25 tracked feature points coordinates X (landmarks) in 2D. The feature point locations were obtained from fitting person specific Active Appearance Models to input video frames (320 × 240 pixels). AAMs are parameterised models which contain information on shape and texture. Fitting an AAM into an image is essentially finding the model parameters which minimise the error between input image and the instance of the active appearance model [2]. The search for an optimum fit is initialised with the model mean and the size of the face and it’s coordinates supplied by a face detector [19]. Initialisation is repeated each time the error threshold between input image and AAM instance is exceeded. Figure 1 shows the AAM shape (left) which is then fitted to frames which were supplied by video, yielding 25 landmarks (right). Subsequently, the landmarks are centred and normalised.
The model output is given in form of a vector of length equal to the number of degrees of freedom or number of servo motors in the robot head. It provides information about the predicted robot pose in terms of servo positions.

Figure 2 gives an overview of the a) training and b) prediction process: A video sequence or camera stream supplies training footage (left) and frames which the robot pose is predicted from (right). The training frames, along with the appropriate robot pose make up the training set, from which a model is created. Training in PLS aims to form a latent space $T^1$ which dimensions best describe the covariance between input $X$ and output $Y$ [20]. Usually the first few latent dimensions describe most of the data variation between $X$ and $Y$ and hence the remaining ones can be omitted. Figure 3 illustrates the prediction of output variables $Y$ from model input $X$. The input is compressed and projected onto a lower dimensional space given by latent vector $T$ (PLS components). The next step is to project the data into the output space.

![Figure 3. Principle of PLS Algorithms [20]](image)

**IV. GAUSSIAN PROCESS REGRESSION**

Gaussian processes [21] can be seen as a generalisation of a Gaussian distribution. The mean and variance become mean function and covariance function, respectively. The mean function is formed according to modelled observations and determines the mean output $y$ for a given new observation $x$. The covariance function provides a measure of confidence for a certain input-output mapping. The output confidence interval is small for new observations which are very similar to the modelled data. On the contrary, unlikely observations yield a large output interval. Gaussian processes ‘assume’ errors in the training data. Hence, multiple or similar examples of a particular facial expression are preferred. This is unlike the linear approach, where similar training samples impair the mapping due to noise and best to be avoided. Gaussian process regression pursues two goals: The first one is to find a process which describes some observed data $X$ under the assumption that the data is corrupted by noise. Secondly, the model is used to infer likely model outputs from new, previously unseen observations.

Training of a such a model is essentially maximising the likelihood of the model, generating the training data, by optimising the model parameters. In other words, one wants to find the model that is most likely to generate the given training data. The likelihood function is given by:

$$
\mathcal{L}(\theta) = f_0(x_1, \ldots, x_n \mid \theta)
$$

where $x_1, \ldots, x_n$ is the training data and $\theta$ the model parameters to be optimised so that:

$$
\hat{\theta} = \arg\max_\theta \mathcal{L}(\theta)
$$

Further the likelihood in terms of products of univariate probability densities is given by:

$$
\mathcal{L}(\theta) = \prod_{i=1}^{n} f_0(x_i \mid \theta)
$$

This expression can be written as a sum (monotone transformations, logarithm) and gives the log likelihood.

$$
\mathcal{L}^*(\theta) = \sum_{i=1}^{n} \log f_0(x_i \mid \theta)
$$

We assume the data to be normal or gaussian distributed. The probability density function for a one-dimensional, continuous normal distribution $\mathcal{N}(\mu, \sigma^2)$ is given by:

$$
f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi} \sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)
$$

where $\mu$ and $\sigma^2$ are mean standard deviation, respectively. Both are the model parameters that determine the gaussian distribution: A general notation for $n$-dimensional data is the multivariate normal distribution:

$$
f_X(x_1, \ldots, x_N) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x-\mu)^\top \Sigma^{-1} (x-\mu)\right)
$$

where $\Sigma$ and $\Sigma^{-1}$ are the determinant and inverse of matrix $\Sigma$, respectively.

For centred training data ($\mu = 0$) the equation to be maximised becomes:

$$
\mathcal{L} = \frac{1}{2} \log |\Sigma| - \frac{1}{2} X^\top \Sigma^{-1} X - \frac{n}{2} \log 2\pi
$$

The covariance matrix $\Sigma$ can be replaced by one or a combination of kernels $K$ to allow for non-linearities. This substitution called the ‘Kernel Trick’ [22]. It essentially...
projects observations from $X$ to high-dimensional Hilbert space $H$ using a mapping $\Phi$ (see Equation 8).

The observations $x$ are mapped to the where they are represented as a vector $x$ (Equation 9). The vectorial representation $x$ in inner product space ($H$) allows using linear algebra and the application of linear classification to non-linear problems.

\[
\Phi : X \rightarrow H
\]

\[
x \rightarrow x := \Phi(x)
\]  

(8)

(9)

For the purpose of this work a compound of a radial basis function (rbf) (Equation 10) and a noise term has been chosen. Radial basis functions tend to predict the mean for unlikely observations which is beneficial in case of input corruption. The application of a noise term for describing the modelled data aims to avoid over-fitting and to compensate for errors in the training data.

\[
K(x,x') = (\Phi(x) \cdot \Phi(x')) = \exp(-\beta||x - x'||^2)
\]  

(10)

The model parameters are given by the kernel parameters which are concatenated in vector $\theta$.

A scaled conjugate gradient (SCG) algorithm is used to find model parameters $\theta$ which maximises $\mathcal{L}$. In other words, the kernel parameters are jointly optimised so that the likelihood of the model assumes a maximum value. Since the SCG algorithm is designed finds local minimas, the negative log likelihood is used as the objective for the optimisation.

Similar to the previous linear approach explained in Section III, the input data for training and mapping consists of 2D-point data (shape) obtained from fitting a person specific AAM (Figure 1(left) to a face (right) shown in an input frame. Fitting the AAM into input images gives the coordinates of 25 landmarks $X$ in 2D (see points on the face in the right portion of Figure 1). The next step is to ‘unfold’ the data by simply concatenating the $x$ and $y$ feature positions to give an input vector of length 50. The output of the model is a vector of length determined by the number of used servo channels. The motors take angular positions according input values in the range of 1 to 255. The head mechanism requires limitations of servo ranges to keep expressions realistic and avoid damage of the skin and motors. Hence each servo channel’s allowed range is scaled to take values from 0 - 100. The training input and output data is centered and scaled to unit standard deviation and unused servo channels (such with zero variance) are omitted. New observations and the resulting model outputs have to be pre- and post-processed accordingly.

\[2\text{Instead of holding the } x \text{ and } y \text{ feature positions in a two dimensional matrix they are all lined up in one vector.}\]

A. The Robotic Face

For the experiments the robot head called ‘Jules’ (supplied by David Hanson\(^3\)) shown in Figure 4, has been employed. It has 34 Hanson servo motors which emulate the majority of the muscle groups of the human face and neck. The actuators pull and/or push control points attached underneath the skin. It can move upper and lower eye lids and the eyes move vertically and horizontally. There are three servos manipulating each eye brow. Jules can nod, turn and tilt its head and is able to perform frowns and has a rich repertoire of sneers and smiles. The servo motors are controlled by servo controllers that receive desired positions for each servo via an RS232 interface at up to 30fps (frames per second).

\[\text{Fig. 4. The employed robot head ‘Jules’ has 34 servo motors emulating the majority of the human facial muscles around face and neck.}\]

B. Training Process

To initialise the training process a neutral face is captured and the robot’s neutral pose assigned to it. The next few training examples can involve arbitrary input poses but it must be made sure that the appropriate robot pose is very similar to the input face. The next stage is an iterative process which involves tweaking of existing training samples and adding new ones. Rather than including novel expressions to improve the mapping of certain expression types, it is important to revise existing training data first because it may contain badly mapped expressions. The training process is complete, once the mapping produces satisfying output poses for the desired range of expressions.

The pseudo code below summarises the training process:

- capture neutral input expression and assign appropriate robot pose
- test a desired expression
- capture another input expression in combination with a head position that doesn’t work well yet.
- adjust robot pose of existing training pair
- capture similar input frame
- repeat for a small number of arbitrary input expression
do until overall mapping satisfying {
  - test a desired expression
  IF (mapping performs well){
    - capture another input expression in combination with a head position that doesn’t work well yet.
  }
  ELSE{
    - adjust robot pose of existing training pair
  }
  IF (mapping performs well){
    - capture another input expression in combination with a head position that doesn’t work well yet.
  }
}

\[\text{web site: www.hansonrobotics.com}\]
C. Robot Pose Prediction

The prediction of the robot pose is performed alongside with facial feature tracking using AAM-fitting as already described in Section III. The mapping can predict robot pose from diverse footage provided by a live camera stream, rather than just a limited range of facial expressions supplied by a video, previously reported.

The mapping algorithm has been implemented using Neill Lawrence’s GP-LVM Matlab toolbox. It has subsequently compiled using Matlab’s mcc compiler to run alongside a facial feature tracker employing the AAM-API\(^4\). The mapping updates the desired servo position every 80ms (12.5fps) running on an Intel Pentium D925 at 3GHz.

V. Results

A. Correlation between Servo Channels and Feature Points

To assess the mapping quality it has been investigated how well the model describes the covariation between input point locations and servo outputs given by the training data. The covariance, inherent in training set and model, has been compared by summing up the covariance for each input and its changes in output. This appears a good measure of how well the model actually explains the training set. Firstly a \(i \times j\) matrix \(C\) has been formed that contains the covariance between each input variable \(X_i\) and each output \(y_j\). The sum over all rows and columns give the overall variance. The model describes about 81.33\% of the training set’s covariance. Table I shows that a considerable amount (about one third) of modelled variance is described by the noise kernel. The radial basis function kernel accounts for about one third of the training samples are omitted starting from sample number one, then successively from number two, three and so forth. Doing this for 64 training samples, results in about 55 ‘missing data’ models. The next step is to determine the average performance of both, ‘missing’ and ‘full data’ in respect to the ground truth. To do this, the outputs have been compared to the ground truth for each training sample and for each ‘missing’ and ‘full’ model. The deviation of model output from the ground truth is shown by the average root-mean squared error \(RMSE\) over all outputs. All output errors and standard deviations of errors are in percent and scaled to the maximum range of output values obtained from the full model. Figure 5 shows the errors and the standard deviations of servo channels 1 – 17 out of 33 (sufficient for illustration purposes).

The average error \(RMSE\) over all \(c\) servo channel of the output is given by equation 11. Ground truth of channel \(k\) is given by \(\hat{y}_{i,k}\). Similarly, the test output is denoted by \(y_{i,j,k}\). The test output results from testing the \(p^\text{th}\) training sample on the \(j\)th model. Unlike the full model, the missing model is trained from different training data for each \(j\).

\[
RMSE = \frac{1}{\alpha \beta} \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{c} (\hat{y}_{i,k} - y_{i,j,k})^2}
\]  

Factor \(\alpha\) is responsible for scaling the sum to maximum range (in percent) and \(\beta\) is the total number of summands.

![Fig. 5. Root Mean Squared Errors for servo channel 1 – 17 (out of 33) over one full model (blue/left portion of each bar) and the average RMSE (red, right portion of each bar) over 55 models obtained from omitting training samples in turn. Additionally, the graph shows the standard deviations for each channels RMSE. Despite omitting 10% of the training data, the output errors and their range hardly changes which indicates a high degree of robustness. Index ‘L’ and ‘R’ denote left and right, respectively. ‘Frown’ - pulls lip corner down, ‘OO’ - orbicular oris(responsible for o-shaped mouth), ‘EE’ - lip stretcher](image)

The average error and standard deviation of the error between ‘full’ model has been calculated for each of the \(m = 55\) ‘missing’ models and all \(n = 64\) training samples. The results in table II show that the ‘full’ model performs slightly better as one would expect. However, these results show that the model still performs well, even with 10% training data missing.

<table>
<thead>
<tr>
<th>Servo Channel</th>
<th>Error and std dev. in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>1.1</td>
</tr>
<tr>
<td>5</td>
<td>1.3</td>
</tr>
<tr>
<td>6</td>
<td>1.5</td>
</tr>
<tr>
<td>7</td>
<td>1.7</td>
</tr>
<tr>
<td>8</td>
<td>1.9</td>
</tr>
<tr>
<td>9</td>
<td>2.1</td>
</tr>
<tr>
<td>10</td>
<td>2.3</td>
</tr>
<tr>
<td>11</td>
<td>2.5</td>
</tr>
<tr>
<td>12</td>
<td>2.7</td>
</tr>
<tr>
<td>13</td>
<td>2.9</td>
</tr>
<tr>
<td>14</td>
<td>3.1</td>
</tr>
<tr>
<td>15</td>
<td>3.3</td>
</tr>
<tr>
<td>16</td>
<td>3.5</td>
</tr>
<tr>
<td>17</td>
<td>3.7</td>
</tr>
</tbody>
</table>

TABLE I

MODELLED VARIANCE OF RADIAL BASIS FUNCTION AND NOISE TERM:
THE TRAINING SET CONTAINS A CONSIDERABLE AMOUNT OF NOISE.

\(\beta\)

4Active Appearance Model API available at http://www2.imm.dtu.dk/ aam/
C. Robustness to Input Corruption

The model input has been deliberately corrupted and as a result the output error and range are expected to change. How much the corruption affects the output error and its range tells us something about the robustness of the approach. The high range of distribution of the errors demanded the additional presentation of results in terms of standard deviation of the errors. Future work aims to give further insight into the model’s output characteristics.

Table III shows the percentage of changes in output error and its standard deviation, as a result of deliberate partial input corruption. The input has been corrupted by setting \( n_z \) randomly picked points of the centered and scaled \(^5\) shape (25 points) to either -1, 0 or 1. The input error for individual corrupted feature points that make up the input is relatively low and amounts approximately half the percentage of the partial corruption of the input (12.27%). Increasing the number from \( n_z = 4 \) to \( n_z = 9 \) corrupted input values (set to \(-1(-12.27\%)\)) results in a slight increase from 3.92 % to 5.25 %. Setting input values to the mean (zero) cause considerably small, average changes of around 1.71 - 2.15%.

### TABLE III

<table>
<thead>
<tr>
<th>( n_z )</th>
<th>( \Delta \text{RMSE} )</th>
<th>( \Delta \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3.92%</td>
<td>3.14%</td>
</tr>
<tr>
<td>9</td>
<td>5.25%</td>
<td>2.69%</td>
</tr>
</tbody>
</table>

\(^5\)scaled so that standard deviation equals 1 for each channel

The covariance between model inputs and outputs is another useful characteristic to investigate. This is another measure of model quality. To do this the following steps where necessary: Firstly, we have tested the training data and obtained the corresponding model outputs. The covariance obtained for all inputs and outputs are then summed up. The next step was to determine sum of the covariance in the training data. The final step was to calculate the difference between both sums of covariance. Comparing the covariance sum of all input points shows that the model ‘holds’ about 75% of the training data’s variation.

The errors of both models are in similar range. Equally the standard deviation of the error remains unchanged. Omitting 10 per cent of training data, does affect the quantitative quality of the mapping.

D. Comparison with Linear PLS Approach

A linear PLS model has been trained using the full training set. To test the performance, the root mean squared error \( \text{RMSE} \) between ground truth (training samples) and model output is determined. Both, the linear PLS model and the non-linear Gaussian Process are tested using the training set. This allows us to evaluate the modelling capabilities of both model types.

As already described in section III, the PLS model projects the input data \( X \) into a latent space \( T \) of dimensionality \( d \), from which, subsequently, the location in the output space \( Y \) is determined. \( T \) is formed to best describe covariation between \( X \) and \( Y \). The dimensionality \( d \) of \( T \) can be specified before the training process, depending on the required or desired amount of modelled variation in \( X \) and \( Y \). For the given training data, the output range of the PLS model is strongly dependent on dimensionality \( d \). It was necessary to empirical determine the maximum number of used latent dimensions \( d \), by observing the output range of test sequence.

The number of dimensions \( d \) was then chosen to so that outputs of both models lie in a similar range. Reducing the dimensionality of \( T \) inevitably results in discarding a lot of modelled variation the linear model. However, reduction of \( d \) was essential for obtaining a usable model, otherwise the model training process gave unrealistic solution due to matrix singularity problems. A maximum latent dimensionality of \( d = 51 \) caused output values up to \( 10^8 \) times higher than the expected range \((-1 < Y < +1)\) which clearly shows that the model is not feasible. Comparable outputs where given by models with latent space of \( d = 8 \). Table IV shows the modelled data variation in \( X \) and \( Y \). The first 8 latent dimensions describe the input space \( X \) with 99.82% almost entirely. However, it omits about 60% of the output space \( Y \) (41.49%). Again, omitting the information was crucial for obtaining a usable linear model.

For the purpose of illustration, Figure 6 shows only the first half of all \( n_x = 30 \) utilised servo channels.

The PLS model test outputs yield an \( \text{RMSE} \) which is consistently and significantly higher for all \( n_x \) servo channels. Also, the error bars showing the standard deviation of the errors and hence the range of errors is much greater too. In previous work, where small amounts of training data were used, the PLS was found very suitable and generated facial motion had a ‘direct feel’ due to the models linear nature. However, for a larger amount of training data, the model
outputs would often saturate which dramatically changes the dynamics of robot movement. GP Regression provides a much better way of mapping facial motion as its outputs hardly saturate. Further more in case of unseen observation, it predicts the mean, rather than unrealistic output linear extrapolation.

The Rooted Mean Squared Error (RMSE) between ground truth \( \hat{y} \) and test output \( y \) is computed as follows:

$$
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}
$$

(12)

The standard deviation for each channel shown in the illustrations is computed over the error between ground truth and test outputs over all training samples and shown in percent in respect to the maximum range of values in the training set.

Figure 7 shows four pairs of the robot’s expression (left) and the corresponding input expression. For illustration purpose only a small subset of the robots expression repertoire and possible mapping is presented. Top left portion shows the activation of smile muscles with jaw closed. The pair at the bottom left shows mapping of jaw activation. At the top right, the lip stretcher is activated and at the bottom right shows ‘head up’.

VI. SUMMARY AND CONCLUSION

Gaussian Process Regression has been proposed to overcome the problem of limitations of being restricted to a model of single emotional state and a limited number of training samples and over-fitting as it occurred with the linear approach. A multi-emotional probabilistic model has been shown to be applicable for mapping sequential face information, extracted from video, to a robotic face. It has been shown that the GP-regression, performs better than the linear model as it handles multiple examples of one input-output sample with a great level of robustness to inherent noise. An experiment has been conducted to determine the robustness to corrupted input data as it is likely to happen during live tracking of facial features when parts of the face are occluded. This has very small effect on the output data. The RBF-kernel allows the mapping to performs great even in case of input corruptions.
Training and inference is computational inexpensive and can be performed in real time. Live tracking and mapping was implemented to demonstrate the ability to mimic a set of facial movements (inner and outer eye-brows, sneers, a number of smiles, jaw- and head movements). The work presented here is part of ongoing research to explore techniques for facial behaviour mapping in robotics. It further requires the facial behaviour produced by the robot to be assessed by psychological experiments. The application of a non-linear approach aids compensation for rotary movement in the robotic hardware. However, investigations regarding those compensating capabilities are subject to future research. The investigation of other kernel types, such as multi layer perceptions, seems in place. A soft saturation could be employed to avoid dramatic changes in dynamic characteristics due to clipped output values exceeding the maximum range.

We propose to tackle problems of ambiguities, which cause the same input point configurations resulting in a number of different robot poses. The use of dynamics shall aid to disambiguate input expressions and dominate the model outputs in case of unlikely inputs or the absence of stimuli.

VII. ACKNOWLEDGEMENTS

The authors thank Neil Lawrence and Carl Ek of the University of Manchester for their support and for providing the GP-LVM toolbox for Matlab.

REFERENCES


Robot learning from environment interaction and observation of human-behaviour

Pablo Quintía, Roberto Iglesias, Carlos V. Regueiro, Eva Cernadas and Miguel Rodríguez

Abstract—The aim of this paper is to develop a system capable of comparing robot and human behaviour and thus provide a feedback—reinforcement— that is used by the robot to learn. Our system will learn how to move in a building by observing the areas where humans walk on. The observation will be made with a group of agents placed in order to cover the entire environment. Then a robot will be trained using a combination of reinforcement learning, genetic algorithms and a dynamic representation of the environment. The reinforcement used is automatically generated from the prior observation of human-behaviour and it will guide the learning process to achieve a control policy that makes the robot move through the same areas as people. The experimental results we obtained, although preliminary, show the usefulness of our proposal.

I. INTRODUCTION

The development of robotic systems that can sense and interact with humans in useful ways is a big challenge in the robotics field. Robots are meant to become part of everyday life, as our appliances, assistants at home, helpers and elder-care companions, co-workers at the workplace, etc. Nevertheless, to get robots operating outside research centres and beyond the supervision of engineers or robot experts, it is necessary to face different technological challenges, amongst them, advanced autonomy: robots must be able to act in very different/dynamic environments, carrying out a wide set of tasks and operational modes.

In this context, all those strategies that allow robot learning from its interaction with the environment acquire a special relevance. As part of these techniques, we want to highlight reinforcement learning, since all the robot needs to reach an interesting behaviour is a feedback – reinforcement function– which tells the robot whether its performance is acceptable or not. Nevertheless, the formulation of this feedback is still an important drawback: we can not ask elderly people to provide a set of rules which specify when the robot is doing right or wrong. Therefore, the learning process can not yet be considered autonomous or expert-independent.

The investigation of how to overcome this limitation is the aim of this article, and part of the nowadays research at the University of Santiago de Compostela, Spain. We want robots that are able to get their extrinsic motivation autonomously. The reinforcement the robot receives must be automatically generated and it should not require expert or clever formulation.

We propose the observation of human-behaviour to reach our aim. The robot will get a feedback that says whether or not the robot’s motion is similar to what the “majority” of people moving in the same environment do. This feedback will be automatically generated and will not depend on a particular task or behaviour.

II. LEARNING FROM OBSERVATION OF HUMAN-BEHAVIOUR

We propose a system that works on two stages:

- **Automatic generation of reinforcement**: This stage is carried out by several agents that observe the environment (a camera or a laser connected to a computer), figure 1. Each agent will register the paths of the humans in the covering area. This information will be used to generate a reinforcement that will tell the robot if its current position has been traversed by people moving in the same area or not.

- **Robot training**: Once there is a reinforcement created after human-behaviour observation, the robot can use it to learn. In this article, robot’s behaviour is learnt in simulation. Nevertheless, in future work the robot will receive external feedback while it is moving in the real world.

It’s important to notice that the strategy we propose is very different from learning by imitation [11], [12]. The
automatically generated reinforcement will only provide information about the “margins” of what is acceptable or not – if there aren’t people colliding with the walls we do not expect robots doing so either. People move normally, they do not need to consider robot’s restrictions, since the robot will not try to replicate human actions. What we propose is not a supervised strategy, the only information the robot will receive is a feedback which says whether the robot is doing something that could be considered as “normal” or “usual” in the environment. The robot must come up with a behaviour which satisfies these restrictions. There is a very important benefit derived from the strategy we propose, any person could get our software and a set of cameras, install them at home or at work, and get a robot which learns from a feedback/reinforcement that the user does not have to program at all.

III. EXPERIMENTAL SETUP

We applied our approach in a corridor of the Department of Electronics and Computer Science, at the University of Santiago de Compostela, Spain, figure 2. This corridor is about 16 metres long and 2 metres wide. The building does not have any video surveillance system, so we placed two cameras at the locations shown in Figure 3.

The robot we used for our tests is a Pioneer 2AT equipped with a SICK laser scanner and 16 ultrasound sensors.

IV. STAGE I: AUTOMATIC GENERATION OF REINFORCEMENT

This is the first stage of the system we propose. By means of artificial vision and neural networks we will create a system that automatically provides reinforcement for the robot that tells whether the robot is running across an area that has been previously traversed by people or not. This will minimize the need of a human expert configuring the system.

The reinforcement generation has two main modules (Figure 4): the first one is dedicated to the gathering of position data from a video stream (sections (a), (b) and (c) in Figure 4); the other one is dedicated to the creation of the reinforcement signal from the information gathered in the previous module (Figure 4(d)). In fact, as we can see in figure 4, this second module segments the area under observation (corridor shown in figures 2 and 3), to differ regions where people usually moves from those other areas where people don’t go. These two modules will run in parallel, updating the reinforcement while getting new data from the cameras.

We want to have the minimum interaction of an expert in the setting up of the system. Hence our desire is to develop a fully automatic generator of the reinforcement. As the behaviour chosen was to run the robot along the same paths followed by people moving inside a building, we will observe how people move, and then we will make the robot run inside these paths by generating a negative reinforcement whenever it moves outside the walked areas. For the creation of the reinforcement we will need to perform two different tasks: the detection and localization of people in a video and the segmentation of the map to indicate if the robot is within a walked area. Figure 4 shows a schema of the reinforcement creation system. Our system can be composed of one or more observation agents, each of these agents will record data and perform the tasks of people detection and positioning in the map (figures 4(a) 4(b) and 4(c)). The segmentation of the map can be done either by the agents in a distributed way, or by a specific agent only dedicated to the creation of the reinforcement signal starting from the data provided by the observation agents.

A. People detection and positioning

The first step in order to position people on the map is a correct detection of those people in a video. The strategy has two requirements: it must run at real time with videos of 25/30 fps; and avoid the false positives. The environment where the system is going to be used is interior with artificial lighting, this facilitates the detection task as the illumination will be practically constant and the mobile objects in the scene will be restricted to persons. As can be seen in Figure 4(a), before detecting people we perform a motion segmentation of the current frame of the video. The technique used is based on the subtraction of a background image (static or periodically updated) from each frame. This method has proven to provide good results where the background is constant or it changes smoothly [3] [4].
After the subtraction of the background image we get a grey level image that needs to be segmented in order to have a clear distinction between foreground and background objects. In spite of the existence of methods for the dynamic calculation of the threshold [5], we have noticed that the thresholds provided were sometimes too low, what caused false positives, so after several tests we chose a static threshold. The resulting image (from now on motion image) is a binary image that contains foreground and background pixels 4(a).

Now the task is to detect people in the motion image. Amongst the foreground pixels in the image not all of them correspond to actual people, but to noise in the image, shadows or other moving objects, etc. The detection would not be perfect if one person is segmented as different blobs. Because of this, we will detect people by adjusting an ellipse to the contour of the blobs, and only those ellipses that have the proportions of a human body will be taken as people. The ellipses that doesn’t fulfil these requirements will be stored in a list of discarded ellipses. Some of the discarded ellipses might belong to a person whose body was partially detected by the motion detection process, and it is divided among several ellipses. We carry out an iterative process where the discarded ellipses are joined together two by two if they are close enough. The new ellipses are evaluated to see if the requirements to be considered a person are accomplished. If they are not, the new ellipse will be added to the discarded list. The process continues until no more linkages can be made.

The last process in the detection of people is the tracking and search for lost people, targets detected in previous frames but that are not detected at the current one. The tracking is made by assigning to each new target (person moving) an identification tag (ID), and in subsequent frames the same tag will be assigned to a target close enough to the centre of the old target. It may happen that one person detected in previous frames cannot be correctly segmented due to the presence of noise in the image. Figure 5 shows an example of this. We can see how the person moving in the corridor is initially segmented as several blobs (figure 5(b)). If these blobs are not big enough or are too far away to be merged together and thus make a new ellipse, the ID of the target will remain unassigned. The search of people that are missing in the current frame consists on finding blobs that might reflect a person moving and then draw a bounding box around them (figure 5(b)). We will consider only the blobs in the region of the image where the target was moving. If the proportions of the resulting box are similar to the lost target it is considered as the same person.

After the detection process we have a list of targets (people) detected and we need to transform their coordinates in the image \((x,y)\) to the coordinates in the real world map \((x',y')\):

\[
\begin{bmatrix}
  x'_1 \\
  x'_2 \\
  x'_3
\end{bmatrix}
= 
\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3
\end{bmatrix}
\]

Eq.1 provides the relationship between image coordinates \((x,y)\) and their corresponding map coordinates \((x',y')\):

\[
x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}
\]

H is a non singular 3 x 3 matrix called homography matrix.
Fig. 5. Detecting a person segmented in several blobs: a) original image provided by the camera showing one person walking towards the camera (white circle); and b) image showing the motion segmentation and the calculated bounding box containing the person.

\[ y' = \frac{x_2^2}{x_4} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \]  \hspace{1cm} (2)

Since \( H \) is a homogeneous matrix, if \( H \) is a solution to the problem, \( kH \) is a solution as well. Because of this, and if we consider \( k = 1/h_{33} \), we obtain a matrix with 8 unknown elements, since the last element is 1. Therefore equations 1 and 2 can be rewritten as:

\[
\begin{bmatrix}
  x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\
  0 & 0 & 0 & x & y & 1 & -y'x & -y'y
\end{bmatrix}
\begin{bmatrix}
  h_{11} \\
  h_{12} \\
  h_{13} \\
  h_{21} \\
  h_{22} \\
  h_{23} \\
  h_{31} \\
  h_{32}
\end{bmatrix}
= \begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
\]  \hspace{1cm} (3)

In order to solve eq. 3 we need to know four points of the image and their corresponding points in the map. It is obvious that we will need an exact map of the environment, but this should not be a problem in any building. This is the only calibration needed for the camera system. Figure 6 shows the perspective correction and the transformation of the coordinates of the four points used to calculate \( H \) in this environment.

Eq. 2 gives us a way to translate any pixel in the image into map coordinates. Therefore, we can obtain the coordinates of any person detected in an image, by only translating the coordinates of the lowest vertex of the ellipse around the person (figure 5(a)) into map coordinates.

As every person has a certain volume in space, its position in the map is represented by a square 20 cm wide.

### B. Segmentation of the map

Starting from the map used in the previous section to track people’s trajectories – figure 6–, we now want to obtain a reinforcement function that says when the robot is within the areas usually traversed by people (figure 7(a)). To do this, we used a LVQ-artificial neural network [8] [7], trained to recognize and generalize when a position in the map corresponds to an area where it is usual to find people moving.

The LVQ neural network can be seen as a classification artificial neural network with supervised learning. This network is able to divide the input space (coordinate positions in the map, figure 7(a)), into a set of regions (Voronoi regions, figure 7(b)), so that all the input patterns belonging to the same region are associated with the same class. In our case, there are two possible classes: walked and not walked. To train the LVQ neural network we need a training set and a testing data set. Those sets are made of random points taken in the map and labelled as walked or not walked. In fact, this information is obtained in the first module (figure 7.c) Once the network has been trained, it can be used to provide reinforcement while the robot is moving, in our particular case, the position of the robot every 250 ms is fed into the LVQ network. The output of the network – reinforcement – will be negative if the robot is outside the areas usually traversed by people and zero otherwise.
V. STAGE II: ROBOT LEARNING

Once the reinforcement generator has been obtained, the robot can learn from its interaction with the environment.

In the previous section we saw how to get a system that is able to monitor the environment, observe movement, and create reinforcement that will help to teach the robot to move where people do. Therefore the robot will receive a negative reinforcement when it enters a not walked area and no reinforcement while it runs over walked areas. There is also another reinforcement that can be considered as a natural reflex on the robot: when the robot is closer than 25 cm to any obstacle in the environment it will receive negative reinforcement. One of the reasons why this last reinforcement is needed is because people could have traversed doors that might be closed when the robot is moving (one can notice this effect in figure 7(b), there are green regions – corresponding to zero reinforcement – close to the limits of the map, these regions are near the doors located in the corridor, and the reinforcement is zero because people traversed them).

To teach the robot we have used a combination of reinforcement learning, genetic algorithms, and a dynamic representation of the environment, that has been proposed in the past by our research group, [1] and [2].

Through RL the robot is able to learn on its own, through trial and error interactions with the environment, using only the feedback provided by a very simple reinforcement function. Nevertheless, the large number of random actions taken by the robot, especially at early stages, makes this paradigm too slow and forces the use of robot-simulator to learn the optimal control policy [9]. The approach based on the combination of RL, GA and a dynamic representation of the environment through a finite set of different states increases the stability and speeds up the learning process.

Basically, when this approach is applied the robot goes through three cyclical and clearly differentiated stages (Figure 8): a) looking for a new starting position or convergence, b) exploration, and c) generation of a new population of solutions (chromosomes) for the next exploration stage.

During the first stage the robot uses the greedy policy to move according to what has been learnt so far. When the robot finds a new situation where it does not know how to move (local problem) it applies a genetic algorithm to find a local solution. Each chromosome codifies the action to execute in each state. Once the solution has been achieved a new population of chromosomes is generated and the
robot moves again using the greedy policy until a new local problem is detected. The mutual influence between RL, GA and adjustable number of states representing the environment varies throughout the learning process: the influence of the GA is very high at the beginning but it decreases over time. The influence of the RL is higher and higher as the number of robot-environment interactions increases and, finally, the dynamic number of states helps the robot to learn first how to behave in those situations that are more frequent, while the less frequent ones are learnt in a second stage.

The number of states used to represent the environment around the robot influences the time required to learn a satisfactory greedy policy. The properties of the Markov chains [10] are used to dynamically adjust the number of states involved in the learning process. As the robot is able to translate the different situations it may detect through its sensors into a finite number of states \( S = s_1, ..., s_N \), there is a \( N \times N \) transition matrix \( P \), where the element \( P_{ij} \) represents the probability of going from the state \( s_i \) to the state \( s_j \).

Therefore, if we know the current state of the robot a time instant \( t \), we can try to predict where the robot will be at \( t+1 \). \( \chi_t \) represents the probability distribution corresponding to the time instant \( t \). If \( P \) is the transition matrix for a Markov system, and \( \chi_{t+k} \) is a distribution vector with the property that \( \chi_{t+k+1} = \chi_{t+k}P = \chi_{t+k} \), then we refer to it as a steady vector. This means that the probability of finding the robot in state \( i \) is the same all the time: although the robot is jumping from one state to another, the probability distribution looks the same. As the steady vector \( \chi_{t+k} \) does not depend on \( t \), we will dispense with the subscript and denote it merely as \( \chi \).

In our case, during the learning process the transition matrix is estimated so that \( \chi \) can be calculated. Once this is done, and since \( \chi \) defines how the robot sees the environment in the long term, only those states for which the probability in the long term is not null are considered in the learning procedure.

Due to space restrictions it is not possible to provide more information about this learning strategy, nevertheless further information can be found in [1], where the methodology is described in detail.

VI. EXPERIMENTAL RESULTS

We now show the results we obtained when we applied our strategy to get robot learning from observation of human-behaviour at the Department of Electronics and Computer Science. As it was described in section III (experimental setup), we placed two cameras in a 16 meters long corridor. Each camera recorded 5 minutes of people moving in normal situations. The position data gathered with the two videos was joined to create a unique set of information for the entire map.

A. Results obtained for the first stage: automatic generation of reinforcement

The results obtained in the first stage, people detection and localization, were satisfactory. People walking alone or without occlusions were detected with no remarkable errors. When there were occlusions the system did not detect the movement silhouette as people, thus avoiding false positives. The fact that some people is not detected does not affect the system behaviour, that only means that more time will be needed to record a good amount of data. There are some problems with the shadows cast on the floor.

The localization in the map is precise provided that \( H \) is correct. Common errors came from wrong correspondences between the points chosen to solve the equation 3. We observed that the camera position was a key factor in the precision of the system. We got very good precision in the X axis whatever the distance of the target was, but the precision in the Y axis degraded as the target moved away from the camera– from 9 metres onwards we observed errors in the measurements above 30 cm.–. The higher the position of the camera was the better the precision obtained in the furthest area covered by it.

For the segmentation of the map we chose a LVQ network of 300 neurons that divided the environment in areas of around 0,10 \( m^2 \). These areas are big enough to provide a good generalization of all the positions and small enough to have good precision. The training set was composed of 50,000 examples and the test set had 25,000 examples.

The segmentation of the map provided by the LVQ network can be seen in Figure 7(b). The corridor is perfectly defined and there are even ways to get to some of the doors.

B. Results obtained for the second stage: robot learning

The robot used was a Pioneer 2-AT. In all the experiments the linear velocity was calculated in function of the angular velocity, the speeds where in [0.1,0.7] m/sec. The set of possible actions of the robot were the different angular speeds that the robot could choose. A total of 19 actions were taken, with angular speeds in the range \([-40,40]\) degrees/sec. The robot perceived its surrounding environment with a SICK LMS 200 laser rangefinder, taking measures every degree from 0 to 176. The environment around the robot is represented by a vector of 16 elements. Each of these elements is the minimum distance detected by the laser in groups of 11. We used a Kohonen network to translate the large number of different situations that the laser sensor may detect, into a finite set of 256 neurons (states). The robot was taught how to perform each task in a simulation of our corridor environment, the Player-Stage simulator was used in this stage. The criterion for convergence was that the greedy policy should be able to control the movement of the robot without errors for an interval of 10 minutes.

The learning of the behaviour using the combination of RL, GA and dynamic representation took about 2 hours and 30 minutes. It is a slow convergence time compared to the results obtained in [1] with the same learning algorithm. This can be explained by the fact that here the states (the perceptions of the robot) are not directly generating the reinforcement, the reinforcement comes from the coordinates of the robot in the map, which is not part of the robot’s internal state. Figure 9(a) shows the trajectories recorded.
from the cameras in the real environment. Figure 9(b) shows the trajectory of the robot in the simulated environment. Comparing the figures 9(a) and 9(b), we can see that the robot’s trajectory is similar to the one that a person would follow. The doors in the simulated environment are all closed, even at the ends of the corridor. In the real environment the ends of the corridor were open, so people could walk in and out of the area of interest. In the simulation the robot had to turn when it reached any of the ends of the corridor. When it turns, the robot has to make smoother turns than people would do, because our robot has a minimum turning radius due to the constant linear velocity. In this example the robot does never go towards the doors because they are all closed –the robot will receive negative reinforcement if it goes too close to any obstacle–. The global result shows that the idea of generating reinforcement from the observation of people walking is valid.

VII. CONCLUSIONS

In this article we suggest a methodology for learning behaviours with an automatic definition of reinforcement. Our final aim is the development of a robot system that can be used by anyone with minimum knowledge of robotics and computers, a plug and play system where the user just takes the robot and it starts to learn by itself.

The system has several modules that could be improved, but the overall result shows that the system is feasible. In our environment we only used two cameras, but with more cameras covering a complete building the robot would learn how to move in a complex environment without the help of human experts.

The modularity of the methodology makes possible the improvement of the system without compromising the stability of the results. The people detection system, although its implementation gives good results for our objective, could be improved to make it more robust and less sensitive to shadows. For the learning stage, any learning algorithm that learns from robot past experiences could be used.

A. Future Work

We are working on the testing of the system with a real robot, this is a “must do” to prove the capabilities of the system. These tests will be carried out on a bigger and more complex environment.

One qualitative step forward is the learning and classification of people trajectories. Now we learn individual positions in a map, but if we learn full paths between areas of interest of the map the robot will be able to perform more complex behaviours. We could also weight the regions into which the map is segmented, so that it is possible to emphasize the paths that are walked more often.

One drawback of our approach is the calibration of the cameras, it is easy but tedious to do it. An automatic map creation system which use the images from the cameras would release us of this task and would also make the system more self-sufficient.

The convergence time is too long. With a new definition of states which avoids perceptual aliasing with a fully adaptable and dynamic number of states the convergence time should drop quite significantly.

VIII. ACKNOWLEDGMENTS

This work has been financed by research grants FIT-350503-2007-8 and TIN2005-03844.

REFERENCES

Towards Binocular Active Vision in a Robot Head
Haitham Fattah, Gerardo Aragon-Camarasa and J. Paul Siebert

Abstract—This paper presents the first results of an investigation and pilot study into an active, binocular vision system that combines binocular vergence, object recognition and attention control in a unified framework. The prototype developed is capable of identifying, targeting, verging on and recognizing objects in a highly-cluttered scene without the need for calibration or other knowledge of the camera geometry. This is achieved by implementing all image analysis in a symbolic space without creating explicit pixel-space maps. The system structure is based on the ‘searchlight metaphor’ of biological systems. We present results of a first pilot investigation that yield a maximum vergence error of ~6.5 pixels, while seven of nine known objects were recognized in a high-cluttered environment. Finally a “stepping stone” visual search strategy was demonstrated, taking a total of 40 saccades to find two known objects in the workspace, neither of which appeared simultaneously within the field of view resulting from any individual saccade.

I. INTRODUCTION

The recent maturation of digital imaging hardware and the continual advancement of image processing and analysis techniques has vastly improved the potential for uses of computer vision in real-world scenarios. Binocular robotic vision has an advantage over monocular vision in potentially being able to compute range maps (i.e. distance fields to visible surfaces) by decoding the local parallaxes between captured stereo-pairs. Binocular imaging can also be used in object recognition to provide more information and therefore generate stronger object presence/identity hypotheses than would be possible with monocular vision alone. The development of an active vision control mechanism for a binocular camera system featuring object recognition and automated visual field exploration has potential applications such as: autonomous roving vehicles, automatic surveillance, telespresence systems and military applications. In this paper we present a system that integrates visual attention, vergence, gaze control and object recognition based on point matches extracted by means of the Scale Invariant Feature Transform [1] (SIFT). The system as devised provides an efficient means for controlling a binocular robot head system and a unified framework for binocular camera control.

This paper is organized as follows: in section II, we describe related work and the motivation that led us to design this particular system. We then describe the design of the vergence, object recognition and gaze control systems, in sections III, IV and V, respectively. Finally, section VI contains a summary of the system validation, its results and contributions to the field of active vision research.

II. RELATED WORK AND MOTIVATION

Several Binocular robot heads have been developed in recent decades. For example, the "Richard the First" head [2] and the KTH robot head [3] were capable of mimicking human head motion. More recent robot heads include the LIRA-head [4], where acoustic and visual stimuli are exploited to drive the head gaze; the Yorick head [5] or the Medusa head [6] where high-accuracy calibration, gaze control, control of vergence or real-time speed tracking with log-polar images were successfully demonstrated.

Despite advances in binocular robot heads, few systems are reported in the literature that integrate vergence and object recognition (operating on highly cluttered images) into a complete system capable of autonomously exploring its visual field. Therefore, our motivation is to investigate the potential for state-of-the art image processing techniques to enhance the performance of binocular robotic vision systems.

Vergence, in a biological context, is the act of adjusting relative angles of a pair of eyes to centre a real-world region of interest in the fovea of both eyes such that the dynamic range of parallaxes induced is minimised. In turn, this process maximises the visual information that can be extracted and perceived by the observer. There are many different possible models for implementing vergence in the context of a robotic binocular system. For example, by means of saliency detection or stereo-matching techniques such as: cepstral filtering [7], area based matching [5] and feature-based matching [8].

In this work, feature based matching offers advantages over area based techniques, such as [9]. For example, these advantages are evident when the surfaces are jagged or “spiked” or the local disparity gradient is near to an occlusion.

There are many different possible models for implementing vergence based on point matches. In the
context of vergence, these different models are concerned with: selective versus non-selective point matching, image independent vs. image dependent/inferred selective vergence and attended vs. non-attended vergence.

The above different models could be viewed as a Behavioural Hierarchy [10] that defines how the system should behave in given circumstances. The concept of modes of behaviour in gaze control is discussed in section III.

In the context of autonomous robot vision systems, the ability to identify and categorise objects imaged within the environment is essential. Accordingly, a system that can reliably identify objects in its field of view would find use in a broad range of applications. With regard to techniques which currently exist, however, generally applicable and robust methods are scarce. Approaches include: shape-based methods such as Belongie’s [11], which identifies correspondences between points on a shape and uses them to estimate an aligning transform; and Gevers [12] which combines colour and shape information into a high-dimensional descriptor of the object for recognition purposes.

It has already been stated, however, that the integrated framework designed in this paper makes use of the SIFT data generated by the vergence system for the purposes of object recognition. SIFT-based object recognition has been implemented in several systems such as Eklundh on the Yorick head, which could localize, attend and recognize objects [5] and the work by Kragic on robotic vision in a domestic context [13]. The SIFT algorithm was adopted in this work since implementations of the SIFT algorithm are readily available and SIFT can provide the basis for a reasonably general purpose object recognition system. In addition, SIFT can serve as a framework for point matching based on other sensing modalities. Our laboratory has now developed a version of SIFT adapted to operate on range images [14], offering the potential to extend the developed system in the future to take full advantage of its binocular imaging ability.

The process of using SIFT for object recognition is described concisely by Eklundh in [13]. Assuming a set of ‘known’ objects and a database that contains images of a number of poses of each, SIFT features are extracted for every image in the database. The integration of object recognition is included in this system in order to demonstrate its utility and the means of doing so in a structured and computationally parsimonious manner.

On the other hand, human visual attention is often described as being governed by the searchlight metaphor [15]. This suggests that human visual attention is separated into two modalities of analysis running simultaneously, with the output of one feeding into the other. These modalities are known as ‘pre-attentive’ and ‘attentive’ (further discussed in section V).

In machine vision, the above paradigm was adopted by Westelius [16] to drive the attention of his hierarchical gaze control. Earlier attempts of modelling attention in a computer vision context include Milanese’s [17] use of multiple feature maps. More recently reported developments in gaze control include the systems implemented in [18] that perform automatic saccadic gaze control in a mobile robot unit with active binocular cameras based on keypoint features; or in [13], which uses depth recovery to segment the scene by distance as part of an object-search strategy.

As discussed above, there are several distinct elements which drive an attention mechanism. The gaze control system adopted in this paper has been modelled on the searchlight metaphor of attention, including pre-attentive and attentive elements working in conjunction to guide the cameras.

To ensure that visual search progresses without endless backtracking, a mechanism for implementing inhibition of return (which also operates in a purely symbolic space) has been developed and integrated within the gaze control system.

III. VERGENCE

The requirements of the vergence system specify that the cameras are driven such that they target the same real-world position. There are several different modalities of vergence conceivable, including those operating on the following contexts when: the system is verging on a specific object or part of the scene, the content of the scene is known a priori and one camera already targets the desired location.

Thus, the behaviour of the system is contextually defined and task-motivated. We have attempted to structure the vergence system as a hierarchy of behaviours, related to Brooks’ Subsumption Architecture [19]. The two modalities considered are; Global, non-selective vergence and Attended, selective vergence.

The selective vergence case was developed as an adaptation during the design of the gaze control system as a special case of the non-selective vergence case (section V). The remainder of this section, therefore, refers mainly to the development of the design of the non-selective vergence.

The working hypothesis during the design of the vergence system was that it is possible to cause the cameras to verge by considering a global set of SIFT keypoint matches between the two camera images, i.e. keypoint correspondences between the images of the stereo-pair. For each pair of corresponding (i.e. matched) keypoints identified, the x-axis positions of these keypoints in each image are compared to produce a single-point disparity. For any given stereo-pair of images, there is likely to be a large number of such matches. An example of a stereo-pair captured by the robot head is shown in Fig. 1(a). Matched keypoints are joined by lines.

The algorithmic design is summarised in Fig. 2. To facilitate closed loop vergence, the disparity is measured again after the first iteration. If the post-verge disparity is
reported as larger than a tolerance value, another iteration is initiated.

Fig. 1. (a) The stereo-pair view of a highly cluttered scene. (b) x-coordinate (left) and y-coordinate histogram of disparities.

In a scene that does not contain depth (that is to say, one in which all viewable information exists in one plane, parallel to the camera baseline), all correctly identified keypoint matches will exhibit the same disparity value. However, we know this condition is not usually true for almost all non-trivial cases. Image keypoints that correspond to real-world locations at a range of distances from the cameras will exhibit a range of disparities.

The solution developed is to use the raw disparity data to infer information about the structure of the scene by identifying clusters, or peaks, of disparities. Since each disparity corresponds to a point somewhere on a surface at a specific distance from the cameras, we hypothesise that a large number of roughly similar disparities sample the surface of a potentially interesting object (i.e. an object comprising visual structure). An implicit assumption of this approach is that any object that is spatially compact in depth will form a disparity cluster around some mean distance to the cameras. Where several objects are present, the object with the most structure represented by keypoints, will give rise to the largest cluster and this can be identified by means of a simple histogram of the keypoint disparity values. An example of such a histogram (with bin width of 10 pixels), can be seen in Fig. 1(b).

An examination of the y-coordinate disparity histogram shows a clear peak around zero. This is expected, as the cameras should always be in vertical alignment, and therefore all correctly-matched keypoints will exhibit a near-zero vertical disparity. This assumption holds when the cameras are in a fronto-parallel position, however, as the cameras rotate away from this position, epipolar tilt induces a non-zero vertical disparity between corresponding keypoints.

Therefore, we created two constraints associated with each SIFT keypoint matched, these comprise the rotation and scale constraints defined as:

\[ \left| \theta_{\text{left}} - \theta_{\text{right}} \right| \leq 20^\circ \]  
\[ \frac{\sigma_{\text{left}}}{\sigma_{\text{right}}} \leq 45\% \]  
where \( \theta \) is the rotation value of the keypoint matches in left and right cameras which denotes the plane rotation of both images; and \( \sigma \) is the scale of the keypoint matches of both camera images.

In order to mitigate false SIFT feature matches, the disparity histogram proposed in combination with (1) and (2) made this algorithm robust with high cluttered scenes.

IV. OBJECT RECOGNITION

The design of the object recognition system is a direct adaptation of the SIFT-based object recognition first described by Lowe [1]. The relevance of the design to this project is found in the means of integrating the object recognition system in the overall framework. For completeness, a brief overview of the design is given below.

The basic function of the object recognition procedure is to compare each input image captured by the binocular camera-pair to all pre-stored object examples held in a database (in the form of sets of keypoints rather than images). Having applied the SIFT algorithm to all training images, the generated keypoints are stored in a data structure to facilitate searching during the subsequent object recognition phase.

The object recognition system will take the keypoints extracted from the current camera images and then match these to all keypoints in the database and then apply the Generalized Hough transform (GHT) [20]. There are typically several images of each object class in the database. Each database keypoint must, therefore, remain logically associated with an object class. When a keypoint match is found, it is registered as one vote for that object class. The integration of the recognition function into the overall framework is summarised in Fig. 3.

In an object recognition context, the GHT as Lowe described in [1], is used to strengthen a recognition hypothesis by establishing a measure of geometrical consistency between the test object and a reference object. This is performed by assigning votes into Hough-space bins for each matched SIFT feature. When a peak or cluster of votes is detected in Hough space, this indicates a consistent
interpretation for a number of features which has a much higher probability of being true than a single feature match. When the affine pose estimation is applied to a winning cluster of keypoint votes in the GHT, described in [1], this can provide a precise location of the centre of the hypothesized object.

To obtain the position of any point of a database object in the scene, the affine pose estimator is used as follows;

\[ y = Ax \cdot \text{PS}_{\text{ratio}} \]  \hspace{1cm} (3)

Fig. 3: The method of interfacing the recognition function. Note the use of passing keypoint data instead of image data.

where, \( A \) is the affine transformation, described above, \( x \) the centre points of the image, \( \text{PS}_{\text{ratio}} \) a pixel-space ratio which corresponds to the total number of motor steps per pixel of translation in the image and \( y \) is the spatial location of the object in the scene. The actuators can then be driven to target the cameras at that point.

The interface to the object recognition function is intentionally low-level to provide the maximum level of flexibility in its use. Notably, the recognition module does not return a set of recognized objects, but a set of all objects in the database, with the associated number of matches to each database object.

The confidence for any given object is defined as the confidence of the highest-peak in Hough space for the recognized object which is used in section V to saccade the cameras towards the object with highest confidence value.

V. GAZE CONTROL

The design of the behavioural system aims at achieving a gaze control strategy for search that uses the vergence and object recognition functions (described in sections III and IV) to explore the scene that it is presented with. We have developed an attentional system that operates purely in symbolic space by use of the SIFT keypoints. This allows a single set of image features to be used for the entire, heterogeneous set of tasks required.

A flow chart of the behaviour of the system can be seen in Fig. 4. The pre-attentive and the attentive functions work in a quasi-parallel manner, with the output of one feeding to the input of the other.

The pre-attentive function is concerned with analyzing the current field of view to detect salient features. This phase does not make recognition decisions; it is solely responsible for detecting areas of the field of view that may be of interest to the search strategy. These highlighted image features are passed to the attentive function (as in 3), which then guides the ‘searchlight’ to examine these areas in detail.

The attentive phase uses the information provided by the pre-attentive function to target the cameras and make recognition decisions. This phase selects which attentive item to visit next, and directs the cameras to the reported location.

As a consequence of the pre-attentive phase, the gaze control system will ‘notice’ objects and keypoints only when they appear in the view of the dominant eye (the left camera). Since the cameras are only driven to look at objects and keypoints, salient items will only be registered if they appear in the field of view when the cameras are targeting another salient item. This system, therefore, follows a ‘stepping-stone’ search pattern, due to the way that the system will notice a second object when saccading to target the first. An object will only reach the attention of the system if it appears close enough to a targeted object or can be reach by a ‘bridge’ of other objects and keypoints. Salient keypoints are those keypoints in the left camera image that are found not to match to a database image and exhibit a saliency score above a threshold. The saliency score (\( S_{\text{Saliency}} \)) for each keypoint is then computed as:

\[ S_{\text{Saliency}} = x_{\text{offset}} \times y_{\text{offset}} \times \sigma \times 10^{-3} \]  \hspace{1cm} (4)

Note that \( x_{\text{offset}} \) and \( y_{\text{offset}} \) denotes the horizontal and vertical distance from the left image centre respectively and \( \sigma \) the scale value of the keypoints matched.

The mean and standard deviation is computed over all saliency scores from unrecognized keypoints in each fixation. These scores are then filtered to keep only those that exceed three standard deviations of the currently input keypoint population. In selecting which keypoint to target, the attentive function selects the unvisited keypoint with the highest saliency score from working memory.

To inhibit return to salient image features that have already been attended, a list is maintained of those salient, unrecognized keypoints that have been attended (verged on) by the system (6). When the pre-attentive phase is analyzing the current image for new salient keypoints, each keypoint is compared to all keypoints in this list using the Lowe’s matching algorithm. If an input keypoint is found to match to a keypoint in the visited list, it will be discarded.

\[ \text{Objects} = \{ S_{\text{Saliency}}, \text{Descriptor}, \text{Location in the image, Attended} \} \]  \hspace{1cm} (5)

This principle leads to the inclusion of the unrecognized, salient keypoints as a target of the pre-attentive system. It is hypothesised that by allowing the cameras to follow
unrecognized image structures, it provides a semi-guided method for exploring parts of the scene that would not be reached if the cameras only followed recognized objects.

![Flow Chart](image)

**Fig. 4.** A flow chart showing a high-level view of the behaviour of the gaze control system.

When saccading to a recognized object, it is necessary to verge the camera-pair such that the object of interest is centred in the field of view of each camera (3rd order case in section III). The approximate location of the object is known at saccade-time as its coordinates are passed from the pre-attentive phase, where they are calculated by means of the affine pose estimator (section IV).

To ensure that the vergence operation targets only the desired object, the coordinate frame is translated to actuator units to calculate the required actuator movement to centre the object in view. Subsequently, only those database keypoints that match to the target object are used in the disparity calculation (algorithm of Fig. 2), and hence only that object will be considered.

**VI. EXPERIMENTAL DESIGN AND RESULTS**

**A. Binocular camera robot head configuration**

The physical robot head [21] used in this work comprises the following: two SONY cameras XCD700 (1024×768 pixels resolution) fitted with IEEE Firewire interfaces and four high-accuracy stepper-motors and motor-controllers (Physik Instrumente GmbH & Co.).

The hardware was interfaced to a Pentium 4 computer with a CPU clock speed of 2 GHz, with 2 GB in RAM running under Windows XP and MATLAB.

**B. Vergence system validation**

As previously explained, the vergence mechanism, by applying different modes of operation based on different visual search conditions, can be viewed as implementing a behavioural hierarchy. The non-selective and the selective vergence levels of this hierarchy have been implemented in this system. As is described in previous sections, the selective vergence case is implemented as a special case of the non-selective case. Therefore, the experimentation on the vergence system is aimed primarily at the non-selective case. The correct function of the selective vergence case is validated as part of the gaze control system.

The objective of the non-selective vergence case is to minimise the total horizontal disparity between a global set of uniquely corresponding locations identified in the current camera images when no target has been identified in the current field of view. The statistical accuracy and reliability of the vergence system is measured by observing the system behaviour when presented with a number of scenarios: a situation when all keypoints appear in a single depth plane; a situation in which a disparity step (resulting from two juxtaposed planes at different distances to the cameras) is present in the field of view; and a realistic situation in which keypoints come from a continuous range of possible depths.

To create a scene in which all identifiable detail occurs on a single plane, a printed image was mounted to a board, which was mounted on a bench at known distance from the camera baseline. The vergence algorithm in Fig. 2 was initiated and allowed to execute until it settled with a tolerance value of ±4 pixels. This process was repeated six times at different depth locations. In every case, it took 2 iterations for the vergence to settle. It can be seen from Fig. 5 that there appears to be no correlation between the distance of the target from the cameras and the accuracy of the vergence. The worst single vergence error observed in all 36 verges was an error of ~5.3 pixels from optimal. The average overall accuracy is ~1.4 pixels of error. Both values are objectively small and therefore acceptable for most applications.

Likewise, to create a scene that contained identifiable detail in two separate depth planes, a second different printed image was mounted to a board mounted adjacent to the previously mentioned printed-image. The first image was kept at the same distance to the camera baseline, while the distance of the second image was varied. The same number of iterations as in the first experiment was performed. The vergence error was measured in the same manner and the resulting data is shown in Fig. 6.

It is notable that a lower overall accuracy is observed when comparing the results in Fig. 6 with Fig. 5. The overall mean vergence error is over twice that of the first experiment. Objectively, the mean vergence error is still sufficiently small to allow dense disparity fields to be recovered through stereo-matching. The worst single vergence result observed was ~6.5 pixels of error.
In a real-world scenario, accuracy of vergence is harder to measure quantitatively. A precise value of the vergence error could be calculated in the previous experiments as there is a clearly definable ‘optimal’ verge point. A sample of the images produced during this experiment is shown in Fig. 7(a). The most notable of these images is the anaglyph showing the camera views after the verge (Fig. 7(b)). The object that exhibits fewer matches, in this case the lion toy is disregarded in the vergence. The resulting alignment of the skull shows the left eye to be precisely verged, whereas regions of the skull that are further away are, naturally, less verged. This level of overlap of the skull is, therefore, probably as good as could be expected. However, the vergence actually achieved, it is satisfactory for the purposes of 3D reconstruction. The average execution time required to verge the cameras was 73.8 seconds.

C. Gaze Control System

The gaze control system was developed to demonstrate how the SIFT-based vergence technique could be combined with SIFT-based attention and recognition in a search strategy. To test this system, it is first necessary to isolate the various functions and operational modalities of gaze control. The functions of the system are listed below as three individual units of functionality that can each be verified.

1. The system should detect the presence of a recognized object when it is in the field of view of the dominant camera, recording its position in the actuator space. When the system is aware of one or more possibly recognized objects, it will saccade to and verge both cameras on the object with the highest confidence of recognition (selective case of vergence).

2. The system will use a combination of recognized objects and salient keypoints in a ‘stepping stone’ process to explore the scene, reporting all objects recognized therein.

3. If an object has been previously recognized, no attempt will be made to return attention to that object again. When the system has not seen any possible but unverified objects, it will saccade to the most salient, previously seen keypoint.

To verify the correct operation of the first function, as detailed above, we present the system with a highly-cluttered scene, as shown in Fig. 8(a). In the pre-attentive cycle, the skull and the car were identified, the confidence value, evaluated in section IV, was used to discriminate which object to attend (in this particular case, the car was attended).

The correct identification of the car object corroborates the ability of the system to detect and classify correctly objects in the field of view. Fig. 8(b) represents the views of both cameras after the saccade to the object is performed. Note that this image was captured before the vergence cycle; hence the poor amount of overlap. It can be seen that the car is correctly positioned near the centre of both images. This validates the requirement of the system to be able to correctly identify the actuator-space location of an object in the field of view.

To verify the second function, the system was presented with a scene containing all known objects and allowed to run until a halt condition was reached, i.e. no new objects could be detected. While it is therefore expected that the behaviour of the system will conform to the functionality described, it does not necessarily follow that 100% identification of objects in the field of view will be achieved. Fig. 9(a) and (b) shows the actuator-space motion trace of both cameras. The stepping stone search pattern can be seen by further examination of the image in Fig. 10.

Fig. 5. The verge errors on a single plane over six iterations at each six distances. The RMS error is given in pixels, the distance in millimetres.

Fig. 6. The verge errors in two separate depth planes over six iterations at each six distances. The RMS error is given in pixels, the distance in millimetres.

Fig. 7. (a) An anaglyph of the left and right camera images before verging. (b) An anaglyph showing the camera images after vergence had settled.

Fig. 8. (a) Initial field of view of the camera, (b) Anaglyph of the camera images after the saccade to the position of the car and before the vergence cycle.

It can be seen that, the object being targeted appeared in
the field of view in the previous fixation (the target process is represented with capital letters in Fig. 10). The system failed to identify one object in the scene, a bike toy, which does not have a registered fixation. The saccade denoted as C in Figure 10, which corresponds to the mouse, does not centre the object as expected, thus it is considered a false positive match. It is not necessary that the next object to be saccaded to is selected from the current camera image; it is the most highly matched object of all unattended candidate objects that is selected. In the results presented there are no examples of a saccade to an object that was identified several cycles prior, however this is not due to design, but simply a property of the way in which the scene is structured in this particular run. The execution time required to explore the scene and to recognize the objects shown in Fig. 10 was 31.5 minutes.

The objective of the work reported here is to develop a binocular robot vision system capable of autonomous scene exploration, with the goal of identifying and localising objects within known classes while maintaining binocular vergence. We present a system that demonstrates the application of several novel design principles in a functional, integrated framework that essentially achieves the above objectives.

Adopting SIFT features as the underlying visual representation for our active gaze control system allowed a single mechanism to combine elegantly the key functions of binocular vergence, object recognition and saccade selection.

The approach of computing the vergence signal that drives the binocular camera pair, based on finding the highest feature density peak within a SIFT derived disparity histogram, proved to be robust and effective. The maximum vergence error observed of ~6.5 pixels remains within viable limits for any subsequent depth recovery task based on stereo-matching. We anticipate that by couching the vergence mechanism as a behavioural hierarchy, it will be possible to structure this algorithm efficiently to meet the needs of different operational contexts.

Saccade selection drives our gaze control system (section V) and likewise adopts SIFT keypoints as the basis of...
attention and inhibition of return mechanisms.

Additionally, a functional implementation of a standard SIFT object recognition module has been embedded in a conceptually uncomplicated manner. In the preliminary results presented, seven of nine objects were recognized in a highly cluttered scene. Accordingly, the system was able to demonstrate centring its gaze on each detected object of interest in each fixation of the camera-pair, as expected. In addition, we demonstrated that the implemented ‘searchlight-metaphor’ of visual attention could navigate between two widely separated known objects embedded within unknown clutter.

Hence, it is now possible for a binocular robotic vision system to direct its gaze on a scene such that it: maintains binocular vergence, detects salient image features, directs its gaze to investigate these features, verifies the identification of objects and continues to investigate the workspace for recognized objects based on visual cues. All these characteristics are combined in a computationally parsimonious manner using SIFT descriptors. Fig. 14 shows the experimental configuration of the robot head cameras and the scene.

It must be emphasised that we have presented the first preliminary results of validating the active binocular vision system reported here. The next objective is to perform a more complete validation involving a wider range of scenes and randomised initial fixation points in order to generate sufficient statistics to characterise reliably the performance of the system.

It should also be noted that the current system implementation is not intended for real-time operation, however, we believe that this can be achieved by means of GPU acceleration of both the SIFT algorithm [22] and critical sections of the vergence and saccade selection mechanisms.

Our current work now focuses on automatic clustering in a continuous Hough space to allow multiple same-class object instances to be localised accurately. In the future we propose to investigate range-map recovery [23] and use of 2.5D SIFT [14] features in conjunction with 2D SIFT features to improve object identification and 3D pose recovery.

REFERENCES

Carpet, Field and Air: Mobile Robots for Real World Environments
Roland Siegwart
Autonomous Systems Lab
Institute of Robotics and Intelligent Systems
ETH Zürich, Switzerland
rsiegwart@ethz.ch

Abstract

Robots are rapidly evolving from factory “workhorses” that are physically bound to their work-cell to machines evolving in our environment, fulfilling challenging tasks. This development makes systems obviously much more complex and raises important new questions concerning robot design and control. In this talk various projects of the speaker’s research group addressing challenging robot designs will be presented and discussed. This includes:

- Insbot, a tiny little robots socializing with cockroaches. Based on simple interaction rules and behaviors, these robots are able to significantly influence the collective decision process of the mixed society in test environment.

- Crab, an all terrain rover ready to negotiate rough terrain on future exploration missions.

- Sky-Sailor, a solar micro-airplane designed for continues flight on Earth and Mars.

- RoboX, a family of eleven tour guide robots that got in touch with hundred thousands of visitors during the Swiss national exhibition expo.02.

- SmartTer, the intelligent car for increased road safety. Thanks to its on-board sensors it is capable to build consistent 3D maps of its surrounding.

Biography

Roland Siegwart is full professor for autonomous systems at ETH Zurich since July 2006. He has a Diploma in Mechanical Engineering (1983) and PhD in Mechatronics (1989) from ETH Zurich. In 1989/90 he spent one year as postdoctoral fellow at Stanford University. After that he worked part time as R&D director at MECOS Traxler AG and as lecturer and deputy head at the Institute of Robotics, ETH Zrich. In 1996 he was appointed as associate and later professor for autonomous microsystems and robots at the Ecole Polytechnique Federale de Lausanne (EPFL). During his period at EPFL he was Deputy Head of the National Competence Center for Research (NCCR) on Multimodal Information Management (IM2), co-initiator and founding Chairman of Space Center EPFL and Vice Dean of the School of Engineering. In 2005 he held visiting positions at NASA Ames and Stanford University.

Roland Siegwart is an IEEE Fellow, member of the Swiss Academy of Engineering Sciences and board member of the European Network of Robotics (EURON). He served as Vice President for Technical Activities (2004/05) and Distinguished Lecturer (2006/07) and is currently AdCom Member (2007-09) of the IEEE Robotics and Automation Society. He is coordinator of two European Projects and co-founder of several spin-off companies.
Abstract

The ICRA contingency challenge was held for the first time this year. The challenge was to devise robotic solutions to problems occurring at a human habitat on Mars (for practical reasons simulated here on Earth). The problems were unknown to the teams beforehand, but still they had to develop solutions within four hours only using what they could fit in a suitcase.

In this talk I will present USD Modular Robotics Lab’s approach to this challenge. The main elements of which are a heterogeneous modular robot, that allows for rapid construction of task-optimized robots, and a flexible distributed programming and control framework.

I will continue to describe some of the basic research topics we are addressing to further increase the usefulness of modular robots. These topics include collective actuation, distributed learning and control, and software engineering for modular robots. Finally, I will present two potential applications of modular robots that we are currently working towards: robust locomotion and morphing production lines.

Biography

Kasper Støy is a co-director of USD Modular Robotics Lab and is an associate professor at The Maersk McKinney Moller Institute, University of Southern Denmark. Kasper received his MSc from the University of Aarhus, Denmark and his PhD from USD in 2003. He spent a year of his PhD studies at USC’s Information Sciences Institute and a year at USC Robotics Labs, USA. In the research field of modular robots Kasper has published thirteen first-author papers of which three received awards. He organizes international workshops, serves as reviewer for IEEE conferences, International Journal of Advanced Robotics, Journal of Autonomous Robots, and Journal of Simulation of Adaptive behaviour and several more. Kasper currently manages the “Morphing Production Lines” research project funded by the Danish Research Council for Technology and Production and the “Manipulation Using Robotic Deformable Material” research project funded by Intel Research Pittsburgh. He also developed the first version of the Player component of the multi-robot simulation tool Player/Stage, which is the most widely used simulation in this field. Kasper’s research interests include modular robotics, distributed control and embodied artificial intelligence.
Robot Localization Using Seismic Signals

Theocharis Kyriacou1, Peter Styles2, Sam Toon2

Research Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM)
1School of Computing and Mathematics, 2School of Physical and Geographical Sciences
Keele University, United Kingdom, ST5 5BG

Abstract—A numerical method is proposed that estimates the position of a moving robot using the seismic signals caused by the robot on the surface upon which it moves. The work presented here describes an initial attempt to solve this problem using an indoors model setup. This method of localizing moving vehicles will be more effective in outdoor environments where longer travel distances for seismic waves make separation between different types of waves more prominent and therefore measurable. Contrary to existing methods of localizing moving outdoor vehicles, this method requires no prior setup on the vehicle itself. Two experiments are presented: first, using a single robot in a seismically “quiet” environment and second, using two identical robots while trying to estimate the location of one of them.

I. INTRODUCTION

In many robotic applications, the true (not estimated from odometry) position of a robot in its environment is very useful information. This is especially true when robots express their behavior in terms of their trajectory.

Several methods exist to find the location of the robot. In relatively small and usually enclosed spaces, vision tracking systems are used whereby one or more active or passive markers are placed at various points on the robot and are continuously being tracked using specialized video cameras (see [2], [3]). Several cameras are normally used that look down on the robot’s environment from different angles thus minimizing the possibility of complete occlusion of a particular target. Another advantage of using many cameras is that the accuracy of the system is increased. Such systems are very accurate (typical errors are less than a millimetre [1]) and very fast (typical speeds are several hundred samples per second for each target). Such systems however require considerable setup and calibration time which can discourage their use for constantly changing setups. Furthermore, such systems are usually unsuitable for outdoors applications because of the larger environments that need to be covered but also due to external illumination problems or reflections that can interfere with their principle of operation.

Robot localization methods that use various types of rangefinder sensors (stereo vision, sonar and laser rangefinders) are well researched. Some examples include [4], [5], [6]. Here the robots localize themselves by continuously tracking several fixed objects in the environment with their sensors. By subsequently using the estimated location of the static objects the location of the robot itself can be estimated. An often desirable product of these methods is the map of the environment, hence the common acronym SLAM (Simultaneous Localization And Mapping) in this context. Such localization methods pose a significantly large computational load and can therefore be slow for relatively fast moving vehicles. In outdoor environments the effectiveness of these methods depends on the type of rangefinder sensors used and also on the availability of features to track. A large flat terrain would perhaps pose a challenge for this method of localization.

For outdoor applications, perhaps the most popular solution for finding the location of a robot is the use of the Geostationary Positioning System (GPS). Examples include [7], [8]. The GPS receiver on board the robot calculates its position using its distance from several satellites orbiting the earth. The distance of the receiver from each of the satellites is established by measuring the time delay that an electromagnetic signal takes to travel from the satellite to the receiver. As well as providing a synchronization mark, the signal from the receiver also contains the location of the satellite and time synchronization information. Even though the GPS system is capable of sub-metre accuracy, this accuracy is variable and it depends, among other things, on the number of satellites tracked by the GPS receiver at any given moment as well as obstructions in the environment. In extreme cases, such as the situation where the robot is under a canopy of trees, under a bridge or in a tunnel the error can worsen by one or two orders of magnitude ([9], [10]). The use of this localization method depends on the reliability of operation of the GPS system of satellites.

For the last two localization methods mentioned above, if the position of the robot is required in real time by a remote host, a wireless radio link needs to exist between the robot and the host for the robot to communicate its position to the host. This methods therefore require prior setup on the robot.

The work presented here is a first attempt to develop a method that can be used to estimate the location of a robot (but more generally any moving vehicle) using the seismic activity produced by its motion. The seismic activity is measured by using a number of seismic sensors placed in or around the robot’s environment. The seismic sensors are essentially very sensitive accelerometers that can measure acceleration along one or more of the tree coordinate axes.

One solution to the problem of calculating the robot’s position for the seismic sensor outputs is to use analytical methods. This however, is a very complex problem to solve analytically. It would require precise knowledge of the robot’s terrain and sub-terrain morphology as well as the
kinematic and physical characteristics of the robot itself.

In this paper we propose a numerical method for solving the above problem. We achieve this by finding the structure and parameters of a NARMAX (Nonlinear Auto Regressive Moving Average Model with eXogenous inputs) polynomial ([11], [12], [13]) by using corresponding seismic and robot location data while a robot is moving in its environment. The resulting polynomial is a mathematical expression giving the robot’s position as a function of the signals obtained from the seismic sensors. A brief explanation of the NARMAX model estimation methodology is given in section II-A.

A. Seismic Waves and Seismic Event Localization

A full discussion of seismic waves is beyond the scope of this text. The interested reader may wish to refer to a more in-depth explanation of seismic wave properties, their formal description and analysis in [14]. Only a short introduction that serves the purposes of the work described in this paper is given below.

When a seismic event is caused in a medium, two main types of waves are produced that carry the energy of the event away from its location. These are the \( P \) waves (Primary waves) and \( S \) waves (Secondary waves) and they fall under the broader category of body waves. \( P \) waves are compressional vibrations where the direction of particle motion is in parallel to the direction of propagation of the waves. In gaseous media these take the form of sound waves and their speed depends on the density of the medium. In air for example the speed of \( P \) waves is around 330m/s. In water \( P \) waves travel at speeds of about 1500m/s and in solids they can reach speeds of more than 5000m/s depending on the density of the solid. \( S \) waves travel at speeds around 2/3 those of \( P \) waves. \( S \) waves can only propagate in solids and the distortion of the solid medium is at right angles to the direction of propagation of the wave. The polarization of the \( S \) wave determines the displacement direction in the medium on the plane that is perpendicular to the wave propagation.

When free surfaces exist in a medium, interacting systems of body waves (\( P \) and \( S \) waves) give rise to another class of waves called surface waves. These are also subdivided into two categories: Rayleigh and Love waves. Rayleigh waves travel as ripples similar to those on the surface of water. The displacement of the medium in which they travel contains both vertical and radial components. Love waves on the other hand are purely transverse (i.e. the medium displacement is vertical to the direction of motion of the wave. Surface waves travel at speeds around half those of \( P \) waves with Love waves being slightly faster than Rayleigh waves. The amplitude of surface waves is maximum at the surface of the travel medium and diminishes fast with depth. The nature and location of seismic events (such as earthquakes or explosions on earth for example) can be found using a network of seismic monitoring stations at different locations on earth. Data such as the intensity and nature (\( P \) or \( S \)) of the waves reaching each station, their relative time of arrival and frequency and the relative position of the seismic stations are used to identify and locate the seismic event.

In the work presented here, the seismic events that we try to locate are produced by a moving robot. For these initial experiments a model indoors setup was constructed and used that includes small robots moving on a wooden platform. The use of the relative time of arrival of seismic waves in order to locate the seismic events cannot be applied in this case because the distances between the robot and the seismic sensors used are relatively small. This scale prevents any wave propagation time differences from being measurable. Furthermore wave reflections off the boundaries of the robot’s small environment make things a lot more complex to analyze formally. Instead, a numerical analysis is used to find the robot’s position from the seismic activity that it produces. In this first instance only the most prominent characteristic of the seismic activity is used: it’s amplitude. This is measured along one or more of the three coordinate axes using seismic sensors.

II. EXPERIMENTAL PROCEDURE AND METHODS

A. The NARMAX model estimation methodology

NARMAX (Nonlinear Auto-Regressive Moving Average model with eXogenous inputs) is a polynomial function that we can use to represent the input/output relationship between the inputs \( u \) and the output \( y \) of a Multiple-Input/Single-Output (MISO) non-linear system. The aim of the NARMAX model estimation methodology is to determine the structure and parameters of the polynomial using input/output examples of the system under investigation. By structure, we mean the terms of the polynomial function and by parameters we mean the coefficients of those terms. The purpose of the estimated model is twofold: to use it in order to predict the output for new or unseen values of the input but we also hope, by studying the model structure and parameter values, to be able to characterize the system. This model estimation methodology has been used in several domains. Some examples include [15], [16], [17].

For a noisy MISO system, the output \( y \) at discrete time \( n \) can be generally represented by:

\[
y(n) = f(u_1(n), u_1(n-1), u_2(n-2), \ldots, u_1(n-N_u), u_1(n)^2, u_1(n-1)^2, u_1(n-2)^2, \ldots, u_1(n-N_u)^2, \ldots, u_2(n)^2, u_2(n-1)^2, u_2(n-2)^2, \ldots, u_2(n-N_u)^2, \ldots, u_d(n)^2, u_d(n-1)^2, u_d(n-2)^2, \ldots, u_d(n-N_u)^2, \ldots, u_d(n), u_d(n-1), u_d(n-2), \ldots, u_d(n-N_u), u_d(n)^2, u_d(n-1)^2, u_d(n-2)^2, \ldots, u_d(n-N_u)^2, \ldots, u_d(n))^j, \ldots, u_d(n-1)^j, u_d(n-2)^j, \ldots, u_d(n-N_u)^j, \ldots, \]

\[= \sum_{j=1}^{N_f} \sum_{i=1}^{N_u} \sum_{k=1}^{N_d} a_{ijk} u_i(n)^j u_k(n)^k + r(n),\]

where \( N_f \) is the number of model parameters and \( r(n) \) is the model error. When a seismic event is caused in a medium, two main types of waves are produced that carry the energy of the event away from its location. These are the \( P \) waves (Primary waves) and \( S \) waves (Secondary waves) and they fall under the broader category of body waves. \( P \) waves are compressional vibrations where the direction of particle motion is in parallel to the direction of propagation of the waves. In gaseous media these take the form of sound waves and their speed depends on the density of the medium. In air for example the speed of \( P \) waves is around 330m/s. In water \( P \) waves travel at speeds of about 1500m/s and in solids they can reach speeds of more than 5000m/s depending on the density of the solid. \( S \) waves travel at speeds around 2/3 those of \( P \) waves. \( S \) waves can only propagate in solids and the distortion of the solid medium is at right angles to the direction of propagation of the wave. The polarization of the \( S \) wave determines the displacement direction in the medium on the plane that is perpendicular to the wave propagation.

When free surfaces exist in a medium, interacting systems of body waves (\( P \) and \( S \) waves) give rise to another class of waves called surface waves. These are also subdivided into two categories: Rayleigh and Love waves. Rayleigh waves travel as ripples similar to those on the surface of water. The displacement of the medium in which they travel contains both vertical and radial components. Love waves on the other hand are purely transverse (i.e. the medium displacement is vertical to the direction of motion of the wave. Surface waves travel at speeds around half those of \( P \) waves with Love waves being slightly faster than Rayleigh waves. The amplitude of surface waves is maximum at the surface of the travel medium and diminishes fast with depth. The nature and location of seismic events (such as earthquakes or explosions on earth for example) can be found using a network of seismic monitoring stations at different locations on earth. Data such as the intensity and nature (\( P \) or \( S \)) of the waves reaching each station, their relative time of arrival and frequency and the relative position of the seismic stations are used to identify and locate the seismic event.

In the work presented here, the seismic events that we try to locate are produced by a moving robot. For these initial experiments a model indoors setup was constructed and used that includes small robots moving on a wooden platform. The use of the relative time of arrival of seismic waves in order to locate the seismic events cannot be applied in this case because the distances between the robot and the seismic sensors used are relatively small. This scale prevents any wave propagation time differences from being measurable. Furthermore wave reflections off the boundaries of the robot’s small environment make things a lot more complex to analyze formally. Instead, a numerical analysis is used to find the robot’s position from the seismic activity that it produces. In this first instance only the most prominent characteristic of the seismic activity is used: it’s amplitude. This is measured along one or more of the three coordinate axes using seismic sensors.

II. EXPERIMENTAL PROCEDURE AND METHODS

A. The NARMAX model estimation methodology

NARMAX (Nonlinear Auto-Regressive Moving Average model with eXogenous inputs) is a polynomial function that we can use to represent the input/output relationship between the inputs \( u \) and the output \( y \) of a Multiple-Input/Single-Output (MISO) non-linear system. The aim of the NARMAX model estimation methodology is to determine the structure and parameters of the polynomial using input/output examples of the system under investigation. By structure, we mean the terms of the polynomial function and by parameters we mean the coefficients of those terms. The purpose of the estimated model is twofold: to use it in order to predict the output for new or unseen values of the input but we also hope, by studying the model structure and parameter values, to be able to characterize the system. This model estimation methodology has been used in several domains. Some examples include [15], [16], [17].

For a noisy MISO system, the output \( y \) at discrete time \( n \) can be generally represented by:

\[
y(n) = f(u_1(n), u_1(n-1), u_2(n-2), \ldots, u_1(n-N_u), u_1(n)^2, u_1(n-1)^2, u_1(n-2)^2, \ldots, u_1(n-N_u)^2, \ldots, u_2(n)^2, u_2(n-1)^2, u_2(n-2)^2, \ldots, u_2(n-N_u)^2, \ldots, u_d(n)^2, u_d(n-1)^2, u_d(n-2)^2, \ldots, u_d(n-N_u)^2, \ldots, u_d(n), u_d(n-1), u_d(n-2), \ldots, u_d(n-N_u), u_d(n)^2, u_d(n-1)^2, u_d(n-2)^2, \ldots, u_d(n-N_u)^2, \ldots, u_d(n))^j, \ldots, u_d(n-1)^j, u_d(n-2)^j, \ldots, u_d(n-N_u)^j, \ldots, \]

\[= \sum_{j=1}^{N_f} \sum_{i=1}^{N_u} \sum_{k=1}^{N_d} a_{ijk} u_i(n)^j u_k(n)^k + r(n),\]

where \( N_f \) is the number of model parameters and \( r(n) \) is the model error.
were $u(n)$ and $e(n)$ are the sampled input and error signals at time $n$ respectively, $N_u$, $N_e$ and $N_y$ are the regression orders of the input, error and output respectively and $d$ is the input dimension. $f(\cdot)$ is a polynomial multi-resolution expansion of the arguments. The degree $l$ of the polynomial is the highest sum of powers in any of its terms. The degree specification provides a limit to the polynomial expansion.

Noise is always present in physical systems. During the NARMAX model estimation methodology noise is accommodated in the error terms. In the final model the structure of the error terms can reveal the nature and significance of the noise present in our system. We can keep the error terms in our final model if the error (i.e. the difference between the model predicted output and actual output) will be revealed to us for samples at time equal or earlier than $n - 1$. This is useful if we want to use our model for m-step ahead prediction.

In most cases however, the error will not be known in which case the error terms will be discarded from the model after their calculation. This of course will leave the noise unaccounted for but provided this is small enough, compared to our output, we can treat it as an acceptable error.

Theoretically, if the noise in the system is white and we had infinite model estimation data then it would be possible for the NARMAX methodology to identify the underlying system even without incorporating error terms in its structure. In the work presented here we assume that the noise in our system is white and relatively small compared to the output. This assumption is based on model estimation attempts that revealed no significant relationship between error terms and the system inputs or output. We therefore do not use any error terms in our model structure.

The NARMAX methodology is an iterative process that tries to minimize the model’s prediction error (i.e. the difference between actual and model predicted output) by changing the structure and parameters of the model (see [18]). A brief overview of the algorithm is given below.

Two sets of data are used: one for estimating the model structure and its parameters and the other for validating the model. During every iteration of the algorithm the structure of the model polynomial is changed by removing non-contributing terms and the remaining model parameters (the coefficients of the remaining terms) are re-calculated so that the best possible fit is achieved on the estimation set output. This iterative process stops when the remaining polynomial terms are considered to be significant contributors to the calculation of the output. In other words, the removal of any of the remaining terms would cause the prediction error of the model on the estimation data set to increase beyond a preset acceptable limit.

The significance of a model term is expressed using its corresponding Error Reduction Ratio (ERR). This value is calculated for every term as part of the NARMAX estimation process. The ERR is an indication of the reduction in the model’s prediction error that occurs when the model term considered is introduced in the model. This reduction is expressed in proportion to the maximum error (a constant) that results by removing all the terms from the model. The value of the ERR is therefore proportional to the significance of the term it corresponds to.

The initial structure of the NARMAX polynomial is determined by its input dimension $d$, the input, output and error (if present) regression orders $N_u$, $N_y$ and $N_e$ respectively and the degree $l$ of the polynomial. The number of terms in the initial model can be very large depending on these variables however only a few of them will be left in the final model as most of them will have negligible or no effect on the error (very small ERR).

After the iterative estimation process ends the resulting NARMAX model is tested using a validation data set. The true performance of the model is assessed in this test.

B. Experimental setup

For the purposes of this work an indoors setup was used (see figure 1) comprised of a wooden platform (the arena) of dimensions $1.6 \times 1.0$ m, one overhead camera for target tracking and a seismic logging system comprised of six uniaxial accelerometers connected to a data acquisition and monitoring system. The seismic system is capable of recording seismic vibrations in the range of 0-100 Hz. It samples these vibrations at a frequency of 200 Hz and 24-bit resolution from each individual sensor.

In this setup, the six accelerometer sensors were used in two groups of three and these groups were placed in two diagonally opposite corners of the arena (see figure 2). Each sensor measures acceleration in one direction and so in order to be able to measure as fully as possible the effects of seismic waves, the three sensors of each group were oriented and fixed on the arena so as to provide measurements along each of three orthogonal axes.

The overhead camera was connected to a laptop PC running a simple colour-based multiple-target detection algorithm. This setup is able to track multiple targets at a rate

2 UNIBRAIN Fire-i firewire camera fitted with 1.9mm wide angle lens.
3 CMG-5 Uniaxial Accelerometers.
4 CMG-DM24S12AMS data acquisition and monitoring system.
of up to 30 Hz. An example of a camera image is shown in figure 3. This image suffers from extreme barrel distortion due to the wide angle lens used. This distortion is removed post-process for all data obtained from the vision tracking system. The error of the target tracking system is less or equal to 1cm.

The robot used in these experiments was the Mark III\(^5\) (pictured in figure 4). This is a small robot (10x8x9cm) utilizing two frontal infra-red proximity sensors and three infra-red encoder sensors under its front scoop for ground


During the experiments presented here the robot was programmed to perform a “wall-bounce” behavior in the arena using only its proximity sensors. While executing this behavior the robot would move forward on a curve until it detected an obstacle in which case it would turn on the spot and away from the obstacle. Some randomness was introduced in the behavior in order for the robot to visit as much of the platform surface as possible in a given duration of time. The speed of the robot during this behavior was set to approximately 15cm/s.

C. Experiment 1

In the first experiment one robot was placed on the arena and allowed to move for approximately 30 minutes. During this time the seismic activity on the arena and the position of the robot were recorded. The seismic and position data were subsequently corresponded using absolute time. Finally, the data was down-sampled to 1Hz. This was done because we are only interested in the amplitude of the seismic signals and not in any phase relationship at this time. The
full resolution of the seismic signals would therefore add unnecessary computational load to our procedure.

The trajectory described by the robot during this experiment is shown in figure 5. The values of the x and y-coordinates (with respect to the overhead camera image origin) are plotted together in figure 6 for the first 60 seconds of the robot’s trajectory (this part of the trajectory is indicated by the bold line in figure 5). Figures 7 and 8 show the seismic activity recorded from each of the two groups of sensors respectively during the first 60 seconds. Note that only the magnitude (and not the direction) of the seismic sensors was considered for this experiment.

The model is a system of two NARMAX polynomials that were produced using the methodology described in section II-A. One polynomial was found in order to model the x-coordinate and the other to model the y-coordinate of the robot’s location. Each polynomial was of degree l=2 and

\[ x_p(n) = +1.2648869816 -0.0047797787 \cdot u1(n) -0.0049652524 \cdot u1(n-1) -0.0024675856 \cdot u1(n-2) -0.0025336538 \cdot u1(n-3) -0.0027836239 \cdot u2(n) +0.0016061752 \cdot u2(n-1) -0.0001829564 \cdot u2(n-2) -0.0001603919 \cdot u2(n-3) +0.0002805444 \cdot u3(n) +0.0000055609 \cdot u3(n-1) +0.0000143261 \cdot u3(n-2) -0.0000099753 \cdot u3(n-3) +0.00002678556 \cdot u4(n) +0.0000019760 \cdot u4(n-1) + \cdots \]

Due to space limitations, only some of the terms of each polynomial are shown above.
\[ y_9(n) = \] 
\[ +0.0000000757 \ast u1(n - 1) \ast u5(n) \] 
\[ -0.0000001010 \ast u1(n - 2) \ast u5(n - 1) \] 
\[ -0.0000001170 \ast u1(n - 3) \ast u5(n - 3) \] 
\[ +0.0000000710 \ast u2(n) \ast u5(n - 1) \] 
\[ +0.0000000403 \ast u2(n) \ast u6(n - 1) \] 
\[ -0.0000000466 \ast u2(n - 1) \ast u6(n - 1) \] 
\[ +0.0000000423 \ast u2(n - 2) \ast u5(n) \] 
\[ +0.0000000704 \ast u2(n - 2) \ast u5(n - 2) \] 
\[ +0.0000000758 \ast u2(n - 3) \ast u5(n - 3) \] 
\[ +0.0000000121 \ast u3(n) \ast u5(n - 1) \] 
\[ +0.0000000158 \ast u3(n - 1) \ast u5(n - 2) \] 
\[ +0.0000000149 \ast u4(n - 1) \ast u6(n - 2) \] 
\[ +0.0000000183 \ast u4(n) \ast u5(n - 1) \] 
\[ +0.0000000246 \ast u4(n) \ast u5(n - 2) \] 
\[ +0.0000000200 \ast u5(n) \ast u5(n - 3) \] 
\[ +0.0000000135 \ast u5(n - 2) \ast u6(n) \] 
\[ -0.0000000746 \ast u5(n) \ast u5(n - 1) \] 
\[ -0.0000000135 \ast u5(n - 2) \ast u6(n) \] 
\[ -0.0000000217 \ast u5(n - 2) \ast u6(n - 3) \] 
\[ -0.0000000246 \ast u5(n - 3) \ast u6(n - 1) \] 
\[ +0.0000000183 \ast u5(n - 3) \ast u6(n - 2) \] 
\[ +0.0000000070 \ast u6(n - 2) \ast u6(n - 3) \] 

The two polynomials above were tested on the validation data set. Figure 9 shows the value of the x-coordinate in the validation data set against the model-predicted value of the coordinate for the first 5 minutes (300 seconds) of validation data. Similarly figure 10 shows the y-coordinate against its model-predicted value for 2.5 minutes (150 seconds). Less of the validation data are shown in the latter figure for clarity reasons – the y-coordinate varies faster as the robot is physically more constrained along the y-axis (see figure 6).

In order to produce a more meaningful error measure of the model, the Euclidean distance between the actual position of the robot (in the validation data set) and the model predicted location (obtained from the model – the system of the two NARMAX polynomials) was found. The histogram of position errors is shown in figure 11. The mean error is 0.21m and the standard deviation is 0.13m.

This error is significantly large when compared to the scale of the setup used (the size of the robot and its environment). It is however interesting at this point to see that the modeling methodology discriminates the important seismic signal components that contribute to the calculation of the robot’s position. We believe that this error will stay the same if we follow the same procedure in a larger robot arena and perhaps a larger robot. In such a case, even though the error will stay the same, its significance will decrease.

D. Experiment 2

In order to see how robust our modeling method is against input that is uncorrelated with the output but of significant magnitude compared with the correlated input we conducted a second experiment.

This experiment was conducted with the same setup but now two identical Mark III robots (A and B) were placed in the arena and allowed to execute the same “wall bounce”
behavior explained earlier. The robots were thus avoiding collisions with the boundaries of the arena and with each other. The robots were allowed to run in the arena for approximately 30min while their positions were recorded using the overhead camera vision system. The seismic activity on the arena was also recorded during this time using the same setup and configuration as in the previous experiment.

The collected data location and seismic data were aligned and down-sampled again to 1Hz. The trajectories of the two robots are shown in figure 12.

Fig. 12. The trajectories described by the two robots after approximately 30min. The trajectory of robot A is shown in black colour and that of robot B is shown in grey colour.

We were now interested to see whether we could still obtain a model of the x and y-coordinate of robot A using the seismic sensor signals that now contained additional and equally intense seismic information produced by the robot B.

Following the same methodology as described in section II-C and using the first half of the collected data (estimation set), two NARMAX polynomials (one for the x and the other for the y-coordinate of robot A) were obtained with the same specifications as before (i.e. degree l=2 and input order \( N_u=3 \)).

The performance of the obtained model was subsequently tested on the remaining data (validation set) in the same way as before. The position error for this model is summarized by the error histogram shown in figure 13. The mean error in this case was 0.38m and the standard deviation is 0.23m.

Fig. 13. The histogram of position errors for robot A (Euclidean distance between actual and model-predicted robot position in the validation data). The mean error is 0.38m and the standard deviation is 0.23m.

The error has now significantly increased and the prediction is almost as bad as a random prediction of the robot’s position. This was expected as this situation closely resembles the “Cocktail Party” problem. There is nothing in the seismic data to distinguish the effect of robot A from that of robot B.

A further model was obtained in order to see whether knowledge of the position of robot B would improve the prediction of the position of robot A. In this case therefore, as well as the seismic signals from the six accelerometers the position of robot B (x and y-coordinates) was also provided as an input to the model estimation process. Again the degree \( l \) and input order \( N_u \) were 2 and 3 respectively. The position error distribution for this model is shown in figure 14. In this case the mean error is 0.34m and the standard deviation is 0.19m. This error is significantly different than that shown in figure 13 (Mann-Whitney U-test, \( p<0.01 \)).

Fig. 14. The histogram of position errors for robot A using the model that calculates the position of robot A as a function of the seismic sensor signals and the position of robot B. The mean error is 0.34m and the standard deviation is 0.19m.

This last experiment has looked at the extreme case where the two seismic sources are completely identical and has
shown that there can be an improvement to the prediction of the robot’s position if some information is provided about the source of the uncorrelated input.

III. SUMMARY, CONCLUSIONS AND FUTURE WORK

Using an indoors setup for a pilot study, we have presented a numerical method for estimating the position of a moving robot using the intensity of the seismic waves that it causes on the arena upon which it moves. The mean prediction error of the model obtained is 21 cm (standard deviation: 13 cm) in an arena with dimensions 1.6 x 1.0 m. This is a much worse error than existing indoor tracking methods, however, the purpose of the method proposed here is not to track indoor robots and certainly not in such a small scale. We believe that seismic waves can provide us with a lot more information that will enable more precise localization of the robot in larger environments. This information was discarded here (by subsampling the data collected from the seismic sensors) because the time resolution of our system could not allow us to measure them.

As well as a single robot in a seismically “quiet” environment, we have tried to use the same method in order to estimate the location of a robot when another similar robot was moving in the same environment. The purpose of the second robot was to cause seismic signals of similar intensity but largely uncorrelated to those produced by the first (target) robot. As expected, the position prediction error increased significantly in this case compared to the first experiment (mean: 38 cm, standard deviation: 23 cm). This error decreases slightly but significantly) when the position of the second robot is known and it is used in the estimation process.

For the same reasons discussed earlier, we believe that the accuracy of the solution to the above problem can be greatly improved if the same experiment is performed over a larger area outdoors. By using seismic wave delay information it will be possible to distinguish more than one moving vehicle. We are particularly interested in detecting potential collisions between vehicles in cases where visual or other means of locating the vehicles would not be possible. One example of such a problem is collisions in airports during severe weather conditions. Even though aircraft are equipped with quite accurate means of knowing their location there is no world wide standard system of communicating this information to other aircraft in their vicinity when on the ground. In the case of very low visibility, the control tower cannot warn fast enough of a potential collision. Such is the case for example when a pilot makes a mistake while following taxiing instructions. A system monitoring the ground seismicity of the airport could automatically provide a warning to ground traffic using existing communication channels.

The second stage of this work will involve deploying seismic sensors in an outdoor environment over a large area (about 2000 m²) to apply the methods described here in order to estimate the position of moving vehicles. By doing this we hope to be able to detect and use differences in seismic signal arrival times and the signal frequencies and magnitudes in order to estimate the position of moving vehicles more accurately and in a more realistic scenario.

REFERENCES

A Proposal of a Methodology for the Analysis of Robot-Environment Interaction through System Identification

Otar Akanyeti¹, Iñaki Rañó², Ulrich Nehmzow³ and Steve Billings⁴

Abstract—This paper presents a novel method to analyse robot-environment interaction from a mathematical point of view in order to understand and identify the underlying rules governing this interaction. The method finds its theoretical roots in robot training, system identification and dynamical systems theory.

The method is based on identifying the whole system using two models — i) the controller model which describes how the actions of the robot are related to its own perception and ii) the perception model which describes how the perception of the robot changes as a result of the actions of the robot — and then put them together into a single coupled dynamical system in order to analyse or simulate the emergent behaviour.

We tested the proposed methodology by analysing the robot-environment interaction where the desired robot behaviour was a door traversal task.

I. INTRODUCTION

The control of an autonomous mobile robot is different from systems considered in control theory, where the plant to be controlled can be clearly identified and dealt with independently from the rest of the system. In contrast, the behaviour of the robot guided by a control program cannot be isolated either from the environment or the robot hardware. Because this robot-environment interaction usually exhibits highly complex and non-linear characteristics, to date there is no established theory which can help us to understand and identify the underlying rules governing this interaction. To know what real robot behaviour will result from a specific control program, one actually has to run the program on a real robot and observe the behaviour.

However, it would be very useful if we had the formal tools which allow us to have a better understanding in robot-environment interaction so that we can generate more reliable and robust control programs, or predict the behaviour of the robot for unforeseen circumstances.

So far only few researchers have attempted to solve the problem mentioned above. In [Akanyeti et al., 2007a] [Nehmzow et al., 2007] and [Akanyeti et al., 2007b], we demonstrated that generating control programs for sensor-motor tasks — such as wall following, door traversal or route learning — using transparent polynomial models does not only produce robust controllers to achieve the desired behaviours, but also allows us to analyse the controllers from a mathematical point of view, since the produced controllers are transparent mathematical functions. Presented examples include analysing sensitivity of the robot behaviour to particular sensors [Iglesias et al., 2005] and making predictions about the behaviour of the robot for certain input signals [Akanyeti et al., 2007a].

The research mentioned above is a step towards the development of a theory of robot-environment interaction that will enable a more focused and methodical design of robot controllers. However in order to describe robot-environment interaction in a complete form, besides knowing how the controller reacts according to the perception of the robot, it is also important to be able to comprehend how the perception of the robot changes according to its actions along the desired path. In this paper, we therefore propose a method for the analysis of robot-environment interaction which also takes perception into account.

The method is based on dynamical systems theory and system identification methods and can be analysed in two stages. First we identify the whole robot-environment interaction using two main models: i) the controller model which produce the actions of the robot based on the perceived world and ii) the perception model which simulates how the perception of the robot changes as a result of its actions.

These models are obtained using system identification techniques such as ARMAX (Auto-Regressive Moving Average models with eXogenous inputs) [Eykhoff, 1974] and NARMAX (Nonlinear ARMAX) [Billings and Chen, 1998], where they produce linear or nonlinear polynomial functions that model the relationship between the robot’s sensor perception and motor response. Once the models are identified, we use established techniques from dynamical systems theory to analyse the obtained models — for instance to check the stability of the system or to predict the behaviour of the robot in unforeseen circumstances.

II. METHODOLOGY FOR THE ANALYSIS OF ROBOT-ENVIRONMENT INTERACTION

A. Robot-Environment Interaction

The robot interacts with the environment in two different ways; i) it acquires information from the environment through its sensors to provide the necessary input signals to the controller and ii) it performs actions in the environment in order to achieve the desired task. Here sensing and acting are coupled dynamically and can not be analyzed independently since the perception of the robot influences the actions of the robot and the actions of the robot influence how the robot perceives the environment — where as robot moves along
the desired trajectory, the perception of the robot changes according to the new position of the robot (figure 1).

![Diagram](image)

**Fig. 1.** The robot-environment interaction can be decomposed into two phases: i) action, and ii) perception. Here sensing and acting are coupled dynamically and cannot be analyzed independently since the perception of the robot influences the actions of the robot and the actions of the robot influence how the robot perceives the environment.

In general, this interaction exhibits complex and unpredictable characteristics and it is difficult to identify the whole system using a generic method. However, the analysis can be simplified by the following assumptions:

1) The controller of the robot is reactive where the output of the controller does not depend on the internal states of the controller, but only on the current input signals provided to the controller.

2) The robot operates in a controlled environment with no other external factors which influence the environment — for instance another autonomous robot situated in the same environment. Therefore, we can assume that, when the robot performs actions in the environment, the change in the perception of the robot will be only dependent on the actions of the robot.

Under these assumptions the whole robot-environment interaction can be described in a complete form using two models: i) the controller model which computes the desired motor responses of the robot according to its perception, and ii) the perception model which emulates how the perception of the robot is affected from its own actions.

In this paper, we obtain these models using system identification techniques such as ARMAX (Auto-Regressive Moving Average models with eXogenous inputs) [Eykhoff, 1974] and NARMAX (Nonlinear ARMAX) [Billings and Chen, 1998], where they produce linear or nonlinear polynomial functions that model the relationship between the robot’s sensor perception and motor response. The main advantage of using polynomial NARMAX models compared to other non-linear mapping techniques is that the obtained models are transparent mathematical functions and they can inform us about the underlying rules which govern how the robot interacts with the environment in order achieve the desired behaviour.

**B. The NARMAX System Identification Methodology**

Having identified the models set needed in order to describe robot-environment interaction in closed form, our next step is to obtain these expressions using established system identification methods, where both models can be approximated using inputs and output from the controller.

Roughly, the procedure to identify a system can be split into the following steps: i) data recording from the system to be identified, ii) model structure selection, iii) model computation guided by the data, and finally, iv) model validation. The last two steps are performed iteratively using two sets of collected data: (a) the estimation and (b) the validation data set. Usually a single data set collected in one long session is split in half and used for this purpose.

The data collected for identification purposes must be rich enough to contain all the relevant information which describes the system under investigation. In our case this means that — for both controller and perception models — the training data sets should contain input-output pairs the environment can produce for the desired robot-environment interaction as much as possible. On the other hand, there is no point in exploring all the possible input-output set for the model, since the perception and action space of the robot is bounded with the environment it is operating in.

Once the data from the systems is properly collected, the next step is to select an appropriate model. One possible way of obtaining models for the above expressions is to approximate them as polynomials. It is well known that polynomials with a sufficient order can approximate any continuous function in an effective way within an arbitrary degree of accuracy [Weierstrass, 1885]. Therefore, assuming that the functions describing the controller and perception models are continuous we can use polynomial approximations to identify them. Since we made no assumption on the systems’ linearity a general enough non linear model will be used, specifically a polynomial NARMAX model.

As part of the model structure selection we have to choose a set of appropriate inputs \( u(n) \) and output \( y(n) \). The general rule in choosing suitable inputs and outputs is that there must be a causal relationship between the input signals and the output response. The form of an general polynomial NARMAX model is

\[
y(n) = \sum_{\alpha \in I} p_{\alpha} \cdot y(n-1)^{\alpha_1} \cdot \ldots \cdot y(n-n_y)^{\alpha_{n_y}}
\]

\[
\cdot u_1(n)^{\alpha_{n_u+1}} \cdot \ldots \cdot u_d(n)^{\alpha_{n_u+d}}
\]

\[
\cdot u_1(n-1)^{\alpha_{n_u+d+1}} \cdot \ldots \cdot u_d(n-1)^{\alpha_{n_u+2d}}
\]

\[
\cdot u_1(n-n_u)^{\alpha_{n_u+d-n_u+1}} \cdot \ldots \cdot u_d(n-n_u)^{\alpha_{n_u+d-n_u}}
\]

where \( d \) is the dimension of the input space, \( n_u \) and \( n_y \) are respectively the lags in the input and output, \( p_{\alpha} \) is the coefficient of the corresponding monomial, and \( \alpha = (\alpha_1, \ldots, \alpha_{n_u+d+n_y}) \) is the index array of positive integers taking values on the possible index set \( I \), such that \( 0 \leq \sum \alpha_i \leq l \), where \( l \) is the order of the polynomial. Therefore, as part of the model structure selection, values have to be chosen for \( l, n_u \) and \( n_y \).

Once the structure has been selected the model parameters, the monomial coefficients, must be computed using the
sampled data. The orthogonal parameter estimation algorithm (OPE) [Korenberg et al., 1988] is a technique which allows each parameter in a NARMAX model to be estimated sequentially and independently of the other parameters in the model. The main advantage of this technique, compared to classical least square learning algorithms, is that it provides an indication of the contribution that each term in the model makes to the desired output variance and this assists the user to detect the structure of the system under investigation yielding at the same time to a parsimonious system model [Chen and Billings, 1989]. Having parsimonious models is vital in system identification since increasing the order of the dynamic terms \((n_l, n_u)\) and the order of the polynomial expansion \((l)\) to achieve the desired prediction accuracy will in general result in an excessively complex model and possible ill-conditioned computations [Chen et al., 1990].

The Orthogonal Parameter Estimation algorithm is a very well established technique and is being widely used in many control applications. More detailed discussion about parameter estimation and model validation can be found in [Korenberg et al., 1988] [Billings and Voon, 1986] and [Billings and Chen, 1998].

III. A Case Study

As a case study we applied our methodology to analyze robot-environment interaction where the desired robot behaviour was an episodic task of “door traversal”, here each episode comprises the movement of the robot from a starting position to a final position, traversing door-like opening en route.

To do so, first a door traversal controller has been designed for the autonomous Scitos G5 mobile robot DAX (figure 2 a), such that the controller computes translational and angular velocities of the robot to make it approach to the door based only on the current laser scan. Therefore, the robot under the control of the designed controller exhibits a purely reactive behaviour in the sense of being a direct input output mapping without using any internal states.

In order to accomplish the desired behaviour, the designed controller used only the laser scanner of the robot. This sensor has a 4 m distance range, 240° angular, range and approximately 0.36° angular resolution (figure 2 b). The environment in which the experiments have been performed is a circular robotics arena of 100 m² where wooden boxes are used to simulate the door like openings.

A. Collecting training data

The control program was tested on an area around the door while all the relevant data, sensor inputs and controller outputs with a frequency of 4 Hz, was collected by starting the robot from 23 different starting positions distributed in the area A shown in figure 3. This area was selected methodically to have significant data for identifying a NARMAX model for the controller while keeping the door in the perceptual range of the robot. The door traversal algorithm drove the robot successfully through the door in all runs, controlling both translational and rotational speeds.

Using a Vicon motion tracking system with high resolution cameras which can deliver position data \((x, y, z)\) all the training trajectories were recorded. Figure 4 represents the trajectories generated by the robot under the control of the hardwired door traversal controller.

B. Identifying the Controller

This section describes the procedure followed in order to identify a hardwired door traversal controller using a polynomial NARMAX model and the Orthogonal Parameter Estimation algorithm. Following the identification procedure, once the training data was collected, we then obtained two controller models; one for the translational speed \(v\) and one for the rotational speed \(\omega\) of the robot.

a) Sensor signal preprocessing: Even though the whole laser scan is available, we decided not to use the raw sensor input as the input to our model and selected a smaller set of input variables computed from them. This does not only help reducing the dimensionality of the input and get simpler models but, as will be seen, allows us to get human-comprehensible models in terms of the input variables.
We processed the raw laser readings to extract three input variables which contain enough information to determine the linear and angular speeds of the robot to drive the robot successfully through the door-like openings. As represented in Figure 5 these inputs are:

- \( d \): The distance from the robot to the midpoint of the door where \( d \) is a positive value smaller than 4 m since the robot’s laser scanner range is limited to 4 m.
- \( \alpha \): The angle between the heading direction of the robot referenced to the center of the door. An obvious restriction on this parameter is \( -\frac{\pi}{2} < \alpha < \frac{\pi}{2} \), since outside this range the door could not be in the angular field of view of the robot. Note that the actual range for the experiments is even smaller: if the robot starts too slanted with respect to the pointing direction of the midpoint of the door, one of the sides of the opening will be very likely missed from the view.
- \( \beta \): The angle of the robot position relative to the normal direction to the door. The value of this input variable falls in the range \( (\frac{-\pi}{2}, \frac{\pi}{2}) \).

The computation of the input vector \((d, \alpha, \beta)\) to feed the NARMAX model is easy starting from the laser scan, through a simple door detection algorithm. First, the gaps generated by the two door blades are detected and the distances \( d_1 \) and \( d_2 \) are computed using simple a edge detection algorithm. Those distances, jointly with the angular distance between the two corresponding beams, are then used to compute \( d, \alpha \) and \( \beta \). Note that even though in complex environments this door detection mechanism is prone to fail, it is enough for the purpose of the current work.

b) Obtaining linear and angular speed models: With the vector \((d_k, \alpha_k, \beta_k)\) computed at each time step \( k \) from the laser scan as input and the corresponding velocity outputs from the hardwired door traversal controller \((v_k, \omega_k)\), two NARMAX models were obtained using the Orthogonal Parameter Estimation algorithm. The algorithm simplified both NARMAX models removing monomials that did not contribute to the outputs to get the following ARMAX models with no lags in the inputs and outputs (i.e. \( l = 1, N_u = 0 \) and \( N_y = 0 \)):

\[
\begin{bmatrix}
    v_k \\
    \omega_k
\end{bmatrix} = \begin{bmatrix}
    0.01 & 0 \\
    0 & 0.25
\end{bmatrix} \cdot \begin{bmatrix}
    0.25 \\
    0.25 \\
    0.5
\end{bmatrix} + \begin{bmatrix}
    d_{k-1} \\
    \alpha_{k-1} \\
    \beta_{k-1}
\end{bmatrix}
\]

where the indexes on the input reflect the fact that inputs do not affect the current output until the next sampling period.

c) Model validation: Having obtained the two polynomials to model the relationship between the input vector \((d_k, \alpha_k, \beta_k)\) and the angular and linear velocities of the robot, we tested the models on the robot in the real environment by starting the robot from 23 different positions. The results showed that the obtained models successfully drove the robot through the door without bumping into the walls (Figure 6).

C. The Perception Model

Once we identified the hardwired door traversal controller using transparent polynomial models, the next step was to obtain perception models which describe how the input vector \((d_k, \alpha_k, \beta_k)\) changes according to the actions of the robot at each time stamp \( k \).
To do so we made the assumption that the current state of the environment which can be perceived by the robot \((d_k, \alpha_k, \beta_k)\) can be computed based on the previous environment state vector \((d_{k-1}, \alpha_{k-1}, \beta_{k-1})\) and the previous velocity commands of the robot \((v_{k-1}, \omega_{k-1})\).

We therefore trained three polynomial NARMAX models to estimate the controller inputs \((d_k, \alpha_k, \beta_k)\) based on the previous input values \((d_{k-1}, \alpha_{k-1}, \beta_{k-1})\) and the previous action commands \((v_{k-1}, \omega_{k-1})\). The obtained perception models were ARMAX models with no lag in the inputs and the outputs (i.e. \(I = 1, N_u = 0\) and \(N_y = 0\)):

\[
\begin{bmatrix}
    d_k \\
    \alpha_k \\
    \beta_k
\end{bmatrix} =
\begin{bmatrix}
    0.21 \\
    0 \\
    0
\end{bmatrix} +
\begin{bmatrix}
    -2.25 & 0 & 1.05 & 0 & 0 \\
    0 & 0.3 & 1.03 & 0 \\
    0 & 0 & 0 & -0.1 & 0.96
\end{bmatrix}
\begin{bmatrix}
    v_{k-1} \\
    \omega_{k-1} \\
    d_{k-1} \\
    \alpha_{k-1} \\
    \beta_{k-1}
\end{bmatrix} +
\begin{bmatrix}
    0 \\
    0 \\
    0
\end{bmatrix}
\]

The performance of the obtained perception models were tested using validation data set obtained during the training session. Figure 7 presents the real and predicted values of \(d, \alpha\) and \(\beta\) along the whole set of testing trajectories chained one after the other. While computing \(d, \alpha\) and \(\beta\), the estimation models were allowed to get real sensor input only at the starting position of each trajectory, then predicting the rest using the real motor commands and the previously estimated input values. Jumps at the beginning of the sequences are the effect of chaining the data of different trajectories.

To evaluate the performance of the models we computed the mean absolute errors and the Spearman rank correlation coefficients between the real and predicted values for \(d, \beta\) and \(\alpha\). The results are given in Table I, indicating quantitatively that the estimates of the prediction models closely match with the real values of \(d, \beta\) and \(\alpha\).

<table>
<thead>
<tr>
<th>Mean Absolute Error</th>
<th>Spearman Rank Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d) (0.081 \pm 0.010) (m)</td>
<td>(0.951, (p &lt; 5%))</td>
</tr>
<tr>
<td>(\beta) (0.117 \pm 0.003) (rad)</td>
<td>(0.757, (p &lt; 5%))</td>
</tr>
<tr>
<td>(\alpha) (0.095 \pm 0.003) (rad)</td>
<td>(0.963, (p &lt; 5%))</td>
</tr>
</tbody>
</table>

**Table I**

*We evaluated the performance of the perception models by computing the mean absolute error and the Spearman rank correlation coefficients between the real and predicted values for \(d, \beta\) and \(\alpha\) respectively.*

\(d)\) **Robot traversing the door without using real sensory inputs:** Having obtained both perception and controller models, we then validated the performance of the models by testing them on the real robot in a real door traversal application. We used the controller model to drive the robot but rather than using real sensory inputs \((d, \alpha, \beta)\), the estimated values computed by the perception models were fed to the controller. The robot was allowed to get real sensor input only at the starting position, and then the whole set of motion commands was generated by the controller using the estimate of \(d, \alpha\) and \(\beta\) at each step.

Figure 8 shows the resulting trajectories of the robot. The
results show the accuracy of the models, since the robot is able to traverse the door successfully even with a single snapshot of the environment at the starting position.

It is obvious that the performance of the robot is not as good as when the robot uses real sensor signals to compute the speed outputs where the deviation of the robot from the mid-point of the door was approximately 2.5cm worse.

On the other hand this approach can be very useful under unexpected circumstances; for instance if at some step of the door traversal behaviour, the door detection algorithm fails to detect the door accurately or if laser sensor of the robot breaks down for some reason. It can also be used for abnormality detection since the robot has an expectation about how its perception changes according to its actions.

A\[A\]

D. Stability Analysis of the whole System

This section demonstrates how having transparent models can be very useful in analysing the models from a mathematical point of view. Note that the obtained perception and controller models are coupled and they cannot be analysed separately. We therefore write a joint dynamical model for the whole system in a matrix form which will be an inhomogeneous linear difference equation:

$$x_k = b + A \cdot x_{k-1}$$

(2)

where \(x_k\) is the whole state vector, \(X_k = [v_k, \omega_k, d_k, \alpha_k, \beta_k]^T\), \(A\) is a 5 \times 5 matrix and \(b\) is a vector. The identified values of \(A\) and \(b\) are:

$$A = \begin{bmatrix} 0 & 0 & 0.03 & 0 & 0 \\ 0 & 0 & 0 & -0.25 & 0.5 \\ -2.25 & 0 & 1.05 & 0 & 0 \\ 0 & 0.3 & 0 & 1.03 & 0 \\ 0 & 0 & 0 & -0.1 & 0.96 \end{bmatrix} \quad b = \begin{bmatrix} 0.01 \\ 0 \\ 0.21 \\ 0 \\ 0 \end{bmatrix}$$

The solution to this system is simply

$$x_k = A^k \cdot x_0 + (I - A)^{-1} (I - A^k) b$$

where \(x_0\) is the state at time step \(k = 0\), and \(I\) is the 5 \times 5 identity matrix.

The stability of this kind of linear systems depends on the eigenvalues \(\lambda_i\) of the matrix \(A\). If \(|\lambda_i| < 1\) for all \(i\) the corresponding system will be stable and the matrix \(A^k\) will go to ‘0’ as \(k\) goes to infinity. In our case all the eigenvalues (0.96 \pm 0.04i, 0.08, 0.07 and 0.98) have an absolute value smaller than ‘1’, and therefore the complete system is stable. Moreover, it has a non zero stability point because of the inhomogeneous term located at \(\lim_{k \to \infty} x_k = (I - A)^{-1} b\). The system will reach the state defined by the vector \(x_\infty = [0.05, -0.01, -2.04, -0.01, 0]^T\).

This final state is not a stable point in the sense that the robot would reach a point in the environment and stop there. Even more, the attractor point is 2 m away from the door and non-zero constant linear velocity violates the fact that the robot would stay at the stability point forever. This is clearly the result of having a linear \(v\) model, which cannot model the fact that the robot, during the training runs, stopped once it traversed the door. Even the robot in the training set never went so far, interestingly the equilibrium point of the whole linear system is behind the door. In conclusion, this behaviour attractor drives the robot to the other side of the door, however nothing ensures it will not collide with the walls.

The optimum attractor point would be the center of the door so that we can say that the robot would pass through the door successfully which also guarantees that the robot would centralize the door while crossing in every attempt.

The next step therefore is to look for the ways of modifying the obtained control models using dynamical system analysis in order to change the stability point to the center of the door so that the the controllers can guarantee that the robot would pass through the door successfully. This is a part of ongoing research.

IV. CONCLUSION AND FUTURE WORK

This work presents a dynamic systems based methodology for analysing the robot-environment interaction through NARMAX system identification techniques. The method consists of identifying the whole system using two models — i) the controller model which describes how the actions of the robot are related to its own perception and ii) the perception model which describes how the perception of the robot changes as a result of the actions of the robot — and then put them together into a single coupled dynamical system in order to analyse or simulate the emergent behaviour.

In order to validate the performance of the proposed approach, we trained autonomous mobile robot DAX to achieve a door traversal behaviour in a simple environment. Besides, generating transparent controllers driving the robot successfully in a desired way, the method also helps us
modelling how the perception of the robot changes as a result of its own actions.

As demonstrated in section III-C.0.d, using both the controller and perception models, the robot was able to traverse through a door-like opening almost blindly without using any real sensor inputs. The robot was allowed to get real sensor input only at the starting position, and then the whole set of motion commands was generated by the controller based on the estimate values of $d$, $\alpha$, and $\beta$ computed by the prediction models at each time step. Several other techniques have been applied to predict sensor inputs on mobile robotics using different learning techniques ([Hoffmann, 2007], [Ziemke et al., 2005]). The main advantage of using polynomial NARMAX models compared to other non-linear mapping techniques is that the obtained models are transparent structures which can then be used to analyze the robot-environment interaction from mathematical point of view.

e) Future Work: In the particular door traversal example, the stability analysis revealed that the stability point of the system is $2m$ behind the door. This is a clear indication that the robot is attracted to traverse to the other side of the door but further analysis is required to make sure that the obtained linear models guarantee a collision free journey. Furthermore for more complex behaviours, it is obvious that the obtained models will result in non-linear forms where the analysis of such models is not so straightforward and more sophisticated techniques are required (see [Rañó, 2007] for an example). Overall, the work already carried out and that proposed forms are part of our ongoing research to develop a theory of robot-environment interaction.

ACKNOWLEDGMENTS

We gratefully acknowledge that the RobotMODIC project is supported by the Engineering and Physical Sciences Research Council under grant GR/S30955/01. This work was partially supported by the Spanish Ministry of Science and Education under grant JC2007-00293.

REFERENCES


Improving 3D Scan Registration for SLAM with Clustering and Deterministic Annealing

Timo Röhling Dirk Schulz
Research Institute for Applied Sciences (FGAN), Wachtberg, Germany
{roehling,schulz}@fgan.de

Abstract—In the context of simultaneous localisation and map building, the performance of scan registration algorithms such as Hillclimbing and Iterative Closest Point can be substantially improved by a clustering of scan points combined with a deterministic annealing scheme. The former permits control over the level of detail that is used for the scan registration, while the latter uses this level of detail as the temperature term in an annealing-based optimisation. This way, the method exploits the fact that real-world environments contain objects on different scales, which are preferably matched in a top-down manner. Experiments show that this new approach can cope with odometry errors that are up to ten times larger, without the need to pre-align data with other feature-extracting algorithms. As most current grid-based map building algorithms employ scan registration as a means of correlating sensory information, this result helps to increase the robustness of map building systems in general.

I. INTRODUCTION

The problem of simultaneous localisation and map building has received a lot of attention over the past years, and numerous techniques have been developed that estimate the most likely map from a series of probabilistic observations. An important precondition for successful map building is the ability to establish the relative pose between two different sensor observations. In the context of 3D SLAM, this is usually achieved by registration of point clouds obtained from 3D laser range scans with Iterative Closest Point (ICP) or Hillclimbing. As these local optimisation algorithms are prone to converge towards a non-global minimum, additional measures have to be taken to ensure the correct registration of two scans.

This paper introduces an enhanced optimisation technique that employs the clustering of scan points to reduce the amount of detail in a scan and subsequently the number of non-global minima. The global minimum is reached with a non-randomised annealing method that utilises the level of detail as the temperature term. The remainder of this paper is organised as follows. Section II briefly discusses related work in this field. Section III recapitulates the basic concepts for scan registration, and section IV introduces the scan clustering method and the deterministic annealing scheme that are novel to our approach. In section V the experimental evaluation of the improved performance of the new algorithm is presented, and section VI concludes with a brief summary and a preview of future work in this field.

II. RELATED WORK

Laser-based SLAM algorithms use point-based registration to derive relative poses from laser scans, which can be used as filter input for computing the actual trajectory [2], [3]. This is in contrast to landmark-based SLAM, which tracks identifiable landmarks in the sensor data; the extraction of such landmarks from laser scans has proved to be very challenging [4]. One of the most advanced systems for laser-based 3D SLAM has been developed by Surmann et al [5], [6]. They present a fast implementation of Iterative Closest Point using approximate octrees. Their approach requires a pre-alignment of the scan octrees which has some similarities to the annealing technique we propose in this paper. Pre-alignment has a long standing tradition in ICP-based approaches. For example Lu and Milios [7] apply a rotation search algorithm as preconditioner that pre-aligns 2D scans sufficiently well to achieve global ICP convergence. However, their technique is difficult to translate into a 3D approach. Another method related to our approach is by Luck and Hoff [8], who combine ICP with simulated annealing to provide the ICP with better restart locations, but they do not exploit the hierarchical structure in the 3D data itself to guide the search. Finally, Stamos and Leordeanu [9] use a clustering of line segments to facilitate the detection of overlapping scan regions, in order to improve the scan registration.

Deterministic annealing as a non-randomised annealing technique, which is employed by our approach, is predominantly used for soft-clustering problems, compression and speech recognition [10]. It has originally been introduced by Qiu, Varley and Terrell [11].

Most researchers who examine 3D SLAM and its applications today, focus on the computation of the robot’s trajectory and assume that a working scan registration implementation is available. Newman and Cole [12], [13] track the residual registration errors with Kalman filters, Grisetti et al [14] developed a GraphSLAM-related mapping technique which is
the basis for the mapper implementation that has been used for some of the results presented in this paper. The main contribution of this paper will be the improvement of scan registration as prerequisite for successful map building.

III. SCAN REGISTRATION

In this section, we briefly introduce the mathematical concepts of 3D scan registration, mainly following the notation of [15].

A. Terminology

Unless mentioned otherwise, the variables \( p \) and \( q \) denote non-oriented points in \( \mathbb{R}^3 \). An oriented relative pose \( \delta \) designates a translation vector \( x_δ \in \mathbb{R}^3 \) and a rotation \( ρ_δ \) for a unit quaternion \( Q \) to \( \mathbb{H} \). Pose transformations are defined as in Lu and Milios [15], in particular

\[
\delta \oplus p := ρ_δ(p) + x_δ,
\]

which transforms a relative position \( p \) with respect to a given pose \( \delta \) (see figure 1).

B. Problem description

Let \( S_i \subset \mathbb{R}^3 \) and \( S_j \subset \mathbb{R}^3 \) denote two 3D scans in a local frame of reference and \( δ_{ij} \) the relative position and orientation of \( S_i \) with respect to \( S_j \). Assuming that there are two non-empty subsets \( F_i \subset S_i \) and \( F_j \subset S_j \) of features that overlap, there is a mapping \( π_{ij} \) that assigns each scan point \( p \in F_i \) to its counterpart in \( F_j \), so that

\[
δ_{ij} \oplus p = π_{ij}(p).
\]

In practice, the robot’s odometry provides only a coarse estimate \( δ_{ij} \) which does not satisfy the above equation. The introduced registration error can be quantified as

\[
ε_{ij}(δ) = \sum_{p \in F_i} ||δ \oplus p - π_{ij}(p)||^2,
\]

which is sought to be minimised. For known \( F_i \) and \( π_{ij} \), closed-form solutions exist [16] that optimise \( δ_{ij} \) with respect to \( ε_{ij} \).

However, neither the subsets \( F_i \) and \( F_j \) nor the correspondence mapping \( π_{ij} \) is known in advance. In fact, a bijective mapping as asserted by equation 2 may not even exist due to the type of range sensors usually employed (see figure 2). Thus, \( π_{ij} \) is approximated and the optimisation process is iterated until \( \tilde{π}_{ij} \) and \( \tilde{δ}_{ij} \) converge towards \( π_{ij} \) and \( δ_{ij} \), respectively, usually by employing a standard algorithm such as Hillclimbing or ICP.

As is shown by Rusinkiewicz and Levoy [17], the nearest neighbour heuristic

\[
\tilde{π}_{ij}(δ, p) = \arg\min_{q \in S_j} ||δ \oplus p - q||
\]

for \( p \in S_i \) proves to be one of the most robust approximations. Unfortunately, the described technique is quite sensitive to the initial estimate of \( δ_{ij} \), so that the scan registration may fail to converge even for moderate estimation errors. This is due to the complex structure of the error function \( ε_{ij} \), which tends to have many local minima when applied to real-world sensor data.

In order to enhance the convergence radius, the influence of local minima has to be mitigated. This can be achieved either with a smoothed error function, e.g. by applying a convolution filter, or with an improved initial pose estimate. However, the former decreases the accuracy of the scan registration considerably, and the latter requires at least an approximate solution to the scan registration problem itself. Therefore, this paper proposes a recursive solution based on deterministic annealing that uses a cluster-based scan registration as substitute for pre-alignment to improve the robustness of the optimisation.

IV. CLUSTERING AND ANNEALING

Laser scans of outdoor environments are inherently hierarchical. Large-scale entities such as buildings and open spaces contain smaller objects such as cars or trees. Depending on the scan resolution, this hierarchy may well extend down to leaves or wall bricks. Obviously, it does not make sense to match these fine structures if the estimated position is still off by a dozen metres. This section introduces the necessary data structures and algorithms to deal with different levels of details during scan registration.
A. Cluster definition

Let \( S_i \subset \mathbb{R}^3 \) be a 3D scan. A cluster \( c_p \subset S_i \) is a grouping of scan points which lie close together, i.e.

\[
\forall p, q \in c_p : \| p - q \| \leq 2r.
\]  

(5)

A cluster set \( C_{r,i} \) for a 3D scan \( S_i \) is a minimal set of non-empty clusters \( \{c_{r,1}, \ldots, c_{r,i}\} \) that assigns each point \( p \in S_i \) to a unique cluster, i.e.

\[
\forall p \in S_i \exists c \in C_{r,i} : p \in c
\]

(6)

and

\[
\forall c, c' \in C_{r,i} : c \cap c' = \emptyset \vee c = c'.
\]

(7)

Clusters may consist of a single point, thus \( S_i \) itself can be interpreted as a cluster set \( C_{0,i} \).

B. Scan clustering

Let \( r \) be the cluster radius and \( C_{r,i}, C_{r,j} \) cluster sets that are derived from the scans \( S_i, S_j \) respectively, and let \( c_i, c_j \) be two clusters from \( C_{r,i}, C_{r,j} \). Assuming that the clusters can be represented as Gaussian distributions with means \( \mu_i, \mu_j \) and standard deviation \( \sigma \), the likelihood of both clusters being equivalent is

\[
P(c_i \equiv c_j) = \eta \exp \left( -\frac{\| \mu_i - \mu_j \|^2}{2\sigma^2} \right).
\]

(8)

with \( \eta \) as normalisation factor. The value of \( \sigma \) is chosen so that a sampling of the distribution has 99 per cent of its samples lie within the radius \( 3\sigma = 2r \), i.e. \( \sigma = \frac{2}{3}r \). Omitting \( \eta \), switching to log-likelihoods, and taking the square root yields the distance function

\[
d_r(c_i, c_j) = \frac{3}{2\sqrt{2r}} \| \mu_i - \mu_j \|.
\]

(9)

The function \( d_r \) allows clusters to be interpreted as simple points. Therefore, the mapping \( \pi_{r,i,j} \) between two scans can be adapted to cluster sets by

\[
\tilde{\pi}_{r,i,j}(\delta, c_i, c_j) = \arg\min_{c_j \in C_{r,j}} \{ d_r(\delta + c_i, c_j) \}
\]

(10)

As outliers may occur, especially in situations where no bijective mapping exists (figure 3), the distance function \( d_r \) is discounted exponentially, yielding a score function

\[
s_{r,i,j}(\delta) = \sum_{c_i \in C_{r,i}} \exp \left( -d_r^2(\delta + c_i, \tilde{\pi}_{r,i,j}(\delta, c_i)) \right)
\]

(11)

that is to be maximised.

The nearest neighbour computation required for \( \tilde{\pi}_{r,i,j} \) proves to be one major bottleneck in the calculation of equation 11, apart from the generation of \( C_{r,i} \) itself. For both problems, the octree data structure proves to be very efficient.

An octree is the \( \mathbb{R}^3 \)-equivalent of a binary tree, where the region of each inner node is subdivided into eight octants that form the child nodes, and each leaf node contains at most one data point (see figure 4). As

![Fig. 3. The rightmost three points of \( S_i \) have no valid association in \( S_j \), leading to an incorrect approximation \( \tilde{\pi}_{r,j} \).](image)

![Fig. 4. The illustrative quadtree in this figure as well as the octree belong to the class of \( 2^n \)-trees that are generalised binary trees for the \( \mathbb{R}^n \).](image)

the tree depth adapts according to the density of the underlying point set, octrees are particularly efficient for 3D scans, as these surface scans usually have a highly heterogeneous density. Therefore, an octree is generally not balanced, but its depth is limited by

\[
\log_2 \frac{m}{r_{map}},
\]

(12)

where \( m \) is the diameter of the mapped area, and \( r_{map} \) is the desired resolution of the map.

The hierarchical structure of the octrees allows nearest neighbour queries to be completed in logarithmic time, and can be extended to permit the fast approximation of cluster sets \( C_{r,i} \). For this, each node \( K \) is assigned a tuple \((n_K, s_K)\) of the number of data points \( n_K \) located in its region and the sum \( s_K \) of their coordinate vectors. These values can be computed recursively, as the tuple \((n_L, s_L)\) of each empty leaf node \( L \) is initialised with

\[
n_L = 0, \quad s_L = 0,
\]

(13)

and the tuple of each leaf node \( L \) containing a data point \( p_L \) is initialised with

\[
n_L = 1, \quad s_L = p_L,
\]

(14)

and the tuple of each inner node \( K \) is calculated by

\[
n_K = \sum_{K' \in \text{children}(K)} n_{K'}, \quad s_K = \sum_{K' \in \text{children}(K)} s_{K'}.
\]

(15)

In order to enumerate a cluster set \( C_{r,i} \) for a fixed \( r \), the octree is traversed until a leaf node is reached.
or the edge length of the node region falls below $2r$. Then, all remaining child nodes of this node $K$ are skipped, a cluster represented by the mean point

$$
\mu_K = \frac{1}{n_K} s_K
$$

(16)
is added to the cluster set, and the traversal continues. The resulting set is a rather crude clustering of $S_i$, but has proved sufficiently exact and most of all quick to compute.

As can easily be seen from the described method of construction, the octree for a cluster set $C_{r,i}$ is a subset of the octree of $S_i$, more precisely a $S_i$-octree that is pruned according to the required level of detail. Thus, it is not necessary to compute distinct octrees for each cluster set; the nearest neighbour queries in the $S_i$-octree must just terminate at the highest tree level with a region diameter below $2r$ and use the corresponding $\mu_K$ as data point.

For growing values of $r$, the scan data becomes increasingly distorted. However, the mean point $c_K$ of a node $K$ must be within the node’s region. If the region edge length is greater than $2r$, the node must be a leaf node with a single data point, thus no distortion is possible. If multiple scan points are combined to a cluster, the region edge length is below $2r$, which means that the distance between any scan point $p_K$ and the cluster mean $\mu_K$ is limited by the region diameter. Therefore, the distortion for each scan point is bounded by the diagonal of the cube-shaped region, i.e.

$$
\|p_K - \mu_K\| < \sqrt{(2r)^2 + (2r)^2 + (2r)^2} = \sqrt{12r}.
$$

(17)

**C. Deterministic Annealing**

In order to achieve both robust convergence and high scan precision, a deterministic annealing technique is applied that reduces the cluster size $r$ gradually. The motivation for this approach stems from the codebook clustering problem which is defined as follows. Given a set

$$
X = \{x_1, \ldots, x_n\}
$$

(18)
of vectors and a codebook size $k \ll n$, find a set

$$
V = \{v_1, \ldots, v_k\}
$$

(19)
of codebook vectors that minimises the distortion

$$
\varepsilon^2 = \sum_{x \in X} \min_{v \in V} \|x - v\|^2.
$$

(20)
The classical simulated annealing optimises $\varepsilon$ with respect to the full codebook size, i.e. all $k$ codebook vectors are initialised randomly and then optimised. Local minima are avoided by a randomised walk along $\varepsilon$ that accepts a temporary deterioration of $\varepsilon$ with a probability that slowly converges towards zero. While this method is known to find the global minimum, the required time is potentially high, especially for large numbers of local minima.

In contrast, the deterministic annealing does not randomise the optimisation itself, but relies on a fuzzy probabilistic assignment $P_T(x, v)$ of input vectors to codebook vectors (hence the name soft-clustering), such that

$$
\sum_{v \in V} P_T(x, v) = 1.
$$

(21)
The distortion function then becomes

$$
\varepsilon_T^2 = \sum_{x \in X} \sum_{v \in V} P_T(x, v)\|x - v\|^2.
$$

(22)
The temperature $T$ measures the fuzziness of this assignment, with $P_0(x, v) = \delta_{x, \pi(x)}$ for a unique mapping $\pi$, and $P_T(x, v)$ uniformly distributed for $T \to \infty$. Beginning with a high temperature $T$, where the optimal codebook contains the mean of $X$ only, the influence of each input vector over the codebook is gradually confined to a smaller subset of potential codebook vectors, increasing the complexity of $\varepsilon_T$, and the size of $V$, until the required number of codebook vectors is reached. Note that while the assignments are probabilistic, the optimisation itself is strictly deterministic.

In the context of scan registration, the notion of fuzzy assignment translates to the clustering of scan points. For very large $r$, the scans are collapsed into their mean point. With progressive cooling of $r$, the clusters begin to split, until each cluster contains a single scan point. At this point, the optimisation is equivalent to a non-clustered, non-annealing version, with the crucial difference that the scans are already well aligned from previous optimisation steps.

In practise, the annealing is controlled by the exponential sequence

$$
r_k = r_0 \alpha^k
$$

(23)
with $0 < \alpha < 1$ and

$$
r_0 = \frac{1}{2} r_{\text{map}} \alpha^{-K}
$$

(24)
for a given number of iterations $K$ and the final map resolution $r_{\text{map}}$. Each iteration matches the corresponding cluster set $C_{r,i}$ against $C_{r,i-1}$. Although the earlier described clustering technique using octrees can only provide cluster sizes

$$
r = 2^n r_{\text{map}}.
$$

(25)
for $n \in \mathbb{N}$, an annealing constant $\alpha = 0.8$ has proved to be more robust for Hillclimbing, as the cluster size is also used as step size for the local optimisation.
V. EXPERIMENTS

A. Self-matching of real data

For this experiment, a single 3D scan was acquired with a laser scanner, and then matched against itself. While this test does not represent a typical scenario, as all data points can be uniquely paired and there are no issues with different points of view, the ground truth is known and the performance of the algorithm can be measured easily. Translation and rotation errors have been analysed separately.

The used optimisation algorithm was a simple Hillclimbing with step size $r$. For the non-clustering version, $r = r_{map}$ was fixed. Apart from the additional clustering and annealing, all test runs were completely identical. The used 3D scan contained a partial view of two buildings from the computer science department of the University of Bonn with a few additional structures, as can be seen in figure 5.

The translational displacements have been generated randomly in different magnitudes, then the scan registration has been performed both with and without clustering. The residual displacements are shown in figure 6. Below displacements of about a metre, scan registration with and without clustering are equivalent, although the clustering shows less fluctuations. For displacements above a metre, the simple scan registration becomes essentially useless, while the clustering can provide correct registrations up to ten metres.

Rotational displacements have been restricted to orientation errors, as this type of error is usually predominant. The scan has been rotated clockwise and counter-clockwise by varying degrees up to a quarter-turn. The residual displacements after scan registration are shown in figure 7. Generally, rotational errors can be corrected with the plain Hillclimbing up to about 15 degrees. Between 15 and 50 degrees, the correction becomes less reliable. As can be seen in the graph, e.g. with an initial displacement of 20 degrees, the success can be dependent on the direction of the rotation, indicating a strong dependency on the actual data. Above 50 degrees virtually no correction takes place. In contrast to that, the enhanced scan registration with clustering and annealing can cope even with a quarter-turn mismatch.

Furthermore, the Hillclimbing algorithm is much faster with clustering and annealing (see figure 8), as the larger step size in the initial phase leads to fewer required optimisation steps in total. However, this effect would be less apparent with a directed search, i.e. Iterative Closest Point.

B. Implementation as part of a mapping software

The scan registration algorithm has been used as part of a 3D SLAM software that is based on GraphSLAM [14], [18]. The exploration has been performed by a Pioneer 3 outdoor robot (see figure 9) in several scenarios as depicted in figures 10, 11, and 12. All maps are created from multiple scans which have been correlated by pairwise registration using our improved technique. First, we matched consecutive scans with regard to the exploration order. After these $n - 1$
registrations, we had a preliminary position estimate that helped to rule out those of the remaining $O(n^2)$ potential pairings which did not overlap. Based on our experience, we believe that most real-world scenarios can be represented by $O(n)$ constraints. The resulting network of constraints was input to the GraphSLAM algorithm to minimise the distortion, and solve the loop-closing problem.

The laser scans were recorded every five to ten metres while the robot was standing still. The market place scenario was mapped with 40 scans in a closed loop, starting next to the old town hall. The removal of the robot from the scans leaves characteristic holes in the floor plane, which are especially noticeable in figure 13. The quality of the scan registration is generally good and permits straight-forward loop closing with minor adjustments by the global relaxation. Each scan contained about 200,000 points. Most of the overall mapping time was spent on scan registration.

VI. CONCLUSION

This paper presented an improved algorithm for scan registration of 3D scans, that exploits hierarchical properties of real-world data to mitigate the effects of local minima. A fast approximative clustering of point data is achieved using an enhanced octree implementation, and a deterministic annealing technique smoothes the registration error function sufficiently to increase the convergence radius of vanilla local optimisation algorithms. The soundness of this approach has been demonstrated using a hillclimbing algorithm in a controlled setting and as part of a SLAM software that has successfully built several outdoor maps.

Potential future work includes the evaluation of other optimisation algorithms that can be improved with clustering and annealing, and faster queries for nearest neighbours in the octrees, for these queries tend to become very slow when two scans are poorly aligned.

Fig. 8. Runtime of the algorithm for different initial rotational errors. This is a single benchmark that illustrates the general tendency.

Fig. 9. The Pioneer 3 robot with 3D laser scanner that has been used to evaluate the mapping software.

Fig. 10. Partial map of the Mensa at the Pädagogische Fakultät Bonn, created from scan registration only. The mapped area is about 90 metres by 70 metres.

Fig. 11. Map created from the robot testing grounds at the FGAN.

Fig. 12. Map created from the market place of Bonn. The point of view lies above the old town hall.
Fig. 13. A large view of the map created from the market place of Bonn. The old town hall is the building left of the gap in the background.

REFERENCES


Application of Unsupervised Clustering to Complex Robot Training Tasks

Ulrich Nehmzow¹, Otar Akanyeti² and Steve Billings³

Abstract—The interest in obtaining sensor-motor competences by robot training — by expressing the desired actions of the robot in terms of its sensory perception using non-linear mapping techniques — has been growing steadily in mobile robotics applications. However if the sensor-motor task under investigation is subject to state transitions — in the sense that the causal relationship between the sensor perception of the robot and the desired motor commands exhibits different characteristics over time or space —, it may not be possible to identify the whole sensor-motor relationship in a single model using standard non-linear mapping techniques.

This paper proposes a novel method based on first using a classifier which divides the perception-action space of the robot into subspaces and then generating a separate model for each subspace. The viability of the proposed methodology has been demonstrated by teaching Scitos G5 mobile robot to achieve right wall following and complex route learning behaviours.

I. INTRODUCTION

Many sensor-motor competences in mobile robotics applications exhibit complex, non-linear characteristics and can not be treated satisfactorily using linear systems theory. Previous research has shown that artificial neural networks and polynomial NARMAX models have the ability to learn sophisticated non-linear relationships and they provide an ideal means of modelling non-linear systems ([Chen et al., 1990], [Chen and Billings, 1992], [Nehmzow, 1995]).

The common approach to generate sensor-motor couplings using non-linear mapping techniques can be summarized as follows: The programmer demonstrates the desired behaviour to the robot by driving it manually in the target environment. During this run, the sensor perception and the desired velocity commands of the robot are logged. Having obtained the training data, the sensor based control models — which link the perception of the robot to the desired motor commands to achieve the desired task — are obtained using non-linear mapping techniques ([Nehmzow et al., 2005], [Akanyeti et al., 2007a]). Usually, it is assumed that a single model would be enough to identify the whole relationship successfully.

It has been demonstrated that this approach is an efficient way of generating robot control programs and good results have been achieved in different mobile robotics applications ([Nguyen and Widrow, 1990], [Pomerleau, 1993], [Akanyeti et al., 2007b]). However there are cases where trying to represent the relationship between robot’s perception and action with a single network would fail. These cases are:

1) If the system under investigation is highly non-linear and complex, trying to represent the whole relationship in a single model would probably lead to huge models with a lot of parameters to fit. Training such models is extremely difficult and usually the obtained models exhibit poor model performances.

2) If the sensor-motor causality of the system under investigation varies with respect to time or space — in the sense that the relationship between sensor perception of the robot and the desired motor responses exhibit different characteristics at different times or at different spatial places in the environment —, then again trying to represent the whole system in a single model will often lead to poor model performance and possibly numerical and other problems.

This paper addresses these issues with a general overlook and propose a novel approach to tackle with them upto certain extent. The proposed method can be analysed in two stages. In the first stage the robot tries to learn distinctive landmarks in the environment it is operating in using standard classification techniques. Once the robot is capable of knowing it’s location using these landmarks, in the second stage, for each recognized landmark the robot tries to obtain a different perception model which links the perception of the robot to the desired motor response locally.

With this method, the desired global behaviour emerges from local behaviours of the robot. The robot divides the environment into distinctive subregions and for each region it uses a different local perception model to navigate. Each local perception model is independent from each other, therefore the robot can cope with the dynamic elements of the robot-environment interaction much easier.

II. PROBLEM FORMULATION

The more generalized form of the problem discussed above can be visualized using synthetic data: Let \( y(n) \) be a two input, noiseless linear system:

\[
y(n) = \begin{cases} 
  x_1, & n \leq 500 \\
  x_2, & n > 500 
\end{cases}
\]  

(1)

where \( n = 1 \ldots 1000 \) is a time sample and \( x_1 \) and \( x_2 \) are randomly generated numbers between \(-1\) and \(1\), \( x_1 = \text{rand}(1000,1) \) and \( x_2 = \text{rand}(1000,1) \).

When we try to model the above system using any non-linear modelling methodology (polynomial NARMAX
models, multi-layer perceptrons or RBF networks) in the form of \( y = f(x_1, x_2) \), the resultant models can not distinguish the state difference at 500 therefore they result in poor model performances. As an example, trying to model \( y(n) \) with polynomial regression method, the algorithm ends up with a linear polynomial which tries to average between the two inputs (2).

\[
y(n) = 0.5 * x_1(n) + 0.5 * x_2(n)
\]  

(2)

Obviously, the obtained polynomial model is very poor in terms of performance and can not cope with state transition at 500. On the other hand if we give the additional state information to the models as an input \( y = f(x_1, x_2, b) \) where

\[
b(n) = \begin{cases} 
0, & n \leq 500 \\
1, & n > 500 
\end{cases}
\]

(3)

then the non-linear modelling techniques are able to come up with good models for \( y(n) \). (4) shows the resultant polynomial model which is exactly the same with the original function \( y(n) \)

\[
y(n) = x_1 n - x_1(n) + b(n) + x_2(n) + b(n),
\]  

(4)

where when \( n \leq 500 \), \( b(n) = 0 \) and (4) simplifies to \( y(n) = x_1(n) \) and when \( n \geq 500 \), \( b(n) = 1 \) and (4) results in \( y(n) = x_2(n) \).

The above example indicates that if there are different states in the system under investigation (e.g. sudden jump as our linear model divided at \( n = 500 \)), it can be difficult trying to represent the whole system using a single model. Trying to fit one model over a very wide range will often lead to poor model performance and possibly numerical and other problems. Therefore it can be crucial to identify these states hidden in the system where input/output causality of the system changes and to obtain different models for each state.

In mobile robotics applications, usually we come across the state transition problem when the sensor-motor task under investigation is time dependent, highly complex or when the relationship between sensor perception of the robot and the desired motor commands is subject to change for different parts of the environment contact.

To demonstrate state transition problem, figure 1 presents a thought experiment where the relationship between the sensor perception of the robot and the desired motor responses vary along the desired trajectory. If the desired trajectory is defined as a straight line starting from point A and ending at point F we can see that sensor-motor relationship of the robot can be investigated in two different stages. Along the trajectory between point A and point R the desired task can be related to the wall found on the left side of the robot. The pillars on the right side of the robot are distributed randomly therefore the information coming from the sensors on the right side of the robot are nothing but noise. On the other hand, once the robot crosses point R, this time the desired task can be related to the wall on the right side of the robot where now the left sensors do not contain any useful information.

![Fig. 1. The experimental scenario which demonstrates how the causal relationship between the sensor perception of the robot and the motor responses can vary along the trajectory.](image)

### III. Methodology

As discussed in the previous section, if the sensor-motor task under investigation is subject to state transitions — in the sense that the causal relationship between the sensor perception of the robot and the desired motor commands vary —, it may not be possible to identify the whole sensor-motor relationship in a single model using standard non-linear mapping techniques.

In this paper, we therefore propose a novel method in order to cope with state transitions in the data while generating robot control programs. The followed approach can be analysed in two stages:

1. In the first stage the robot clusters the environment into subspaces using standard classification techniques based on its own sensor perception. Then in the second stage it obtains a model for each cluster separately using system identification techniques such as ARMAX (Auto-Regressive Moving Average models with eXogenous inputs) [Eykhoff, 1974], [Eykhoff, 1981] and NARMAX (Nonlinear ARMAX) [Chen and Billings, 1989], [Billings and Chen, 1998]. These techniques produce linear or nonlinear polynomial functions that model the relationship between the robot’s perception and the motor responses in order to achieve the desired behaviour (figure 2).

With this method, the state transitions are related to the robot-environment interaction which allows the robot to identify the state changes automatically using its own perception. In this paper, K-means algorithm is used as a classifier, described in the next section (section III-A).

#### A. K-means Classifier

The K-means algorithm ([MacQueen, 1967]) is an unsupervised clustering algorithm which is used to classify a given data set through a certain number of clusters. The main idea is to define one centroid for each cluster and the algorithm attempts to satisfy two conditions: i) each class has a center which is the mean position of all the samples
Fig. 2. The proposed methodology in order to cope with state transition problem while generating robot control programs. One way of addressing this issue would be a classifier which divides the perception-action space of the robot into subspaces and generates a separate model for each subspace.

in that class and ii) each sample is in the class whose center is closest to.

The algorithm starts by partitioning the input points into \( k \) initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and algorithm repeated by repeat application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed).

K-means algorithm is a simple and effective way of clustering data but it has its own disadvantages. The algorithm has the problem of selecting the thresholds determining if the distance between the sample and the centroid is small enough so that the sample can be classified in that particular class. The algorithm also requires that the number of classes be known a priori which is rarely the case in robotics applications.

Note that the purpose of this paper is not to evaluate and measure the performance of the clustering algorithms but to illustrate that the proposed methodology for obtaining robust controllers to achieve sensor-motor competencies can actually illustrate that the proposed methodology for obtaining robust controllers to achieve sensor-motor competencies can actually achieve the desired prediction accuracy will in general result in an excessively complex model and possible ill-conditioned computations [Chen et al., 1990].

Having parsimonious models is vital in system identification since increasing the order of the dynamic terms \((n_y,n_u)\) and the order of the polynomial expansion \((l)\) to achieve the desired prediction accuracy will in general result in an excessively complex model and possible ill-conditioned computations [Chen et al., 1990].

Once the structure of the polynomial model is determined, the coefficients of each term in the polynomial is determined as follows:

1) The auxiliary model is defined such that the terms in the model are orthogonal over the training data set.
2) The coefficients of the auxiliary model are estimated in the least square manner.
3) The individual contribution of each term in the auxiliary model to the desired output is computed using error reduction ratio (ERR).
4) The terms having minimal or no contributions are deleted from the model.
5) The steps between 2 and 4 are repeated until the model convergences to a desired degree of accuracy.
6) Finally the coefficients of the NARMAX model are estimated from the remaining auxiliary model.

The OPE algorithm is a very well established technique and is being widely used in many control applications. More detailed discussion about parameter estimation and model validation can be found in ([Korenberg et al., 1988] [Billings and Voon, 1986] [Billings and Chen, 1998]).

IV. EXPERIMENTAL SETUP

We tested the proposed approach by teaching the SCITOS G5 autonomous mobile robot DAX, to perform two different sensor-motor competences: right wall following and route learning. The experiments were conducted in the 100 square meter circular robotics arena of the University of Essex. The arena is equipped with a Vicon motion tracking system which can deliver position data (x, y and z) for the full range of targets using reflective markers and high speed, high resolution cameras. The tracking system is capable of sampling the motion upto 100Hz with an accuracy of better than 1mm.

DAX is equipped with a Hokuyo laser range finder which can deliver distance readings upto 4m (0m ≤ d ≤ 4m) present on the front part of the robot. This range sensor has a wide angular range (240°) with a radial resolution of 0.36° and distance resolution of less than 1cm. The base of the robot is circular and the diameter of the base is 60cm.

DAX is equipped with a Hokuyo laser range finder which can deliver distance readings upto 4m (0m ≤ d ≤ 4m) present on the front part of the robot. This range sensor has a wide angular range (240°) with a radial resolution of 0.36° and distance resolution of less than 1cm. The base of the robot is circular and the diameter of the base is 60cm.

V. EXPERIMENT 1: WALL FOLLOWING

The first experiment presents result about teaching a robot to achieve right wall following behaviour. First the programmer drives the robot in the train environment manually using a joystick to demonstrate the desired wall following behaviour to the robot (figure 3). During this run, the laser readings of the robot and the motor commands (lv and av) were logged in every 250ms to obtain the training data set.

![Fig. 3. The trajectory of the robot under the control of the programmer. During this run the sensor perception of the robot (laser readings) and the motor responses (angular and linear velocities) were logged in every 250ms.](image)

A. Classifying robot’s perception

Once the training data set is obtained, in order to decrease the dimensionality of the input space to K-means algorithm, we coarse coded laser readings into 11 sectors by averaging 62 readings for each 22° intervals. Also, the coarse coded laser readings bigger than 1.5m were clamped to 1.5m so that during classification, the robot would only take into account the objects which are close enough to the robot.

We then used K-means algorithm to recognize 3 distinctive regions (regions A, B and C) in the train environment using the 11 dimensional laser signature of the robot at each time stamp along the desired trajectory. Each region is represented with a centroid here each centroid is an 11 dimensional vector where each dimension stands for one laser reading from the robot. Figure 4 shows the graphical illustration of the three centroids.

![Fig. 4. The graphical representation of the three centroids for regions A, B and C. Region A represents convex corners, therefore all the readings around the robot except from the readings from the left side has small values due to the detected walls around the robot. Region B represents regions where straight walls are therefore the front readings of the robot has also high values since there is no wall infront of the robot. Region C represents concave corners, therefore it has small values only on the very right side of the robot since there are no walls in other directions.](image)

Once the robot gets a new laser signature (11 dimensional vector, u_1 \cdots u_{11}) from the environment, the K-means classifier computes the Euclidean distance d_i of the new signature with the three centroids (equation 6). The smaller the Euclidean distance, the higher similarity between the centroid and the class, therefore the class with the smallest Euclidean distance is chosen. Figure 5 shows how the robot classifies the environment along the desired route.

\[
d_i = \sum_{j=1}^{11} |u_{ij} - u_j| \quad i = 1, 2, 3
\] (6)

B. Obtaining perception models

Having classified the environment into 3 regions, the next step was to model the relationship between the sensory perception of the robot and the desired motor responses using NARMAX system identification technique for each region separately. In order to decrease the dimensionality to NARMAX models the coarse coded laser readings were reduced into two element input vector (\(\hat{u}_1\) and \(\hat{u}_2\)) where \(\hat{u}_1\) is the minimum coarse coded laser reading among all the coarse coded readings and \(\hat{u}_2\) is the right most coarse coded laser reading in the signature.

For each region, two polynomial models were obtained; one for the translational velocity (lv) and one for the angular
velocity \( (av) \). The all models were chosen to be linear ARMAX polynomial structures of first degree with no regression in the inputs and the outputs (i.e. \( l = 1, N_u = 0, N_y = 0 \)):

\[
\begin{bmatrix}
  l_{VA} \\
  av_A \\
  l_{VB} \\
  av_B \\
  l_{VC} \\
  av_C 
\end{bmatrix}
= \begin{bmatrix}
  0.001 & 0 & 0.198 & 0.205 & 0.196 & -0.510 \\
  1.459 & -2.936 & 0.205 & 0.196 & -0.400 & 0 \\
  -0.002 & 0 & 0 & 0 & 0 \\
  0.979 & -1.980 & 0 & 0 & 0 \\
  0.002 & 0.196 & -0.400 & 0 & 0 \\
  0.182 & -0.510 & 0 & 0 & 0 
\end{bmatrix}
\cdot \begin{bmatrix}
  \hat{u}_1 \\
  \hat{u}_2 
\end{bmatrix}
\]

C. Models validation

Having obtained the sensor based controllers, we let them drive the robot in the train environment for 15 minutes (corresponding to 5 laps around the arena). During the test run, first the k-means algorithm is used to recognize the location of the robot along the trajectory and then select the polynomial modelled for that particular region to drive the robot. The results showed that the robot is able to follow the wall accurately without losing the track or bumping into the walls (figure 6).

VI. EXPERIMENT 2: ROUTE LEARNING

The second experiment is more complex than the first one here the robot has to follow a particular route in a complex environment where the causal relationship between the sensory perception of the robot and the motor responses vary significantly along the desired trajectory (figure 7). As in the previous experiment, first the programmer drives the robot manually using a joystick to demonstrate the desired route to the robot and during this run, laser perception of the robot \((u_1 \cdots u_{11})\) and the desired motor responses were logged every 250ms.

Having obtained the training data set, we used the K-means algorithm to divide the perception space of the robot into five clusters, \( A, B, C, D \) and \( E \). The internal map of the robot along the desired route which shows how the sensory data is clustered into 5 classes is illustrated in figure 8.

Once we classify the sensor perception of the robot into distinctive classes, we then obtained one polynomial model for each cluster which links the laser readings \((u_1 \cdots u_{11})\) of the robot to the desired angular velocity \( (av) \). In order to simplify the experiments, we clamped the linear velocity of the robot at \( 0.15 \text{m/s} \). All models were chosen linear ARMAX polynomial structures of first degree with no regression in the inputs and the outputs (i.e. \( l = 1, N_u = 0, N_y = 0 \)). The models for each cluster \( A, B, C, D \) and \( E \) are given in tables I-III respectively. Tables also present the error reduction ratio (ERR) value of each term presented in the polynomial where the ERR values provide an indication of the contribution that each term in the model makes to the desired output variance.
A. Models validations

Having obtained 5 polynomial ARMAX models, one for each cluster, we then tested them by letting them drive the robot in the training environment. Starting the robot anywhere along the desired trajectory, first the k-means algorithm classifies the environment according to the sensory perception of the robot. Once the robot recognizes the region it is in, it then selects the correct polynomial obtained specifically for that region to drive the robot. Once the robot reaches the next region, the second polynomial is activated and the procedure goes on in this way.

The results show that the robot was successfully following the desired route without losing the track or crashing into the boxes. Note that when we tried to model sensor-motor relationship of the robot in order to achieve the desired behaviour with a single polynomial — rather than dividing the environment into subregions and obtaining one polynomial for each region — the resultant model failed to drive the robot successfully along the desired route.

B. Transparent models allow hypothesis postulation and testing

Having transparent models like the one given in Table I has a number of advantages, for example the possibility of identifying which sensory inputs are important for the models or to optimise an existing model in a principled way.

a) Analysis of ERR values: Besides the obtained models, the tables I-III also present error reduction ratio (ERR) value of each term presented in the polynomial. The ERR values provide an indication of the contribution that each term in the model makes to the desired output variance. Since the obtained models are linear, these values can be directly related to how much information each laser reading ($u_1, \ldots, u_4$) delivers to the obtained models — the bigger the ERR value the more important the laser reading is for the model.

For example when we analyse the model obtained for region A we can say that laser reading $u_6$ is the most important information source of the model A since it has the highest ERR value. Further analysis based on the ERR values revealed that the important sensory inputs for model...
A and B are roughly same where $u_1$, $u_4$, $u_6$, $u_8$ and $u_{11}$ are the dominant laser readings for both models.

b) Obtaining a combined model for regions A and B:
Once we identified that models A and B are generally relying on the information delivered from the same laser readings, we wanted to see what would happen if we obtain one model for both regions rather than modelling them separately. We therefore used the NARMAX system identification method to obtain a global model for regions A and B expressing the rotational velocity of the robot in terms of coarse coded laser readings ($u_1 \cdots u_{11}$). The new model was chosen to be degree one no regression in the inputs and the output (i.e. $l = 1$, $N_d = 0$, $N_f = 0$) (table III).

We then validated the new model in the test environment. The results show that the performance of the new global model is as good as the previous two models (see figure 10).

Fig. 10. The trajectory of the robot under the control of polynomial ARMAX models where model AB replaces the models A and B. The results show that the performance of the new global model AB is as good as the previous two models (model A and model B) (see figure 10).

VII. CONCLUSION AND FUTURE WORK

This paper presents a novel approach to deal with the state transition problem while generating robot control programs. In this paper, the state transition problem is considered based on the cases where the causal relationship between the perception of the robot and the desired motor responses varies along the desired route. In these circumstances, trying to identify the whole sensor-motor relationship in a single model using standard non-linear mapping techniques often results in poor model performances.

One way of addressing this issue would be a classifier which divides the perception-action space of the robot into subspaces and generates a separate model for each subspace. This approach can be analysed in two stages: In the first stage the robot clusters the environment into subspaces using standard classification techniques based on its own sensor perception. Then in the second stage it obtains a model for each cluster separately using system identification techniques by producing linear or non-linear polynomial functions that model the relationship between the robot’s perception and the motor responses in order to achieve the desired behaviour.

Our experiments demonstrated that complex sensor-motor competences can be achieved by using the proposed methodology. These tasks were right wall following and route learning behaviours.

Additionally, our modelling approach produces transparent mathematical functions that can be directly related to the task. This allows an analysis of how much each sensory input is important for the overall behaviour of the robot. In the example presented here, we demonstrated this fact by identifying that models A and B roughly rely on the same sensory inputs in the desired route learning behaviour. This analysis lead us to obtain a single global model for the regions A and B rather than having separate models.

c) Future Work:
In general the proposed approach achieving the desired global behaviour using locally generated controllers, is good for local controllers but bad for modelling. The latter follows because imagine a surface we want to model — think of an example like a crumpled up piece of paper — if we fit a non-linear model then we are fitting to the surface over the paper. If we fit local linear models we tend to fit models just over one route over the surface of the paper — where really to approx the paper surface we need to tessellate the whole surface with loads of local linear models — too many models usually.

This then leads to the problem — how do we know when we go from one local model to another or how many local models to fit and obviously standard classification algorithms like the one used in this paper can not tell us the answers we are looking for. Therefore there is a need for a formal classification methodology which can identify the locations in the environment where the sensor-motor causality of the task changes and cluster these regions automatically. This is a part of ongoing research in the Universities of Ulster, Essex and Sheffield.

ACKNOWLEDGMENTS

We gratefully acknowledge that the RobotMODIC project is supported by the Engineering and Physical Sciences Research Council under grant GR/S30955/01.

REFERENCES


Abstract—This paper explores the idea that robots can learn safe behaviours by learning to reverse actions. Previously we have demonstrated that obstacle avoidance behaviour emerges when a robot learns to suppress irreversible actions and we have also demonstrated emergence of territorial behaviour in case of more complicated scenarios.

In this paper we represent comparative experiments with two different robots to investigate if a code based on this abstract principle is applicable on different robots with different shapes, size and polarity of proximity sensors in different environments. Furthermore, we compare the performance of the algorithm based on the reversibility of actions to a dedicated Q-learning obstacle avoidance algorithm. The experimental results show that the performance of the algorithm is the same on both platforms and is 10% lower than of Q-Learning algorithm. We interpret this as the evidence confirming the hypothesis. We conclude that the reversibility based algorithm can be used on different robotic platforms with minor modifications to sensory-motor interface.

I. INTRODUCTION

This paper is concerned with safety of robot behaviour by applying an abstract principle of reversibility on real robots. In [1] we demonstrated that the principle “Don’t do things you can’t undo” generates a concrete safe behaviour of obstacle avoidance. We speculated further that this abstract principle can be applied to different robots in different environments. Furthermore we speculated that this principle could generate variety of safe behaviours. In [2] we demonstrated that a more complex territorial behaviour can emerge as a result of avoiding irreversible sequences of actions.

We speculate that a robot governed by such an abstract principle will behave safely in a wide variety of environments, since many undesirable actions such as damage of the robot/environment or getting stuck is characterized by irreversibility. Although not all irreversible actions are undesirable, it is safe to say that all reversible actions are safe.

Reversibility, or absence of irreversibility, is an extension of stability in the way that reversibility can be task-specific: positive changes after “good” actions will be identified as non-stable, but reversible.

The idea of using abstract principles to govern robot behaviour has already been studied before. Kaplan and Oudeyer in [3] showed that a robot can develop visual competences from scratch driven only by internal motivations independent of any particular task: predictability, familiarity and stability.

The main benefit of using the abstract principle, instead of specific routines such as avoiding obstacles, falls, traps, risky regions or routes or staying near some known landmark, is its generality. It explains “why” a robot should behave that way and if a new problematic action/situation occurs, a robot avoiding irreversible actions will avoid these new dangers after some learning period.

The main contribution of this article is a comparative test to confirm/reject the hypothesis that the code based on this abstract principle can be run without major changes on different robots with different shapes, sizes and polarities of proximity sensors in different environments.

In the following section we present our ideas in a more formal way. In section 3 we describe the experimental setup, the algorithms used, explain the differences between the two robots used in experiments, their test environments and specific implementation details. In section 4 we present the experimental results and discuss them together with applicability of the concept of reversibility. In the last section we make conclusions and speculate about some possible directions of future work.

II. THEORETICAL FRAMEWORK

This section briefly describes the general theoretical framework used to ground the reversibility based algorithm and to test the robots. Emergence of obstacle avoidance behaviour is also explained in the end of this section. The reader is referred to [2] for more details.

A. Definitions

A robot’s world is a labelled transition system \((S, \Lambda, \rightarrow)\), where \(S\) is a set of experienced states, \(\Lambda\) is a set of labels (a label contains an action or a sequence of actions), and \(\rightarrow\) is a set of labelled transitions between the states. When the result of an action \(a\) in state \(s\) is not wholly determined by the robot, multiple transitions from \(s\) are labelled with the same action \(a\) and it is the world that determines which transition actually happens.

A reversibility for world \(W\) is a quintuple of three states and two actions: \((s_0, a_0, s_1, a_1, s_2)\). Generally speaking, a
composite action $a_0a_1$ produces a transition from $s_0$ to $s_2$ through $s_1$ in $W$. Also, the action sequence $a_0a_1$ is expected to work for any states $x$ and $y$ with $d_{\text{orig}}(x,s_0) \leq \varepsilon_{\text{orig}}$ and $d_{\text{dest}}(y,s_1) \leq \varepsilon_{\text{dest}}$, where $d_{\text{orig}}$, $d_{\text{dest}}$ are metrics on states and $\varepsilon_{\text{orig}}$, $\varepsilon_{\text{dest}}$ are their thresholds.

The reversibility $(s_0,a_0,s_1,a_1,s_2)$ holds in $W$ if there exists a transition path from $s_0$ to $s_2$ through $s_1$ consisting of two transitions labelled accordingly $a_0$ and $a_1$, and $d_{\text{rev}}(s_0,s_2) \leq \varepsilon_{\text{rev}}$, where $d_{\text{rev}}$ is a prametric $(d_{\text{rev}}(x,y) \geq 0$ and $d_{\text{rev}}(x,x) = 0$) on states and $\varepsilon_{\text{rev}}$ is a threshold; fails otherwise.

An action $a_0$ in an arbitrary state $s$ is expected to be reversible (by action $a_1$), if the reversibility $(s_0,a_0,s_1,a_1,s_2)$ holds and $d_{\text{orig}}(s,s_0) \leq \varepsilon_{\text{orig}}$.

A reversibility model of the robot is a set of reversibilities that are expected to hold.

B. Explanations

A reversibility model can be given to the robot in advance, transferred from another robot, extracted by a human from the knowledge about the world or learned by the robot. Using this model a robot can predict whether the action from the state is reversible by iterating through its experience and using obtained reversibilities to ground the predictions.

The actions used are symbolic actions and it is irrelevant whether they are atomic or complex actions. These actions can also be interpreted as discrete choices if used by a high level symbolic decision maker. The only requirement is that every action must have a reverse action, i.e. the action that undoes (reverses) it.

States are also discrete but with metrics $d_{\text{orig}}$ and $d_{\text{dest}}$ defined on the set of the states. These metrics are used to search for the reversibilities to ground the predictions. Metric $d_{\text{orig}}$ together with its threshold value $\varepsilon_{\text{orig}}$ are used to filter reversibilities by calculating the distance between its initial state and the current state. The smaller the distance, the higher is the probability that the actual outcome of making the same action from the current state will generate a similar reversibility.

A prametric $d_{\text{rev}}$ is used to calculate how strongly the reversibility holds. A prametric is used instead of a metric to reward transitions from “worse” states to “better” ones (in case of goal-oriented learning); if $d_{\text{rev}}$ is a metric, then the calculated number measures stability.

C. Emergence of obstacle avoidance behaviour

Let us explain how and why the obstacle avoidance behaviour emerges as a result of avoiding irreversible actions. To simplify the example we will use a robot with a proximity sensor and two actions - “move 10 steps forward” and “move 10 steps backward”. Without loss of generality we can assume that “steps” and values of proximity sensors are given in comparable units.

The robot tests these actions in different situations and checks whether the obtained reversibilities hold. The ones that fail usually correspond to collisions of some sort or other negative outcomes. Consider the following 4 cases, where the robot makes 10 steps forward and then 10 steps back:

1) If the robot is at least 10 units away from the obstacle, say, 12 then it doesn’t touch the obstacle and we obtain the reversibility which holds:

$((12),(+10),(2),(-10),(12))$

2) If the robot is less than 10 units away from the wall, say, 8 then it touches the wall and its motor stall, we obtain the reversibility which doesn’t hold:

$((8),(+10),(0),(-10),(10))$

3) If the robot touches the wall and its wheels slide on the surface then we obtain the same reversibility as in case 2.

4) If the robot touches the obstacle, but the obstacle is light enough to be moved, then the obtained reversibility will also be identical to case 2 from the robot’s point of view.

This way the robot discovers that running into or pushing an obstacle is “bad” without even knowing what the “obstacle” or “pushing” is. A reversibility model with such reversibilities will allow a robot to distinguish those state-action pairs in which “bad things happen” from those in which they do not.

III. Experimental Setup

The purpose of the experiments is to verify how abstract the implementation of the principle is. For this purpose we compare the performance of the reversibility based algorithm on two different robots and compare these results to another well-known algorithm for obstacle avoidance (Q-Learning).

A. Comparative experiments

The experiments consist of two test runs (5200 steps each) on two different robots. Each test run is divided into two phases – Phase 1 (data collection phase) and Phase 2 (simulation phase).

During Phase 1 the robot makes pseudo-random moves and the input data (sensors data, actions made and outcomes of the actions) is collected and saved into log files. The predictions are made on-line during Phase 2 using data collected in the test runs. The performance is measured by sampling algorithms’ predictions of whether the next action will succeed and calculating the success rate of those predictions.
B. The robots

Comparative experiments are conducted on two common research robot platforms, Khepera II by K-Team and Scitos G5 by MetraLabs. The experiments on Khepera II are reported in our previous work [1]. In this paper these experiments are repeated on Scitos G5 robot in comparable environmental conditions. The size of the environment was increased proportionally to the size of the robot.

The relevant technical aspects of these robots are presented in Table I for comparison. The most important differences between these robots are their shape and polarity of their sensors.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>COMPARISON OF KHEPERA II AND SCTIOS G5 ROBOTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Khepera II</td>
</tr>
<tr>
<td>Width</td>
<td>70 mm</td>
</tr>
<tr>
<td>Length</td>
<td>70 mm</td>
</tr>
<tr>
<td>Height</td>
<td>30 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>0.080 kg</td>
</tr>
<tr>
<td>Payload</td>
<td>0.250 kg</td>
</tr>
<tr>
<td>Number and type of sensors</td>
<td>8 Infra-red proximity and ambient light sensors with up to 100mm range</td>
</tr>
<tr>
<td>Sensors polarity</td>
<td>Counter-proportional to distance</td>
</tr>
</tbody>
</table>

Both Khepera II and Scitos G5 are differential drive robots but with different size and slightly different geometry. Khepera II has the circular shape and the rotation axis is exactly at the centre of the circle. Therefore it can rotate freely in very close proximity (1-2 mm) to the obstacle without touching it.

Scitos G5 also has a circular shape but with an additional compartment at the back side for the passive third wheel, which considerably changes the way it can rotate its own body: a 360° turn can be completed without touching the obstacle only if the distance to the obstacle is larger than approximately 200 mm (the size of the passive wheel compartment).

The sensors of Khepera II and Scitos G5 differ considerably. Khepera’s sensors are counter proportional to the measured distance with non-linear characteristics, while Scitos’ sensors have linear characteristics and are proportional to the distance measured.

C. Environments

The environment for Khepera II is a right-angled triangle with side lengths 196mm, 125mm and 233mm. It was set up by using an additional wall in a smaller rectangular box. The material for the smaller box, as well as the additional wall, is a package box for electronic devices, its surface is flat and robot’s wheels do not slip on it. We set up a very small test environment to make negative outcomes (collisions) appear more frequently. The program was run on a PC, it communicated with the robot through a serial interface to read sensors’ data and give commands to the motors. The cable provided both serial link to computer and power for Khepera II.

The environment for Scitos G5 is a rectangular box of size 970mm x 1500mm. Floor is linoleum and walls are made from corrugated cardboard.

D. Robot Movements

The robot is given a set of actions with corresponding reverse actions: movements forward-backward and turning left-right are pair wise reverse-actions to each other.

Actions are defined in terms of wheel commands. An action \( a = (m_1, m_2) \) consists of a pair of motor
displacement commands, for left and right wheels, expressed in native wheel encoder units for Khepera II. A discrete set of actions is used in the experiments:

- $a_0 = (-200,200)$ rotate counter-clockwise,
- $a_1 = (200,200)$ make a step forward,
- $a_2 = (-200,-200)$ make a step backward,
- $a_3 = (200,-200)$ rotate clockwise.

where

$\begin{align*}
\alpha_0 &= -\alpha_1, \alpha_3 &= -\alpha_0, \alpha_2 &= -\alpha_1, \alpha_1 &= -\alpha_2
\end{align*}$

In other words, going forward undoes going backward and turning right undoes turning left, and vice-versa.

The wheel commands in Khepera II internal units were translated to the speed commands of Scitos G5 as following: 200 units correspond to approximately 150 mm forward/backward movements and approximately 42 degrees rotation angles. For Khepera II these values are approximately 16mm and 30 degrees.

In the world $W$ the state vector is $s = (d_0,d_1,d_2,d_3)$ where $d_i$ are sensor values for front, back, left and right sensors, accordingly. The robot moves using the algorithm described in Fig. 3.

1. Record current state $s_i = (d_0,...,d_3)$
2. Execute a random action as $a_i$
3. Record the state $s_{i+1} = (d_0,...,d_3)$
4. Execute the reverse action for $a_i$ as $a_{i+1}$
5. Record the resulting state as $s_{i+2}$
6. Execute a random action as $a_{i+2}$
7. Add 3 to $i$ and repeat.

Fig. 3. Movement algorithm (Phase 1)

In other words, the robot makes a random move followed by its reverse action, then makes another random action, but without a reverse action, and then repeats the pattern. The purpose of the first two actions is to generate at least one pair of actions to test if the reversibility holds. The purpose of the next (random) action without a matching reverse action is to make the robot to explore the environment.

**E. Software design**

The code consists of the following units (see Fig. 4):

- an independent agent that generates the sequence of actions to move the robot during the first phase
- Q-Learning and reversibility based algorithms running in parallel.
- a “switch” to route data between the agent and the algorithms, or to simulate the test run in the second phase

In Phase 1 real-world data is gathered from the test run and saved into a log file. The file contains sensor readings, actions made and the outcomes of the actions.

Phase 2 is a simulation and can be executed without a robot. In the beginning the log file from Phase 1 is loaded into memory, parsed as sensor readings and actions and then this history is fed to the algorithms, getting predictions of actions’ successfulness simultaneously (see Fig. 5).

**F. Reversibility based algorithm**

The aim of the reversibility based algorithm is to predict if a certain action from a certain state is reversible or not.

The algorithm is described in Fig. 6. It takes a sequence of states and actions as an input: $s_0,a_0,s_1,a_1,s_2,a_2,s_3,...$.

At every $i > 1$, if $a_{i-1} = -a_{i-2}$ then the reversibility $(s_{i-2},a_{i-2},s_{i-1},a_{i-1},s_i)$ is added to robot’s experience, which is a vector of reversibilities.

To predict the outcome of making action $a_i$ from state $s_i$, an expected irreversibility value $v_{rev}$ is calculated using a set of reversibilities selected from the experience vector (in the experiments we select reversibilities with the same forward action and $d_{org}(s_0,s_i) < \varepsilon_{org}$, where $s_0$ is the initial state of the reversibility under consideration).

The value of $v_{rev}$ is a weighted average of $d_{rev}(s_{i-2},s_i)$ values of selected reversibilities. Reversibilities are sorted by $d_{org}(s_0,s_i)$ in an ascending order and their weights are $1/i^2$ (1, 1/4, 1/9, 1/16, etc.), i.e. reversibilities with a “closer”
initial state have a stronger influence.

In the experiments we use the Euclidean metric to calculate $d_{\text{orig}}$ and $d_{\text{rev}}$; the values $\epsilon_{\text{orig}}$ and $\epsilon_{\text{rev}}$ are finite and selected manually. The metric $d_{\text{dest}}$ was not used in the experiments, i.e. $\epsilon_{\text{dest}} = \infty$.

1. Read current state $s_i = (d_0,\ldots,d_3)$ and the next action $a_i$ from log.
2. Choose a number of reversibilities from the set of experienced ones with $a_i$, forward action, based on $d_{\text{rev}}$ between $s_i$ and experienced reversibility’s initial state.
3. Calculate the expected irreversibility value $v_{\text{rev}}$ using $d_{\text{rev}}$ of experienced reversibilities’ initial and final states.
4. If no reversibilities are selected, make no prediction.
5. If $v_{\text{rev}}$ is greater than $\epsilon_{\text{rev}}$, then predict negative outcome, predict positive outcome otherwise.
6. If $i < 2$, add 1 to $i$ and repeat.
7. Read the last action as $a_{i-1}$ and the previous action $a_{i-2}$ from log.
8. If $a_{i-1}$ is not a reverse-action of $a_{i-2}$, add 1 to $i$ and repeat.
9. Add the new obtained reversibility as $(s_{i-2},a_{i-2},s_{i-1},a_{i-1},s_i)$ to the set of experienced reversibilities.
10. Add 1 to $i$ and repeat.

**G. Reinforcement learning algorithm**

Reinforcement learning is a commonly used learning method to learn obstacle avoidance by trial and error ([5], [6], [7]). Therefore we have chosen a Q-Learning algorithm to compare the performance of the reversibility based learning to a standard method.

The main difference between reinforcement learning algorithms and the reversibility based algorithm is that a reinforcement learning algorithm receives an external reward signal indicating the success of an action. Reversibility based algorithm, on the other hand, uses only sensor data to determine the success of an action.

In the Q-Learning algorithm the expected reward of a state-action pair is updated using the following expression:

$$Q(s_i,a_i) \leftarrow Q(s_i,a_i) + \alpha_i(s_i,a_i)[r_i + \gamma Q(s_{i+1},a_i) - Q(s_i,a_i)] \quad (1)$$

Our experiment consists of random movements. Therefore the long-term reward is irrelevant and only short-term reward should be used, for this reason we take $\gamma = 0$.

The prediction value is calculated as $\text{sign}(Q(s_i,a_i))$, i.e. negative $Q$ means a negative prediction, positive $Q$ means a positive prediction. Initially, $Q$ values are set to 0 and if $Q$ still has the initial value, no grounded prediction can be made.

**H. Other implementation details**

Khepera’s infra-red sensors are very sensitive to indoor ambient light, therefore its test environment was placed into a box to reduce sensor noise. The corrugated cardboard for Scitos G5 environment was chosen because it reflects ultrasound much more uniformly than other non-corrugated materials.

Scitos G5 default configuration file was altered to change the way sensor readings were made (low noise mode is on, reading interval is 50ms, 4 sensors per measurement) and rotational PID controller’s Kp was set to 0.2. Sensor values for Scitos are in metres, therefore they are multiplied by 1000 to be of equal scale to the ones of Khepera. This doesn’t affect the reversibility based algorithm, but makes saving and loading the log file simpler.

During the experiments $\alpha_i(s_i,a_i)$ for Q-Learning update expression was set to 0.01. Threshold values $\epsilon_{\text{orig}}$ and $\epsilon_{\text{rev}}$ were constant throughout the experiments, but the experiments for different robots used different values. In experiments with Khepera II the settings were: $\epsilon_{\text{orig}} = 6300$ and $\epsilon_{\text{rev}} = 5000$. In experiments with Scitos G5 the settings were: $\epsilon_{\text{orig}} = 35000$ and $\epsilon_{\text{rev}} = 48000$.

**IV. RESULTS**

![Fig. 7. Test run results for Khepera II](image-url)
obstacles. Fig. 7 represents the test results for Khepera II, it compares prediction success rate for the Q-Learning and reversibility based algorithms. These results are also reported in [1]. Fig. 8 represents the comparative experimental results for Scitos G5.

![Graph showing test run results for Scitos G5](image)

**Fig. 8. Test run results for Scitos G5**

### A. Q-Learning vs. Reversibility based learning

It appears that on both robots Q-Learning converges to a 10% higher prediction success rate than the reversibility based learning.

Let us remind the reader that while the Q-Learning algorithm is explicitly designed to avoid obstacles (at every collision the robot gets a negative reward signal proportional to the size of unfinished movement), the robot learning a reversibility model has no concept of an obstacle or collision.

The reversibility based algorithm, at the same time, does not use the reward signal and only tries to predict whether the action will be reversible or not. If the robot suppressed the irreversible actions it would emerge to obstacle avoidance behaviour very similar to the one achieved by a dedicated obstacle avoidance Q-Learning algorithm.

The 10% higher performance of the Q-Learning algorithm can be easily explained. The method of measuring the success of predictions always works in advantage of the Q-Learning algorithm. The Q-Learning algorithm predicts future rewards based on the experienced rewards, while the reversibility based algorithm predicts future rewards based on sensor data alone.

**B. Khepera II vs. Scitos G5**

It appears also that reversibility based algorithm performs equally well on both robots: it converges to about 70% success rate in predicting collisions after about 3000 steps.

The Khepera’s graph has several drops in prediction success rate around regions of 800, 1700, 3100 and 3900 steps. The Khepera II robot was stuck occasionally during those periods of time, which decreased the learning and prediction success rates.

The aim of these experiments was to confirm/reject the hypothesis that the code based on an abstract principle of avoiding irreversible actions can be run without major changes on different hardware in different environments.

We interpret the results as positive, since, indeed, a concrete robot behaviour of obstacle avoidance is observed on two different robots to emerge from the abstract principle “Don’t do things you can’t undo”. However, there are problems with this straightforward plain-sensor approach: it is influenced by many factors like sensor precision, sensor noise, actions’ precision, etc. However, this problem belongs more to the realm of the state identification: Q-Learning algorithm severely suffers from the same problems.

It is difficult to distinguish sensors by their importance for the particular action. For example, sensors on the back side of the robot are useless when predicting whether moving forward will succeed or not. Different kind of sensors can also be a problem, since Euclidean distance takes all numbers equally into account. Thus, a sensor returning current time stamp or a sensor returning distance in millimetres and others in metres will be a huge problem in this case and will render both algorithms almost useless without additional tuning.

Q-Learning uses discrete states, thus, space state tiling is a problem, also the source of the reward signal must be chosen carefully to reward only collision-free movements and to penalize only collisions.

The reversibility principle based algorithm has a similar problem of state identification, sensors’ linearity must be as strong as possible, and the scale of sensor values must be the same or proportional to the importance of the sensor for state identification. Another problem is to choose threshold values $\epsilon_{\text{orig}}$, $\epsilon_{\text{dest}}$ and $\epsilon_{\text{rev}}$. We chose those values manually using statistical information of the particular test run data.

### V. CONCLUSIONS AND FUTURE WORK

The goal of this paper was to verify whether a concrete behaviour of obstacle avoidance can emerge from an abstract principle of avoiding irreversible actions. Also we wanted to compare the performance of the strategy on different robotic platforms.

We conclude that both robots involved in the experiments demonstrated similar performance compared to each other and to Q-Learning algorithm.

We see the future of this research as a cooperative work of environment-model-aware algorithms in conjunction with abstract principles to guide them on a higher level of control with the higher level of abstraction. Another direction is to use the principle of reversibility to make other learning algorithms learn faster or safer, or both.
REFERENCES


A Developmental Algorithm for Ocular-Motor Coordination

Fei Chao, Mark H. Lee and Joseph J. Lee

Abstract—We present a model of ocular-motor development, inspired by ideas and data from developmental psychology. The learning problem concerns the growth of the transform between image space and motor space necessary for the control of visual saccades. An implementation is used to produce experimental results and these are presented and discussed. The algorithm is simple, extremely fast, incremental and cumulative, and exhibits emergent stages of behaviour as learning progresses.

I. INTRODUCTION

The recent enormous advances in brain science have provided much data and stimulation for research in developmental robotics [1]. Much of this work draws on computational neuroscience and detailed structural models of brain systems. However, an alternative approach is to study human behaviour and build models that exhibit similar performance as observed and reported by psychologists. Nevertheless, despite the existence of a rich source of data and theory provided by psychology, there is a large gap between our psychological theories of development and our ability to implement working developmental algorithms in autonomous agents. This paper describes one study on the path towards closing that gap.

The problem of sensory-motor coordination is a fundamental issue for both animals and robots [2] and we are investigating this topic by drawing on psychological knowledge of very early infant development. Our work deals with development in a hand/eye system [3] and in this paper we deal with the learning issues involved in the sensory-motor control of a robotic “eye”.

In order to fixate a stimulus in the centre of the visual field, humans move their eyes in very fast actions known as saccades. This is a very efficient process that must rely on close coordination between the image space of the retina and the motor space of the ocular muscles. There are many structural or neural models of saccade generation [4], but we use our developmental approach [5] to produce a method that learns much faster than neural network based approaches and also displays emergent stages in its behaviour.

This paper is structured as follows: Section 2 presents an overview of the human ocular-motor system; Section 3 describes our visuo-motor coordination mapping model and the associated developmental learning algorithm; Section 4 explains an experimental implementation; Section 5 describes results from experiments; Section 6 discusses the implications of this work; and finally, Section 7 gives a summary.

II. HUMAN VISUAL SENSING AND THE OCULAR-MOTOR SYSTEM

The human visual system provides inspiration for our research and we briefly describe its basic features.

Each human eyeball is moved by six muscles; operating in pairs, they rotate the globe in three degrees of freedom. The muscles are rich in spindle receptors that give high quality information about the stretch of the muscles which corresponds to the position of the eyeballs relative to the rest of the head. Our robot system has only one eye and the head is fixed in space. This simplifies the system compared to the human, although we note that infants do not integrate head movements until around two months [6].

Newborn infants are unable to track objects (smooth pursuit) or to discriminate motion direction, but they are able to perform saccadic eye movements. Saccades are fast movements of the eyes that usually bring a stimulus from the periphery to the centre of the retina. In humans this brings the image onto the macula and in particular the fovea, the region of densely packed cones which allows the greatest acuity and colour sensitivity. Studies of human infants’ saccades have shown that there are stereotyped age related changes in the way that their eyeballs move and that the more advanced movements coexist with earlier movements [7].

Saccades can move at up to 900° per second; too fast for visual feedback to occur during the movement. This means that they must use a form of feedforward control that is pre-programmed, and the program must be either innate or learned. In order for the image to fall upon the fovea, the eyeball must be aligned in the correct orientation. For this to occur innately would require prior knowledge of the muscular system and the optical characteristics of that particular eyeball, and it seems almost inconceivable that this would happen in the infant.

The issue of innate versus learned behaviour has been debated between the psychological empiricists and nativists over many decades. Because newborn infants can produce saccades it is generally assumed that this is an innate competency, but we demonstrate here that very rapid learning is a feasible alternative hypothesis.

III. A MODEL OF OCULAR-MOTOR COORDINATION LEARNING

The main control issue for the ocular-motor system is a sensory-motor coordination problem: what are the necessary motor variables to drive the eye to move the foveal area to a specific sensed peripheral region? It is important to state that we do not model the dynamical aspects of saccades, that is, parameters such as velocity, gain and duration which
have been extensively examined [8]. Our concern is with the acquisition of targets for the necessary open-loop saccadic feedforward mechanisms. The motor systems of the eye are exercised in the womb [9] and so motor activation signals can be correlated prenatally with proprioceptive signals from the muscle spindle receptors. But the position of the eyeball can not be related to image data until after birth. Hence, the problem we address is the growth of the transform between image space and motor space.

In our work we have used a mapping technique to model sensory-motor relationships [3] and we adopt this method here. Each channel of either sensory or motor information is provided with a two-dimensional map consisting of many overlapping elements. These elements, known as fields, represent patches of receptive area in which the values are equivalent. In our models we use sheets of fields that are circular and overlapping. Our system has image data as the sensory input and a two-degree motor system for moving the image. Thus, two map layers are needed and these are illustrated in figure 1. The left layer is a visual sensory map which uses polar coordinates because a polar mapping is the natural relation between central and peripheral regions on the retina [10]. The layer on the right in figure 1 is the associated motor drive layer; this is a motor map in two degrees of freedom (we ignore axial rotation of the eyeballs) and encodes the horizontal (left-right), and vertical (up-down) eye movements. As correspondences between fields on different layers are discovered by experience so they become directly linked. That is, when a movement causes an accurate shift of the fovea to a periphery stimulus, then the sensory field (giving the stimulus location) is explicitly coupled to the motor field (giving the motor variables that produce the change). By this means, the sensory-motor relations for accurate saccades are discovered and learned.

Following the human retinal structure, we designed the field density to be higher in the central area than the periphery. This was achieved by a simple generation rule that allows field radius to be proportional to distance from centre. This design means that the longer saccades may be more inaccurate, but this is in accord with studies showing that early infant saccades are not accurate and precise [11], nor are they in adults [12]. The motor coordinate system is Cartesian, because each motor is independent and orthogonal.

A. The Developmental Learning Algorithm

Previous research shows that two week-old infants scan geometric figures rather randomly, while fourteen week-old infants direct their saccades to stimulus contours more consistently [13]. It seems that saccadic eye movements are refined over a period. However, N.J. Butko et al [14] proposed a rapid learning hypothesis which argues that very fast learning might occur just after birth. We suggest such fast learning for eye saccades might be obtained by the following process: when an infant senses an object appearing in her field of vision, the infant’s brain is stimulated to try to move her eyeballs to fixate the object. However, because all visual data is novel to a newborn infant there can be no prior coordination of retinal image space with motor acts. Hence, the appropriate motor values are not known and so the infant’s brain may generate spontaneous (random) motor values in an attempt to move towards the object. When, eventually, the infant’s eyes fixate on the object, then the brain can record the parameters of the successful experience for future reference.

An autonomous learning algorithm can be developed to reflect the above learning process and this is summarised as pseudo code in Figure 2.

![Fig. 1. The Ocular-Motor Map Layers](image)

**Fig. 1. The Ocular-Motor Map Layers**

sensory input and a two-degree motor system for moving the image. Thus, two map layers are needed and these are illustrated in figure 1. The left layer is a visual sensory map which uses polar coordinates because a polar mapping is the natural relation between central and peripheral regions on the retina [10]. The layer on the right in figure 1 is the associated motor drive layer; this is a motor map in two degrees of freedom (we ignore axial rotation of the eyeballs) and encodes the horizontal (left-right), and vertical (up-down) eye movements. As correspondences between fields on different layers are discovered by experience so they become directly linked. That is, when a movement causes an accurate shift of the fovea to a periphery stimulus, then the sensory field (giving the stimulus location) is explicitly coupled to the motor field (giving the motor variables that produce the change). By this means, the sensory-motor relations for accurate saccades are discovered and learned.

Following the human retinal structure, we designed the field density to be higher in the central area than the periphery. This was achieved by a simple generation rule that allows field radius to be proportional to distance from centre. This design means that the longer saccades may be more inaccurate, but this is in accord with studies showing that early infant saccades are not accurate and precise [11], nor are they in adults [12]. The motor coordinate system is Cartesian, because each motor is independent and orthogonal.

For each session

- If object in peripheral vision at $\theta$, $\gamma$
  - Access the ocular-motor map
  - If a covering field exists
    - Use motor values for this field
  - Else
    - Record the object’s position, make a spontaneous motor move
    - If the object is within foveal region
      - Generate a new field, enter the object’s location and the associated motor values
    - Else
      - Iterate a new session
  - End If
- Else
  - Do not move
- End If

Iterate a new session

This outline algorithm is transformed by the addition of the following two modifications.

1) Nearest Field Selection: Suppose that the ocular-motor map has not yet generated any fields that cover the current stimulus location, let this be $(\theta, \gamma)$. The nearest field to the stimulus can then be selected as an approximation to the target. For this, the following nearest selection procedure was designed: first, an angular tolerance is set to select the fields which have a similar angle with the target field $(\theta)$, this tolerance is thus defined: $\theta \pm \delta_1$. Then, a distance tolerance...
is set to select the fields nearest to the target field from amongst the candidate fields in the above set. The distance gap is defined as: \( \gamma \pm \delta_2 \) pixels. The angular parameter is given precedence over distance because, in polar coordinates, the angular coordinate alone is sufficient to determine the trajectory to the origin. From this we obtain a set of fields which fall within the (broad) neighborhood of the stimulus, amongst the candidate fields in the above set. The distance gap and the following formula

\[
\text{MIN}(\sqrt{(\gamma - \gamma_x)^2 + (\theta - \theta_x)^2})
\]

is used to choose the nearest field from this collection, where \( \gamma_x \) and \( \theta_x \) are the access parameters of the fields in the collection. This is summarised as pseudo code in Figure 3.

<table>
<thead>
<tr>
<th>If no fields exist for location ( \theta, \gamma ):</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. For each field, ( f_x \in \text{Fields} )</td>
</tr>
<tr>
<td>If ( \theta - \delta_1 &lt; f_x(\theta) &lt; \theta + \delta_1 )</td>
</tr>
<tr>
<td>( \text{Candidates} = \text{Candidates} \cup { f_x } )</td>
</tr>
<tr>
<td>b. For each field, ( f_x \in \text{Candidates} )</td>
</tr>
<tr>
<td>If ( \gamma - \delta_2 &gt; f_x(\gamma) ) or ( f_x(\gamma) &gt; \gamma + \delta_2 )</td>
</tr>
<tr>
<td>( \text{Candidates} = \text{Candidates} \setminus { f_x } )</td>
</tr>
<tr>
<td>c. Apply the MIN formula to ( \text{Candidates} ) to find the nearest field to ( \theta, \gamma ).</td>
</tr>
</tbody>
</table>

Fig. 3. The Nearest-Field Selection Algorithm

Note that although the variables in the nearest algorithm use different units their ranges are compatible for this purpose (being 0-360 degrees and 0-300 pixels). In the experiments (see Section V), \( \delta_1 \) is set to 15\(^\circ\) and \( \delta_2 \) is set to 10 pixels.

2) Vector Field Generation: In the basic algorithm given in Figure 2, a new field cannot be generated until the camera has fixated an object at that location, and this process typically takes a long time because most spontaneous moves will not result in a target fixation. However, we note that there is a change in the location of the stimulus in the image after each movement. A vector can be produced from this change by:

\[
\text{Vector} = \text{Position}_{old} - \text{Position}_{new}
\]

where \( \text{Position}_{old} \) denotes the object position before movement and \( \text{Position}_{new} \) the object position after. This vector represents a movement shift of the image produced by the current motor values. Consequently, the vector can be used to access a field in the image layer together with its corresponding motor values on the motor layer. In so doing, a new field can be generated after each spontaneous movement.

During early learning many spontaneous movements will be needed until a fixation is achieved and by using the movement vector idea each fixation can generate many vectors. The current vector will be a sum of the previous vectors, thus:

\[
\text{Vector}_{sum} = \sum_{i=1}^{n} \text{Vector}_i
\]

and the corresponding motor values can also be produced by summation:

\[
M_{sum(p,t)} = \sum_{i=1}^{n} M_{(p,t)}
\]

This is an incremental and cumulative system, in that the resultant vectors can be built up over a series of actions by a simple recurrence relation:

\[
\text{Vector}_{sum}(t + 1) = \text{Vector}_{sum}(t) + \text{Vector}_i(t + 1)
\]

IV. System Implementation

Our laboratory robot incorporates a motorised camera system that acts as an “eye”. Figure 4 shows the hardware components consisting of a video camera mounted on a pan-and-tilt head.

A. The Motor Subsystem

The motor system is implemented by a motorised pan-and-tilt device which provides two degrees of freedom. The pan motor can drive the video camera to rotate about an axis that translates the image in one direction, and the tilt motor can drive rotation about an orthogonal axis, giving image translation at 90 degrees. Combined movements of pan and tilt motors cause motion along an oblique axis. The pan/tilt device can effectively execute saccade type actions based on supplied motor values from the learning algorithm. Each motor is independent and has a value \( M_p \) for pan and \( M_t \) for tilt) which represents the relative distance to be moved in each degree-of-freedom.

B. The Sensor Subsystem

The camera captures workspace images and image processing software is used to implement two sensors: a periphery sensor and a centre or foveal sensor. The periphery sensor detects new objects or object changes in the visual periphery area and also the positions of any such changes (encoded in polar coordinates). The centre sensor detects whether any objects (i.e colour blobs) are in the central (foveal) region of the visual field. Figure 5 shows the workspace which is a white table, with objects (painted green). This setup was arranged to simplify the image processing task, especially object detection.

The camera capture rate is one frame per second. Each object is represented by a group of green pixels flocking together in the captured image. The position of the centroid of the pixels is used as the location of the object. The image
processing program compares the currently captured image against the stored previous image and, if the number or the position of any central pixels within these two images differs markedly, the object is considered to have changed and the location of the centroid is encoded in polar coordinates. A circular area, of radius 20 pixels, in the centre of the image is defined to be the foveal region. If the centroid of an object is in this central area, it is considered that the object is fixated; otherwise the object is not fixated. Rather than only

V. EXPERIMENTAL RESULTS

The experiments are designed to investigate the relationship between image space and ocular motor proprioception. The experimental procedure is ordered as follows: an object is placed within the camera’s field of view, then the developmental learning algorithm drives the camera until the object is fixated; after fixation, the object is moved to a new position (still within the camera’s view) and this process iterates. During this procedure, no people or other agents are involved except for moving the object’s position. Fields in the

A. Observations

From a large number of experiments carried out, we observed that this system’s behavior can be described as falling into three stages: (1) at the beginning of a new ocular-motor map, (2) after a few fields have been generated, and (3) after most fields have been created.

When a moderate number of fields have been generated, the algorithm is normally still unable to find an exact corresponding field for the stimulus, but a nearby field is usually available. This means the end position of each movement tends to be towards the visual centre region. Figure 7 illustrates the process of this second stage: large spontaneous movements do not happen any more, and the movement traces are near the image centre.

At the third stage, because most fields have been generated, the learning algorithm is able to find the correct corresponding field (and the associated motor values) each time, the camera movement is simple, merely fixating the object directly. Figure 8, comprising six experimental results in one plot, shows the traces as radial movements, from periphery to centre.

Figure 9 illustrates the outcome of the set of experiments: the upper figure presents the sensory layer and the lower figure the motor layer. It can be seen that much of the image map has been covered with fields, in this run a total of 72 fields were produced. The fields in the sensory layer are plotted in polar coordinates and marked by numeric labels, which give correspondence with the motor fields.
During the experiments each movement was recorded and flagged as one of three types: spontaneous, using a neighbouring field, or direct. Figure 10 is a cumulative plot that shows the mix of movement types over time:

- The number of spontaneous movements (line A) dominates during the first thirty movements, however, this type of movement occurs very little from then on.
- Movements using nearest neighbour fields (line B) do not exist at the beginning, but this type of movement increases sharply after that, and then after a period of growth this movement is seen much less, after about 70.
- Direct, accurate movements using the correct corresponding fields (line C) do not occur at all during the first eighteen movements, however at the end of the experiments these have the fastest rate of increase, until finally only these single saccades exist.

In order to illustrate where the fast learning occurs, Figure 11 shows the rate of new field generation over an entire experiment. As can be seen, the field generation rate produced by the developmental learning algorithm is very fast for the first 70 movements, then the rate decreases, and finally, field creation becomes very rare.

Another illustration of the learning process is seen in Figure 12, taken from another, different run of experiments. This data covers 88 movements in total, separated out into numbers per individual fixation. This shows how the number of movements per fixation falls away very rapidly, the reason being that even a sparse covering of fields aids convergence because a near neighbour can usually be found.

It is important to note that the emergence of the observed stages are not controlled by any switching or thresholds. Indeed, at any point the system might revert to an earlier behaviour type, as at any time a new field might need to be introduced. Eventually the early behaviours will be extinguished but as this is an asymptotic process there is always a finite possibility of regression.

VI. DISCUSSION

It has been suggested that the preference of newborns to orient towards faces is not innate, as generally believed, but could be learned very rapidly, even in the first six minutes of life [14]. Our robot model has demonstrated that the fundamental process of visual saccades could also be learned rather than innate. If such rapid learning does occur it would be quite difficult to detect but it could be very significant evidence for the empiricist stance.
There is a large literature on eye movements and saccades but very little is relevant to the first few hours and days after birth. However, we find considerable support for our model which seems entirely compatible with current knowledge. For example, it is known that infants execute smaller steps than adults with reports of “sometimes making as many as four or five consecutive saccades to reach the peripheral target” [15], and “larger step sizes were used to localize more distant targets” [15].

As in the robot, the average number of saccades an infant uses to fixate an object onto its fovea reduces with time. Roucoux et al [7] found that infants fixed targets onto the fovea with successive small saccades even though they were capable of larger saccades. They also found that the more eccentric the target the higher the average number of saccades, and for targets at all angles the average number of these saccades declined with age. This is similar to the robot model, in which the number of saccades decreases over time.

result in the image

Hainline et al examined saccade peak velocity, amplitude and duration in infants and compared them to adults [16]. Their sample of 64 infants of ages 14 to 151 days produced a significant proportion of mature saccades that were comparable to adults. This was confirmed by [17] who also showed that infant saccades may even be faster than adults. The mix of infant and adult movement types reported by such authors can be interpreted as part of a learning process and the decline of the early patterns with age reflects the increasing dominance of the more efficient saccades. This pattern is seen in the robot model, where mature and immature movement patterns (from the 3 stages) coexist until the system has fully learnt the relationship between its motor and visual maps. An interesting hypothesis [18] is that the commonly observed undershoot in saccades may be an optimum strategy to minimise the total flight-time, because the total flight-time is less with corrective saccades for undershoot as compared with overshoot. This effect also occurs in our method - the foveal fields are predominant among the first to develop and so when a neighbour is selected it is likely to be on the near side of the target. We analysed the data for a run of fixations and found that undershoot occurred in 75% of the cases.

Harris also argues strongly that sophisticated control theory is inappropriate for modelling saccade control because this is an un-referenced control problem (i.e. no error reference is available) [18]. This means exploration of the problem space is necessary and “randomness (variability) of activity reflects an active process for exploring … rather than being simply neural noise” [18].

Regarding other models of saccading and ocular-motor coordination, we find that the learning times reported are usually orders of magnitude greater than for our system. Connectionist methods have been widely used for sensory-motor modelling and radial basis functions have similarities with our mappings. However these methods often involve extensive training regimes, typically involving 20,000 cycles [19], and the number of neurons involved can scale up exponentially [20]. Some systems have been primed with an initial linearly normalised map and then learning is used to adjust the errors to the true map, but even with this prior knowledge over 2000 learning trials were required [21]. In comparison our method produces a nearly complete mapping by 200 trials.

Regarding accuracy, the model can easily match reported infant accuracy [11], [22]. In the mapping, the average error in saccading to a given image location is $0.35R$ for field radius $R$ for double overlapping fields but note that multilateralation can be performed with a simple linear field function to give much higher accuracy if needed. Full details of noise analysis and the effects of overlapping fields on accuracy requires a further paper.

Although this work has no direct link with neuroscience data or models, we note that the resulting system is not incompatible with a neural interpretation and it would be feasible to implement the algorithm as a version based entirely on artificial neural networks.

VII. CONCLUSIONS

Developmental psychology recognises a key characteristic of animal development: the sequencing of development phases where some competencies always precede others. These regularities are known as stages and are believed to be the basis of development processes that underpin the gradual consolidation of control, coordination and competence [23]. The challenge from this viewpoint is in finding effective algorithms that support progressive and qualitative growth in behavioural competence without requiring significant structural change.

This paper shows how psychology can provide guidance for building effective learning algorithms that can both shed insight on the possible mechanisms involved in human behaviour and also have value for applications. Our model draws on the relevant literature and our resulting experimental system is extremely fast, incremental and cumulative in its learning; all desirable characteristics for real-time autonomous agents. This relates to human infant learning and adaptation which has often been observed to be very fast [24]. The simplicity of the method is important in this regard, as the fast performance would not be obtained with conventional large updating procedures. Moreover, three distinct stages of behaviour were shown to emerge from a single process; this shows how qualitative change in behaviour may occur without structural change.

Regarding potential applications, our algorithm provides automatic fixation of stimuli points in a visual field and thus would be very valuable in moving camera applications such as surveillance, monitoring, underwater, and rescue situations, particularly when the system is mobile, temporary or vehicle mounted. The avoidance of any calibration, set-up, or training periods is a great advantage. Many existing methods deem it necessary to establish exact correspondences between video images and the 3D sensed environment and this requires elaborate computations of intrinsic and extrinsic camera geometry, often with the use of calibration objects [25]. These methods can take up to 30 minutes for the
calibration process [26]. Our simpler approach does not need the full camera parameters and yet can handle non-linear image-motor transforms. A patent application is currently in progress.

New work is exploring the system as a substrate for the growth of further behaviours, including corrective saccades, smooth pursuit, head integration and gaze analysis.

The gap between psychological theories of development and our ability to implement working developmental algorithms is much larger than that for cognitive neuroscience. We believe this gap is a very fruitful area for developmental robotics.

VIII. ACKNOWLEDGEMENTS

This work was mainly supported by UK EPSRC grand GR/R69679/01. The first author was also supported by an ORS award.

REFERENCES

Abstract—We have implemented four bio-inspired self-organizing haptic systems based on proprioception on a 12 d.o.f. anthropomorphic robot hand. The four systems differ in the kind of self-organizing neural network used for clustering. For the mapping of the explored objects, one system uses a Self-Organizing Map (SOM), one uses a Growing Cell Structure (GCS), one uses a Growing Cell Structure with Deletion of Neurons (GCS-DN) and one uses a Growing Grid (GG). The systems were trained and tested with 10 different objects of different sizes from two different shape categories. The generalization abilities of the systems were tested with 6 new objects. The systems showed good performance with the objects from both the training set as well as in the generalization experiments, i.e. they mapped the objects according to shape and size and discriminated individual objects. The GCS-DN system managed to evolve disconnected networks representing different clusters in the input space (small cylinders, large cylinders, small blocks, large blocks), and the generalization samples activated neurons in a proper subnetwork in all but one case.

I. INTRODUCTION

When designing a neural network based self-organizing perception system a natural question comes up, namely what kinds of neural network architectures are most suitable to use. A common choice is the self-organizing map (SOM) [18] that we have used in previous work. This is often a very good choice but it suffers from some limitations, e.g. the topological structure is fixed and the number of neurons in the neural network has to be preset by the system designer. Other limitations are that parameters like the learning rate, the initial neighbourhood size and the decreasing rate of the neighbourhood size also have to be set manually by the designer. To address these limitations we have explored and compared three alternative neural network architectures that avoid some or all of these problems and compared them with the SOM in the context of a proprioception based haptic system. These alternative neural network architectures are the Growing Cell Structures (GCS) and the Growing Cell Structures with Deletion of Neurons (GCS-DN) [3][5], and the Growing Grid (GG) [4].

The choice of a haptic perception system as the context for these explorations is due to our extensive experience in this area with investigations of haptic size perception systems [7][8][9][10], of haptic shape perception systems [11][12][13][14], of proprioception based haptic shape/size perception systems [15], and of haptic texture/hardness perception systems [17]. We also have experience in the use of GCS [6] and we have access to a suitable robot platform consisting of an anthropomorphic robot hand, the LUCS Haptic Hand III [16] which has been employed in the experiments for this paper. However, our findings should be equally applicable in systems mimicking other sensory modalities.

The LUCS Haptic Hand III is a five fingered 12 dof anthropomorphic robot hand with 11 proprioceptive sensors (Fig. 1). The wrist can be flexed/extended. The thumb has two and the other fingers have three phalanges. The thumb can be adducted/abduced and separately flexed/extended in the proximal and the distal joints. The other fingers can be separately flexed/extended in their proximal joints whereas the middle and the distal joints are flexed/extended together. This is similar to the human hand. The phalanges are made of plastic pipe segments and the force transmission from the actuators in the forearm are handled by tendons inside the phalanges. All fingers, except the thumb, are mounted directly on the palm. The thumb is mounted on a RC servo, which enables the adduction/abduction. This RC servo is mounted proximally on the palm. Internal potentiometers in the actuators are used as proprioceptive sensors. The resistances of these potentiometers are proportional to the angles of the different joints.

This paper compares four similar bio-inspired haptic size/shape perception systems based on proprioception and the anthropomorphic robot hand LUCS Haptic Hand III. The systems differ in one respect, namely in the kind of self-organizing neural network employed to cluster the input. The first system uses the SOM, the second uses the GCS, the third uses the GCS-DN and the fourth uses the GG.

II. SELF-ORGANIZING ANNS

A. Self-Organizing Map

The SOM consists of a $I \times J$ grid of neurons with a fixed number of neurons and a fixed topology. Each neuron $n_{ij}$ is associated with a weight vector $w_{ij} \in \mathbb{R}^n$. During adaptation the weight vectors for the neurons are adjusted to a degree which is determined by a neighbourhood function $N_{ijc}(t)$ with a size that decreases with time. The adaptation strength $\alpha(t)$ also decreases with time. The SOM variant used in our experiments is a dot product SOM with Gaussian neighbourhood. The adaptation algorithm works as follows:
At time $t$ each neuron $n_{ij}$ receives an input vector $x(t) \in \mathbb{R}^n$. The neuron $c$ associated with the weight vector $w_c(t)$ most similar to the input vector $x(t)$ is selected:

$$c = \arg \max_{c} \{ ||x(t)w_c(t)|| \}$$  \hspace{1cm} (1)

The weight vectors $w_{ij}$ of the neurons $n_{ij}$ are adapted according to:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)N_{ijc}(t)\left[ x(t) - w_c(t) \right]$$ \hspace{1cm} (2)

where $0 < \alpha(t) \leq 1$ is the adaptation strength with $\alpha(t) \to 0$ when $t \to \infty$ and the neighbourhood function $N_{ijc}(t)$ is a Gaussian function decreasing with time.

**B. Growing Cell Structures**

The GCS has a variable number of neurons and a $k$-dimensional topology where $k$ can be arbitrarily chosen. The adaptation of a weight vector in the GCS is done in a similar way as in the SOM, but the adaptation strength is constant over time and only the best matching unit and its direct topological neighbours are adapted. The GCS estimates the probability density function $p(x)$ of the input space by the aid of local signal counters that keep track of the relative frequencies of input signals gathered by each neuron. These estimates are used to indicate proper locations to insert new neurons. The insertion of new neurons by this method will result in a smoothing out of the relative frequencies between different neurons. The advantages of this approach is that the topology of the network will self-organize to fit the input space, the proper number of neurons for the network will be automatically determined and the learning rate and neighbourhood size parameters are constant over time. The basic building block and also the initial configuration of the GCS is a $k$-dimensional simplex. Such a simplex is for $k = 2$ a triangle. The variant of the GCS algorithm used in our experiments works as follows:

The network is initialized to contain $k+1$ neurons with weight vectors $w_i \in \mathbb{R}^n$ randomly chosen. The neurons are connected so that a $k$-dimensional simplex is formed.

At time step $t$ an input vector $x(t) \in \mathbb{R}^n$ activates a winner neuron $c$ for which the following is valid:

$$c = \arg \min_{c} \{ ||x(t) - w_c(t)|| \}$$ \hspace{1cm} (3)

where $|| \cdot ||$ is the Euclidean distance, and the squared distance between the input vector and the weight vector of the winner neuron $c$ is added to a local error variable $E_c$:

$$\Delta E_c = ||x(t) - w_c(t)||^2.$$ \hspace{1cm} (4)

The weight vectors are updated by fractions $\varepsilon_b$ and $\varepsilon_n$ respectively according to:

$$\Delta w_c(t) = \varepsilon_b(x(t) - w_c(t))$$ \hspace{1cm} (5)

$$\forall i \in N_c : \Delta w_i(t) = \varepsilon_n(x(t) - w_i(t)),$$ \hspace{1cm} (6)

where $N_c$ is the set of direct topological neighbours of $c$.

A neuron is inserted if the number of input vectors that have been generated so far is an integer multiple of a parameter $\lambda$. This is done by finding the neuron $q$ with the largest accumulated error and the neuron $f$ among its direct topological neighbours which has the weight vector with the longest distance from the weight vector of the neuron $q$, insert the new neuron $r$ in between, remove the earlier connection $(q, f)$ and connect $r$ with $q$ and $f$ and with all direct topological neighbours that are common for $q$ and $f$. The weight vector for $r$ is interpolated from the weight vectors for $q$ and $f$:

$$w_r = (w_q + w_f)/2.$$ \hspace{1cm} (7)

The local error counters for all neighbours to $r$ are decreased by a fraction $\alpha$ that depends on the number of neighbours of $r$:

$$\forall i \in N_r : \Delta E_i = (-\alpha/|N_r|) \cdot E_i.$$ \hspace{1cm} (8)

The error variable for $r$ is set to the average of its neighbours:
\[ E_r = \langle 1 / |N_r| \rangle \cdot \sum_{t \in N_r} E_i, \quad (9) \]

and then the error variables of all neurons are decreased:

\[ \forall i : \Delta E_i = -\beta E_i \quad (10) \]

In GCS-DN a neuron (or several if that is necessary to keep a consistent topological structure of \( k \)-dimensional simplices) is deleted, provided that the network has reached its maximum size, at the same occasions new neurons are inserted. Thereafter new neurons are inserted again according to the algorithm described above until the network has reached its maximum size again. This process is repeated a preset number of times, in our experiments 250 times.

C. Growing Grid

The GG can be seen as an incremental variant of the SOM. It consists of an \( I \times J \) grid of neurons with a fixed topology but with \( I \) and \( J \) increasing with time as new rows and columns are inserted. In addition to a weight vector \( w_{ij} \in R^n \) each neuron \( n_{ij} \) also has a local counter variable \( T \) to estimate where to insert new rows or columns of neurons in the grid. The self-organizing process of a GG is divided into two phases: a growth phase and a fine-tuning phase. During the growth phase the grid grows by insertion of new rows and columns until the wanted size of the network has been achieved. During the fine-tuning phase, the network size does not change and a decreasing adaptation strength \( \alpha(t) \) is used. The size of the neighbourhood is not decreasing with time. Instead the network is growing with a constant neighbourhood size and therefore the fraction of all neurons that are adapted decreases over time. The variant of the GG algorithm used in our experiments is described below:

Growth Phase:

Initialize the network to contain \( 2 \times 2 \) neurons with weight vectors randomly chosen.

At time \( t \) an input vector \( x(t) \in R^n \) is generated and received by each neuron \( n_{ij} \) in the grid.

The neuron \( c \) associated with the weight vector \( w_c(t) \) most similar to the input vector \( x(t) \) is selected:

\[ c = \arg \max_c \{ ||x(t) - w_c(t)|| \} \quad (11) \]

Increment the local counter variable \( T_c \) for \( c \):

\[ T_{c} = T_{c} + 1 \quad (12) \]

The weight vectors \( w_{ij} \) of the neurons \( n_{ij} \) are adapted according to:

\[ w_{ij}(t + 1) = w_{ij}(t) + \alpha N_{ijc} [x(t) - w_{ij}(t)] \quad (13) \]

where \( 0 \leq \alpha \leq 1 \) is the adaptation strength and the neighbourhood function \( N_{ijc} \) is a Gaussian function. Notice that \( \alpha \) and \( N_{ijc} \) are not functions of \( t \) though.

A new row or column is inserted if the number of input vectors that have been generated so far is an integer multiple \( \lambda \) of the current number of neurons in the grid. This is done by finding the neuron \( q \) with the largest value of the local counter variable \( T \) and the neuron \( f \) among its direct topological neighbours which has the weight vector with the longest distance from the weight vector of the neuron \( q \). Depending on the relative positions of \( q \) and \( f \) a new row or a new column is inserted.

If \( q \) and \( f \) are in the same row, then a new column is inserted between the columns of \( q \) and \( f \). The weight vectors for the new neurons are interpolated from their direct neighbours in the same row.

If \( q \) and \( f \) are in the same column, then a new row is inserted between the rows of \( q \) and \( f \). The weight vectors for the new neurons are interpolated from their direct neighbours in the same column.

Adjust \( I \) or \( J \) to reflect the new numbers of rows and columns in the grid.

Reset all local counter values:

\[ T_{n_{ij}} = 0 \quad (14) \]

If desired network size has not been reached, then go to step 2, i.e. generate a new input vector.

Fine-tuning Phase:

This phase is similar to the growth phase but the adaptation strength \( \alpha(t) \) is now decreasing with time and no insertions of new rows or columns are done. This phase stops after a preset number of iterations.

III. MODELS

The models differ in one respect, namely the kind of self-organizing neural network employed. The models consist of the LUCS Haptic Hand III, sensory and motor drivers, a Self-Organizing Neural Network (SO-ANN) and a commander module to commands appropriate for the robot hands servo controller board. When the commander executes a grasp, and the robot hand is fully closed around the object, the sensory driver scans the 11 proprioceptive sensors when requested to do so by the commander module, while the motor driver translates high level motor commands from the commander module to commands appropriate for the robot hands servo controller board. The sensory driver scans the 11 proprioceptive sensors and outputs an eleven-elements vector to the SO-ANN, which is adapted.

The SOM model uses a 225 neurons dot product SOM with plane topology, which uses softmax activation with the softmax exponent equal to 10 [1]. It is trained by 2000 iterations.

The GCS model grows, by inserting a new neuron every 19th iteration, until a size of 225 neurons has been reached.
The GCS-DN model grows until a size of 225 neurons has been reached, also by inserting a new neuron every 19th iteration, then the deletion/insertion process described in section 2.2 is repeated 250 times. Finally this yields a number of disconnected networks with altogether 225 neurons.

The GG model grows by inserting a new row or column each time the number of time steps $t$ since the previous insertion equals a multiple $\lambda$ of the current grid size, i.e. until $t = \lambda \cdot J$ with $\lambda = 19$. The growth phase lasts until a minimum grid size of 225 neurons has been reached, then the model runs in fine tuning mode for 1000 iterations.

IV. TESTING THE MODELS

We have trained the models with 10 objects (see Table 1 objects a-j). These objects are either cylinder shaped or block shaped. There are five objects of each shape category. All objects are sufficiently high to be of a non-variable shape in those parts grasped by the robot hand, e.g. a bottle is grasped on the part of equal diameter below the bottle neck.

During the grasping tests the test objects were placed on a table with the open robot hand around them. If the objects were block shaped we always placed the longest side against the palmar side of the robot hand.

To simplify the testing procedure each object was grasped 5 times by the robot hand, i.e. in total 50 grasps were carried out, and the sensory information were written to a file. Then the SO-ANN were trained and tested with this set of 50 samples. The training phase for the SOM model lasted for 2000 iterations. The GCS model was trained until a network size of 225 neurons was reached. The GCS-DN model was trained until a network size of 225 neurons was reached and then the insertion/deletion process described in section 2.2 was repeated 250 times. The GG model was trained with a growth phase which lasted until the minimal network size reached 225 neurons, and then for 1000 iterations in fine tuning mode.

Each fully trained model was tested with the original training set and in addition with three new block shaped and three new cylinder shaped objects of variable sizes (see Table 1, objects 1-6).

V. RESULTS

The results are depicted in Fig. 3. Fig 3A shows the centres of activation in the SOM in the fully trained SOM model when tested with the training set and the test set. The SOM seems to be organized according to shape. Four groups of objects can be distinguished in the map, large block shapes, small block shapes, large cylindrical shapes and small cylindrical shapes. The SOM also seems to be organized in a clockwise manner according to size. The result of the generalization experiment shows that all test objects are mapped so that they are ordered according to size in the same way as the objects in the training set, and that they are also correctly mapped according to shape. The activations in the SOM also indicate that it is possible to discriminate individual object of the training set to a large extent and this is also true for the test objects, since each of the test objects is also mapped so that it can be identified as the most similar object of the training set. The results with the SOM model are thoroughly described in [15].

Fig 3B shows the centres of activation in the GCS in the fully trained GCS model. Only the part of the GCS which is activated by some object is shown in the figure. This model produces similar results as the SOM model, i.e. the organization of the GCS separates large block shapes, small block shapes, large cylinder shapes and small cylinder shapes. The GCS is also organized according to size with the smallest objects represented uppermost in the GCS and the largest in the lowest part. The ability for discrimination of individual objects is approximately similar as that for the SOM model. Also this model activates neurons at proper locations when fed with the objects of the generalization test set.

Fig 3C shows the final network structure of the fully trained GCS-DN model. As can be seen this network structure consists of several disconnected subnetworks. This is due to the removal of neurons that represent parts of the input space with a low value of the probability density function. As a result, such a network tends to self-organize into subnetworks that represent different clusters in the input space. This is also what happened in our experiments. As indicated in the figure one or more subnetworks can be seen as representing one of the categories large block shapes,
Fig. 3. The test results of the four models. A: The SOM model is organized according to shape and size. Groups of large blocks, small blocks, large cylinders and small cylinders can be distinguished. The activations tend to be located according to size of the objects in a clockwise manner. Individual objects can be discriminated to a large extent. B: The GCS model produces similar results as the SOM model, i.e. it is organized according to shape and size. The GCS model separate large blocks, small blocks, large cylinders and small cylinders, and the objects are represented according to size with the smallest objects uppermost and the largest lowermost in the GCS. Also individual objects can be discriminated to a large extent. C: The GCS-DN model self-organized into sub networks, where one or more sub networks represent the categories large blocks, small blocks, large cylinders and small cylinders. D: The GG model separate large blocks, small blocks, large cylinders and small cylinders, and the grid is organized according to size. Individual objects can be discriminated to a large extent. The 6 test objects (indicated with the numbers 1-6) not included in the training set activated neurons at proper locations perfectly in all models but the GCS-DN model. In that model object 1 triggered activation in the wrong subnetwork. (See Table 1 for the meaning of the labels).
small block shapes, large cylinder shapes and small cylinder shapes. The objects of the generalization test set activate neurons in the proper subnetworks except in one case, namely the test object 1 is a large block but is identified as a large cylinder.

Fig 3D shows the centres of activation in the GG in the fully trained GG model. This model produces similar results as the SOM model and the GCS model, i.e. the organization of the GG separates large block shapes, small block shapes, large cylinder shapes and small cylinder shapes. As indicated in the figure the GG is also organized according to size. The ability for discrimination of individual objects is approximately similar as that for the SOM model. All 6 objects of the generalization test set are mapped so that they can be associated with the correct shape category and identified with the most similar object of the training set.

VI. DISCUSSION

We have experimented with four self-organizing models for clustering of proprioceptive data collected by our anthropomorphic robot hand, the LUCS Haptic Hand III. All four models were able to cluster the sensory information according to shape, and all four of them resulted in networks which preserves the size ordering of the training objects. The models have proven to have an excellent generalization capacity. This is clearly illustrated in the categorization of the 6 new objects that offered different characteristics of shape and size.

The SOM, the GCS and the GG performed at approximately a similar level. This could be an argument for using the alternative neural network architecture GCS and GG instead of the SOM, because that reduces the number of parameters that have to be set. According to Fritzke [2] the performance of the GCS is actually slightly better than the performance of the SOM in complex and realistic problems. The results of our experiments in [6] also points in that direction.

The GCS and the GCS-DN also have the virtue to get organized into networks whose topology reflect the probability density function of the input space. The GCS-DN is especially interesting since it has the property to automatically form disconnected subnetworks that represent clusters in the input space. It should be possible to implement an online version of the GCS-DN algorithm that never stops and that should result in a set of networks, that reflects the probability density function of the input space, which changes if the probability density function happens to be non-stationary. In other words, if the probability density function of the input space changed then the set of subnetworks would change by the deletion of some subnetworks and the split, followed by growth of others.

It should be mentioned that the graphical presentation of GCS and GCS-DN could be improved. Fritzke [3] suggests a method on how to embed these kinds of networks in the plane for better visualizations. In this method a physical model is maintained where the neurons are considered as discs influenced by attractive and repulsive forces.

The success with the GCS, the GCS-DN and the GG suggests an increased focus on our part on these kinds of self-organizing neural networks. The advantage of getting rid of several parameter settings like network size, learning rate and neighbourhood settings can be important to succeed with more complex cognitive models with several coupled neural networks at multiple levels. To be forced to set all the parameters in a good way for all included neural networks with complex dependencies in such a model could prove to be overwhelming.

In the future we plan to increase the use of neural networks like GCS and GG as an alternative to the SOM in our haptic systems. By doing so we will reduce the number of parameters that have to be set explicitly and this should yield more robust systems.

VII. ACKNOWLEDGMENTS

We want to acknowledge the support from Stiftelsen Landshövding Per Westlings Minnesfond for financial support.
REFERENCES


Distributed Control of Multi-Robot Systems using Bifurcating Potential Fields

Derek Bennet and Colin McInnes
Department of Mechanical Engineering
University of Strathclyde
Glasgow, G1 1XJ
{derek.bennet, colin.mcinnnes}@strath.ac.uk

Abstract—The distributed control of multi-robot systems has been shown to have advantages over conventional single robot systems. These include scalability, flexibility and robustness to failures. This paper considers pattern formation and reconfigurability in a multi-robot system using bifurcating potential fields. It is shown how various patterns can be achieved through a simple free parameter change. In addition the stability of the system of robots is proven to ensure that desired behaviours always occur.

I. INTRODUCTION

Over the past two decades distributed robot systems have been developed as a method of solving a variety of engineering problems [1]. Most of the research in this area is influenced by the early work of Brooks [2] in the mid 1980’s who introduced the concept of behavioural robotics. Although the majority of previous research had been concerned with single robot systems it was suggested that a significant step forward would be to draw on inspiration from nature and utilise the idea of emergent behaviour through decentralised control. This form of control has the advantages of being robust, scalable and flexible and has been applied to areas such as surveillance, exploration and transportation [3], [4].

The purpose of this paper is to investigate the distributed control of multi-robot pattern formation and reconfigurability. To achieve this we consider the use of the artificial potential field method and extend previous research by considering bifurcation theory in order to have a reconfigurable formation. Dynamical systems theory is used to demonstrate mathematically the stability of the system so that desired behaviours always occur.

Artificial potential fields were first introduced in Khatib [5] in the area of obstacle avoidance for manipulators and mobile robots. More recently they have been applied successfully in the area of autonomous robot motion planning [6], [7], as a form of distributed behavioural control in [8] and in space applications [9], [10], [11]. The basic idea behind potential field theory is to create a workspace where a robot is attracted towards a goal state with a repulsive potential ensuring collide avoidance [7]. As a multi-robot team may be required to achieve different tasks a desirable property of the system would be reconfigurability. In order to minimise computational expense bifurcation theory can be used to reconfigure the formation through a simple free parameter change.

For real, safety critical applications it is essential that the behaviour of each robot is verified in order to ensure that no unwanted behaviours will occur. Winfield [12] has introduced the term ‘swarm engineering’ to highlight the key issues that are involved in real, safety critical applications as opposed to those based on simulation. Through the use of dynamical systems theory this paper aims to replace algorithm validation with mathematical proof in order to prove that desired formations always occur.

The paper proceeds as follows. In the next section we describe the model used and explain the artificial potential field method and bifurcation theory. We also discuss the linear and non-linear stability of the models developed. Section III shows the numerical results of simulations carried to demonstrate pattern formations and reconfigurability.

II. FORMATION MODEL

We consider a system of homogeneous autonomous robots (1 ≤ i ≤ N) interacting via an artificial potential function U. It is assumed that all robots can communicate with each other and are fully actuated. The negative gradient of the artificial potential defines a virtual force acting on each robot so that the dynamics of each robot can be described by Eq. 1 and 2 with mass, m, position, x_i, and velocity, v_i:

\[
\frac{dx_i}{dt} = v_i
\]

(1)

\[
m \frac{dv_i}{dt} = -\nabla_i U^S (x_i) - \nabla_i U^R (x_{ij}) - \sigma v_i
\]

(2)

From Eq. 2 it can be seen that the virtual force experienced by each robot is dependent upon the gradient of two different artificial potential functions and a dissipative term, where \( \sigma > 0 \) controls the amplitude of the dissipation. The first term in Eq. 2 is defined as the steering potential, \( U^S \) which will control the formation, whereas the second term in Eq. 2 is the collision avoidance pairwise repulsive potential, \( U^R \).
The repulsive potential is based on a generalized Morse potential [13] as shown in Eq. 3:
\[
U_R^{ij} = \sum_{j,j \neq i} C_r \exp^{-|x_i - x_j|/L_r}
\]  
(3)

Where \(C_r\) and \(L_r\) represent the amplitude and length-scale of repulsive potential respectively and \(|x_i - x_j|\) is the distance between the positions of the two robots.

The total repulsive force on the \(i^{th}\) robot is dependent upon the position of all the other \((N-1)\) robots in the formation. The repulsive potential is therefore used to ensure that as the robots are steered towards the goal state they do not collide with each other. Once all the robots have been driven to the desired equilibrium state the repulsive potential also ensures that they are equally spaced for symmetric formations.

A. Artificial Potential Function Scale Separation

As noted in the previous section the force experienced by each robot is dependent upon the gradient of two different artificial potential functions. The steering potential is a function of position only. We now consider a supercritical pitchfork bifurcation equation in Eq. 4, with bifurcation parameter \(\mu\) and length scale \(R\). A detailed explanation of this steering potential is given in section II B. The repulsive potential noted in the previous section is given in Eq. 5:
\[
U_S = -\frac{1}{2}\mu(X - R)^2 + \frac{1}{4}(X - R)^4
\]  
(4)

\[
U_R = C_r \exp^{-X/L_r}
\]  
(5)

For illustration we consider a simple 1-dimensional system with position coordinate \(X\).

Defining an outer region dependent upon the steering potential only and an inner region dependent upon the repulsive potential only we can show that these two regions are separated so that the robot move under the influence of the long-range steering potential, but with short range collisions (for \(L_r/R < < 1\)) effectively creating a boundary layer between them. This can then be used to determine the non-linear stability of the system considering the steering potential only.

For 1D motion of a robot of mass \(m\) we have:
\[
m \frac{dV}{dt} = -\frac{dU^R}{dX} - \frac{dU^S}{dX} - \sigma V
\]  
(6)

So that,
\[
m \frac{dV}{dX} = \frac{C_r}{L_r} \exp^{-X/L_r}
\]  
(7)

\[
\frac{1}{R} mV \frac{dV}{dS} = \frac{C_r}{L_r} \exp^{-S/L_r}
\]  
(8)

\[
+ \mu R(S - 1) - R^3(S - 1)^3 - \sigma V
\]

Now define \(\varepsilon = \frac{L_r}{R} < < 1\) so that;

\[
m \frac{dV}{dS} = \frac{C_r}{\varepsilon} \exp^{-S/\varepsilon}
\]

\[
+ R \left[ \mu R(S - 1) - R^3(S - 1)^3 - \sigma V \right]
\]  
(9)

Let \(\varepsilon \to 0\) in order to consider ‘far field’ dynamics which forms a singularly perturbed system;

\[
\lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \exp^{(-S/\varepsilon)} = 0
\]  
(10)

Therefore at large separation distances the repulsive potential vanishes and we can consider the steering potential only when considering the stability of analysis of the system.

Conversely if we define \(\overline{S} = \frac{S}{\varepsilon}\) we find that the ‘near field’ dynamics are defined by;

\[
m \frac{dV}{dS} = C_r \exp^{-\overline{S}}
\]

\[
+ \varepsilon R \left[ \mu R(S - 1) - R^3(S - 1)^3 - \sigma V \right]
\]  
(11)

and letting \(\varepsilon \to 0\);

\[
m \frac{dV}{dS} = C_r \exp^{-\overline{S}}
\]  
(12)

Thus, at small separations the steering potential vanishes and we can treat the collisions separate in the subsequent stability analysis.

B. 1-Parameter Static Bifurcation

Referring back to Eq. 2 the steering potential can be based on a supercritical pitchfork bifurcation [14] as shown in Eq. 13. The aim of this potential is to drive each robot to a goal distance, \(r\), from the origin in the \(x-y\) plane thus forming a symmetric ring.

\[
U_S(x_i; \mu, \alpha) = -\frac{1}{2}\mu (\rho_i - r)^2 + \frac{1}{4} (\rho_i - r)^4
\]  
(13)

Where \(\rho_i = (x_i^2 + y_i^2)^{0.5}\).

Depending on the sign of \(\mu\), the steering potential can have two distinct forms. Fig. 1 shows how the potential bifurcates from a single local minimum into two local minima when \(\mu = 0\), while Fig. 2 shows the shape of the potential when \(\mu < 0\) and \(\mu > 0\).
The equilibrium states of the potential occur whenever \( \partial U / \partial \rho_i = 0 \). Therefore:

\[
\frac{\partial U}{\partial \rho_i} = -\mu (\rho_i - r) + (\rho_i - r)^3
\]  

(14)

If \( \mu \leq 0 \) equilibrium occurs when \( \rho_i = r \). If \( \mu > 0 \) equilibrium occurs when \( \rho_i = r, r \pm \sqrt{\mu} \). Therefore, a single ring will bifurcate to a double ring using \( \mu \) as a control parameter.

The stability of the potential is determined from the sign of the second derivative, given in Eq. 15, and summarised in Table I:

\[
\frac{\partial^2 U}{\partial \rho_i^2} = -\mu + 3(\rho_i - r)^2
\]  

(15)

**TABLE I**

<table>
<thead>
<tr>
<th>Bifurcation parameter, ( \mu )</th>
<th>Equilibrium position, ( \rho_{eq} )</th>
<th>( \partial^2 U / \partial \rho_i^2 )</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; 0)</td>
<td>( r )</td>
<td>( &gt; 0 )</td>
<td>stable minimum</td>
</tr>
<tr>
<td>( &gt; 0)</td>
<td>( r )</td>
<td>( &lt; 0 )</td>
<td>unstable maximum</td>
</tr>
<tr>
<td>( r + \sqrt{\mu} )</td>
<td>( &gt; 0 )</td>
<td></td>
<td>stable minimum</td>
</tr>
<tr>
<td>( r - \sqrt{\mu} )</td>
<td>( &gt; 0 )</td>
<td></td>
<td>stable minimum</td>
</tr>
</tbody>
</table>

1) Linear stability: 1-parameter static bifurcation: In order to determine the linear stability of a system of robot subject to such a 1-parameter bifurcation steering potential we perform an eigenvalue analysis on the linearized equations of motion assuming that at large separation distances the repulsive potential can be neglected through scale separation as explained in section II A. The linear stability analysis will be used to determine the local behaviour of the system by calculating its eigenvalue spectrum. Therefore, the equations of motion for the model are re-cast as:

\[
\begin{pmatrix}
\dot{x}_i \\
\dot{v}_i
\end{pmatrix}
= \begin{pmatrix}
-\sigma v_i - \nabla_i U^S(x_i) \\
f(x_i, v_i) \\
g(x_i, v_i)
\end{pmatrix}
\]

(16)

Let \( x_o \) and \( v_o \) denote fixed points with \( \dot{x}_i = \dot{v}_i = 0 \) so that:

\[
f(x_o, v_o) = 0
\]

(17)

\[
g(x_o, v_o) = 0
\]

(18)

Thus, \( v_o = 0 \) and \( \nabla U^S = 0 \) at equilibrium. This occurs when \( \rho_o = r \) if \( \mu < 0 \) and \( \rho_o = r, r \pm \sqrt{\mu} \) if \( \mu > 0 \). Defining \( \delta x_i = x_i - x_o \) and \( \delta v_i = v_i - v_o \) and Taylor Series expanding about the fixed points to linear order the eigenvalues of system can be found using:

\[
\begin{pmatrix}
\delta x_i \\
\delta v_i
\end{pmatrix}
= J \begin{pmatrix}
\delta x_i \\
\delta v_i
\end{pmatrix}
\]

(19)

where,

\[
J = \begin{pmatrix}
\frac{\partial f(x_i, v_i)}{\partial x_i} & \frac{\partial f(x_i, v_i)}{\partial v_i} \\
\frac{\partial g(x_i, v_i)}{\partial x_i} & \frac{\partial g(x_i, v_i)}{\partial v_i}
\end{pmatrix}
\]

(20)

The Jacobian, \( J \), is then a 2x2 matrix given by;

\[
J = \begin{pmatrix}
0 & 1 \\
\frac{\partial f}{\partial x_i} & -\sigma
\end{pmatrix}
\]

(21)

Substituting a trial exponential solution into Eq. 19 we find that:

\[
\begin{pmatrix}
\delta x_i \\
\delta v_i
\end{pmatrix}
= \begin{pmatrix}
\delta x_i \\
\delta v_i
\end{pmatrix} e^{\lambda t}
\]

(22)

Therefore, the eigenvalues, \( \lambda \), of the system are found from \( det(J - \lambda I) = 0 \).

As shown previously, if \( \mu < 0 \) equilibrium of the system occurs when \( x_o = (r, 0) \) and \( v_o = 0 \). Evaluating the Jacobian matrix given in Eq. 21 we find that:

\[
J = \begin{pmatrix}
0 & 1 \\
\mu & -\sigma
\end{pmatrix}
\]

(23)

The corresponding eigenvalue spectrum is therefore:

\[
\lambda = 1/2(\sigma \pm \sqrt{\sigma^2 + 4\mu})
\]

(24)

As \( \sigma > 0 \) and \( \mu < 0 \) the eigenvalues are always either negative real or complex with negative real part as and \(-\sigma \pm \sqrt{\sigma^2 + 4\mu} \neq 0 \). The equilibrium position can therefore be considered as linearly stable.
If $\mu > 0$ equilibrium of the system occurs when $x_{o1} = (r, 0)$, $x_{o2} = (r + \sqrt{\mu}, 0)$ and $x_{o3} = (r - \sqrt{\mu}, 0)$ with $v_i = 0$. The Jacobian matrix evaluated at the three different equilibrium positions is given by Eq. 25, 26 and 27 respectively as:

$$J_1 = \begin{pmatrix} 0 & 1 \\ \mu & -\sigma \end{pmatrix}$$ (25)

$$J_2 = \begin{pmatrix} 0 & 1 \\ -2\mu & -\sigma \end{pmatrix}$$ (26)

$$J_3 = \begin{pmatrix} 0 & 1 \\ -2\mu & -\sigma \end{pmatrix}$$ (27)

The eigenvalues for $J_1$ are:

$$\lambda = 1/2 \left( -\sigma \pm \sqrt{\sigma^2 + 4\mu} \right)$$ (28)

Considering the pair of eigenvalues in Eq. 28 we can show that $-\sigma \pm \sqrt{\sigma^2 + 4\mu} > 0$ since, $\sigma^2 + 4\mu > \sigma^2$ and therefore we always have at least one positive real eigenvalue. This equilibrium position is therefore always linearly unstable.

The eigenvalues for $J_2$ and $J_3$ are:

$$\lambda = 1/2 \left( -\sigma \pm \sqrt{\sigma^2 - 8\mu} \right)$$ (29)

Again as $\sigma > 0$ and $\mu > 0$ the eigenvalues are always either negative real or complex with negative real part as $-\sigma \pm \sqrt{\sigma^2 - 8\mu} < 0$. The equilibrium positions can therefore be considered as linearly stable.

2) Non-linear stability: 1-parameter static bifurcation:

To determine the non-linear stability of the dynamical system we consider Lyapunov’s Second Theorem as expressed by Kalman and Bertram[15], [16];

“If the rate of change of $dE(x)/dt$ of the energy $E(x)$ of an isolated physical system is negative for every possible state $x$, except for a single equilibrium state $x_e$, then the energy will continually decrease until it finally assumes its minimum value $E(x_e)$.”

The aim of the steering potential is to drive the robot to the desired equilibrium position that corresponds to the minimum potential. Therefore, if Lyapunov’s method can be used for the system, as time evolves the system will relax into the minimum energy state.

The Lyapunov function, $L$, is defined as the total energy of the system, where $U^S(x_i)$ is given in Eq. 13 so that for unit mass;

$$L = \sum_i \left( \frac{1}{2} v_i^2 + U^S(x_i) \right)$$ (30)

Where, $L > 0$ other than at the goal state when $L = 0$.

The rate of change of the Lyapunov function can be expressed as;

$$\frac{dL}{dt} = \left( \frac{\partial L}{\partial x_i} \right) \dot{x}_i + \left( \frac{\partial L}{\partial v_i} \right) \ddot{v}_i$$ (31)

Then, substituting Eq. 16 into Eq. 31 it can be seen that;

$$\frac{dL}{dt} = -\sigma \sum_i v_i^2 \leq 0$$ (32)

From Lyapunov’s Second Theorem [14] it states that if $L$ is a positive definite function and $\dot{L}$ is a negative definite the system will be uniformly stable. A problem arises in the use of superimposed artificial potential functions as $\dot{L} \leq 0$. This implies that $\dot{L}$ could equal zero in a position other than the goal minimum suggesting that the system may become trapped in a local minimum. In order to ensure that our system is asymptotically stable at the desired goal state the LaSalle principle [17] can be used. It extends the above constraints to state that if $L(0) = \dot{L}(0) = 0$ and the set $\{x_i|L < 0\}$ only occurs when $x_i = x_e$, then the goal state is asymptotically stable. Therefore, for the quadratic potential considered in this paper the LaSalle principle is valid. As we have a smooth well defined symmetric potential field, equilibrium only occurs at the goal states so the local minima problem can be avoided and the system will relax into the desired goal position.

C. 2-parameter Static Bifurcation

An extension to the 1-parameter pitchfork bifurcation is to consider 2-parameter bifurcations such as the so-called cusp catastrophe given in Eq. 33. Fig. 3 shows the variation of the equilibrium position with the two parameters, $\mu_1$ and $\mu_2$.

$$U(\rho_i; \mu_1, \mu_2) = \mu_1 (\rho_i - r)^2 + (\rho_i - r)^4 + \mu_2 (\rho_i - r)$$ (33)

Fig. 3. Cusp catastrophe surface [18]

Mapping the cusp potential onto the $\mu_1 - \mu_2$ plane we can see how the system behaviour changes for different bifurcation parameters as shown in Fig. 4, which is similar to a phase diagram for water for example. As pressure and temperature are varied different phases can be achieved[19] which is analogous to the different patterns we can achieve as the bifurcation parameters are altered.
length-scale of the dissipation; C aligning the velocity vectors of members of the swarm where 36 shows the orientation term that dissipates energy whilst dissipative orientation term as shown in Eq. 34 and 35. Eq. 1 and 2 are now modified to include a field methods. Eq. 1 and 2 are now modified to include a

\[ \sum_{i \neq j} C_{ij} (\mathbf{x}_i - \mathbf{x}_j) \exp(-|\mathbf{x}_i|/L_j) \mathbf{x}_i \]

The emergence of vortex like formations can be seen through the conservation of angular momentum, H:

\[ \sum_i \mathbf{x}_i \times \mathbf{v}_i = \frac{d}{dt} \sum_i (\mathbf{x}_i \times m \mathbf{v}_i) \]

\[ \frac{d\mathbf{H}}{dt} = 0 \] (37)

It can also be shown that as time evolves the system of robots will relax into the minimum energy, E, state where \( \sum_i \mathbf{v}_i \cdot \Lambda_i = 0 \). The swarm therefore dissipates energy while conserving angular momentum and so relaxes into the rotating ring[20] where \( \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} = 0 \).

III. NUMERICAL RESULTS

A. Static Bifurcation Formation Patterns

Fig. 6 shows the three different robot formations that can be formed using a 1 parameter static bifurcation. The system considers a swarm of 30 robots with unit mass and \( \sigma = 10 \).

D. Rotation of the Formation

Recent work by McInnes [20] has shown how vortex like swarming can be achieved through artificial potential field methods. Eq. 1 and 2 are now modified to include a dissipative orientation term as shown in Eq. 34 and 35. Eq. 36 shows the orientation term that dissipates energy whilst aligning the velocity vectors of members of the swarm where \( C_0 \) and \( L_0 \) are constants representing the magnitude and length-scale of the dissipation;

\[ \frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i \] (34)

\[ m \frac{d\mathbf{v}_i}{dt} = -\Lambda_i - \nabla_i U^S(\mathbf{x}_i) - \nabla_i U^R(\mathbf{x}_i_j) \]

\[ \Lambda_i = \sum_{i \neq j} C_{ij} (\mathbf{x}_i - \mathbf{x}_j) \exp(-|\mathbf{x}_i|/L_j) \mathbf{x}_i \]

The first formation corresponds to the case when \( \mu = -4 \) and \( r = 0 \). The robots are driven towards the origin with the repulsive potential ultimately causing a uniform cluster to form. The second formation consists of a ring with the radius of the ring determined by the magnitude that the steering potential has been moved along the \( \rho_c \)-axis (in this case \( r = 3 \)). The final formation consists of two rings with
\( \mu = 1.5 \). The stable equilibrium state in the second formation has become unstable and the system bifurcates into two rings.

### B. 1-Parameter Static Bifurcation

Figures 7 shows the transition of a formation of 30 robots in the x-y plane. As it can be seen, the system changes from a ring to two rings to a cluster then back to a ring. This is achieved through a simple parameter change and is one of the advantages of using the pitchfork bifurcation equation as a basis for the artificial potential function. Rather than controlling each robot individually the global pattern of the formation can be manipulated via \( \mu \).

![Fig. 7. Transition between different formations with \( \sigma = 1 \)](image)

### C. 2-Parameter Dynamic Bifurcation

Figure 8 demonstrates how a 2-parameter bifurcation can be used to manipulate a robot formation. As can be seen if we start in the two ring case when \( \mu_1 = -2 \) and \( \mu_2 = 0 \) and then vary \( \mu_2 \), therefore performing a bifurcation on the system, we can either tip the system into a large or small ring.

![Fig. 8. Evolution of cusp catastrophe results](image)

### D. Rotation of the Static Bifurcation Formation

Figures 9 shows the rotation of the ring formation using Eq. 36. The formation relaxes into to a single ring and conserves angular momentum by rotating about its centre of mass.

![Fig. 9. Time evolution of vortex ring](image)

### IV. Conclusion

We have shown that the control of a multi-robot system can be achieved through the use of the artificial potential function method. We have extended previous research in this area through the use of bifurcation theory to demonstrate that through a simple parameter change a formation of robots can be made to alter their configuration and shown how 1 and 2 parameter static bifurcations can be used to this effect. An important step in real engineered systems is to ensure that the formation can form reliably. Through dynamical systems theory we have demonstrated the stability of a system of robots driven to the equilibrium position to ensure that desired behaviours always occur. Future work will consider generalising the potential function method in order to achieve arbitrary patterns whilst also considering nonholonomic constraints in order to make the model more realistic.

### References


Combining Coordinated Navigation and Reactive Collision Avoidance for GPS-based Convoying

Frank Hoeller, Timo Röhl, and Achim Königs
Research Establishment for Applied Science, Wachtberg, Germany

Abstract—This paper presents a navigation system, which is able to control a GPS-waypoint guided convoy of transportation robots. In contrast to common reactive approaches to convoying our system uses delayed communication of GPS-waypoints for the motion coordination of the robots. This way oscillatory behavior of the robots in the convoy can be prevented to a large extent. In order to circumnavigate obstacles that could not be taken into account during waypoint planning, the system employs a probabilistic roadmap technique to plan collision-free paths. An enhanced Kalman filter which integrates the measurements of the low-cost sensors is utilized for estimating the global position of all robots. We tested our method on real robots and in simulations. The experiments carried out in a common outdoor environment confirm that the system generates a stable convoy and every robot avoids obstacles reliably while keeping the convoy connected even in unexpected situations.

I. INTRODUCTION

Transportation tasks belong to the principal application areas for autonomous mobile robots. One of the more challenging problems in this domain is the coordination of a heterogeneous team of mobile robots to carry out convoying missions. This task requires that the robots maintain an in-line formation with a certain distance between the vehicles, while safely avoiding obstacles and staying on drivable ground. An especially demanding situation arises, if some of the robots have only limited sensor equipment that does not allow to directly detect the drivable ground. These robots have to rely on the sensing capabilities of the leading robots instead. The challenge here is to achieve a high precision navigation, which enables the following vehicles to reliably follow the defined route. However, collision avoidance cannot be left out of focus completely. Unforeseen or suddenly appearing obstacles should still be avoided and circumnavigated if possible.

In this article we present an integrated waypoint navigation and collision avoidance approach that enables a group of mobile robots to continuously follow a predefined route. The route is specified by a list of waypoints which can be created manually beforehand or automatically on-line by the leading vehicle or a leading person. The waypoint navigation system is divided into two components: (1) The global navigation system that translates global waypoints into destination points relative to a robot’s position, based on global position data provided by an affordable class-II GPS-receiver in combination with an electronic compass; (2) The local navigation that generates sequences of robot motion commands to the destination coordinate. The obstacle detection and avoidance is also carried out by the local navigation. To establish the convoying functionality waypoints are communicated between robots with the following passing-technique: When a robot reaches a waypoint, the coordinate of the waypoint is transmitted to the next robot in line and added to its waypoint list. To ensure a smooth convoy-like behavior, intermediate targets are generated by linear interpolation between the waypoints given to the robots.

The local navigation of a robot tries to reach a given waypoint solely based on the robot’s sensory input, i.e. without use of a map of the environment. In our current implementation, each robot uses a 2D laser range scanner with 180 degree field of view and its odometry for this purpose. The planning of the motion controls for a robot is performed based on an expansive-spaces tree algorithm (EST) [1], [2], [3], a randomised roadmap technique, which we integrated into the local navigation. This combination was originally developed for people accompanying in indoor environments [4] but turned out to work in moderate outdoor scenarios as well. The EST repeatedly computes collision-free paths from the robot’s current position to a given destination by carrying out a randomised search in configuration space. This search is guided by a heuristic cost function similar to A* search. In this manner the EST algorithm is able to plan trajectories and guide the robot to its target location.

However, since the odometry is not reliable, especially in outdoor applications, and since the local navigation approaches destination points based on a robot-centric coordinate system, the global navigation has to detect and compensate emerging deviations from the intended route. For this purpose, the global positions of each robot are estimated based on the GPS, compass, and odometry data. The data is fused using a modified Kalman filter,
which is sufficient for our purpose.

The remainder of this article is organised as follows. After discussing related work in Section II, we introduce the local navigation with the modified expansive-spaces tree algorithm in Section III. Section IV then presents the global navigation in detail including the Kalman filter used for position estimation. Before we conclude, we describe several experiments carried out in simulations and on real robots to illustrate the capabilities and the robustness of our approach.

II. RELATED WORK

Since the first robots left the laboratory and entered rougher terrains robotic science was interested in waypoint navigation. Some approaches e.g. [5] or [6] used local navigation strategies with either no or simple try-and-error-based obstacle avoidance algorithms. These methods fail if there are obstacles that were not taken into account during the waypoint generation or they need a very long time to circumnavigate these obstacles. Other attempts use a more sophisticated navigation technique. Langer et al. [7] and Lacaze et al. [8] define a limited set of predefined commands or command-sequences and greedily decide in every computing interval which entity should be applied. One could think that it might be possible to reach every potential destination with such methods, but once one command or sequence is chosen, it renders many locations unreachable because the deceleration ability of every vehicle is limited. If the terrain is known in detail, the whole trajectory can be computed before departure. This is shown in [9] with simultaneous consideration of diverse parameters. Of course dynamic environment changes generate serious problems for this approach.

The technique used by the local navigation planning presented in this article is closely related to active collision avoidance techniques based on the Dynamic Window Approach (DWA) [10], [11]. The DWA chooses velocity controls for the robot by optimizing an objective function, that aims at achieving a goal-directed behavior while avoiding collisions. However, the DWA restricts the search to a single time step, i.e. it selects the next best controls based on the current sensor input and a model of the robot dynamics. It does not take dynamic changes within the environment into account.

The restriction to one step look-ahead also sometimes leads to oscillatory behavior. For this reason Stachniss et al. [12] integrate DWA style collision avoidance with path planning and carry out an A*-search for sequences of velocity commands which drive the robot to the goal. They limit the computational complexity of the planning problem by limiting the effective configuration space to a tubular region around the shortest-path to the target location. The integrated planning and collision avoidance requires a map of the environment and proper localization and is therefore limited to known environments. The EST technique of the local navigation shown here carries out a randomised search for a sequence of good velocity controls solely based on the current sensor information. Following an idea by Phillips et al. [13], the sampling of candidate controls is guided by a heuristic function similar to A*. In contrast to the deterministic approach, the quality of the resulting control sequence depends on the time available for planning. The algorithm is able to generate robot control in an anytime fashion.

The idea of controlling a large group of robots with limited computing power has been in the focus of computer science before. Belta and Kumar [14] present a particularly interesting control technique that forces a group of robots to stay within a predefined shape. The approach consists of a set of control laws which ensure that the robots keep their relative positions during motion. However, the technique requires a global observer that has to monitor all robots and switches control laws whenever necessary.

Over the last months the interest in convoys of robots has increased which is reflected in the robotics literature. Many solutions use markers on the rear of a leading vehicle that can easily be recognised by the perception of a following robot. Although systems using e.g. laser scanners in combination with reflection foil or video cameras together with coloured markers ([15], [16]) achieve appreciable performances, these systems have their weaknesses. In general all these approaches only work, if the leader’s marker is within the sensor’s field of view. As soon as the marker leaves it, the method fails or has to use fallback mechanisms, which do not guarantee a successful recovery. Another flaw emerges from the very short distance between two convoying robots which the approaches want to achieve. Less space means shorter reaction times yielding in short computing intervals and therefore limited local navigation intelligence. If for example a moving obstacle succeeds to get in-between a convoy of two or more robots all following robots are stuck, because they can only stop and wait for the obstacle to disappear.

III. LOCAL NAVIGATION

A. Trajectory Generation using Expansive-spaces Trees

The actual motion control of the overall approach is performed by a local navigation planning component. This planer directly controls the robot’s velocities in order to steer it on a collision-free path towards the destination position provided by the global navigation.
Collision tests are carried out in the configuration space of the robot solely based on the robot’s proximity sensor data. In our case a configuration $x_k = (x, y, \theta, v, \omega)$ of a robot at time $k$ consists of the robot’s position and heading $(x_k, y_k, \theta_k)^T$ as well as its translational velocity $v_k$ and its rotational velocity $\omega_k$.

Destinations $y$ of the local navigation are also expressed as states in this five dimensional configuration space. The robot is controlled by issuing velocity commands $u_k = (v_k, \omega_k)$. We use a motion model which assumes that velocity changes are carried out instantaneously and that the velocities remain constant during fixed short time intervals $\Delta t$. Under these assumptions, the state transitions of the robot can be described by

\[ x_k = f(x_{k-1}, u_k) = \begin{pmatrix} x_{k-1} + \frac{v_k}{\omega_k} \sin \theta_{k-1} + \frac{v_k}{\omega_k} \sin(\theta_{k-1} + \omega_k \Delta t) \\ y_{k-1} + \frac{v_k}{\omega_k} \cos \theta_{k-1} - \frac{v_k}{\omega_k} \cos(\theta_{k-1} + \omega_k \Delta t) \\ \theta_{k-1} + \omega_k \Delta t \\ v_k \\ \omega_k \end{pmatrix}. \]

Based on this model, we use a variant of expansive-spaces trees (EST) to perform the motion control. EST algorithms belong to the probabilistic roadmap techniques for path planning. For a given target state, they generate a roadmap by growing a random tree $T = (V, E)$ consisting of nodes $V$ and possible transitions $E$ between nodes. A node $v \in V$ consists of a state in configuration space and additionally some management information e.g. the time required to reach the node.

Our EST algorithm actually performs a randomised search for good sequences of velocity commands, which steer the robot towards the target. This is achieved by successively

1) selecting a node $v$ with state $q$,
2) generating a new node $v'$ with state $r$ by choosing a pair of velocities $(v', \omega')$ reachable from $q$ within one time interval and computing the state $r = f(q, (v', \omega'))$,
3) And finally adding the node $v'$ to $V$ and the new transition $(v \rightarrow v')$ to $E$, if both are admissible, i.e. if the trajectory from $q$ to $r$ is collision-free.

In order to achieve an efficient target-directed expansion of the tree, the node selection is controlled by an importance function, which takes the distance to the target state into account. Each node is selected with a probability proportional to

\[ A_c(v) = \exp(-\text{outdegree}(v) - (g(v) + h(q, y))^2). \]

Here, $g(v)$ is the time required to reach the node $v$ starting from the root node of the tree. This time is proportional to the depth of $v$ in the tree. $h(q, y)$ is a heuristic function, which provides an optimistic estimate of the time required to reach the target state $y$ from $q$. It is defined as

\[ h(q, y) = \frac{d(q, y)}{v_{\text{max}}} + \frac{|\alpha(q, y)|}{\omega_{\text{max}}}, \]

where $d(q, y)$ represents the line-of-sight distance to the target, $v_{\text{max}}$ denotes the robot’s maximum translational speed, and $\omega_{\text{max}}$ is the maximum rotational velocity. The angle $\alpha(q, y)$ describes the difference between the robot’s heading in state $q$ and the line-of-sight between $q$ and $y$. Finally, $\text{outdegree}(q)$ penalises nodes which have already been selected often and the exponential function is used to convert from the expected costs to a smooth density over states, which prefers nodes closer to the target. Figure 1 shows an example EST built by our implementation of the technique.

![Fig. 1. An expansive-spaces tree generated by our system. The tree is grown from the robot’s initial state R towards the destination state D. Obstacles detected by the laser range finder are shown in black. Note, that for given velocities the robot moves on circular trajectories, as a consequence, the tree obtains a bush-like shape. The optimal path extracted from the tree is highlighted in the lower right.](image-url)
target state exactly. For this reason it is proposed in the planning literature to use a bidirectional search [17] by growing a second EST starting from the target state towards the robot’s state and check, if the two trees can be connected. While we adopt the idea of bidirectional search, we think that the connectivity-tests involved in using a second EST are too time consuming. Instead, we use dynamic programming in the five dimensional configuration space starting at the target location, in order to find a complete control policy in a compact area around the target. To limit the computational burden, the value iteration involved is limited to a coarse fixed five-dimensional grid. The computed policy enables us to complete the control sequence generated by the EST planner, as soon as a node of the EST reaches a grid cell.

The presented local navigation is even capable of predicting and processing dynamic obstacles. These features and more details about this module are described in [4].

B. Fast Trajectory Generation

The use of randomised planning strategies for local navigation has one main disadvantage: It cannot be guaranteed, that a good trajectory to a destination is found in the short time available.

To compensate this weakness, we added a simple planning mode which maximises the translation velocity and consumes nearly no computing time. It is applied, if only the x- and y-coordinates of the target position are used, and if no obstacles interfere. Basically the robot’s field of view is partitioned into three zones, which are shown in Figure 2: If a target coordinate is located in the area with a defined apex angle in front of the robot, a node with maximum translation acceleration and zero rotation is added and automatically selected as next expandable. The second area is a narrow sector on the left and right side of the first. If a destination happens to be in this zone, a small trajectory of two nodes with maximum translation velocity is added to the existing path, one with rotation acceleration, the other with the compensating deceleration in the appropriate direction. Targets in the third zone covering the remaining area cannot be handled by this fast planning mode. Here the normal EST planning is used for turning the robot until the target is in zone two or one. Some EST components have been enhanced for this purpose. When the given destination is reached using the fast planning mode, the constructed path is modified in such a way that the robot comes to a stop at its end.

IV. GPS-Navigation

The previous section described, how the actual motion control of the individual robots of a convoy is performed in our system. We will now explain how the robots are coordinated to actually build a convoy that effectively stays en-route. To achieve this task, two sub-problems have to be solved. First, the individual robots have to be able to reliably estimate their global position in order to stay on track and to communicate meaning-full waypoints between each other. Second, the motions of the robots need to be coordinated to achieve a convoy formation, i.e. the robots should continuously keep an appropriate distance form each other. In the following, we will address both topics separately, starting with the pose estimation.

A. Position Estimation using Kalman Filtering

To increase the precision of our planning, we do not use the sensor’s output directly. We have two major localization sources, low cost GPS and the robot’s odometry, both with their drawbacks: Lower class global positioning systems tend to generate position stepping in certain situations and robot odometries often skew. To let the sensors compensate each others flaws Kalman Filters are commonly used (e.g. [18]). In our approach we use a enhanced Kalman Filter to combine the GPS, odometry and compass data.

Our filter uses the relative odometry values as control inputs and the global GPS and compass data as measurements. Translations and rotations are tracked independently, as GPS and compass data come from separate units, and the rotation filter has to take the periodicity of angular values into account. The covariance matrix for GPS input is derived from the GPS expected position error, all other covariances have been evaluated experimentally.

B. Global Waypoint Navigation

In order to achieve the convoying behavior, the robots communicate waypoints between each other. Waypoints are transmitted in the global reference frame provided by the position estimation component. To force the robots
to drive in-line, each robot communicates the waypoint it just reached to the global navigation component of the up following robot. Of course, this approach only leads to a robust convoy navigation, if the number of waypoints en-route is sufficiently dense. To enable the technique to work even with small numbers of waypoints and large groups of robots, intermediate targets are added between the given waypoints using linear interpolation. The distance between these artificial waypoints roughly describes the minimum clearance between two consecutive robots of the convoy and can be chosen appropriately.

A waypoint is considered as reached, if the reported estimated global position of the robot is within a certain distance of the waypoint. The threshold distance is part of the waypoint’s description, so different waypoints can be approached with different precisions, a technique which has also been used in [5]. If a waypoint is reached, it is not passed to the next robot immediately. The propagation is delayed, because otherwise the leading robot would partly block the following robot’s direct trajectory to its next destination most of the time, rendering its local planning unnecessary complicated. A waypoint is communicated, as soon as the robot has passed it, i.e. when the distance to the last waypoint increases again, which is an indication that the robot is really moving away from it. Thus, waypoints are not propagated when they are reached, but when they are left again. The precision of the distance to which a given waypoint should be approached can be specified by the user; for the artificially added waypoints this precision is defined by the leading robot. If all robots have sufficient sensors for obstacle detection, the distance should be large, leaving the robots space for avoidance maneuvers. Notice that the local navigation always tries to approach the exact position of all waypoints, only the point in time, when the local navigation switches to the up following waypoint can be influenced with this parameter. In other words, the system still tries to move on the direct connection between two main waypoints, if this is possible. An example run is shown in figure 4. In the case that the following robot does not have sensors for obstacle avoidance, and the GPS is sufficiently precise, the robots can be forced to follow the given route exactly by setting the minimum approaching distance for the intermediate waypoints to a very small margin. Together with a large number of artificially added intermediate waypoints, this forces all the robots to pass through a narrow corridor. Figure 5 shows an example run, where the parameters were chosen this way.

It is also the task of the global navigation to translate between the global waypoints, which are communicated between robots and the local robot-centric destination points, which are passed on to the local navigation component. The robots keeps track of its own position in the robot-centric coordinate frame using its odometry. This coordinate frame can differ considerably from the global GPS-based coordinate frame and more importantly, the mapping between these two coordinate systems changes over time due to odometry errors. The global navigation keeps track of these changes and performs the necessary corrections. In detail, the global navigation carries out the following steps: Given the next waypoint for a robot and the robot’s position in global coordinates, the global navigation translates the position of the waypoint into robot-centric coordinates and sends it to the local navigation as its next destination point. The global navigation maintains the mapping between the global and the robot-centric reference frame over time for this purpose. The mapping is continuously updated based on the global position estimates, the global navigation receives from the position estimation component, and the robot’s position provided by the odometry. Notice that the coordinates of the relative destination points need to be updated as well, if the mapping between the global and the local reference frame changes. Destination points are re-transmitted, if the error between the new and the old values exceed a certain threshold. Figure 3 illustrates these relations. In our current implementation we use the angular difference between the old and the current destination to decide if generating a new command is necessary.

V. EXPERIMENTAL

We implemented the approach using the robot middleware RoSe [19] and tested it with three simulated and two real RWI all-terrain robots (ATRV). The simulated robots had 360 degree laser range finders, while the
A. Comparison of Convoying Approaches

The first experiment we want to present is a comparison of different approaches which could be used for convoying. In every example, the collision avoidance and local navigation is handled by the earlier described EST-planner. The leading robot is always controlled by our global waypoint navigation, while the method for the two following robots is altered. The precision of every given waypoint is set to 0.5m, intermediate destinations have a precision of 2m.

To begin figure 6 shows the nine main waypoints and the routes the robots moved using the described pass-on system. Notice that all waypoints are almost exactly reached. At the corners the navigation tends to overshoot, but this is necessary to allow a quick and exact waypoint passing and does only happen, if the needed space for such maneuvers is available. This variant of the experiment was repeated 20 times and a histogram of the averaged distances from the leader to the first following robot was created (figure 7). The mean distance is at about 4m. The numerous peaks are results of the straights, where both robots reach their maximum speed and maintain a constant distance. The leader’s average translation velocity over all 20 runs was 0.479m/s; the two following robots reached 0.475m/s and 0.467m/s. The averaged distance between the leader and the first follower was 4.853m. The third robot moved 4.912m behind the second.

Originally one of the key features of the local navigation described in section III was person accompanying. Therefore the person’s trajectory was predicted using a potential field which processed all detected obstacles. For this procedure it was helpful, that the robot tried to follow the person at his/her side, so its sensors had a free sight to the upcoming obstacles which increased the performance of the potential field prediction. This was changed for this experiment to a position right behind the object to follow. Further details about the accompanying functionality can be obtained from [4]. To allow the local navigation to follow another robot, it needs the position of its leader relative to its own, and this information can easily be computed using the GPS- and compass-data. Altogether, this system can be thought of as an intelligent following that directly approaches the leading robot. As expected figure 8 shows that the followers do not move in respect of the waypoints. Since the local navigation always chooses the fastest way to get closer to a leader, corners can be
Fig. 8. The outcome of a simulation run using the local navigation’s person accompanying algorithm. cut, overshot or both. As result, only very few waypoints are passed in an adequate way. On the other side, this approach reached shorter distances between the robots (leader to follower 1: 2.664m, follower 1 to follower 2: 2.418m). The averaged velocities are comparable to the previously achieved (leader: 0.479m/s, 1st follower: 0.467m/s, 2nd follower: 0.461m/s).

Another idea that can be used for convoys is the communication of passed coordinates. Here the leader generates waypoints from the coordinates it has passed in a specified spacing and sends them to its following robot. While the first follower tries to reach these generated waypoints, it creates waypoints itself and sends them to the second follower. Again, all original waypoints are not known to the followers, yielding to low precision, which is displayed in the result figure 9.

A second negative effect can be seen at waypoint 3: Even at high precision demands the first follower is allowed to cut the corner. The trajectory, including the cut, is used to generate waypoints for the second follower, which is again allowed to take a less sharp turn. Thus it is possible that every additional following robot moves a little further away from the original waypoint. Since this method uses the waypoint navigation on every robot, the average velocities and distances between the robots are similar to the first experiment’s results (leader: 0.488m/s, follower 1: 0.465m/s, follower 2: 0.449m/s, average distance between leader and follower 1: 4.893m, follower 1 to follower 2: 4.642m).

B. Waypoint Navigation with Real Robots

To illustrate the capabilities on real robots we set up a waypoint track of about 250m for our ATRVs. They completed the course in about 9min 15s with average velocities of 0.501m/s and 0.521m/s. The mean distance between the robots was 4.800m. The recorded GPS-Tracks can be seen in figure 10. The precisions were set to 1m for given and 2m for artificial waypoints. Notice that the path is surprisingly curvy. This is a result of false positives the 2D laser range sensors detected when the robot tilted forward aiming the scanner into the ground.

C. Convoying with Unexpected Obstacles

The primary feature of the system proposed in this paper is the use of an intelligent local navigation to handle unexpected obstacles. We tested this by setting up a straight track of about 75m with the same precision parameters as before. When the robots were already moving, two persons tried to obstruct the robots’ path: After the robots evaded the first person, their trajectories split up within the limits of the required precision. The second person first interferes the following robots trajectory. When it has circumnavigated him, he quickly moved in front of the leader, who also planned a
path around the just appeared obstacle. The resulting trajectories are shown in figure 11.

VI. SUMMARY AND CONCLUSION

In this paper we presented a technique for combining waypoint navigation with a dynamic collision avoidance without the need for expensive sensor hardware. The local navigation component uses an expansive-spaces tree approach to handle unforeseen obstacles while our global navigation module handles odometry inaccuracies and controls the convoying behavior by passing-on waypoints that have been reached and are being left. Additional targets are linearly added as intermediate waypoints if too few or too distant waypoints are given. Furthermore an approaching distance for every waypoint can be set influencing our system’s degree of autonomy. With appropriate values convoys of robots with collision detection sensors can circumnavigate unexpected obstacles while sensorless vehicles have to trust their GPS-receivers and follow the given route strictly. Our results demonstrate that safe waypoint navigation and convoying can successfully be combined with active collision avoidance.

Future research will focus on further improving the quality of the convoying. The leading robot’s global navigation should check the status of the whole convoy, eventually reduce velocities to enable a catch up of falling back robots, or reassign positions in the convoy, in case robots are immobilised. Furthermore the position estimation could be enhanced by including the positioning sensor data from a robot’s leader and follower. The measurements of laser range finders or cameras could be used to approximate the distance between two robots, if needed. By supplying the Kalman filter with these additional information it should be able to increase the precision of its estimation.

REFERENCES

Cooperative Target Tracking of Mobile Robots

Zongyao Wang and Dongbing Gu

Abstract—The tracking system of multiple robots introduced in this paper contains two components. One is the estimation of target position where each robot uses its on-board pan/tilt camera to detect a moving target. Another is the tracking control of mobile robots where each robot controls its motors to move toward the detected target. The challenge is the distributed implementation of the cooperative estimation of target position and the coordinated control of robot tracking. A distributed Kalman filter is developed for the cooperative target estimation. The key to the cooperation is the use of a consensus algorithm which reaches an agreement about the target position via local communication. A distributed flocking algorithm is used for the coordinated tracking control. The robots use this flocking algorithm to track the target position and avoid collision. In the experiments, a group of wifibots are used to test the proposed algorithm. The experiment results show that the cooperative target estimation algorithm provides a satisfactory estimation of the target and the flocking group can smoothly track the target.

I. INTRODUCTION

Multi-robot systems can often deal more effectively with tasks that are difficult for a single sophisticated robot to accomplish due to the attractive structure characteristics, such as simplicity, flexibility, and redundancy. Being inspired by the swarming behavior of living beings (flocks of birds, schools of fish, herds of wildebeest, and colonies of bacteria), multiple robots cooperatively track a target is one of the salient features of multi-robot systems.

A biological behavior introduced in [1] shows that an animal flock can detect and follow a target with limited sensor information. In the flocking group, only a few individuals (leaders) can “see” the target. The others (followers) do not know which individual is the leader and do not know where the target is. However, they can flock as a group to a destination. In this paper, we will introduce a similar robot flocking system. The robots use local vision sensors to detect the target position. The neighbor robots also interact with each other to share target information and all members in the group gradually reach an agreement about the target position. All robots use the target position as a tracking destination to control their motions to flock cohesively.

The robot flock can be considered as a movable sensor network. The problems of distributed estimation and data fusion of sensor networks have been investigated in the past few years. The paper [2] focused on the network optimization. The system used a tree type configuration in which a root is needed to fuse all the data. The paper [3] explored the trade-off between energy and application quality of sensor networks. The paper [4] analyzed the energy consumption of p2p sensor networks. The sensor nodes are fixed in these systems and several cluster heads are needed for data fusion.

Kalman filter is commonly used in the target estimation. To apply Kalman filter in sensor networks, a distributed Kalman filter (DKF) was introduced in [5][6]. To implement the distributed computation in the DKF, a consensus algorithm is used, which requires local communication between nodes and reaches an agreement regarding a certain quantity of interest that depends on the state of all robots [7].

On the other hand, the consensus algorithms are also used in flocking control. Vicsek introduced a flocking system with the consensus control in the paper [8]. A theoretical explanation of the Vicsek system was given in [9]. Authors analyzed the flocking system with assistance of graph theory and Wolfovitz’s stochastic matrix theory [10]. The paper [11] applied local information to implement the consensus algorithm of the Vicsek system. Depending on local information, robots gradually update their headings to move in the same direction.

Using a group of robots to track a target requires that all robots estimate the target position and flock as a group simultaneously. This paper proposes a distributed algorithm for such a purpose. Firstly, the robots estimate the target position cooperatively via vision sensors. Each robot uses its vision sensor to measure the relative distance and relative angle to the target. Each robot uses Kalman filter to estimate the target position, and a consensus algorithm is used to reach an agreement about the target estimation. Secondly, the robots need to avoid collision with each other and keep the flocking cohesiveness by tracking the target. The estimated target position is used in the tracking control.

The outline of this paper is arranged as: Section 2 introduces the preliminary knowledge of our flocking algorithm and Kalman filter. Section 3 develops a consensus algorithm to estimate the target position. Section 4 presents the flocking algorithm. The experimental platform is introduced in section 5. Section 6 describes the experiment results. A brief conclusion is given in section 7.

II. PRELIMINARY

A. Flocking model

We consider a flocking system consisted of robots moving in a 2D coordinate space. The state of these robots can be represented by a set of position vectors \( q = [q_1^T, q_2^T, \cdots, q_N^T]^T \in \mathbb{R}^{2N} \) where \( q_i = [x_i, y_i]^T \) is the 2D position. Robot velocities can be described as \( p = \dot{q} = [\dot{q}_1^T, \dot{q}_2^T, \cdots, \dot{q}_N^T]^T \), where \( \dot{q} \) is the differential of \( q \) with respect to time.
to time. In our flocking system, \( p \) is considered as the system output and \( q \) will be used as the system input.

The topology of a flocking network can be represented by a graph \( G = (V, \xi) \) where \( V \) is the set of vertices (robots) and \( \xi \) is the set of edges (communication channels). The communication range of robots is denoted as \( C \). The network topology depends on the distances between robots; that is the links only exist between robot pairs whose relative distance is smaller than \( C \). If there is a link between robot \( i \) and robot \( j \), we call them neighbours. The position between \( i \) and \( j \) should satisfies the condition:

\[
||q_i - q_j|| \leq C
\]

(1)

The set of neighbors of robot \( i \) can be written as:

\[
N_i = \{ j \in V, j \neq i : ||q_i - q_j|| \leq C \}
\]

(2)

We assume that the communication range is the same for all the agents, so the graph of flocking system is undirected.

### B. Kalman filter

The target position is defined as \( q_i \). The target observation of robot \( i \) is represented by \( z_{ir} \). The observations of all robots can be written in the vector form:

\[
z_{ir} = [x_{ir}^T, z_{ir}^T]^T
\]

The estimation of target position of robot \( i \) is \( \hat{x}_{ir} \in \mathbb{R}^4 \), which includes the estimations of target position and target velocity:

\[
\hat{x}_{ir} = [\hat{x}_{ir}^T, \hat{\dot{x}}_{ir}^T]^T
\]

(3)

where \( \hat{x}_{ir} \) is the estimation of target position. The estimation vector of all robots is

\[
x_{ir}(k+1) = Ax_{ir}(k) + w(k); \quad p(w) \sim N(0, R)
\]

(4)

where \( R \) is the covariance of state error and

\[
A = \begin{bmatrix}
1 & 0 & t & 0 \\
0 & 1 & 0 & t \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(5)

where \( t \) is the sample time of Kalman filter. The sensor model of robot \( i \) is defined as:

\[
z_{ir}(k) = Hx_{ir}(k) + v(k); \quad p(v) \sim N(0, Q)
\]

(6)

where \( Q \) is the covariance of observation error and

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]

(7)

Here we assume the sensor models used by different robots are the same and the noise property is the same.

The estimation model of robot \( i \) can be written as:

\[
\hat{x}_{ir}(k+1) = Ax_{ir}(k) + K_i(k+1)[z_{ir}(k) - H \hat{x}_{ir}(k)]
\]

(8)

where \( K_i(k) \) is the gain at step \( k \) and

\[
K_i(k+1) = P_i(k+1)H^T[HP_i(k+1)H^T+R]^{-1}
\]

(9)

where \( P_i(k) \) is the covariance of estimation error at step \( k \).

\( P_i \) is updated with equation:

\[
P_i(k+1) = AP_i(k)A^T + Q
\]

(10)

### III. CONSENSUS DISTRIBUTED KALMAN FILTER

In the flocking network, robots can reach an agreement on their aggregate information via consensus between adjacent robots. In this section, we will introduce how to embed a consensus algorithm into Kalman filter and create a consensus DKF. The purpose is to reach an agreement about the estimation of target position among the robots.

Consider the alignment consensus algorithm which is introduced in [9]:

\[
\theta(k+1) = (I - \mu L)\theta(k)
\]

(11)

where \( L \) is the Laplacian matrix of the network graph. The \( I \) is a \( N \times N \) identity matrix where \( N \) is number of robots in the flocking system. \( \theta \) is the vector of robot headings. \( \mu \) is a positive constant. The properties of Laplacian matrix have been discussed in [12]. Because the Laplacian matrix is a semi-positive definite matrix if the graph is connected, the system (11) is a stable system. Moreover, the system state \( \theta \) would asymptotically reach equilibrium: \( \theta_1 = \theta_2 = \cdots = \theta_N = \frac{1}{N} \sum_{t=1}^{N} \theta_i \). So the alignment consensus algorithm drives all the robot headings point to the same direction. The common direction at the final state is the average of all the robots’ headings at the initial state. If we replace the robot headings (\( \theta \)) in (11) with the estimation of target position of each robot, we can get a consensus algorithm:

\[
\hat{x}_{ir}(k+1) = (I - \mu \hat{L})\hat{x}_{ir}(k)
\]

(12)

where \( \hat{L} = L \otimes I_4 \) is an 4D Laplacian matrix of the flocking network and \( \otimes \) is the Kronecker product. Similarly, \( \hat{I} = I \otimes I_4 \).

The outcome of the consensus algorithm is \( \hat{x}_{ir} = \hat{x}_{jir} = \cdots = \hat{x}_{nr} = \frac{1}{N} \sum_{t=1}^{N} \hat{x}_{ir} \) which is the mean of the estimations of target position.

According to the definition of Laplacian matrix, the algorithm (12) can be written in a local form:

\[
\hat{x}_{ir}(k+1) = \hat{x}_{ir}(k) - \mu \sum_{j \in N_i} (\hat{x}_{ir}(k) - \hat{x}_{jr}(k))
\]

(13)

where \( N_i \) is set of neighbors of the robot \( i \). The robot \( i \) sends its estimation to all the neighbors via local communication, and all the neighbors send their estimations to robot \( i \) as well. Each robot uses the consensus algorithm to calculate its own estimation of target position. Asymptotically, all the robots in the flocking system can reach an agreement about the estimation of target position.

Assume that the estimation of target position of robot \( i \) at step \( k \) is \( F_i(k) \), which can be written as follows according to Kalman filter:

\[
F_i(k) = AX_{ir}(k) + K_i(k)[z_{ir}(k) - H \hat{x}_{ir}(k)]
\]

(14)

Using the consensus algorithm, we have:

\[
\hat{x}_{ir}(k+1) = F_i(k) - \mu \sum_{j \in N_i} [F_i(k) - F_j(k)]
\]

(15)

In general, the robots “negotiate” with the neighbors about the estimation of target position at the current step. In the next step, the result of the consensus algorithm will be used as new estimation. In the next section, we will talk about how to use the estimation to achieve the tracking control.
IV. FLOCKING ALGORITHM

There is no specific leader or follower in our flocking system. Any robot who can detect the target is considered as leader. The leaders would become followers if they lose the view of target and vice versa. Although the followers do not have current observation, they can estimate the target position via consensus DKF. As a result, each robot in the group has an estimated target position which can be used for flocking purpose.

The flocking algorithm consists of two components. One is the avoiding collision. Simply, adjacent robots should keep a specific distance. If the distance between two robots is too small, they attempt to separate. There are several approaches to design a separation potential function. In this paper, we use the potential function introduced in [13] to achieve the separation. We use \( H_i(d_{ij}) \) to denote the collision avoiding potential function between robot \( i \) and \( j \) (\( d_{ij} \) is the distance between them). Another is the tracking control that each robot makes motion towards the estimated target position. We can use \( H_i(q_i, \hat{q}_{ir}) = \frac{1}{2}||q_i - \hat{q}_{ir}||^2 \) to denote the tracking control potential function. The controller of robot \( i \) should be is designed as:

\[
\dot{q}_i = -\sum_{j \in N_i} \frac{\partial H_i(d_{ij})}{\partial d_{ij}} - k_r(q_i - \hat{q}_{ir}) \tag{16}
\]

where \( k_r \) is a positive gain and \( \hat{q}_{ir} \) is the estimation of target position. The robot model we used is a kinematic model:

\[
\begin{align*}
\dot{x}_i = v_i \cos \theta_i \\
\dot{y}_i = v_i \sin \theta_i \\
\dot{\theta}_i = \omega_i
\end{align*}
\tag{17}
\]

where \( \theta_i \) is the robot heading and \( q_i = [x_i, y_i]^T \) is the hand position which is a point located at the heading axis with distance \( L \) to the center of robot. A coordination transformation can be made to obtain \( v_i, \omega_i \) from \( q_i \):

\[
\begin{align*}
v_i &= \dot{x}_i \cos \theta_i + \dot{y}_i \sin \theta_i \\
\omega_i &= \frac{1}{L} \left[ \dot{x}_i \sin \theta_i - \dot{y}_i \cos \theta_i \right] 
\end{align*}
\tag{18}
\]

V. EXPERIMENTAL PLATFORM

In the experiments, the robots get their own positions \( q_i \) from a VICON system. At the same time, they use the on-board pan/tilt cameras to get relative target distances and angles and convert them into the target positions by using their own coordinates.

A. Algorithm architecture

Three wifibots are used for the experiments. Actually the algorithm proposed in this paper is not limited to three robots. It can be applied to \( N \) robots. The algorithm architecture is shown in figure 1. Each robot uses the on-board camera to detect the target. The observation image is processed locally and an observation \( z_{ir} \) is produced each time. The observation \( z_{ir} \) is sent to the consensus DKF.

The consensus DKF algorithm needs further two inputs: \( q_i \) from the VICON system and the intermediate estimate \( F_j \) from neighbor \( j \) via wireless communication. It produces two outputs: the intermediate estimate \( \hat{q}_i \) and the estimation of target position \( \hat{x}_{ir} \). \( F_j \) is sent to its neighbors via wireless communication and \( \hat{x}_{ir} \) is sent to the flocking controller. Due to the limited experiment space used, all the robots can receive signals from all the other robots. To simulate the limited range, a robot only receives signals from neighbors who are located within the distance of \( C \) and discards signals from other robots whose distances are larger than \( C \).

The flocking controller takes the estimation of target position \( \hat{x}_{ir} \) as its input and outputs two velocities: \( v_i \) (forward speed) and \( \omega_i \) (rotation speed).

B. Vision processing and camera motion control

The target is detected by using the on-board pan/tilt camera. To simplify the vision processing, a robot equipped with a red cylinder is used as the target (see figure 2(a)). The on-board camera only detect the red cylinder by using the color threshold techniques. The image obtained by the on-board camera is in RGB space. Because the image in HSV space is more sensitive to colors, the RGB image is converted to the HSV image. It is known that different colors have different hue degrees in HSV images. A band pass filter is used to filter the red color in the HSV image. For the purpose of de-noise, an erosion algorithm is applied to erase high frequency component in the image. Finally, we get a image with target only (figure 2(b)). The mean position of all the pixels of target is calculated. The means of row and column are the target relative position in the image.
To measure the distance to the target, the width of the target image is also calculated during the process. To find the relationship between distance and width of target image, we recorded the distances from robot to target and also recorded the corresponding width of target image. The Table I shows the data we recorded:

<table>
<thead>
<tr>
<th>distance to target (mm)</th>
<th>target image width (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>170</td>
</tr>
<tr>
<td>400</td>
<td>140</td>
</tr>
<tr>
<td>600</td>
<td>88</td>
</tr>
<tr>
<td>800</td>
<td>60</td>
</tr>
<tr>
<td>1000</td>
<td>44</td>
</tr>
<tr>
<td>1200</td>
<td>32</td>
</tr>
<tr>
<td>1400</td>
<td>25</td>
</tr>
<tr>
<td>1600</td>
<td>18</td>
</tr>
<tr>
<td>1800</td>
<td>14</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
</tr>
<tr>
<td>2200</td>
<td>6</td>
</tr>
<tr>
<td>2400</td>
<td>5</td>
</tr>
<tr>
<td>2600</td>
<td>2</td>
</tr>
<tr>
<td>2800</td>
<td>0</td>
</tr>
</tbody>
</table>

The distance to target should be larger than 0.3m, otherwise the wifibot will crash to the robot who carries the target. The furthest distance to target is 2.8m. Larger than 2.8m, the robot will lose the view of target. We plot these data with star markers in the figure 3. The X-axis is the target image width. The Y-axis is distance to the target. It is found that relationship between the distance and the image width can be fitted to a logarithm function which is the curve plotted in figure 3:

\[
\log_{10}(d) = -590 \log_{10}(\alpha/195) + 300
\]

where \(\alpha\) is the width of target image.

The on-board camera equipped on the wifibot can pan from -170 to 170 degree to increase the view field. A feedback control loop is used to control the camera pan angle. The control objective is to keep the target in the middle of the image. The pan degree \(\theta_r\) is used as the relative angle of the target. The camera observation is \(z_{ir} = [x_{ir}, y_{ir}]^T\) and

\[
\begin{align*}
\dot{x}_r &= x_i + d_i \cos(\theta_i + \theta_r) \\
\dot{y}_r &= y_i + d_i \sin(\theta_i + \theta_r)
\end{align*}
\]

VI. EXPERIMENTAL RESULTS

The experimental tests are conducted in the Robot Arena at University of Essex. We exam the tracking performances of single robot and multiple robots without the cooperation in the estimation. The results are given in the first part of this section. The second part demonstrates a result with the consensus DKF.

A. Performance of vision target estimation

To test the performance of vision target estimation, a wifibot and a target are placed in the arena in the first test. The target robot moves along a trajectory. The wifibot uses its on-board camera and Kalman filter to estimate the target position. At the same time, the wifibot use the estimation result to track the target. Figure 4 shows the target true trajectory (solid line) and estimated target trajectory (dashed line). The cross marks represent the observation \(z_{ir}\) from the on-board camera. It can be seen that the camera observations scatter around the true trajectory while the dashed line is a smooth trajectory and approximates to the true trajectory.

In the experiment, the target estimation and robot tracking control are processed simultaneously. The figure 5 illustrates the target trajectory (solid line) and the robot trajectory (dashed line). It shows that the wifibot makes a good use of the estimation results and continuously follows the target.

In the second test, multiple robots are used. Each of them uses Kalman filter to estimate the target position without the cooperation in the estimation. But they control their motions using the flocking algorithm to keep them in a flock. The problem in the estimation of target position in the flocking system is the disagreement about the estimations of target position among robots. The figure 6 illustrates the estimations of target position of all three robots. It can be seen that each robot can estimate a smooth target trajectory. But the differences between these estimations are enormous. Due to the disagreement, it makes the flocking control more difficult. In the testing, the robot group easily
split (which is also called fragmentation [14]) because robots track “different targets”. Thus a cooperative estimation of target position is necessary.

B. Performance of consensus DKF

In the following experiment, we will use the consensus DKF to reduce the disagreement. Three wifibots use the consensus DKF (15) to estimate the target position. The figure 7 shows the results of the consensus estimation. The solid line denotes the true trajectory of target; the dashed lines are the estimation results of three robots. It can be seen that, the disagreement of estimations has been reduced. All the estimations are asymptotically approximated to a single trajectory which is close to the true trajectory.

To evaluate the performance of the consensus DKF, we plot the estimation radius in figure 8(a). The estimation radius is the radius of the minimal circle which can cover all the estimations of robots. It can be seen that, the estimation radius of the consensus DKF (solid line) is much smaller than that of DKF without the consensus (dashed line). The estimation differences between robots are stable at around 10cm.

The figure 8(b) illustrates the mean errors of estimations to the true trajectory. The error of consensus DKF is much smaller than that of the estimation without consensus (the dashed line).

The cohesion radius of flocking is a parameter to evaluate the cohesiveness of flocking. Similar to the estimation radius, it is the radius of the smallest circle which can cover all the robots in the flocking. The figure 8(c) illustrates the cohesion radius of flocking with the consensus DKF (solid line) and that without the consensus. It is clear that the flocking with consensus DKF is more cohesive and the flocking radius keeps stable during the tracking.

The figure 9 shows the trajectory of flocking robots. The solid line represents the trajectory of the target. The other three dashed lines are the trajectories of the three wifibots. It shows that the flocking smoothly tracks the target and flocking keeps cohesive during the process. It also proves that the consensus DKF can provide a relative accurate estimation of the target position.

VII. CONCLUSIONS AND FUTURE WORKS

This paper presents a vision tracking algorithm for multi-robot systems. It uses a consensus DKF to establish the cooperation of vision target estimation. Each robot in the flocking can benefit from the observation of other robots and accordingly increases the tracking performance. Meanwhile all robots in the group can use the agreed estimation of target position to control their motions to flock cohesively. The agreement is achieved via a consensus algorithm embedded in Kalman filter. Both the estimation of target position and flocking control are implemented in a distributed way.

Substantial work has been done to implement the algorithms in real robots. A group of wifibots are used to test the performance of flocking algorithm, each of which is equipped with a pan/tilt camera. The experiment results prove that the consensus DKF can provide satisfactory estimation of target position and the flocking algorithm can be applied in real robots.

105
Our further work will focus on the improvement of the cooperative vision tracking algorithm. We will also investigate the influences of communication delay to the consensus DKF and flocking algorithm.

**REFERENCES**


Sliding Mode Control for Agents and Humans

S M Veres and N K Lincoln

Abstract—A new sliding mode controller for attitude and position of an autonomous satellite within a cluster is presented. Asymptotic stability is proven by Lyapunov’s method. The result is unique in that it handles discrete time control and includes potential function guidance in the scheme. This theoretical result has been written down in a second contribution of the paper: the introduction of natural language publications that can be read by humans and agents (machines) as well. As a first example of natural language programming, system English (sEnglish for short) is introduced to the reader as a new programming paradigm, not a natural language interface. The most important feature of sEnglish is that it compiles into MATLAB code unambiguously. sEnglish is suitable to program complex programs, personal assistants and autonomous agents but its main purpose is to be used for passing on knowledge to agents. As such it is a publication for humans as well as for machines.

This paper is a call for participation in a community effort to use natural language programming and sEnglish in our work in academia and companies.

I. INTRODUCTION

The main purpose of this paper is to introduce natural language programming and illustrate it. Instead of describing its philosophy we decided to illustrate its use on a control theoretical result that merits publication in itself.

There is a substantial amount of literature within the field of spacecraft control which deals with the separate issues of position and attitude control; comparatively little work has been carried out on the joint nonlinear problem of spacecraft position and attitude control. [5], [6], [7] provide control solutions for satellite position control, where the task is to follow a leader satellite. [5] uses a continuous time Lyapunov stability based approach in conjunction with disturbance estimates to facilitate tracking of a reference trajectory with unknown spacecraft mass. [6] explores a discrete time constrained control method for stabilization and control of two nano-satellites using time optimal control but, similarly to [5], is only applied to position control. [10] implements the control method within [8] but replaces the angular velocity feedback with a nonlinear filter of the quaternion to remove the need of a dynamic observer and to exploit the passivity of the system. Although implementing a more stable control approach, the system is designed for a continuous time environment. [11] presents a combined sliding mode controller for an individual satellite, controlling both attitude and position in the presence of zero disturbance and without guidance within a continuous time environment.

The advance in this paper relative to [11] is that we use a different, discrete time sliding mode controller and we integrate control with potential function based guidance. Real computer control systems work within a discrete time environment. Sensor data is sampled at discrete time intervals for implementation within a control regime which is only capable of outputting control signals at a finite rate.

Programming languages such as Fortran, Basic, C, C++, Pascal, Python, etc. and .NET languages are universal algorithmic languages that are available on computers with von Neumann architectures. This technical approach to computing has brought tremendous productivity combined with very high reliability for banks, sales networks, company administrations, science, publishing and the Internet. The achievement is not only the speed and amount of operations performed but their almost perfect determinism. Although verification of software is a big topic today, still one can confidently say that the essential feature of digital computing is its speed and reliability: software behaves in a precisely prescribed way. This reliability resulted in dependence and reliance to such a degree that digital computers have become vital in our everyday lives.

As a result todays computers are very different from biological computation and subsequently programming and communication with machines has not been an easy problem. To improve the situation, graphical windows, use of the mouse and user interfaces have been introduced. Language as a means of communication was mainly limited to one word commands to the operating system. Although natural language interfaces have been created the main problem of creating understanding in the machines remained. One may wonder whether creating machine understanding, in the human sense, is technically feasible at all by a few connected von Neumann machines?

As introduced in this paper, natural language programming (NLP) of intelligent machines is a new programming paradigm and it is not natural language interfacing of traditional software. It is a programming paradigm where our natural conceptual creative processes are used to develop code for our existing computers with reliable behaviours. Creating artificial intelligence this way (instead of copying biological ways) is not easy initially but the rewards are more than what we could get by reproducing an animal’s nervous system. We gain more control over nature this way. We have the prospect to totally automate the planning and production of complex technological products.

On the other hand NLP is also suitable for recording engineering and scientific knowledge in a form that is readable by humans and can also be used by intelligent agents made

This work was supported by SysBrain Ltd, www.sysbrain.com.
S. M. Veres and N. K. Lincoln are with the School of Engineering Sciences, University of Southampton, SO17 1BJ, UK s.m.veres@soton.ac.uk
capable of using them. This is among the first initiatives to enable the production and distribution of human/machine sharable publications. Finally NLP can make computer and robot programming accessible to the wider public. In the authors view engineers, students and other professionals should not have to use too low-level programming but should have access to friendly natural language programming in terms of NLP publications, books and toolboxes.

II. CONTROL OF SATELLITE ROBOTS
A. Dynamics
The development of the dynamics and kinematics will be based upon a single satellite, considered to be holonomic with respect to control, meaning that the degree of freedom of the controller exceeds that of the spacecraft dynamics. In addition, the notation will be as follows:

- Capital letter, bold face = Matrix, \( \mathbf{A} \)
- Lower case letter, bold face = Vector, \( \mathbf{a} \)
- Lower case letter, italic = Scalar, \( a \)
- \( m \)=total mass of spacecraft (s/c),
- \( \mathbf{d} \)=position of mass centre of s/c in body frame, \( [d_x, d_y, d_z]^T \)
- \( \mathbf{v} \)=velocity of mass centre of s/c in body frame, \( [v_x, v_y, v_z]^T \)
- \( \mathbf{\omega} \)=angular velocity of s/c in body frame, \( [\omega_x, \omega_y, \omega_z]^T \)
- \( \mathbf{f} \)=total force of thrusters, \( [f_x, f_y, f_z]^T \)
- \( \mathbf{\tau} \)=total torque of thrusters, \( [\tau_x, \tau_y, \tau_z]^T \)
- \( \mathbf{J} \)= inertial matrix of s/c

By using Newton/Euler equations to resolve for forces and moments respectively, we generate the well documented six degree of freedom translational and rotational motion equations for a rigid spacecraft:

\[
\begin{align*}
mv + m\mathbf{\Omega}v &= \mathbf{f} \quad (1) \\
\mathbf{J}\dot{\mathbf{\omega}} + \mathbf{\Omega}\mathbf{J}\mathbf{\omega} &= \mathbf{\tau} \quad (2)
\end{align*}
\]

where the skew symmetric matrix \( \mathbf{\Omega} \) represents the angular velocity cross product matrix as is given below:

\[
\mathbf{\Omega} = \begin{bmatrix}
0 & -\omega_3 & \omega_2 \\
\omega_3 & 0 & -\omega_1 \\
-\omega_2 & \omega_1 & 0
\end{bmatrix} 
\]

The kinematic equation describing the evolution of the spacecraft position and orientation are given by:

\[
\begin{align*}
\dot{\mathbf{d}} &= -\mathbf{\Omega}\mathbf{d} + \mathbf{v} \quad (4) \\
\dot{\mathbf{q}} &= \frac{1}{2}\mathbf{\Omega}\mathbf{q} \quad (5)
\end{align*}
\]

where

\[
\mathbf{\Omega} = \begin{bmatrix}
0 & \omega_2 & -\omega_3 \\
-\omega_2 & 0 & \omega_1 \\
\omega_3 & -\omega_1 & 0
\end{bmatrix}
\]

In this notation \( \mathbf{d} \) represents the satellite position in the body frame and \( \mathbf{q} \) is the quaternion vector.

A complete representation of the dynamics and kinematics for a single rigid body is given by a 13-dimensional state vector \( \mathbf{x} \) as

\[
\dot{\mathbf{x}} = \begin{bmatrix}
\mathbf{d} \\
\mathbf{v} \\
\mathbf{q} \\
\mathbf{\omega}
\end{bmatrix} = \begin{bmatrix}
-\mathbf{\Omega}\mathbf{d} + \mathbf{v} \\
-\mathbf{\Omega}\mathbf{v} \\
\frac{1}{2}\mathbf{\Omega}\mathbf{q} \\
-\mathbf{J}^{-1}\mathbf{\Omega}\mathbf{q}
\end{bmatrix} + \\
\begin{bmatrix}
\mathbf{f} \\
\mathbf{\tau}
\end{bmatrix} + \\
\begin{bmatrix}
0_{(3,3)} \\
m^{-1}I_{(3,3)} \\
0_{(4,3)} \\
J^{-1}\Phi_{(3,3)}
\end{bmatrix} \begin{bmatrix}
0_{(3,3)} \\
0_{(3,3)} \\
0_{(4,3)} \\
0_{(3,3)}
\end{bmatrix} \begin{bmatrix}
\mathbf{f}_d \\
\mathbf{\tau}_d
\end{bmatrix}
\]

where \( \mathbf{\Phi} \) is the cross product of control torques, a consequence of actuator misalignment resulting in parasitic torques generated from translational impulses. In the nominal dynamical model for control the \( \mathbf{\Phi} \) will be taken as \( 0_{(3,3)} \) but robustness of the sliding mode controller against disturbances, and specifically against modelling inaccuracies, will be ensured. \( \mathbf{u} \) is the vector of total applied control \( \mathbf{u} = [\mathbf{u}^T, \mathbf{\tau}^T]^T \) and \( \mathbf{u}_d \) is a vector representing external force and torque disturbances. It will be assumed that the thrusters allow for 6DOF setting \( \mathbf{u} \) within bounds due to low thrust by electrical or cold gas thrusters.

Rows 3-6 and 11-13 of matrix \( \mathbf{B} \) and \( \mathbf{C} \) will be joined together to form two square non-singular sub-matrices \( \mathbf{B}^* \) and \( \mathbf{C}^* \) that will be used later within Section II-E in the formulation of the sliding mode control. In conjunction with \( \mathbf{B}^* \) and \( \mathbf{C}^* \), \( \mathbf{x}^* = [\mathbf{v}, \mathbf{\omega}]^T \) will also be used.

B. Time Discretisation
The generation of a discrete time equivalent model to a continuous system model, commonly referred to sampled model involving time discretisation is normally performed using a zero-order-hold process, i.e. keeping the control signals constant between sampling instants. For linear time-invariant systems discretisation is well documented for models of the form \( \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \) for constant sampling period \( h \) for \( \mathbf{x}(kh + h) = \mathbf{A}\mathbf{x}(kh) + \mathbf{B}\mathbf{u}(kh) \) then \( \mathbf{y}(kh + h) = \mathbf{C}\mathbf{x}(kh) + \mathbf{D}\mathbf{u}(kh) \) as \( \mathbf{x}(kh + h) = e^{\mathbf{A}h}\mathbf{x}(kh) \).

This transformation is perfectly applicable to linear systems, however for nonlinear systems as presented in (7) such a transformation is not possible. The discrete system which we wish to generate is of the form

\[
\mathbf{x}_d(k + 1) = f(\mathbf{x}_d(k), \mathbf{u}_d(k), \mathbf{d}_d(k))
\]

where \( \mathbf{x}_d(k) = \mathbf{x}(kh) \in \mathbb{R}^n \) - state vector, \( \mathbf{u}_d(k) = \mathbf{u}(kh) \in \mathbb{R}^n \) - control input, \( \mathbf{d}_d(k) = \mathbf{d}(kh) \in \mathbb{R}^n \) - nonlinear function.
Time propagation of the above nonlinear system (7) can be approximated by
\[ x(k + 1) = x(k) + h \dot{x}(k) \] (8)
where \( \dot{x}(k) = \frac{d}{dt}x(kh) \) can be evaluated from the continuous time model (7) based on the current system state which has been extracted at the current discrete time interval \( kh \).

Whilst such a method is inferior to the linear discrete model and would not be suitable for accurate long term propagation of the state variables without making the sampling time prohibitively small, it is suitable for achieving a discrete time prediction of the next state for control purposes, subject to the current conditions. Conceptually, viewed from the perspective of the digital system, completing the calculation (8) means that a model predictive approach is being used to determine the anticipated future state one time step in advance.

C. Guidance and Control

Guidance for the desired satellite state will be provided through the use of potential functions in a similar manner to the method in [13] but with the added novelty of including potential functions for attitude guidance. Separate potential functions will be used to prescribe desired velocity vectors for position \( v_d \) and attitude \( \omega_d \) in the inertial system and in the form of \( \Psi v_d \) and \( \Psi \omega_d \), where \( \Psi \) is the inertial to body conversion matrix. Using the actual velocity vectors, \( v \) and \( \omega \) in the body frame, in combination with the desired velocity vectors, a feedback controller will be based upon the velocity vector errors:
\[
\begin{align*}
\varepsilon_v &= v - \Psi v_d \\
\varepsilon_\omega &= \omega - \Psi \omega_d
\end{align*}
\]
which will be used within the sliding mode control regime presented in Section II-E. Prior to this, within Section II-D, the potential functions used to achieve the desired guidance for position and attitude will be developed.

D. Guidance Law- Potential Functions

The potential function used as guidance for position will be constructed as a function of the agent distance from a desired point in space, referred to as a sink location. Compound potentials can be formed by introducing \( n \) attractors towards the \( n \) sinks, \( e_j \). A general expression of the negative gradient of a potential gathering behavior of satellite \( i \) to sink \( e_j \) is
\[
\lambda_i^{\text{gather}} = -\| (e_j - x_i) \| \cdot (e_j - x_i)
\]
For the instance of moving to a fixed point in space, only a single sink location is required. If collision avoidance a waited sum of \( \lambda_i^{\text{gather}} \) can be used for several \( j \) sinks.

The potential function used as guidance for rotational position will be constructed as a function of the agent orientation error from a desired orientation in space, using the quaternion notation. The error quaternion, \( q_e = [\sin(\alpha_e/2)q_0^e,\cos(\alpha_e/2)]^T \), is given below as
\[
q_e = Q_d q
\] (9)
where \( Q_d \) is the matrix multiplication of the desired quaternion vector obtained by using the components of \( q_d = [\sin(\alpha_d/2)q_0^d,\cos(\alpha_d/2)]^T = [q_{d1},q_{d2},q_{d3},q_{d4}]^T \):
\[
Q_d = \begin{bmatrix}
q_{d4} & q_{d3} & -q_{d2} & q_{d1} \\
-q_{d3} & q_{d4} & q_{d2} & q_{d1} \\
q_{d2} & -q_{d1} & q_{d4} & q_{d3} \\
-q_{d1} & -q_{d2} & -q_{d4} & q_{d3}
\end{bmatrix}
\]

Nick, this \( Q_d \) definition does not seem to be correct...?

There exists a one-to-one equivalence between the direction cosine matrix elements and the elements of the quaternion vector and so a suitable output potential function can be based on a desired orientation can be represented in the following form. Using the attitude reference vector \( q_i^d \) as a sink an attitude reference guidance vector for \( \omega_d \) of spacecraft \( i \) is defined as
\[
\lambda_i^{\text{orient}} = \dot{q}_{\text{ref}} \cdot \begin{bmatrix} q_{e1} \\ q_{e2} \\ q_{e3} \end{bmatrix}
\]
where \( [q_{e1},q_{e2},q_{e3}]^T = Q_d^T q_e \). Similarly to the positional potential function development, the potential function for required attitude could consist of multiple weighted components, indexed by \( j \), to result in a more complex behavior for “collision avoidance” in terms of electric thruster exhausts hitting another spacecraft. In the presented scenario, only the acquisition of a specified attitude is considered.

The overall position and angular velocity reference for the satellite can then be defined as the sum of all central contributions as:
\[
\Lambda = [v_d,\omega_d] = \sum_j w_j \begin{bmatrix} \lambda_i^{\text{gather}} \\ \lambda_i^{\text{orient}} \end{bmatrix}
\] (10)

E. Discrete Time Sliding Mode Controller Development

The initial consideration when developing a sliding mode control system is the choice of an appropriate sliding surface, \( \sigma \), such that when \( \sigma = 0 \) is reached, the desired system motion is exhibited. Traditionally, the development of a sliding mode controller proceeds using the theory of variable structure systems applied to continuous time systems which are described by ordinary differential equations [14],[15],[3],[2].

Firstly an equivalent control term, \( u_{\text{eq}} \), is determined such that when on the sliding surface, motion is constrained to the surface \( \sigma_i = 0 \) for all time having intercepted the manifold. This equivalent control term must be augmented with a switching term, \( u_{\text{sw}} \), so as to guarantee global convergence to the sliding surface. Methods for determining a suitable \( u_{\text{sw}} \) are given within [15]. This discontinuous switching term, developed within a continuous-time model, would be switched at a theoretical infinite frequency. Within a
micro-controller based implementation, an infinite switching frequency is impossible to achieve since it is limited to the sampling rate. Subsequently implementing a continuous time model in discrete time will lead to discretisation chatter, a separate issue to chatter resulting from unmodelled system dynamics as reported in [3]. The question that arises is how to develop a control system which can be implemented through a discrete time micro-controller which is controlling an object existing in continuous time.

A combined position and attitude controller implies the necessity of two sliding surfaces relating to the separate issues of translational and rotational movement. Using a discrete time notation, appropriate sliding surfaces for satellite control in six degrees of freedom are given as

\[
\begin{align*}
\sigma_1(k) &= k_v \cdot v_e(k) \\
\sigma_2(k) &= k_a \cdot \omega_e(k)
\end{align*}
\]

With \(k_v\) and \(k_a\) representing fixed scalar gains for velocity and attitude respectively, which will be used to tune magnitude of the error signals. The above sliding surfaces can be placed into matrix form:

\[
\begin{align*}
\begin{bmatrix}
\Sigma(k) = [\sigma_1(k), \sigma_2(k)]^T \\
k_v \Theta_{1(3,3)} \Theta_{3(3)} \\
k_a \Theta_{2(3,3)} \Theta_{3(3)}
\end{bmatrix} \\
= \begin{bmatrix}
v_e(k) \\
\omega_e(k)
\end{bmatrix} \\
= A \cdot x_e(k) \\
= A \cdot [x(k) - \Psi x_d(k)]
\end{align*}
\]

Numerous reaching conditions have been presented for quantifying what represents system motion for ‘sliding modes in discrete time’ [3][16][17]. The most simplistic of these requirements is the need to achieve \(|\sigma(k+1)| = 0\), but due to control bounds this requirement can in some cases be extended to \(|\sigma(k+1)| < |\sigma(k)|\). Extending this requirement to (11), achieving sliding motion in discrete-time for the presented six degree of freedom satellite system entails the satisfaction of

\[
\Sigma(k+1) = A \cdot [x(k+1) - \Psi x_d(k+1)] = 0
\]

or

\[
||\Sigma(k+1)|| < ||\Sigma(k)||
\]

For the present scenario, \(x_d\) is a constant hence \(x_d(k+n) = x_d(k), \forall n \in \mathbb{R}\), and the satisfaction of (12) reduces to attaining \(x(k+1) = \Psi x_d\). Using (7) and (8) this requirement expands to

\[
x(k) + h \cdot [f(x(k)) + B \cdot \dot{u}^*_d(k) + C^*u_d(k)] = \Psi x_d(k)
\]

from which the desired control at the time instant \(k\) can be formed as

\[
\dot{u}^*_d(k) = B^{-1} \left( \frac{-x_e(k)}{h} - f(x(k)) - C^*u_d \right)
\]

This function will tend towards infinity as \(h \to 0\), since \((x_e)(h B^{-1}) \to \pm \infty\). However, \(B^{-1} (\cdot f(x_k) - C^*u_d)\) takes finite values. This implies that the bounds for control should be taken into account.

Assuming that the control available may vary within the domain \(||u|| \leq u_B\), where \(u_B\) is a vector constant representing the maximum control output, it is an obvious requirement that the control resources need to be sufficient by satisfying

\[
u_B \geq ||(B^*)^{-1}|| \cdot \left( ||f(x_k) - C^*u_d|| \right)\]

This necessary condition can be used to modify the control signal to

\[
u_k = \begin{cases} 
\dot{u}_d^* & \text{if } ||u_d^*|| \leq u_B \\
\frac{u_B}{u_d^*} & \text{if } ||u_d^*|| > u_B
\end{cases}
\]

will return the control values to the admissible domain as will be proven. Considering the situation in which \(||u_d^*|| > u_B\), we have

\[
\Sigma(k+1) = \Sigma(k) + Ah (f^*(x) + B^*u_k + C^*u_d)
\]

hence

\[
\Sigma(k+1) = \left( \Sigma(k) + Ah (f^*(x) + C^*u_d) \right) \cdot \left( 1 - \frac{u_B}{||u_d^*||} \right)
\]

It follows that

\[
||\Sigma(k+1)|| \leq ||\Sigma(k) + Ah (f^*(x) + C^*u_d)\| \left( 1 - \frac{u_B}{||u_d^*||} \right) \leq \frac{||\Sigma(k)\| + ||Ah (f^*(x) + C^*u_d)\|}{(||B^*)^{-1}||} \leq \frac{||\Sigma(k)||}{(||B^*)^{-1}}
\]

therefore \(\Sigma(k)\) decreases and after a finite amount of steps, \(u_d^*\) will be within the admissible domain of \(||u_d^*|| < u_0\).

Conceptually, the approach presented could be viewed as an amalgamation of model predictive and receding horizon control [9][18]. Within the context of model predictive control, the satellite views its current state and determines the required control output to achieve a desired state based upon an internal model. This process is calculated progressively in which the target state approaches monotonously through the discrete time intervals; hence the analogy to receding horizon control. Within the framework adopted here however, sliding states, \(\Sigma\), are developed and a discrete control regime is constructed to enforce \(\Sigma \to 0\) within a finite time; hence the justification for referring to the scheme as 'discrete-time sliding mode control'.

It is important to note that the internal model used within the control strategy can be updated during the control process in order to refine the control process thus implementing an adaptive control element within the scheme, as is completed.
within [17] and [12]. This would be advantageous when internal parameters such as the inertial matrix, $J$, are not known accurately or time varying.

III. sEnglish Paper Concepts and Sentences

The steps of creating an sEnglish paper, i.e. an NLP program, are as follows:

- Create conceptual structures for an application area that describes concept classes and their properties. All classes are derived back to basic MATLAB classes.
- Define relationships between concepts and their properties in sentences. Define action sentences and procedure descriptions in terms of English sentences.

To use the paper you can do the following:

- Define your World Model in terms of concrete objects, people and environment.
- Use sentences to describe change or instruct a personal assistant (program).
- Use the associated MATLAB code to program agents.

For conceptual structures the notation used below is simple. In the hierarchy of concepts outlined $>$ means a root concept with no super classes, $>>$ means that the class is a subclass of the previously class declared with single $>$. Classes declared with $>>>$ are subclasses of the previous class declared with $>>$, and so on. $@$ means declaration of a property for a class, and the text after the colon is the class of the property. char, cell and double refer to MATLAB classes. @@ means constraint declaration. Sub classes inherit properties of super classes. For the satellite sliding mode control example the ontology use is as follows:

ALGEBRAIC AND NUMERICAL CONCEPTS

>variable
  @notation: char
  @assignment: char
  @domain: set
>scalar real variable
>scalar complex variable
>vector variable
>matrix variable

>set -- {1,’a’,[2,3]}
  @members: char
  @members: char
  @members list: cell

>algebraic term
  @member variables: set of char
  @formula: char
  @reduced form: char

>algebraic equation
  @left hand side: algebraic term
  @right hand side: algebraic term
  @unkowns: set of variables

>>algebraic definition
  @defined variable: variable

>algebraic function
  @input variables: set of char
  @formula: char

>numerical function
  >>m-function
    @file name: char
    @input variables: set of char
    @output variables: set of char
    >>>kernel

text : char
  >>sentence
  >>string -- char(round(66+rand(1,20)\*30))
    @@..: size(..,1)==1

>>array : double -- s=rand(10,2)
  >>periodic time axis -- s=0.1*(1:1000)
    @@..: min(size(..))==1

>>vector
  >>repressor vector
  >>classification surface
  >>matrix -- rand(2,2)
    @@..: min(size(..))>1

>>symmetric matrix -- [3 1; 1 2]

>>time lagged array
  >>scalar -- s=1000*randn(1)
    @@..: max(size(..))==1

>>>learning rate -- rand(1)
  @@..: ..>0

>>>integer -- s=round(1000*randn(1))
  @@..: round(..)==..

>>>positive number
  >>>nonnegative number
    @@..: ..>=0

>>>negative number
  @@..: ..<0

>>>imaginary number
  @@..: imag(..)=0 & real(..)==0

>>>complex number
  @@..: imag(..)=0

>>>real number -- s=-round(1000*randn(1))
  @@..: imag(..)=0

>>time period : physical quantity

>quaternion: vector
>unit matrix : matrix

SPACECRAFT SPECIFIC CONCEPTS

>spacecraft : physical object
The sentence types that can be used with these concepts are listed next:

Use 'smc01-control' to obtain control torque T and control force U for spacecraft Spc01 from dynamical state X and guidance reference Xnow and Diffdot.

The X is used to denote the numerical 'X'. Use X to denote 'X'. Use X to denote the numerical 'X'. The X is to denote 'X'.

Compute attitude error Qe from desired position Xdes and dynamical state X.

Compute control torque T and control force U from matrix J2, surface weights Alpha, special dynamical force F, smoothed sign function SG2 and Diffdot.

Compute dynamical state derivative Xdot from dynamical state X for spacecraft Sc01 using control force U, control torque T and guidance reference Xnow.

Compute position velocity error Ve and angular velocity error Oe from dynamical state X, guidance reference Xnow.

Compute guidance omega gradient V om and guidance direction Vx from attitude error Qe and position error Xe.

Compute matrix J2 as the inverse of J.

Compute position error Xe from desired position Xdes and dynamical state X.

Compute the smoothed sign function SG2 from the joint sliding surface G2 with sign threshold 0.02.

Compute special dynamical force F from dynamical state X and surface weights Alpha.

Concatenate vector Vx and vector V om to get vector Xnow.

Define 'bac, cop' as global variables. Let 'bac, cop' be global variables.

Define the joint sliding surface G2 from the position velocity error Ve and angular velocity error Oe using the surface weights Alpha.

Define O as the 'property' of O2. Define O by 'property' of O2.

Let O be the 'property' of O2. Let 'property' of Obj become P.

Define the 'property' of Obj by P. Make P be the 'property' of Obj.

Define surface weights Alpha as "[0.5, 0.5]". Demonstrate sliding mode control.

Display spacecraft movements XX in 'graphs of variables'.
Show spacecraft movements XX in 'animation'.
Get desired dynamical state Xd of spacecraft Sc01.
Get data for spacecraft Sc01 by 'human input'.
Imagine spacecraft movements XX of spacecraft Spc01 for initial dynamical state X0 and desired dynamical state Xd.

Initialize dynamical guidance state X0 with matlab code "[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]".
Initilise matrix Phi as a 'unit matrix'.
Compute dynamical state derivative Xdot for Time , from dynamical state X with numerical Options and Flag.

The computation of 'Theta' is based on the algebraic definition
\[ \Theta = (X^T X)^{-1}X^T Y p \]. Algebraically the 'Theta' is defined by
\[ \Theta = (X^T X)^{-1}X^T Y p \]. The numerical 'Theta' is defined by
\[ \Theta = (X^T X)^{-1}X^T Y p \]. The computation of 'Theta' and 'Theta2' is based on the algebraic definition
\[ \Theta = (X^T X)^{-1}X^T Y p , \Theta_2 = (X^T X)^{-1}X^T Y p \].
Algebraically the 'Theta' and 'Theta2' are defined by
\[ \Theta = (X^T X)^{-1}X^T Y p , \Theta_2 = (X^T X)^{-1}X^T Y p \].
Notations 'Theta' and 'Theta2' are defined by
\[ \Theta = (X^T X)^{-1}X^T Y p , \Theta_2 = (X^T X)^{-1}X^T Y p \].
Quote Q is ‘dinfo’.

These are all the sentences that are used in the following document in the next section. All of its parts, except the italics, can be interpreted by a suitable agent that can parse sEnglish texts.

IV. AN sEnglish PAPER - ILLUSTRATION

This section lists the resulting sEnglish paper from natural language programming of spacecraft control operations using the concepts and sentences as listed above. It starts with the contents of the sEnglish paper.

CONTENTS

1. Conceptual structures

2. Spacecraft systems

Apply sliding mode control
Concatenate vectors to get guidance
Demonstration of sliding mode control of spacecraft
Display imagined movements
Get desired dynamical state
Get future time horizon
Get spacecraft data
Imagine spacecraft movement
Initialize dynamical guidance state

3. Spacecraft control details

Compute attitude error
Compute control torque and force
Compute dynamical state derivative
Compute errors to guidance
Compute gradients of guidance potentials
Compute position error
Compute smoothed sign function
Compute special dynamical force matrix

Define joint sliding surface
Define sliding surface weights
Spacecraft state change

This paper is a PDF version of a document that can be read by agents that have the ability to interpret sEnglish sentences. Paragraphs in italics, such as this, are informal English inserted by the human author of this document to enhance understanding by human readers. All non-itals text, including titles of sections, are precisely formulated and can be interpreted by agents.

A. Conceptual structures used in this paper

This section first outlines the main concepts and objects used, then the basic data object constraints and finally the attributes of concepts and objects that make up the substance of their meaning.

1) The main concept and their relationships: The main concepts are: algebraic equation, algebraic function, algebraic term, attitude error, desired attitude, desired dynamical state, desired position, matrix variable, numerical function, position error, scalar complex variable, scalar real variable, set, spacecraft movements, spacecraft movements, variable, vector variable. These do not have any sup classes, they represent root concepts.

The concepts that are subclasses of larger complex classes are as follows:

An kernel is a special case of m-function and numerical function.

2) The special cases of the most significant concept classes: Special cases of numerical function are m-function and kernel.

3) Attributes of an algebraic definition: An ‘algebraic definition’ has the following properties: its ‘defined variable’ that is a variable, its ‘left hand side’ that is an algebraic term, its ‘right hand side’ that is an algebraic term and its ‘unknowns’ that is a set of variables.

4) Attributes of an algebraic equation: An ‘algebraic equation’ has the following properties: its ‘left hand side’ that is an algebraic term, its ‘right hand side’ that is an algebraic term and its ‘unknowns’ that is a set of variables.

5) Attributes of an algebraic function: An ‘algebraic function’ has the following properties: its ‘input variables’ that is a set of char and its ‘formula’ that is a text.

6) Attributes of an algebraic term: An ‘algebraic term’ has the following properties: its ‘member variables’ that is a set of char, its ‘formula’ that is a text and its ‘reduced form’ that is a text.

7) Attributes of an attitude error: An ‘attitude error’ has the following properties: its ‘dimension’ that can be one of a {‘euler angles’, ‘quaternion’}, its ‘values’ that is a vector and its ‘reference frame’ that is a text.

8) Attributes of a desired attitude: A ‘desired attitude’ has the following properties: its ‘dimension’ that can be one of a {‘euler angles’, ‘quaternion’}, its ‘values’ that is a vector and its ‘reference frame’ that is a text.

113
9) Attributes of a desired dynamical state: A 'desired dynamical state' has the following properties: its 'dimensions' that is a set of char, its 'values' that is a vector and its 'reference frame' that is a text.

10) Attributes of a desired position: A 'desired position' has the following properties: its 'dimension' that can be one of a \{'km','m','mm'\}, its 'values' that is a vector, its 'reference frame' that is a text and its 'time horizon' that is a number array.

11) Attributes of a kernel: A 'kernel' has the following properties: its 'file name' that is a text, its 'output variables' that is a set of char, its 'members' that is a text and its 'members list' that is a cell array.

12) Attributes of an m-function: An 'm-function' has the following properties: its 'file name' that is a text, its 'input variables' that is a set of char and its 'output variables' that is a set of char.

13) Attributes of a position error: A 'position error' has the following properties: its 'dimension' that can be one of a \{'km','m','mm'\}, its 'values' that is a vector and its 'reference frame' that is a text.

14) Attributes of a set: A 'set' has the following properties: its 'members' that is a text and its 'members list' that is a cell array.

15) Attributes of an spacecraft movements: An 'spacecraft movements' has the following properties: its 'craft name' that is a text, its 'data' that is a number array, its 'data labels' that is a set of char, its 'time dimension' that can be one of a \{'s','h','ms'\} and its 'data dimensions' that is a set of char.

16) Attributes of an spacecraft movements: An 'spacecraft movements' has the following properties: its 'time axis' that is a number array, its 'rotations' that is a set of quaternions and its 'translation' that is a set of vectors.

17) Attributes of a variable: A 'variable' has the following properties: its 'notation' that is a text, its 'assignment' that is a text and its 'domain' that is a set.

B. Spacecraft systems

1) Apply sliding mode control: Sentences to use:

Use 'smc01-control' to obtain control torque T and control force U from matrix J2, surface weights Alpha, special dynamical force F, smoothed sign function SG2 and Diffdot. The computation of 'T' is based on the algebraic definition

\[ T = \text{sign}(G2) \times |F| \]

. Execute "Diffdot=zeros(6,1);". Compute control torque T and control force U from matrix J2, surface weights Alpha, special dynamical force F, smoothed sign function SG2 and Diffdot. The computation of 'F' is based on the algebraic definition

\[ F = [\alpha_1 \times F_1; \alpha_2 \times F_2], \quad F_1 = -\omega \times p + v, \quad F_2 = -\omega \times v, \quad p = X^{1:3}, \quad v = X^{4:6}, \quad \omega = X^{11:13} \]

. Finish conditional actions. Natural English comment:

Resulting things are: T(control torque), U(control force).

2) Demonstration of sliding mode control of spacecraft:

Sentences to use:

Demonstrate sliding mode control .

Details of the meaning:

Get data for spacecraft Sc01 by 'human input'. Get desired dynamical state Xd by 'human input'. Get time horizon Tspan by 'human input'. Define dynamical state Xd for time horizon Tspan. Display spacecraft movements XX in 'graphs of variables'.

3) Display imagined movements: Sentences to use:

Display spacecraft movements XX in 'graphs of variables'.

Show spacecraft movements XX in 'animation'.

Available things are: XX( spacecraft movements), U_(quote).

Details of the meaning:

If U_ is 'smc01-control', then do the following. Define surface weights Alpha as "[0.5, 0.5]". Use \( \alpha \) to denote 'Alpha'. Initialise matrix Phi as a 'unit matrix'. Define J as the 'inertia matrix' of Scp01. Compute matrix J2 as the inverse of J. Algebraically the 'J2' is defined by \( J_2 = J^{-1} \). Compute position velocity error Ve and angular velocity error Oe using the surface weights Alpha. The computation of 'Ve' is based on the algebraic definition \( V_e = X_{now}^{1:3}, \quad V_e = X^{4:6} - V_e \). The computation of 'Oe' is based on the algebraic definition \( O_e = X^{11:13} - V_e \).

Define the joint sliding surface G2 from the position velocity error Ve and angular velocity error Oe using the surface weights Alpha. The computation of 'G2' is based on the algebraic definition

\[ G2 = \alpha_1 \times V_e; \alpha_2 \times O_e \]

Compute the smoothed sign function SG2 from the joint sliding surface G2 with sign threshold 0.01. Algebraically the 'SG2' is defined by \( SG2 = G2, \quad |G2| < 0.01 \Rightarrow \text{sign}(G2), \quad |G2| > 0.01 \Rightarrow \text{sign}(G2) \).

114
4) Get desired dynamical state: The desired dynamical state contains position, attitude and their derivatives. It can be obtained in several ways such as by 'human input' at a GUI or by 'plan P2' where the plan P2 may have been prepared by the agent itself.

**Sentences to use:**
Get desired dynamical state Xd by 'human input'.

**Available things are:** U

**Details of the meaning:**
If U is 'human input', then execute "Eud=input('Enter desired attitude Euler angle rotations(www vector in degrees):');Qdes=Euler2Quat3(Eud);
Xd.values=[Xdes';0;0;0;Qdes;0;0;0']'. The qd is used to denote the numerical 'Qdes'. Use xd to denote 'Xdes'. Use xd to denote the numerical 'Xd'.

Resulting things are: Xd(desired dynamical state)

5) Get future time horizon: Time horizon is used in anticipation or future predictions by an agent. In simulation it can be obtained by 'human input'. Normally it is by 'agent input' when it is set by the agent while 'thinking' about alternatives.

**Sentences to use:**
Get time horizon Th by 'human input'.

**Available things are:** U

**Details of the meaning:**
if U is 'human input', then execute "Th=input('Enter future time horizon(s):');".

Resulting things are: Th(time horizon)

6) Get spacecraft data: Spacecraft data can be typed in through a GUI but in general it should be in the agent's memory that keeps updating it if the spacecraft suffers changes.

**Sentences to use:**
Get data for spacecraft Sc01 by 'human input'.

**Available things are:** U

**Details of the meaning:**
Execute "global Bsc;global Sc;Bsc=0;1;get_spacecraft_data_gui('Sc01');while Bsc, pause(0.2);end;Sc01=Sc;".

Resulting things are: Sc01(spacecraft)

7) Imagine spacecraft movement: Imagine is actually simulation if a human observes it. For an agent it is indeed a computation to predict, imagine what will happen. When the simulation is displayed in animation, using the geometric data stored in the spacecraft object, the analogy is even more obvious to human imagination.

**Sentences to use:**
Imagine spacecraft movements XX of spacecraft Spc01 for initial dynamical state X0 and desired dynamical state Xd for time horizon Tspan.

Available things are: Spc01( spacecraft) , X0( dynamical state) , Xd( desired dynamical state) , Tspan( time horizon) .

**Details of the meaning:**
Execute "global Xdes;global Sc01;global Diffdot;
Xdes.values=Xd;Sc01=Spc01". Define X0 as the 'current dynamical state' of Sc01. Execute "Options=odeset('RelTol',1e-13,'AbsTol',1e-22);Diffdot=zeros(6,1);
[t,X]=ode113('spacecraft_state_change',Tspan,X0,Options,[]);
XX.data_labels='x','y','z','vx','vy','vz','q1','q2','q3','q0','ox','oy','oz';
XX.time_dimension=t;XX.data=X;".

Resulting things are: XX(spacecraft movements)

8) Initialize dynamical guidance state: This can be used for dynamical guidance state only.

**Sentences to use:**
Initialize dynamical guidance state X0 with matlab code "[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]".

Available things are: U

**Details of the meaning:**
Execute "X0=U;".

Resulting things are: X0(dynamical guidance state)

C. Spacecraft control details

1) Compute attitude error: **Sentences to use:**
Compute attitude error Qe from desired position Xdes and dynamical state X.

Available things are: Xdes( desired position) , X( dynamical state) .

**Details of the meaning:**
Use \( \alpha \) to denote the numerical 'Alpha'. Algebraically the 'Qe' is defined by \( q = x^{7:10} \), \( Q_e = O^4(q) \). Execute code "Qe=QuatError(X(7:10),Xdes.values(7:10));".

Resulting things are: Qe(attitude error)

2) Compute control torque and force: **Sentences to use:**
Compute control torque T and control force U from matrix \( J_2 \), surface weights Alpha , special dynamical force \( F \), smoothed sign function \( SG2 \) and Diffdot .

Available things are: J2( matrix) , Alpha( surface weights) , F( special dynamical force) , SG2( smoothed sign function) , Diffdot( *) .

**Details of the meaning:**
The \( \alpha \) is used to denote the numerical 'Alpha'. Use \( J_2 \) to denote 'J2'. \( D_\gamma \) is used to denote the numerical 'Diffdot'. Use \( SG_2 \) to denote 'SG2'. The computation of 'U' and 'T'
is based on the algebraic definition

\[ B = [\alpha_1 I_3, 0; 0, \alpha_2 J_2], \quad U_{in} = B^{-1} (D_d - F - SG_2), \]

\[ T = \text{sat}(U_{in}^{(1:3)}), \quad U = \text{sat}(U_{in}^{(4:6)}). \]

Execute code " B=[Alpha(1)*eye(3), zeros(3,3); zeros(3,3), Alpha(2)*J2]; Uin=pinv(B)*(DiffdotF-SG2);U=Tbound(U(1:3),1,0);T=Tbound(U(4:6),1,0)."

Resulting things are: \( T (\text{control torque}) \), \( U (\text{control force}) \).

3) Compute dynamical state derivative: **Sentences to use:**

Compute dynamical state derivative \( Xdot \) from dynamical state \( X \) for spacecraft \( Sc01 \) using control force \( U \), control torque \( T \) and guidance reference \( Xnow \).

Available things are: \( X (\text{dynamical state}) \), \( Sc01 (\text{spacecraft}) \), \( U (\text{control force}) \), \( T (\text{control torque}) \), \( Xnow(\text{guidance reference}) \).

**Details of the meaning:**

Let \( J \) be the 'inertia matrix' of spacecraft \( Sc01 \). Use \( J \) to denote 'J'. Define \( M \) as the 'mass' of \( Sc01 \). Use \( M \) to denote 'M'. The \( X \) is used to denote the numerical 'X'. Algebraically the 'Om' and 'V' are defined by \( \omega = X_{11:13}^{11:13} \), \( v = X^{4:6} \). The numerical 'P' is defined by \( p = (X^{1:3}) \). The numerical 'Q' is defined by \( q = (X^{7:10}) \). The computation of 'F' is based on the algebraic definition

\[ F = [-\alpha \times p + v; -\alpha \times v; 0.5 \times Q^6(om) \times q; J^T \ast \alpha \times (J \ast \alpha)] \]

Execute code " F1= -crossM(X(11:13))*X(1:3) + X(4:6); F2= -crossM(X(11:13))*X(4:6); F3=0.5*crossMM(X(11:13))*X(7:10); F4=inv(J)*crossM(X(11:13))*J*X(11:13); F=[F1,F2,F3,F4].". The numerical 'B' is defined by

\[ B = \begin{bmatrix} 0 & 0 & 1/M; 0 & 0; 0, 0, J^T \end{bmatrix} \]

Execute code " B=[zeros(3,3), zeros(3,3); (1/M)*eye(3,3), zeros(3,3); zeros(3,3), zeros(3,3); zeros(4,3), zeros(4,3); zeros(3,3), inv(J)] ;". Use \( X_{t} \) to denote 'Xdot'. The \( \tau \) is used to denote the numerical 'U'. The \( \tau \) is used to denote the numerical 'T'. Use \( X_{t} \) to denote 'Xnow'. The computation of 'Xdot' is based on the algebraic definition \( X_{t} = F + B \ast [u; \tau] + X_{g} \). Execute code " Xdot=[F+B*[U,T]]; ".

Resulting things are: \( Xdot(\text{dynamical state derivative}) \).

4) Compute errors to guidance: **Sentences to use:**

Compute position velocity error \( V_e \) and angular velocity error \( O_e \) from dynamical state \( X \), guidance reference \( Xnow \).

Available things are: \( X (\text{dynamical state}) \), \( Xnow(\text{guidance reference}) \).

**Details of the meaning:**

Use \( x \) to denote the numerical 'X'. \( x_{e} \) is to denote 'Xnow'. The numerical 'V' is defined by \( v = X^{4:6} \). \( v_{e} \) is to denote 'Ve'. Algebraically the 'Vg' is defined by \( v_{g} = x_{e}^{1:3} \). The computation of 'Ve' is based on the algebraic definition \( v_{e} = v - v_{g} \).

Execute " V=X(4:6,1); Vg=Xnow(1:3); Ve=V-Vg; ". The numerical 'Om' is defined by \( \omega = X_{e}^{11:13} \). \( \omega_{e} \) is to denote 'Oe'. Algebraically the 'Vg' is defined by \( \omega_{g} = x_{e}^{4:6} \). The computation of 'Ve' is based on the algebraic definition \( \omega_{e} = \omega - \omega_{g} \).

Execute " Om=X(11:13,1); Og=Xnow(4:6); Oe=Om-Og; ".

Resulting things are: \( Ve (\text{position velocity error}) \), \( Oe (\text{angular velocity error}) \).

5) Compute gradients of guidance potentials: **Sentences to use:**

Compute guidance omega gradient \( Vom \) and guidance direction \( Vx \) from attitude error \( Qe \) and position error \( Xe \).

Available things are: \( Qe (\text{attitude error}) \), \( Xe(\text{position error}) \).

**Details of the meaning:**

The \( x_{e} \) is used to denote the numerical 'Xe'. \( q_{e} \) is to denote 'Qe'. Numerical 'Vx' is defined by \( V_{x} = -0.05 \ast x_{e} / \| x_{e} \| \). The computation of Vom is based on the algebraic definition \( v_{o} = 0.05 \ast q_{e} / \| q_{e} \| \). Execute " [Vx,Vom]=Potentials(Xe,Qe);".

Resulting things are: \( Vom(\text{guidance omega gradient}) \), \( Vx(\text{guidance direction}) \).

6) Compute position error: **Sentences to use:**

Compute position error \( Xe \) from desired position \( Xdes \) and dynamical state \( X \).

Available things are: \( Xdes (\text{desired position}) \), \( X(\text{dynamical state}) \).

**Details of the meaning:**

Use \( x \) to denote 'X'. The \( x_{d} \) is used to denote the numerical 'Xdes'. Algebraically the 'Xe' is defined by \( x_{e} = X_{d}^{1:3} - x_{d}^{1:3} \).

Execute " Xe=X(1:3)-Xdes.values(1:3); ".

Resulting things are: \( Xe(\text{position error}) \).

7) Compute smoothed sign function: **Sentences to use:**

Compute the smoothed sign function \( SG2 \) from the joint sliding surface \( G2 \) with sign threshold \( 0.02 \).

Available things are: \( G2 (\text{joint sliding surface}) \), \( Q \), \( number \).

**Details of the meaning:**

Use \( \sigma \) to denote 'Sth'. Algebraically the 'SG2' is defined by

\[ SG2 = G2; \text{if} \ |G2| \leq \sigma, = \text{sign}(G2), \text{if} \ |G2| > \sigma \]

Execute " Sth=Q;SG2=SatFunc(G2,Sth); ".

Resulting things are: \( SG2(\text{smoothed sign function}) \).
8) Compute special dynamical force matrix: 

Sentences to use:
Compute special dynamical force F from dynamical state X and surface weights Alpha.

Available things are: X (dynamical state), Alpha (surface weights).

Details of the meaning:
Execute " F=[Alpha(1)*F1; Alpha(2)*F2]; ".

Resulting things are: F (special dynamical force).

9) Define joint sliding surface: 

Sentences to use:
Define the joint sliding surface G2 from the position velocity error V and angular velocity error O using the surface weights Alpha.

Available things are: V (position velocity error), O (angular velocity error), Alpha (surface weights).

Details of the meaning:
Execute " G2=[Alpha(1)*V; Alpha(2)*O]; ".

Resulting things are: G2 (joint sliding surface).

10) Define sliding surface weights: 

Sentences to use:
Define surface weights Alpha as "[0.5, 0.5]".

Available things are: Ln (matlab code).

Details of the meaning:
Use "Alpha" to denote 'Alpha'. Execute "Alpha=Ln; ".

Resulting things are: Alpha (surface weights).

11) Spacecraft state change: 

Sentences to use:
Compute dynamical state derivative Xdot from Time, from dynamical state X with numerical Options and Flag.

Available things are: Time (scalar), X (dynamical state), Options (vector), Flag (vector).

Details of the meaning:
Execute " Xdot=Xdes; global Xdes; global Sc01; global Diffdot; ".

Resulting things are: Xdot (dynamical state derivative).

V. CONCLUSIONS

The paper provided a sliding mode control solution for satellite robots and also illustrated the use of an sEnglish paper to program this control task for an agent. This type of programming can significantly reduce human effort to code complex systems, such as agents with artificial intelligence. The use of sEnglish allows human scientific research results in control, mathematics and natural sciences to be passed on to machines.

This paper is a call for participation in a community effort to use NLP and sEnglish in our work in academia and companies. A community website has been established at sysbrain.org.

REFERENCES

A new thin membrane rotary motor for robotic applications based on the dielectric elastomer actuator

Iain A. Anderson, Emilio P. Calius, Todd Gisby, Thomas McKay, Benjamin O’Brien, and Scott Walbran

Abstract—We present a novel thin rotary motor that is based on dielectric elastomer actuators (DEA). Dielectric elastomers are a type of soft electroactive polymer with impressive performance characteristics, including high energy density. DEA technology enables motor designs that are very thin and lightweight. Proof of concept motors have been fabricated and tested in our laboratory. These have produced good torque and power characteristics when compared with conventional small motors. We envisage substantial improvements for the DEA rotary motor with further development. This will pave the way for a wide range of applications in autonomous and mobile robotics.

Keywords: Rotary Motor, Dielectric Elastomer Actuator

I. INTRODUCTION

Rotary motion is a key element of robotics and rotary motors are in widespread use in robotic systems of every kind. In this paper we describe a novel rotary motor based on a polymeric artificial muscle technology that has the potential to open new horizons in robotic design. This motor’s driving mechanism consists of soft dielectric elastomer actuator (DEA) membranes and a thin plastic supporting structure, and as a result is very low profile, ultra lightweight and flexible, with few discrete components. It is anticipated that these advantages will lead to many applications but make it particularly suitable for mobile and autonomous robots, and facilitate the development of pervasive actuation and many-degree-of-freedom (continuum) manipulators.

A DEA is essentially a compliant capacitor consisting of a soft polymer membrane dielectric with compliant electrodes applied on both sides. The charge accumulated on the electrodes after a voltage is applied gives rise to electrostatic forces that generate deformation in the DEA. Charges of opposite polarity act to draw the positive and negative electrodes together while the charges of the same polarity act to expand the area of the electrodes (Fig. 1). When the charge is removed, the elastic energy stored in the dielectric returns it to its original shape. DEA’s have demonstrated active strains in excess of 300%, strain rates up to 34,000%/s, pressures of 7.7MPa, and energy densities as high as 3.4MJ/m^3 [1].

Fig. 1. Actuation process of DEA

The pressure capable of being generated by a DEA is widely accepted to be defined by equation 1 below [2]:

\[ P = \varepsilon_0 \varepsilon_\text{r} E^2 \]  

(1)

Where \( P \) is the electrostatic Maxwell pressure, \( \varepsilon_\text{r} \) is the relative permittivity of the dielectric material, \( \varepsilon_0 \) is the permittivity of free space (\( \varepsilon_0 = 8.854 \times 10^{-12} \) F/m) and \( E \) is the electric field strength (V/m).

Typically an elastic bias mechanism is put in series with the electro-active elastomer. One way of achieving this is to stretch a dielectric elastomer in a frame and electrode one half of it as depicted in Fig. 2. Actuation causes the thinning of the active region and a consequent apparent reduction in tension. Thus the area of the active part grows larger and this results in a migration of the central zone towards the passive side.

Fig. 2. Deactivated (a) and activated(b) states of a stretched membrane DEA

We have adopted this principle to rotating a shaft using actuation of separate segments of a DEA. When actuated in the manner of a stepper or AC motor, these segments move a central “orbiter” around a rotor in a closed path (Fig. 3 and
The orbiter consists of an internally toothed gear with an internal diameter greater than the rotor, which is an externally toothed gear. When the orbiter contacts the rotor the no-slip condition can only be maintained if the rotor rotates as the orbiter translates. This principle can be seen schematically in Fig. 4 where the outer orbiter moves in a circular path without rotating, and contact with the axially fixed rotor causes the rotor to rotate.

An orbital motor based on gears also has substantial advantages over crank based DEA rotary motors. In a crank based motor (see Fig. 5 and Fig. 6) it is necessary for the stroke of the DEA to be at least equal to twice the radius of the crank (Fig. 6a). This high stroke requires either high voltages, which would increase the chance of motor failure due to dielectric breakdown, or low stretch ratios, which would increase the influence of viscoelastic effects in the membrane and subsequently reduce the frequency at which the motor could be operated. This design also means that the output torque is necessarily coupled to the stroke of the DEA. With the orbital motor design however stroke and torque can effectively be decoupled. The stroke required is determined not by the radius of the rotor gear, but by the necessary clearance between the teeth of the inner and outer gears (Fig. 6b). This enables the motor to work in a low stroke/high force/high speed regime (as higher stretch ratios can be used), all factors that also improve the reliability of DEA devices [3].
Fig. 7: Kinematics of friction based orbital rotary motor. The driving force can never exceed the limitation imposed by the coefficient of static friction.

Other designs for rotary motors based on electro-active materials have also been proposed. A rotary motor that uses an orbiter mechanism was developed by Zhou et al. [4]. Their motor uses magnetostrictive actuators that require bulky hinges to amplify the stroke. Also, a friction drive was used rather than gears. Hunter et al. [5] and Kornbluh et al. [6] present alternative electroactive polymer rotary motor designs. Hunter et al produced a conducting polymer driven crank based rotary motor. Kornbluh et al. produced a dielectric elastomer driven rocker/ratchet based rotary motor.

We have produced a prototype of our motor and measured the torque it can generate at different speeds.

II. MATERIALS AND METHODS

Two motors were fabricated: a three phase with three electroded zones (120°/zone) (Fig. 8) and a four phase with four zones (90°/zone). To begin, a square segment of 3M VHB4905 tape was stretched equibiaxially to 16 times its original area using a purpose built stretching rig and attached to a rigid circular frame. At this stretch ratio the thickness of the membrane was reduced from 500μm in the unstrained state to approximately 31μm. An orbiter gear was fitted to the center of each stretched membrane using a purpose-built positioning jig. Areas internal to each frame were brush painted top and bottom with carbon grease to provide deformable electroded zones, each separated by a narrow insulting dead zone. Electrode pathways were painted on the frame to provide coupling to external electrical control circuitry. Each zone was actuated with voltages up to 2.5kV delivered by an Ultravolt HV (Ultravolt, USA) voltage conditioner. Circuit switching for each zone was provided by high voltage optocouplers (OC100HG, Voltage Multipliers, Inc.) that allowed charging and discharging of each active part of the membrane. These optocouplers were controlled by a Labview program via a National Instruments PCI Data Acquisition card (NI-6221). Separate experiments were conducted in which square and sine wave inputs were used to sequentially activate the phases on each motor.

For the measurement of torque/power characteristics a grooved wheel was attached to the end of the rotating shaft with a string fixed to its outer circumference. The string was wound around a pulley and various masses were hung from it (Fig. 8). Starting with no load the frequency of activation of the motor phases was gradually increased until just before the point where the rotor began to skip and no longer turned with a smooth motion. The corresponding angular velocity of the rotor at this point was taken to be the maximum speed of the motor. Following this a small mass was added to the free end of the string and the frequency of activation of the phases was gradually decreased from the peak no-load speed until the motor was able to lift the mass without skipping/stalling. Once this was achieved the rotor speed and the size of the mass were recorded, the mass was increased and again the frequency of activation was gradually decreased until the motor ceased to skip. This process was repeated until frequency of activation reached 0.1Hz.

Fig. 8: Photograph of torque measurement.

III. RESULTS

Activation of the 3 phase motor with a square wave input (Fig. 9) resulted in a higher peak speed (3.8rad s⁻¹) and higher peak power (1.36mW) compared to the 3 phase motor driven by a sinusoidal input (Fig. 10), however with the sine wave input the peak torque output of the motor increased (from 1.51mNm to 1.72mNm).
Similarly, the 4 phase motor (Fig. 11 and Fig. 12) had a higher peak speed and peak power output when actuated using a square wave input as opposed to the sinusoidal input. However the peak torque for both square wave and sinusoidal inputs were effectively the same.

Using image processing techniques the centre of the orbiter was plotted as it moved throughout the cycle. In Fig. 13 and Fig. 14 the path of the orbiter is shown with and without the rotor gear for both motor types operating as stepper motors. The inner path is with the rotor gear, and the outer path without. It can be seen that the force developed by the motor is proportional to the difference between the outer and inner path at some angle $\theta$ during the cycle and good motor design maximizes the minimum distance around the cycle to eliminate slip.

Figure 13. Trajectory for 3 phase motor operated in half step mode both with and without the rotor.

Figure 14. Trajectory for 4 phase motor operated in half step mode both with and without the rotor.

IV. DISCUSSION

Based on the torque and power versus speed curves the specific power and specific torque outputs of the motors were calculated for both the entire mass of the motor (including the rigid frames, the gears, the rotor shaft and the bearings) and for just the mass of the DEA membrane itself. The results are presented in Table 1 and Table 2. As can be seen the specific power and specific torque of both motors when just the membrane is considered are over two orders of magnitude greater than that calculated when the entire mass of the motor is taken into account. This is due to the fact that in the current configuration the mass of the membrane at
0.256g represents merely 0.28% of the total mass of the motor.

TABLE 1: MOTOR POWER

<table>
<thead>
<tr>
<th>Motor Phase</th>
<th>Signal</th>
<th>Peak power (mW)</th>
<th>Specific Power (mW/g)</th>
<th>Speed at peak power (rad/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Square</td>
<td>1.361</td>
<td>0.005</td>
<td>5.312</td>
</tr>
<tr>
<td>3</td>
<td>Sinusoidal</td>
<td>0.840</td>
<td>0.009</td>
<td>3.276</td>
</tr>
<tr>
<td>4</td>
<td>Square</td>
<td>2.087</td>
<td>0.011</td>
<td>7.950</td>
</tr>
<tr>
<td>4</td>
<td>Sinusoidal</td>
<td>1.480</td>
<td>0.016</td>
<td>5.776</td>
</tr>
</tbody>
</table>

TABLE 2: MOTOR TORQUE

<table>
<thead>
<tr>
<th>Motor Phase</th>
<th>Signal</th>
<th>Peak torque (mNm)</th>
<th>Specific Torque (mNm/g)</th>
<th>Speed at peak torque (rad/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Square</td>
<td>1.513</td>
<td>0.017</td>
<td>5.905</td>
</tr>
<tr>
<td>3</td>
<td>Sinusoidal</td>
<td>1.716</td>
<td>0.019</td>
<td>6.697</td>
</tr>
<tr>
<td>4</td>
<td>Square</td>
<td>1.595</td>
<td>0.017</td>
<td>6.225</td>
</tr>
<tr>
<td>4</td>
<td>Sinusoidal</td>
<td>1.595</td>
<td>0.017</td>
<td>6.225</td>
</tr>
</tbody>
</table>

It is important to note that for a single membrane motor the plastic frames have yet to be optimized and could be reduced by more than 70% by using thinner frames. Alternatively a key advantage of this motor design is that performance can be increased with additional membrane layers stacked on top of each other. In this configuration each layer only increases the thickness of the motor by 31µm and the mass by 0.256g, yet the increase in contact forces able to be delivered between the motor gears will result in improved torque and power output from the motor. To put this in context, the AM-2224 two-phase miniature stepper motor commercially available from Arsape has a mass of 43g and a peak operating torque of 15mNm and power of 510mW. An estimate of how many membrane layers must be added to the 4 phase orbital motor (driven with a square wave) to make it competitive with the AM-2224 is presented. Improvements in materials and design could meet this challenge. Consider equation 1: the Maxwell stress upon which the actuation principle is based is proportional to the relative permittivity of the dielectric material εr and the square of electric field strength E. The Maxwell stress could be boosted, for the same voltage, using a dielectric elastomer with improved permittivity (3M VHB tape has a relative permittivity of only 3 or 4). This can be achieved using high permittivity additives. However, such additives could significantly compromise dielectric breakdown strength, dielectric loss, and failure strain. Perhaps the best road to voltage reduction would be the use of stacked and electroded membrane layers that are significantly thinner than our current membrane design. In this way we could maintain the same electric field but with much reduced voltages. A method to manufacture such a DEA is presented in [9]. Thus there are clear roads for improvement that could substantially reduce voltages for membrane actuation.

The presented rotary motor is inherently scalable, amenable to deposition-based manufacturing approaches, and uses relatively inexpensive materials. It is also silent in operation and provides a good power to weight ratio. A notable feature is that the motor need not be planar, and in fact can be designed to fit curved geometries to allow close integration into robotic components.

V. CONCLUSIONS

We have presented a new design for a DEA based rotary motor that is potentially very thin and flexible. One of the prototypes produced a specific power of approximately 8mW/g (based on the DEA membrane weight), which compares well with other motor technologies. When the mass of the frame was included a peak specific power of 0.022mW/g was calculated. Although significantly lower than the membrane-only case, we expect that this value can be substantially improved beyond that of electromagnetic motors by reducing the frame mass and by using a multilayer DEA configuration. This opens the way for the motor to deliver high torque at low speeds allowing direct drive applications for a range of robotic applications.
REFERENCES


Associating SOM Representations of Haptic Submodalities
Magnus Johnsson and Christian Balkenius

Abstract—We have experimented with a bio-inspired self-organizing texture and hardness perception system which automatically learns to associate the representations of the two submodalities with each other. To this end we have developed a microphone based texture sensor and a hardness sensor that measures the compression of the material at a constant pressure. The system is based on a novel variant of the Self-Organizing Map (SOM), called Associative Self-Organizing Map (A-SOM). The A-SOM both develops a representation of its input space and learns to associate this with the activity in an external SOM or A-SOM. The system was trained and tested with multiple samples gained from the exploration of a set of 4 soft and 4 hard objects of different materials with varying textural properties. The system successfully found representations of the texture and hardness submodalities and also learned to associate these with each other.

I. INTRODUCTION

An efficient multimodal perceptual system should be able to associate different modalities and submodalities with each other. This provides an ability to activate the subsystem for a modality even when its sensory input is limited or nonexistent as long as there are activities in subsystems for other modalities, which the subsystem has learned to associate with certain patterns of activity, which usually comes together with the patterns of activity in the other subsystems. For example, in humans the sensory information gained when the texture of an object is felt in the pocket can invoke visual images of the object or a feeling for its hardness.

To study the association of different submodalities we used the two haptic submodalities texture and hardness, thus gaining experience with these submodalities in robotics as well. There have been some previous studies of texture and hardness in robotics. For example, Hosoda et al [6] have built an anthropomorphic fingertip with distributed receptors consisting of two silicon rubber layers of different hardness. The silicon rubber layers contain two different sensors, strain gauges and polyvinylidene fluoride films, which yield signals that enabled the discrimination of five different materials pushed and rubbed by the fingertip. Mayol-Cuevas et al [14] describes a system for tactile texture recognition, which employs a sensing pen with a microphone that is manually rubbed over the explored materials. The system uses a supervised Learning Vector Quantization (LVQ) classifier system to identify with 93% accuracy 18 common materials after signal processing with the Fast Fourier Transform (FFT).

Edwards et al [4] have used a vinyl record player with the needle replaced with an artificial finger with an embedded microphone to quantify textural features by using a group of manufactured discs with different textural patterns. Campos and Bajcsy [3] explored haptic Exploratory Procedures (EPs) based on human haptic EPs proposed by Lederman and Klatzky, among them an EP for hardness exploration in which the applied force is measured for a given displacement.

We have done some previous experimentation with texture/hardness perception [11]. In this experiments we test our hardness and texture sensors together with a self-organizing systems that develops monomodal as well as bimodal representations of texture and hardness. Our other previous research on haptic perception has resulted in the design and implementation of a number of versions of three different working haptic systems. The first system [7] was a system for haptic size perception. It used a simple three-fingered robot hand, the LUCS Haptic Hand I, with the thumb as the only movable part. The LUCS Haptic Hand I was equipped with 9 piezo electric tactile sensors. This system used self-organizing maps (SOMs) [12] and a neural network with leaky integrators and it successfully learned to categorize a test set of spheres and cubes according to size.

The second system [8] was a system for haptic shape perception and used a three-fingered 8 d.o.f. robot hand, the LUCS Haptic Hand II, equipped with a wrist for horizontal rotation and a mechanism for vertical re-positioning. This robot hand was equipped with 45 piezo electric tactile sensors. This system used active explorations of the objects by several grasps with the robot hand to gather tactile information, which together with the positioning commands to the actuators (thus a kind of pseudoproprioception) were cross-coded by, depending on the version, either tensor product (outer product) operations or a novel neural network, the Tensor Multiple Peak SOM (T-MPSOM) [8]. The cross-coded information was categorized by a SOM. The system successfully learned to discriminate between different shapes as well as between different objects within a shape category when tested with a set of spheres, blocks and cylinders.

The third system [9] was a bio-inspired self-organizing system for haptic shape and size perception based solely on proprioceptive data from a 12 d.o.f. anthropomorphic robot hand with proprioceptive sensors [10]. The system was trained with 10 different objects of different sizes from two different shape categories and tested with both the training set and a novel set with 6 previously unused objects. It was able to discriminate the shape as well as the size of the objects in both the original training set and the set of new objects.

Magnus Johnsson is with the Department of Computer Science and Lund University Cognitive Science, Lund University, Sweden
Magnus.Johnsson@lucs.lu.se

Christian Balkenius is with Lund University Cognitive Science, Lund University, Sweden
Christian.Balkenius@lucs.lu.se
This paper explores a bio-inspired self-organizing texture and hardness perception system, which automatically learns to associate the representations of the two submodalities with each other. The system is based on a novel variant of the SOM, called Associative Self-Organizing Map (A-SOM), and it employs a microphone based texture sensor and a hardness sensor that measures the compression of the material at a constant pressure. The system is bio-inspired in the sense that it employs a variation of the SOM to represent the two submodalities texture and hardness, and the SOM shares many features with brain maps [13]. It is also bio-inspired in the sense that different submodalities are integrated. That different submodalities are integrated in unimodal association areas in the human brain is a well-established fact [15]. The texture sensor is also bio-inspired. Our system is based on the transduction of vibrations from a metal edge and which are transmitted to a microphone. In humans the mechanoreceptors respond to vibrations as well [5].

II. A-SOM

The A-SOM (Fig. 1) can be considered as a Self-Organizing Map (SOM) [12] which learns to associate the activity of an external A-SOM or SOM with its own activity. It consists of an \( I \times J \) grid of neurons with a fixed number of neurons and a fixed topology. Each neuron \( n_{ij} \) is associated with two weight vectors \( w_{ij}^a \in \mathbb{R}^n \) and \( w_{ij}^b \in \mathbb{R}^m \) where \( m \) equals the number of neurons in an external A-SOM or SOM. \( w_{ij}^a \) is initialized randomly to numbers between 0 and 1, whereas all elements of \( w_{ij}^b \) are initialized to 0.

At time \( t \) each neuron \( n_{ij} \) receives two normalized input vectors \( x^a(t) \in \mathbb{R}^n \) and \( x^b(t) \in \mathbb{R}^m \).

The neuron \( c \) associated with the weight vector \( w_{ij}^c(t) \) most similar to the input vector \( x^a(t) \) is selected:

\[
    c = \arg \max_c \{ \| x^a(t) - w_{ij}^c(t) \| \} \tag{1}
\]

The activity in the neuron \( n_{ij} \) is given by

\[
    y_{ij}(t) = \left[ y_{ij}^{\text{input}}(t) + y_{ij}^{\text{extern}}(t) \right] / 2 \tag{2}
\]

where

\[
    y_{ij}^{\text{input}}(t) = G(\| n_{ij} - c \|) \tag{3}
\]

and

\[
    y_{ij}^{\text{extern}}(t) = x^b(t)w_{ij}^b(t) \tag{4}
\]

\( G() \) is a Gaussian function with \( G(0) = 1 \), and \( \| \cdot \| \) is the Euclidean distance between two neurons.

The weights \( w_{ij}^a \) are adapted by

\[
    w_{ijk}^a(t + 1) = w_{ijk}^a(t) + \alpha(t) G_{ijc}(t) \left[ x_k^a(t) - w_{ijk}^a(t) \right] \tag{5}
\]

where \( 0 \leq \alpha(t) \leq 1 \) is the adaptation strength with \( \alpha(t) \to 0 \) when \( t \to \infty \) and the neighbourhood function \( G_{ijc}(t) \) is a Gaussian function decreasing with time.

The weights \( w_{ij}^b \) are adapted by

\[
    w_{ij}^b(t + 1) = w_{ij}^b(t) + \beta x^b(t) \left[ y_{ij}^{\text{input}}(t) - y_{ij}^{\text{extern}}(t) \right] \tag{6}
\]

where \( \beta \) is the constant adaptation strength.

III. SENSORS IN THE EXPERIMENT

The system discussed in this paper employs two sensors (Fig. 2) developed at Lund University Cognitive Science (LUCS). One of these sensors is a texture sensor and the other is a hardness sensor.

The texture sensor consists of a capacitor microphone with a tiny metal edge mounted at the end of a moveable lever, which in turn is mounted on an RC servo. When exploring a material the lever is turned by the RC servo, which moves the microphone with the attached metal edge along a curved path in the horizontal plane. This makes the metal edge slide over the explored material, which creates vibrations in the metal edge with frequencies that depend on the textural properties of the material. The vibrations are transferred to the microphone since there is contact between it and the metal edge. The signals are then sampled and digitalized by a NiDaq 6008 (National Instruments) and conveyed to a computer via a USB-port. The FFT is then applied to the input, thus yielding a spectrogram of 2049 component frequencies.

The hardness sensor consists of a stick mounted on a RC servo. During the exploration of a material the RC servo tries to move to a certain position, which causes a downward movement of the connected stick at a constant pressure. In the control circuit inside the RC servo there is a variable

Fig. 1. The connectivity of the A-SOM network. During training each neuron in an A-SOM receives two kinds of input. One kind of input is the native input, which correspond to the input an ordinary SOM receives. The other kind of input is the activity of each neuron in an associated SOM or A-SOM. In the fully trained A-SOM, activity can be triggered by either native input or by activity in the associated SOM or A-SOM, or both.
resistor that provides the control circuit with information whether the RC servo has reached the wanted position or not. In our design, we measure the value of this variable resistor at the end of the exploration of the material and thus get a measure of the end position of the stick in the exploration. This end position is proportional to the compression of the explored material. The value of the variable resistor is conveyed to a computer and represented in binary form.

The actuators for both the sensors are controlled from the computer via a SSC-32 controller board (Lynxmotion Inc.). The measure of the resistance of the variable resistor in the RC servo for the hardness sensor and the microphone signal of the texture sensor are digitalized using a NiDaq 6008 (National Instruments) and conveyed to the computer via a USB-port.

IV. EXPLORATION OF OBJECTS

The system described in this paper have been trained and tested with two sets of samples. One set consists of 40 samples of texture data and the other set consists of 40 samples of hardness data. These sets have been constructed by letting the sensors explore each of the eight objects described in Table 1 five times.

During the hardness exploration of an object the tip of the hardness sensor stick (Fig. 2d) is pressed against the object with a constant force and the displacement is measured.

The exploration with the texture sensor is done by letting its lever (Fig. 2c) turn 36 degrees during one second. During this movement the vibrations from the metal edge (Fig. 2b) slid over the object are recorded by the microphone (Fig. 2a) mounted at the end of the stick.

The output from the texture sensor from all these explorations has then been written to a file after the application of the FFT. Likewise the output from the hardness sensor has been written to a file represented as binary numbers. The hardness samples can be considered to be binary vectors of length 18 whereas the texture samples can be considered to be vectors of length 2049. The eight objects have various kinds of texture and can be divided into two groups, one with four rather soft objects and one with four rather hard objects. During the exploration, the objects were fixed in the same location under the sensors.

V. EXPERIMENT

Our system is a bimodal model of haptic hardness and texture perception (Fig. 3). It consists of two monomodal subsystems (hardness and texture), which develop monomodal representations (A-SOMs) that are associated with each other. The subsystem for hardness uses the raw sensor output from the hardness sensor, represented as a binary number with 18 bits and conveys it to an A-SOM with 15 x 15 neurons. After training, this A-SOM will represent the hardness property of the explored objects.

In the subsystem for texture, the raw sensor output from the texture sensor is transformed by a FFT module into a spectrogram containing 2049 frequencies, and the spectrogram which is represented by a vector, is in turn conveyed to an A-SOM with 15 x 15 neurons. After training, this A-SOM will represent the textural properties of the explored objects.

The two subsystems are coupled to each other in that their A-SOMs also receive their respective activities as associative input.

Both A-SOMs begun their training with the neighbourhood radius equal to 15. The neighbourhood radius was decreased at each iteration by multiplication with 0.998 until it reached the minimum neighbourhood size 1. Both A-SOMs started out with $\alpha(0) = 0.1$ and decreased it by multiplication with 0.9999, $\beta$ where set to 0.35 for both A-SOMs.

The system was trained with samples from the training set, described in the previous section, by 2000 iterations before evaluation.

VI. RESULTS AND DISCUSSION

The results of the experiment are depicted in Fig. 4. The 6 images depict the centres of activation when the fully trained system was tested with the test set (described above) constructed with the aid of the objects a-h in Table 1. Images 4A, 4B and 4C correspond to the texture representing A-SOM. Likewise the images 4D, 4E and 4F correspond to the hardness representing A-SOM. Each cell in an image represents a neuron in the A-SOM. In the images 4A, 4B, 4D and 4E there are black circles in some of the cells. This means that the corresponding neurons in the A-SOM are the
<table>
<thead>
<tr>
<th>Label</th>
<th>Object</th>
<th>Estimated Hardness</th>
<th>Estimated Texture</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Foam Rubber</td>
<td>Soft</td>
<td>Somewhat Fine</td>
</tr>
<tr>
<td>b</td>
<td>Hardcover Book</td>
<td>Hard</td>
<td>Shiny</td>
</tr>
<tr>
<td>c</td>
<td>Bundle of Paper</td>
<td>Hard</td>
<td>Fine</td>
</tr>
<tr>
<td>d</td>
<td>Cork Doily</td>
<td>Hard</td>
<td>Rough</td>
</tr>
<tr>
<td>e</td>
<td>Wood Doily</td>
<td>Hard</td>
<td>Fine</td>
</tr>
<tr>
<td>f</td>
<td>Bundle of Denim</td>
<td>Soft</td>
<td>Somewhat Fine</td>
</tr>
<tr>
<td>g</td>
<td>Bundle of Cotton Fabric</td>
<td>Soft</td>
<td>Somewhat Fine</td>
</tr>
<tr>
<td>h</td>
<td>Terry Cloth Fabric</td>
<td>Soft</td>
<td>Rough</td>
</tr>
</tbody>
</table>

The eight objects used in the experiment. The objects A–H were used both for training and testing. The materials of the objects are presented and they are subjectively classified as either hard or soft by the authors. A rough subjective estimation of their textural properties is also provided.

Fig. 3. Schematic depiction over the architecture of the haptic hardness and texture perception system. The system consists of two monomodal sub-systems, which develop monomodal representations (A-SOMs) of hardness and texture that learn to associate their activities. The hardness subsystem uses the raw sensor output from the hardness sensor as input to an A-SOM, which finds a representation of the hardness property of the explored objects. The texture subsystem transforms the raw sensory data by the aid of a FFT module and then forwards it to another A-SOM, which finds a representation of the textural properties of the explored objects. The two A-SOMs learn to associate their respective activities.

Fig. 4A depicts the texture representing A-SOM in the fully trained system when tested with the test set (only native texture input). As can be seen, most objects are mapped at separate sites in the A-SOM (c, d, e, f, h). There are some exceptions though, namely a, b and g. So the system is able to discriminate between individual objects when provided with native input only, although not perfectly.

The hardness representing A-SOM in the fully trained system when tested with the test set (only native hardness input), depicted in Fig. 4D, also maps different objects at different sites in the A-SOM but not as good as the texture representing A-SOM. The hardness representing A-SOM recognizes b, f and h perfectly and blurs the other more or less. However, the hardness representing A-SOM perfectly discriminates hard from soft objects.

When the texture representing A-SOM receives native texture input as well as external hardness input (as can be seen in Fig. 4B) its activations are very similar to those in Fig. 4A. Likewise when the hardness representing A-SOM receives native hardness input as well as external texture input (as can be seen in Fig. 4E) its activations are very similar to those in Fig. 4D.

Fig. 4C depicts the activations in the texture representing A-SOM when it receives only external hardness input. As can be seen this external hardness input very often triggers an activity similar to the activity following native texture input. Likewise Fig. 4F depicts the activity in the hardness representing A-SOM when it receives only external texture input. Even in this case the external input very often triggers an activity similar to the activity following native input. This means that when just one modality in the system receives input, this can trigger activation in the other modality similar to the activation in that modality when receiving native input. Thus an object explored by both sensors during training with the hardness and texture sensors together with ordinary SOMs [11]. The encirclings are also present in the other four images. This is so because we want to show how the A-SOMs are activated when there are both native and external input provided to the system (4B and 4E), and when there are only external input provided (4C and 4F). External input should be understood as hardness input in the case of the texture representing A-SOM, and as texture input in the case of the hardness representing A-SOM.
of the system can trigger a more or less proper activation in the representations of both modalities even when it can be explored by just one sensor during testing. However, as can be seen in Fig. 4C and Fig. 4F, the activity triggered solely by external input does not map every sample properly. The worst cases are the objects c, d and g in the texture representing A-SOM (Fig. 4C) and the objects a, b and g in the hardness representing A-SOM (Fig. 4D). As can be seen in Fig. 4D, the objects c, d and g are not distinguishable in the hardness representing A-SOM, and the objects a, b and g are not distinguishable in the texture representing A-SOM (Fig. 4A). Thus the external activity patterns for these objects are overlapping and the receiving A-SOM cannot be expected to learn to map these patterns correctly even if the objects where well separated by the A-SOM when it received native input.

VII. CONCLUSION

We have experimented with a bimodal self-organizing system for object recognition, which is based on textural and hardness input and with associated representations of the two submodalities. The texture sensor employed is based on the transmission of vibrations to a microphone when the sensor slides over the surface of the explored material. The hardness sensor is based on the measurement of displacement of a stick when pressed against the material at a constant pressure. The results are encouraging, both for the developed monomodal representations and the systems ability to associate the activity in these representations. The system is able to discriminate individual objects based on input from each submodality and to discriminate hard from soft objects. In addition input to one submodality can trigger an activation pattern in the other submodality, which resembles the pattern of activity the object would yield if explored with the sensor for this other submodality.

Our experiments with texture complement those done by Edwards et al [4] and Hosoda et al [6] because they only show that the signals from their sensors are in principle useful as texture sensors whereas we actually implement a working self-organizing system. When compared to the work done by Mayol-Cuevas et al [14] our texture experiments differ in that we use a sensor that is not manually rubbed over the material as their pen, but moved by an actuator built into the sensor. A couple of extensions in our experiments when compared to all the previously mentioned experiments and to the work done by Campos and Bajcsy [3] are that we also experimented with both hardness and texture and the association between these two submodalities.

Because of the successful approach to base a texture sensor on a microphone and base hardness perception on the measurements of displacements at a constant applied pressure, we will in the future try to integrate this approach with our haptic systems. We will also continue our experimentations with the A-SOM. We will continue by testing the ability of the A-SOMs when there are very many categories to see if the A-SOM works equally good in that case. We will also try to implement an extended version of the A-SOM, which can be associated with several external A-SOMs or SOMs. In this way we could build multimodal systems, which when receiving input from just one modality would trigger proper activation patterns in the other modalities as well. This extension should be quite straightforward. It could be done by just adding a new weight vector to each neuron for every new associated A-SOM or SOM. The activity of the neurons would be calculated by adding the native activity and the activities coming from all associated A-SOMs or SOMs and divide the sum with the total numbers of activities.

Another very interesting continuation, since we focus much of our research on haptics, would be to test this A-SOM technology in systems that integrate visual and haptic subsystems. In this way we could probably get a visual system to trigger a proper apprehension of a robot hand, in this very bio-inspired way, when it is about to grasp an object.

REFERENCES

Fig. 4. The mapping of the objects used in the experiments. The characters a-h refer to the objects in Table 1. The images in the uppermost row correspond to the texture representing A-SOM and the images in the lowermost row correspond to the hardness representing A-SOM. Each cell in an image represents a neuron in the A-SOM, which consists of $15 \times 15 = 225$ neurons. A filled circle in a cell is supposed to mean that that particular neuron is the centre of activation for one or several explorations. The occurrence of a certain letter in the rightmost images means that there are one or several centres of activation for that particular object at that particular place. The centres of activation from the samples in the test set corresponding to each object in Tab 1 when only native input was provided have been encircled in the images. A: The texture representing A-SOM when tested with native texture input. Most objects are mapped at separate sites so the system is able to discriminate between individual objects when provided with native input, although not perfectly. B: The texture representing A-SOM when tested with native texture input together with external hardness input. Its activations are very similar to those in A. C: The texture representing A-SOM when it receives only external hardness input. This often triggers an activity similar to the activity following native texture input. D: The hardness representing A-SOM when tested with native hardness input maps different objects at different sites and it perfectly discriminates hard from soft objects. E: The hardness representing A-SOM when tested with native hardness input together with external texture input. Its activations are very similar to those in D. F: the hardness representing A-SOM when it receives only external texture input. This often triggers an activity similar to the activity following native hardness input.
Abstract—An eye tracking study of people engaged in a cloth sorting task is reported. Both eye fixations and hand movements were tracked during the experiments. The eye and hand movements were categorized and presented along a timeline. The observations are consistent with a model of task-driven covert global processing of the scene followed by overt sequential processing of fixated areas of interest for grasping. A computational model is presented.

I. INTRODUCTION

The idea of robots helping in the home has existed since the original concept of a robot was conceived many years ago. Whilst inventions such as the washing machine, dishwasher and vacuum cleaner have helped to reduce the time and effort spent doing household chores a significant proportion of our lives are spent doing household jobs [1], [2]. A common theme between many of the most laborious household chores [3] is the ability to grasp and manipulate highly deformable objects such as clothing. If a robot is to become a useful household appliance [4]-[7] then it must be capable of completing challenging real world tasks such as cleaning, washing & ironing [8]. This premise also holds true for the service and manufacturing industry which still rely heavily on intensive human labour to perform fabric manipulation tasks.

Our main research interest is the design of a domestic robot which can perform a practical real world fabric manipulation task such as sorting clothes. There is relatively little research into robot fabric manipulation systems in comparison to robot handling for solid objects. Early work on fabric handling was aimed at the automated manufacture and not for service robotics [9]. Ono et al. [10] were among the first to investigate robot fabric manipulation using a robot hand and computer vision to perform a simple pick and place operation for fabrics on a stacked on a flat surface. More recently they have extended their work by introducing a 3D simulator to help teach the robot how to pick up the fabric [11]. In a more dynamic industrial setup [12] texture information is used to segment a colour image in order to select a fur from a pile of furs. A morphological analysis of the fur is then performed to determine the pick-up location for the robot arm manipulator.

In research aimed at service robotics [13], [14] a vision system uses colour to segment a washed mass, typically found in a domestic environment, to isolate an item of clothing. The grasp location is then found by calculating the centre of segmented region. In the same paper the identification of subsequent grasping points are discovered enabling the task of unfolding the item of clothing to be performed. This is achieved by using visual information for the hemline of the clothing, in a hung-up state, found from the outline in the image and the shadow cast by the hemline. A predefined template for the type of clothing which contains a specific number for each different hemline is then used to help select the best grasping locations needed to re-orientate the item ready for positioning onto a flat surface. The process of identification is then repeated until the desired pose is achieved. If the hemline information is inconclusive then the lowest point on the hanging item is selected as the next grasp point.

Another method of estimating the pose of a hanging clothing item is to use a simulation of a deformable model [15], [16]. By comparing the actual pose of the item against a simulation of the model, the model can be adjusted until it matches the actual pose. It can then be used to determine the best locations to grasp the item in order to perform an unfolding task. One constraint of this method is the amount of prior knowledge required for each item of clothing to initially build and simulate the model. Although to improve the accuracy the model more prior knowledge is required.

One common theme between all these approaches to fabric manipulation is the use of vision to obtain the necessary information needed to decide what to select and where to grasp. So as part of an initial investigation into our design of a domestic robot we observed humans performing everyday fabric manipulation tasks in a domestic, service and industrial environment. From the observations it was unclear exactly what role vision played in the completion of the tasks. Research which has studied the role of human vision in more detail, when everyday tasks such as making a cup of tea [17] or preparing a sandwich [18] are being performed, suggest that it is the requirements of the task which dictate the information needed by the vision system.

We decided to investigate the use of vision by using a head mounted eye tracking system to monitor human eye movements whilst performing set of fabric manipulation tasks. The tasks involved sorting a pile of clothing into two.
separate piles by either their size or colour. This was chosen after initial observations into human fabric manipulation in different environments highlighted sorting as a common task.

This relatively simple task of sorting clothes provides a complex search space containing objects that immediately change shape once handled. Observing in greater detail how humans perform the task we aim to gain useful clues as to what information may be being used by the visual system. While this may not necessarily lead to the design of a robot capable of completing a simple human task it does, however, provide a measure of progress for the development of the robotic system against a successful working system.

This paper details the method used to investigate human eye movements using an eye tracking system whilst a simple fabric manipulation task is being performed. The results and analysis from a set of experiments are presented detailing the findings as well as the limitations of the experimental design. The final sections of the paper discusses the implications of the finding from the experiments and also what conclusions can be drawn from them for the design and implementation of a robotic system.

II. EXPERIMENTAL METHOD

A. Participants

Three volunteers participated in the experiments. Each participant was given an instruction sheet explaining the purpose of the research and the details of the experimental procedure. They were asked to sign a consent form giving their permission to participate in the experiments and also for the subsequent use of any media as part of the research. The eyesight of each participant was sufficient that they could complete each experiment unaided, i.e. without the use of glasses or contact lenses. A fourth volunteer was unable to take part in the experiments due to calibration issues. The length and angle of their eye lashes meant that part of the eye was obstructed preventing the eye tracker from obtaining a consistent reading.

B. Tasks & Instructions

The experiments focused on two short tasks associated with the household laundry process. The tasks required sorting items of clothing, commonly found in a household, into two separate baskets, as shown in Fig. 1, by either size or by colour. The clothing was initially positioned in a random pile on a desk in front of the participant. Before the start of each task, instructions were communicated verbally to the participant. For example, the participant was asked “Please could you sort the pile of clothing by placing the smaller items into one basket and larger items into the other basket”.

Each task is performed whilst sitting at a desk and required a head mounted eye tracking system to be worn. The setup is not identical to that found in a domestic situation where the process would usually be performed standing up, and typically involve a linen basket, the floor or a low surface, such as a bed, to sort items of clothing on. However it was decided that using a seated position at a desk would not affect the validity of the research and would enable the collection of better data. There were several reasons that contributed to the decision to use a seated position for conducting the experiments. The head mounted eye tracking equipment had a cable connecting it to the main processing unit that restricted the range of movement. This reduction in the range of heads movements would help to minimise blurring in images captured by the scene camera and also reduce the risk of the head mount moving which could result in the loss of calibration. Finally, the eye tracking system was unable to capture the full range of eye movements. By using a seated position at a desk it was possible to reduce the range of both eye movements and keep them within the limits of the eye tracking system.

C. Equipment

The experiments used a head mounted Applied Science Laboratories 4000SL eye tracker system to capture the eye movements and produce the point-of-gaze information. The eye information is detected by using an infrared light source attached to the head mount which emits a beam of light off a two way mirror into the eye. The infrared light is reflected back off the eye, bounced off the two way mirror to a sensor and transmitted along a cable to the eye tracking system. The point-of-gaze is created by using a method of calculating the offset between the pupil centre and corneal reflection at a frequency of 50 Hz. The lower sampling rate means the setup was suited to capturing fixation-based

![Fig. 1. Experimental setup.](image-url)
information rather than quicker saccade-based information. The system has at minimum of 40° horizontal and a 30° vertical field of view, a resolution of 0.5° and an accuracy of < 1° rising to < 2° at the periphery. The scene camera has a field of view of 50° and produced a black and white PAL image with a resolution of 768 x 576. Once the point-of-gaze was calculated it was superimposed onto the scene camera image as a set of white cross hairs, as shown in Fig. 2. The image was then captured by computer as a movie file. Due to problems recording the movie file the output image from the eye tracking system could only be captured 25Hz with a resolution of 240x180. This resulted in two sets of point-of-gaze cross hairs being superimposed onto a single image, as shown in Fig. 2. This still enabled fixations to be classified.

### III. Data Analysis

The data analysis was performed manually using still images captured by the head mounted eye scene camera. Where clarification was required or data was missing, as seen in Fig. 2 where an arm is out of view, the image from the static scene camera was used.

Each image represented durations of 40 milliseconds. These were grouped together into events associated to an individual item of fabric. The events were split into either eye movements or physical grasping of an item. These movements were further divided into a list of categories through an iterative method of grounding the observations, reflecting on any anomalies then revising and re-labelling the categories until the results became self consistent following grounded theory methodology [17].

The physical grasps were categorised into either left or right hand grasps. These are displayed in Fig. 3 as blue and green to represent the smaller items and in light blue and light green to represent the larger items.

The eye movements were split into seven main categories, displayed in Table 1. The out of range category represented instances where the point-of-gaze data was not captured by the system because either the eye movements were outside the field of view or when the eyes blinked. The eye movement information for the first 6 items of clothing was not used for the eye analysis because they were in full view of the participants before the experiment began.

### IV. Results

### A. Sorting Behaviour

The sorting of items predominantly involved using both hands in alternate actions. Larger items are sorted individually by gathering into a single hand before being placed into the basket, as illustrated by items 11, 13 and 18 in Fig. 3. The smaller items are initially picked up with one hand then collected together in the other hand before being dropped into the basket. For instance, items 7, 8 & 9 in Fig. 3 are collected together and dropped at the same time. By gathering the smaller items together before placing them into the basket there is a 42% overall reduction in the

---

**Table 1**

<table>
<thead>
<tr>
<th>Type of Fixation</th>
<th>Details of Fixation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grasp point</td>
<td>Fixation at a subsequent grasp point</td>
</tr>
<tr>
<td>Local</td>
<td>Fixation close to previous fixation</td>
</tr>
<tr>
<td>Non-local</td>
<td>Fixation not close to previous fixation</td>
</tr>
<tr>
<td>Manipulated item</td>
<td>Fixation on item during manipulation</td>
</tr>
<tr>
<td>Uncovered area</td>
<td>Fixation at location uncovered by the manipulation of an item.</td>
</tr>
<tr>
<td>Sorting basket</td>
<td>Fixation at one of the sorting baskets</td>
</tr>
<tr>
<td>Out of range</td>
<td>Fixation outside the limit of the eye tracker</td>
</tr>
</tbody>
</table>

---

D. Calibration

Before a participant could perform an experiment they need to be calibrated to the eye tracking system. This involves several stages. Initially the head mounted system was secured onto the head. The position of the infrared camera and the angle of the two way mirror were adjusted to ensure a clear reflection could be captured from the eye and for the required range of eye movements. The scene camera was adjusted so that a nine-point calibration sheet filled the image. This consisted of a large sheet of paper that covered the workspace and contained a dot at each corner, at the centre point along the edges and one in the centre. To calibrate the participant’s were required to kept their head perfectly still by biting onto a bite bar. They were then asked to fixate their eyes at each dot on the calibration sheet whilst the eye tracking system recorded the position of their eyes. The accuracy of the calibration was checked by asking the participant to read the washing instructions on the label on an item of clothing. The accuracy was checked before and after each experiment was sure it was still correct and that no drift had occurred.
In the majority of instances participants fixated directly at the grasp point just prior to grasping the item. Item 16 was technically not fixated; the participant did fixate an area near to the fixation point, but on a neighbouring item.

The fixations at the grasp point are often the first instance that an item has been fixated on. Earlier fixations at items, such as items 11, 13 & 18 in Fig. 3, could be attributed to the item appearing in the line of gaze due to uncovering after moving an item above or drifts in the gaze off the edge of a neighbouring item.

Several fixations were located on the item whilst it was being manipulated. These were usually associated with large items, such as item 11 and item 18 as shown in Fig. 3. Some fixations, although categorised as local, were possibly a stare while the participant was waiting for new visual information, as there was no obvious visual task being conducted, either following an action or planning one, while larger items/hand were passing in the field of view.

Fixations on the sorting baskets occurred periodically prior to placement of large items, for example item 11 and 18, or a collection of smaller items. Occasionally fixations were directed onto items which had been subsequently uncovered by the prior manipulation of an item above it. Item 9 in Fig. 3 shows an example of this type of fixation.

What stands out is the fact that the eye saccades from an item to be grasped to the next item to be grasped. There is no apparent scan of the scene. This suggests that global image processing takes place during fixations. So what is the function of the fixations? The only suggestion we can make is that the control/planning of the grasp requires fixation, giving some special role to the foveal region and reference frame for grasp control.

**B. Conceptual Model**

The conceptual model illustrated in Fig. 4 has been designed to explain the observations from the sorting behaviour. Components supported by the observations are marked by (O). Components hypothesized by the authors are marked by (H). According to the model, a subject first conducts a covert global search of the visual scene (box I). The result of this search is a list of regions of interest (ROIs) (box II) from which a single ROI is selected based on task constraints (box III). Once a ROI has been selected...
the eye saccades towards it (box IV) where a local examination is conducted (box V). The grasp action is then executed ballistically (box VI) before subsequent manipulation actions (Box VII).

At some point during the fixation, when the local examination occurs, the visual processing may return to a covert global search (box I). In the case that a list of ROIs has been memorized, an intermediate step might be to check for the next ROI in the list (box VIII).

V. DISCUSSION

The model provides a computational framework derived from observations made from the sorting behaviour. This is sufficiently detailed to enable a future robot implementation.

The main uncertainty in the model is when a search of the visual field is filtered to produce a single region of interest specific to the task. It is possible that the global search is task dependant and generates only the preferred next ROI. Also it is not clear whether this process runs in parallel or if it occurs after the local examination is complete. The model does, however, reflect that there is no scanning of the scene and that the eyes saccade from one grasp point to the next.

Real-world tasks are multi-faceted and can conclude in satisfactory outcomes, rather than optimal solutions. The model does not show how the task is broken down in smaller subtasks such as the economy of movements observed by the grouping of smaller items together before being placed into the basket. From a robot design perspective this could significantly improve the energy efficiency of the robot as well as giving it human like characteristics. Further work will be focussed in this area.

The simplification of the task combined with the direct fixation at the grasp point immediately before grasping does suggest that the visual information is task driven. This is similar to the finding of other eye tracking experiments, such as making a cup of tea [18] or preparing a sandwich [19], supporting our model that eye fixations are task directed and occur just prior to a motor action.

VI. CONCLUSION

The analysis of eye movements has led to the development of a general computational model for the selection of grasping points during a sorting task. A number of details of this model need to be developed or investigated further, including the specifics of the global image processing for the selection of grasping regions of interest and also the processing of shape for fabric manipulation. An interesting side effect of the implementation of such a model is that the generation of robot eye movements will be similar to that of humans performing the same task. For the observer this may make the robot “understandable”.

REFERENCES

Monocular Omnidirectional Vision based Robot Localisation and Mapping

Chris Burbridge, Libor Spacek, Joan Condell, Ulrich Nehmzow

Abstract—Robot localisation is an important task in mobile robotics. In this paper a 2D simultaneous localisation and mapping implementation that uses only monocular omnidirectional vision is presented. Experiments are performed using a novel hybrid-simulation method to evaluate the SLAM algorithm based on the sensor model and the layout of the environment. Tests are then carried out that show the advantage gained by using omnidirectional vision. The application of this work to a real robot is discussed.

I. INTRODUCTION

The ability to localise in an unknown environment is of key importance in many robot applications, and there has been a large amount of research carried out on the subject. In most robot applications this is realised through a process known as Simultaneous Localisation And Mapping (SLAM) [1], in which a map of the environment is autonomously constructed as the robot moves. The constructed map is subsequently used to aid the localisation, which in turn is used to aid the map building.

SLAM is usually formulated in a probabilistic manner, with the current estimate of the robot position and map being maintained as a probability distribution. The two main approaches used for maintaining the estimate are the extended Kalman filter (EKF), and the particle filter. The extended Kalman filter represents the probability distribution as Gaussian, allowing simple parameterisation as a mean and covariance matrix. This approach is computationally appealing, but can lead to inaccurate estimation when the probability is non-Gaussian. Particle filters attempt to overcome this by representing the distribution with a discrete set of samples referred to as particles, allowing multi-modal non-Gaussian distributions to be modeled. However, this approach generally requires more computational resources in order to maintain the set of samples [2].

The current literature is split between particle filter based algorithms such as FastSLAM [3], and EKF algorithms such as “Divide and Conquer SLAM” [4]. A comparison between an EKF and a particle filter for object tracking is provided by [5].

Until recently the most common sensor used for SLAM was a laser range finder. The slow speed, high cost and limited sensing capabilities of laser range finders has lead many people to investigate the use of vision as a primary sensor for SLAM [6]. However, unlike laser range finders, standard monocular vision does not directly give range information. This can be overcome by using stereo vision, which allows range information to be calculated so that landmarks can be easily initialised into the map as in [7] and [8].

Methods of SLAM with a single camera have also been developed, perhaps the best known of which is “MonoSLAM” by Andrew Davison [9]. In his approach features are only initialised in the map after having been seen in a few consecutive frames, after which their depth can be approximated. A single extended Kalman filter is used to track both landmarks and the camera pose. An alternative to Davison’s approach is that by [10], in which a modified particle filter is used.

Omnidirectional vision has had relatively little attention as a sensor for SLAM, despite being ideally suited to the process of localisation. Features remain in view in omnidirectional images for longer than in regular images, allowing increased robustness, less frequent loop closing and the ability to cope with fewer features in the world.

Kim et al [11] propose a SLAM system making use of omnidirectional stereo vision using only a single camera. They claim that stereo vision provides increased robustness and faster operation as fewer particles are required for the Monte-Carlo type algorithm used. Limited results are provided, but an exact comparison to the ground truth is not given. In a similar method [12], side by side omnidirectional stereo vision is used in a SLAM system.

A method that combines the use of monocular omnidirectional vision with a laser range finder has been developed by [13]. In their approach the omnidirectional vision is used to provide the bearing to vertical edge features, while the laser range finder provides both horizontal line features and the range to the vertical features. Wheel encoder information is used for state prediction.

A real world implementation of an EKF based 3D SLAM system using monocular panoramic vision has been demonstrated by [14]. Harris features are mapped, and an appearance based method is used for loop closing in large scale environments. A comparison between bundle adjustment optimisation based and Kalman filter based methods for SLAM in the context of omnidirectional vision is presented in [15]. Experiments are performed on both simulated and real data, and a method of combining the two approaches is presented.

In this paper we perform experiments to quantify the ad-
vantage gained by using omnidirectional vision over narrow field of view vision. We present a simple SLAM implementation that uses only monocular omnidirectional vision to localise in a 2D environment, and perform experiments using a novel hybrid-simulation method to evaluate the SLAM algorithm based on the sensor model and the environment layout. Using our simulation method we are able to make realistic comparisons within controlled conditions, without resorting to oversimplified dynamic models.

The remainder of this paper is organised as follows. In section II we give details of the simulation method, and detail the EKF based SLAM algorithm implemented. Details of a number of simulation experiments that show the performance of the localisation when changing the number of features and their spatial distribution, the camera field of view, and the sensor noise are given in section III. The application of this work to a real robot is then discussed in section IV, and finally conclusions are drawn in section V. Future work is described in section VI.

II. MONOCULAR OMNIDIRECTIONAL SLAM

A. Simulation

1) A Hybrid Simulation: In order to avoid simulating robot kinematics, the trajectory of a real physical robot being driven around an arena is logged using a Vicon tracking system. This logged trajectory is then used as the trajectory of the robot in the simulation (see figure 1), avoiding any bias towards the simulator caused by over simplified “perfect” trajectories. In addition, we are able to compare the results against the odometry of the real robot.

This “hybrid simulation” method means that the only thing being simulated is the sensor, for which realistic models are more readily available. Throughout the rest of this paper (except section IV), when we write “robot” we are referring to the real physical robot and not a software simulation, and when we write “sensor”, “feature” or “world”, we are referring to a software simulation.

![Diagram](image)

Fig. 1. The hybrid simulation method used. The only thing being simulated is the sensor perception.

2) The World and Sensor: The simulated world is composed of a number of point features, with a given spatial distribution. The sensor is modeled very simply as providing a bearing to each feature relative to the bearing of the robot, with a random amount of additive Gaussian noise. The feature identity is known, thus the data association is known. No image is simulated as in [16], separating the performance of the SLAM algorithm from the performance of the vision algorithm.

This approach may at first appear simplistic but in reality, this separation of the vision from the localisation is beneficial as it allows us to be sure that it is the sensor choice that is being evaluated and not the visual processing method. A suitable method of extracting features from omnidirectional images has been developed by [13], and we provide a simple improvement of the method in section IV-B.

B. The EKF SLAM Algorithm

In any Kalman Filter based SLAM algorithm the system state is stored as a vector. In this algorithm, the state is represented by the robot position \((x, y, \theta)\), it’s angular and linear velocities \((\ell_v, \omega_v)\), and the world positions \((x_f, y_f)\) of all features in the map \(m\).

\[
x = \begin{bmatrix} x & y & \theta & \ell_v & \omega_v & m^T \end{bmatrix}^T
\]

\[
m = \begin{bmatrix} x_{f1} & y_{f1} & x_{f2} & y_{f2} & \ldots & x_{fn} & y_{fn} \end{bmatrix}^T
\]

The aim of the algorithm is to maintain an estimate of the state \(x\) in the form of a probability distribution \(P(x_{t}|x_{t-1}, Z_t)\), where \(Z_t\) is the measurement vector at time \(t\). The Kalman filter models the probability distribution as Gaussian, allowing it to be represented by a covariance matrix \(P\) and the mean state vector \(x\). Any linear function of the state estimation will produce an altered Gaussian distribution, meaning that the Kalman filter is an optimal state filter for linear systems with Gaussian probability distributions.

In an extended Kalman filter (EKF), the non-linear measurement and motion models are linearised about the current state estimate as Jacobian matrices. This means that for non-linear systems the EKF is only a rough non-optimal approximation. However, despite this it is still widely used and in many applications still provides adequate results.

The algorithm follows the usual predict-refine cycle, whereby the state \(x\) is predicted at timestep \(t + 1\), and measurements are used to refine the prediction.

1) Motion Model: A motion model is employed in order to predict the position of the robot at the following time step. Many SLAM algorithms use wheel odometry information as the motion model. This is an appealingly simplistic model, although problems could potentially occur when the accuracy of the odometry varies significantly throughout the robot traversal, requiring additional methods such as [17] in order to estimate the accuracy of the odometry. In addition to this, some small robots such as [18] have no wheel odometry. We therefore use a constant velocity motion model and the kinematics of a differential drive robot as follows.

\[
x_{t+1} = f(x_t, n)
\]

\[
x_{t+1} = f(x_t, 0)
\]

\[
x_{t+1} = x_t +
\]
\[
\begin{bmatrix}
- \ell_v + \ell_v \omega_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
\ell_v + \ell_v \omega_v n \\
0 \\
0 \\
\end{bmatrix}
\]

where \( n = [ \ell_v, \omega_v n ] \) is the process noise.

The covariance of the state estimate is updated using the Jacobian of the motion model with respect to \( x(\Delta f_x) \)
and with respect to \( n(\Delta f_n) \):

\[
P = \Delta f_x P x^T + \Delta f_n Q x^T f_n
\]

where \( Q \) is the covariance of the process noise. \( \Delta f_x \) does not depend upon the map or velocity elements in the state vector, so state augmentation [19] is employed to decrease the computational complexity of this operation.

2) Measurement Model and Estimate Update: The measurement model predicts what the measurement will be from the current state estimate. In this work, the predicted measurement for each feature \( i \) is a single scalar \( \hat{z}_i \) giving the relative bearing to the feature.

\[
\hat{z}_i = h(x, v)
\]

\[
= h(x, 0)
\]

\[
= \tan^{-1}\left( \frac{\frac{x_i}{y_i} - y}{x_i - x} \right) - \theta + v
\]

where \( v \) is the measurement noise.

The state estimate and covariance are refined after each feature measurement \( z_i \) using the standard equations of the EKF as follows:

\[
S = (\Delta h_x P x^T + \Delta h_v R h_v^T)^{-1}
\]

\[
x = x + P x^T h_x S (z_i - \hat{z}_i)
\]

\[
P = (I - P x^T S) P
\]

where \( \Delta h_x \) and \( \Delta h_v \) are the Jacobians of the measurement model (1) with respect to the state and the noise, and \( R \) is the covariance of the measurement noise — in this case simply a scalar representing the variance. \( S \) represents the certainty of the predicted measurement, and can be used to provide a search bound for the feature.

3) Feature initialisation: With only the bearing to a feature able to be measured, measurements from two different views are required in order to initialise a feature. As is generally the case in monocular vision based SLAM [9], several features need to be known and initialised into the map beforehand. Without this or any odometry, the robot would be unable to know how far it had travelled in order to initialise features, and scale would have to be left as an unknown as in [20].

In 2D at least three features are needed to constrain the robot location, one for each unknown \( x, y, \theta \). We start with four known features in order to increase the stability.

As soon as the robot starts moving, new features are tracked for potential mapping. We initialise features by freezing a copy of the robot state at the time a first measurement of a potential feature is made, and use the frozen copy when the feature is measured again with a sufficient baseline as in [21]. This simple feature initialisation method may not perform as well as alternatives such as [9], but it performs adequately for the experiments in this paper. We leave a comparison of feature initialisation methods as future work.

4) Algorithm Overview: An outline of the algorithm is presented in Algorithm 1.

Algorithm 1: SLAM Algorithm

while running

foreach feature in the map

if feature visible

refine state and covariance estimates based on feature measurement

else if feature not found

delete frozen copy of state

end if

else if baseline and disparity above THRESHOLD

insert feature estimate into state.

augment covariance matrix with covariance estimate for new feature based on frozen copy

remove corresponding frozen copy of state

end if

foreach remaining visible feature

freeze a copy of the robot state.

end foreach

end while

III. EXPERIMENTS AND RESULTS

In this section the SLAM algorithm is tested in simulation to evaluate the effect of the sensor field of view, sensor noise, feature distribution and trajectory smoothness on the accuracy of the localisation.

A. Experimental Method

A trajectory was acquired by driving a robot around while capturing the exact position at 30Hz with a Vicon tracking system. The linear and angular velocities were varied throughout the journey as shown in Figures 2, 3 and 4. This trajectory realistically violates the constant velocity model described in section II-B.1, maintaining the applicability of these experiments to real robot applications.

The same trajectory was then used for each experiment. The range of the simulated sensor was fixed at 10 meters, which covered all features placed within the direction of view.

In all experiments, Gaussian noise was added to the measurements with \( \sigma = 0.025 \). This was added to the measurement as a percentage of the field of view of the sensor, simulating the idea that the sensor has a fixed resolution that is ‘spread’ across the field of view. For a field of view of 360° this equates to a standard deviation of 0.9°.
1) **Measuring the Performance:** The performance of the localisation is assessed by calculating the root mean squared error (RMSE) of the position of the robot over the whole journey. The RMSE of the on board odometry system for the above trajectory is 0.296 metres.

In this work no drift error occurs because data association is known and four pre-initialised features are provided. However, when measuring performance in a real-world setup a measure of drift should be included in the evaluation.

**B. The effect of the number of features**

In order to evaluate the effect that the number of features in the environment has on the localisation error, worlds were generated with a number of features uniformly distributed in a $6m \times 6m$ grid centred at the robot start position. The number of features was varied, and the accuracy of the localisation for each was compared. The results are shown in figure 5.

As can be seen from the graph, even with only the four initially known features the algorithm performs well, achieving an average error of approximately 5cm. As more features are added to the world the accuracy increases. However, as the number of features increases, the improvement in accuracy decreases. This would suggest that maintaining a map of only 40 well spaced features will perform almost as well as maintaining a similar map of 50 features. The marginally lower accuracy achieved using the smaller map may be compensated for by the increased speed of the computation.

**C. The effect of the field of view**

The accuracy of localisation achieved when using cameras of different fields of view was compared by running the algorithm in a world of 25 evenly spaced features with fields of view ranging from $120^\circ$ to $360^\circ$. The results are shown in Figure 6.

It is clear that the larger the field of view the more accurate the localisation. This is due to a combination of more features being visible when the angle of view is greater, and those features being more spread out allowing better triangulation. When both the field of view and the number of features in the world are small, there are locations in the trajectory at which no features are visible. At these times the error in the estimated position of the robot will increase until features are visible again, at which point the filter trusts the
measurement model more than the motion model and the robot is re-localised. Since no odometry information is used, the motion model can quickly become inaccurate and large errors in the predicted position result.

D. The effect of measurement noise

To find out what effect the noise present in the measurements has on the accuracy of localisation, we tested the algorithm in a world of 25 evenly spaced features, adding zero mean Gaussian noise levels of $\sigma = 0^\circ, 0.9^\circ, 1.8^\circ, 2.7^\circ, 3.6^\circ, 7.2^\circ$ and $10.8^\circ$ to the measurements. The results are shown in Figure 7.

The results show that the localisation is very sensitive to noisy measurements. This was found to be mostly due to poor initial estimates of newly initialised features, subsequently resulting in detrimental refinement of the robot location estimate.

E. The effect of the robot trajectory

The robot trajectory acquired by driving a real robot around an arena was altered at each time step by a uniformly distributed random distance, and then the SLAM algorithm was run with the new trajectory on an evenly spaced grid of 50 features. Alterations from 0mm to 25mm were tested, and the accuracy of the localisation to the new trajectory recorded. The accuracy was found to remain constant at about 0.01m, suggesting that the motion prediction stage of the SLAM plays only a very small role in the overall success of the algorithm.

This is as expected. Every visible feature refines the prediction, and when using omnidirectional vision there are enough features at every time step for the prediction to play only a small role. In SLAM implementations using real images the prediction is often used in order to select the feature search bounds (as in [9]). In these circumstances, a poor motion model may be more of an issue.

F. The effect of the feature distribution

The distribution of features in the environment is an important factor in the accuracy of the localisation of the robot. To demonstrate this the SLAM algorithm was run in a world of 25 features randomly distributed about the origin, with all 25 features visible in every frame. The random distribution was Gaussian, and the standard deviation was varied between 0m and 2m. The results are shown in Figure 8.

When the features are not very spread out (ie. a small standard deviation) they are mostly at the same bearing as the robot moves away from the origin. This non-uniform distribution of features about the robot causes a decrease in accuracy coming from poor triangulation and the features being more distant and thus more affected by noisy measurements. As the standard deviation of the feature distribution is increased, the features become more spread across the whole of the robot trajectory and the overall accuracy increases.

IV. THE REAL WORLD

In this section the future application of the algorithm on a real robot is discussed. An algorithm capable of extracting the type of features used in the simulation is presented and evaluated.
A. Omnidirectional Vision

Omnidirectional vision can be achieved in many different ways. In this work we use a 190° circular fish eye lens. This is a miniature M12.5 mount 1/3 inch sensor format lens that fits a 30 frames per second (fps) 640 × 480 Unibrain Fire-i firewire camera with a 1/4 inch sensor, providing good use of the available sensor resolution while still maintaining a horizontal field of view of 190°. See Figure 9 for an example image taken with this lens.

B. Detection of Features

Like most omnidirectional sensors, the fish eye lens causes severe distortions to the image that require complicated calibration procedures to rectify [22]. However, by mounting the camera as in Figure 9 and only considering vertical edges in the environment, calibration can be avoided since vertical edges will be projected to approximately radial lines in the image.

This is the same type of feature as used by [13], who also present an algorithm for detecting vertical edges. In their approach, Canny edge detection is first applied to the distorted image, resulting in a list of edge pixels. A polar edge histogram is then constructed, and an adaptive threshold used to select features. Our algorithm uses only the approximate partial derivative of the image in the θ direction, allowing reliable feature detection at high frame rates:

1) Unwrap the original image $I$ into an $m \times n$ rectangular image $U$ by transforming each pixel from polar coordinates to a position in the new image using the nearest neighbour pixel rather than interpolating:

$$U(x, y) = I(c_x + y\sin(2\pi \frac{x}{m}), c_y + y\cos(2\pi \frac{x}{m}))$$

where $(c_x, c_y)$ is the centre of the image circle.

2) Obtain the approximate partial derivative of $I$ with respect to θ ($I_θ$) by convolving $U$ with the template

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

wrapping at the edge of the image. Store the absolute value of the operator response in a new array (same size as $U$).

3) Add the columns of the new array to form a polar edge histogram of the $I_θ$ gradients.

4) Perform efficient 1D non-maximum suppression on the polar histogram.

5) Remaining entries in the histogram represent potential features at a bearing $2\pi \frac{x}{m}$.

Using no interpolation means the computationally expensive unwrapping process only needs to be carried out once, constructing a two dimensional array of pointers to the correct image location. The algorithm can then be applied directly to the original image.

An example of the feature detection is shown in Figure 10. Averaged over 500 different images the feature extraction takes approximately 6ms to locate potential features, leaving 27ms for further processing while still achieving 30Hz.

V. CONCLUSION

We have shown that monocular omnidirectional vision is a viable choice of sensor for SLAM algorithms. Furthermore, we have shown that an algorithm can produce accurate localisation without the aid of external odometry information, and that the SLAM algorithm proposed considerably out performs the on board odometry. The initial steps towards applying this algorithm in real-time on a robot have been shown.

While many SLAM implementations have been proposed and demonstrated, results and comparisons are usually limited to one or two experiments in a single lab. The effect of the environmental setup in the form of available features is often not discussed, and there have been very few scientific comparisons between SLAM implementations. In this paper experiments have been performed to evaluate SLAM based on feature availability and under different sensing capabilities using a simulation approach close to reality.

VI. FUTURE WORK

Future work includes a comparison of Bayesian filtering algorithms for SLAM, the application of the algorithm developed in this paper to a real robot, and the extension of the algorithm to 3D for further experimentation. When
working in 3D alternative feature initialisation methods such as the inverse depth parameterisation used in [9] may prove beneficial.

Data association is essential in any SLAM algorithm, and incorrect association will cause the algorithm to fail. In this paper we have assumed perfect data association, which is only realistic because we work in simulation. We have presented a simple feature detector, but some form of feature description and matching such as image patches or invariant features will be necessary in order to perform data association. This forms part of our ongoing work.

VII. ACKNOWLEDGEMENTS

We thank the University of Essex for use of the Robot Arena, where this research was conducted.

REFERENCES


A Co-evolutionary Approach toward Face Localization in Color Images

Farshid Hajati and Soheila Gheisari
Amirkabir University of Technology, K. N. Toosi University of Technology
E-mails: hajati@aut.ac.ir, gheisari.s@gmail.com

Abstract— In this paper a new method for face localization in color images, which is based on co-evolutionary systems, is introduced. The proposed method uses a co-evolutionary system to locate the eyes in a face image. The used co-evolutionary system involves two genetic algorithm models. The first GA model searches for a solution in the given environment, and the second GA model searches for useful genetic information in the first GA model. In the next step, by using the location of eyes in image the parameters of face's bounding ellipse (center, orientation, major and minor axis) are computed. To evaluate and compare the proposed method with other methods, high order Pseudo Zernike Moments (PZM) are utilized to produce feature vectors and a Radial Basis Function (RBF) neural network is used as the classifier. Simulation results indicate that the speed and accuracy of the new system using the proposed face localization method which uses a co-evolutionary approach is higher than our previous system [10].

I. INTRODUCTION

Automatic face recognition has been an active research topic due to its extensive range of applications, such as access control systems, criminal identification and authentication in secure systems [1]. A face recognition system generally consists of three steps: face localization, feature extraction and classification. In the first step, localization of the face region from 2-D images through segmentation is carried out. This step is very challenging due to unknown position, orientation and scaling of faces in an arbitrary image [2]-[4]. The second step involves determination of pertinent feature vector from the localized image [5], while the third step prepares classification of the face image based on the information created from the feature vectors [5]-[6].

As mentioned, face region segmentation or face localization is a fundamental step in the process of face recognition. The accuracy of the localized face's ellipse has a heavy influence on the recognition performance [7]. Genetic algorithms (GA's) are optimization techniques based on the mechanism of natural selection [8]. Recently, GA's have been widely used as a tool in pattern recognition applications. Because of their problems in convergence, in this paper we use co-evolutionary GA's for face localization. In co-evolutionary systems, by contrast, the fitness of an individual depends not only on how well it solves a problem but also on other individuals [9].

At first, the location of eyes in the face image is extracted by using a co-evolutionary system. Then, by using the location of eyes in the face image, the parameters of face's bounding ellipse (center, orientation, major and minor axis) are computed and the face's ellipse is extracted. Next, for evaluating of the proposed system we use Pseudo Zernike Moments (PZM) to generate the feature vector elements and an RBF neural network with the structure described in [10] as the classifier. Experiments show that the accuracy and speed of the proposed method are higher than the older system [10].

The organization of this paper is as follows: Section II presents the face localization method. Feature extraction and classifier design are described in section III and IV, and finally, sections V and VI present the experimental results and conclusion.

II. FACE LOCALIZATION

Many algorithms have been proposed for face localization and detection. A critical survey on face localization and detection can be found in reference [11]. To ensure a robust and accurate feature extraction that distinguishes between different individuals effectively, the location of the face region should be determined as accurate as possible. In this paper, we have proposed a new method for face localization in color images which uses the co-evolutionary GA's to locate the eyes and compute and extract the face’s ellipse. The proposed face localization method consists of three steps: first, skin filter step which detects the skin area in a face image; second, eyes localization step in which the location of eyes in the face skin region will be detected; and finally, face's ellipse extraction step in which the face's bounding
ellipse is extracted. The more detailed description of these steps is presented in the following:

A. Skin Filter

In face images many information such as background and hairs are redundant for eye detection. So, we first detect the skin area of faces by the skin filter. Also, the appearance of the skin color depends on the lighting conditions; therefore, we use a lighting compensation technique introduced in [12] to normalize the color appearance in face images. In this technique we regard pixels with the top 5 percent of the luma (nonlinear gamma-corrected luminance) values in the image as the "reference white". The $R$, $G$ and $B$ components of a face color image are adjusted so that the average gray value of these reference-white pixels are equal to 255.

The color of face’s skin of human is composed of a combination of blood (red) and melanin (yellow, brown). Therefore, human’s face skin has a restricted range of hues and is somewhat saturated. The skin filter is based on the Fleck and Forsyth algorithm [13]. The skin filter starts by transforming the input $R$, $G$ and $B$ values into log-opponent values:

$$I = \frac{L(R) + L(G) + L(B)}{3}$$  \hspace{1cm} (1)

$$R_g = L(R) - L(G)$$  \hspace{1cm} (2)

$$B_y = L(B) - \frac{L(G) + L(R)}{2}$$  \hspace{1cm} (3)

Where, $L(X) = 105 \log(X+1)$. In the log transformation, 105 is a convenient scaling constant. The log transformation makes the $R_g$ and $B_y$ values intensity independent. Next, smoothed texture and color planes are extracted. The $R_g$ and $B_y$ arrays are smoothed with a median filter. To compute texture amplitude, the intensity image is smoothed with a median filter, and the result is subtracted from the original image. The absolute values of these differences are run through a second median filter. The texture amplitude and the smoothed $R_g$ and $B_y$ values are then passed to a tightly-tuned skin filter. It marks as probable skin all pixels whose texture amplitude is small, and whose hue and saturation values are appropriate. Hue and saturation are simply the direction and magnitude of the vector $(R_g, B_y)$.

Because, the color of facial features such as: eyes, eyebrow and noise is not similar to skin color, these areas will not pass the tightly-tuned skin filter, creating holes in skin region. Therefore, the output of the initial skin filter is expanded by some morphological operations to include these regions. Fig. 1(a, b) shows an input image and its output for skin filter. Now, by means of this segmented color image we can limit the localization process of eyes in the face region and remove redundant information such as background and hairs.

B. Eyes localization

Among the various facial features, eyes are the most prominent features for recognition and estimation of head pose [14]. Most approaches for eye detection in gray scale images are template-based [15], [16]. In these approaches a template of eye is moved over the image and at each place the correlation between the template and subimage is computed. Finally, a place in which amplitude of the correlation is maximum, is selected as the eye location. Because of the large number of data and long time of process, this method is not appropriate for color images. For this reason, we proposed a new method to locate the eyes which uses co-evolutionary systems. A co-evolutionary system is adopted in our work to solve the most troubles of traditional methods [9]. Due to its good reputation in the application of searching and optimization, good template matching performance (in term of speed and matching error) can be expected.

The used co-evolutionary system involves two genetic algorithm models [17]. The first GA model searches for a solution in a given environment, and the second GA model searches for useful genetic information in the first GA model. The resulted system has high search ability due to the co-evolution consisting of these two GA models. As depicted in fig. 2, the first GA model (GA-1) is the traditional GA model, storing individuals that may be "transcripted" by the genetic information which is discovered by the second GA model (GA-2). On the other hand, the GA-2, searches for useful genetic information on the GA-1, and the fitness value of each individual of the GA-2 is calculated by referring to the population distribution of the individuals in the GA-1.
The used co-evolutionary system.

Fig. 2. The used co-evolutionary system.

The GA-1 is the traditional GA model that searches for a good individual fit to the given environment. Each chromosome in this environment has an associated fitness value that is defined as

$$f_i = R_r^i + R_l^i + \sigma \cdot \cos(\theta_i)$$

(4)

$$\theta_i = \tan^{-1}\left(\frac{(y_r - y_l)}{(x_r - x_l)}\right)$$

(5)

Where, $R_r^i$ and $R_l^i$ are correlation values between candidate eye regions and right and left eye templates, respectively. Scaling parameter $0 < \sigma < 1$ is selected experimentally and $\theta_i$ is the angle between the line joining the eyes and the horizontal line ($\theta$ in fig. 5). Because the face images we used for face recognition were frontal and upright images, to eliminate the candidate eyes which are not horizontal, the third term in (4) is added.

Because each eye in the subject image is localized by its coordinates $(x, y)$, each chromosome in the population during the evolutionary search has four genes: the coordinates of the left eye $(x_l, y_l)$ and the coordinates of right eye $(x_r, y_r)$. Chromosomes are coded in binary form as shown in fig. 3. In this form the length of each chromosome is 32.

<table>
<thead>
<tr>
<th>x_r - 8 bits</th>
<th>y_r - 8 bits</th>
<th>x_l - 8 bits</th>
<th>y_l - 8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101101110010100</td>
<td>1100101100011100</td>
<td>1100101000001111</td>
<td>1100101100011100</td>
</tr>
</tbody>
</table>

Fig. 3. The chromosome for localization of eyes.

To improve search ability of GA-1, GA-2 is adopted to inform GA-1 the candidate subspaces to be searched for. The GA-2 individuals consist of 0’s, 1’s and *’s representing genetic information in GA-1. The “superposition” operator uses genetic information in the GA-1 to calculate the fitness values of GA-2 individuals, and the “transcription” operator propagates effective genetic information in GA-2 into GA-1 population. These two operators are shown in fig. 4.

GA-2 searches for useful genetic information in GA-1. Then, the fitness value of $j$-th individual of GA-2, $F_j$, is described as follows:

The superposition operation of each GA-2 onto GA-1 is carried out $n$ times.

(1) First, it is randomly decided which GA-1 individuals are superposed by GA-2 individual $j$.

(2) GA-1 individuals that are superposed by GA-2 individual $j$ are denoted as $i_1, \ldots, i_n$ and the resultant superposed GA-1 individuals are denoted by $i_1^\sim, \ldots, i_n^\sim$.

(3) To calculate the fitness value of GA-2 individual $j$, the fitness evaluation of each of the superposition operations is defined as the contribution of the superposition operation to GA-1 individual defined by the following equation:

$$f_k = \max(0, f_{i_k} - f_{i_1}) \quad k = 1, \ldots, n$$

(6)

Where, $f_{i_k}$ denotes the fitness value of the $i_k$ individual in the GA-1.

(4) Finally, the fitness $F_j$ of GA-2 individual $j$ is given by the following equation:

$$F_j = \sum_{k=1}^{n} f_k$$

(7)

A superposed chromosome is used to calculate the fitness value of GA-2 individuals. The probability of applying the transcription operation is defined as:

$$P = \begin{cases} \frac{f_k}{f_{\max} - f_{\min}} & f_k > 0 \\ 0 & \text{other} \end{cases}$$

(8)

Where, $f_{\max}$ and $f_{\min}$ denote maximum and minimum fitness values in GA-1, respectively, and hence $P$ is a value such that $0 < P < 1$.

By applying this optimization system to the face region extracted in previous step, the position of eyes in
the face image is found precisely. Now, by using the location of eyes we can extract the face's bounding ellipse in the next step.

C. Face's ellipse extraction

In face localization methods, it is well known that faces are characterized by an elliptical shape and an ellipse can approximate the shape of a face [7]. An ellipse is exactly defined by its center \((x_0, y_0)\), its orientation \(\theta\) and the length \(a\) and \(b\) of its minor and major axis, respectively. In this paper, the parameters of face’s bounding ellipse are computed by using the location of eyes in the face region. Experimental results show that the center of face’s ellipse in each person is located on the perpendicular bisector of the line joining the eyes, \(0.43d\) (\(d\) is the distance between the eyes) below the mentioned line. Therefore, the following relations for computing the parameters of face’s ellipse hold (fig. 5):

\[
\begin{align*}
\theta &= \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) \\
x_0 &= (x_l + x_r)/2 + 0.43 \times d \times \sin(\theta) \\
y_0 &= (y_l + y_r)/2 - 0.43 \times d \times \cos(\theta) \\
\beta &= 1.3 \times d \\
\alpha &= 0.9 \times d
\end{align*}
\]

Where, \((x_l, y_l), (x_r, y_r)\) and \(d = (x_l-x_r)^2 + (y_l-y_r)^2)^{1/2}\) indicate the coordinates of left eye and right eye and the distance between the eyes, respectively. Now, by means of these parameters we can accurately extract the face’s ellipse.

![Fig. 5. Computation of face's ellipse parameters.](image)

III. FEATURE EXTRACTION

For feature extraction step, we use Pseudo Zernike Moments. The advantages of these moments are that they are shift, rotation and scale invariant and very robust in the presence of noise [7]. Pseudo Zernike polynomials are orthogonal sets of complex-valued polynomials defined as:

\[
V_{nm}(x, y) = R_{nm}(x, y) \exp\left(j \frac{m \tan^{-1}(y/x)}{x}\right)
\]

(14)

Where \(x^2 + y^2 \leq 1\), \(n \geq 0\), \(|m| \leq n\) and radial polynomials, \(R_{nm}\), are defined as:

\[
R_{nm}(x, y) = \sum_{s=0}^{n-|m|} D_n|m| s \left(\frac{n-s}{2}\right)^2
\]

(15)

Where

\[
D_n|m| = (-1)^{s+m} \frac{(2n+1-s)!}{s!(n-m-s)!/(n-m-s+1)!}
\]

(16)

The PZM of order \(n\) and repetition \(m\) can be computed using the scale invariant central moments \(CM_{p,q}\) and the radial geometric moments \(RM_{p,q}\) as follows:

\[
PZM_{nm} = \frac{n+1}{\pi} \sum_{(n-m-s)\text{even}, s=0}^{n-|m|} D_n|m| s \sum_{a=0\text{ or }b=0}^{m} \binom{k}{m} \binom{j}{b}
\]

(17)

\[
CM_{p,q} = \frac{\mu_{p,q}}{M_{00}^{(p+q+2)/2}}
\]

(18)

\[
RM_{p,q} = \frac{\sum_{x,y} f(x,y) j^{(p-q)/2} x^p y^q}{M_{00}^{(p+q+2)/2}}
\]

(19)

Where, \(x = x-x_0, y = y-y_0\) and \(x_0, y_0, M_{p,q}\) and \(\mu_{p,q}\) are defined as follows:

\[
x_0 = \frac{m_0}{m_0}, \quad y_0 = \frac{m_0}{m_0}
\]

(20)

\[
M_{p,q} = \sum_{x,y} f(x,y) x^p y^q
\]

(21)

\[
\mu_{p,q} = \sum_{x,y} f(x,y) (x-x_0)^p (y-y_0)^q
\]

(22)
Where, \( M_{pq} \) and \( \mu_{pq} \) are geometrical and central moments of the 2-D image \((f(x,y))\), respectively.

IV. CLASSIFIER DESIGN

Radial Basis Function (RBF) neural networks have found to be very attractive for many engineering problems, because: (1) they are universal approximators, (2) they have a very compact topology, (3) their learning speed is very high because of their locally tuned neurons [5]. The RBF neural network has a feed forward architecture with an input layer, a hidden layer and an output layer. In this paper, an RBF neural network is used as the classifier in the face recognition system and the inputs to this network are feature vectors derived from the feature extraction method described in the previous section.

For this propose, we use an RBF neural network with the structure presented in [10] for classification of feature vectors. This structure is depicted in fig. 6.

![Fig. 6. Structure of the RBF neural network used.](image)

In this structure three feature vectors corresponding to \( R, G \) and \( B \) components are fed to the RBF units in a parallel form. In this case, the distance function of RBF units is as follows:

\[
D_i(X) = \frac{\|X_r - C_{ir}\| + \|X_g - C_{ig}\| + \|X_b - C_{ib}\|}{\sigma_i}
\]

(23) Where, \( X_r, X_g \) and \( X_b \) represent the n-dimensional input feature vectors of \( R, G \) and \( B \) components, respectively. \( C_{ir}, C_{ig} \) and \( C_{ib} \) are n-dimensional vectors called the centers of \( R, G \) and \( B \) components of \( i \)-th RBF unit. \( \sigma_i \) is the width of \( i \)-th RBF unit and is selected as:

\[
\sigma_i = \alpha \min_{i \neq j} \frac{\|C_{ir} - C_{ir}\| + \|C_{ig} - C_{ig}\| + \|C_{ib} - C_{ib}\|}{i \neq j}
\]

(24) Where, the parameter \( 0 < \alpha \leq 1 \) has to be selected experimentally.

We set the number of input nodes in the input layer of neural network equal to the total number of feature vector elements extracted for the three color components. The number of nodes in the output layer is set equal to the number of face classes. In this structure, the RBF unit parameters and output connection weights are determined using the procedures presented in [10].

V. EXPERIMENTAL RESULTS

To evaluate the algorithm, we used a database which we gathered from Persian faces. Since our system is going to be practically used for recognition purposes in our laboratory, this database which is a representative of Persian faces would lead to better results. Experimental studies carried out on our database including the frontal color images of 500 faces from 100 individuals with different facial expressions, have been used to evaluate the performance of the proposed method. In this database each person has changed his/her face expression in each of the 5 samples. Some examples of our database images are shown in fig. 8.

![Fig. 8. Some examples of prepared database showing different facial expressions.](image)
In the feature extraction step, because of PZM advantages [7], we extract PZM of localized face images with different orders and different element numbers. Simulation results (Fig. 10) show that the classification error rate using PZM with the order of 9 or higher is lower than the other orders.

Simulation has been done in three steps based on the order (n) of PZM as described in table I. A total of 300 images are used to train and another 200 are used to test (3 images for training and 2 images for testing, for each of the 100 individuals). The neural network classifier is trained in each category based on training images. The outcome of experimental results is shown in the table II.

This table shows that the third category (n=9,…,12) is more appropriate for classification of faces within the proposed system. To compare this system with the older system [10], the classification error and classification time of the two systems are listed in table III. These experiments have been carried out by using a Pentium 4 processor, in the Matlab (version 7.2) environment. The simulation results presented in table III show that the proposed system is faster and more accurate than the older system.

### Table I: Feature Vector Elements Based on PZM

<table>
<thead>
<tr>
<th>Category no.</th>
<th>PZM feature elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n=0, m=0, n=1, m=0, 1, n=2, m=0,1,2, n=3, m=0,1,2,3, n=4, m=0,1,2,3,4, n=5, m=0,1,2,3,4,5, n=6, m=0,1,2,3,4,5,6, n=7, m=0,1,2,3,4,5,6,7, n=8, m=0,1,2,3,4,5,6,7,8</td>
</tr>
<tr>
<td>2</td>
<td>n=6, m=0,1,2,3,4,5,6, n=7, m=0,1,2,3,4,5,6,7, n=8, m=0,1,2,3,4,5,6,7,8, n=9, m=0,1,2,3,4,5,6,7,8,9, n=10, m=0,1,2,3,4,5,6,7,8,9,10</td>
</tr>
<tr>
<td>3</td>
<td>n=9, m=0,1,2,3,4,5,6,7,8,9, n=10, m=0,1,2,3,4,5,6,7,8,9,10, n=11, m=0,1,2,3,4,5,6,7,8,9,10,11, n=12, m=0,1,2,3,4,5,6,7,8,9,10,11,12</td>
</tr>
</tbody>
</table>

### Table II: Experimental Results

<table>
<thead>
<tr>
<th>Category no.</th>
<th>NFE</th>
<th>NM</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>13</td>
<td>6.5%</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>5</td>
<td>2.5%</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
<td>2</td>
<td>1%</td>
</tr>
</tbody>
</table>

1: Number of Feature Element for each component  
2: Number of Misclassification  
3: Error Rate = NM/Number of total testing images

### Table III: Comparison of Two Systems

<table>
<thead>
<tr>
<th>Category no.</th>
<th>NFE</th>
<th>Proposed method</th>
<th>Classification error</th>
<th>Proposed method</th>
<th>Classification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>6.5%</td>
<td>7%</td>
<td>15.5 s</td>
<td>20.1 s</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>2.5%</td>
<td>3%</td>
<td>20.1 s</td>
<td>24.6 s</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
<td>1%</td>
<td>1.2%</td>
<td>28.3 s</td>
<td>33 s</td>
</tr>
</tbody>
</table>

VI. Conclusion

In this paper, a new method for face localization in color images is presented. This new method uses a co-evolutionary approach to locate the eyes in face images. This co-evolutionary system consists of two traditional genetic algorithms. The first GA model searches for a solution in a given environment, and the second GA model searches for useful genetic information in the first GA model. In the next step, by using the location of eyes, the parameters of the face’s bounding ellipse (center, orientation, major and minor
axis) are computed and the face’s ellipse is extracted. To
evaluate the proposed method the PZMs are extracted
and an RBF neural network is used for classification. The
experimental results show that the proposed system is
faster and more accurate than our previous system.

REFERENCES

basis function neural network", Third Int. Conf. On Human and
extraction based on shape and color information", IEEE Int. Conf.
September 1996.
recognition based on moment invariants", Proceeding of IEEE
International Conference on Image Processing, Thessaloniki,
face recognition”, IEEE Trans. On Pattern Recognition and
recognition system using pseudo zernike moment invariant and
radial basis function neural network”, Int. Journal of Pattern
Recognition and Artificial Intelligence, Vol. 17, No. 1, 41-62,
2003.
for Genetic Algorithms”, Proceeding of Fourth Int'l Conf. on
Artificial Neural Network & Genetic Algorithms, Slovinia,
April, 1999.
face localization and recognition in color images", Proceeding of
IEEE Int. Conf. on Systems, Man and Cybernetics, Taipei,
Taiwan, pp. 4214-4219, 8-11 October, 2006.
Vision and image understanding, Vol. 83, No. 3, pp. 236-274,
color images”, IEEE Transaction on pattern analysis and
1, pp. 63-77, August 1999.
[14] A. Nikolaidis and I. Pitas, "Facial Feature Extraction and
to face and eye detection from images with cluttered
background”, Proc. of Int’l Conf. on Pattern Recognition, Vol.
[16] K. M. Lam and H. Yan, “Locating and extracting the eye in
human face images”, Journal of Pattern Recognition, Vol. 20,
No. 5, pp. 771-779, 1996.
coevolutionary mechanism to search for effective genetic
information”, Proceeding of Int’l Conf. on Evolutionary
Internal Agent States: Experiments Using the Swarm Leader Concept

Mohamed H. Mabrouk, Craig W. Murray, Kevin Johnstone and Colin R. McInnes

Abstract — In recent years, an understanding of the operating principles and stability of natural swarms has proven to be a key problem. In this paper, we introduce applications using the swarm leader concept through simulation and test-bed facility. Experimental and simulation results match closely in a way that confirms the efficiency of the algorithm as well as its applicability.

I. INTRODUCTION

Research activity on autonomous robots has witnessed a surge, especially in the field of artificial robotic systems that inspire ideas from real biological systems due to their important commercial applications [1]. Swarms of self-organizing agents that exchange information have a greater functionality than single robots [2]. In these systems, large numbers of identical autonomous robots are controlled using architectures that are inspired from natural systems such as insect swarms, bird flocks and fish schools [3]–[6].

We use a model designed to simulate the motion of a swarm of robots, which consists of $N$ agents. The $i^{th}$ agent is represented with mass $m_i$, position $\mathbf{r}_i$ and relative distance $\mathbf{r}_{ij}$ between the $i^{th}$ and $j^{th}$ agents. The generalized Morse potential, which decays exponentially at large distances and represents a comparatively realistic description of natural swarming agents, is used to define the interactions amongst the swarm agents $V_{\text{interaction}}(\mathbf{r})$, the attraction potential of the goal $V_{\text{goal}}(\mathbf{r}_g)$, and the repulsive potential of the $N_o$ obstacles $V_{\text{obstacle}}(\mathbf{r}_o)$. Unit mass agents are considered for simplicity. To prevent the agents from reaching large speeds, a dissipative frictional force with coefficient $\beta_i$ for the $i^{th}$ agent is added.

The potential is characterized by attractive and repulsive interaction potential fields of strength $C_i$ and $C_r$ with ranges $l_i$ and $l_r$, respectively, while $C_{zz}$, $t_{zi}$, $C_{zz}$ and $l_{zz}$ are the attraction potential strength and range the goal point and the repulsive potential strength and range of the $z^{th}$ obstacle point, respectively. The equations of motion for $N$ agents that contains $N_o$ obstacle points and a goal point at position $G$, are then defined for the $i^{th}$ agent as

$$\mathbf{v}_i = \dot{\mathbf{r}}_i$$

$$m_i \ddot{\mathbf{r}}_i = -\beta_i \mathbf{v}_i - \nabla V_{\text{global}}(\mathbf{r}_i)$$

$$V_{\text{global}}(\mathbf{r}_i) = V_{\text{interaction}}(\mathbf{r}_i) + V_{\text{goal}}(\mathbf{r}_g) + V_{\text{obstacle}}(\mathbf{r}_o)$$

$$V_{\text{global}}(\mathbf{r}_i) = \sum_{j=1}^{N} \left( C_{\text{zz}} e^{- \frac{l_{zi}}{l_{zi}}} - C_{\text{zz}} e^{- \frac{l_{zi}}{l_{zi}}} \right)$$

$$+ \sum_{j=1}^{N_o} \left( C_{\text{zz}} e^{- \frac{l_{roz}}{l_{roz}}} - C_{\text{zz}} e^{- \frac{l_{roz}}{l_{roz}}} \right)$$

In earlier work [7], we introduced sets of first order differential equations to describe the free parameters of the potential field (internal state) to solve the local minimum problem. For artificial potential field based navigation, there have been several attempts to solve the local minimum problem. The problem for a swarm of agents attracted to a goal point at position $G$ is defined such that an artificial potential field at $G$ induces motion towards the goal. However, in order to prevent collision with a static obstacle, an additional repulsive potential field is required. In general, a local minimum may form due to the superposition of the goal potential and that of the obstacles, resulting in the agent, or swarm of agents, becoming trapped in a state other than the goal $G$. Considering this problem, the entire swarm, or part of the swarm will be trapped at the obstacle since the agents trapped inside the obstacle will experience two virtual opposite forces, as shown in Fig. 1. We introduced three new concepts; the swarm leader concept [7], the swarm aggregation concept [8],[9], and the swarm vortex-like behaviour concept [10] to enhance the performance of the internal state model for agents behaviour that allows them to effectively solve this key problem. In this paper, we introduce applications using the internal state model enhanced with the swarm leader concept through simulation and test-bed facility.
II. AGENT INTERNAL STATE MODEL WITH SWARM LEADER CONCEPT

There is a considerable body of research concerning building artificial systems inspired by swarm leaders phenomenon in real biological systems. In [11]–[12], the authors introduced dynamic models based on the distance and the angle between leaders and followers, which indicates that the agents must know who and where their leaders are. Different leader roles were discussed in Wang and [13] and a convergent condition, in which the followers need to have the leaders’ states by sharing global information, was constructed by using contraction theory. In [14], the swarm members know which members are the leaders in a leader based control strategy. Similarly, the followers need to know who the leaders are in [15] where the leader-follower systems are investigated in terms of controllability and optimal control. In [16], the authors discuss the importance of updating the follower information concerning the leader position. Experimentally, much research work focuses on leader-follower relationship in multi-agent systems. In [2], the author tested a set of communication techniques and a library of behaviors, among which follow-the-leader behaviour is programmed, on a swarm of 100 physical robots.

From this background it can be seen that although local information can be used to control the relative distances and angles amongst swarm members, the followers must know which members are their leaders. In addition, the followers need to share global information about the updated position of their leader. In the algorithm used in this paper [7], the agents have information about the leaders by sharing global information, which is the agents potential parameters that are employed to express the leader-follower relationship. The swarm leader concept enables the swarm in Fig. 1 to efficiently solve the problem by making the agents follow the agent that finds a clear way to the goal, as shown in Fig. 2.
III. RESPONSES TO A TEMPORARY LEADER

For a complete understanding of the role of the leader in a free system and how it affects the swarm behaviour using the artificial potential approach for the agents interactions, the motion of a free swarm whose agents experience an attraction to one of them will be considered. For a swarm of agents, let agent \( h \) be the temporary leader that has higher \( C_h \) and \( l_h \) than the other agents of the swarm. Recalling Eq. (1–4) for obstacles and goal free environment and using \( V_h \) instead of \( V_{\text{interaction}} \) for simplicity, the global potential equation for \( h \) agent is

\[
V_{\text{global}}(\mathbf{r}_h) = V_h(\mathbf{r}_h) = \sum_{i=1}^{N_p} \left( C_{i} e^{r_{ij}} |\mathbf{h}_i - \mathbf{r}_{ij}| - C_{i} e^{r_{ij}} |\mathbf{h}_i - \mathbf{r}_{ij}| \right)
\]  

(5)

Defining the swarm center velocity as \( \dot{\mathbf{r}}_s = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{r}_i \)
and noting that \( \nabla V_h(\mathbf{r}_h) = \nabla V_h(\mathbf{r}_h) \mathbf{r}_s \), the equation of motion of the swarm center will be

\[
\dot{\mathbf{r}}_s = -\frac{1}{N_p} \sum_{i=1}^{N_p} \beta \mathbf{r}_i + \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \nabla V_h(\mathbf{r}_h) \mathbf{r}_s = -\mathbf{r}_s + \sum_{i=1}^{N_p} \nabla V_h(\mathbf{r}_h) \mathbf{r}_s
\]

(6)

where \( V_h \) is the interaction potential that affects \( h \) agent by the leader agent \( h \). Since \( \dot{\mathbf{r}}_h = -\mathbf{r}_s \), the double summation in Eq. (6) will cancel to yield

\[
\dot{\mathbf{r}}_s + \frac{\beta}{N_p} \mathbf{r}_s = -\frac{1}{N_p} \sum_{i=1}^{N_p} \nabla V_h(\mathbf{r}_h) \mathbf{r}_s
\]

(7)

which represents a damped oscillator with a forcing term generated by the lead agent. It can therefore be concluded that the agents will be attracted to any agent considered as a temporary goal if it has a larger attraction interaction parameters according to some task; for example a scout agent finding something of interest for the swarm.

IV. SIMULATION RESULTS

A. Escaping a trap I

To show the swarm leader concept, Eq. (1) – Eq. (4) are now used to simulate the agents’ motion for \( N_p \) identical agents which are trapped behind a barrier that consists of \( N_o \) identical obstacle points, as shown in Fig. 3, where \( G \) is a goal point which has an attraction potential of low interaction range \( l_g \). These conditions now make any succeeding agent a temporary leader for the rest of the agents and the swarm center therefore accelerates to the leader position, leading them out of the trap. Then, the leader and subsequently the swarm are attracted to the goal. Figure 3 shows the agents randomly moving inside the trap until one agent succeeds in escaping and so becomes a temporary leader leading the rest of the agents out of the trap.

B. Escaping a trap II (a ‘rescue’ mission)

We now consider another application of internal states using communication through interaction between swarm members. The application is a ‘rescue’ mission that has been given to one of the team members, which uses the internal state model with the swarm leader concept, to assist other team members that use fixed internal states and fail to reach the goal according to the local minimum that forms behind a C-shape obstacle (which has not been prior known).

The key idea is that the rescuing member gains leadership characteristics as it realizes it has a clear way to the goal. This application has the advantage of using the internal state model with only some agents in the swarm whose task will be to act as ‘leaders/scouts’ for the rest of the swarm, mimicking the behaviour in real biological systems [17].

The scenario, shown in Fig. 4, demonstrates a swarm that uses fixed internal states in two groups. The first group has a clear path to the goal while the other group is trapped in a local minimum formed behind a C-shape obstacle. The agents that successfully reach the goal clearly did not lead the individuals of the swarm, which are trapped in the local minimum.
Fig. 3. Swarm leader concept in a trap application, $t=0.5$, $t=9$, $t=12$, $t=15$, $t=18$, $t=22.3$

Fig. 4. Behaviour of a swarm using conventional fixed internal states, $t=2-70$

Fig. 5. $t=1$, $t=10$
We now compare this scenario with another scenario through which one of the agents is assigned the rescue task by using the dynamic internal state model with the swarm leader concept, as show in Fig. 5. As the leader agent has a clear way to the goal, it gains leader properties (large $C_a$) according to the internal state model with the swarm leader concept. This will enable the leader agent to manipulate the potential in the workspace such that the individuals trapped in the local minimum are attracted to it rather than to the goal, as shown in Fig. 5, resulting in successful escape from the trap.

V. EXPERIMENTAL RESULTS

The ‘rescue’ mission application is implemented in a swarming behaviour test-bed: University of Strathclyde. The test-bed has been built to test the behaviour of agents, which interact via pair-wise interactions of short-range repulsions and long-range attraction. Every agent, as shown in Fig. 6, is equipped with an external source of light, a short-range touch sensor to implement the repulsion type forces, and long-range light sensors to implement the attraction type forces. The same procedures in the rescue mission application are implemented using only 3 agents for simplicity.
Fig. 7. Swarm leader concept implementation, $t = 1-36$ sec

- **Fig. 7.a)** $t=2$ sec (agent trapped in local minimum)
- **Fig. 7.b)** $t=18$ sec (leaders approach)
- **Fig. 7.c)** $t=23$ sec (trapped agent is attracted to the leaders)
- **Fig. 7.d)** $t=26$ sec (trapped agent follow the leaders)
- **Fig. 7.e)** $t=29$ sec (trapped agent escapes)
- **Fig. 7.f)** $t=36$ sec (trapped agent reaches the goal)
The scenario of the local minimum problem is implemented using an agent that is attracted to a visible goal (light source located at the goal position) through a transparent barrier. When the agent reaches the barrier, it is repelled according to the function of its touch sensor. However, as the agent moves away from the barrier it is attracted again according to the function of its light sensor and then tries again to reach the goal through the barrier, as shown in Fig. 7.a-b). In this case, the agent gets stuck in a position away from the goal position defining the local minimum problem. In the simulation, using the internal states model enables the stuck agent to solve the problem by following another agent ‘the leader’ that has a clear path to the goal. This idea is implemented through a modification of the leader agents such that their external light source becomes brighter as they have a clear way to the goal (they gain leader properties which is represented by higher $C_1$ according to the internal state model with the swarm leader concept). This enables the leader agents to attract the trapped agent more than the goal such that the individual trapped in the local minimum is attracted to the leaders rather than to the goal, as shown in Fig. 7.c-4).

VI. CONCLUSIONS

This paper presents some applications based on earlier work which aimed to enhance APF based navigation performance of multi-agent systems using one of the most common swarming behaviours in natural systems; the swarm leader concept. Two applications are introduced using simulations. The first application is escaping a trap application, through which one of the agents finds its way through the exit, it will gain a higher attraction potential coefficient $C_1$ and higher attraction potential range $I_a$ becoming a temporarily leader to the rest of the individuals inside the trap to help them escape. The second application is a ‘rescue’ mission application to use the internal state model with only some agents in the swarm whose task will be to act as ‘leaders’ for the rest of the swarm, mimicking the behaviour in real biological systems. The implementation of the second application using a swarming behaviour test-bed confirms the applicability of the used model as well as its ability to enhance the performance for a real swarm of robots.

REFERENCES

A Mathematical Model, Implementation & Study of a Swarm Conglomerate and its Formation Control

Blesson Varghese and Gerard T. McKee

Abstract—The work reported in this paper is motivated by the need to develop mathematical foundations for Swarm Robotics. The work is also inspired by the concepts of Chemical Equilibrium, Particle Physics and behaviors of Dynamic and Open Systems. A model based on mapping a multi robotic system onto the Complex Plane is proposed. The model takes into account the Modeling Axis described in a recent study in Swarm Robotics, incorporating specifically micro and macroscopic models of the system. A number of mathematical parameters describing the behaviors of the system have been investigated. Simulation studies have been developed to validate the model in simple environments with both regular and irregular polygonal robot formations. Physics simulations environments were found to be more suitable for these studies than robot simulators.

I. INTRODUCTION

SWARM Robotics research specifically deals with some classical problems which have been of concern to researchers over the years. An overall review of the work done in Swarm Robotics presented by Bayindir and Sahin [11] identifies some of these problems. It is worth noting that solutions proposed for these problems have been studied and analyzed on specific swarm environments or applications. In general, the problems identified relate to the microscopic (the individual robot behaviors) or macroscopic (the robot swarm behavior) properties of a swarm system. The mathematical definitions proposed in existing work solve the problems temporarily without considering a generic swarm model.

Various solutions have been proposed for problems originating due to the lack of definitions for microscopic behaviors. One such problem is regarding isolated robots independently searching for their peers in order to form into a cluster. This is termed as Coalescence. Recent studies have dealt with reduction of coalescence time and its analysis. Terminology referred to as Meeting Time and Hitting Time conveying the idea of the encounter time between robots has also been introduced [1]. Though a Communication, Mobility and Domain model have been used in the formulation of a mathematical model dealing with Coalescence, a generic swarm model has not been considered.

Another problem which includes Energy Optimization has also been lately investigated. The usual assumption that cost for communication is less than the cost for mobility has been falsified with ample statistical evidence [2]. The necessity for energy optimization has been described and an algorithm using Dijkstra’s shortest path method has been proposed in the approximation of minimal energy path. Though the work presents Energy models, an algorithm and a mathematical model for energy optimization, a swarm model has not been articulated.

Formation Control, yet another issue in swarm robotic research, has been addressed in a recent work by notification of decentralized information governed by a Control law [3]. The Control law, consisting of the sum of a Repulsive Potential and an Attractive Potential field, proved useful for the classical problem of obstacle avoidance. The paper describes the parameters influencing the microscopic behaviors and presents a mathematical model for the control strategy, yet lacks description of a swarm model describing the macroscopic properties.

Another problem is the identification of a common set of coordinates for referencing robot headings [4]. The proposed solution in [4] includes the selection of a pivot robot, maximum likelihood location estimation and heading direction estimation. The solution describes microscopic parameters; however parameters influencing macroscopic behavior and a generic swarm system have not been defined.

The use of potential functions has also been employed to tackle and study the classical problems in swarms. Methods applying potentials and sliding mode control have been recently reported [8, 9]. Studies on stability of swarms have also been pursued and various analytic results have been presented by Gazi and Passino [6]. The same authors have also studied a general class of Attractive or Repulsive functions used to achieve swarm aggregation [7]. Though studies employing potential functions have yielded results, the mathematical models have been constructed devoid of generic swarm models and less consideration of macroscopic behavior of the system.

Another approach employed by researchers to resolve problems in swarms is the behavior based approach. A few ideas of behavior based robotics and its biological inspiration has been presented by Mataric [14]. The characteristics of a behavior based approach to control have also been discussed [15]. On similar lines important aspects of behavior based robotics has been articulated by Birk [12]. These important aspects have been studied on a few applications. The reuse of existing designs, formulated as design patterns have been presented by Graves and
Czarnecki [13]. Formation behaviors for multi-robot teams including the line, column, diamond and wedge geometric formations have been demonstrated and evaluated [11]. These are integrated with obstacle avoidance and other navigational behaviors, and have been implemented on the MissionLab Robot Development Environment [21]. Though the microscopic properties of the system have been defined, the group behavior of a system cannot be explicitly determined by the behavior based approach. Moreover the work done using this approach does not lend itself to mathematical analysis and formulations. Hence, the approach fails in articulating a generic swarm model with macroscopic parameters.

In summary, most of the classical problems in swarm robotics have been defined and solved without consideration of a generic model. The work presented in this paper is motivated towards the development of such a model. The techniques adopted for modeling the swarm are based on an overall taxonomy of swarm robotics literature describing six main axes [10]. The work reported in this paper models a Swarm Conglomerate presented on the Complex Plane for analysis and future works. The microscopic and macroscopic model of the swarm has been specifically dealt with.

While studying the general properties of the modeled system it was essential to incorporate the ideas of equilibrium. The system proposed is dynamic in nature and requires the achievement of equilibrium as in Chemical processes. A swarm model based on a gas expansion model has been reported and its effectiveness proven through self repairing mechanisms of the agents [5]. The swarm model proposed here is unique considering its mobility in an open system environment. Hence, a generic dynamic mathematical model in an open system is proposed.

The necessity to investigate Robotic Development Environments to implement and simulate the system under consideration was inevitable. A survey and analysis paper of various Robotic Development Environments and the potential and limitation of these tools was conducted in depth by Kramer and Scheutz [18]. The swarm model presented in this paper is simulated on the MissionLab toolkit and on Traer Physics, a particle physics simulation engine using the Processing environment. Pattern formations based on regular and irregular polygonal robot formations have been demonstrated.

The work reported in this paper is also on similar lines to the approach of Belta and Kumar [19] [20]. A geometrical approach to control large number of robots in a swarm based on the macroscopic parameters has been proposed in their work. The robotic agents are bound within an abstract shape which is manipulated geometrically by varying its abstract features including the variance and centroid. The group trajectory of the swarm system remains unaltered while changing shape. The formation of the robots in the swarm is as a rigid body. Their notion of rigid formations requires the inter robot distances to remain fixed. Hence, the measure of rigidity is considered with respect to the inter robot distances.

In contrast, the work reported in this paper deals with both the macro and microscopic properties of the system. The term abstraction used by Belta and Kumar relates to the term macroscopic parameters in this paper. Though the macroscopic parameters of the system are controlled in the approach presented here, the microscopic behaviors of the system are also accessible for control. For instance, the individual robot positions define the configuration or shape of the swarm system, though the macroscopic parameters are varied in the approach. The notion of flexibility in rigid patterns is also introduced which facilitates the dynamic behavior of the pattern and the ability to switch between rigid and flexible patterns. The patterns formed by the swarm are also strictly geometric in nature, contributing to the measure of rigidity.

The remainder of the paper is organized as follows. Section II reviews the Modeling Axis which has been the baseline for modeling the system presented in this paper. Section III describes the micro and macro parameters of the swarm model. Simulation and experiments on the Swarm model are presented in Section IV followed by conclusions and future work in Section V.

II. THE MODELING AXIS

A recent study – ‘A Review of Studies in Swarm Robotics’ [10] was a key element that aided in designing and modeling the robotic system under consideration. The authors prescribed modeling as an important factor in comprehending a system and vital for perceiving how the model conceived would behave. The importance of modeling in swarm robotics was further illustrated with two examples on why it was inevitable. Firstly, despite human intervention and periodic maintenance efforts, the possibility of losing robots in experiments still exists. Hence it is suggested to model the experiments and to simulate them. Secondly, the cost of individual robot systems and the requirement for their coordination makes it impractical to use large numbers of real robots in experiments, hence preventing study of scalability of the models to large robot swarms. Though such models are advantageous, mapping such models and simulations onto real world robotic systems is a cumbersome task. It has to be noted that there is an immense difference between simulation results and real world results [10].

The Taxonomy of Swarm Robotics Literature is divided into six major axes [10]: (i) Research Axis, (ii) Modeling Axis, (iii) Behavior Design Axis, (iv) Communication Axis, (v) Analytical Studies Axis and (vi) Problem Axis. The studies on the modeling axis were taken into consideration for modeling the system presented in this paper. The Modeling axis is categorized into four different methods:

1) Sensor based modeling, which incorporates the model of sensors and actuators of the robots and objects in the environment. This is a primitive technique and studies
the interactions of the robot agents with the environment.

2) Microscopic modeling, which models the individual robots of the system and their interactions mathematically. In [10] these models are mathematical definitions of micro behaviors, states and their transitions.

3) Macroscopic modeling, which describes the system as a whole. In [10] a macroscopic model is derived considering difference equations and the system states (variables of the difference equations) at each time step.

4) Cellular Automata modeling, which is the simplest mathematical model of a complex system.

The first three of these methods were incorporated into the modeling of the system proposed in this paper. The sensor based modeling is implicit in the MissionLab simulation studies described in Section IV. The swarm model presented in this paper is based on microscopic and macroscopic modeling, and the latter is the basis for the Traer Physics and Processing simulation studies presented in Section IV.

III. THE SWARM MODEL

A. Inspiration

Analysis and close scrutiny revealed that there was a need for clarity in mathematical models for swarms (Section I). Mathematical definitions describing potential functions, stability and analysis of swarms have been discussed in various research papers [6-9]. Though fruitful work has been undertaken in the field inclusive of these partial mathematical definitions, there persists a deficiency. The advantage of modeling swarms based on mathematical principles enhances their theoretical study. Models derived mathematically also enable the study of parameters that could control the swarm effectively. The Complex Plane was chosen in the work presented here as the basic mathematical framework since it offers tools for studying and implementing mappings and transformations. Transformations when applied to various patterns would enable the geometric configuration of the pattern to be altered. The well known De Moivre’s formula initiated and led to the formulation of the swarm model proposed in the paper.

B. The De Moivre’s formula

If \( z = x + iy \) [16] and is represented in the polar form as \( z = r(\cos \theta + i \sin \theta) \) and \( r \) is called the absolute value or the modulus of \( z \), then

\[
z^n = r^n(\cos n\theta + i \sin n\theta) \text{ for } n = 0, 1, 2,\ldots \quad (1)
\]

To obtain the \( n \)th root of \( z \), the following formula is valid:

\[
\sqrt[n]{z} = \sqrt[n]{r} \left( \cos \left( \frac{\theta + 2k\pi}{n} \right) + i \sin \left( \frac{\theta + 2k\pi}{n} \right) \right) \quad (2)
\]

where \( k = 0, 1, \ldots n-1 \). These \( n \) values lie on a circle of radius \( \sqrt[n]{r} \) with center at the origin and constitute the vertices of a regular polygon of \( n \) sides. If the circle is of radius 1 then the \( n \) vertices are called the \( n \)th roots of unity. The result of joining these \( n \) roots is an \( n \)-sided polygon with vertices located on the circle. The polygon is circumscribed by a circle otherwise referred to as the circumcircle of the polygon. These results were mapped onto the area of swarm robotics. It is assumed that robotic agents are positioned on the vertices of the polygon. Hence the robots form a closed polygon pattern and the system can be mobile with appropriate communication and coordination mechanisms. Figure 1 shows the patterns for \( n = 3 \) to 6. A swarm system would then comprise an aggregation of polygonal patterns. This paper focuses on the behavior of individual polygonal patterns.
C. Microscopic behavior

The microscopic behavior of the system required the design of individual robots and their representation on the plane. Figure 2 shows the robot agent considered as a point P. Assuming that the robot has a communication radius of $\rho$ from P, communication could be established with an adjacent robot if the communication boundaries of the robots intersect with each other.

D. Macroscopic Behavior

The main parameters of the macroscopic model are shown in Figure 3. The angular distance between the robotic agents is denoted by $\theta$. If there are $n$ robots in the swarm at any instance, then the angular distance between these robots is given by

$$\theta = \frac{2\pi}{n}$$

The angular distance is important to determine the coordinates of the robots on the circle.

The distance between each robot in the polygon is a constant if the polygon is of regular type. To compute the distance between the robots the coordinates of the robot have to be known at any instance. The $(h, k)$ values, denoting the centroid of the pattern, are used to compute the coordinates of the robots under consideration and then the Euclidean distance between them are computed. The linear distance $d_{AB}$ is given by the equation:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

The communication radius of a robot is an important factor while considering the above system. The major problem considered is the optimal value for communication. In general, to satisfy the above criterion of robots communicating with peers, it is necessary that the boundary of the communication area of each robot at least intersects at a single point.

The Formation Radius determines the area occupied by the polygonal pattern. This parameter can be used to deflate or inflate the pattern laterally and longitudinally. Hence this parameter acts a control parameter for the maintenance of patterns in the proposed model. The equations that define the formation radii are given by:

$$x_B = h + x_B \times \cos \theta$$
$$y_B = k + y_B \times \sin \theta$$

Varying the magnitude of the formation radii results in regular and irregular patterns:

1) Regular Patterns: These patterns refer to those rules adopted by the swarm which govern the formation of regular polygons. The coordinates of the robots lie on the vertices of a regular polygon. Figure 4 (top) shows scaling (deflate and inflate) of the patterns laterally and

longitudinally with equal magnitude of formation radii. Hence the regularity of the polygon is preserved.

2) Irregular Patterns: These patterns refer to those rules adopted by the Swarm governing the formation of irregular polygons. The coordinates of the robots lie on the vertices of a regular polygon. Figure 4 (bottom) shows the scaling of patterns laterally and
longitudinally with unequal magnitudes of formation radii. Hence a regular polygon could be scaled to an irregular polygon by controlling the formation radii.

IV. SIMULATION STUDIES

Having established the fundamentals of the proposed model, it was necessary to simulate and experiment with various scenarios in which the model would be effective. A pre-requisite involved identifying the right simulation tool for the work. Various simulation tools were identified. The robotic development environment MissionLab was used first in implementing the model.

A. MissionLab Simulations

The MissionLab tool is a result of the initiative of Georgia Tech in implementing a behavior based robotic development Environment. The tool provides both simulation and real time support. Working in the environment comprises designing the Finite State Automata for each robot [21]. The Finite State Automata (FSA) consists of various states, triggers and transitions from one state to the next. The FSA hence allowed the modeling of the robot on the micro level. Each robot and its states had to be manually defined using the FSA. The CfgEdit (Configuration Editor), a GUI based tool, was used for designing robots using FSA. The tool translates the FSA to source code and compiles it to create a robot executable [21].

Several experiments were conducted to investigate the behavior of the robots when obstacles are present and the model consists of 3, 4 or 5 robots. These implementations were done using several overlay files. The overlay files describe the size of the mission area, the origin of the overlay and the obstacles that are placed in the environment [21]. The different overlay files used included:

1) Overlays without obstacles, to examine the random motion and coalescence of the robots of the swarm in such an environment.
2) Overlays with distributed obstacles.
3) Overlays similar to a Martian landscape.
4) Overlays with and without obstacles for 3, 4 and 5 robots to study regular and irregular patterns.

It was noted that the Mission Lab environment was not the best suitable approach for simulation. Firstly, it did not facilitate in scaling the number of robots; each robot had to be manually designed. Secondly, the obstacle avoidance and wandering behaviors were implicitly defined within higher level behaviors and could not be changed at run time. Hence, there was a lack of dynamic control of the behaviors for the model. Thirdly, the simulation results obtained were the outcome of only a partial inclusion of the mathematical model. Fourthly, the equilibrium and particle physics concepts could not be incorporated within the simulation.

Due to these limitations the MissionLab studies were set aside and an alternative simulation platform was sought that...
could meet these challenges. Extended investigations on the Internet for a Physics Simulation Engine led to the discovery of Processing [23] and Traer Physics [24] as a suitable environment for simulations of the model. The simulations were successfully carried out overcoming the challenges in scaling, inclusion of the mathematical model and concepts of Equilibrium and Particle Physics.

B. Processing & Traer Physics Simulations

Processing is an open source programming language and environment enabling visualizations for learning and prototyping [23]. Traer Physics is a particle physics simulation engine for Processing [24]. This environment was found suitable for the simulations of the model proposed. The robotic system was designed as particles in an open environment and is one of the first approaches of its kind. The other notable feature of mapping a robotic model onto a physics simulation environment includes the effect of distinct forces acting on the system.

The model included macro and micro level forces of attraction and repulsion. The macro level forces include repulsive forces, which act on the centroid of the swarm. The forces of repulsion were generated from obstacles in the environment. All the robotic agents align themselves around the centroid with respect to the forces. The computation of the net force acting on the group of robots provided knowledge of the proximity of the system to the obstacle. The group reacts to the net force of repulsion by deflation of the pattern. The group of robots inflates to the normal pattern as the net force acting on the centroid decreases, such as when the system is escaping from obstacles. The intra-agent bonding force and the forces of interaction with the centroid contributed to the micro level forces. The system also enabled the enforcement of a propulsive force on the required particles to trace paths against repulsive forces.

Simulations to obtain regular and irregular patterns of the model in motion were pursued. Regular pattern formations were studied when the model displaced itself through obstacles similar to bridges and tunnels. The model had to deflate itself when subjected to a potential above a minimum threshold, continue motion through the obstacles and inflate beyond the obstacles. Simulation results for regular pattern formation with varying number of robots are presented in figure 5.

Irregular pattern formations were studied when the model displaced itself through obstacles that converged and hence offered a narrowed path of movement. This replicated the motion through a funneled path. The model deflated itself laterally or longitudinally when subjected to a potential above a minimum threshold. The formation radius was used as the control parameter. Simulation results for regular

Fig. 6. Irregular pattern formations for swarm model with distinct number of robots. (i) Inflated swarm model before entry into obstacle path. (ii) Longitudinally deflated swarm model on entry to obstacle path. (iii) Longitudinally deflated swarm model on transit through obstacle path. (iv) Longitudinally inflated swarm model on exit of obstacle path.
pattern formation with various numbers of robots are presented in figure 6.

In summary, the simulation studies confirm the proposed model for polygonal robot configurations with obstacle avoidance. It is observed that in the method employed, obstacle avoidance is an inherent property of the system and is hence automatically guaranteed due to the modeling of obstacles as forces. The requirement to use a particle physics simulation engine illustrates the importance of swarm researchers exploring alternative simulation environments to traditional robot simulation environments in order to investigate general model for swarm systems.

V. Conclusion

A swarm model based on mathematical principles has been derived and simulated. It is interesting to note that though existing robotic development environments failed in providing an accurate implementation, the swarm model was implemented using the concept of Potential fields in a particle physics simulator. This paper builds a bridge between a mathematical model and a swarm model using a particle physics simulation engine. The parameters describing the micro and macroscopic behaviors of the system have been reported. Unlike standard potential field models employing control laws based on an attractive force to the goal, the model proposed in this paper employs a self-propulsive force to steer the robotic model towards its goal.

The modeling and results presented in this paper offer preliminary confirmation of the approach. Future work will aim to refine both the model and offer a richer set of experimental studies. Further enhancements to the model include determining the optimal scaled model for energy consumption, implementing scalable communication radius mechanisms and studying the model on undulated terrains. Patterns formed due to rotation, shearing and transformations will be the consideration for further study. Aggregation of the polygonal patterns to form swarms will also be considered.

REFERENCES


[23] Processing website: http://www.processing.org  

Abstract—This paper presents results obtained whilst attempting to use only image comparison techniques to actively stabilise a pan-tilt camera mounted on an aerial platform. We use the camera to take consecutive images, calculate the difference and pass the output to the control system in order to move the camera to its original location. The goal is to stabilise the camera’s view no matter how far we move the camera. Image processing constraints interact with the control system constraints. In this paper we present various techniques and experiments that are focused on the selection of pixels to be used in generating suitable error surfaces to allow the camera to be returned to its original view. Patch positioning and whole image sampling are the two techniques examined.

I. INTRODUCTION

Aerial photography is often done from light-weight platforms such as radio-controlled aircraft or kites. The advantages of such platforms are that they can be transported to remote places and deployed rapidly. However, they also suffer dramatically from difficult wind conditions, resulting in their on-board camera moving in difficult to control ways. Some stabilisation can be achieved using gyroscope-based systems, which maintain the pose of the camera rather than making sure it always points at the feature of interest.

In this project, we are interested in performing stabilisation using the images themselves, ensuring that they always contain a specified area on the ground. More specifically, patches of images are compared to the current image which is “moved” using an active pan-tilt platform to maintain the correct aim. In this paper we present different schemes to select these patches based on different strategies.

II. ERROR SURFACE

A. What are “Error Surfaces” and how do we use them?

To calculate the distance between two images, we capture selected pixels from a reference image and calculate the Euclidean distance between this selection and the corresponding pixels from the current image. In order to generate the error surface we repeat the same process over the entire image by moving the patch taken from the reference over every possible point on the current image but bearing in mind that the reference patch edges shouldn’t cross the current image border. The distances calculated at each location can be used to represent the height of the “error surface” at each point. The distance between two patches is calculated as follows:

$$d(I_i, I_j) = \sqrt{\sum_{k=1}^{h} \sum_{l=1}^{w} (I_i(k,l) - I_j(k,l))^2},$$

where $h$ and $w$ are the patch’s height and width and $k$ represents the current pixel. $I_i(k,l)$ and $I_j(k,l)$ are the $l$th colour component of the $k$th pixel of images $I_i$ and $I_j$ respectively [5].

In order to stabilise the camera, we need to find the best match between the patches under consideration. This can be represented as finding the minimum on the error surface generated using this method.

B. What image features generate what error surface?

For this work we use the RGB colour space, although we are aware that other colour spaces may have different and possibly beneficial properties.

The shape and location of the patches used in this process can dramatically affect the outcome. For example a reference patch drawn horizontally or vertically across the image may match well with a number of similar regions in the image, which can cause the error surface to have local minima and/or an indistinct global minimum. For example, Fig. 1(a) shows an image consisting of a vertical line drawn in the middle. Using a Central patch (see Section IV) causes an indistinct global minimum that leads to ambiguity in finding the optimal camera location, Fig. 1(b).

![Error Surface Generation Techniques for Appearance-based Stabilisation of an Intelligent Kite Aerial Photography Platform (iKAPP)](image-url)

Heidar Hosseini, Mark Neal, Frédéric Labrosse
Department of Computer Science, Aberystwyth University, Ceredigion, Wales, UK, SY23 3DB
{hhh05, mjn, ffl}@aber.ac.uk
the large number of minimal points we encounter by targeting a central patch in the above image.

![Fig. 2: Error surface (b) from an image with a black box in the middle (a)](image)

C. Error surfaces in real world images.

The above comparisons were produced based on some artificial images but, is it the same in real images? The above comparisons of artificial images show clearly that the final outcome depends on the patches we target as well as the image that we process. In real images small patch movements may cause large changes in the image’s characteristics and may result in dramatic changes in error value and therefore to very steep error surfaces. The challenge is therefore to find a sampling strategy that minimises this tendency whilst maximising the likelihood of generating a well-defined and unambiguous global minimum. Thus, establishing what regions to target when dealing with real world images will be the key to generating well-behaved error surfaces. In order to test these ideas we have applied these techniques to some real world images. Repeating colour patterns, large regions of constant colour and complex textures are all features which need to be assessed and thus images containing these features have been selected for testing (see Fig. 4).

III. CONTROL ALGORITHM (“P” ALGORITHM)

In order to meaningfully assess the utility of the error surfaces we need a control system that can use them to find the minimum. There are a variety of control algorithms that can be used to control systems such as that presented here. PID (Proportional, Integral, and Derivative) [3] is one of the simplest and best understood and a slightly modified proportional controller will be used in this work. The positioning system itself is beyond the scope of this paper although the authors have designed, built and tested such a system which has sufficient speed, precision and resolution for this application. Our “P” algorithm enhances the controller movement by increasing the P gain iteratively when the control system fails to generate a movement. This is a simple low-overhead heuristic to avoid getting stuck on flat regions of the error surface. The P gain is reset to 1 after each movement. This algorithm is not intended to be used in a final control system, but is simply used to assess the error surfaces generated in this work.

IV. STRATEGIES FOR PATCH POSITIONING

What and where to target on the image is the current focus of our work. We have tested various techniques for patch positioning. These include fixed locations, regions containing a high concentration of edges, and clear and fuzzy artificial patches. These techniques are described in the following sections.

A. Fixed Locations (Central, Individuals and Merge)

This strategy has been tested using three different layout patterns: Central, Individuals and Merge. Each was used to identify the patches’ locations. Each technique has different numbers of patches, patch sizes and patch locations. The patches’ properties will help us categorise the effect on the shape of the error surfaces. Applying different techniques on the same image will result in different error surfaces because different strategies capture different views and perform the calculation process based on different pixel selections.

![Fig. 3: Locations of each strategy: (a) Central, (b) Individuals, (c) Merge](image)

Fig. 3 shows visually where each strategy is targeting and what pixel samples it includes when applying them to images.

![Fig. 4: Three different testing images: (a) Garden, (b) Door, (c) Flower](image)

Examining Fig. 3(a), the Central strategy, shows that it is obvious that when applying it on the Garden image (Fig. 4(a)) the grass will occupy most of the patch. By looking back at the artificial images we can see that large single colour regions such as the big black square can lead to a lot of similarity and have many close minimal points. The grass is thus a poor region to target as the Garden image includes a lot of grass and is likely to yield a wide ambiguous global minimum. The situation is somewhat similar when applying the Central strategy to the Door image but rather better when applied to the Flower image (Figs. 4(b) and 4(c)). The Central patch in the door image also contains little variation in colour and there is the possibility that the error surface will be very flat in this region. This strategy worked best on the flower image, primarily because of the variety of colours captured by the Central strategy in the Flower image. In Fig. 3(b), the
**Individuals** strategy, the central box is divided into four equal size parts and distributed around the diagonals. In the Garden image this distribution is beneficial because it leads to collecting more variation and results in a smoother error surface with a well-defined minimum. Edges are important in terms of these sorts of variation and by examining the regions of the images captured by the **Individuals** boxes (Fig. 3(b)), we can see that for these images we tend to gather more variety of colour than with the **Central** strategy. This variety helps in producing error surfaces which descend more reliably towards the global minimum.

Fig. 3(c) shows that the situation with the **Merge** strategy is quite similar to the **Individuals** strategy except that we decreased the four patches size and included an extra patch in the centre position. The extra patch we added in the centre still contains only grass for the Garden case, but the other patches sample other colours and offset this effect.

Fig. 5 shows a comparison of some representative regions of the error surfaces for all three strategies for all three images. Each figure contains three overlapping surfaces representing the **Merge**, **Individuals** and **Central** strategies.

Fig. 5(a) shows the error surfaces for all three strategies when applied to the Garden image. It shows that the **Individuals** and **Merge** strategies perform very similarly. They both have better surfaces than the **Central** one (which is the lowest, flattest surface) because of the larger slope they produced. This is because the **Central** patch mainly covers grass areas, which is homogeneous in colour and widespread. The situation differs from one image to another and the shape of the error surfaces differs according to the image characteristics. Fig. 4(b) shows an image of a door with a brick wall surrounding it. It has a repeating pattern in the brickwork and a contiguous region of colour in the door itself: two features that we expect to cause problems. Local minima are likely to be generated by the brickwork and flat regions by the contiguous colour regions. Fig. 5(b) shows the error surfaces when applying all three strategies to this image: all three plots having relatively well behaved overall shapes, but contain local minima and ripples. The **Central** plot is less smooth than the other two (and shows some ripples and local minima), but the **Merge** and **Individuals** error surfaces are very similar. The door has repeating brickwork patterns (causing the ripples) and targeting anywhere within the (contiguously coloured) door may reduce the slope. The **Merge** and the **Individuals** patches happen to be located mostly in the corners where we have some colour variety: moving the patches in these areas generates a better error surface.

Fig. 5(c) is a comparison between the three strategies for the Flower image. The graph shows all three plots smoothly heading down and having nice shapes. The **Central** plot is slightly more rippled than the other two.

**B. Edge Detection**

Edges (such as those in the brickwork in the door image) play an important role which can have a significant effect on the shape of the generated error surface. There are many ways of defining edges and we define them as places where there is a sudden variation in brightness because that variation might be helpful in identifying the minima on the error surface. There are numerous algorithms to identify edges and they vary significantly in terms of their performance, speed and accuracy. These techniques have been used for different purposes such as object tracking [2] and image comparison. There are also many applications to track moving objects such as vehicles which were based on these techniques [1]. Sobel [4], Moravec [6] and Robert [7] are examples of those widely used techniques which we have tested and applied over the images to see if any algorithm will help to detect patches suitable for generating error surfaces for minimisation and stabilisation.

For simplicity we converted the images to binary before applying the edge detection algorithm. This will identify the edges more accurately and produces less noise [8]. Fig. 6 shows the results of the process: the grey areas in Fig. 6(c) are the edges that are detected.

The next step is to identify the “best” patch and in our case we chose areas with the largest number of edges. The patch selected is shown in Fig. 7(a) and the resultant error surface in Fig. 7(b).
The minimum in the error surface in Fig 7(b) is very narrow where it’s heading down to the global minimum. This will cause problems when trying to find the global minimum from any large displacements in the camera’s angle of view.

C. Clear and Fuzzy

Edges are an important feature in generating the shape of error surfaces. Clear images usually have hard edges whereas fuzzy ones contain soft edges.

1) Artificial Images

Images with clear resolution will usually have sharp edges and we expect sharp variations on the surface as patches used for generating error surfaces pass through such image regions. In non fuzzy images we expect error surface slopes to be steeper than those images with smoother variation.

Fig. 8 shows a visual comparison between two error surfaces which were generated from clear and fuzzy images.

![Clear Circle](image1.png) ![Fuzzy Circle](image2.png)

**Fig. 8:** Two error surfaces after running the comparison process over the clear and fuzzy images

The visual comparison shows clearly the difference between the two and the shape of the area surrounding the global minimum. Both of these surfaces have well-defined global minima and the minima could be found effectively using simple control systems such as PID. The surface generated from the clear image is steeper than the smoother surface generated by the fuzzy image. Fuzzy images will have more gradual changes which makes them a good choice if we need a smooth error surface for the control system to work on.

2) Real World Images

Extensive experiments were undertaken to investigate the use of edge detection for patch selection, a typical example is shown in Fig. 9(a), which contains a box indicating the area with the largest number of edges.

![Area with highest number of edges](image3.png)

**Fig. 9:** Area with highest number of edges (a) applied over the fuzzy image (b)

As in the previous synthetic images case, we fuzzified the Garden image, Fig. 9(b). Fig. 10 shows a comparison between the two surfaces obtained with the two images using the patch indicated.

![Comparing hard edges](image4.png) ![Comparing smooth edges](image5.png)

**Fig. 10:** Comparing hard edges (b) with smooth edges (a) error surfaces

The smooth edges provide a wider minimum, as expected, although the surfaces are generally disappointingly flat.

D. Artificial Patches

A further experiment examined what artificial shapes produce the best error surface shapes. We use fuzzy blobs of different sizes and gradually changing colour from the centre towards the circle’s edge. The patch used captures the entire fuzzy blob and some white space surrounding it. The purpose is to investigate which of those blobs produce the best error surface for minimisation. We then use that artificial patch to iterate through the real world image to find the best match between real world features and the artificial shape. This best match patch is then used as the target in the hope that the error surface will be close to what was produced in the artificial examples (Fig. 8). Fig. 11(a) shows the shaded circle in the centre which will be used to find the best error surface shape that can be used for the comparison process. We tested sizes for the fuzzy blobs varying from 10 to 50 pixels in diameter but the patch we capture is from 30 to 150 pixels diameter. The main reason is to capture a patch which includes a dark blob in the middle with some white space surrounding it. Different blob sizes produce different error surfaces with different slopes (Fig. 11(b)).
The flattest surface corresponds to the smallest patch with a size of 30 x 30 pixels. The other surfaces were produced with blobs of sizes 60, 90, 120 and 150 pixels. Fig. 11(b) clearly shows that increasing the blob size improves the surface from the controller’s point of view. Such artificial patches can be used to find similar features in images such as the three test images. We tested it on all three images (Garden, Door and Flower) and experimented with different patch sizes to see what features were captured for the comparison process. The experiment was based on only one of the RGB colour channels (informal experiments showed no noticeable difference in performance between them). We arbitrarily chose the green channel for later experiments. The boxes indicated in Fig. 12(a-c) show the patches selected for each colour channel and after applying the comparison process between the artificial patch and the real image over the different channels. The surfaces generated are shown in Fig. 13(a-c). These results used the smallest patch size (30 x 30).

We have used an artificial patch size of 150 x 150 pixels to find the best patch possible.

We now apply the comparison with different artificial patch sizes to see how the error surface can differ when capturing a bigger patch and apply the process using a large patch size.

The difference between the patch sizes is clear and dramatic. For instance, if we compare the Door error surface (Fig 13(b)) generated by the 30 x 30 patch with the error surface generated by the 150 x 150 patch (Fig 15(b)), we can see how the large patch removed a lot of noise on the surface and made it much smoother and easier for the controller to navigate. A similar evaluation also applies for the other two error surfaces where the patch’s enlargement made a significant difference for both Garden and Flower images.

V. WHOLE IMAGE SAMPLING STRATEGIES

We cannot in general predict what image properties we will encounter and where to apply which strategy to capture the best patch possible. We therefore applied some “whole image strategies” which ensure that no matter what features and what colours the image includes, we still target the entire image and use the entire image as a patch. Using all the pixels requires a prohibitively large amount of processing time and therefore, we applied different ways of capturing limited number of pixels but distributed around the image. There are two ways used to distribute the pixels in the image, one is static positioning and the other is random positioning.

A. Static Positioning

This strategy uses most of the image space to capture a variety of pixels from the entire image. The static positioning method uses a fixed grid to select which pixels are targeted. There are many ways of targeting the pixels...
but we attempted to define a layout to ensure that on all translations some sampled pixels will overlap. Selecting based on a regular grid does not achieve this: if we sample every $n$ pixels we will not get any overlap if we displace the image by $(n-1)$ pixels.

We resolved this issue by defining a simple way to guarantee some overlap on every movement and therefore we made the selection process to take place every 10, then 9, then 8, etc., down to a spacing of 1. This process of selecting the pixels is repeated right across the image. Fig. 16 shows the selected pixels in the Static Positioning strategy. We will be using the above indicated locations as our patch for the comparison process.

![Fig. 16: Locations of the Static Positioning strategy](image1)

We initially ran the comparison process for each colour channel separately and the result was that the error surfaces for them all was quite similar and therefore we decided to use only a single colour channel for the comparison process (again we chose green for no particular reason). Fig. 17 shows error surfaces for the Garden, Door and Flower images using static positioning over the green channel only.

![Fig. 17: Error surfaces for different images after applying the Static Positioning strategy](image2)

It is clear that all three error surfaces have a smooth slope down towards their minimum and the noise and local minimal are almost completely eliminated (although local minima due to the brickwork texture are apparent in Fig. 17(b)), which is potentially valuable from the controller’s point of view. The next stage is to test how well the controller navigates over these surfaces. Running the control algorithm described above from all starting points on the error surface will give us the ultimate result which verifies from what parts of the surface we can reach the target and get ourselves back to the original location. Fig. 18 is produced from Garden’s green channel error surface. Fig. 18(a) shows a number of dots on the contour map which indicate the locations from which the minimum was successfully reached using the simple control algorithm. Fig. 18(b) shows the distribution of the number of steps required to reach the global minimum from all the successfully minimised starting locations.

![Fig. 18: Successful areas on the contour map for the Garden image](image3)

The statistics show that 77% of the positions on the image were successful. Fig. 19 shows the number and locations of the successful starting points on the error surface for the Flower image.

![Fig. 19: Successful areas on the contour map for the Flower image](image4)

This poor success for the Door image is due to the homogeneous areas and repeating patterns in the image.

**B. Random Positioning**

Random Positioning of pixels is another technique which we used to target the entire image rather than positioning a patch(s). Fig. 21 shows the uniform random distribution of locations used in this strategy.

![Fig. 21: Distributions of the pixel locations using the Random Positioning strategy](image5)
In this approach we also ran the comparison process over each of the colour’s channels separately to see if there was any difference amongst any of the produced surfaces. As in previous cases, we have chosen the green channel to base our experiments on as there was no noticeable difference.

Fig. 22 shows the error surfaces generated using the random positioning strategy based on green channel only. The error surfaces are quite similar to the surfaces generated by the Static Positioning strategy. This similarity comes from the way we generated the surfaces where in both cases we targeted the entire image to be the patch for the comparison process. Fig. 23(a) is a contour map for the Garden image with the indication of the successful points and the Fig. 23(b) is the graph showing how many points took how many steps in order to achieve the process.

In that case, using only the green channel, 59% of the starting locations succeeded using the Random Positioning strategy. Fig. 24(a) shows the contour map for the Flower image with the indicated points showing the successful area, which is 52% of the image locations.

Finally we apply the same technique to the Door image. Fig. 25(a) shows the successful area covering 24% of the image: a slight improvement over the Static Positioning.

VI. CRITIQUE

The purpose of choosing different strategies was to implement a solution for the best patch found within the image. The fixed locations strategies perform well only if the patches happen to capture some good features in order to build the error surface but this is unacceptably dependent on the properties of each individual image. The patch size is also a major consideration in the shapes of the error surfaces: larger patches tend to produce better error surfaces and the effects of “noise” decrease. The strategies using edge detection and artificial patches also performed unreliably in general.

There is a dramatic change in performance between patch positioning and whole image sampling strategies. For example Fig. 26(a) was generated by applying the 60x60 Artificial Patch to find the best positioning before applying the comparison process and the surface in Fig. 26(b) was generated by applying the Random Positioning strategy. The random strategy uses fewer pixels for the comparison process and the difference between the two is clear: random surface is smooth, evenly sloping and has a well-defined single global minimum whereas the artificial patch surface is noisy and has multiple local minima. Both static and random positioning strategies perform well and quite similarly by targeting pixels distributed over the entire image.

Both whole image sampling strategies are effective although in these experiments more pixels were sampled in the static sampling strategy. Despite the large number of pixels the error surface contains a number of regions from which our simple controller was unable to minimise. This may be partly due to the simplistic nature of the controller, but in general the randomly positioned pixels yield steeper and deeper global minima: compare Figs. 27(a) and (b).

In general the Random Positioning strategy does not suffer from regions in which our controller was unable to minimise and therefore seems to be a better solution.
The minimisation method used on the surfaces generated by this work was essentially a P controller moving the target location iteratively across the surface. The controller used information only from the local slope of the error surface which was assessed using three points to obtain the direction of the slope; this minimises the computational load by only requiring the sample pixels to be compared three times between each movement of the actuators. A summary of the numbers of pixels sampled to generate some error surfaces can be seen in Table 1.

In order to control the physical platform a more refined and sophisticated system is likely be required. In addition a controller that takes advantage of the nature of the kite platform’s typical motion will assist in improving performance. For example the platform is suspended on strings (using a “picavet” arrangement) beneath the kite string itself and thus the main mode of motion is as a pendulum with a simple harmonic motion. Thus the use of information about the platform’s trajectory in previous iterations could help in predicting the location of the minimum on the error surface. The picavet arrangement of strings also minimises rotation of the platform which (along with scaling) is ignored in this work. Rotation of the image due to rotation of the platform and scaling of the image due to changes in altitude are both factors which can be addressed in similar ways to the translations considered here although this will increase the computational load.

VII. CONCLUSION

The ultimate conclusion behind all the patch positioning strategies and the applied experiments is that wider distributions of pixels will capture more of the variation in real images and therefore will produce better error surfaces and less noise. In order to achieve this some form of whole image sampling is the best technique. This leads to successful navigation on the error surface generated by the entire image and not depending on a specific feature, shape, colour, etc. To design a successful visual stabilisation control system for our Intelligent Kite Aerial Photography Platform (iKAPP) system we need a strategy which reliably generates a good error surface where the controller will have the opportunity to get us back to our target from most positions in most images. In addition the processing load is important as the platform uses a fairly low specification: 1GHz processor with 512Mb of RAM. Fast processing will allow us to process images at a higher frequency and therefore move the actuators at a higher frequency which will improve the stabilisation performance. Every error surface has a certain region within which the controller can succeed and if the kite system encounters a lot of turbulence then, more images and faster processing will help to keep the camera within this region.

VIII. FUTURE WORK

We are currently working on refining the pixel sampling strategy with the main aim of ascertaining how few pixels will still produce a reliable error surface for minimisation. The next step will be the deployment of the complete control system on the platform under laboratory conditions where we will use VICON motion capture equipment to assess the motion of the platform and response rates of the complete system. Once this work is complete and we have determined whether it is possible to obtain performance sufficient for useful stabilisation for translations we will deploy the system in the field and move on to examine the problems of rotation and scaling due to changes in altitude of the platform under real world conditions.

REFERENCES

Induced Policy Segmentation in Multi-Agent Reinforcement Learning with Communicative Actions

Matthew B. Whitaker\textsuperscript{1} and Gillian Hayes\textsuperscript{2}

Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh, UK

Abstract—The non-Markovian nature of multi-agent systems poses a challenge to existing reinforcement learning algorithms based upon Markovian Decision Processes (MDPs). In this paper we examine the effects of extending the type of actions available to a grid-world agent, by adding a ‘communicative’ action to influence other agents in the environment. This action is designed to be an optimal action from any state, re-introducing the Markovian property into the environment. We examine the policies developed by a Sarsa(\(\lambda\)) learning agent when sharing the environment with an automated agent. We show that the policies learned unexpectedly segment into distinct sub-policies which use the communicative action to differing degrees. Finally, we show how this segmentation is linked to the degree of determinism found in any particular state, and comment on how this can be used to improve learning efficiency in multi-robot systems.

I. INTRODUCTION

Getting multiple agents, including robots, to learn to carry out a joint task can be difficult. The state space can potentially become extremely large; the state information of each agent must be available to all, giving state vectors that increase in length with the number of agents. If reduced state vectors are used, so that only incomplete information is available to each agent, then the world becomes non-Markovian and the use by each individual agent of a single-agent learning algorithm can result in non-optimal solutions.

The work in this paper looks at a multi-agent movement task. Specifically it investigates the use of a ‘shout’ action – the simplest way in which one agent can communicate with another – in addition to movement actions. The hope was that the addition of this simple form of communication would allow a learning agent using a single-agent reinforcement learning algorithm to solve a task in a multi-agent non-Markovian world.

This hope was satisfied to some extent. An unexpected outcome was that the communication action segmented the policies learned into sub-policies that used the communication action to differing degrees.

A. Single-agent Reinforcement Learning

Single-agent reinforcement learning is based on describing the environment using a Markov Decision Process (MDP) [1]. A finite MDP is defined by sets of states and actions, by the state transition probabilities associated with taking an action from a state, and by the expected amount of reward received for taking an action from a state. More formally we can say that a finite MDP is described by the tuple \(\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle\) with time represented by \(t\):

- the set of states that can occur, \(\mathcal{S}\),
- the set of actions available to the agent, \(\mathcal{A}(s)\) where \(s \in \mathcal{S}\),
- a set of transition probabilities, \(\mathcal{P}(s,a,s') = \text{Pr} \{s_{t+1} = s' | s_t = s, a_t = a\}\), for all \(s \in \mathcal{S}, a \in \mathcal{A}(s),\) and \(s' \in \mathcal{S}\),
- a set of expected immediate rewards, \(\mathcal{R}(s,a,s') = E\{r_{t+1} | a_t = a, s_t = s, s_{t+1} = s'\}\), for all \(s \in \mathcal{S}, a \in \mathcal{A}(s),\) and \(s' \in \mathcal{S}\).

When an agent commits action \(a \in \mathcal{A}(s)\) in state \(s \in \mathcal{S}\) the state changes to \(s' \in \mathcal{S}\) according to the state transition probability distribution \(\mathcal{P}(s,a,s')\). Which action the agent chooses will depend on the policy, \(\pi\), it is following. This policy is a mapping from each state \(s \in \mathcal{S}\) and action, \(a \in \mathcal{A}(s)\) to the probability \(\pi(s,a)\) of an agent taking action \(a\) when in state \(s\). The agent then receives a reward signal, \(r_{t+1}\) for this state-action combination calculated by reward function \(\mathcal{R}(s,a,s')\). The reward is determined by the current state and action taken, as well as by the state the action taken leads to. This resulting state may differ depending on the action chosen in the preceding state and the state transition probability distribution. As some resulting states will be better than others this will need to be reflected by differing rewards. Based on the rewards encountered when taking an action from a state we can begin to estimate the value of a state or, more specifically, a state-action pair \((s,a)\).

This can be seen as describing how desirable it is to be in a given state, or how much future reward we can expect if we take a particular action while in a state. The basic premise of an MDP is that state information at time \(t\) is all that is necessary to choose the optimal action.

A common scenario in single-agent reinforcement learning involves a learning agent, situated in a grid-world, developing an optimal policy for movement between a start location and a static goal location [3]. The agent is given an absolute position in the world to use as a state value, and receives negative reward for each action that fails to result in the goal state. While several algorithms have been developed that produce optimal policies for this class of problem [1][5][2], the precise information needed by these algorithms (for example the unique state value based on an exact location in the world) is rarely available in a more realistic task.

B. Multiple Agents

Another challenge, especially when considering an environment containing multiple agents, is that the world the

1. m.b.whitaker@sms.ed.ac.uk
2. gmh@inf.ed.ac.uk
agent is exploring is not guaranteed to be static, and thus may violate the Markovian property required for guaranteed convergence with most single-agent reinforcement learning algorithms [4][5].

The scenario described in this paper causes the Markovian nature of the world to be violated in three ways:

1) Firstly, the method by which an agent observes the state of the world lacks unique state values (known as ‘aliased’ of state). That is, for a given ‘observable state’ the agent learns to associate with actions and rewards, there may be many possible ‘global states’.

2) Secondly, there is a non-static goal location within the environment. The goal condition in the scenario is that the two agents should be on adjacent squares. This is based upon the relative position between two agents in the environments, and so alters as the agents move around the environment. It should be noted that the presence of ‘aliased’ observed states may actually make the attainment of this non-static goal ‘easier’. This is due to the fact that unique states are often defined by the spatial coordinates of the agent within the world, but in a non-static environment these can be misleading as the agent’s state information is ‘fixed’ in the world, whereas the goal location is not. Therefore actions committed in these unique states may exhibit non-deterministic results. With a state based upon observations of the environment local to the agent there is increased generalisation when associating particular actions with states.

3) The third violation of Markovian principles is strongly linked to the second, in that the world contains multiple agents whose internal states are not observable and whose actions alter that global state of the world. A given agent may see the state of the world change (e.g. due to the movement of another agent) even if it does nothing itself.

All three of these conditions make the scenario very difficult for a single-agent reinforcement learning algorithm based upon an MDP to solve.

A family of algorithms exist which attempt to address learning in a world exhibiting non-Markovian properties. These algorithms consider Partially Observable Markov Decision Properties (POMDPs). POMDPs describe worlds where the perceptions of the agent do not fully describe the world state. POMDPs are much more difficult to analyse than MDPs as observable states are no longer independent of preceding states. This means that the MDP guarantee of convergence to a solution cannot be taken for granted in a POMDP. The algorithms attempt to solve this problem in several ways. One method is to use adapted algorithms for learning hidden Markov models (HMM) to learn a model of the POMDP [7]. Another approach involves the addition of agent actions which increase state information to a degree that the problem becomes Markovian [6]. This method of re-introducing a Markovian element to a problem through agent action is the approach taken in this paper.

In order to address the non-Markovian problem one final major difference exists between the scenario in this paper, described in detail in the following section, and a standard learning agent gridworld scenario. In addition to the standard range of movement actions available to the learning agent, in our experiments there is also a communicative action designed to influence the behaviour of other agents in the world. The standard physical actions can be described as altering the ‘local’ state of the agent in the world, moving the agent one increment in one of the four possible directions (North, South, East and West). The communication action has no effect on the current agent’s position in the world. Instead this action affects the position of the other agent sharing the environment, causing it to move one square closer to the source of the communication action. This communication action has the effect of allowing the agent to have access to an action that is ‘correct’ from every possible state: it only needs to communicate and the other agent will come to it, thereby solving the task. This re-introduces a Markovian aspect to the problem as the agent can now choose an optimal action regardless of the three non-Markovian influences listed above.

The purpose of the work described in this paper is to ascertain whether the presence of this communicative action, used to influence another agent, will allow a learning agent using single-agent reinforcement learning to develop a successful strategy for achieving its goal in this non-Markovian world. This would have applications in multi-agent systems, and certainly in multi-robot systems, where the ability to utilise communication with other agents in order to achieve goals would appear to be a powerful and useful tool.

The rest of the paper is structured as follows. Section 2 describes the mechanics of the world and the abilities of the agents it contains. Section 3 outlines the experimental set-up and data gathering necessary for assessment of agent performance. Section 4 gives an overview of the results obtained. Section 5 provides an analysis of the results and discusses their implications. Finally, Section 6 summarises the findings and suggests possible directions for further work.

II. Environment

The environments considered in this work are all basic ‘empty’ grid-worlds of varying dimensions. Each individual square on the grid can contain either an agent, an obstacle, or nothing. In an ‘empty’ grid-world obstacles are found only in the squares that form the boundary of the world. This can be considered to be a simulation of a common robotics arena, with robotic agents and obstacles placed within a bounded environment. Agents can be placed on any empty square within the environment. Each agent is either a learning agent or an automated ‘homing’ agent. The simple physics of the environment only implements collisions between agents and obstacles, but not between agents and agents. This is reasonable as the goal condition prevents this situation from arising.
A. Learning Agents

Learning agents are able to execute five different actions in the world. Four of these actions are physical movement actions based upon the compass bearings North, East, South and West. Figure 1 illustrates the relationship between these bearings and the agent in the grid-world. The last remaining action is a communication action named Shout. A Shout committed at time-step $n$ can be perceived by other agents in time-step $n+1$, but an individual Shout action does not have any effect beyond this time-step.

At any time-step $n$ a learning agent can retrieve a state vector composed of observations from the immediate environment. These observations can be made at a varying ‘visual depth’ from the position of the agent, and display the contents of the squares within the visual field defined by the depth. For example, an agent with a visual depth value equal to one will be able to observe the 8 squares immediately adjacent to its location. With a visual depth value of two the state vector will consist of observations from the 8 immediately adjacent squares, plus the 16 squares that are adjacent to the initial 8. This is illustrated in Figure 2. Note that in this particular environment the learning agents (as opposed to the automated ‘homing’ agents; see Section II-B) cannot ‘hear’ the effects of a Shout action, despite the fact that they can commit the action, and thus these effects are not included in the learning agent’s state vector.

The learning algorithm employed by the learning agents is Sarsa($\lambda$) [2]:

Algorithm 1 Sarsa($\lambda$)

```
1: Initialise $Q(s, a) \leftarrow 0, \forall s \in S, a \in \{A(s)\}$
2: For each episode:
3: loop
4:    Reset $e(s, a) = 0, \forall s \in S, a \in \{A(s)\}$
5:    Initialise $s, a$
6:    For each step of episode:
7:        loop
8:        take action $a$, observe $r, s'$
9:        Choose $a'$ from $s'$ using policy derived from $Q$
10:            (e.g., $\epsilon$-greedy)
11:        $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
12:        $e(s, a) \leftarrow 1$
13:    For all $(s, a)$:
14:        loop
15:        $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
16:        $e(s, a) \leftarrow \gamma \lambda e(s, a)$
17:    end loop
18:    end loop
19: end loop
```

B. Automated ‘Homing’ Agents

The other type of agent the environment can contain is the aforementioned ‘homing’ agent. This agent does not learn or adapt, relying instead on a pre-programmed behaviour that uses a simple state vector, containing the relative direction of a perceived Shout, to move towards the Shout source. Two types of ‘homing’ agent are examined, the first will cause differences in the world state). In this case the simplest fact that they can commit the action, and thus these effects policy is for the learning agent to commit a Shout action from on this policy. The learning agent has to learn a policy that already converged, and is exhibiting ‘greedy’ action selection viewed in this case as a learning agent with a policy that has as it is entirely possible this agent could also be searching for the goal condition and would not have the option to take no action. This second behaviour is also slightly more challenging for the learning agent to adapt to as it adds an additional element of non-determinism to the environment.

C. Task

A learning agent and a ‘homing’ agent are placed randomly in the environment. The ‘homing’ agent could be viewed in this case as a learning agent with a policy that has already converged, and is exhibiting ‘greedy’ action selection on this policy. The learning agent has to learn a policy that will bring about the ‘proximity’ goal. The transition function in this environment is entirely deterministic, therefore an agent can rely on consistency between the action committed and the resulting state (at least on the observable local scale - on a global scale the other agent in the environment can cause differences in the world state). In this case the simplest policy is for the learning agent to commit a Shout action from every state, as this action is guaranteed to reduce the distance between the two agents and bring about the goal condition.

For the goal condition to be satisfied the agents must be in ‘proximity’ to each other, i.e. in adjacent squares in the grid-
world. All actions leading to a non-goal state will receive a reward of $-1$, with the reward for achieving the goal set at $0$.

This task reflects the observation that in many multi-robot scenarios it is desirable for the robotic agents to be able to locate each other within the environment and achieve some degree of spatial proximity. This may then allow robotic agents to cooperate in a lifting or pushing task, exchange information over a limited range, or affect the environment in some other way that a single robotic agent would be unable to achieve.

**D. Expectations**

Due to the automated actions of the non-learning 'homing' agent in the world the obvious best-policy for the learning agent is to consistently use the Shout action. Using this policy the learning agent will be guaranteed to get the distance between itself and the 'homing' agent reduced with each action: by shouting (and remaining stationary) it will 'attract' the automated 'homing' agent to it. It thus achieves the goal state in the minimal possible number of steps, and thus with minimal cost. The expectation, therefore, is that the policy developed by the learning agent should be dominated by the Shout action when assessed with no probability of exploratory actions.

**III. METHODOLOGY**

An experimental run consists of two alternating phases - the learning phase and the evaluation phase.

**A. Learning Phase**

In the learning phase the learning agent selects an action from the current state every time-step by using $\epsilon$-greedy action selection with an $\epsilon$ value equal to $0.1$. All actions committed that fail to result in the goal condition receive a reward of $-1$, with a reward of $0$ for an action that results in the goal being attained. When the goal condition is satisfied the 'episode' ends and both agents are randomly repositioned in the environment before the next learning step. Based upon the reward received for the committed action, and the resulting observed state, the Sarsa($\lambda$) algorithm is used to update the policy Q-table for the agent. The learning algorithm in this case has a $\lambda$ value of $0.9$, an $\alpha$ value of $0.1$, and a $\gamma$ value of $0.9$. Additionally, the eligibility trace values used by the algorithm are replaced rather than accumulated and there is no truncation. The agent is able to learn and develop its policy for the maximum number of learning steps specified in the algorithm configuration, with learning suspended every arbitrary number of steps to allow the current policy to be evaluated in the evaluation phase. The default visual depth for the learning agent is $2$. A visual depth value of $1$ was initially investigated but was found to be insufficient to allow the agents to learn effective policies. This is because the goal condition is satisfied at the same time as the 'homing' agent enters the observation range of the learning agent. As a result the learning agent is never able to 'see' the other agent and so the goal appears to be reached randomly, with no one strategy giving better results than another.

**B. Evaluation Phase**

The evaluation phase occurs periodically during an experiment, interrupting the learning phase every time a certain number of learning steps have passed (in these experiments this number was set at 1000). No learning or exploration takes place during the evaluation phase, the agent selects all actions based upon 'greedy' action selection over the current learned policy. In each evaluation phase both agents are placed in all possible pairs of locations within the boundaries of the environment, with the exception of initial positions where both agents would be sharing a square.

![Fig. 3. Frequency of Manhattan Distances in an 11x4 grid-world.](image)

![Fig. 4. Frequency of Manhattan Distances in a 20x20 grid-world.](image)
would be hoped agents can learn successful policies in a reasonable time-span (the distances are much greater in the 20x20 grid-world, and so if this world was used it might be wise to further expand the visual field of the learning agents).

From each starting position the learning agent was allowed to commit $2^n$ actions (where $n$ is the maximum MD possible between two squares in the world) under the policy learned up to that step. The order in which starting positions were assessed was based upon the MD between the two points. Figure 5 shows the sum of the number of Manhattan Distances that apply to each square in the examined grid-world. The squares that are covered by every MD value are found in the corners, as only these squares will be included in the set of points for the highest possible MD value. The squares in the centre of the grid-world are only covered by the lower MD values (roughly up to half the maximum MD possible). This also provides an indication of how frequently agents will encounter various perceptual states during the evaluation phase.

**Fig. 5.** Sum of Manhattan Distances per square in the 11x4 grid-world.

### C. Experiments

Each experiment was repeated 20 times and the results averaged across all runs. The number of learning steps in each experiment was 100,000, with the learned policy being evaluated every 1000 steps. The experiments were repeated for both types of ‘homing’ agent (‘limited’: no action when no Shout present vs. ‘full’: random action when no Shout present).

**IV. Results**

Figures 6 and 7 show the quantity of Shout actions executed by the agent as a percentage of the total number of actions committed throughout the evaluation phase. For the learning agent sharing the environment with a limited action ‘homing’ agent (which moves towards the source of a Shout if one is detected, but is otherwise stationary) we can see that although the Shout action quickly dominates the set of actions committed, it reaches a plateau at around 67% of the actions the agent chooses. In the case of the full action ‘homing’ agent this plateau is slightly higher, but still fails to exceed 70%. This is much higher than would be expected if the agent was choosing the Shout action with the same probability as the other available actions (there are five actions in total, four of movement and one of communication). However, it is lower than the proportion of Shout actions that would be expected from the naive ‘Always Shout’ optimal policy (even when taking exploration into account). This unexpected ‘upper limit’ on the amount of Shouting an agent uses needs to be explained. We will attempt to account for this in the following section.

**V. Analysis**

From the results obtained it is clear that the learning agents develop a preference for the Shout action over the other actions - but the Shout action never totally dominates the policy even when no exploration is present. This is an interesting result as it raises the question of why the agents learn to prefer the Shout mostly, but not totally (which would lead to optimal performance).

To investigate this further the polices developed were examined in detail, observing in which states a Shout action was preferred, and in which states another action was chosen. The policy developed by the learning agent appears to be segmented into two sub-policies depending on whether the other agent is within observation range or not. The naive optimal policy of choosing Shout in every state is indeed the...
dominant behaviour in states where the ‘homing’ agent is beyond observation range. However, once the ‘homing’ agent is observable by the learning agent there is a policy shift that favours the movement actions. The policy is effectively ‘segmented’ depending on the presence of the ‘homing’ agent in the state vector.

The degree of segmentation differs depending on the type of ‘homing’ agent present in the environment. Figure 8 shows the percentage of states in which a Shout is the preferred action when the state-space is split into two parts - states in which the learning agent can observe the ‘homing’ agent, and states in which it cannot. The learning agent rarely visits all possible states in the course of one trial, and so only states that had been visited were included in these sub-sets. Out of the total state-space there are only 20 possible observation states that do not include the ‘homing’ agent, and a typical run would see the learning agent experiencing approximately 80 states in which the ‘homing’ agent was visible. Clearly there is a large bias towards choosing a Shout action in those states where the ‘homing’ agent is not visible, with movement actions being preferred in other states.

Based on this we can form a hypothesis from the distribution which says that learning agents are much more likely to commit a Shout action in states where they cannot see the ‘homing’ agent. To assess the statistical significance of this result, and find the probability that the distribution occurred by random chance, we can apply the Chi-Square test to the data, which results in a Chi-Square value of 8.1402. The probability of this value occurring by chance is 0.0043, and thus the result can be said to be significant. This means that this distribution of actions across the policy space is extremely unlikely to have happened by chance, and so we can conclude that there is a distinct segmentation of policy.

Figure 9 shows the same evaluation carried out on the experiments where the learning agent is paired with the full action ‘homing’ agent (an agent that moves randomly when a Shout is not detected). The segmentation here is less clear, with the agent still preferring the Shout action in states where it cannot see the other agent, but also exhibiting a bias towards shouting when the other agent is observed. Applying the Chi-Square test to this data gives a value of 0.9971, which equates to a probability of 0.318. This probability is not statistically significant and so does not indicate a clear segmentation of policy.

Upon consideration it seems that the reason a policy developed by an agent becomes ‘segmented’ is due to the determinism of the observable state at any one time-step. If a learning agent cannot see the ‘homing’ agent in the immediate vicinity then the current state is non-deterministic from the agent’s perspective, as any action chosen is not guaranteed to cause a consistent state change. Even when the agent finds itself within sight of the environment boundary there is no increase in determinism, despite there being more ‘information’ in the state vector. This is because the position of the other agent varies, and so environmental cues are of limited value. The Shout action becomes favoured in this scenario because it is the only action that will consistently bring the agent closer to a goal state, even though the effects of the action may be beyond the visual depth of the observable state. The correct movement action would have the same effect as a Shout action, but at any one time there is at most a 50% chance of choosing the right movement action (when the ‘homing’ agent is equi-distant from the learner in both of the world dimensions), and more frequently only a 25% chance of a movement action reducing the distance to the goal.

The above results show, however, that the agent does not Shout all the time when it cannot observe the ‘homing’ agent. The most likely reasons for this are due to the agent being positioned in a certain way in the environment which ‘hints’ at ‘good’ actions. If an agent finds itself in a corner in the world, then the range of ‘good’ actions is narrowed to the two movement actions that move out of the corner, and the Shout action. Other actions will cause the agent to move into the wall, thus the agent’s position will not change and it will most likely not be any closer to the goal.

The situation is different when the learning agent can observe the relative position of the ‘homing’ agent. In the case of the limited action ‘homing’ agent, solving this sub-problem becomes much more deterministic as the agent can observe all of the ‘relevant’ global information in its local state vector. In this case the agent can learn an optimal movement action which would perform just as well as the
Shout action. The strong bias away from the Shout action in this scenario may be explained by the ratio of movement to communication actions in the action set, and the effect this has on exploration.

The change in policy caused by the presence of a full action ‘homing’ agent is logical, as the determinism of the states where the ‘homing’ agent can be observed is decreased by the random movements of the ‘homing’ agent when a Shout is not perceived. In this case, the correct movement actions become ‘worse’ than Shouting, as even if the learning agent moves in the correct way, the random agent may also move in a way that delays attainment of the goal state.

VI. SUMMARY

In this paper we have shown that the addition of a ‘communication’ action, an action with the ability to directly influence other agents in a multi-agent system, can allow a learning agent to develop a successful policy using only a single-agent reinforcement learning algorithm. As an unintended consequence of this work we have also shown that the policy developed may consist of several sub-policies influenced by the state vector available to the learning agent.

This result could be of use when using learning agents in multi-agent systems, as the identification of ‘sub-problems’ could allow the ‘segmentation’ of the agents’ policies to be anticipated. The separate segments could then be learned individually by shaping the environment the learning agent is situated in, before recombining the policy segments into a coherent policy for the overall task.

Additionally, the results show that the most obvious way to use a communication action is not necessarily the method that will be developed by a learning agent. Despite the naive optimal policy of this particular multi-agent system being one where the learning agent always uses the Shout action, we see that other policies develop due to unexpected influences from the states the agent encounters. Building upon the results shown in this paper may lead to greater understanding of how communication actions will be used by agents to solve a problem.

Further work in this area could initially involve investigating the effects of different ratios of physical and communication acts available to an agent. Additional experiments in larger worlds and with differing visual depth values would examine the relationship between the proportion of the world state observable, the proportion outside observation range, and the resulting segmentation of the policies learned by the agents. Similarly, placing the agents in a ‘boundless’ environment without walls would highlight the effects on action selection of ‘obstacles’ in the agent’s observation state.

Finally, from a more general and theoretical viewpoint, this work underlines the need to understand a task in terms of the joint state-space of the agents, rather than looking only at the grid-world state-space. In an extended system with two learning agents, both with visual field depths (perhaps differing in value), the challenge posed by the task needs to be understood in terms of the observations and actions of both agents in relation to the world dimensions.

VII. ACKNOWLEDGEMENT

Matthew Whitaker receives funding as part of the Interdisciplinary Research Collaboration in Dependability (DIRC) project supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Grant GR/N13999/01.

REFERENCES

Simplification of Artificial Neural Networks using Neural Complexity Measures

Thomas D. Jorgensen, Barry Haynes and Charlotte Norlund

Abstract—This paper investigates different aspects of simplification and methodologies for simplifying artificial neural networks. Simplification describes the process of decreasing an information-theoretic measure of the neural complexity of an artificial neural network. A number of novel methodologies for simplifying a network are proposed and tested in this paper with varying success. The neural network created using these methodologies are tested as controller for a robot solving a complex navigation task. The methods proposed here provides a methodology for matching network complexity to the complexity of the task or problem it is meant to solve. These methodologies provides a step towards truly unconstrained open-ended artificial evolution of neural network controllers for robot control.

I. INTRODUCTION

Artificial Neural Networks (ANNs) has been used in many different applications for years, with varying success. The most common applications of artificial neural networks in both evolutionary robotics and in common AI systems utilize a fixed network structure, in which the connection weights are trained [1]. The success of a neural network, in a given application, depends on a series of different factors, to mention a few: ANN topology, learning algorithm and learning epochs. Furthermore all of these factors can be dependent or independent of each other. Network topology is the focus of this research, as finding the network topology be a tedious and difficult process. Ideally all network topologies should be able to learn every given task to competency, but reality is that a given topology can be a bottleneck and constraint on a system. Selecting the wrong topology can potentially mean it will not learn the task at hand [1]-[3]. It is commonly known that a too small or too large network do not generalise well, i.e. learn a given task to an adequate level. This is due to either too few or too many parameters used to represent a proper and adequate mapping between inputs and outputs.

There are generally 4 ways to find the topology of an ANN [1], [5]-[6]. (1) Trial and Error, is the simplest way to find a topology. Find some topology at random and test it, if the network performs in an acceptable way, the network topology is suitable. If the network does not perform satisfactory, select another topology and try it. (2) Expert selection, means the network designer decides the topology based on a calculation or experience [1]. (3) Evolving Connections and Topology through Complexification, here extra connections and neurons added as the evolutionary process proceeds [6]-[9]. (4) Simplifying and pruning overly large neural networks, by removing redundant elements [10], [11]. The 4 methods mentioned can be subdivided into different techniques and approaches, but they all support the fact that continually growing or pruning network topologies yields the most unconstrained and open-ended evolution [12].

This paper investigates different methods for changing the topology of neural networks. 4 different methodologies for changing the topology of a neural network controller for robot control are proposed in this paper. The proposed methodologies encompasses the use of pruning, structural additions or reorganisation of existing network topology. The approach used in this paper is to investigate the 4 different methodologies separately and test them against a benchmark network and each other in a simulation of robot solving a navigation task. This research uses a measure of the information-theoretic complexity of a neural network to investigate the relationship between network complexity and the fitness and performance of a network. Decreasing this measure through the structural changes should yield a change in the fitness, as some research indicates a link between the neural complexity and the fitness of a given network [9], [13]-[14]. The hypothesis is that complex networks, not necessarily large networks, yields a higher probability of achieving complex behaviour. This paper investigates if this is true by decreasing the complexity rather than increasing it.

The first methodology adds connections to a benchmark network, these connections are placed so they reduce the complexity of the network. The second method reorganises networks, so no connections are added or removed, but only moved around in the network. Again the complexity is reduced by these reorganisations. Finally two different pruning methods are tested. One that maximises the complexity loss and one that minimises it. These methodologies select the connection out of all possible that has respectively either the greatest impact on the complexity or the smallest.

In all methodologies, the neural complexity measure is used to make sure that the complexity of the network
decreases. Changing the topology of a network whilst decreasing the neural complexity is interesting for several reasons. Firstly it provides a methodology to make the network complexity match the complexity of the task or problem at hand. This should yield a step towards unconstrained open-ended artificial evolution, as a network and a robot can adapt to changes in the environment or the task at hand. Secondly these tests should help increase the understanding of the circular relationship between a neural network controller, its performance and the environment it is working in. Furthermore this research should give some indications to the importance of the neural complexity of a network, which should prevent future methodologies from applying random or semi-random structural changes to a neural network, when evolving both connection weights and topology. Applying this or other information-theoretic complexity measures should yield methods and algorithms that apply some sort of intelligent component selection, which is what happen in natural systems as described by (Dawkins, 1996).

II. BACKGROUND

There are many different ways of changing the structure of a neural network either by growing or pruning them. Some of these methods focus entirely on the number of connections and neurons in a network, whereas other focus on how the network is connected. The later methods apply some sort of complexity measure, be it the magnitude of the weights or the correlation between elements as in this paper. Applying this measure yields terms like complexification and simplification, as focus is on changing this complexity and not directly on adding or removing element from the network. The terms cover the process of increasing and decreasing the complexity of a network.

Mostly complexification is related to the addition of connections or neurons to a network and simplification to the pruning of neural network. This paper makes a clear distinction between the changes in complexity of a network and the changes in topology.

Complexification research started with [6] through [15] to most prominently the NEAT framework [8]. The NEAT framework cross breeds neural networks of different topology. In their NEAT model they introduce mechanisms to evolve network structure, either by adding neurons or connections, in parallel with the normal evolution of weights. Furthermore they can cross breed different controllers using a gene tracking methodology. The NEAT framework does not encompass the use of any complexity measure unlike research in [14] proposes the use neural complexity measures for the complexification of neural network with some success, which is reconfirmed in [9],[13].

Research on simplification is limited to different strategies for pruning networks. Most successful are information-theoretic pruning strategies like Optimal Brain Damage proposed in [10] and Optimal Brain Surgeon proposed in [11]. Both of these use Hessian matrices to measure the complexity of the different network. Older pruning strategies include the use of penalty terms in the cost-function or genetic algorithms as a search tool for finding minimal network structures. The most used pruning method, and surprisingly effective, is Magnitude Based Pruning, which prunes the connection with the lowest connection weight. Reference [12] proves the success of simplification in artificial evolution, especially when combined with continuous complexification. This despite the fact they do not use complexity measures, but merely an extension of the NEAT framework.

III. SIMPLIFICATION METHODOLOGIES

4 methodologies for simplifying neural network are introduced here. One simplification methodology where structure is added to the network, one where networks are reorganised and finally two where simplification is caused by pruning. Only one methodology is tested at a time, so multiple prunings, multiple additions or multiple reorganisations can take place and not for example a pruning followed by an addition or reorganisation. All networks will be derived from the same benchmark network, which will act as a comparison for all other networks. The networks will be tested thoroughly in the same navigation task. When a simplification takes place either the benchmark network or an already simplified network will be subject to alterations. After the simplification the network will be retrained to regain some or all of its previous fitness.

Simplification is in this paper used as it is defined as:

**Definition of a Neural Simplification:**

Simplification is the process of reducing the neural complexity of an artificial neural network.

This definition of simplification is in line with the definition of complexity and complexification as defined in [14]. In order to be able to make a neural simplification of a neural network one needs a measure of the neural complexity, this is given in the following.

A. The Neural Complexity Measure

The neural complexity measure is an information-theoretic measure of the complexity of the neural network and not a measure of the magnitude of weights or of the number of elements in the network [14]. The neural complexity is a measure of the correlation between neurons and connections, it measures the integration and the specialisation of neural groups in a neural network.
Complex systems are characterised by the fact, that they have highly specialised clusters, which are highly integrated with each other. Systems that have very highly independent functional components or have very highly integrated clusters will have a low complexity. $X$ is a neural system with $n$ neurons, represented by a connection matrix. The entropy $H(X)$ is used to calculate the integration between components. The integration between individual neurons can be expressed by:

$$I(X) = \sum_{i=1}^{n} H(x_i) - H(X) \tag{1}$$

The integration $I(X)$ of segregated neural elements equals the difference between the sum of entropies of all of the individual components $x_i$ of the neural network and the entropy of the network as a whole. In order to be able to give an estimate of the neural complexity of a neural network, not only the integration between the individual elements is needed, but also the integration of any neural clusters in the network. It is very likely that neurons in an artificial neural network cluster together form some sort of functional cluster. The average integration between functionally segregated neural groups with $k$ (out of $n$) elements is expressed with $<I(X)>$. $j$ is an index indicating that all possible combinations of subsets with $k$ components are used. The average integration for all subsets with $k$ components is used to calculate the neural complexity:

$$C_N(X) = \sum_{k=1}^{n} \left( \binom{n}{k} \cdot I(X) - \langle I(X^k) \rangle \right) \tag{2}$$

The neural complexity $C_N$ of a neural system $X$ is the sum of differences between the values of the average integration $<I(X)>$ expected from a linear increase for increasing subset size $k$ and the actual discrete values observed. This neural complexity measure yields an estimate of the information-theoretic complexity of a neural network by measuring the integration between individual components and possible combinations of subsets.

**B. Simplification by Structural Elaboration**

The first simplification methodology adds connections to the network. The purpose of this methodology is to structurally elaborate a neural network, whilst trying to decrease the neural complexity. Furthermore, the goal of this experiment is to test if it is possible to decrease the fitness of a neural network by adding connections to it. This method is counter-intuitive as one expects structural elaborations to yield the same or better levels of fitness. This is however not entirely true, as some animals and the Neanderthal man have larger brains than humans, but are capable of less complex behaviour than us humans as described in [16].

This methodology adds connections to the network one by one and test the resulting network. Connections are placed at random, as long as they decrease the neural complexity of the network. If a newly placed connection does not decrease the complexity it is removed again and placed elsewhere.

**C. Simplification by Reorganisation**

The second methodology does not remove or add any connections is merely reorganises them. A reorganisation consists of one connection being removed and a connection added elsewhere. Again the only criteria for the reorganisation is, that it reduces the neural complexity of the network, if it doesn't it is undone and another reorganisation is tried. In subsequent reorganisations, connections that already have been reorganised are left untouched. This methodology is occurring in natural system, as occasionally some synapses detaches themselves from one neuron and reattaches to another. The effect of this in a brain like ours is infinitesimal due rarity of this happening and the vast number of neurons and connections in our neural networks as described by [17].

**D. Simplification by Pruning Elements**

The last type of simplification happens by pruning. Two methodologies are proposed here, which are fundamentally different. One method tries to minimise the complexity loss caused by a pruning, whereas the other tries to maximise it. Both methods prune one connection at a time. To use this algorithm, all possible ways to prune one connection has to be considered, and the one that cause the greatest loss or smallest loss is selected for pruning. One method tries to maximise the complexity loss, this method is called destructive pruning, as is should decrease the neural complexity and the fitness of the network. The second methodology tries to minimise the complexity loss, this method is called normal pruning. It should be used to prune away redundant elements of the network, without the fitness being reduced significantly. This can be useful in systems with large networks with plenty of redundant elements, as it can reduce the computation time and the learning time. These two methods are compared to the most commonly used pruning strategy Magnitude Based Pruning. It prunes away the connection with the lowest connection weight. If any neurons should happen to lose all of its input or output connections during the tests, the neuron will be removed. Pruning is a naturally occurring phenomenon, which happens constantly in any brain. Unused connections and neurons waste away, whilst other are strengthened as by [17], [18].
IV. EXPERIMENTAL SETUP AND ENVIRONMENT

The different simplification methodologies will be tested in a robot navigation task. The evolved networks are used as controllers for the robot. The first test is with the benchmark network, hereafter the network is simplified using one of the methodologies and tested again. This is done for all networks.

A. The Simulated Track and Robot

The controllers evolved here are tested in a simulated environment with a robot. In this environment a robot has to drive around a track, which consists of 32 sections. The objective of this task is to complete 3 laps in the shortest amount of time. If a robot fails to complete 3 laps, the distance covered is the measure of its performance. The robot has to drive around the track covering all of the sections of the track, it is not allowed to skip any sections. In total the robot has to complete 3 laps, with 32 sections in each, all in the right order. If the robot is to slow at driving between two sections the simulation is terminated. The following Fig. 1, illustrates the task to be completed:

![Fig. 1. The figure illustrates the track and the robot in the simulator.](image)

Fig. 1 illustrates the track and the robot driving around it. The robot is not limited in its movement, i.e. it can drive of the track, reverse around the track or adapt any driving patterns desired, as long at its drives over the right counter clockwise sequence of sections. The following Fig. 2 illustrates how the robot perceives the track and its environment.

![Fig. 2. The figure illustrates the robot and its sensors.](image)

This benchmark network will after a training session be simplified in the different ways. Connections will either be added, pruned or reorganised based on this network.

B. The Fitness Function

Fitness is rewarded according to normal motorsport rules and practice. 3 laps of the track have to be completed and the controller that finishes in the fastest time wins the race, i.e. it is the most fit controller. If a controller fails to finish 3 laps, the controller with the most laps or longest distance traveled wins. In the case that two controllers have reached the same distance the racing time determines the most fit controller. The fitness function can in general terms be...
described by the following, equation 3:

\[ \text{Fitness} = \frac{\text{Distance Covered}}{\text{Time}}. \]  

(3)

The fitness of a given controller can be calculated using this equation. The equation states that the longest distance covered in the shortest amount of time yields the best fitness. Time is the time it takes to complete the track. If a controller fails to finish this time is set to 480 seconds, which is the absolute slowest a controller is allowed to be, before a simulation is stopped. In the likely event that two controllers have covered the same distance, the controller with the fastest time will be favored for further evolving. The precise version of the fitness function can be seen in the following, equation 4:

\[ \text{Fitness} = \frac{\text{Sections} + (\text{Laps} \times \text{Track Length})}{\text{Time}}. \]  

(4)

The fitness is equal to the distance divided by the time. The distance is equal to the number of track sections covered in the current lap, plus the number of sections covered in previous laps. Track length is the total number of sections, which is 32. The minimum fitness obtainable is 1/480 \approx 0.0021 and there is no limit on the maximum, but 50 seconds seems to be the best expectable result, which is equal 65/50 \approx 1.3. All results are multiplied by 100 for convenience.

V. EXPERIMENTS AND RESULTS

A total of 6 different experiments were conducted and the results can be seen in table 1. In the table only results from the first, the third and the fifth topology changes are displayed for convenience sake. The first experiment was with a benchmark network, which was trained for comparison. All networks, regardless of simplification method originate from this benchmark network. The second test was the simplification through structural elaboration methodology. The third test was simplification by reorganisation. The fourth test was simplification whilst keeping the neural complexity high called N-Pruning. The fifth test was pruning based on the magnitude of the connection weights MBP. The final test was pruning whilst trying to reduce the neural complexity called D-Pruning. The benchmark network has been trained for 1000 generations and new network will subsequently be trained 500 generations. The simulation environment and the results are described in the following.

A. The Simulation Environment

The evolved neural network controllers are tested in a physics simulator to mimic a real world robot. The genetic algorithm has in all tests a population size of 50 and the number of tests per method is 15. Uniformly distributed noise has been added on the input and output values to simulate sensor drift, actuator response, wheel skid and other real world error parameters. The simulated robot can be seen in the following snapshot from the simulator:

![Fig. 4. The figure illustrates the robot and the track in the simulator.](image)

Fig. 4 shows a snapshot from the simulations. The car, the wheeled box in the middle, is driving along the track, which is the rectangles of alternating colour. Fig. 4 is similar to Fig. 1 and it gives an idea of how the artificial neural network controllers simulated robot driving around a virtual track. To give the simulation some of the same effects as in on real racing track, the track has been given edges, which can be seen in Fig. 4. Whenever the robot drives off the track it goes down this edge onto another slower surface. This means, that if the robot cuts corners, it could potentially have wheels lifting off the ground, due to the edge coming back onto the track.

B. Results from Structural Elaborations

The results obtained indicate that it is possible to add connections to a network and thereby reduce the neural complexity as well as the fitness of the network. Reducing the network complexity by adding connections also reduces the probability of finding an adequate solution given the here available time frame. If the network had much longer learning times it would probably learn the task at hand even better than the benchmark network. Having extra connections means having more parameters and this inevitably means longer learning time. The fitness loss is visible from the table, but it is not statistically significant given a t-test with a 5% significance level, but it is very close being significant.
C. Results from Reorganisations

The reorganisation tests indicate that it is possible to reorganise networks, i.e. the removal and reinsertion of a connection, and reduce the neural complexity in the process. This reduction of complexity yields a network that has on average a longer learning time and a lower probability of learning a task adequately. The results are statistically significant after 4-5 prunings, tested with a t-test with a 5% significance level.

D. Results from Pruning

The three different pruning strategies tested in this paper yield completely different results, they do however confirm a general trend.

N-Pruning yields after 5 prunings no statistical difference in fitness levels compare to the benchmark network despite a small fitness loss. D-Pruning however yields a significant loss of fitness after 3 prunings. MBP is much more complex to describe, as every test varies. Sometime the algorithm selects the same connection for pruning as the two other algorithms, but mostly it doesn't. On average MBP is a good all-round algorithm, sometimes is chooses the correct connection to prune other times it doesn't. There is after 5 prunings no significant fitness loss compared to the benchmark network. N-Pruning is statistically better than MBP and both are statistically better than D-Pruning. This was the conclusion after a 5% t-test. These results might change if the retraining time after a structural change is extended to much more than the current level of 500 generations. This is subject to further research. Interestingly enough, despite the fact that the number of parameters is reduced by the pruning the learning time remains high. This can be due to the complexity decrease or the limited retraining time.

E. Evaluation and Comparison of the Results

Pruning is the best and the most intuitive methodology for simplifying neural networks. N-Pruning proves to be best methodology for keeping fitness loss to a minimum, where D-Pruning expectedly proves to be the worst. MBP is on average a good method, but sometimes it does select the wrong connection, i.e an important one. Interestingly enough a series of reorganisations can decrease the neural complexity and the fitness enough to make fitness loss significant, which wasn't possible after a series of additions to the network.

The difference between a good controller and an average controller can be seen in the following two figures. The race car starts in (0,0) and drives to (20,0) where it turns. Hereafter it continues to (20,11) where it turns and continues to (-2.5,11) and from here it continues to (-2.5, 0) and on to (0,0). The controller tries to align the car on the straight line between the points.

![The route of an average controller.](image)

The better controller has less overshoot than the average
controller this ultimately means it is able to drive faster than the over controller. It is also better at anticipating the turns than the average controller, this also leads to faster lap times.

![Graph](image)

Fig. 6. The route of a good controller.

VI. Conclusion

This paper presented 4 different methodologies for reducing the neural complexity of an artificial neural network. The 4 methods described here used fundamentally different approaches.

This results indicate that it is possible to reduce the fitness of an artificial neural network by reducing the neural complexity of the network. These results are in line with previous research done in this field [13],[14], where fitness increases with the neural complexity. The implications of this research are somewhat subtle. It is doubtful that anybody deliberately wants to reduce the fitness of a given robot or network. This paper shows 4 ways of reducing the fitness if one desires. The paper does however confirm the fact, that if topological changes of a neural network are desired, one should consider the neural complexity. Failing to take the neural complexity into account, when pruning or adding components to a network, can have significant negative effect on the fitness of a network.

The methodologies proposed herein can be used in artificial evolution to make a neural network controller adapt to the complexity task at hand and the environment where the agent or the robot is based. This is would yield a step towards truly open-ended artificial evolution.

One of the methodologies proposed herein, simplification through pruning, is shown to be able to keep fitness high whilst removing redundant connections. This means that the methodology can help reduce the size of a network, and thus decrease the computational overhead, whilst keeping the fitness loss at insignificant levels.

References

Abstract—Directed sensing poses the problem of sensing in specific directions in synchronisation with robot motion while avoiding collisions with objects in other directions. The rebuild of an outdoor mobile robot, with the goal of mimicking a blind person navigating with echolocation, has provided the opportunity to experiment with a state machine based software architecture for landmark navigation. In this paper, we discuss the rebuild of the robot, the software architecture and an initial experiment in collision avoidance.

I. INTRODUCTION

Blind people have demonstrated exceptional ability when navigating with mobility aids using Continuous Transmission Frequency Modulated (CTFM) ultrasonic sensing [1, 2]. Our research goal is to understand how they use echolocation to navigate by mimicking their navigation with a mobile robot. Achieving this goal requires a software architecture that supports directed sensing synchronised with robot motion.

In this paper we describe the rebuild of a mobile robot for this project. The rebuild involved updating both the computer system and the navigation software. To demonstrate the capability of a state machine based approach to directed sensing, we present the results of a simple collision avoidance experiment. The purpose of this experiment is to study the question of whether the robot can avoid collision in a particular direction while only sensing in that direction occasionally. It confirms that the software architecture is suitable for exploring a new approach to navigation that mimics how humans find their way.

In heavy industry it is common for a machine to outlive its control system. A cold rolling mill in a local steel plant was originally installed in 1955. Its control systems have been updated every decade. It continues to produce high quality cold rolled steel. Fifty-three years after initial installation most of the original mechanical components and electric motors are still in continuous use.

Many of us have mobile robots in our laboratories that were switched off years ago. While possibly in need of a little maintenance, the mechanical components and motors are still operational. Many of the sensors still work. But the computers, software and interface electronics have failed and spares are unavailable. After the last student finished his project, the robot was turned off and new students are not interested in projects with obsolete equipment.

In 1998, we built an outdoor mobile robot (Titan), from a 4-wheel drive wheelchair [3]. As a Segway™ RMP 400 4-wheel Robotic Mobility Platform [4] for £16,000 plus customs and delivery was beyond what we could afford, we were faced with the challenge of either continuing to work with obsolete and failing hardware and software or updating.

The decision to rebuild (Fig. 1.) resulted in a requirement to develop new software from the old to work with new input/output (i/o) drivers. Also, it created an opportunity to improve the software architecture and libraries. As we had developed our own software we chose to go the code reusability route [5] rather than re-code in a standardised mobile robot language [6] such as CARMEN, [7].

The software for echolocation and robot navigation had been written by research students from scratch. It was monolithic and difficult to understand (one student gave up and quit his PhD). To overcome this problem we put a lot of work into developing libraries of low-level routines with program templates as a starting point for new applications. We wanted to keep the libraries and redesign the templates to enable a new set of research projects.

In the next three sections, we review the ability of a blind man to navigate with echolocation. A key skill that blind people develop is directed sensing by synchronising sensor panning with their body motion. The implications for mobile robot navigation are explored in Section V. From these, we developed the design requirements for the mobile robot. Its rebuild...
is described in Sections VI - IX. Finally, Section X reports on using the software architecture to control a simple experiment where collision avoidance is achieved with minimal sensor data.

II. NAVIGATION BY BLIND PEOPLE

"... I, as a thirty-six-year-old blind person, am able to thread my way through heavy pedestrian traffic smoothly, gracefully, and without collision, and can find an empty seat on the bus, an empty desk in a classroom, or an empty booth or table in a restaurant ..."  Gissoni, 1986

Fred Gissoni [1] is a blind man who has learned to navigate among sighted people using echolocation. Other blind people have also achieved significant navigation ability with CTFM ultrasonic mobility aids [8]. With the audio information produced by these aids they can perceive the geometric structure of the environment with sufficient clarity to enable them to move among both stationary and moving obstacles.

The abilities that blind people develop to navigate include: detection of object presence, recognition of object type, perception of object motion, prediction of their own motion, directed scanning of the sensor and synchronisation of perception with motion. Detecting object presence can be as simple as checking that there is an empty space to move into, to as complex as detecting that a door is ajar or a chair is empty.

Perceiving an object’s type involves recognizing that the object is a wall, a chair or a pot plant for example. Perceiving object motion is required to follow a person or to walk along a busy footpath. Predicting your motion is planning your next move(s) before you move. Directed scanning is sensing in the direction that you are going to move prior to moving. Synchronization of sensing with motion is required to successfully navigate at a walking pace in any cluttered environment.

Fred Gissoni made 10 audio lessons on how to use a CTFM sensor to navigate, because the sighted trainers of the blind did not understand how to use it. Sighted trainers cannot see the ultrasonic beam and struggle to identify which parts of the environment produced the echoes that they hear in the audio output of the CTFM mobility aid.

III. ECHOLOCATION

Echolocation is a sense of perception that is normally associated with bats not with humans. Bats provide a model of what is possible. By contrast, people were not created with echolocation as a normal sensing skill, so they have to learn it. Because echolocation is outside the experience of sighted people, many are sceptical of the ability of blind people to safely find their way with it.

Some blind people have learned to sense the environment with audible clicks vocalised from their mouth [9, 10]. As these are at audio frequency, their ability to discriminate between objects is probably limited to detecting large geometric differences where the range difference is sufficient to create a time delay sufficient to create reverberation.

CTFM ultrasonic sensors work at an order of magnitude higher frequency and thus should give an order of magnitude better discrimination. They continuously sweep down from 100 to 50 kHz [8]. Echoes from objects are demodulated with the transmitted signal to produce an ensemble of audio tones in the range 0 to 5 kHz. The frequency of each tone is proportional to the range to the surface feature that reflected that component of the echo.

In previous research [11], we demonstrated recognition of plants [12] and recognition of surfaces based on roughness [13]. All were done by extracting features representing surface geometry from the echoes. As the best results were for classification of rough surfaces, we are conducting research into navigating paths based on sensing roughness. It is for this research that we are upgrading our outdoor mobile robot Titan.

IV. DIRECTED SENSING

Blind people purposefully pan the CTFM mobility aid to hear echoes from objects in the environment so that they can both recognise those objects and determine the spatial relationships between them. Carefully thought-out movement of the sensor in synchronism with the motion of their body becomes a habit that helps them turn meaningless tones into meaningful sounds from which they build spatial maps of the environment.

An example from Gissoni’s lessons is how to detect a corridor [14]. Hold the sensor horizontal and pan slowly from side to side. Near the ends of the pan you should hear tones caused by echoes from the walls. Straight ahead you should hear nothing. Adjust the angle over which you pan so that you can clearly separate the walls from the corridor in the middle. Continue the horizontal pan and slowly tilt the sensor down until the gap in the middle is replaced by a tone of a different quality (a soft swishing sound). This signal is the floor. Finally, tilt the sensor back up, while horizontally scanning, until the signal from the floor is barely audible. Then when a person walks towards you their echo will be clearly distinguishable from the softer swishing sound of the floor.

Based on this description we set out to develop a software architecture that enables directed sensing in synchronisation with robot motion. As the robot moves, the sensor should be panned so as to detect the empty space in the corridor through which the robot plans to pass. This involves detecting the walls on both sides to define the corridor, detecting the floor to ensure there are no down steps or low lying objects and detecting objects in front to avoid collision.
V. MOBILE ROBOT NAVIGATION

To give a mobile robot the ability to navigate like a human will be a major innovation. We have chosen not to use a common SLAM (simultaneous localisation and mapping) approach with Kalman filters because there is no evidence that the human brain uses Kalman filters and our goal is to mimic human navigation.

Second, while ultrasonic sensor based navigation systems have been developed that use Kalman filters for odometer correction [5] and localisation [16], they have to reduce the sensed information to point features in order to use the Kalman filter. Observations of blind people navigating indicate that humans use an alternate approach that relies on the quality of their sensing of landmarks. Echolocation data provides a richer description of objects than points and we wish to use that additional information in the navigation system.

Achieving similar navigation capability to people with a mobile robot requires the ability to sense the location of objects, to track those objects, to recognise them and to decide which objects are important in the current navigation task. While many of our ideas build on prior research, combining rich echolocation data with directed sensing makes this approach new.

Part of our research is to determine what information is useful for navigation and how to represent it in an echolocation map. We have observed that blind people, like sighted people [17], increase their speed of navigation by reducing the sensed information to the minimum required by the task enabling them to increase the update rate by panning over a smaller angle.

To achieve the goal of programming a mobile robot to mimic human navigation, we are redeveloping the software on Titan to achieve the following navigation architecture. At the top level a command is given to carry out a task (such as "fetch a hoe"). Achieving this command requires a number of functions to be performed, including a navigation function (such as go to the garden shed).

First, we decompose this navigation function into a sequence of simpler navigation tasks from a set of available tasks stored in a map in a graph data structure [18]. We are examining how to represent this sequence as a set of connected states as a solution to the problems of planning and tracking the robot’s motion from one navigation task to the next. For example, a navigation task may be to navigate along a brick path from the garden to the shed. While navigating the path the robot is in the brick path navigation state. The first commercial mobile service robot, the Helpmate, decomposed navigation tasks into sequences of hallway navigation commands [19].

Based on its current perception of where it is on the path, the robot will predict where it will travel in the next three intervals of time if it continues along the current trajectory. An interval includes the time to pan the sensor and to scan the environment (a form of model predictive control [20]). Then it will move along the trajectory specified in the first interval, if the space is clear.

At the same time, it will direct the sensor to view the region in the second interval (i.e. predict where to sense) [21]. On a path, the sensor has four sensing positions to choose from: ahead (empty space), right border, ahead declined (path) and left border. To achieve directed sensing the robot will pan the sensor to scan each of these regions.

The robot will synchronise the panning speed with its velocity so that the space in the next interval is sensed before the robot attempts to move into it. Then it will adjust its velocities so that it continues to track down the path by turning to avoid obstacles and slowing down in narrow spaces.

VI. ROBOT REBUILD

We ran tests to determine what had failed, what was obsolete and what was operational. We found that the wheel-chair mechanics, motors and power amplifiers worked. The pan and tilt units, the encoders and the gyro-stabilised compass were also working.

The traction batteries needed replacing, a PCI interface card had failed, and other PCI interface cards were not supported in new software. We had written the control software in LabVIEW™ 5 running on a G3 Macintosh™ Powerbook under Mac OS9. The interface cards were PCI cards plugged into a Magna™ expansion chassis. We had modified it for battery operation using aircraft quality inverters. The inverters were operational.

In the last decade, computer technology has changed dramatically. As we wanted to retain as much as we could of the software, particularly libraries that we had thoroughly tested, we chose to update to LabVIEW 8.5. But this also meant changing to new serial drivers (VISA) and to new hardware drivers (NI-DAQmx_Base).

Titan was controlled by an on-board Macintosh Powerbook. So all software processes (development, control, data collection and some analysis) were performed on the robot. However, this meant that when the robot was moving we had to run after it to see the graphical user interface and to change command parameters.

In our new design (Fig. 2.) we have chosen to control the robot with an Apple™ Mac Mini with no keyboard, mouse or monitor on the robot. Communication with the user is done via a remote Apple MacBook over an IEEE802.11g wireless adhoc (peer to peer) network using remote desktop software. We wrote an application to set up the Mac Mini as the network server on start up. We also added an aerial to extend the range of the network (Fig. 2.).

Setting the robot up as the network host enabled us to stop all other network traffic from it, particularly applications that go looking for a network. We had
found that applications that went looking for a network and couldn’t find one (because the robot was in a field) hung the network drivers and caused delays in communication to the remote desktop. These delays lasted until the drivers timed out so they were long enough to be dangerous. Because the Mac Mini on the robot has no connection to any network, there is no network for these application to use so they do not call the drivers to access a network.

The user of the robot can sit in one spot with access to all software running on the Mac Mini while the robot runs around the field. Software development tools and control, data collection and analysis software all run on the Mac Mini. This has proved to be a very workable arrangement. Now we only run after the robot to push the emergency stop button when experiments result in a potential collision.

Both computers run Mac OSX. It has useful features including remote desktop and real-time threads. The main issue caused by the operating system upgrade was that the i/o drivers were replaced.

The serial PCI card that we used does not have a Mac OSX driver, so now all serial inputs are handled with USB to serial converters. Quality converters (e.g. Keyspan™) have individual serial numbers so they can be uniquely identified by the software. Cheep converters don’t, which creates problems when a USB card is unplugged because the USB dynamically reconfigures and the software relies on serial numbers to identify individual serial ports.

With one PCI card failed, one without a driver, and the Magma PCI expansion chassis having to be replaced to work with Mac OSX, we decided to do all the i/o with USB cards and Firewire for vision. We had had very good experience with the Keyspan USB to serial cards. On another project we had used one to read a 25 character packet from an IMU every 20msec, with excellent performance.

The change to USB i/o required rewiring the interface to the sensors and actuators. The change to a Mac Mini required the installation of an additional power inverter. One issue that we have to investigate further is the significant increase in power consumption. We get about an hour of continuous movement from a full battery charge.

VII. SOFTWARE LIBRARIES

We put a lot of work into developing and testing libraries of low-level routines for the previous version of Titan [22]. We wanted to keep what we could of these libraries. We found that, in the main, we could keep the logic but we had to rewrite the software that read data from the sensors and wrote commands to the actuators. The new LabVIEW hardware drivers are lower-level than the previous ones, so we had to develop routines to configure the USB i/o ports.

We developed 3 libraries: an In/Out library, a Control library and a Feature library. The In/Out library (Table 1.) includes routines to read all sensors and calculate values in physical units. Some, e.g. the Steering angle, return a single reading. Others, e.g. Odometer, run as a parallel process and produce a set of readings every 100msec.

Two important functions in the In/Out library are Logic and Control. The Logic function ensures the safe operation of the robot. It interacts with the hardware logic to switch to computer control and will return control to manual when any stop button is pushed. Also, the logic hardware switches the analogue hardware to choose manual or computer calculated outputs from the Control function to the motor power amplifiers.

Every 100 msec, the Sonar Producer process reads 1024 echo samples for each frequency modulated (fm) sweep transmitted by the K-Sonar [2] and places this...
echo array onto a queue for a sonar consumer to read and process. The Feature Library has routines for calculating the Power Spectral Density of the echo and extracting features from it for object recognition and environment mapping.

Table 1. Applications and libraries developed to control Titan.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Control Library</th>
<th>In/Out Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test sensors and actuators</td>
<td>Linear velocity controller</td>
<td>Logic</td>
</tr>
<tr>
<td>Controller tuning</td>
<td>Angular velocity controller</td>
<td>Control</td>
</tr>
<tr>
<td>Square</td>
<td>Bearing controller</td>
<td>Odometer</td>
</tr>
<tr>
<td>Corridor follow</td>
<td>Bearing fusion</td>
<td>Pan Tilt</td>
</tr>
<tr>
<td></td>
<td>Steering controller</td>
<td>Sonar Producer</td>
</tr>
<tr>
<td></td>
<td>PID loop</td>
<td>Video grab</td>
</tr>
<tr>
<td></td>
<td>PID loop cyclic</td>
<td>Compass</td>
</tr>
<tr>
<td></td>
<td>Motion</td>
<td>Steering angle</td>
</tr>
</tbody>
</table>

The Control library groups the Logic and Control functions into a Motion process that runs in parallel with other processes to provide safe control in applications. The Control library includes closed loop controllers for linear velocity, angular velocity, steering and bearing control. Two PID (Proportional Integral Derivative) functions are included. A separate PID controller is used for bearing because the feedback is cyclic.

Bearing varies clockwise from 0° (North) to <360°. Thus, in turning left from north the value jumps by 360. The gyro-stabilised compass produces this step. The odometer calculation of bearing was modified to produce this step, also. This step results in a step in the error between reference and feedback in the region of North which the PID controller has to handle.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The Control library groups the Logic and Control functions into a Motion process that runs in parallel with other processes to provide safe control in applications. The Control library includes closed loop controllers for linear velocity, angular velocity, steering and bearing control. Two PID (Proportional Integral Derivative) functions are included. A separate PID controller is used for bearing because the feedback is cyclic.

Bearing varies clockwise from 0° (North) to <360°. Thus, in turning left from north the value jumps by 360. The gyro-stabilised compass produces this step. The odometer calculation of bearing was modified to produce this step, also. This step results in a step in the error between reference and feedback in the region of North which the PID controller has to handle.

VIII. APPLICATIONS

The Control library groups the Logic and Control functions into a Motion process that runs in parallel with other processes to provide safe control in applications. The Control library includes closed loop controllers for linear velocity, angular velocity, steering and bearing control. Two PID (Proportional Integral Derivative) functions are included. A separate PID controller is used for bearing because the feedback is cyclic.

Bearing varies clockwise from 0° (North) to <360°. Thus, in turning left from north the value jumps by 360. The gyro-stabilised compass produces this step. The odometer calculation of bearing was modified to produce this step, also. This step results in a step in the error between reference and feedback in the region of North which the PID controller has to handle.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

VIII. APPLICATIONS

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.
produce a map of the region scanned by the sensor. The state machine is synchronised with the sonar update to minimise delays between range reading and velocity control. 

One example of a directed panning pattern is the corridor following pattern where the sensor pans front, left, floor and right in sequence. When following a corridor the left and right pans are at waist level to detect walls. By contrast when following a path they are depressed to path level to detect the path edges. Another is a wall follow pattern where the sensor pans front, left, and floor.

The state machine is time based. It calculates a new state every time step (100msec). Other processes run during the thread wait between time steps. The time that it stays in a state can be determined either by time or by events. In the case of a sensor panning pattern it is determined by time.

The corridor following state machine in Fig. 5. has 7 states. In each state it loops for a number of time steps (change state = F) and then on the last step (change state = T) it calculates the output commands to the pan and tilt unit and the robot to move into the next state.

Table 2. State machine set up for collision avoidance experiment. Key: C = copy previous value, E = go to final state, F = false, L = loop at state, T = true, Tr = transition to next state, \( V_T \) = velocity target mm/sec, En = enable output to Pan and Tilt unit

<table>
<thead>
<tr>
<th>State</th>
<th>Next</th>
<th>Time</th>
<th>Event</th>
<th>( V_{ref} )</th>
<th>( A_{ref} )</th>
<th>( S_{ref} )</th>
<th>En</th>
<th>( P_{out} )</th>
<th>( T_{out} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1L</td>
<td>2</td>
<td>10</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>2</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2L</td>
<td>2</td>
<td>8</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>4</td>
<td>%( V_T )</td>
<td>0.1</td>
<td>0</td>
<td>T</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4L</td>
<td>4</td>
<td>8</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>6</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>T</td>
<td>0</td>
<td>-20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6L</td>
<td>6</td>
<td>8</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>8</td>
<td>C</td>
<td>0.1</td>
<td>C</td>
<td>T</td>
<td>-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8L</td>
<td>8</td>
<td>8</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>2</td>
<td>C</td>
<td>C</td>
<td>0</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10L</td>
<td>10</td>
<td>2</td>
<td>0.1</td>
<td>0</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tr</td>
<td>0</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0L</td>
<td>2</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

IX. STATE BASED ARCHITECTURE

A state machine was first used as a high-level controller for mobile robot navigation in the ROBOL/0 language in the Yamabico robots [23]. The first program that students had to write was a state machine to drive the robot in a square. We are using a state machine to combine directed sensing with motion control.

![Fig. 4. Architecture of corridor following navigation software, with 6 parallel processes communicating through common variables, where \( V_{ref} \) = velocity reference, \( V_{back} \) = odometer velocity, \( V_{cmd} \) = output of velocity control loop, \( V_{out} \) = velocity command to motor controller, \( A_{ref} \) = acceleration per time step, \( S_{ref} \) = steering angle reference, \( P_{out} \) = pan angle reference, \( T_{out} \) = tilt angle reference.](image)

![Fig. 5. Time based state machine for path scanning. [Value] is the number of 100 msec time steps in state.](image)
State 1 is an initialisation state that initialises the robot to be ready to start moving. States 2, 4, 6, and 8 perform the directed panning and robot motion. At any time the user can press a switch in the GUI to stop the state machine. At the end of the current state it transitions to state 10 which shuts the robot down, and then it exits the state machine (state 0).

X. AVOIDING COLLISIONS

A concern with directed sensing is that the robot may collide with an object in one direction while it is sensing in another direction. When a particular direction is only sensed occasionally, will the robot detect an object in that direction in time to avoid collision?

To answer this question and in the process determine whether the collision avoidance that blind people achieve with directed sensing can be done with a mobile robot, we set up a collision avoidance experiment. The parameters of this experiment were chosen to represent a “hard” case scenario so that if the robot can be programmed to achieve it then it can be programmed to achieve simpler cases.

The parameters of the experiment are:

- The sensor pans a region in front of the robot 100° wide and 20° high using a corridor follow panning pattern of look ahead (0°, 0°)(pan, tilt) look left (-50°, 0°); look floor (0°, -20°); look right (+50°, 0°); repeat. A complete pan cycle involving 4 motions with stops between them takes 3.2 seconds. An angle of 50° to a surface is greater than the 40° used for excellent recognition of rough surfaces [13].
- The minimum amount of sensor data is used: range to the nearest object from a single echo during the 3.2 second pan cycle. The sensor scan used is the one pointing directly ahead of the robot. As the sensor emits an frequency modulated sweep every 100 msec, only 1 out of 32 echoes is used. Also the ultrasonic sensor is set to minimum range where it senses up to 2 metres.
- The robot’s linear velocity target is set to 400 mm/sec. While this is considerably less than its maximum velocity of 1 m/sec, it allows us to conduct the experiment in the laboratory and not have to go outside. So in 3.2 seconds the robot travels 1.28 m. As a result, at target velocity, the sensor reads the echo at least once while travelling the 2m maximum range of the sensor.
- The robot’s linear velocity reference is calculated based on the distance to the object reported by the sensor. It is set at 100% of target velocity for ranges greater than 1.7m, at 0% for ranges less than 0.8 m and at a percentage between 0 and 100 for ranges between 0.8 and 1.7m.
- As only a single sensor reading is used, the object is placed about 4 m directly in front of the robot, out of range of the sensor, but within the sensing cone of the sensor when it is in range. The object is a tall (780mm high, 570 mm wide) thin (55mm) block of dense styrofoam.
The robot was stopping too late because the linear velocity control was taking longer to settle than we had allowed for. The gains of the controller were tuned when running on grass. We reduced the gains to obtain less overshoot on carpet and we increased the range for 0% from 600 to 800mm. Most times it stops a small distance from the object (up to 100mm).

XI. CONCLUSION

The above experiment demonstrates that the software architecture enables the development of navigation programs to use directed sensing. Also, it shows that it is possible to avoid collision when only sensing occasionally in the direction of travel. At other times, the sensor can be panned to sense in other directions, for example to detect the border of a path to control steering.

The above experiment is a hard case with minimal sensor information, limited sensor range, and maximum update time. It can be made more robust. First the update time can be halved by looking for objects in echoes when sensing the floor, and using their range when calculating the reference velocity as well. Second, the amount of sensor information used can be increased significantly by using echoes from several sensors in different directions of interest and by using more features from the echoes.

Third, in many tasks the panning angle can be decreased from 50° to reduce the total cycle time, particularly when you only want to detect the presence of a path border and not recognise whether it is grass or leaf mulch, etc. Fourth, the range of the sensor can be doubled to 4 m but at the cost of doubling the update time to 200 msec.

To achieve our goal of mimicking human navigation using echolocation we have a lot more work to do. The state machine has to be made easier to program to make setting up experiments easier. The next experiment we plan to do (corridor navigation) requires the fusion of echo data from all the echoes in a pan cycle to control both linear velocity and steering. Then we have to add a higher level, where several navigation functions are combined to achieve a navigation task.

REFERENCES

Modelling the Lizard Ear: Directness of Phonotaxis with Noise Distractor

Lei Zhang, John Hallam, Jakob Christensen-Dalsgaard

Abstract—The auditory system of the lizard is highly directional because of acoustical coupling of the two eardrums through the large mouth cavity. A lumped-parameter model was made to simulate the system and used to control a small mobile robot. Previous work has proven that the model can successfully localise one pure tone sound source over a broad band. However, in the wild, there must also be distracting noises. In this paper, the model was tested with pure tone signal and a pink noise distractor. When the power of the noise was low, the model could locate the tone signal robustly. As the noise power was increased, the model become unstable and lost the tone signal. When the noise was very powerful, the model began to localise the noise instead.

I. INTRODUCTION

Lizard ears have strong directionality which is attributed to acoustical coupling of the two eardrums through the mouth cavity. The vibration of the eardrums is generated by the difference between external and internal sound pressure. The strength of the vibration depends on the amplitude of the sound pressures and the phase difference between them. This pressure difference receiver system is widely used by frogs [1], birds [2], insects [3] and lizards [4]. The cricket auditory system has already been deeply researched by biologists about the calling song [5], [6], the physical construction [7], [8]. Engineers have also modelled the peripheral system of cricket [10], [13] and internal neural structures [14], [15].

The auditory system of the lizard (see Fig. 1), a representative and robust coupled-eardrum system [16], was modelled by a simple lumped-parameter model shown in Fig. 2. Previous work showed that the model has strong directionality for pure tones over a wide band [18]. However, there are still interesting questions to ask of the model. In the previous work, the model was tested with one pure tone signal. When the noise was very powerful, the model would locate the tone signal. When the noise became more powerful, the model began to be unstable and finally lost the tone signal. When the noise was strong enough, the model would locate the noise instead.

The ear model and robotic model used in the experiments are described in section II. In section III, the experiment setup and methods are introduced. Section IV shows the experiment results and discussions. Finally comes the conclusion in the last section.

II. MODELS

A. Ear model

As pointed out in section I, the lizard auditory system determines sound direction by coupling the signals arriving at the two ears. The vibration of the eardrum depends on the direct ipsi-lateral signal and an indirect contra-lateral signal which arrives via the mouth cavity (the shadow part in Fig. 1). Whether the vibration is strengthened or weakened depends on the phase difference between these two signals, which is related to the direction to the sound source. So the system changes a small, difficult-to-measure time difference into a large easy-to-measure amplitude difference. In this way, the lizard can compare the vibration strength of the two eardrums to determine the sound direction.

In the model shown in Fig. 2, $Z_e$ is the tympanum impedance for the two ears. $P_e$ is the mouth impedance. $P_{1,2}$ simulate the ipsi-lateral sound pressures of the two ears. They are implemented by two voltage sources $v_{1,2}$. $i_{1,2}$ simulate the vibrations of the two eardrums. In the electric model, the tympanum impedance is a series connection of a resistor $R$, an inductor $L$ and a capacitor $C$. The mouth impedance is a capacitor $C_v$. In the implementation, the model is a linear filter system with two inputs $v_{1,2}$ and two outputs $i_{1,2}$. $R$, $L$, $C$, and $C_v$ are calculated based on the experimental measurements from the lizard Mabuya: the inter-aural distance, cavity volume, tympanum area, thickness, resonance frequency, mass and quality factor were used to calculate...
the parameters in the electric model (see [16] for details). These animal-derived values determine the most directional frequency of the model, which in this case is 1600 Hz.

Based on the model shown in Fig. 2, 

\[ \begin{align*}
    i_1 &= G_{11} \cdot V_1 + G_{12} \cdot V_2 \\
    i_2 &= G_{21} \cdot V_1 + G_{22} \cdot V_2
\end{align*} \]

(1)

\(G_{11}\) and \(G_{22}\) are the ipsi-lateral filters while \(G_{12}\) and \(G_{21}\) are the contra-lateral filters. The filter transfer functions are computed from \(Z_r\) and \(Z_v\), which depend on the signal frequency.

Fig. 3 shows the flow chart of processing the sound signals by the model.

\[ \Gamma = 20 \log \frac{||i_1||}{||i_2||} \]

is positive\(^1\) the model indicates a turn to the left, while a negative \(\Gamma\) implies a turn to the right. If \(\Gamma\) is 0, the decision is to go forward. These decisions trigger appropriate behaviours causing the robot to move in the chosen fashion, resulting in a turn to the louder side. In the experiments with the robot, a small range around 0 was used for going forward to avoid excessive turning when the two received amplitudes were very similar.

C. Robotic model

A small mobile robot made of LEGO was used in the experiment. The robot was 26(l)x15(w)x20(h) in centimetres. Two small Knowles Electronics FG series microphones were used to simulate the two ears of the lizard. The distance between the microphones was 13 mm, which is the interaural distance of the lizard species from which the parameters \(R, L, C_r\) and \(C_v\) were computed. The sound signals from the microphones were preamplified and input to a self-contained, portable DSP processor StingRay (Tucker-Davis Technologies), implementing the ear model — that is, the \(G_{ij}\) filters, summation and decision components illustrated in Fig. 3. The decision value \(\Gamma\) was also calculated by StingRay and sent to an RCX (LEGO product) box which controlled the two motors of the robot. The robot could turn right and left, and go forward. The turn radius of the robot was less than 10 cm. Fig. 4 shows the robot used in the experiments.

\(^1\)\(||i_1||\) means the amplitude of \(i_1\).

---

1. **Fig. 1:** Schematic Diagram of Lizard Ear Structure from Wever [11], redrawn and altered: TM is the tympanal membrane, ET the Eustachian tubes, MEC the middle ear cavity, C the cochlea, RW the round window and OW the oval window.

2. **Fig. 2:** Lump-Parameter Circuit Model of Lizard Ears: sound Pressure \(P_{1,2}\) is represented by voltage inputs \(V_n\) while tympanal motion maps to current \(i_{1,2}\). \(Z_r\) is the tympanal impedance and \(Z_v\) is the mouth cavity impedance.

3. **Fig. 3:** Flow chart of processing the sound signals by the model.

4. **Fig. 4:** Flow chart of processing the sound signals by the model.
Fig. 4: Small mobile robot used in the experiments. The robot was made of LEGO bricks and RCX.

III. EXPERIMENTS

All the experiments were done in an indoor 3 by 4 metre arena. The area was souded by sound screens covered by fleecees and furry blankets to damp sound reflection. A commercial video camera was fixed to the ceiling overlooking the experimental area. The camera took pictures continuously and sent the pictures to a computer which would localise and track the robot at video rate. Fig. 5 is a picture taken by the camera, which shows the environment of the experiments.

In the experiments, two sound speakers were used: one emitting the pure tone signals, the other the pink noise. The noise was white (Gaussian) noise filtered by a bandpass filter with 3 dB bandwidth from 1000 Hz to 2200 Hz — that is, the band in which the ear model shows directional sensitivity. The speakers were 2.5 metres in front of the robot and 1.5 metres to left and right of the initial heading direction of the robot. The robot started by going forward. So the robot must turn to reach a speaker, or fail.

In the experiments, 5 pure tones were chosen from 1400 Hz to 1800 Hz every 100 Hz. For each pure tone signal, noise with five different powers was tested against the tone signal. The power of the tone signal was always 60 dB while the noise power was one of 54 dB, 57 dB, 60 dB, 63 dB and 66 dB (each measured with a Brüel & Kjær measuring amplifier). For each tone and each noise power, 5 trials were recorded. The experiments were repeated with the tone signal on the left and on the right side.

A. Methods

The behaviour of the robot was classified as reaching the tone, reaching the noise or failure. ‘Reaching’ means the robot hit or passed the appropriate loudspeaker within 40 cm. Compared to the size of the speaker 20x20 cm, the size of the robot and the arena for the experiment, 40 cm is very near to the speaker. Failure means the robot did not reach either loudspeaker. In the experiments, the robot was tracked in real-time by the camera system; if the robot left the experimental arena, the trial was finished, even though the robot, sometimes, might later come back into the area.

This analysis shows whether the robot finally reached the tone, the noise or failed. However, it does not show how directly the robot approached the tone signal. For example, although the robot may reach the tone speaker by different tracks, the directness with which it does so can differ. So a directness vector encoding the motion of the robot to the tone signal was calculated for each trial.

The directness vector $v_{dir}$ of the robot is:

$$v_{dir} = \frac{1}{\sum_{i=1}^{n} l_i} \left( \sum_{i=1}^{n} l_i \cos \theta_i, \sum_{i=1}^{n} l_i \sin \theta_i \right)$$

Each track is separated into sections by turning points. $l_i$ in Eq. 2 is the length of segment $i$ of the track. $\theta_i$ is the angle between the direction of segment $i$ and the vector from its start point direct to the sound source.

From the definition of the directness vector, if the robot goes straight toward the sound source, all $\theta_i$ are equal to 0. So in this ideal case $v_{dir}$ is $(1, 0)$. If the robot approaches the sound source anticlockwise, most $\theta_i$ are positive. The horizontal part of $v_{dir}$ is then less than 1 and its vertical part is positive. $v_{dir}$ is thus in the first quadrant. On the other hand, if the robot approaches the sound source clockwise, most $\theta_i$ are negative and $v_{dir}$ is in the fourth quadrant.

Finally, the directness error $Dir_{err}$ was used as an extra variable to test the repeatability of the performance of the model. The directness error is the distance from the directness vector $v_{dir}$ to the ideal vector $(1, 0)$.

The directness error $Dir_{err}$ is:

$$Dir_{err} = ||\{1, 0\} - v_{dir}||$$

The mean and variance of $Dir_{err}$ over the five trails with the same setup (same tone and same noise level) were calculated to see how directly the robot moved to the tone and how repeatable its performance is.

IV. RESULTS

A. Localisation

Fig. 6 shows the tracks of the robot for the 1500 Hz tone signal when the tone speaker was located on the right side. When the tone was 6dB louder than the noise, shown in
Fig. 6: Tracks of the robot for a 1500 Hz tone signal when the tone speaker is located on the right side. The bold squares in the plots are the positions of the tone speaker. The circles in the plots are the positions of the noise speaker.

Fig. 6a, the robot went directly to that speaker. The effect of the noise on the robot was very small. As the noise became stronger, the robot began to be attracted by the noise, although it was still able to localise the speaker successfully. When the power of the tone signal and the noise were the same, as shown in Fig. 6c, it became more difficult for the robot to locate the tone speaker. Sometimes the robot would lose the target signal and sometimes the robot localised the noise instead. As the noise became more powerful, the robot could not go to the tone speaker. On the contrary, the robot was attracted by the noise and finally hit the noise speaker.

Whether the robot reached the speakers or not in the different cases is shown in Table I. In the table, ‘SNR’ means the signal to noise ratio. ‘6 dB’ means the tone signal is 6 dB stronger than the noise. The ‘Signal Position’ means the position of the tone speaker. If the tone speaker is on the left, the noise speaker is on the right. The ‘Tone’, ‘Noise’ and ‘Fail’ in the table means the robot reached the tone speaker, the noise speaker or failed.

From the results shown in Table I, when the tone signal was 6 dB stronger than the noise, the model worked very stably. Whether the tone signal was on the left or right, for each frequency, the robot could localise the tone speaker. When the tone signal was 3 dB higher than the noise, the robot still worked very stably for most cases, except for the 1500 Hz signal when located on the left. This is because, in most of those trials, the robot went just out of the area and then came back and reached the tone speaker, but the behaviour of the robot after it left the area was not recorded by the camera system.

When the power of the tone and the noise was the same, the performance of the robot began to degrade. In these cases, for the 1600 Hz signal — which was the most directional frequency of the model — the robot could still localise the tone speaker on both right and left sides. However, for other frequency signals, the robot began to fail. In some cases, for example the right case for 1800 Hz tone signal, the robot even localised the noise speaker instead of the tone. This is reasonable, because the tone frequency is included in the bandwidth of the noise and the noise is itself a legitimate target signal for the model to localise. So when the noise was powerful enough, the robot would turn to the louder side according to the decision model. As the noise became more powerful, more trials failed or localised the noise. When the noise was 6 dB stronger than the tone, and the tone was on the left, only for the 1600 Hz signal could the robot locate the tone signal with one failure. For all other frequencies, all the trials failed. When the tone was on the right, half of the trials failed and the other half localised the noise.

In Table I, even for the same tone and noise signal, the results can differ depending on whether the tone is on the left or on the right. This could be caused by three potential effects.

- First, the model is not absolutely symmetric in practice.
TABLE I: Localisation results of the robot.

<table>
<thead>
<tr>
<th>SNR</th>
<th>6 dB</th>
<th>3 dB</th>
<th>0 dB</th>
<th>−3 dB</th>
<th>−6 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Position</td>
<td>Left</td>
<td>Right</td>
<td>Left</td>
<td>Right</td>
<td>Left</td>
</tr>
<tr>
<td>1400 Hz</td>
<td>Tone</td>
<td>5 5</td>
<td>5 5</td>
<td>4 5</td>
<td>0 5</td>
</tr>
<tr>
<td></td>
<td>Noise</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
<td>5 0</td>
</tr>
<tr>
<td>1500 Hz</td>
<td>Tone</td>
<td>5 5</td>
<td>1 5</td>
<td>0 2</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Noise</td>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
<td>0 4</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0 0</td>
<td>4 0</td>
<td>5 2</td>
<td>5 1</td>
</tr>
<tr>
<td>1600 Hz</td>
<td>Tone</td>
<td>5 5</td>
<td>5 5</td>
<td>5 5</td>
<td>5 0</td>
</tr>
<tr>
<td></td>
<td>Noise</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 2</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 3</td>
</tr>
<tr>
<td>1700 Hz</td>
<td>Tone</td>
<td>5 5</td>
<td>5 5</td>
<td>3 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Noise</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0 0</td>
<td>0 0</td>
<td>2 5</td>
<td>5 5</td>
</tr>
<tr>
<td>1800 Hz</td>
<td>Tone</td>
<td>5 5</td>
<td>5 5</td>
<td>5 0</td>
<td>2 0</td>
</tr>
<tr>
<td></td>
<td>Noise</td>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
<td>0 3</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0 0</td>
<td>0 0</td>
<td>0 4</td>
<td>3 2</td>
</tr>
</tbody>
</table>

The two microphones, which were used to simulate the two ears of the lizard, had different spectral sensitivities. This causes amplitude and phase differences depending on the frequency being transduced. Because the model uses (indirectly) phase information to detect the sound direction, it was very sensitive to phase errors. Although we designed an FIR filter to compensate the two microphones and got acceptable results, the model still was not perfectly symmetrical.

- Second, the experiments were not done in an anechoic room. Although the experiment area was surrounded by sound screen covered by fleeces and furry blankets, reflection from ceiling and walls were still possible and might affect the movement of the robot.
- Third, the robot itself could also be a source of asymmetric behaviour. The robot had two wheels actuated by two motors. Because the two motors can not be identical, it is difficult for the robot to go straight forward.

These potential factors could affect the behaviour of the robot. When the target signal was strong enough, it was the dominant factor for the movement of the robot and the effects of asymmetry are not apparent. When the distractor and target are of similar strength, the asymmetry present in the implementation influences the decisions strongly enough to determine the behaviour of the robot.

B. Track Directness

The directness vector $v_{dir}$ described in section III was calculated for each trial. The average vectors $\overline{v_{dir}}$ of the five trials for the same case (same tone and same noise power) were calculated and are shown in Fig. 7. In each plot, 10 $\overline{v_{dir}}$ are shown, five for the tone on the right for each different noise level and five likewise for the tone on the left. In the plots, ‘+6’ represents the vector when the tone was 6 dB stronger than the noise and the tone speaker is on the left. ‘+6’ also means the tone was 6 dB stronger than the noise, but the tone was located at the right. So the operator indicates that the tone was stronger(+ or weaker(−) than the noise and its position signals the position of the tone speaker, before(left), after(right).

From Fig. 7, when the tone was on the right almost all the vectors are in the first quadrant, while for the tone on the left most vectors are in the fourth quadrant. This is because the robot started by going forward. For the right hand tone, the robot approached the tone clockwise, while for the left hand case the robot approached the tone anticlockwise. In the plots, there are also some vectors are in the second quadrant. In these cases, the robot localised the noise instead of the tone and all these vectors are for trials when the noise was much stronger than the tone.

For each frequency, whether the tone was on the right or left, when the noise became stronger the directness vector moved farther away from the ideal vector [1, 0]. This means that the performance of the robot became worse. This is very clear for the 1700 Hz tone signal.

C. Directness error

From the average vectors shown in Fig. 7, we can see how far the vector is from the ideal one and know how good the performance of the robot is. We can also know how the robot approached the tone source. However, because the vectors in Fig. 7 are averages, we can’t know the details of the five trials the average vector summarises. So we use the directness error defined in section III as an extra variable whose mean and variance were calculated.

In Fig. 8, the $x$ axis is the signal-to-noise ratio from −6 to +6 in dB. The $y$ axis is the directness error. In the plots, the dashed curves are the results for the left-tone case, while the dot curves are for the right-tone signal. The curves show the mean value of the directness error. The small vertical lines show the variance of the directness error.
From the plots in Fig. 8, it is clear, whether the tone is on the left or right, that as the noise becomes stronger the directness error becomes bigger, that is, the performance of the model in localising the tone signal worsens.

When the tone signal is on the right (dotted curves), from the plot in Fig. 8, the mean value for 1400 Hz is the smallest one. The variances are also small. That means the model has a strong directionality and the behaviour of the robot is also repeatable. The model can also localise the 1600 Hz tone signal when the noise is not very strong. But as the noise becomes more powerful, the robot sometimes loses the target. That means the robot begins to lose the tone signal and its performance becomes less repeatable. For other frequencies, the robot can only localise the tone well when the noise level is low.

When the tone signal is on the left (dashed curves), the model shows best performance for the 1600 Hz tone signal. The mean value of the directness error for that signal is small even when the noise is 6 dB stronger than the tone. The variances are also very small, meaning the robot worked very repeatably. For 1800 Hz signal, the model can also work well for a strong tone signal. However, as the noise becomes powerful, the robot sometimes loses the target. This is shown by the big variance. For other frequencies, the mean values are much bigger than for these two frequencies, indicating poorer performance.

V. CONCLUSION AND FUTURE WORK

A. Conclusions

Lizard ears have very strong directionality because of coupling of the sound signals between the two ears. In this way, the ear systems change a small difficult-to-measure time delay into a big easy-to-measure amplitude difference of eardrum vibrations.

A lumped-parameter model was used to simulate the auditory system of the lizard ear and the model was used to control a small mobile robot to localise sound signals. In previous work, the model was shown to successfully locate pure tone signals over a wide band. However, in that experiment, there was only a single sound source, a pure pure tone signals over a wide band. However, in that experiment, there was only a single sound source, a pure
Fig. 8: Mean and variance plots of Directness Error (see text). The dashed curves in the plots are for the tone-on-the-left, the dotted curves are for tone-on-the-right. They show the mean values. The small vertical lines show the variance.

tone, without distracting noise. For the lizard in the wild, there must be noises.

In this paper, the model was tested with two sound sources, one the target signal (pure tones) and the other a pink noise distractor. From the experiment results, while the noise power was low the model successfully localised the tone signal through the band from 1400 Hz to 1800 Hz. As the noise became stronger, the behaviour of the robot became less repeatable and it began to lose the tone signal. If the noise was strong enough, the model even began to localise the noise instead. A directness vector was used to analyse performance: for low noise level the model can always localise the tone signal, and as the noise becomes stronger, the directness of the path taken decreases. Measurements of directness error clearly indicate that as the noise level grows, the directness error becomes bigger and the behaviour of the robot becomes less repeatable.

B. Future Work

The auditory model used in this paper is (theoretically) symmetrical. For animals, although they can adjust the
system during the process of growing up, there must be a limit how symmetric the system is. Therefore, next, some biases will be added into the model to make it asymmetric and explore the effect of the distracting noise on such a system. The decision model used in this paper is also a very simple model, which just turns to the louder side. The next step is to use a neural network in the decision model. This is also a potential way to compensate for the possible asymmetry of the real auditory system.

REFERENCES


Relating Textural Concepts to Tactile Sensors

James Edwards, Jonathan Lawry, Jonathan Rossiter and Chris Melhuish

Abstract—The aim of this work is to generate a mapping between low-level features extracted from tactile sensors and high-level concepts, as represented by natural language labels. In [3] we have investigated feature extraction from data generated by an artificial finger applied to a number of discs, based on Fast Fourier Transforms and Principal Components Analysis. Here we report on an experiment to elicit labels for the same discs from human subjects. It is shown that the labels are allocated in a consistent, intuitive manner by the subjects. The potential of a mapping between the low-level feature space obtained in [3] and the texture labels is then investigated. K Nearest Neighbours is applied as a classification method linking feature values to labels, and a high-level of accuracy is obtained.

I. INTRODUCTION

To communicate effectively with humans robots will require an understanding of high-level concepts describing characteristics such as texture. Dargahi [1] identifies tactile sensing, and the ability to detect features such as surface texture, as an important area of research for robots working in unstructured environments. We previously presented a method of recording textural features from artificial discs [2] using a tribological (surface friction) sensor. Previous work into acquisition of sensor data has replicated human sensing modalities [8] or classified textures [7]. Whilst proving the sensor methodologies used are suitable for use with texture they do not provide a space for generalised semantic labels, or classification of unseen textures using linguistic definitions. Hollins [6] examined clusters of texture produced by similarity experiments, and building upon this Picard [10] investigated the semantics of touch and texture to create a representation of textural descriptors. Both of these studies examined the concept of texture, but did not link this work to sensor data. The aim of this work is to link the tactile sensor recordings from [3] to natural language descriptors of texture.

A. Previous Work

This paper extends previous work by the authors [2]. An artificial finger embedded with a microphone was used to record artificial textural discs revolving on a turntable. Figure 1 shows the experimental equipment and a textural disc. 22 discs [3] were created to constrain texture so as to examine the effect of different characteristics, notably frequency and height, using a square wave for frequency (pulse discs, 1 to 8), and a basic pattern of four squares for height and shape (square discs 9 to 22). Four discs (discs 11 to 14) have no frequency component because of the non-cyclic arrangement of squares in the pattern. Table I shows the 22 discs, their type and expected frequency. Using these discs a library of 40 one-second clips was recorded for each disc, and subsequently converted using a Fast Fourier Transform (FFT) for analysis. Principal Components Analysis (PCA) was used to extract a set of low-level features from the FFTs. In most cases and in the sequel a feature set consisting of the 5 most-significant principal components was used for analysis.

Fig. 1. Photograph of recording equipment and a textural disc.

Each disc can be reliably identified (87.6%) using the K Nearest Neighbours (KNN) method with only the 5 most significant components [3]. In addition similar textures are closely located in Principal Component Space (PCS), and recordings of each disc are tightly grouped compared to the distance between discs.

B. Conceptual Spaces

Gardenfors [4] has introduced the idea of Conceptual Spaces for concept representation, corresponding to a metric space of (possibly) dependent high-level dimensions. Such spaces provide an intermediate link between low-level features resulting from perceptual experiences and symbolic or linguistic descriptions of those experiences. Examples of conceptual spaces include the colour spindle and taste tetrahedron. The previous work on recording textures is a bottom up approach to creating a concept, by constraining the situation the effect of specific features can be quantified and used to build a space where these textures can be compared (currently the PCS obtained in [3]). To link this
TABLE I
TABLE OF PATTERNS, INCLUDING HEIGHT AND TYPE. GREYSCALE INDICATES HEIGHT. MOTION IS DOWN THE PAGE.

<table>
<thead>
<tr>
<th>Disc</th>
<th>Height (mm)</th>
<th>Type</th>
<th>Frequency (Hz)</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>Pulse</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>Pulse</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>Pulse</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>Pulse</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>Pulse</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>Pulse</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>Pulse</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>Pulse</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.3</td>
<td>Square</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.6</td>
<td>Square</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.3</td>
<td>Square</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.6</td>
<td>Square</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.15</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.3</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.45</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.9</td>
<td>Square</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

work to an abstract concept of texture we used a top down approach, similar to [6] and [10], to elicit labels for texture from human subjects. These experiments established which linguistic labels could be applied to the different textural discs. This paper presents analysis of the PCS using labels from the experimental subjects.

II. ELICITATION OF TEXTURE LABELS

In this section, we describe experiments carried out to elicit natural language descriptions of texture from human subjects. The experiments are divided into a feasibility study and a main study as described below. In both studies subjects ranged from under 18 to between 51 and 60 years old, with the majority in the 21 to 26 years age range. 9 subjects in the main study were female, however only one female subject participated in the feasibility study.

All of the subjects involved in the experiment were blindfolded before being presented with the discs. By blindfolding each subject they could not visually inspect the discs before touching them, restricting them to classifying the discs only on the basis of tactile information.

A. Feasibility Study

An initial study of fourteen subjects was carried out to refine the experimental method. Before fixing the procedure for the main study this smaller study explored which labelling tasks subjects are comfortable with, and adept at carrying out. This section has a short explanation of the Experimental Method, followed by a subsection on each task performed by the subjects, as listed below:

- Experimental Method
- Label Generation
- Clustering
- Ordering
- Application of Labels

1) Experimental Method: In this experiment, the tasks required of each subject were not revealed until they were wearing the blindfold. However, before this stage a brief explanation of the type of tasks and locations of textures on each disc was given. All test subjects were shown the flat disc as a control to illustrate the task and areas of interest on each disc. An inspection of the flat disc also helped some subjects to clarify the nature of the task that they were being asked to carry out.

These initial experiments used a variety of discs to examine the participant’s responses to the range of textures available. An initial set consisted of discs 3, 6, 9, 14, 16, 17, 19 and 22. These provided examples of pulse discs (3 and 6), a non-cyclic disc (14) and a selection of square discs (9, 16, 17, 19 and 22). In the clustering task a second set of discs were added (as described in section II-A.3), consisting of discs 1, 5, 10, 11, 15, 18 and 21.

Subjects used a variety of active methods to explore the textures, in contrast to the passive method used by the artificial finger. Passive methods of exploration are constrained rubbing of the sensor by the texture; active methods vary from constrained movement of the sensor over the
surface to free exploration of the texture by manipulation of sensor orientation. It has been shown [5], [11] that using active or passive exploration methods does not affect human perception. Active methods used by the subjects included using a single finger to explore texture flat against the table, and holding the disc in similar fashion to a steering wheel to explore the surface using their thumbs. In addition to the variety of exploration methods, the way subjects compared discs also varied considerably. In general, subjects felt one texture at a time, directly compared two using different hands, or compared a single disc to all previously examined discs by using one hand on the constant disc and the other to explore. Subjects generated qualitatively similar results independent of the method used to explore and compare discs, for example if a disc is labelled Very Rough it would not also be labelled Very Smooth.

2) Label Generation: Initially subjects were asked to examine the discs and generate verbal textual labels. At the start of this task subjects were prompted with Rough and Smooth as examples of textural labels and asked to provide labels for the other discs. Nine subjects were asked to perform this task and their replies were split into three categories: description, comparison and labelling.

Initially the subjects describe the structure of the disc, for example “uniform ridges going around”. The second response was to compare the discs to other discs in the set. This response only emerged after the second disc as subjects did not compare the first disc that they examined blindfolded to the control disc viewed prior to the start of the task. The comparisons between discs vary in complexity from the simple “smoother” to complex responses, for example the recognition that disc 17 is the “inverse” of disc 16.

The smallest group of responses are the labels requested at the start of the task because subjects have difficulty identifying applicable labels from their vocabulary to apply to the textural discs. This part of the task was the most time consuming since subjects searched their vocabulary for applicable labels. Subjects who were fast at this task often completely restricted their responses to descriptions and comparisons. Instead of generating textural labels, subjects would use structural features to describe discs, for example “tyres”, “grooved”, “corrugated” and “checkerboard”. However, subjects were able to change the given labels of “Rough” and “Smooth” to suit the current disc by applying hedges such as “Quite Smooth”. The duration and lack of ability to generate textural labels made this task unsuitable for use with a larger number of subjects; however, it does demonstrate that the subjects can appropriately apply labels to different discs.

3) Clustering: After the label application task each subject performed a clustering task. Participants were asked to cluster the initial set of discs according to similarity. Once clusters were established, the subjects assigned the previously unfelt clustering set of discs into the existing groups. When requesting an example of similar discs subjects were given a metaphor of colours to ensure there was no predilection toward a specific type of grouping. For example, “orange and red are similar when compared with blue”. Compared to the labelling task subjects grouped the initial sample of discs rapidly, although this may be in part because they had already explored all of the discs. Although no example clusters were given to the subjects consistent clusters emerged from the results. Pulse discs were grouped together, as were discs 11 to 14. The “chequered” discs (19, 21 and 22) were either together or split with 19 and 21 in a “smoother” group and 22 was allocated to a “rougher” group (which normally also included the pulse discs).

4) Ordering: To build on the successful clustering task in subsequent experiments subjects carried out an ordering task. By asking the subjects to order discs from Rough to Smooth we were able to investigate if the concept of roughness could be reliably applied to a set of discs by subjects in isolated environments. Each subject performed the task in a timely fashion with much more ease than the labelling task. The orderings produced had similar patterns (pulse discs were Rough, non-cyclic and discs with low height were Smooth), but enough variation to suggest that measures of disc texture are not rigidly fixed across subjects. Figure 2 is a photograph of a subject performing the ordering task. Subjects kept the unfelt discs together, incorporating each current disc into the order laid across the front of the desk. Although blindfolded, most subjects used the same layout of discs on the desk when performing the task.

Fig. 2. Photograph of a subject in the feasibility study performing the ordering task. The stack of unexamined discs is visible in the right of the photograph, whilst the subject’s order from Rough Smooth is seen across the left foreground of the table.

5) Application of Labels: As an extension of both the clustering and ordering tasks, a second labelling task was devised to test if subjects could confidently apply a range of predefined labels to discs. Each subject was given the labels Very Rough, Rough, Medium, Smooth and Very Smooth and asked to allocate a label to each disc in the sample. The subjects performed this task comfortably and quickly, with slight variation between results, suggesting that suitability of the task for a longer study. Although the subjects excelled at this task, they felt there were too many discs presented to them (13 discs).
B. Main Study

For the main study the discs were split into two sets of 11. This ensured a fair spread of disc types across both sets while restricting the number of discs that each subject needed to examine to a manageable number. Set A consists of discs 1, 3, 5, 8, 10, 11, 14, 15, 18, 19 and 21. Set B consists of discs 2, 4, 6, 7, 9, 12, 13, 16, 17, 20 and 22.

The subjects were asked to label the discs from Very Rough to Very Smooth and then order the discs from Roughest to Smoothest. Subjects were allowed upon request to order the discs first, and then to label them. Whilst the experiment was underway no encouragement or feedback was provided, even when directly requested by the subject. Interactions would only occur if the subject lost a disc, by either pushing the disc out of reach or off the table.

Twenty subjects were examined for each set of discs, approximately 25% of which were female. Tables II and III illustrate the labels applied by each subject for sets A and B respectively.

Three male subjects of varied age decided that one disc did not sufficiently match the other ten in the set and needed a separate label, each agreeing that the disc was between Smooth and Very Smooth, and hence required the label Quite Smooth. This was recorded but did not sufficiently alter the results as the majority of subjects assigned those discs to a pre-specified label. Disc 3 is the only bimodal disc with 8 Very Roughs and 8 Roughs; as 4 subjects also labelled the disc as Medium disc 3 is subsequently labelled as Rough. The results presented below use the mode of each disc to define the label associated with it.

When applying labels participants needed to accustom themselves to the range of textures present in the sample of discs. This enabled them to scale their responses to fit across the five labels. It was found that if subjects started to label discs without a preliminary exploration of the sample then they tended to group the discs into a small number of labels, normally Rough and Smooth. Exactly this problem occurred for three subjects who were then excluded from the analysis.

Some subjects felt all the discs before beginning the tasks; these subjects tended to be faster at clustering and ordering as there were no unfelt textures. Most subjects examined a small number of discs (around four) before feeling comfortable with the range of textures presented, and proceeded to perform the tasks. Subjects often changed their mind about labels or ordering whilst progressing through the unfelt discs to produce an order or set of clusters as listed in Tables II and III.

Analysis of the results using a Student t-test shows that there are relatively few subjects with anomalous results. The statistical significance threshold (p-value) was 0.05. Only one student in set A did not match the modal values accurately enough, compared to four subjects in set B. This is strong evidence for consistent labelling across the two samples.

III. DISTRIBUTION OF LABELS IN PCS

The textual labels in this experiment are restricted to Rough and Smooth. Ideally, the PCS should be arranged so that discs with the same label should be close. Also given the natural ordering of labels from Very Smooth to Very Rough we would expect discs which are labelled to be close in PCS to discs with a neighbouring label in the ordering. As shown in Figure 3 this is not always the case with the three most significant components of the PCS. The components of PCS shown in Figure 3 relate to the FFTs through an eigenvector transformation. The significance of each component is calculated by the amount of variation in the recordings. Since similar textures are located closely in PCS some discs with different labels are close together. For example, discs 19-22 are tightly clustered because of their similar chequered pattern, but have different heights and associated labels.

Discs 11 to 14 are tightly clustered nearest the origin of components 2 and 3 and are most negative in component 1. Physically these discs have no repeated pattern to create impacts on the finger, resulting in a much lower powered signal than other discs. The pulse discs are more loosely clustered compared to the textural discs, occupying the empty regions of the space furthest from the origin of components 2 and 3, and are mostly labelled Rough or Very Rough. Most of the Medium discs have little variation in components 2 and 3, and greater variation in component 1.

IV. CLASSIFICATION

Table IV shows the results of combining both sets A and B (set AB) in PCS and using K Nearest Neighbours (KNN) [9] to classify the recordings according to the labels provided by the human subjects. Based on earlier work [3] k was set to three neighbours and each recording was represented by the five most important components. Research reported in [3] suggests that five components classify 87.6% of discs correctly when using disc number as class. Table IV shows that 96.3% of recordings are correctly classified using labels; this result means that some previously confused recordings are labelled correctly because they are next to discs with the same label. However, 9 Very Rough recordings are misclassified as Medium or Smooth whilst 6 Medium recordings are misclassified as Very Rough or Smooth. The misclassifications are due to the close grouping of chequered discs in PCS. One other recording is misclassified; a single recording of disc 11 is misclassified as Medium due to it being confused with disc 17.

KNN of the sets A and B considered independently gives better results than when combined as over both sets there are only five recordings misclassified. This increase in accuracy occurs because the discs confused in set AB are separated into different datasets, with the exception of discs 20 and 22, which are both in set B.

Tables V and VI show the results of KNN analysis of set A trained on set B and vice versa. Training set A on B and set B on A gives 55.7% and 45.2% correctly classified recordings respectively. Nearly half of the correct recordings
### TABLE II
LABELS APPLIED TO SET A (VERY ROUGH (VR), ROUGH (R), MEDIUM (M), SMOOTH (S) AND VERY SMOOTH (VS))

<table>
<thead>
<tr>
<th>Subject</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>11</th>
<th>14</th>
<th>15</th>
<th>18</th>
<th>19</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>VS</td>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>VS</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>VS</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>VS</td>
<td>VS</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>11</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>12</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>VS</td>
<td>M</td>
</tr>
<tr>
<td>15</td>
<td>M</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>16</td>
<td>M</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>17</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>18</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>21</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>22</td>
<td>S</td>
<td>VS</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>23</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>25</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>26</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>R</td>
<td>R</td>
<td>VR</td>
</tr>
<tr>
<td>27</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>S</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>28</td>
<td>S</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>29</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>30</td>
<td>VR</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>31</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>32</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>K</td>
<td>S</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>33</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>34</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>R</td>
<td>R</td>
<td>VR</td>
</tr>
<tr>
<td>35</td>
<td>S</td>
<td>VR</td>
<td>R</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>36</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>VS</td>
<td>M</td>
</tr>
<tr>
<td>37</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>38</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>39</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>40</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
</tbody>
</table>

### TABLE III
LABELS APPLIED TO SET B (VERY ROUGH (VR), ROUGH (R), MEDIUM (M), SMOOTH (S) AND VERY SMOOTH (VS))

<table>
<thead>
<tr>
<th>Subject</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>9</th>
<th>12</th>
<th>13</th>
<th>16</th>
<th>17</th>
<th>20</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>M</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>VR</td>
</tr>
<tr>
<td>22</td>
<td>S</td>
<td>VS</td>
<td>R</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
</tr>
<tr>
<td>23</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>S</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>25</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>26</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>R</td>
<td>R</td>
<td>VR</td>
</tr>
<tr>
<td>27</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>S</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>28</td>
<td>S</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>29</td>
<td>R</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>30</td>
<td>VR</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>31</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>32</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>K</td>
<td>S</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>33</td>
<td>M</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
<tr>
<td>34</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>R</td>
<td>R</td>
<td>VR</td>
</tr>
<tr>
<td>35</td>
<td>S</td>
<td>VR</td>
<td>R</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>36</td>
<td>S</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>VS</td>
<td>M</td>
</tr>
<tr>
<td>37</td>
<td>M</td>
<td>VR</td>
<td>R</td>
<td>VR</td>
<td>M</td>
<td>VS</td>
<td>VS</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>38</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>39</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>VR</td>
<td>R</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>40</td>
<td>M</td>
<td>R</td>
<td>VR</td>
<td>VR</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>VR</td>
</tr>
</tbody>
</table>

Mode | R | R  | VR | VR | R  | VS | VS | M  | S  | S  | M  |

in both sets are because of the very similar PCS locations of discs 11 to 14. The other half of correct recordings come mainly from the PCS Medium class.

Set A has a number of Very Rough recordings that are closest to set B’s Very Rough discs, although nearly all misclassified recordings of set A are classed as Very Rough in set B space. All of the misclassified discs in set A are from the 28 recordings of disc 19 and 10 recordings of disc 18. These are misclassified as Medium due to their proximity to disc 20 in PCS.

Set B has all of its Very Rough recordings split across the other classes. 27 recordings of disc 4 are misclassified as Medium and 15 recordings of disc 20 are misclassified as Smooth. In addition to these, 79 recordings from discs 2 and 17 are misclassified as Very Smooth.

V. Histograms

The distribution of labels in PCS could provide insight into how discs are misclassified when using labels for classification. Figures 4 and 5 are histograms of each label.
Fig. 3. 3D plot of first three components in PCS. Each disc’s colour represents roughness from black (Very Rough) to light grey (Very Smooth).

TABLE IV
K Nearest Neighbours of PCS using Labels as class variables. Test data is odd recordings of set AB. Training data is even recordings of set AB. K=3, 1st 5 components of PCS.

<table>
<thead>
<tr>
<th></th>
<th>VR</th>
<th>R</th>
<th>M</th>
<th>S</th>
<th>VS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Rough</td>
<td>91</td>
<td>8</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rough</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>5</td>
<td>134</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smooth</td>
<td></td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Smooth</td>
<td>1</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE V
K Nearest Neighbours of PCS using Labels as class variables. Test data is set A. Training data set B. K=3, 1st 5 components of PCS.

<table>
<thead>
<tr>
<th></th>
<th>VR</th>
<th>R</th>
<th>M</th>
<th>S</th>
<th>VS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Rough</td>
<td>77</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rough</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>27</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smooth</td>
<td>42</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Smooth</td>
<td>2</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

for the first three principal components for set A and B respectively. Each principal component contains one point for each FFT (recorded in [3] using the artificial finger). Points in PCS are related to the FFTs via the eigenvector transform stage of PCA.

The histograms for Very Rough show very little crossover between the two sets. There are also significant differences in the distribution of Rough and Smooth between the two sets. However, the distribution of Medium is very similar, and the distribution of Very Smooth is almost identical in both sets. This explains the high level of accuracy for training Medium and Very Smooth labels across sets, and the high rate of misclassification for Very Rough, Rough and Smooth labels.

VI. DISCUSSION

In the label elicitation experiments subject’s uncertainty about the textures before seeing the flat disc may be because of the wide variety of textures in the natural world. Once subjects understood that they would interact with discs constrained to variation on a plastic surface they were more confident in carrying out the tasks.

The feasibility study suggested that it was difficult for subjects to identify a coherent and focused set of labels for the discs. One possible explanation for this is that each disc merely feels like plastic and that the difference between discs
lies mainly in their surface structure. Another explanation is the lack of applicable textural labels in the vocabulary of the feasibility study. Importantly subjects apply a predefined set of labels in a consistent manner. This suggests that humans have similar interpretations of textural labels all be it with some variation.

Three subjects decided that a certain disc needed to be allocated as a new sixth label. This behaviour may be an artifact of asking subjects to label 11 discs with 5 labels as the discs will not divide into groups equally. Whilst it did not alter the results, it does provide insight into how people group together these textures, suggesting a tendency to distribute the number of discs equally between labels.

When using labels provided by human subjects the increase in accuracy of classification shows that some discs with the same label are closely related in PCS. Whilst this effect is desired, the confusion of *chequered* discs highlights the limitations of this PCS. This close grouping shows that PCS arranges discs near similar patterns, not similar labels.

Classification of each set using the other set as training data highlights a problem with splitting the discs into two. The misclassifications of set A trained on B and B trained on A show that although the sets have been chosen to provide a broad range of similar discs the recordings in A are classed *rougher* when trained on B, and set B is classed *smoother* when trained on A. For example disc 1 is labelled as *Rough* whilst disc 2 is *Medium*, so disc 1 is classed *Medium* when trained on B and disc 2 *Rough* when trained on A.

Histograms shown in V illustrate that the two sets of discs exist in different areas of PCS space. A consequence of this is that the sets A and B cannot be compared to each other using points in the current PCS, which supports the observation that the *Very Rough* discs of A cannot be classified on B using KNN and vice versa. This could have been avoided by choosing sets that cover the space more appropriately, however the effect of changing the discs in each set is unpredictable as the subjects may apply different labels in a redistributed set of discs.

### VII. Conclusions and Future Work

#### A. Conclusions

Subjects in the label elicitation experiment were not comfortable generating linguistic labels to describe the textural discs. However, they were capable of clustering and ordering the discs according to either their own perceptions of similarity or class. Labels assigned are consistent across the subjects over the experiment, although some discs have a greater spread in labels applied.

The current PCS and KNN method cannot generate suitable labels for the recordings. This may be because of splitting the discs into two sets. However, even though the sensor discriminates reliably between discs it is not in the desired manner relating to the labels. The acquisition method used here is simplistic and possibly not recording certain aspects of the textural discs that are important.

Sensor data recorded by the artificial finger can be used to generate labels, but it is meaningless information unless it is joined to the human concepts of texture. The labels created by the finger should be in harmony with how humans think, although human’s thoughts about texture are not consistent in labelling these discs when split into sets.

#### B. Future Work

Results suggest that PCS may not be optimal for classification of texture into natural categories. There are several ways of improving performance using existing data. For example,
we may use patches to define convex regions of space, or fundamentally transform the space or perhaps use a different technique to classify the discs.

An extension to this work would be to use the recording method on a robotic hand and compare the resulting PCS with the current space. By using a robotic hand we will be able to simulate human movement across the disc and analyse the differences between the two data acquisition methods.

Sensor technology is constantly advancing. The work presented here could benefit from additional sensing capabilities on the finger such as pressure.

It would be an interesting addition to the human factors work to look at how people label different numbers of discs. Additionally sets A and B could be mixed to create two alternative sets C and D, these would examine whether discs retain the same labels.

VIII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Professor Jan Noyes, Department of Experimental Psychology, University of Bristol, for guidance about experimental methodology.

REFERENCES


Abstract— We consider the embodiment of a microbial fuel cell using artificial muscle actuators. The microbial fuel cell digests organic matter and generates electricity. This energy is stored in a capacitor bank until it is switched to power one of two complimentary artificial muscle technologies: the dielectric elastomer actuator and the ionic-polymer metal composite. We study the ability of the fuel cell to generate useful actuation and consider appropriate configurations to maximally exploit both of these artificial muscle technologies. A prototype artificial sphincter is implemented using a dielectric elastomer actuator. Stirrer and cilia mechanisms motivate experimentation using ionic polymer metal composite actuators. The ability of the fuel cell to drive both of these technologies opens up new possibilities for truly biomimetic soft artificial robotic organisms.

Keywords— EcoBot, Artificial Muscles, Dielectric Elastomer Actuator, Ionic Polymer Metal Composite

I. INTRODUCTION

Autonomous robots that can work in remote terrestrial and/or underwater environments present many design challenges for the engineer. Challenges include practical and efficient use and replenishment of power sources. For example, robots developed for underwater pollution monitoring will be expected to carry high density secondary power sources that can be recharged either onboard (through solar radiation or charging stations) or remotely by the human operator. In either case, the robot territory will probably be limited, as the distance between the farthest or deepest operating points cannot be greater than the distance that can be covered by the energy reserve available onboard. The same principles may apply to robots developed for terrestrial operation or even space exploration. In this context, energy becomes a scarcity and highly energy efficient modules become a necessity.

Nature provides many examples of biological agents that are successful energy managers in such environments. Animals are not only equipped with extremely efficient actuation mechanisms, but are also capable of extracting their energy from the environment in the form of food. Opportunistic examples of animals include the raccoon, which can survive in a wider range of habitats, utilizing almost anything edible and the salp, a urochordate oceanic drifter that is capable of feeding on virtually all microscopic life out of the water that filters through its body. The idea of developing autonomous robots that mimic how these animals operate, and maintain homeostasis is therefore very appealing.

With nature as our inspiration we seek to design an autonomous robot with the ability not only to extract energy from the environment but also to use it efficiently so that mission goals can be achieved in a timely manner. This will create a tool for scientists and engineers that will allow access to otherwise inaccessible areas thus offering the opportunity of performing functions that would otherwise be extremely difficult if not impossible to achieve. These systems will be designed according to a pre-defined mission and follow action selection rules that have the energy budget as the currency. Admittedly, the realization of such systems is still very far into the future.

Like a living animal, an autonomous robot will extract energy in the form of raw organic substrates from the environment and metabolise it into useful electricity. Such a robot would include in its duty cycle the collection of substrates such as plant material (fruits and vegetables) or even insect pests and other ingredients such as water to support an artificial-metabolism [1].

Current research is focused on producing a truly autonomous robot that harvests energy from its environment. The EcoBot project is developing an autonomous robot technology that employs microbial fuel cells (MFCs), as the onboard energy supply, which digest organic matter such as food waste and insects into useful power [1][2]. The low energy generated by the MFCs is stored in an on-board accumulator in order to be charged-up to a useful level, and then released to power the robot’s systems for tasks such as environmental measurement, data processing, communication and locomotion. This of course implies that the robot’s activity is limited by the amount of time it takes for the energy in the accumulator to reach the critical firing threshold and therefore results in a pulsed actuation [1][2].

Two exemplar robots have been developed as parts of the EcoBot project, namely EcoBot-I and EcoBot-II and the principle of operation is similar in both robots. Each robot is driven by 2 DC motors operated in a pulsed manner: each actuation interval is characterized by an active time, when the drive motors are on and an inactive time, when
the system rests to allow the accumulation of electrical energy. Improvements to system performance are being sought through research into MFC technology, and the efficient utilization of devices such as the DC motors that drive the robot. However great benefits might arise from employing new and novel ways of actuation.

Nature can provide the inspiration to find a means for improving the robot’s performance: by giving it “muscle” power! A muscle is a structure that can produce and control motion. Natural muscles are soft and compliant. They can act as both springs and dampers and are capable of large strains of the order of 20% [3]. They can be organized to act in antagonistic pairs around a joint and configured into a system that can traverse difficult terrain. Even without a skeleton, a muscle system can provide controlled multi-degree of freedom movement for manipulation and locomotion.

One good example is the octopus arm that uses sets of radial circumferential and longitudinal bands of muscle in a hydrostat mechanism [4]. In addition to animal propulsion natural muscle systems enable peristalsis for food digestion, pumping of blood, valve control by sphincters and many other tasks necessary for animal survival. Muscles act directly without the need of a gearbox and are self-sensing: the forces they develop can be tailored to provide precise force and motion, enabling actions such as locomotion, mastication and violin playing.

II. ARTIFICIAL MUSCLES

To synthesize a muscle we can look to emerging electroactive polymer technologies. Two technologies are under consideration for EcoBot: the dielectric elastomer actuator (DEA) and the ionic polymer-metal composite (IPMC):

A. Dielectric Elastomer Actuators

Consider first the DEA that is essentially a compliant capacitor consisting of an incompressible soft polymer membrane dielectric with compliant electrodes applied on both sides. When a voltage is applied, the charge accumulated on the electrodes gives rise to electrostatic forces that generate deformation in the DEA. Charges of opposite polarity act to draw the positive and negative electrodes together while the charges of the same polarity act to expand the area of the electrode (Fig. 1). When the charge is removed, the elastic energy stored in the dielectric returns it to its original shape. The linear motion produced by this deformation response can be used for actuation purposes.

![Fig. 1. Basic Operating Principle of a DEA](image)

DEA’s are driven by electric fields. The pressure capable of being generated by a DEA is defined by the following equation [5]:

\[ P = \varepsilon_r \varepsilon_0 E^2 \]  

Where \( P \) is the electrostatic Maxwell pressure, \( \varepsilon_r \) is the relative permittivity of the dielectric material, \( \varepsilon_0 \) is the permittivity of free space (\( \varepsilon_0 = 8.854 \times 10^{-12} \) F/m) and \( E \) is the electric field strength with units (V/m). This is twice the pressure capable of being generated by a rigid plate electrostatic device due to the interaction of the area expansion and thickness compression upon activation. Also note that the pressure is related to the square of the electric field, and not the voltage. Minimizing the thickness of the dielectric membrane therefore reduces the voltage required to generate a given electric field.

The level of deformation achieved at any given field is dependent on the stiffness of the polymer dielectric and electrode materials combined. DEAs have demonstrated active strains in excess of 300%, strain rates up to 34,000%/s, pressures of 7.7MPa, and energy densities as high as 3.4MJ/m³ [3].

Like natural muscles, DEAs can be controlled for position, speed or stiffness. DEA’s are capable of maintaining a steady state position and, due to their capacitive nature, can store energy. By controlling the rate of charging of the device the speed of actuation can also be controlled. Similarly, utilizing the geometry of the device and the level of charge stored on the DEA, it is possible to determine the electroactive forces, which in conjunction with knowledge of the mechanical behaviour of the DEA itself, can be used to control stiffness.

Some of the issues associated with DEA are linked to stepping up the voltage from the accumulator bank to the level required for actuation. While DEAs are inherently low power devices and require very low currents, at current scales of membrane thicknesses voltages of the order of 2.5-3kV are necessary for significant actuation. It is necessary therefore to implement a DC-DC converter to step up the voltage from the accumulator bank to that required by the DEA for actuation. The output voltage of the DC-DC converter is proportional to the voltage across its input terminals, but will lag behind the input voltage. As the robot’s capacitor discharges there will be a drop in
voltage across its terminals. Below a certain voltage it
might not be possible to produce satisfactory force or
movement in the DEA. Thus there is a race against time to
actuate before the voltage on the accumulator falls below a
critical level. The required force/displacement will
determine the effective size of the DEA artificial muscle
and this in-turn has implications on the power delivery
considerations described above.

In this paper we have investigated the feasibility of MFC
powered DEA actuators for EcoBot. In particular we have
developed a robotic sphincter mechanism for controlling
the outflow of waste effluent from its header tank. In the
course of doing this we have identified key design features
for the actuator and the circuitry that promote efficient and
economic use of energy.

B. Ionic-Polymer Metal Composites

Now let us consider an artificial muscle material that can
be considered as the compliment to the DEA described
above. Ionic-polymer metal composites [7] are members of
the family of electro-active actuator materials that rely on
ion migration to achieve actuation. Other members of this
group include ionic polymer composites of conducting
polymers, and carbon nano-tubes. In contrast to DEA
actuators, IPMCs operate at much lower voltages in the
range 1-3V (in the order of 10kVm⁻¹) and they undergo a
bending action rather than the expansion action of the
DEA. The lower operating voltage and contrasting
actuation modality make this material an interesting
prospect for integration into microbial fuel cell-powered
soft robots.

IPMCs are tri-layer membrane devices consisting of two
thin large-area electrodes sandwiching a polymer layer.
Fig. 2a shows the structure of an IPMC in cross section.
The electrode material is most commonly a noble metal
such as gold or platinum which is electro-less plated onto
the polymer layer. The polymer is an ion-exchange
membrane such as the fluoropolymers Nafion (from
DuPont) or Flemion (from Asahi Glass). The
electromechanical actuation process requires the migration
of mobile ions, thus an electrolyte is infused into the
polymer to facilitate this migration. The electrolyte has
traditionally been water or an alcohol such as ethylene
glycol. More recently non-volatile electrolytes such as
ionic liquids have been successfully used [6]. Ionic liquids
have the advantage that they do not evaporate and as a
result IPMCs fabricated using these liquids will operate for
long periods in air, in contrast to those containing hydrous
electrolytes which evaporate and result in a reduction in
actuator performance over time.

The electro-mechanical actuation mechanism of IPMCs
is illustrated in Fig. 2. When no electric field is applied to
the electrodes the mobile cations and the immobile anions
- which are bound to the polymer backbone - naturally form
a distributed equilibrium (Fig. 2a). When a voltage is
applied to the electrodes the cations migrate to the cathode
but the anions remain fixed in place. The presence of a
large concentration of ions at the cathode causes localized
expansion of the cathode material due to inter-ion
repulsions from Coulomb forces. The reduction in cation
concentration at the anode causes a complimentary
contraction of the anode. The net effect of these localized
expansions and contractions is a bending of the IPMC
towards the anode (Fig. 2b). This explanation is a
simplification of the complete electro-mechanical process
and does not consider the effects due to, for example,
solvent flux. For a more complete description of the
actuation mechanism see [8].

![Fig. 2. Structure and actuation of an IPMC](image)

We have used the EcoBot II energy harvesting system
to explore the capacity of both technologies (DEA and IPMC)
to produce useful actuation from MFC power.

III. MATERIAL AND METHODS

A. Microbial Fuel Cell Preparation

Energy harvested from MFCs was used for actuating the
artificial muscle devices. A stack of 8 MFCs charged a
bank of 64 x 6800µF capacitors up to 5V. The MFCs (see
Fig. 3) were made from silicone rubber with 5mm diameter
inflow and outflow ports on the top and bottom, to allow
for continuous fluid flow. The MFCs had an open window
on one side, to allow the fitting of a 30 x 40 mm
cation-selective membrane (VWR International), on the outside
of which the open to air cathode electrode was to be attached.
Carbon fibre veil was used for both the anode and cathode
electrodes, with a surface area of 270 cm². A 100mm length
nickel-chromium wire was used to provide the connection
point for the electrode. The microbes employed in these
experiments were of the same type found in activated
sewage sludge.

Activated sewage sludge samples were provided by the
Wessex Water Scientific Laboratory (Saltford, UK). The
samples were collected from the Cam Valley urban
wastewater treatment plant, which mainly deals with
domestic sewage. The plant is designed for a population
equivalent of 6,000 (360kg BOD day⁻¹) and has a sludge
age of hours. Activated sludge samples were taken from the

---

[8] This citation appears to be missing from the text or is not included in the provided data. It might be necessary to check the original source to provide accurate information.
aerobic process tank, in which suspended solids were 99.8%.

The collected samples were pre-processed, during treatment at the water works in order to remove pathogenic viruses. Samples were kept in their original water-based suspension, at 4°C anaerobically, and used within 3 weeks following anaerobic treatment. The sludge samples (pH 7.3) were mixed with sterile nutrient broth (25 g/L) (Oxoid, Basingstoke, U.K.) and given 24 h at room temperature prior to usage as start inocula in the experiments.

The open to the air cathode electrodes were moistened with 0.1M potassium hexacyanoferrate (a.k.a. ferricyanide, K₃Fe₂(CN)₆), mixed with 0.1M potassium buffer (K₂HPO₄). The pH of the electrolyte was adjusted to 7.

The microbial power was used for actuating two types of artificial muscle: DEA membrane actuators in two stretch ratios and bending IPMC devices in three sizes.

The experiments were set-up as shown in Fig. 4 below. The stack of 8 MFCs joined in series was connected to a bank of 64 x 6700mF (6.3V dc) capacitors, giving a total capacitance of 0.428F. The energy generated by the MFCs was accumulated in the bank of capacitors up to a predefined threshold, at which point the actuator was connected. The capacitor voltage drop, as a result of driving the actuator, was also recorded.

B. Dielectric Elastomer Actuator Preparation

A DEA sphincter (Fig. 5) was prepared and used for the actuation of a valve mechanism that was to function as a proof-of-concept sphincter on the EcoBot robot. This consisted of an acrylic membrane (3M VHB 4905) that was stretched equi-biaxially to nine times its original area and supported in a rigid plastic frame. Stretching reduced the membrane thickness to approximately 56µm. Electrically conductive carbon loaded grease was applied top and bottom to one half of the membrane area. At the center of the membrane was clamped a circular plate with a hole through which a post protruded. A thin walled rubber tube was pinched between the post and the inner surface of the plate. Actuation of the membrane shifted the circular plate sideways relieving pressure of the inner plate against the tube, thus allowing fluid stored in the tube to flow downwards by gravity feed. Two voltage converters were trialed: an EMCO Q50 DC-DC converter with the ability to boost voltage by 1000 times and an Artificial Muscle Inc. unit (serial T-3005) that could boost voltage by 600 times.

In a complimentary study we subjected the AMI converter to different capacitive loads, representing large and small DEA. We also fabricated additional sphincter membranes to investigate the influence of stretch ratio (3 and 4 times equibiaxial) versus MFC capacitance on membrane actuation. For this final study, actuation was characterized as the free displacement of the central valve on the membrane (pillar removed) for different actuation voltages.

C. IPMC preparation

The IPMC samples used in this study were fabricated from Nafion 112, 117 and 1110 ion-exchange polymer membranes. These membranes were five times chemically coated with gold using Asaka and Oguro’s method [9] to yield a dense fractal-like electrode structure with point-to-point surface resistance of less than 0.5Ωcm⁻¹. After final cleaning with HCl the membranes were soaked in a doping solution of NaCl overnight so that the majority of mobile H⁺ ions within the membranes were exchanged for larger...
Na\(^+\) ions. The membranes were kept in a purified water bath when not in use.

Since the actuation mechanisms of DEAs and IPMCs are so different it is clear that they have their own niche applications within the EcoBot platform. The low force, high displacement bending actuation of IPMCs makes them suitable for application within a microbial fuel cell as a stirrer mechanism and as a cilium. The stirrer is needed within the MFC to ensure that organic matter and microbes are uniformly mixed. The cilia mechanism can be used for fluid transport and control within the MFC digestive system and also as an external propulsion mechanism for the EcoBot body. To study the suitability of IPMCs as stirrer and cilia mechanisms within the EcoBot we have performed a set of experiments to assess the amount and form of actuation of IPMCs when driven by a charged MFC capacitor bank. Three sample IPMCs were used. The characteristics of these samples are detailed in Table 1.

<table>
<thead>
<tr>
<th>Sample number</th>
<th>Base Material</th>
<th>Length (mm)</th>
<th>Width (mm)</th>
<th>Dry thickness (µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nafion 112</td>
<td>12</td>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>Nafion 117</td>
<td>32</td>
<td>2.5</td>
<td>183</td>
</tr>
<tr>
<td>3</td>
<td>Nafion 1110</td>
<td>47</td>
<td>10</td>
<td>254</td>
</tr>
</tbody>
</table>

These three samples cover the typical dimensions and base material used in state-of-the-art IPMC actuators. Note that it is also possible to fabricate these actuators to a much smaller size (In [10] gold coated Nafion actuators of dimensions 30µm wide, 300µm long and 0.4µm thick were fabricated.)

Tests were conducted to determine the amount of fluid mixing that could be achieved using each of the samples 1-3 when powered by the MFC. The test setup in Fig. 4 was modified slightly with the inclusion of a polarity reversing switch SW2 as shown in Fig. 6.

![Fig. 6. Setup for MFC powered IPMC experiments](image)

The IPMC sample was electrically connected and mechanically fixed at one end, and was suspended in a bath of purified water.

Two capacitor bank sizes were used for the IPMC experiments: the same 64 x 6800µF capacitor bank as for the DEA experiments, and an additional smaller 10,000µF capacitor. The capacitor banks were charged to either approx 2.6V or approx 5.0V.

### IV. RESULTS AND DISCUSSION

#### A. Dielectric Elastomer Actuator Results

For the DEA sphincter the effectiveness of converting the low voltage energy of the robot’s accumulator bank into useful mechanical work depends on the inherent electrical characteristics of the DEA and the efficiency and dynamic response of the driving circuitry. During testing we charged the MFC capacitor up to 4.3V and this was boosted up to one thousand times by the EMCO voltage converter. The unit operated successfully but discharging the MFC electrical power into the DEA circuitry highlighted important leakage issues. In particular, once the output voltage of the converter dropped below the voltage across the DEA, the DEA discharged through the converter. Minimizing leakage mechanisms such as this would be beneficial to performance. This was demonstrated during our initial experiments. Despite having a lower peak voltage than the EMCO DC-DC converter, the AMI DC-DC converter proved to be more suitable for actuating the DEA sphincter due to its higher output impedance.

Our subsequent study of the AMI voltage converter demonstrated how its performance can be affected by DEA size. Size is associated with the effective capacitance of the DEA: the larger the surface area of the DEA, the larger the capacitance. The sphincter membrane had a capacitance of approximately 4nF. A range of stock capacitors in the range 2-10nF were coupled to the AMI voltage converter. The outcome of these tests depicted in Fig. 7 suggests that as capacitance on the output increases (i.e. a larger DEA), so more energy will be required from the voltage converter to drive the voltage rise across the artificial muscle membrane. This is evidenced by the time taken to charge in Fig. 7. Thus the capacitance of the DEA can influence voltage converter performance and must be taken into account when designing a DEA mechanism for an autonomous robot.

![Fig. 7. Simulated DEA rise time for AMI voltage converter vs DEA size represented by capacitance](image)
and 9 vs time. These measurements were conducted for a range of simulated MFC capacitors from 18.8 to 75.6 mF, each charged to 5V. While these capacitances were much smaller than the capacitor on EcoBot the experimental results indicate a trend that as capacitance increases so does displacement. They also show that increased stretch ratio, despite introducing greater membrane forces also results in larger actuation. This is in part due to the greater electric field and corresponding Maxwell stress within the thinner membrane. There is also evidence that the viscoelastic response is subdued in the tauter membrane as demonstrated by the sharper increase in displacement with the voltage onset.

![Graph](image)

**Fig. 8.** Central 3x3 membrane displacement vs time for a range of simulated MFC capacitances. Each was charged to 5 volts.

![Graph](image)

**Fig. 9.** Central 4x4 membrane displacement vs time for a range of simulated MFC capacitances. Each was charged to 5 volts.

**B. Ionic-Polymer Metal Composite Results**

To assess the ability of MFC powered IPMCs to act as fluid mixing and fluid transport devices, (i.e. stirrers and cilia) each sample was actuated in a still H₂O bath. Just prior to actuation a small amount of Methylene Blue dye (in crystal form) were sprinkled on the water surface. The slow dispersal of Methylene Blue enabled the observation of fluid flow patterns and hence the assessment of mixing ability. The experiments are outlined in Table 2.

The qualitative assessment of mixing ability in the final column is a rough guide to the performance of an IPMC sample when coupled with a specific capacitance and voltage, and is derived from visual assessment of videos of the IPMCs taken during the experiments. The number of effective actuations refers to the number of times the voltage polarity could be switched, resulting in step actuation, before the actuation became too small to have a measurable mixing effect.

Figs. 10 and 11 show captured images of the fluid mixing experiments. For each of experiments 2 and 6 three images show the state of fluid mixing at three time points. The Methylene Blue clearly highlights the fluid mixing vortices generated by the actuating IPMCs.

![Images](image)

**Fig. 10.** Fluid flow images from experiment 2 (smallest IPMC)

![Images](image)

**Fig. 11.** Fluid flow images from experiment 6 (largest IPMC)

As can be clearly seen from Figs. 10 and 11, the most mixing occurs when the largest and strongest IPMC was coupled to the larger capacitor bank charged to 5V. The mixing is extremely effective and rapid. This is most clearly shown in Fig. 11b where the actuation generates a very large vortex which is efficient at mixing the fluid. After full discharge of the capacitor bank the fluid is almost completely mixed, with only some remaining undissolved crystals of Methylene Blue showing in Fig. 11c.

**TABLE 2

**SUMMARY OF IPMC FLUID MIXING EXPERIMENTS**

<table>
<thead>
<tr>
<th>Experiment number</th>
<th>IPMC sample used</th>
<th>Capacitance bank (µF)</th>
<th>Charge voltage (V)</th>
<th>Number of effective actuations</th>
<th>Mean energy per actuation (mJ)</th>
<th>Mixing ability (1=worse, 6=best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.6</td>
<td>20</td>
<td>1.6275</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10000</td>
<td>5.1</td>
<td>20</td>
<td>6.44</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>10000</td>
<td>2.56</td>
<td>10</td>
<td>3.1518</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>10000</td>
<td>5.0</td>
<td>10</td>
<td>12.375</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>10000</td>
<td>5.0</td>
<td>1</td>
<td>123.75</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>435200</td>
<td>5.0</td>
<td>8</td>
<td>673.2</td>
<td>6</td>
</tr>
</tbody>
</table>

214
It is clear from both Figs. 10 and 11 that, when actuated in water, vortices are generated at the tip of the IPMC. In these experiments we are operating in the region of a high Reynolds number where inertial forces dominate over viscous forces. These same vortices, while effective at mixing fluids, also indicate the potential for thrust generation when IPMCs are configured as cilia. Additionally, from Fig. 10, we observe that the actuation of the IPMC resembles the flapping of a fish tail and the vortices generated can be expected to generate a significant component of thrust. Artificial cilia made from IPMCs have the potential to act not only as external actuators for MFC-powered swimming robots but also as internal fluid transport mechanisms. A MFC soft robot will require the pumping of digestive and waste fluid about the body and arrays of IPMC micro cilia can be configured to beat in a controlled traveling wave pattern to push these fluids along internal vessels or tubes.

Although the large IPMC has been shown to be most effective at fluid mixing, the consumption of such a large (for a MFC) amount of energy in mixing is likely to be unnecessary and too energy expensive. More suitable mixing mechanisms are likely to be found in smaller and thinner IPMCs which, although not able to mix fluid quite so well, consume far less energy.

Now let us consider the energy consumption profile of these IPMCs when connected to a capacitor bank. The actuation of an IPMC is dependant on both the voltage applied to it and the charge that can flow into it. The greater the voltage, the more rapid the ion and electrolyte movement within the IPMC, the faster the actuation and the greater the tip displacement. While accurate impedance modeling of an IPMC is difficult due to the complex electro-chemo-mechanical coupling of the material all equivalent circuit models share a very large capacitive component together with a parallel resistive component. It is these two components that characterize the gross current flow when actuated. In the MFC application therefore, the capacitance bank effectively discharges into a parallel circuit of a capacitor and a resistor. Unlike the DEA, though, the polarity of the equivalent circuit determines the direction of actuation, and as such a reversal of potential is required to achieve a reversal of actuation. In applications such as fluid mixing or micro cilia a repeated beating action is used, which requires the repeated polarity reversal of the IPMC and hence the repeated charge reversal of the capacitive component of the equivalent circuit. Clearly there is some scope to scavenge this charge during the reversal process, but due to the resistive component the amount of energy that can be scavenged is small. It is this resistive component that also represents the route for energy conversion from electrical to kinetic during actuation.

As the MFC capacitor bank discharges into the IPMC, voltage across the actuator falls, thus reducing the actuation speed and displacement. In our experiments, when the voltage across the IPMC fell to approximately 0.5V there was no effective mixing action. We also note from the first four rows of Table 2 that increasing the voltage of the capacitor bank increased the actuation of the IPMC, and hence increased the mixing effect of each stroke, but did not increase the number of actuations. This is illustrated in Figs. 12 and 13 which show voltage traces for IPMC sample 1 when the 10,000µF capacitor bank was charged to 2.56v and 5.0v respectively. Here we see voltage decay of the capacitor bank to approximately the same voltage after 10 cycles (20 actuations), despite different starting voltages.

Table 2 also lists the mean energy per actuation stroke for each experiment, which varied from 1.627mJ to 673.2mJ. Note that the total energy E used in each experiment is calculated as in Equation 2.

\[ E = \frac{1}{2} CV_{\text{init}}^2 - \frac{1}{2} CV_{\text{final}}^2 \]  

where \( V_{\text{init}} \) and \( V_{\text{final}} \) are the initial and final voltages across the capacitor bank respectively and \( V_{\text{final}} \) is taken to be 0.5V.

We observe a monotonic relationship between mean energy used per actuation and the fluid mixing ability of an IPMC. The one exception is experiment 5 where the
10,000uF capacitor can only drive the large IPMC with one very small actuation. In this case the charge in the capacitor is simply too small to be transferred to the large IPMC and maintain a sufficient voltage for a sufficient length of time to achieve an effective actuation. When the larger 0.428F capacitor bank is charged to the same 5V the number of actuations rose from 1 to 8. The voltage profiles in Figs. 14 and 15 respectively illustrate these two cases.

Fig. 14. Experiment 5, IPMC sample 3 driven by 10,000µF charged to 5V

Fig. 15. Experiment 6, IPMC sample 3 driven by 435,200µF charged to 5V

V. CONCLUSION

Actuation efficiency plays a pivotal role in the design of autonomous agents, for which viability greatly depends on their success in collecting their energy from the environment. For this reason it is very important to invest in new technologies that move away from traditional designs. In this study, two different novel technologies were investigated as possible actuation mechanisms for EcoBot and both demonstrated the feasibility of such an approach and outlined areas with room for further improvement. It is envisaged that such systems will form the core of future robot hardware design.

We have shown that the microbial fuel cell can generate enough electrical energy to power the complimentary artificial muscle technologies of dielectric elastomer actuators and ionic-polymer metal composites. This ability enables the possibility of a truly soft EcoBot, one where the MFC forms a soft stomach and gut and the artificial muscles form a soft pliable body. We have shown application of DEAs in this domain through a prototype sphincter mechanism that can control the flow of waste fluid from the MFC. We have also shown the application of IPMCs through a prototype stirrer and cilia configuration. Experimental results with respect to actuation and electrical power consumption have also been shown. Clearly there is much work needed in this area and future research will focus on isolating the key mechanisms in the EcoBot that can be converted to artificial muscle power and efficiently using the small amounts of electrical energy that the MFC can generate.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Andy Levy for producing the silicon-rubber Microbial Fuel Cells.

REFERENCES


216
Abstract—Traversing the complicated terrain of an animal’s natural environment is not difficult for a highly evolved creature. However, complex terrain can pose a great problem for legged robots, even those having undergone many cycles of evolutionary simulation. This is especially true for the potential for great sinkage and slippage of a walking robot when traversing the soil on other planets. This paper investigates the benefits of incorporating the deformation of soil under the feet of a walking robot as a feedforward input to the gait and leg controllers of the vehicle. It also introduces the implementation of the visco-elastic properties found in biological muscle into the joint controller of the walking robot and how this could be synchronised to the soil deformation inputs.

I. INTRODUCTION

The majority of the work that has been conducted on tractive soil interaction relies on physical experiments with foot plates and various terrestrial soil simulators [4,5] or through the investigation of wheeled and tracked locomotion systems [6,7,8]. For a legged vehicle, the main consideration is the sinkage of the footprint as the soil changes under the pressure of each step. Other factors such as slip are exceptionally important with determining overall vehicle trafficability on a given soil. Several laboratories exist in research groups around the world to perform pressure plate trafficability on a given soil. Several laboratories exist in research groups around the world to perform pressure plate trafficability on a given soil. Several laboratories exist in research groups around the world to perform pressure plate trafficability on a given soil. Several laboratories exist in research groups around the world to perform pressure plate trafficability on a given soil.

The aim of this research is to develop a system to accurately predict the tractive performance of walking rovers over a particular soil. This soil simulator programme reproduces Earth-based soil modelling done in laboratories and is validated against these real-world experiments. The programme also includes the flexibility to allow for soils with any known or estimated parameters to be used, such as planetary soils. A leg with footprint of a desired shape is pushed into the soil at the anterior extreme position angle to determine the terramechanic properties based on the applied footprint pressure. The leg is then rotated about the toe, pushing through the soil in front of each leg. The sinkage and slippage parameters determined by this simulation will then be utilized as a feedforward component to the legged vehicle control system, essential for the accurate control of an autonomous legged vehicle on a planetary surface. Future work for this simulator will include modelling the applied load and lower leg segment movement in a 3D physics-based environment.

II. BACKGROUND

A. Hexapod Control

Many forms of control for hexapod robots are widely accepted. From central pattern generators to learning walking behaviours through neural networking, legged locomotion over flat terrain has been well researched, eg. [13,14,15].

The notable subsystems of a legged robot controller include different control algorithms for the joints, legs, gait selection and overall navigation. The joint controller directs
the motion of the joint between the anterior and posterior extreme positions (AEP and PEP). The leg controller regulates movement of the joints in harmony to provide the swing and stance phases of the leg when walking. The gait controller selects the best gait for the vehicle given the terrain, obstacles, desired velocity and other considerations. The navigation controller then directs the system from its current position to its goal.

Standard legged robot controllers have been developed for flat surfaces and to help surmount hazardous terrain, e.g. [1,13,16]. However, complexities begin to exist when attempting to model the response of biological systems in robotic form when traversing more challenging terrain. Some examples of robotic controllers for legged vehicles and their capabilities on complex terrain are noted in [17,18,19,20].

Application of biologically inspired control to a robot can be done through a number of methods. This could include the overall vehicle design, the use of biologically-inspired hardware, controllers or materials, or any other technique inspired from nature. This team has applied several of these bio-inspired methods to the control of a hexapod robot.

B. The Biologically Inspired Vehicle

Primarily, the vehicle itself is biomimetic, with body and leg segment designs based on the stick insect (Carausius morosus), as shown. The stick insect is one of the best biological targets for land vehicle locomotion and is a clear example of legged locomotion superiority on unexpected rough terrain [21]. This is due to its slow walking gait, long leg reach and elongated body, among other reasons.

This vehicle, shown in Figure 1 below, was initially designed at the University of Sussex for the University of Surrey led Bionics and Space Systems Design contract awarded by the Advanced Concepts Team at the European Space Agency [3].

In the initial design of the vehicle, simulated muscles were applied to joints, as opposed to electrical motors, as shown Figure 1. However, for the investigations performed in this research, electric motors replace these artificial muscles due to the heritage they have in the space industry for reliability in extreme environments. The overall system with the electric motors and all hardware required for operation has a mass of 23kg. Beyond the above expressed details, the vehicle design has no further changes from its initial configuration.

C. The Biologically Inspired Controller

The overall control architecture is also based on biologically inspired methods. Although no behavioural control or learning algorithms have yet been implemented, feedforward considerations have been employed. The feedforward contributions to the vehicle controller are each modelled separately:

- The passive plant mechanics of the muscle model are applied at the joint level controllers and the Winters and Stark model [22] is utilised as this feedforward input.
- The anticipated sinkage is calculated as a function of legs in contact with the soil, and incorporated at the leg and gait controller levels.
- The anticipated slip is determined as a function of the gait and forward vehicle velocity, and is incorporated as part of the leg and gait controllers.

The general configuration of the proposed controller is shown in Figure 2 below:

In the vehicle controller diagram above, each controller has a specific purpose and individual responsibility. Cooperatively, they will control the forward motion of the vehicle. The responsibilities of the controllers are as follows:

- Joint controller – controls the actuators in each joint to move from one position to the next, ensuring each leg segment is in the correct orientation for efficient leg movement.
• Leg controller – moves the foot from the posterior extreme position to the anterior extreme position to ensure a full range of movement of the leg. The footprint pressure sensor will ensure contact pressure is near the expected value, the point at which the foot has achieved the expected sinkage based on the number of legs in contact with the soil.

• Gait controller – cycles between the legs of the vehicle to introduce a gait to the vehicle. This gait could be a single-leg moving wave gait a 3-leg moving tripod gait, or any number of other standard gaits. Each gait will allow a different number of legs to be in contact with the ground, therefore varying surface pressures between the footprints and the soil may exist.

• Navigation controller – navigation is beyond the scope of this paper and is not yet considered in this research. However, there is the potential that the anticipation of very high sinkage or slip could be a future input to the navigation controller if an avoidance behaviour would be desired.

The biological inspiration to the above controller is primarily focused on the three feedforward models: passive plant muscle mechanics, footprint sinkage model and leg slip model. Each of these is described in further detail in the sections below.

III. BIOLOGICALLY INSPIRED MUSCLE MODEL CONTROL SIGNAL

A. Muscle and Joint Modelling

Previous research was done by this team to model the visco-elastic effects of biological muscle mechanics on the joints of a walking robot. In its simplest form, applying a standard spring factor to resist joint movement and/or a damping factor to diminish the change in angular velocity of the moment arm would simulate the natural resistance of a biological muscle.

Utilisation of a more detailed first-order function, such as the sigmoid function, could produce models more accurate to the shape function created by biological muscle. The sigmoid function: $f(x) = \frac{1}{1 + e^{-ax}}$ is plotted below for different values of $a$. The similarities between this sigmoid plot and the Winters and Stark passive elasticity shape function of biological muscle are evident, as compared in Figure 3 [22].

B. The Winters and Stark Model

As initially described by Winters and Stark, the left and right antagonist muscle actuators work in unison to move a second-order passive plant. In short, the passive plant includes the natural visco-elastic properties of the muscle that the antagonist muscles must overcome, in addition to the work done to move the rest of the segment.

In biological creatures, the natural muscle forces are subconsciously understood by the body and their response is predicted by the body and taken into account when, for example, a person moves his arm quickly to catch a ball. Not only is the arm segment moved to put the hand in the perfect position to catch the ball, but the natural resistance of the muscle must also be overcome.

The model developed by Winters and Stark to determine joint torque is as follows: $T_j = k_x + Cx + k_1(\text{e}^{k_x} - 1)$

Figure 3 – Winters and Stark position-torque relationship for passive muscle elasticity [22] with sigmoid curve overlaid for shape comparison.

where $k_1$ and $k_2$ are muscle-specific functions relating to joint capability, $k$ and $c$ are spring and damping control coefficients and $x$ and $\dot{x}$ are the position and velocity of the muscle shortening. This modelling of the passive mechanics of biological muscle, when applied to the joint control of an electric motor at each leg joint, will apply a level of bioinspired control to the overall system.

C. Using the Passive Plant as a Control Input

Specifically, the formation of a passive plant design must be carefully determined and compared to biological creatures. However, utilising the muscle-specific characteristics and spring-damping coefficients of a stick insect would not suffice, as the stick insect has evolved over millions of years to attain the idea muscle properties for that specific insect in its specific environment. The muscle properties of an insect that small would not suffice for a 23 kg robot.

Additionally, the determination of accurately simulated muscle properties must be secondary to the completion of all other system influences (such as leg slip and sinkage effects). Otherwise, the fine-tuned muscle properties would only be representative of the systems itself, and not be accurately tuned to take into account the soil deformation effects.

Simulations are in work to compare the muscle properties of the legs of creatures closer to this mass to use as a baseline for the robot. Then, further fine-tuning of the leg
will be required to determine ideal muscle properties for the vehicle on a reduced-gravity planetary surface.

IV. BIOLOGICALLY INSPIRED TERRAMECHANIC CONTROL SIGNAL

In walking, biological creatures anticipate the way the ground beneath them will shift with each step. Reflexes exist to help these creatures recover in case their expectation is not correct. However, barring that ability, they, for the most part, have an expectation of how the ground will deform with each step. Similar to a human walking on a sidewalk and then in the sand, the expectation of the sinkage and slip that will be encountered with each step is differently considered. Some research have done research in this area with respect to wheeled vehicles [23,24] or learning behaviours for legged vehicles [25,26,27,27], but few researcher have investigated the true terramechanic response of a legged vehicle locomotion [28]. In this paper, both sinkage and slip response for the above-described walking robot have been investigated.

A. Draw Bar Pull and Trafficability

Sinkage is the distance below the initial surface of the soil where the foot comes to rest after completing a step. Sinkage is greater for smaller footprint areas and higher contact pressures and is measured in the z-direction, assuming this is perpendicular to the plane in which the vehicle’s forward velocity is located. Sinkage affects a walking vehicle in two ways. First, in cases where a greater sinkage is encountered than desired, the gait pattern should be changed to ensure more legs are in contact with the ground, thereby increasing the contact surface area and lowering sinkage. Additionally, the sinkage should be fed into the leg controller to ensure that the expected final resting position of the foot is not on the surface of the ground, but actually lower than the surface based on the expected sinkage.

Slip is the distance lost from forward tractive effort as soil deforms opposite of the direction of travel. Slip is measured in the x-direction, assuming the vehicle is moving in the x-y plane. Slip also affects the vehicle in two ways. The first is if the vehicle is not making enough forward progress based on the gait selected, a different gait with a lower slip profile should be selected to ensure maximum forward progress of the vehicle. Additionally, slip would affect the leg controller and the final resting position of the foot based on desired forward movement of a single step. If slip is fed forward into the controller based on expected soil conditions, then additional forward progress can be incorporated into each step to counteract the expected slip and attain full use of the stepping capabilities of each leg.

Both sinkage and slip of a lower leg segment in soil are shown in Figure 4.

![Figure 4 - Leg sinkage and slip in soil](image)

Both expected sinkage and slip characteristics of the vehicle capability in the soil must be fed forward into the gait and leg controllers to ensure a more biologically inspired locomotive capability. As such, the pre-programming of these expected soil behaviours must be incorporated into a robotic control system.

The capability of a vehicle to traverse a specific soil is known as trafficability and measured as drawbar pull. Many authors have investigated this behaviour utilising Bekker Theory, based on the classical soil mechanics work of Karl Terzaghi [29,30,31,32,33], is considered the standard method for determining tractive capability for vehicles.

Drawbar pull is measured by determining the maximum capability of the vehicle in a soil, known as soil thrust ($H$) and reducing it by the amount of slip encountered in the soil and additional resistances ($R$) encountered by that vehicle. The accepted way to determine drawbar pull is as follows:

$$DP = H - (R_c + R_b + R_g + R_{other})$$

(2)

This methodology is standard for determination of the capabilities of a wheeled or tracked vehicle in soil, where the general resistances are defined as compaction ($R_c$), bulldozing ($R_b$), gravitational ($R_g$) and any other resistances that may be modelled in the system ($R_{other}$). However, there are differences between wheeled/tracked and legged methodology for determining each of these resistances.

B. Soil Thrust

Determination of soil thrust is done through investigation of the Mohr-Coulomb relationship between the contact areas of the vehicle and the natural soil properties. The maximum soil thrust available to a vehicle is:

$$H_0 = AC_0 + W \tan \phi$$

(3)

And soil thrust modified by slip ($s$) is noted as:

$$H = H_0 \left[1 - e^{-sl/\kappa}\right]$$

(4)

In the above relationships, $C_0$, $\phi$ and $\kappa$ are the soil-specific properties of soil cohesion, friction angle and shear deformation slip modulus, respectively. $A$, $W$ and $l$ are the vehicle-specific properties of soil contact area, weight and
contact length, respectively. These relationships are common between wheeled/tracked and legged vehicles, as they simply express the vehicle’s contact with the soil.

However, it is important to note that vehicle weight may be distributed differently for a legged vehicle due to non-continuous contact of the legs with the soil. Additionally, the contact area of a footprint may not be simplified to an evenly distributed rectangle as it is for a wheel or track if that leg does not have pure vertical contact with the soil.

As shown below in Figure 5, the frictional influence on a mass in soil is expressed in terms of the angle of inclination \( i \) and the soil friction angle \( \phi \). Soil cohesion \( C_0 \), however, is not affected by inclination. Therefore, the soil thrust equation is modified as follows:

\[
H_0 = AC_0 + W \cos i \tan \phi
\]  

(5)

Figure 5 – The frictional influence of gravitational resistance to a mass in soil \([34]\).

Although there have been some attempts to mathematically determine the value of slip for a vehicle \([7]\), the more traditional method for determining slip is through simulation and experimentation. Therefore, it is the slip that must be found for varying inclinations and utilised as the feedforward input for the legged controller.

C. Resistances

There are many resistances that affect vehicle mobility. Regarding wheeled/tracked vehicles, the compaction resistance \( R_c \) represents the force required to displace the soil in front of the wheel/track. Bulldozing resistance \( R_b \) is the force required to displace the soil that has been pushed above the nominal height of the soil in front of the wheel/track. Gravitational resistance \( R_g \) is the force applied to the vehicle due to the inclined contact of the wheel/track with the soil.

Similarly, these resistances can be applied to a legged vehicle. The compaction resistance can be represented as the draft force required to displace soil in front of a partially sunk leg through the soil in front of it. Bulldozing resistance is the additional force required to displace soil that has been pushed above the nominal height of the soil in front of the leg, though this is minimal for thin legs. In cases of wide legs where soil displacement above the horizontal is found to be more significant, it can be determined as part of the draft force (through top surface inclination at an angle of repose, as discussed in \([35]\)). Gravitational resistance is significant for legged vehicles due to the angled approach of the leg in the soil and the frictional interaction between the leg or footprint and the soil.

A more detailed investigation into these resistances is discussed in the following sections.

1) Compaction Resistance

For a wheeled and tracked vehicles, the calculation of compaction resistance was initially developed by Bekker \([30]\) and further investigated by Wong \([33]\) and others. It can be estimated as follows:

\[
R_c = \left( \frac{k}{n+1} \right) z^{n+1}
\]

(6)

where sinkage \((z)\) for a tracked vehicle is:

\[
z = \left( \frac{W}{A \left[ \frac{k_c}{\phi} + k_{\phi} \right]} \right)^{1/\phi}
\]

(7)

and \(k_c\), \(k_{\phi}\), and \(n\) are soil-specific properties. The calculation for sinkage for a wheel is somewhat more detailed and excluded for simplicity.

However, calculating this draft force for a legged vehicle is significantly more complicated. For a legged vehicle, the force required to move a leg through soil varies significantly depending on the angle of approach the leg is positioned. Therefore, the soil bearing capacity factors \((N_{k}, N_{c}, N_{q}, N_{ca})\) \([29]\) must first be found, then a methodology selected for determining draft force. A number of methods exist for solving for both Terzaghi’s soil bearing capacities and the draft force of a narrow blade/leg through soil \([9,36]\). Each of these methods use a representation similar to Figure 6, which shows the angles and forces of affecting a blade cutting through soil.

The most accurate method for determining the draft force for a narrow blade in a soil where the sinkage is greater than the width of the blade is the Hettiaratchi and Reece method for three-dimensional soil failure \([37]\), which utilises McKyes theory for evaluating the soil bearing capacity factors \([36]\). These methods follow Terzaghi’s universal earthmoving equation \([29]\), which is:
\[ P_{\text{draft}} = \chi g z^2 N_c + C_0 z N_c + q z N_q + C_a z N_{ca} \]  

(8)

where \( z \) is the sinkage in soil of the centre of the footprint [38], \( g \) is gravity, \( \chi \) is the soil density, \( C_0 \) is the soil cohesion, \( q \) is the surcharge and \( C_a \) is the soil adhesion.

This draft force \( P_{\text{draft}} \) is the minimum force required to push through the passive resistance \( R_c \) of the soil in front of the blade at an angle of approach \( \alpha \). Rotating the leg about the toe, instead of translating it, will determine the compaction resistance required to push the soil out of the way of the moving leg.

The equations used to determine the soil bearing coefficients can be found in the Appendix at the end of this paper.

2) **Gravitational Resistance**

The gravitational resistance of a legged vehicle in soil is a significant influence to its trafficability. As shown above in Figure 5, the gravitational influence is the primary element in the determination of soil thrust gained from the foot in soil. Similarly, there is a frictional component of the leg that is influenced by gravity: the friction between the soil and the leg shaft when pushing through the soil.

The same methodology applies for determining the resistance of the leg in soil as was applied to determining the soil thrust of the foot in equation 5, only the leg shaft length embedded in the soil makes up part of the desired area, instead of the footprint length. As a result of this frictional force, angles of \( \alpha \) less than 90° will produce a resistive force against in the direction of forward motion. However, at angles greater than 90°, the frictional force will actually reinforce the soil thrust and provide extra capability.

3) **Active Force “Resistance”**

There is one additional force that must be considered in this investigation. The collapse zone of a soil that forms as a result of the leg moving away from the soil is considered in the active state [34]. This active force supports the moving leg and aids in the soil thrust capability. It is classified as a resistance because it is not affected by slip in the same way that soil thrust is. The active force is modelled as follows [29]:

\[ R_a = \frac{\chi^2 K_A}{2 \sin \alpha \cos \delta} \]  

(9)

where \( \alpha \) the angle of approach of the leg in the soil and \( \delta \) is the friction angle between the soil and the leg. \( K_A \) is the coefficient of active earth pressure and expressed in the Appendix at the end of this paper.

This negative resistance (and hence, additional source of thrust) is an important factor in determining draw bar pull. It must be considered, however, that as the leg rotates through its range of motion, the soil behind the leg will collapse and the effective height of this soil will become increasingly smaller and therefore the effective active force will approach zero. Also, as the leg slips in the soil, this will compress the soil behind the leg and create a passive earth pressure against its collapse, thereby slightly increasing the effective active force supporting the leg. In these simulations, the maximum active force is considered as if there was no collapse of the soil or effective compression due to the slipping leg. However, later developments may incorporate this dynamic feature.

V. **DETERMINING TRACTIVE EFFORT FOR A LEG THROUGH SIMULATION**

In order for a walking vehicle to benefit from this analysis, a control signal needs to be incorporated into the vehicle controller based on the relationships expressed above. Each of the forces that affect the leg’s performance in the soil is shown in Figure 7.

![Figure 7 – Diagram of the forces affecting a leg in soil.](image_url)

From the above diagram, it is noted that the motor torque on the tibia segment \( T_m \) has not previously been discussed. This is also true of the force \( F_b \) applied at the top of the tibia caused by the forward motion of the vehicle. It is the control of both of these values that will allow the manipulation of the leg slip in the soil.

Considering only the forces on the leg caused by the soil, the total force on the leg is determined. These forces are each plotted in Figure 8 throughout the range of motion between placing the foot at AEP until lifting the foot for the next swing phase at PEP.
As shown above, the total force required to overcome the soil in front of the moving leg increases dramatically at higher leg entry angles. This is mostly caused by the draft force required to compact the soil in front of the moving leg. The soil thrust is plotted above without slip, since it is slip that will be determined and fed into the system controller.

The above plot also shows that the overall force on the system is always negative and therefore must require an additional torque and/or force from the leg motors to allow any movement of the leg through the soil. Additionally, shortly after the leg reaches a pure vertical position (when $\alpha$ is 90 degrees), the force required to push the soil forward will become prohibitively high and the vehicle will struggle to create enough force to compact this soil without some level of slip.

The case of only the tibia leg segment is first considered without the influence of any other leg segments/motors or the movement of the vehicle body. In this case, only a motor torque from the joint motor at the top of the tibia is applied to push the top of the leg forward from AEP to PEP. This motor torque will cause a force on the foot of the leg opposite of the soil thrust component, which will only further increase the force needed to overcome the soil resistances and provide forward motion.

This will effectively switch the active and passive earth pressures applied by the soil to the leg. Therefore, the base of the foot will slip slightly and compact the soil behind the leg until there is enough active pressure at the base of the leg to push through the soil in front of the top of the leg. When the soil in front of the leg is then compacted enough to resist any further compaction, the soil at the base of the back of the leg will give way again, causing further slip.

Under this circumstance, any level of torque will cause some level of slip, however, higher levels of torque will give higher levels of slip. Therefore, a leg segment controlled only by a single motor will not achieve the desired goal of providing slip-free thrust through the soil. However, some level of tractive effort will be gained with this method. The exact relationship between motor torque and slip is still under investigation.

A second case is then considered to include the other leg segments and motors, as well as the vehicle body forward motion. This causes a horizontal force at the top of the tibia segment in the opposite direction travel of the vehicle and is simplified to a single force $F_h$, for simplicity at this stage in the research.

The same principle discussed in the first case is applicable in this case. However, the additional force caused by the other leg segments and vehicle body, in conjunction with the motor torque providing forward rotation of the leg, can provide better control of the leg slip in the soil. Given the force at the tibia joint $F_h$, the tibia joint motor torque can be adjusted to maintain a slip-free position of the foot until the draft force required to push through the soil in front of the leg becomes too significant to overcome without some level of slip. Then, similar to the previous case, the slip will cause the active/supporting force to become a passive/resistive force at the back of the foot.

At this stage of the research, the complete mathematical relationships to relate slip to joint torque and body force are still under development.

VI. RESULTS AND FUTURE WORK

At this stage in the research, the simulation environment for determining the forces applied to the vehicle leg by the soil has been completed. This includes the determination of passive and active forces on the front and back of a leg moving through soil. Additional fine-tuning of the active force resistance may later be implemented, but is not required for the research at this stage. Numerous simulations have been run, but more are needed to determine the relationship of foot slip to joint torque and body force.

The joint and leg controllers are still under development, but will utilise the above sinkage and slip models, as well as the visco-elastic joint model. These controllers, once completed in the coming months, will be able to impart more accurate force and torque models on the tibia for effective evaluation of tractive capability of a walking vehicle in a planetary soil. Comparative analysis between varying footprint sizes, differing soils, and changing leg controller algorithms will provide a platform to effectively determine the most appropriate leg controller for a legged vehicle system for planetary (including Earth-based) environments.

Then, a full simulation of the entire vehicle with central pattern generator-based gait controller will provide a realistic re-creation of the testing of a legged vehicle. This will also have the added benefit of easily investigating different footprint and vehicle designs with minimal changes to the simulation environment. This full vehicle simulation will be developed in the Open Dynamics Engine [39] open source physics environment with MATLAB [40] based joint, leg and gait controllers.
VII. APPENDIX

Soil bearing capacity factors as developed by Hettiaratchi and Reece [37]:

\[ N_k = \frac{(\cot \alpha - \cot \beta)}{\cot \phi} \]  \quad (10)

\[ N_c = \frac{1}{\sin \phi} \]  \quad (11)

\[ N_q = \frac{1}{2} N_k \]  \quad (12)

\[ N_{ca} = \frac{1}{\cos \phi} \]  \quad (13)

where \( \alpha \) is the entry angle of the leg in the soil, \( \beta \) is the angle below the horizontal where the soil slip line is formed, \( \phi \) is the soil friction angle and \( \delta \) is the friction angle between the soil and the leg.

Coefficient of active earth pressure [29]:

\[ K_A = \frac{\sin^2 (\alpha + \phi) \cos \delta}{\sin \alpha \sin (\alpha - \delta) \sin \phi \sin (\alpha + \phi)} \]  \quad (14)

REFERENCES


ABSTRACT

Previous work on shape recognition from whisker-like sensors has taken two approaches. First, static analysis from beam theory has been used to estimate contact points. Second, dynamic analysis has estimated contact points by considering vibrations set up on contact. This paper proposes additional methods which utilize full time-series data from whiskers. Full generative models may be intractable but approximated by discriminative classifiers. Early results show that object location and velocity may be discriminated simultaneously.

I. INTRODUCTION

The recognition of shape is an useful sub-task of robot navigation and other cognitive tasks. For example, commercial robot vacuum cleaners navigate using point collisions only (e.g. [1]) and could be improved by locating classes of objects in space rather than identical-looking points. The use of tactile whisker sensors for mobile robots has been explored by several authors [2], [3], [4], [5] and has advantages of functioning in foggy environments and of being covert.

The mechanics of whisker behavior are described by the theory of beams, as used in Civil Engineering [6] and briefly reviewed below. On making contact with an external object, whiskers bend, giving rise to measurable curvature at their bases. Whiskers also vibrate when they are stuck by object contact, as described by the theory of mechanical vibration analysis [7]. Static and dynamic analysis of beams are often considered as different fields of Engineering, and this divide has continued into whisker sensor research, yielding two distinct methods for shape detection based on the two theories.

This paper reviews static and dynamic whisker analysis then considers forward simulation models of whiskers and inverse models for obtaining single contact points from whisker sensors. It describes new generative and discriminative time-series approaches which may be capable of making inferences about shapes and motion rather than single points, and gives results from a preliminary discriminative implementation.

II. WHISKER MECHANICS

II-A. Static analysis

The beam theorem of Euler and Bernoulli. For a pure bending moment (i.e. pairs of forces $M$ at the ends of the beam which rotate as the beam bends, and the center of mass pinned), a small segment of a beam of Young’s modulus $E$ and cross-sectional moment of inertia $I$ bends circularly about a center of curvature at radius $R$, with

$$\frac{1}{R} = \frac{M}{EI}.$$  

(A standard proof is given in the appendix). For example, one of the bending pair could be a contact force when a whisker is bent by an external object; the other bending force and the pin are provided by the base of the whisker. The area moment of inertia $I = I(x)$ is that of the cross-sectional disc, about the neutral axis. For homogeneous non-tapered whiskers it is constant along $x$ but in general it could be a function of $x$.

Rather than work with radii of curvature, it is often useful to work with spatial derivatives. By geometry alone, we have for a circle of radius $R$,

$$\frac{1}{R} = \frac{d^2v/dx^2}{(1 + (dv/dx)^2)^{3/2}}$$

where $v$ is orthogonal axis to $x$ in the whisker’s bending
plane. For small deflections \( dv/dx \) this reduces to
\[
\frac{1}{R} \approx \frac{d^2v}{dx^2}.
\]

The Double integration method may be used to find the shape of a bending whisker if we are given the moments applied at each point along it. (There is usually a single contact point, which excerts a variable but effectively unlimited force to oppose the whisker pushing into it. Whiskers are small compared to most external objects so we ignore the effect of the opposite force acting on the object.)

\[
v(x) = \int \int \frac{M(x)}{EI} dx^2 + c_0
\]

The constant \( c_0 \) is set by the boundary condition that the whisker base is fixed in a known position.

Whiskers typically make contact with objects at a single point only, due to their curvature. If a contact force \( W \) is applied at distance \( C \) from the whisker base then the moment experienced at each point \( x \) along the whisker is

\[
M(x) = \begin{cases} 
W(C - x) & : x < C \\
0 & : x \geq C
\end{cases}
\]

Inserting this moment form into the double integration equation shows that the whisker bends cubically up to the contact point, and is linear afterwards:

\[
\frac{EI}{W} v(x) = \begin{cases} 
\frac{1}{4}C^2x^2 - \frac{1}{8}x^3 & : x < C \\
\frac{1}{2}C^2x - \frac{1}{6}C^3 & : x \geq C
\end{cases}
\]

The \( x < C \) condition describes the shape from the base of the beam to the contact, and a small part of this region close to the base is typically observable by sensors. The cubic equation has two unknowns, \( W \) and \( C \), so requires two observations to solve. These could be observations of the position \( v(x) \) at two points, or its gradient and curvature,

\[
\frac{dv}{dx} = \frac{W}{EI}(C - \frac{1}{2}x^2)
\]

\[
\frac{d^2v}{dx^2} = \frac{W}{EI}(C - x)
\]

or other variables from which any of the above may be derived. Note that the position and gradient (but not curvature) are always zero at the base itself, so are not useful measurements. Kaneko [2] used base torque and an angle moved between first contact with an object and at its final location as such a measurement pair. Similarly, Birdwell et al. [3] use the base angle and torque; or alternatively the first temporal derivatives of the same.

If a series of two-variable data points are received, and it is assumed that the whisker is static at each, then a collection of contact points may be located, and used as the basis for shape recognition. In addition, each contact point also contains the information that there is no object in any of the space swept out by the whisker before reaching its static cubic spline. In particular this is useful for distinguishing smooth curved surfaces from sharp edges, as discussed by [4] and [5].

**Figure 2.** Construction used to derive spring-mass model from the beam theorem.

In practice, for real-world whisking speeds, the whisker is not static at each point, but carries kinetic energy which affects its trajectory. One approach to shape recognition is to estimate the noise level in sensor measurements from their equilibrium values introduced by these deviations from equilibria. The estimates of the contact point coordinates \( (C, v(C)) \) may then be computed together with error bounds propagated from the sensor uncertainties. There appears, qualitatively, to be a trade-off between speed of whisking (and thus the number of contact points gathered) and the accuracy of the points due to kinetic effects: a whisker driven infinitely slowly would be perfectly accurate. However this approach appears to be throwing away information, as we may have good models of the whisker dynamics and thus be able to account for the ‘noise’ in terms of the actual dynamic state of the whisker over time.

### II-B. Dynamic analysis

Some previous work has considered the use of simple dynamic models. Kaneko [8] analyzed a massless beam with a large mass at the tip. When such a beam strikes a contact point, it oscillates in two distinct phases. First – with the beam in contact – the region from the contact point to the tip oscillates with fundamental frequency

\[
\omega(C) = \sqrt{\frac{k(C)}{M}} = \sqrt{\frac{12EI}{ML^3(4 - C(3 - C)^2)}}
\]

where \( k \) is the effective spring constant. In the second phase, the beam is not in contact, and oscillates at the natural frequencies of the whole beam. Transitions between the phases may be observed in the base torque, and the contact location deduced from them, under the important assumption that all motion is at stable natural frequencies. Kaneko [8] then proceeded to a non-massless beam, deriving similar equations to estimate \( C \).

In particular this method assumes that there are no transient components due to the initial contact. A low-pass filter was used by Kaneko to remove this ‘noise’ – which may in fact contain useful information. As with the previous static methods, there is no notion of uncertainty in the estimate. Beams without large masses at the tip often produce ambiguous, bi-valued estimator functions, and the wave equation used in the analysis assumes only
small displacements, which are not valid for large deflections as often experienced by real whiskers. The method must also wait for a complete cycle of both phases to begin making estimates. As with static methods, it is unable to handle moving objects.

Non-generative dynamic models have searched for empirically useful features in the frequency domain of strain signals to predict contact surface properties, usually with regard to classifying texture rather than contact location. In the texture case, it has been noted [9] that the ‘ringing’ transients produced as the whisker makes and leaves contact with the surface contain useful information. This is the same part of the whisk signal discarded as ‘noise’ by Kaneko.

III. LIMITATIONS OF CURRENT MODELS

We have reviewed various methods for gaining information about contact location. Static analysis assumes that each data point is at static equilibrium, and discards dynamic effects as noise. State-of-the-art dynamic analysis assumes that all oscillations are at stable natural frequencies, and discards transients as noise; it also only makes use of one sensor. Heuristic classifier-based approaches have found information in transients but do not fuse it with all other available information. All previous methods make point estimates only, not handling uncertainties in various sensors or allowing the possibility to fuse information from multiple sensors. No previous approach has handled moving targets. Other dynamic effects ignored by previous approaches include surface properties such as collision elasticity and surface friction. We thus propose a class of generative time series models capable of accounting for all these effects. We will outline the form of these models, then consider computational inferential and discriminative approximations.

IV. FORWARD SIMULATION

We have constructed a simulation model of a whisker as a series of masses on rotational springs, based on the Open Dynamics Engine (ODE; www.ode.org), as shown in fig. 1. Such systems consist of more than two bodies so lack analytic solutions. The number of segments, $N$, is an important parameter as it controls the complexity of the simulation (and later, of inference) and affects the spring parameters used. Spring constants can be derived as follows. Consider approximating the curved beam with a series of length $L$ segments $\{s_i\}_{i=1}^N$ as in fig. 2. The continuous curvature at segment $s_i$ is now replaced by deflection angle $\theta_{i,i+1} = 2\beta$ at each segment. (We write $\theta_{0,1}$ for the angle between the whisker base on the robot and the first segment.) By geometry:

$$R \sin \beta = \frac{L}{2}$$

for small $\beta$:

$$R \beta = \frac{L}{2}$$

$$R = \frac{L}{\theta_{i,i+1}}$$

![Figure 3. Base angle $\theta_{0,1}$ (solid) and curvature $\theta_{1,2}$ (dashed) during contact with a moving object.](image_url)

Substitute in from the beam theorem:

$$M = \frac{EI}{L} \theta_{i,i+1}$$

This is now in the form of a rotational spring equation, with

$$k = \frac{EI}{L}$$

The springs also have damping coefficient $\gamma$. This may be set using the $Q$-value of an isolated segment, which for light damping is the number of oscillations performed during each loss of $1/e$ of the oscillator energy:

$$Q = \frac{m}{\gamma} \sqrt{\frac{k}{m} - \frac{\gamma^2}{4m^2}}$$

When $Q$ is zero or imaginary the system is critically or heavily damped and no oscillations occur. For light damping (small $\gamma$):

$$Q \approx \sqrt{\frac{m}{\gamma}}$$

The segment $Q$-value could be computed analytically from beam theory, measured empirically from a real whisker segment of length $L$, or – for simulations – simply prescribed. In particular its is often useful to prescribe critical damping.

Collision detection is handled by ODE, as are constraint forces to hold joints together. However we have found that such whisker systems are extremely sensitive to numerical errors and require small time steps to simulate correctly, and we note empirically that there appears to be a monotonic relationship between the joint Q-value and the number of steps per simulation second to avoid error; and between the number of whisker segments and the steps per simulated second. The problem can be reduced to some extend by introducing linear spring elements alternating with the rotational springs. This reduces the magnitude of the required Lagrangian constraint satisfaction forces which are the principal cause of numerical instability. We are considering the design...
of a custom ODE joint which combines rotational and linear springs to reduce error further.

The simulation allows us to record sensor-like observations, for example of base angle \( \theta_{0,1} \) and base curvature, approximated by \( \theta_{1,2} \). It also allows for small random noise forces to be applied at all segments, as would be experienced due to wind and self-motion of the robot body.

Figs. 4 and 3 shows the screen-shots and simulated base angle and curvature for a passive whisker being stuck by a moving square object, whose motion is perpendicular to the line of the whisker. The first oscillation includes the contact period; the others occur after the object has passed by the whisker. The shape, contact location and motion of the object affect the time series during the contact period and the subsequent oscillations. (Real whiskers have a much lower \( Q \) than simulated here; however this requires much longer computation time to simulate as discussed above, so for development purposes a low \( Q \) is used.) Note in particular that in step 5 of fig. 4 the whisker has received large momentum from the object and continuous to move away from it, before springing back to make contact a second time in step 6.

V. INVERSE TIME-SERIES INFERENCE

Fig. 5 shows a slice of a dynamic Bayesian network model for the whisker system. It consists of segment states \( s_1 \) which each comprise current position and velocity; contact object state \( c \) which also comprises position and velocity along with other object properties such as shape, rotation, surface elasticity and friction. The observed base angle \( \theta_{0,1} \) and curvature \( \theta_{1,2} \), and small noise nodes representing additional forces applied to the segments and contact object. Two time slices are shown horizontally; in reality the model consists of many such slices over the simulation time. The transition relation for the contact object is such as to update its position according to its previous velocity; and allow for changes in velocity according to some prior, as well as any noise forces acting on it. The transition relations for the segments are complex as they include constraint satisfaction forces, but may be modeled implicitly by ODE physics simulation steps. Similarly the observations are obtainable from such simulations.

Drawing samples from the model is simple, and effectively amounts just to running the ODE physics simulation, with appropriate random noises added. A much more challenging task is to infer the most probable explanation of a given observed time series. Exact inference is clearly intractable due to the loopiness, continuous-valued nodes, and highly non-linear transitions. However approximate inference by tracking of hypotheses subsets (i.e. particle filtering) [10] may be possible. The whisker domain is especially interesting as the number of segments in the model may be varied, trading off physical accuracy for the number of variables (and hence dimensions) represented in each particle. After drawing random noises, the transition steps can be performed deterministically by calling out to the ODE physics engine to simulate the evolution of each particle.

VI. DISCRIMINATIVE APPROXIMATION

We have shown how to set up inference from whiskers as a dynamic Bayesian network based on physical simulation, and suggested that approximate particle filtering inference may be useful. However it is an open question whether any such inference is feasible in real-time. An alternative approach is to construct and train discriminative functions – such as neural networks – mapping directly but approximately from the time-series data to the object state. As well as potentially replacing full inference models, such discriminative methods may be useful as heuristics within them: for example by suggesting new hypotheses to ‘inject’ into particle filters [11].

VI-A. Template-based discrimination

To demonstrate that at least some useful information is quickly obtainable from the time series by discriminative methods, we here present baseline results for estimating the contact distance \( d \) to a moving object using the curvature, \( \theta_{1,2}(t) \), series only. The object is the square box moving at a vertical fixed velocity as shown in fig. 4. The only variable is the horizontal distance of the box from the whisker base.

Fig. 6 shows examples of curvature series for three contact distances. (Time is unit-less but steps may be thought of as milliseconds.) The series make reasonably smooth shape transitions as the contact distance varies. Series were obtained for distances of 1.5cm to 3.0cm, in 1mm steps. (Contacts closer than 1.5cm are difficult to simulate and were discarded.) The first, last and
central (2.3cm) points were used as template models. The remaining points’ contact distances were then estimated as follows. Define the responsibility $\lambda_o(i)$ of template $m_i(t)$ for an observed series $o(t)$ as the Gaussian

$$\lambda_o(i) = \frac{1}{Z} \exp \left(-\frac{\sum_t (m_i(t) - o(t))^2}{2\sigma^2}\right)$$

where a (unit-less) variance of 5 was chosen by hand as a useful value and $Z$ normalizes the responsibilities over $i$. Responsibilities for the observed contact distances are shown in fig. 7. Taking the average of the template models’ contact distances – weighted by the responsibilities – as an estimate of contact distance, gives the model estimates shown in fig. 8, which have a mean error of 1.2mm.

In a second experiment, both the contact distance and the object vertical velocity were varied together. Nine templates were used, at combinations of minimum, medium and maximum of the two variables, and the same methods as above was applied to estimate both parameters simultaneously. The estimated contact distances across all velocities are shown in fig. 9 and the estimated velocities across all contact distances are shown in fig. 10.

This is the first published result to our knowledge on simultaneous contact distance and object velocity estimation – all previous methods have considered static objects only. It is intended as a proof of concept to show that useful information is contained in whisker time series. However the degradation of accuracy using the simple template method when a second variable is included suggests that this simple method may not
VII. DISCUSSION

To build and test both generative and discriminative models, it is useful to work with a series of tasks of increasing difficulty. In this case, difficulty mostly corresponds to the size of the parameter space. In addition to reducing the number of segments used in the generative model, the dimension of the contact object may be restricted. A simple scenario is to assume that the only type of object in the world is a frictionless, perfectly elastic plane, moving towards the whisker at known constant velocity. Such an object has only three dimensions: initial shortest distance from the whisker base and rotation. Alternatively, the object could be stationary and the whisker moving rotationally into it. Simple shapes such as squares and circles are a little harder, having extra configuration dimensions.

Many interesting collisions in the real world involve three physical dimensions rather than the 2D collisions produced by simple shapes. For example the act of ‘plucking’ a whisker (like a musical instrument string) involves pushing the whisker back but simultaneously lifting the object upwards, away from the whisker. Rather than use a full 3D simulation (which massively increases the hidden state space size), a close approximation is to restrict the whisker motion to the horizontal plane while allowing contact object to move in 3D space.

The linear template model demonstrated here is a crude form of discrimination, and more advanced template-based discriminators exist such as classification on Principal and Independent Component strengths. Template models in general are not invariant to shifts.
in the time domain, so a different class of Markovian discriminators may be useful instead. These include Hidden Markov Models and recurrent neural networks, which may be trained to filter time-series data into class labels online. Such models have similar dynamic network structure to the full generative model, but are acausal, more easily trainable, and faster to run. In particular, the structural similarity may allow some of their parameters to be set or constrained by knowledge about the generative model.

Previous whiskered shape recognition systems yield only point estimates of contact locations. In contrast generative methods (and discriminative approximations to them) can produce a posterior belief over the configurations of whole objects in the world. These beliefs may then be used as input to higher-level systems. For example localization and mapping tasks [10] and higher-level object recognition from parts [12] are improved by allowing for uncertainty. Uncertainty may be reduced by fusing information from multiple whiskers.

VIII. SUMMARY

Previous approaches to whiskered shape recognition have used only limited subsets of information available from the whiskers. They have further neglected aspects such as the motion, elasticity, and friction of objects. Their outputs have been point estimates of contact points with no posterior distributions, and they have not taken account of uncertainties in sensors or noise in the environment such as wind and self-motion effects.

We have shown how to set up the task as a full generative temporal model, which may in theory account for all available time-series data, and make inferences about the additional object and environment properties. Unusually, the model makes use of an external physics engine to make deterministic state transitions once simple noise terms are drawn. We have shown how the whisker beam can be approximated in this model by a series of masses on springs and discussed some practical simulation issues, in particular noting an empirical trade-off between the computational simulation time and the simulatable stiffness of the whiskers. We propose the use of particle filters for generative inference and for calibrating and testing discriminative methods. Once trained, the latter may in turn suggest new particles to inject into online particle filters.

The utility of full time-series data in recognition was demonstrated by our results showing simultaneous recognition of contact distance and velocity. All previous approaches have assumed static objects. Our result was achieved by a simple template-based discriminator, but suggests that more advanced discriminative and generative models will also be useful and possibly more accurate.

The simulation is available under GPL license from sourceforge.net/projects/freebots.

Figure 11. Construction on cross-section of a beam, used to derive the beam theorem.

APPENDIX

Consider fig. 11 which shows the bending. For pure bending, the forces rotate with the beam as it bends so they are always perpendicular. The top of the beam experiences negative strain (due to compression stress) and the bottom experiences positive strain (due to tension stress). There will be some ‘neutral axis’ IOG whose length is unchanged. ST is a short, thin slice at height y from the neutral axis. Consider the strain \( \varepsilon_{ST} \) at ST:

\[
\varepsilon_{ST} = \frac{\Delta ST}{ST_0} = \frac{ST - IG}{IG} = \frac{(R - y)d\theta - Rd\theta}{Rd\theta} = -\frac{y}{R}.
\]

Assuming Hooke’s law holds for the stress \( \sigma_{ST} \),

\[
\sigma_{ST} = -\frac{Ey}{R}.
\]

Consider a thin horizontal slice around from ST to ST + dy, having cross-sectional area \( dA \), and let the cross-sectional area of the whole beam be \( A \). As there is no axial load, the total force horizontally through the beam must be zero (because the beam is static). So

\[
F = \int_A \sigma dA = 0
\]

(From this it follows by symmetry that the neutral axis is in the center of the beam.) The strain at \( ST' \) is similarly the tensile

\[
\varepsilon_{ST'} = \frac{+y}{R}.
\]

Imagine that the beam is split into two at JOK. Across ST there is a compressive force

\[
F_{ST} = \sigma_{ST} dA = -\frac{Ey}{R} dA
\]

acting leftwards on the RHS of the beam and at \( ST' \) there is a similar tensile force \( F_{ST'} = \sigma_{ST'} dA \) acting rightwards on the RHS of the beam. (And vice versa for the LHS of the beam.) This pair of forces is equivalent to moments about \( O \) on the RHS, and the total moment on the RHS at location JOK is given by summing them along JOK:

\[
M = \int_A y \cdot \frac{Ey}{R} dA = \frac{E}{R} \int_A y^2 dA = \frac{EI}{R}
\]

This must be equal to the original bending moment \( M \) applied to the RHS and the theorem follows.
IX. REFERENCES


Programmable Differential Brakes
for Passive Haptics

Yaroslav Tenzer, Manish Patel, Brian L Davies, Ferdinando Rodriguez y Baena
Department of Mechanical Engineering
Imperial College London
London SW7 2AZ, UK
ytencer@ic.ac.uk

Abstract—Passive haptics pertains to the ability of some systems to constrain the end effector motion induced by a human operator without being able to move autonomously. Such systems are particularly suitable in the context of man-machine interfaces (e.g. medical robotics applications), where any unwanted motion (e.g. in the event of a system malfunction) could have serious repercussions. While several systems have been proposed which address the challenge of passive haptics, most attempts are rather highly complex or have been only partially successful, both in terms of force feedback bandwidth and degree of programmability. This paper outlines our work on a novel programmable braking system, which is widely applicable to most passive haptics applications and benefits from a simple design, theoretically infinite positional resolution and the ability to generate stiff collision forces, without the need for any explicit sensor measurements. Results of a preliminary implementation and of a simple 2 Degrees-Of-Freedom (DOF) Revolute-Revolute (RR) manipulator incorporating the programmable brakes are also given. These include performance measures for the joint as well as figures describing the ability of the 2-DOF prototype to constrain the end effector motion to a plane.

Index Terms—Passive Haptics, Medical Robotics, Programmable Brakes

I. INTRODUCTION

Robotic systems can be programmed to have a passive role (e.g. programmed to react rather than to act) or can be passive by design (e.g. prevent motion rather than initiating motion). While the former can pose a safety threat during a malfunction, passive systems are inherently safe, as none of the components in the assembly are capable of autonomous motion. As such, a design philosophy based on passive components, such as brakes and clutches, is of particular relevance to the development of man-machine interfaces, where the robot works in close proximity to the human operator. Specifically, “passive robots” are well suited to medical robotics applications, for instance in the area of robot assisted surgery [5], haptics for training and diagnostics[4], where safety by design is of utmost importance.

A steering mechanism using Continuous Variable Transmission (CVT) based on the concept of “human moves - machine directs” was first implemented in [6]. This five-linkage mechanism, alongside a special purpose control algorithm, directs the movement of the end effector to constrain a user generated motion along a pre-programmed path (trajectory enhancing approach). This is achieved through a set of controllable pulleys, the reduction ratio of which can be adjusted in real-time by changing the radii of each wheel. The system, however, is incapable of providing stiff feedback (e.g. collision with a virtual “hard wall”) because it lacks a locking mechanism (e.g. brakes) to complement the programmable constraints.

Another passive trajectory enhancing robot named P-TER is implemented using brakes and clutches. It was designed to guide the motion of its end effector along a programmable path, given an arbitrary input force [3]. The five-bar mechanical implementation allows 2 degrees-of-freedom (DOF) movement. It uses brakes to remove energy from the system and clutches with a differential gear arrangement to transfer energy between the joints. An advantage of this system over those previously mentioned is that, while providing programmable constraints, it is able to reproduce the high forces occurring during collisions between hard surfaces. The advantage of having a locking mechanism, however, comes at a price: a force sensor is needed to sense the force direction applied by the human operator, which is needed to identify when the breaks should be engaged and disengaged. The need for a six axes load cell adds to the size and complexity of the device and is thus a clear disadvantage of the proposed architecture.

Additionally, a passive haptic device named Programmable Constraint Machine (PCM) was implemented using “non-holonomic” elements [2]. A basic example of a nonholonomic element is a steering wheel. This configuration enables constrained motion in 2D space (i.e. in a plane). The wheel is free to roll in response to forces applied to the handle by the user. When in contact with a virtual wall, the wheel is simply steered tangent to the wall, which prevents further motion in the direction normal to the wall without affecting tangential motion. The authors suggest that additional wheels can be included in the design to improve the ergonomics of the handle, but devices based on this approach are limited to 2D planar constraints and are thus not suitable for applications in 3D space.

The most successful implementation of a truly passive robot is PADyC, which allows constrained motion of a tool...
with respect to a pre-planned task in the context of robotic assisted surgery [5]. The system is based on a number of patented mechanical joints. Each of them consists of two “freewheels” and two electrical motors. The implementation enables programmable joint motion in 4 independent states: free motion in both directions, constrained motion in one direction only (clockwise or counterclockwise) or no motion at all. A clear advantage of the PADyC system is that it can limit the rotating speed of the joints and therefore guide the end effector in a particular trajectory, while being incapable of any autonomous motion (and thus safe by design). The locking mechanism itself has special disks with guiding slots filled with metal balls. The shaft rotation is prevented in one direction because the balls are locked between the outer and inner wall, while rotation in the other direction is not restricted. The implementation of this technology is very robust due to the use of two disks and two motors. To the best of the authors’ knowledge, PADyC is the only passive system built with Programmable Brakes (concept explained later) with infinite movement resolution.

As can be seen, passive robotics can be divided into three main areas according to function; first, passive guidance which steers the movement of the end effector through pre-programmed paths but can not simulate stiff wall feedback. Second, the movement limiting approach which is implemented by some form of joint locking mechanism employing brakes. Joint locking does limit the motion but also requires the inclusion of a force sensor to know when the handle was forced in a different direction and therefore the joints should be unlocked. The force sensor with associated electronics is expensive, which is a clear disadvantage of passive devices belonging to this category. Third, the “golden standard” in passive robotics is provided by the mechanism for guiding and locking implemented in PADyC, which is very robust and truly programmable. This clutch-based system, however, is complex both in terms of control and assembly and results have shown that PADyC is also particularly sensitive to time-dependent phenomena such as wear.

Thus, this paper presents a novel joint design which belongs to the third category of passive robotics, but has much simpler mechanics and does not require the use of a force sensor. The novel concept will be presented next, followed by some preliminary results of a 2 DOF planar RR (revolute-revolute) manipulator employing the our proposed programmable differential brake concept.

II. PROGRAMMABLE DIFFERENTIAL BRAKES - THE CONCEPT

A brake is a mechanical component for slowing or stopping motion by reducing energy between moving parts, mostly by contact friction. The disadvantage of ordinary brakes in a rotary joint is that they affect rotation equally in both directions (i.e. clockwise and counterclockwise) when actuated and thus have only two basic states: free motion in both directions and no motion at all. As it was already proposed [7], a braking mechanism can be enhanced with additional functionality (i.e limit the movement in one direction) by means of the basic concept illustrated in Fig 2, where a wheel with special teeth and a clutching system are connected to a mechanical stopper. If the stopper is actuated (by turning off the current on the coil and allowing the spring to push the stopper onto the tooth), the teeth will prevent rotation in one direction while movement in the other direction is not affected. A joint with two of these locking parts (Example in Fig 1) is a Programmable Break with four states: free rotational movement when both clutches are inactive, one-directional motion when only one clutch is active and no movement when both clutches are active. The main disadvantage of this system pertains to the “discrete” nature of the constrained motion, the resolution of which is directly proportional to the number of teeth in each wheel.

Based on this principle, a novel patent-pending programmable brake is presented in Fig 3. As can be seen, the system is very compact and the braking mechanism (which is concealed by a cover as a patent is currently being filed on the concept) fits entirely on the joint itself. The proposed brake architecture is capable providing 4-state programmable motion with the added advantage of infinite resolution and no need for an additional force sensor. The 4-state behaviour is achieved by means of two electronic components, which can be activated independently or together to achieve the desired constraint. A custom built mechanical interface is also included in the design to sense any external
forces/torques applied on the joint, which enables the brake to be engaged/disengaged according to the actions of a human operator, without the need for explicit force sensing.

In the current prototype, the locking stiffness of the design depends on the supplied current and infinite stiffness can be achieved, providing that infinite current is available. In addition, the design is based on a revolute joint, but can easily be modified for application to prismatic joints or linear actuators as well.

A. Materials and Results

In order to find the maximum reaction torque which the system can exert on the human operator, a number of experiments with different currents were performed. During the experiments, the brake was activated and a variable mass was applied at the linkage end-point until the moment when joint slippage occurred. The moment on the joint and the applied current were recorded and the results can be seen in Fig 4. A linear regression model was applied to the results and a correlation coefficient of $r = 0.96$ suggests a strong linear correlation between stall torque and current applied. Our preliminary results show that, for the current prototype, a maximum torque of $10.2 \text{Nm}$ is achieved with a current of $5 \text{A}$. This is by no ideal, but several simple changes to the hardware would result in a substantial drop in current requirements for similar performance characteristics. These will be addressed during the course of future work.

Another important characteristic of a brake is its timing bandwidth. The deactivation timing was measured using the following procedure; the brake was actuated and a weight at the end was applied. A custom program written in Labview was then employed to measure the time between brake deactivation (manually) and actual movement of the link assembly. A number of experiments were performed with different actuation currents (three measurements per different current value), the average of the measurements per current value can be seen on Fig 5. An average deactivation time, which was found to be normally distributed and independent of current, was found to be $160 \text{ms}$ ($6.25 \text{Hz}$) with a standard deviation of $4.51 \text{ms}$.

B. Discussion

A novel programmable brake was described. In addition to standard brake functionality it provides the ability of locking the movement in only one direction. The mechanism itself is not disclosed due to the application of a patent that has yet to be granted. The stiffness experiment results suggest a very poor mechanical implementation of the current prototype while the novelty functionality of the brake has been shown with the expected advantages. The linear results of the torque vs. current experiment suggest that that break can be used to provide a constant resistance torque. Additional experiments and design improvements will be needed to achieve better results.
III. 2 DOF MECHANISM PROOF-OF-CONCEPT

In order to assess the performance of the new programmable brake mechanism within the wider context of robotic control, a typical 2-link RR manipulator was built and is shown in Fig 6. Each joint consists of the new type of electric brake and a rotational potentiometer for position feedback (1). The position sampling and the actuation brakes control are performed through an NI-6009 sampling card (3) with a custom written program in Labview 8.0 (4,5). An additional, custom developed current driver is located between the card and the brake (2). The movement analysis was performed using a force manipulability ellipsoid (FME)[1]. As can be seen (Fig 7), the movement of the end effector $P$ is related to the speeds of the joints $(\dot{\theta}_1, \dot{\theta}_2)$ and can have eight movement scenarios. The first four are along the two dividing lines, each line passes through the end point and represents the potential movement scenario in case of one of the joints is locked. The four other potential movement areas (1,2,3,4) are the areas between the lines, which represent the moving areas when both joints rotate. Haptic devices provide force feedback and it is common to assess a haptic device on a virtual-wall, which is a virtual constraint which forces the end effector to lyer on or above a predefined plane in 3D space. The results of a virtual-wall experiment performed with the 2DOF concept demonstrator is shown in Fig 7, where the end effector $P$ is very close to the virtual-wall. In this case, it is obvious that the movement of the end effector into areas 1,2 and 4 would result in wall penetration, therefore only motions of $P$ in the direction (or along the borders) of area 3 should be allowed. To implement such restriction, the following joint limits should be programmed on the brakes: $\dot{\theta}_1 < 0$, $\dot{\theta}_2 > 0$. As a point of note, while enforcing a “no-go” area successfully, this restriction would not allow smooth wall following, which could only be achieved if the end point moved tangentially to the virtual-wall and therefore move into areas 2 or 3.

A. Materials and Results

The current joint design includes a rotational potentiometer for position feedback. To measure the position noise of the sensors, the joints were locked and the angular position together with the end point position was sensed and recorded. It was found that the angular positional RMS (Root-Mean-Squared) error of 1.12 degrees for the first joint and 0.81 degrees for the second joint (i.e. noise in the reading with zero mean). The corresponding positional inaccuracy of the end point has RMS error of 3.08mm.

A simple wall following algorithm was implemented to try the prototype and to verify the performance of the joints. During the experiment a user was asked to try and penetrate a horizontal wall at $y = 0$ in a number of points. The control algorithm locked the joints as soon as the wall was penetrated avoiding additional movement in the direction of the wall while still allowing movement away from the wall. The position of the end-effector was recorded and is shown in Fig 8. The average penetration depth was found to be 4mm and the maximum penetration was 12mm.
B. Discussion

It was found that the chosen feedback mechanism using rotational potentiometers without any filtering of the signal to reduce noise affected the positional accuracy of the joints and the position accuracy of the end point. The wall penetration algorithm was implemented and limited the penetration of the end point. Unfortunately, due to the positional sampling inaccuracies, the activation of the brake was not permanent and disturbances occurred, which allowed additional penetration of the virtual-surface, thus, improving the positional sampling accuracy will improve the performance of the device.

IV. Conclusion

Passive robotics has great potential in medical applications where safety has the highest priority. Developed passive prototypes lack in simplicity and therefore are hardly found in final applications. This paper presents a new type of programmable brake for passive haptic devices. Due to its novel (patent pending) design, the brake can be programmed in four states (no motion, motion in one direction (clockwise or counterclockwise), and free motion in both directions), has a simple structure and is cost effective to implement and assemble. The current design is based on sub components, which can provide only low stiffness feedback (with maximum torque of 10.2 Nm). To try the novel programmable brakes, a 2DOF planar RR mechanism was built and a virtual-wall restriction control algorithm was implemented. The experiment shows that the end-point was successfully constrained to lie on or above the virtual-plane.

Due to the novel programmable braking concept, the dual-link design has no force sensors and is therefore much simpler than any other implementation available in the literature. The novel brakes can thus be used in a mechanism where the safety factor is critical and a passive mechanism should be implemented to produce restrictions. In addition to a standalone system, the brakes can also be integrated into a robotic system as an additional safety layer, which can stop the system if the primary controller fails.

V. Acknowledgements

This research has been partially supported by a grant from the Anglo Israel Association.

REFERENCES

A Biologically Inspired Fingertip Design for Compliance and Strength
Craig Chorley, Chris Melhuish, Tony Pipe, Jonathan Rossiter and Graham Whiteley

Abstract—The future of robotics is in being able to successfully make the move out of the factory and into the unstructured and uncontrolled world in which we live. Such an environment places great emphasis on our ability to sense and interact with the world around us. Touch is a physical interaction and as such it places great importance on the mechanical properties of the skin. The glabrous skin of the palms of our hands is both highly compliant and strong in grip and manipulation of objects. Current robot skin designs are generally based on the use of a soft silicone, with a layer of mechanical strength. This paper highlights some key features of human fingertip physiology and presents a biologically inspired fingertip design that achieves both high levels of compliance and mechanical strength.

I. INTRODUCTION

Robotics research has shown the benefits of soft contact skin materials for robotic hands and fingers[1][2]. Most designs in the research of robotic hands and tactile sensing fingers have concentrated on the use of a soft rubber skin, in the order of between 10 and 40 on the Shore Hardness A scale, of different shapes and forms, see examples: [3][4][5]. The use of a soft rubber skin offers some level of mechanical compliance, a high co-efficient of friction for grip, as well as some practical protection from physical impact and shock[1]. If the design includes embedded tactile sensors, these may also benefit from impact protection and from the natural compliance of the rubber to conform to shapes and features. However, it can be seen that a soft rubber design only offers a compromise where the chosen level of hardness of the rubber is a trade-off between its compliance and its mechanical strength. The softer the rubber the lower resistance it has to shearing forces, making it a poorer gripping, lifting and manipulating surface, as the soft rubber skin stretches and deforms under these forces, as well as being more susceptible to wear. The harder the rubber material the less compliance the skin has to conform to surfaces, shapes and features, reducing its cross-sectional area of contact and its protection against physical impact and shock. Although there has been various published work on considering contact skin for robotics, to the authors’ knowledge, none have looked to address or resolve these issues of mechanical strength and compliance. There is a need to recognise that the physical nature of touch places great importance on the mechanical properties of the skin. It is surprisingly hard to find published work in the areas of Physiology, Perceptual Psychology, Biomechanics or similar that highlights how the mechanical properties of the structure of our fingertip skin effects our ability to sense and manipulate, when clearly there is an important relationship going on. In highlighting some key features of human fingertip physiology, this work introduces a biologically inspired fingertip design to demonstrate how such features could be employed in robotics to gain some of the desirable properties of high compliance and mechanical strength.

II. THE HUMAN EXAMPLE

Looking at some key mechanical structures of the human finger skin it is possible to recognise how a biologically inspired design could benefit from a similar structure.

Glabrous skin is the hairless skin found on the palms of the hands and soles of the feet. It consists of a layered structure of the thin, hard, ridged outer epidermal layer surrounding a softer dermal layer and very soft inner core of subcutaneous fat, Fig. 1. The hard outer epidermis layer is less than 1mm thick and so it is very flexible but gives a strong resistance to shear, stretch and wear. Inside the fingertip, the finger pad structure below the bone is mainly made of fat cells, becoming the dermis layer on outer edges that border with the epidermis. Above the bone however, is the harder skin structure of the nail bed that secures the nail and skin to the bone. This layered structure means that shearing forces in line with the outside skin of the finger pad, due to gripping, lifting and manipulating of objects, as well as when sliding over surfaces, are constrained by the strength of the epidermis to resist being stretched. However, forces normal to the epidermis flex the skin and produce large deformations of the fatty inner core allowing for a large level of compliance[6]. The same concepts appear in design
of car tyres, where the hard rubber needs to be strong in resisting the shearing forces in line with the road, where as the rubber’s flexibility and the compliance of the air filled core allows the tyre to conform to bumps and uneven surfaces, maintaining a large constant contact area with the road. This directionally constrained compliance of glabrous skin can be seen as a very important factor in the tactile sensing, gripping and manipulation abilities of our hand and fingers, as well as its importance as the surface we walk on when considering the sole of our feet. Our fingertip skin can conform to surfaces, edges and the smallest detail of features with the maximum area in contact with the skin[7]. This gives our tactile nerve endings the opportunity to cover the greatest surface area in the most detail[8] and allows us to maintain large, controllable contact with uneven surfaces and object shapes. The strength of the skin to resist shearing forces enables us to grip strongly, lift and manipulate objects, as well enabling us to explore with our hands, with little pressure needed to conforming to surfaces and features. A biologically inspired design focusing on the contrasting layered structure of the human fingertip should be able to capture some of these desirable qualities of compliance, conformability and mechanical strength.

III. FINGER DESIGN

An artificial fingertip was designed to try to emulate these key features from the biological example. It was decided to simplify the design to consist of a plastic bone and nail, a single soft inner core material, and a hard outside skin and nail bed. The nail was included in the design to act as an extension of the bone and, along with the nail bed, form a firm backing to the soft fingerpad.

A. Materials

Research was carried out into the materials commercially available. The key to the design was to capture the contrasting difference between the thin hard outside epidermal layer and the very soft inner fatty core. For the inner core, a special grade of commercially available silicone rubber, designed for the prosthetics and special effects industry, was selected. This particular grade has the lowest Shore Hardness rating currently available at 00-10. Within the range of readily available silicones, the maximum Shore Hardness was difficult to obtain beyond a Shore Hardness ‘A’ of 40. Beyond this the best rubber material available is Urethane; fortunately it also has a slightly oily surface not too unlike human skin. For this design a Shore Hardness ‘A’ of 60 Urethane was selected for the outside skin and nail bed and a liquid plastic was selected for casting the bone and plastic nails.

B. Construction

The moulding task took quite a lot of design. The materials needed for the finger were never designed to work together and there were some issues of cure inhibiting and unwanted adhesion or non-adhesion between the materials.
The final moulding itself needed the inner core encapsulated, completely enclosed within the urethane skin.

It also needed to be controlled and repeatable. After trying several methods to achieve this, a moulding method was designed that would successfully construct a finger. To simplify capturing the finger geometry, casts were made of a human index finger with the use of Alginate mould and cast in fine plaster, Fig. 2. The resultant plaster casts were used to make master moulds for the fingers, Fig. 3, and a suitable bone shape was sculpted in 'Plastiline' clay and a mould made similarly with the bones then cast in liquid plastic resin, Fig. 4. The overall moulding design was simplified so it could be moulded in two halves, the finger pad and soft inner core in one half and the nail bed and bone in the other.

A negative was made of the finger pad in plastic that was slightly smaller than the finger pad mould. The mould would then be filled with the liquid urethane skin material and the negative pressed into the mould, setting to form the thin outside skin layer with a level of overlap, Fig. 5. The bone was then glued into place in the finger pad and soft inner material was cast into it. The finished finger pad side was then pressed into the nail bed half of the mould and cast to securely form the nail bed side of the finger. Finally the plastic nail was glued onto the nail bed to get a strong contact with the nail bed and bone. The urethane skin of the finger pad was designed to have an overlap to gain a good adhesion.
and constant enclosure around the finger when cast into the
nail bed mould. Fig. 8 shows the final finger design. The
thickness of the urethane skin of the pad size was measured
to be around 0.5 to 0.7 mm thick. A further four fingertips
were cast in different materials for the task of comparison.
Each of these were as a solid one-piece casting and included
same the nail and bone design for consistancy. One was
casts using the Shore 'A' 60 urethane outside skin material
and one using the Shore 00-10 soft silicone core material
to represent the two mechaehical extremes of the biological
inspired design. A Shore Hardness 'A' 40 rubber and Shore
Hardness 'A' 20 rubber were also cast to represent the
standard material types chosen in robot fingers and gripping
surfaces. Fig. 9 show the range of finished fingertip designs.

IV. COMPARATIVE TESTING

Some simple preliminary comparative testing of the rubber
fingers was carried out to investigate if the biologically
inspired finger design had achieved a greater level of compli-
ance and conformability and maintained a strength in shear.
These were designed to give a quick first impression of the
attributes and feasability of the layered design, with the de-
tailed mechanical experimentation, modelling and simulation
to be the subject of following further work.

A. Contact Area vs. Pressure

The purpose of this test was to record the comparative
levels of contact area with finger pressure on a flat surface
in order to give some indication of compliance. This was
done simply by dipping the finger in a bowl of ink then
slowly and steadily pressing onto a piece of graph paper on
a scale balance; recording the maximum level of pressure
and measuring the surface area of the inky print.

Care was taken to maintain a consistent, constant speed to
minimise possible viscoelastic effects and angle of pressure.
Results were taken for all Shore 'A' rubber fingers, the
biologically inspired finger and the human index finger used
for the moulding process.

B. Conformability

The only test for conformability of fingertips in published
literature was done by Shimoga and Goldenberg[1] where the
artificial fingers were pressed with a constant force onto a 90
degree triangular edge, the amount of overlap of the flat sides
of the edge are taken as a measure of the conformability.
Similar tests were carried out with the rubber fingertips from
this work but at the much lower force of 5N.

C. Shear Resistance

The final test was to look at the resistance to shear forces.
Here the fingertips were vertically pressed, as in the contact
area test, and held with constant force of around 5N on
to a steel rule. The rule was then pulled horizontally at a
constant force of 5N and the level of horizontal deformation
was observed, in a similar vain to that of the conformability
tests.

V. RESULTS

<table>
<thead>
<tr>
<th>Finger</th>
<th>Comformability Measure, h mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shore A 60</td>
<td>0.5mm</td>
</tr>
<tr>
<td>Shore A 40</td>
<td>1mm</td>
</tr>
<tr>
<td>Shore A 20</td>
<td>1.5mm</td>
</tr>
<tr>
<td>Bio-Inspired Finger</td>
<td>3mm</td>
</tr>
<tr>
<td>Index Finger</td>
<td>4mm</td>
</tr>
<tr>
<td>Shore 00-10</td>
<td>5mm</td>
</tr>
</tbody>
</table>

Fig. 8. The final finger design

Fig. 9. The Range of Test Fingertips

Fig. 10. Illustration of Contact Area vs. Pressure Finger Press

Fig. 11. Biologically Inspired Finger Conformability Test
VI. DISCUSSION

The results from the contact area vs. pressure experiments show a clear and significant increase in compliance over the fingers representing standard rubbers used. The biologically inspired design resulted in over double the contact surface area compared to the Shore ‘A’ 20 rubber, a commonly used hardness for robot gripping surfaces. The results from the human index finger, however, showed an even greater level of compliance. It could be said that this is due to the comparative higher level of softness of the finger’s subcutaneous fat compared to the Shore 00-10 super soft silicon used. It is believed that an even softer material would be needed to match this finger’s level of compliance; there can be great variation between different people’s fingers. This candidate material would most likely have to be something other than silicone as Shore 00-10 is reaching the limit of commercially available silicone, further softening of the silicone resulted in a breaking down of the material’s strength and loss of its elasticity. Though using a liquid is a possibility, it would offer a very difficult moulding design and drastic problems would arise should the outside skin split or cut and it would most likely have very different mechanical properties overall too.

Having followed the test method of Shimoga and Goldenberg for conformability, it is doubtful that this method clearly shows the increased conformability the biologically inspired finger has, outperforming the others to even conforming to small features such as Braille dots and characters. However, it is difficult to see another way of illustrating this conformability within the scope of this preliminary study and will have to be the subject of the detailed mechanical study to follow. The tests for shearing strength were also difficult. Rubbers normally used for robot gripping surfaces do not produce easily measurable levels of surface shear at these forces, for their size. However, a more dramatic result comes from the shearing tests of the Shore 00-10 finger. The Shore 00-10 finger is very soft and so achieves a high level of compliance and conformability.

However, where it falls down is in its ability to cope with the shearing forces found in gripping, lifting and sliding. The soft rubber of the finger unable to cope with the shear forces applied to it, 13. The biological inspired finger achieves comparable levels of compliance and yet has little observable deformation under the same shearing forces, 14. This demonstrates clearly the benefits of the layered structure of the biologically inspired design. The level of compliance achieved far out performs that of the Shore Hardness ‘A’ rubbers, but it still maintains a comparable strength in shear, one which a similarly compliant single piece rubber does not. In other experiments, the biologically inspired finger acted...
VII. CONCLUSION

When considering the interactive nature of touch and manipulation, the importance of the mechanical properties and performance of the contact skin cannot be ignored. The need for high levels of compliance and conformability must be matched with a mechanical strength for manipulation. The design presented here offers a demonstration of how biologically inspired fingertips for robotic hands could gain some of the same desirable qualities human fingers have in conformability and compliance for tactile sensing and mechanical strength in shear for manipulation. Having a finger that can explore a surface and be sensitive enough to conform to every detail of its features as well as firmly grip, lift and manipulated that object is the key to robots start to successfully interact in uncontrolled, unstructured world in which we live.

VIII. FUTURE WORK

This was a presentation of the concepts behind a biologically inspired approach and of a preliminary fingertip design. Though a higher level of compliance and strength was achieved, there are a great deal other of mechanical interactions produced from such a multilayered structure. These are both mechanical complexities that need to be analysed and also practical implications to its use as a robotic finger. The mechanics of the current design will be analysed in greater depth and detail in modelling and simulation, its mechanical properties tested physically and it will be put through its paces practically to investigated it’s use as a robotic finger. It is worth researching further materials and construction methods, particularly to achieve a softer inner core, and be able to investigate what affect these have on the performance of the finger. Investigations with highlight the importance of the fingernail, the nail bed and the fingerprint and the possible inclusion of the dermis layer. The result of these studies will better inform and refine the design and aim to be the grounding for the successful inclusion of tactile sensing elements that will work with the layered structure design.

REFERENCES

Author Index

Akanyeti, O., 43, 57
Anderson, I.A., 118, 209
Antoun, S., 185
Aragon-Camarasa, G., 25
Balkenius, C., 79, 124
Bennet, D., 86
Billings, S., 43, 57
Bugmann, G., 130
Burbridge, C., 135
Calius, E.P., 118
Campbell, N., 9
Cernadas, E., 17
Chao, F., 72
Chorley, C., 239
Christensen-Dalsgaard, J., 193
Condell, J., 135
Culverhouse, P., 130
Davies, B.L., 234
Edwards, J., 201
Evans, M., 226
Fattah, H., 25
Fox, C., 226
Gamallo, C., 1
Gavshin, Y., 65
Gheisari, S., 142
Gibbons, P., 130
Gisby, T., 118, 209
Gu, D., 101
Hajati, F., 142
Hallam, J., 193
Hayes, G., 171
Haynes, B., 178
Hoeller, F., 93
Hosseini, H., 163
Ieropoulos, I., 209
Iglesias, R., 17
Jaeckel, P., 9
Johnsson, M., 79, 124
Johnstone, K., 149
Jorgensen, T.D., 178
Königs, A., 93

245