A Steerable Complex Wavelet Construction and Its Implementation on FPGA

C.-S. Bouganis[1], P.Y.K. Cheung[1], J. Ng[2], and A.A. Bharath[2]

[1] Department of Electrical & Electronic Engineering, Imperial College, Exhibition Road, London SW7 2BT, U.K.
[2] Department of Bioengineering, Imperial College, Exhibition Road, London SW7 2AZ, U.K.

Abstract. This work addresses the design of a novel complex steerable wavelet construction and its implementation on reconfigurable logic. The wavelet decomposition uses pairs of bandpass filters that display symmetry and antisymmetry about a steerable axis of orientation. The design is targeted for implementation in hardware, thus one of the desired properties is the small number of unique kernels. A detailed description of the implementation of the design in hardware is given. Moreover, results regarding the speed of our design compared to a software implementation, and the error in the filter responses due to fixed point representation, are reported. To show the applicability of the design to real life situations, a corner detection algorithm is illustrated.

1 Introduction

The applications of wavelets to signal and image compression are well researched [1,2,3]. The work described here contains several points of departure in both the construction and application of steerable filters to feature detection. The main point of departure is that the filter kernels are specified by separable angular and radial functions in the frequency domain which have not been jointly reported in a multi-rate scheme. In addition, an implementation of the algorithm in hardware is performed, targeting real-time applications.

The need for more flexibility and fast prototyping of signal processing algorithms has lead the FPGA community to investigate tools for easy mapping of signal processing algorithms to FPGAs. One approach is to provide the designers with building blocks that are common in DSP applications [5]. Another approach is to provide tools that allow the engineers to describe their design in a high level language [6]. In this work, Handel-C [9] is used as the main tool to describe the steerable complex wavelet pyramid on hardware. Handel-C is based on the syntax of ANSI C with additional extensions in order to take advantage of the specific characteristics of the hardware. It is independent of the targeting platform which makes the design easily transferable to other hardware platforms. The originality of this work is in the design of a novel steerable wavelet construction and the investigation of mapping such a design to reconfigurable logic, targeting real-time applications.

J. Becker, M. Platzner, S. Vernalde (Eds.): FPL 2004, LNCS 3203, pp. 394–403, 2004.

The paper is organized as follows. In section 2.1 the motivation for implementing a complex wavelet decomposition is given. In section 2.2 a detailed description of the pyramid is given as well as how this design differs from other constructions. In section 3, the application of the pyramid to feature detection is demonstrated using as example a corner detection algorithm. The implementation of the design on FPGA is described in section 4. Finally, the impact of the quantization of the variables on the overall performance of the algorithm and a comparison between the hardware implementation to the software implementation of the algorithm regarding the speed is given.

2 The Pyramid Design

2.1 Motivation

The motivation for a new pyramidal decomposition has been a subband decomposition of images into orientation and scale-selective channels that can then be used for analysis purposes. Although there are numerous decompositions that satisfy such a requirement, we are seeking a construction that utilises polar separable functions, so that the orientation selectivity can be specified independently of radial frequency (or scale selectivity), and at the same time a small number of unique kernels for construction is required for implementation of the design in hardware. To provide a variety of scale-selective channels, we have chosen to rely on standard multi-rate techniques, which enhance the computational and representational efficiency of such decompositions. Within each orientation and frequency channel, we wish to estimate the local image symmetry/antisymmetry about an axis. Using a small number of kernels, the axis should be tunable, depending on the local image content. These requirements are met by a steerable quadrature wavelet decomposition, of which some examples can be found in [3,7].

2.2 Design Overview

The design of the pyramid employs decimation in the lowpass channel in order to achieve the scaling of filter response through repeated application. The nature of the decomposition is illustrated in Figure 1. The decomposition is repeated four times in order to detect symmetric and antisymmetric regions in the image in different scales. The design of the filter kernels is performed in the Fourier domain and the inverse two-dimensional Fourier Transform is applied to compute the spatial impulse responses. For convenience in tuning angular and radial characteristics of the filters, we impose Fourier domain polar separability, so that a filter $G_{0,k}(\omega, \phi)$ in the k^{th} direction in a filter set can be specified as the product of a radial frequency function $\Omega_0(\omega)$ and an angular frequency function $\Phi_{0,k}(\phi)$, i.e. $G_{0,k}(\omega, \phi) = \Omega_0(\omega)\Phi_{0,k}(\phi)$.

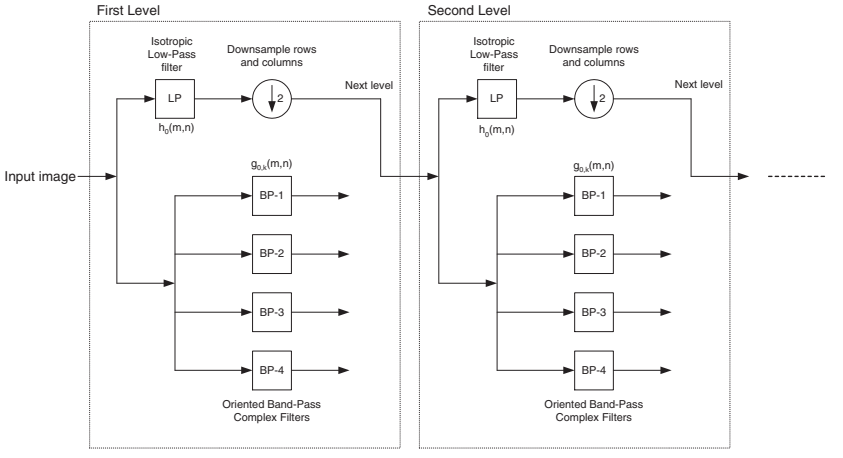


Fig. 1. Pyramid layout. The boxes with the dotted lines illustrate a single level of decomposition. The full decomposition consists of four levels. The responses of the bandpass filters detect symmetric and antisymmetric features in the image.

2.3 Radial Frequency Response

For the isotropic lowpass radial frequency response we have used the following function in the radial frequency domain on the interval $-\pi < \omega \leq \pi$.

$$H_0(\omega, \phi) = H_0(\omega) = \frac{1}{1 + (\omega/\omega_c)^6} \tag{1}$$

where $\omega_c = 3\pi/8$. It was chosen to provide a reasonably flat power response, when used in combination with the bandpass radial frequency response, defined later, for radial frequency components in the range $[0, \omega_{max}]$. ω_{max} is the peak frequency of the bandpass radial frequency response.

The radial response of the bandpass filters, $\Omega_0(\omega)$, is based on Erlang functions which are one sided, smooth, and have the property that $\Omega_0(0) = 0$. The joint localisation of Gaussian kernels in both spatial and frequency domain causes transform coefficients to fall off in magnitude as scale is increased [4]. This is undesirable for a hardware implementation since more coefficients are required to represent the kernels of the pyramid. Using an α value smaller than one (Poisson and Erlang, $\alpha = 0.5$) biases the localisation towards the frequency domain and provides an increased stability of transform coefficients across scales. The filters employed here have radial frequency response

$$\Omega_0(\omega) = \left(\frac{e}{14}\right)^7 \omega^7 e^{-\omega/2} U(\omega) \tag{2}$$

where $U(\omega)$ is the unit step function.

2.4 Angular Frequency Response

The prototype general angular frequency characteristic

$$\Omega_0(\phi) = \cos^3(\phi) rect(\phi/\pi) \tag{3}$$

has been used, where $rect(\phi) = U(\phi + \frac{1}{2})U(\frac{1}{2} - \phi)$. This is a generalization of the sin and cosine angular characteristics used in derivative of Gaussian processing, but with tunable angular selectivity. The prototype angular frequency is rotated to generate the angular characteristics of oriented filters for a full filter set by the following:

$$\Phi_{0,k}(\phi) = \Omega_0(\phi - \phi_k) \tag{4}$$

In our design four orientations are used at $0, \pi/4, \pi/2$ and $3\pi/4$.

2.5 Filter Kernels

For each of the filter prototypes in the Fourier domain, a sampling on the two-dimensional interval $[-\pi, \pi] \times [-\pi, \pi]$ was performed, with a grid spacing of $\pi/64$ in each cartesian direction. The choice of an odd matrix size for constructing the Fourier domain representation is tied to the symmetry of the filter kernels, which we have observed to be better on odd-sized grids.

The inverse two dimensional discrete Fourier Transform was computed to extract 65×65 spatial frequency responses. These responses were each truncated to fit a set of four 7×7 complex arrays. The larger the size of the kernels the better the properties of the filters are preserved, but more area is required to implement these kernels on hardware. The kernels thus extracted are illustrated in Figure 2. The first row corresponds to the real component of the kernels which detect even-symmetric features in the image such as lines. The second row corresponds to the imaginary component responsible for detecting the parts of the image with odd-symmetric content, such as edges. The symmetry properties of the filters fall into various classes. We have identified five classes of coefficient symmetries. More details about kernel construction, the symmetry classes of the filters and the coefficients of the filter blocks can be found in [10,11]. However, the current hardware design has not been optimized with respect to these symmetric properties. Future work will take into account these properties in order to reduce the computational load in the FPGA.

3 Generating Feature Maps

3.1 Corner Likelihood Response

The output of the filters may be used to generate a measure that may be treated as being proportional to the likelihood of a particular location in an image being the corner of some structure. We construct the following feature map

$$C^{\ell}(m,n) = \frac{\prod_{k=0}^{3} |f_k^{(\ell)}(m,n)|}{p + \sum_{k=0}^{3} |f_k^{(\ell)}(m,n)|^4} \tag{5}$$

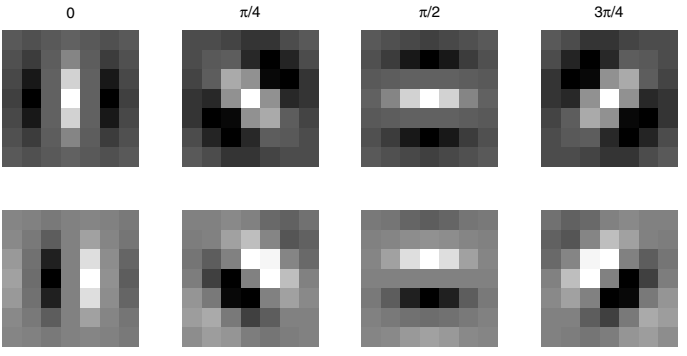


Fig. 2. Real and Imaginary parts of complex bandpass filter kernels. Top row shows the real part of the kernels, where the bottom row shows the imaginary part.

$f_k^{(\ell)}(m,n)$ denotes the response of the filter k of level ℓ. m, n denote the indexes inside the image. p is fixed to be 4% of the maximum pixel intensity in the image preventing the feature map to take large values for small values of $\sum_{k=0}^{3} |f_k^{(\ell)}(m,n)|^4$. The denumerator normalizes the response to a local anisotropic energy. Moreover, we may choose to weight the corner response by anisotropic energy computed at the same, or another scale.

4 Implementation on FPGA

The complex steerable wavelet was designed to be "hardware-friendly" by targeting to a minimum number of distinct and symmetric kernels. A hardware implementation using reconfigurable logic was investigated to accelerate the decomposition part of the algorithm which leaves "high-level" decisions such as the implementation of the feature maps to the host CPU. Only one level of the pyramid is implemented in hardware, and the full decomposition is realised through reuse of the same hardware, having as input the decimated image from the previous iteration.

The target board that is used for implementation is the RC1000-PP from Celoxica. It is a PCI bus plug-in card for PC's with a Virtex V1000 FPGA and four memory banks of 2 MBytes each. All four memory banks are accessible by both the FPGA and any device on the PCI bus. However, at any time instance only one device can access a memory bank. The Handel-C language is used to describe the design.

4.1 FPGA Design

The quantization of the variables in the design is as follows: eight bits are used to represent a pixel in the image and ten bits are used to represent the coefficients of each filter and also the output of each convolution. The impact to the

final accuracy of the algorithm by selecting these numbers of bits to represent the variables is discussed in section 4.3. In order for the decomposition to be performed as fast as possible, the whole design is pipelined in order to produce three convolution results per clock cycle.

Figure 3(a) shows an overview of the design. The pixels are stored in a raster scan in sets of four in the memory allowing the FPGA to fetch four pixels per read cycle. The *ManageBuffer* process is responsible to buffer the data such as to provide a region of the image to the next process in the design. The FIFOs are mapped to block RAMs in the FPGA for a more effective use of resources. The next process, the *ProcessWindow*, performs the convolution between a window in the image and the appropriate masks. It contains three programmable processes *FilterBankA (FBA)*, *FilterBankB (FBB)* and *FilterBankC (FBC)* that each one can apply three different filters by loading a specific set of coefficients. A shift register and a RAM to store the coefficients is selected to form the appropriate masks for each level of the pyramid. The final results are concatenated and stored in the external RAM. Figure 3(b) shows a detailed diagram of the *FilterBank* process. Moreover, the result from the last filter, which represents the input image for the next level of the pyramid, is decimated, saturated in the range $[0, 255]$ and stored in the external memory by the *NextLevelImage* process.

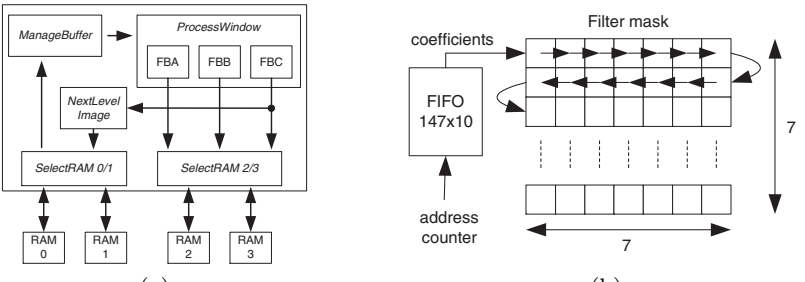


Fig. 3. (a) shows the top level diagram of the design. (b) shows the *FilterBank* process. The FIFO contains the coefficients for the three kernels that are realised in the filter bank.

4.2 Host Control

The CPU controls the operation of the FPGA by a handshake protocol. Due to the associated latency of each transfer through the PCI bus, the data are transferred using DMA access between the CPU and the board [12]. In order to speed up the process, the decomposition of the image and the transfer of the data to/from the host are performed in parallel. The following scheme is used. Out of the four memory banks, the first two are used to store the new frame that is sent by the host for processing, the previous frame that is being processed by the FPGA, and the decimated images that are used for the different levels of the

pyramid. The other two banks are used by the FPGA to store the result of the convolutions. The handshake protocol operates as follows. When a new frame is available, the CPU sends the data to RAM 0 or 1 and signals to the FPGA that a new frame is waiting for processing. In the meantime, the FPGA processes the previous frame from RAM 1 or 0 respectively. The results from the convolutions are stored in RAMs 2 and 3. The output data are distributed between RAMs 2 and 3 such that while the FPGA writes the results from a convolution to one RAM the CPU performs a DMA transfer to the already calculated results from the other RAM. The distribution of the results is necessary, since the design should be able to handle images with size 640 by 480 pixels.

4.3 Implementation Analysis

Experiments were performed to investigate the impact of the number of bits that are used to represent the kernel coefficients (N_c) and the bits that are used to represent the result of the convolution (N_o) to the filter responses. The mean square error of the estimation of each filter response between full precision and fixed point for each combination of N_c and N_o is estimated using the Lena image. Figure 4 shows the average mean square error over all filters using the same combination of N_c and N_o. In our design, N_o is set to 10 in order to be able to store the results of three parallel convolutions by performing only one 32-bit access to the external memory. From the figure, it can be concluded that the number of bits used for the coefficients has a small effect on the error of the response compared to the number of bits used to represent the result of the filters. Also, it should be mentioned that the error in the filter responses increases after changing levels since the decimated result of the low-pass channel is reused for the next bandpass decomposition.

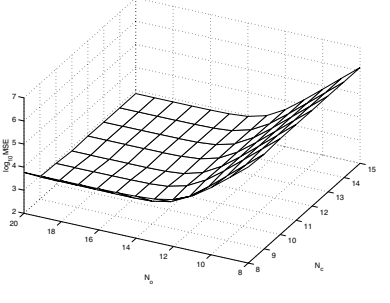


Fig. 4. Mean square error in the filters' response using the Lena image between fixed-point and floating-point arithmetic.

The overall design uses 12,286 slices. Due to the large size of the design compared to the available space in Virtex V1000, the optimum clock rate can not be achieved. The synthesis results of the design using Xilinx ISE 6.1 gives 99%

usage of slices. It is clear that there is not enough available space for optimum routing, which reduces the optimal clock frequency. Furthermore, the slow RAMs that are available on the board reduce the effective speed of the design. Due to the nature of the decomposition, the data that are generated correspond to an equivalent size of 14.6 times the input image. This amount of data cannot be stored in the internal memory of the FPGA and should be transferred to the external RAMs. The available bandwidth to the external memories reduces the effective speed of the current design. The required memory bandwidth by the design for VGA resolution (640x480) at 25 frames per second is 109 MBytes/sec, where the available bandwidth is 66 MBytes/sec assuming only one available memory bank. A rate of 16.6MHz was achieved giving 13.1 frames per second in VGA resolution.

5 Results

5.1 Performance Analysis

Experiments were performed to compare the speed of the new design to a software implementation. A decomposition with four orientations and four levels is performed on two test images with size 256x256 and 512x512. Table 1 shows a summary of the results. The first row of the table corresponds to a machine with Dual Hyperthreading Xeons at 2.66GHz with 2GB of RAM. The software runs under MATLAB and it is optimized using the *Intel SIMD Integrated Performance Primitives* library which also takes advantage of multiprocessors. The second row corresponds to a similar machine but without hyperthreading technology. The software version of the design was implemented using single, dual and quad threads. The RC1000-PP board is placed on a Dual Pentium III machine at 450MHz and 512MB of RAM. The results for the software is the average over 40 frames, where for the hardware the results is the average of 4000 frames. The timing for the FPGA include the DMA transfers. In both cases the required time to read the data from the hard disk is excluded. The *speed up factor* is calculated with respect to the best performance of the software implementation in each row. It can be seen that an average improvement of 2.6 times in the speed can be achieved. Moreover, we placed our design in an XC2V6000 to investigate how fast the current design can be clocked without any restrictions from the size of the FPGA device or by the timing constraints of the external memories. The synthesis tool showed that the design can be clocked up to 50MHz giving an average *speed up factor* of 8 compared to the software implementations.

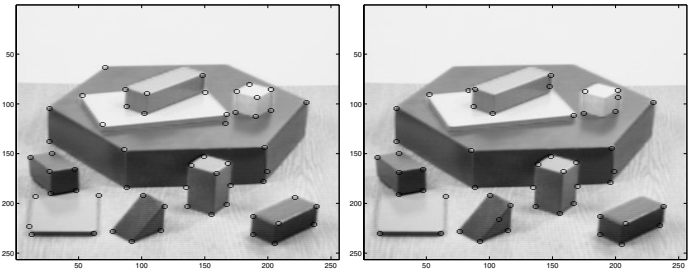
5.2 Corner Detection

Further experiments are performed to assess the performance of the design to real-life situations. The application under consideration is corner detection using the algorithm described in section 3. We investigate how precisely the corner of a structure in the image is detected given the limited number of bits that are used

Table 1. Comparison results in speed between software and hardware implementation. Using a XC2V6000 device a *speed up factor* of 8 is achieved.

Image size	HT	Single Thr.	Dual Thr.	Quad Thr.	FPGA Accel.	Speed up factor
Lena 256x256	Yes	0.06035s	0.04897s	0.05088s	0.0170s	2.88
	No	0.05953s	0.04737s	-		2.78
Boats 512x512	Yes	0.21074s	0.21281s	0.16113s	0.0653s	2.46
	No	0.20720s	0.15724s	-		2.40

to represent the coefficients and the response of the filters. Figure 5 shows the performance of the above design compared to a software implementation. The image on the left is the result of the corner detection when the whole algorithm is implemented in software. The image on the right is the result of the corner detection when the decomposition of the image is performed in the FPGA. It can be seen that most of the features have been detected correctly except of 8 mismatches. Further investigation revealed that by assigning 16 bits to represent the output of the filters gives zero mismatches. However, a 16 bit representation for the results would involve access to two memory banks simultaneously, forcing the FPGA to wait for each DMA transfer to finish. This results in a reduction in performance by a factor of 1.5, using the RC1000-PP board.



(a) (b)

Fig. 5. (a) shows the result of the corner detection using software. (b) shows the same result using the hardware implementation.

6 Conclusions

In this paper we have presented a novel steerable pyramid for image decomposition and feature detection. For speeding up the algorithm, a mapping to reconfigurable logic was performed. We investigated the impact of the quantization of the variables to the filter responses and pointed out potential problems in the design of such multi-level transforms. Due to the nature of the algorithm, a huge amount of data is produced and can be stored only in the external RAMs. The current design is limited by the available bandwidth to the external memories.

The current prototype will be used as a platform for research in word-length optimization [8] over multiple coefficient masks that use the same paths. Moreover, future work involves the investigation of automated tools that optimize the design of wavelet transforms taking into account the symmetry properties of the filters in the case where the same part of hardware is used by different kernels.

Acknowledgement. The authors would like to thank George Constantinides from Imperial College for his help and support. Also, we would like to thank Wayne Luk and Altaf Abdul Gaffar for lending us the RC1000-PP board. This work was funded by the UK Research Council under the Basic Technology Research Programme "Reverse Engineering Human Visual Processes" GR/R87642/02.

References

1. L. Sendur and I. W. Selesnick, "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency," IEEE Transactions on Signal Processing, Vol. 50, No. 11, November 2002.
2. N. Kingsbury, "Image processing with complex wavelets," Philosophical Transactions Of The Royal Society Of London - Series A, Vol. 357, No. 1760, September 2002, pp. 2543-2560.
3. W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," IEEE Transactions on Pattern Analysis and Machine Inteligence, Vol. 13, No. 9, 1991, pp. 891-906.
4. T. Lindeberg, "Principles for automatic scale selection," in Handbook on Computer Vision and Applications. Academic Press, Vol. 2, 1999, pp. 239-274.
5. M. Nibouche, A. Bouridane, D. Crookes and O. Nibouche, "An FPGA-based wavelet transforms coprocessor," IEEE International Conference on Image Processing, Vol. 3, 2001, pp. 194-197.
6. P. Bellows and B. Hutchings, "Designing run-time reconfigurable systems with JHDL," Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Vol. 28, No. 1-2, May/June, 2001, pp. 29-45
7. T. C. Folson and R. B. Pinter, "Primitive features by steering, quadrature and scale," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 11, 1998, pp. 1161-1173
8. G. A. Constantinides, P. Y. K. Cheung and W. Luk "Wordlength Optimization for Linear Digital Signal Processing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 22, No. 10, October 2003
9. I. Page and W. Luk, "Compiling occam into FPGAs," in Will Moore and Wayne Luk (Eds) 'FPGAs', pp. 271-283, Abingdon EE&CS books, 1991
10. J. Ng and A. A. Bharath, "Steering in Scale Space to Optimally Detect Image Structures," ECCV 04. In Lecture Notes in Computer Science. May 2004
11. http://www.bg.ic.ac.uk/Publications/TIP-00621-2003/AnalysisFilterCoefs.htm
12. http://www.celoxica.com/techlib/files/CEL-W0307171JKX-33.pdf