FPGA-based Real-time Super-Resolution on an Adaptive Image Sensor

Maria E. Angelopoulou, Christos-Savvas Bouganis, Peter Y. K. Cheung, and George A. Constantinides

Department of Electrical and Electronic Engineering, Imperial College London,
Exhibition Road, London SW7 2BT, UK
email: {m.angelopoulou, christos-savvas.bouganis, p.cheung, g.constantinides}@imperial.ac.uk

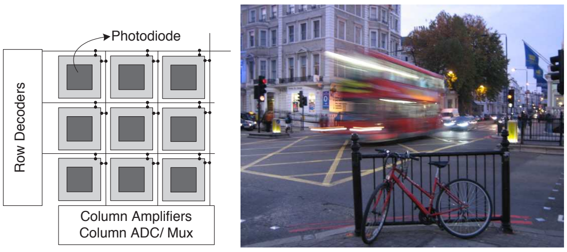
Abstract. Recent technological advances in imaging industry have lead to the production of imaging systems with high density pixel sensors. However, their long exposure times limit their applications to static images due to the motion blur effect. This work presents a system that reduces the motion blurring using a time-variant image sensor. This sensor can combine several pixels together to form a larger pixel when it is necessary. Larger pixels require shorter exposure times and produce high frame-rate samples with reduced motion blur. An FPGA is employed to enhance the spatial resolution of these samples employing Super Resolution (SR) techniques in real-time. This work focuses on the spatial resolution enhancement block and presents an FPGA implementation of the Iterative Back Projection (IBP) SR algorithm. The proposed architecture achieves 25 fps for VGA input and can serve as a general purpose real-time resolution enhancement system.

1 Introduction

Every imaging system is based on an image sensor, a 2-D array of pixels that convert incident light to an array of electrical signals (Fig. 1(a)) [1]. Two types of resolution determine the quality of information collected by the sensor: the spatial and the temporal resolution. The *spatial* resolution depends on the spatial density of the photodiodes and their induced blur. The most intuitive solution to increase the spatial resolution corresponding to the same field of view would be reducing the pixel size, hence increasing the pixel density. However, the smaller the photodiodes become, the smaller is the amount of incident light and, therefore, a longer integration time is required for each photodiode to achieve an adequate signal to noise ratio [1, 2].

In the case of no relative motion between the camera and the scene, the reduction in the amount of light can be compensated by increasing the exposure time of the pixels, *i.e.* increasing the integration time of the photodiodes. However, in real-life systems either the camera is shaking or/and objects are moving in the scene during the integration time. In this case, the integration time spans a large number of real-world 'frames', and the output suffers from motion blur, thus reducing the temporal resolution. In Fig. 1(b), the effect of motion blur is clearly visible: the exposure time was too long for the fast moving bus to be captured. Thus, there is a fundamental trade-off in imaging systems:

an increase in the spatial resolution by reducing the pixel size reduces the temporal resolution and vice-versa. For the rest of the paper, ‘LR’ denotes the low spatial resolution and, thus, high temporal resolution image samples, while ‘HR’ refers to high spatial and low temporal resolution.



(a) (b)

Fig. 1. (a) A hypothetical 3×3 CMOS image sensor. (b) A moving bus as opposed to a still bike. The first creates motion blur, whereas the second is fully captured.

Recently researchers have focused on the problem of enhancing both spatial and temporal resolution. Resolution in both time and space can be enhanced by using multiple cameras to capture a fast moving scene with different subpixel spatial shifts and different subframe temporal shifts [3]. The main strength of the algorithm in [3] is that it treats motion blur independently of the cause of temporal change. Its main weakness lies in the large number of required cameras (such as 18). In real-life systems, this also introduces additional difficulties in the alignment of all the captured images from different cameras, a step known as registration. Apart from having to perform registration on many images, the large number of cameras increases the distances between the camera axes, making accurate registration difficult. This limits the applicability of the system.

In [4] the proposed system consists of a HR and a LR imaging device. The LR device deblurs the image captured by the HR device, by obtaining motion information for the estimation of the motion Point Spread Function (PSF). Then, the HR image is deblurred using deconvolution-based techniques. This approach mainly considers capturing a single image focusing in solving the blur caused by the undesired global motion due to camera shaking. The proposed system uses either two separate image sensors or a sensor with a LR periphery. If two separate image sensors are used, motion trajectories can be detected anywhere in the frame and, thus, the approach can be extended to dealing with the motion of objects. However, the use of two image sensors results in registration-related problems and an increased size of the device. In addition, the pixel size of the LR detector remains fixed over time regardless of the motion magnitude.

In summary, the contributions of the paper are: (1) The introduction of a motion-deblurring system which employs an FPGA to dynamically configure a time-variant image sensor. The size of the pixels is adapted according to local motions within the frame. The FPGA is used for the spatial enhancement of the high frame-rate areas, which are locally formed on the sensor, to provide super-resolution (SR) effects. (2) An efficient FPGA architecture is proposed for the implementation of the resolution enhancement module of the SR algorithm based on the Iterative Back Projection approach, and its performance is investigated. To the best of our knowledge, no FPGA implementation of an SR algorithm has been previously reported in literature.

The structure of the paper is as follows. Section 2 presents the architecture of the proposed FPGA-based motion-deblurring system and focuses on the spatial enhancement block, introducing SR and the Iterative Back Projection algorithm in particular. Section 3 describes the FPGA implementation of the SR block. In Section 4 hardware and quality results of the implementation are presented. Section 5 concludes the paper.

2 Surpassing the Fundamental Trade-off: Our Proposal

2.1 Description of the Motion-Deblurring System

The state of the art in imaging technology has produced sensors that are no longer subject to the constraint of time-invariant fixed pixel size [5, 6]. Elementary pixels can be grouped together over time, to form neighborhoods of different resolution. Taking advantage of what imaging technology has to offer, this work proposes an FPGA-based system that uses an adaptive image sensor to locally form areas of larger pixels and execute on-line, real-time motion deblurring. Fig. 2 presents an overview of this system.

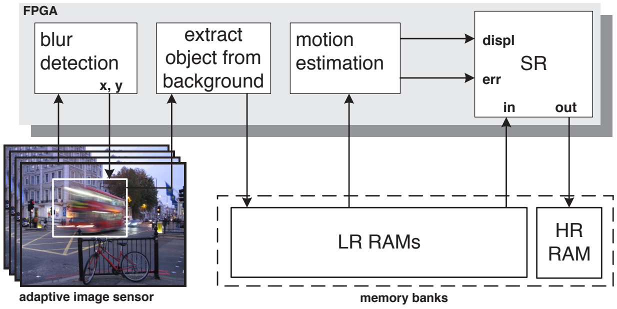


Fig. 2. The proposed motion-deblurring FPGA-based system that uses an adaptive image sensor. Areas of large pixels are formed where motion exists and the LR samples are spatially enhanced.

Let S_h denote the size of the elementary pixel of the sensor, corresponding to resolution HR (*i.e.* the highest spatial and lowest temporal resolution). Let m and n be the height and width of an area of the sensor measured in S_h units. That area may include pixels larger than S_h and, thus, produce multiple time samples during the HR integration. If all pixels, *regardless of their size*, are considered as points in the 3-D space, then during the HR integration $m \times n$ such points will be produced for an $m \times n$ area. The distribution of these points between time and space is determined by the pixel size. Increasing the pixel size of a particular region, decreases the density of these points on the 2-D plane and increases their density along the time axis, as the total number of points should remain $m \times n$ for the given area. Therefore, in one end, there is the still regions - covered with HR pixels - with distribution $m \times n \times 1$ (m in x, n in y and 1 in t), and at the other end lies the configuration $1 \times 1 \times (m \times n)$, if all the available pixels are grouped together to form one large pixel. Thus, if the pixel size of area Q equals 2×2 S_h, the LR spatial resolution is 4 times lower than the HR resolution, while the temporal resolution is 4 times higher, *i.e.* 4 LR time samples are produced for Q during the HR integration. If the spatial relation is 3×3, 9 LR samples are produced, *etc.*

The *Blur Detection* block of Fig. 2 reads the sensor's output and indicates the blurred regions. These regions of the sensor will be configured to larger pixel sizes. If the motion blur derives from camera shaking a single motion region spans the entire sensor. During the HR integration time, a sequence of LR frames will be produced at every motion region, where the blur effect is reduced. Before executing SR on this group of LR frames, the static background should be removed by applying a background extraction algorithm.

The *Motion Estimation* block of Fig. 2 reads the sequence of LR frames and returns the motion vectors, *i.e.* the displacements of selected features between each LR frame and the reference LR frame. Any frame of the LR sequence can be chosen as the reference frame. These displacements will then be used by the *SR* unit to enhance the spatial resolution. The spatial resolution and the frame-rate of the final deblurred output will be those corresponding to the HR sequence.

The robustness of the system is increased by applying the following two techniques. The error information at the output of the *Motion Estimation* block [7] is used by the SR block to weight the information of the different LR samples and decrease the contribution of those with large error values. Additionally, to increase the available spatial information at the input of the SR block, neighboring LR samples before and after the integration interval of interest contribute in SR with adjustable weights.

2.2 Super Resolution

The forward model of generating LR pixels is shown in Fig. 3. Many HR pixels are mapped on a single LR pixel, thus imitating the integration of a group of HR pixels on a single photodiode. The weights with which these HR pixels contribute in the formation of the particular LR pixel form a gaussian kernel–the 2-D PSF shown in Fig. 3. Every LR pixel can be thus expressed as a weighted sum of HR pixels, and the following linear system of equations is formed:

$$A\boldsymbol{h} = \boldsymbol{l} \tag{1}$$

where $\boldsymbol{h}$ and $\boldsymbol{l}$ denote the vectors of unknown HR pixels and known LR pixels, and matrix A contains the relative contribution of each HR pixel to each LR pixel.

The aim of spatial SR is to solve the inverse problem of finding $\boldsymbol{h}$. The HR grid on which reconstruction will occur is the HR grid underlying the LR reference grid (Fig. 3). Thus, $\boldsymbol{h}$ consists of the HR pixels of this grid. Each LR frame adds an extra set of equations in the system, one for every LR pixel.

Spatial SR is based on subpixel shifts on the LR reference grid. If a group of LR frames were shifted on the reference LR grid (Fig. 3) by integer LR pixel units, they would all give the same set of equations since the same groups of HR pixels would form in the same manner their LR pixels. Therefore, for a LR frame to contribute uniquely in the system of Eq. 1, it should be shifted by subpixel units on the LR reference grid compared to the other LR frames. However, although in theory the above statements are true, in practice LR frames with the same integer displacements may give different sets of equations. This is partly due to errors in the motion estimation procedure [7] and partly due to the quantization of the LR grid on the HR grid on which reconstruction is executed. Therefore in practice it is preferable if more LR frames are considered, even if their displacements overlap.

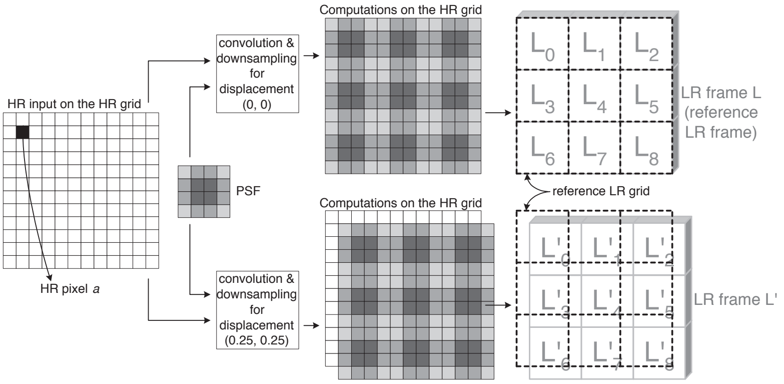

Computations on the HR grid
convolution & downsampling for displacement (0, 0)
HR input on the HR grid
L_0 L_1 L_2
L_3 L_4 L_5
L_6 L_7 L_8
LR frame L (reference LR frame)
PSF
reference LR grid
Computations on the HR grid
HR pixel *a*
convolution & downsampling for displacement (0.25, 0.25)
L'_0 L'_1 L'_2
L'_3 L'_4 L'_5
L'_6 L'_7 L'_8
LR frame L'

Fig. 3. *The formation of the LR output presented mathematically. A* 4×4 *PSF is employed. Two simulated LR frames with displacements* $(0, 0)$ *and* $(0.25, 0.25)$ *are produced.*

The SR methods found in the literature solve the SR problem either in the spatial or in the frequency domain. In this work, a spatial domain method is implemented. This avoids the transformations between the two domains, and also removes the need to handle outputs with large dynamic range as produced by frequency domain analysis. Therefore, the need for long word-lengths in hardware implementations is not required. Among the spatial domain methods the Iterative Back Projection (IBP) [8] approach was selected because of its hardware-friendly characteristics. Instead of solving Eq. 1 for $\boldsymbol{h}$, the IBP produces a simulated LR sequence and iteratively minimizes its difference from the observed LR sequence. This iterative scheme is suitable for hardware due to its potential for maximum parallelism and data re-use, as it will be demonstrated in Section 3.

Iterative Back Projection (IBP) The IBP employs an iterative refinement scheme on the HR grid, starting with an initial HR approximation such as the interpolation of the reference LR frame. Then, at every iteration of the algorithm the forward model of Fig. 3 is applied on the current HR approximation using the displacements of the corresponding observed LR frames to produce a simulated LR sequence. The aim of IBP is to minimize the difference between the observed and the simulated LR sequence, by refining the HR estimation.

All of the observed LR pixels and the corresponding simulated LR pixels which are influenced by a particular HR pixel contribute in the refinement of that HR pixel. This contribution is weighted according to the relative position of that HR pixel and the LR pair. For instance, in the refinement of HR pixel a (Fig. 3), pixel $L0$ of frame L participates with a weight proportional to $PSF(1,1)$, whereas for $L'0$ of L' this weight will be proportional to $PSF(0,0)$.

At iteration i, every pixel of the current HR approximation H_i is refined as follows:

$$H_{i+1}(x_h, y_h) = H_i(x_h, y_h) + \sum_{k=0}^{K-1} \sum_{(x_l,y_l)\in Y} (Lo_k(x_l, y_l) - {Ls_k}^{(i)}(x_l, y_l)) \times W(k, x_l, y_l), \tag{2}$$

where Lo_k and $Ls_k^{(i)}$ denote the kth observed and simulated LR frame, (x_h, y_h) and (x_l, y_l) denote the HR and LR coordinates, Y is the set of LR coordinates of the pixels of Lo_k and $Ls_k^{(i)}$ which are influenced by point (x_h, y_h), W is the weight with which $Lo_k(x_l, y_l)$ and $Ls_k^{(i)}(x_l, y_l)$ contribute in the refinement of $H_i(x_h, y_h)$, and K is the number of LR frames.

3 Architecture of the SR System

Figure 4 shows an overview of the proposed system. For every new group of LR frames, produced during a particular HR integration interval (Sect. 2.1), an SR stage occurs. At the beginning of each SR stage an initial HR approximation is produced by applying interpolation on the reference LR frame. Once this initial phase is completed, the iterations of the algorithm begin. When the iterations are over, the next LR group (associated with the next HR integration interval) is processed, and so on. The rest of the section focuses on the description of the individual blocks. It should be mentioned that the target system has 4 memory banks, each with a word-length of 4 bytes.

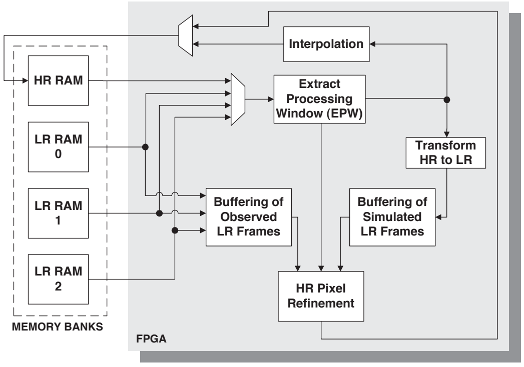
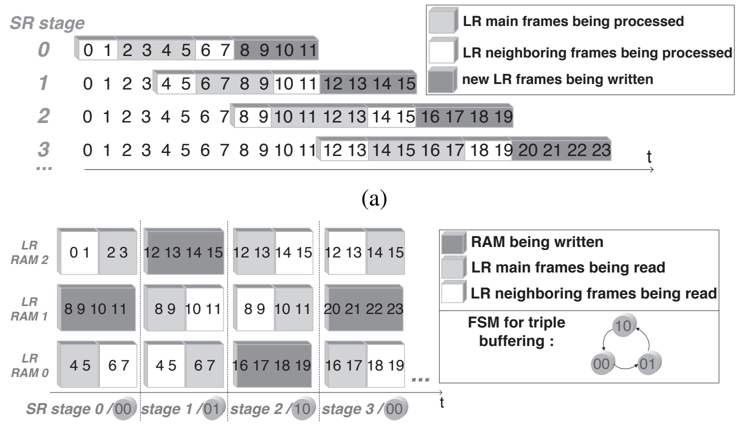


Fig. 4. Architecture overview

3.1 Off-chip Memory Banks

LR RAMs The LR memory banks store the incoming LR frames. As has been mentioned, the processing of the LR frames is performed in groups that correspond to one HR frame. However, in order to increase the spatial information available to the proposed system, a number of neighboring LR frames are used in addition to those produced during the HR integration (Fig. 5(a)). In hardware, this means that two memory banks need to be read in parallel, as Fig. 5(b) illustrates for the case of a 2×2 PSF and four neighboring frames. For instance, in SR stage 1 (Fig. 5(b)) frames 8-11 need to be read together with frames 4-7 which are in a different RAM bank due to the state of SR stage 0. In order to handle this we employ a triple buffering scheme. The access pattern of LR RAMs is shown in Fig. 5(b).



(a)

(b)

Fig. 5. The numbers correspond to the LR frame number. Four LR frames are produced during the HR integration and two pairs of neighboring frames (one pair at each side of the integration interval) are considered. (a) A sliding window indicates the group of LR frames processed during the current SR stage. While the processing occurs on these frames, a new group of four frames is written in the memory banks. (b) Triple buffering scheme applied on the LR RAMs. As the SR stages succeed one another the LR frames are written and read from the LR RAMs according to the current state of an FSM. There are three possible configurations.

HR RAM This external memory stores the computed HR pixels. During the initial phase of the current SR stage, data come into the HR RAM from the *Interpolation* unit. Once the initial estimation is computed, data come from the *HR Pixel Refinement* unit, which iteratively updates the content of the RAM until the end of the current SR stage. Before a pixel is written in HR RAM it is *rounded* to 8 bits. This allows storing HR pixels in groups of four in the 32-bit RAM, thus increasing the available memory bandwidth.

3.2 Individual Processing Units

The *Extract Processing Window* (EPW) unit of Fig. 4 produces the processing window for both the *Interpolation* and the *Transform HR to LR* units, at different phases of the SR stage. Thus, it operates in two modes. In *Mode 1* it returns a 2×2 window to the *Interpolation* unit, while in *Mode 2* it returns an $S \times S$ window to the *Transform HR to LR* unit, with S being the size of the PSF relating the LR to the HR grid. The EPW unit consists of $S-1$ FIFOs which are connected to $S \times S$ registers to form the processing window.

To compute the initial HR guess, the *Interpolation* unit executes bilinear interpolation. Each interpolated HR pixel is a weighted sum of the surrounding 2×2 LR pixels.

The *Transform HR to LR* unit multiplies each HR pixel of an $S \times S$ processing window with the PSF weight corresponding to its location in the window. The Ls pixels of the simulated LR sequence (Sect. 2.2) will be produced by subsampling the output of the convolution of the last HR approximation. All possible subpixel displacements should be covered, therefore the HR pixels should 'move' in the FIFOs of the EPW unit one location at every cycle. This poses a minimum in the number of cycles of every iteration. This minimum will be equal to the number of HR pixels.

The *HR Pixel Refinement Unit* includes parallel processing branches each one of them associated with a LR frame. These parallel branches meet at a final adder, which corresponds to the external summation in Eq. 2, to produce the refined version of the HR pixel which is currently under process.

3.3 Data Re-use and Maximum Performance

To maximize data re-use every HR and Lo pixel are read from the corresponding RAM only once and remain on-chip until all the processing associated with them is over. Also, for maximum performance, one iteration requires the minimum number of cycles imposed by the HR convolution (Sect. 3.2). To achieve this, the EPW unit, which produces the processing window for convolution, is designed to produce the synchronization control signals for the entire system. When a HR pixel is first brought on-chip it is 'pushed' into the FIFOs of the EPW. When it is no longer needed by the EPW it will be the input of the next level of processing, that is the *HR Pixel Refinement Unit* unit. When this happens, all the LR pixels influenced by the particular HR pixel, both actual (Lo) and simulated (Ls), should be available on-chip.

3.4 On-chip Memory

The units *Buffering of Simulated LR Frames* and *Buffering of Observed LR Frames* of Fig. 4 include the Ls and Lo groups of line-buffers, respectively. In order to achieve a throughput of one HR pixel per cycle, at every cycle all Ls and Lo buffers of all LR frames are accessed in parallel, while new data is brought in. Therefore, every group contains a separate buffer for every LR frame. These buffers only get updated when their content will not be used anymore at the current iteration. The width of the Ls buffers is equal to that of the LR frames. The Lo buffers are made wider, to surpass the limited memory bandwidth of the LR RAM. Specifically, the used Lo buffers are twice as wide as the LR frames and are written using a poling scheme.

4 Results

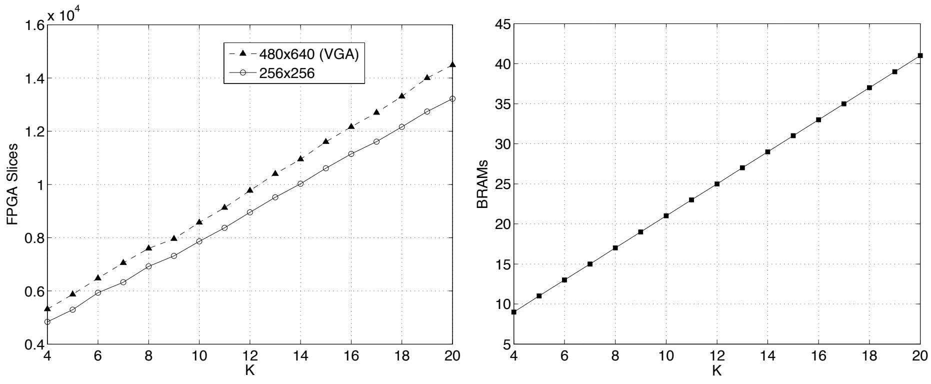
4.1 Implementation Requirements

The design was implemented on a Celoxica RC300 board using the DK5 Handel-C compiler, and was placed and routed using Xilinx ISE v.9.1. The RC300 board hosts a Xilinx Virtex-2 FPGA and on-board ZBT SRAMs. The operating frequency of the design on RC300 is 60 MHz. To meet real-time requirements the system should achieve 25 fps. The required number of cycles is: $C = reset\_cycles + M_l \times N_l + M_h \times N_h \times Iterations + [N_h \times (S-1) + S] + Latency$, where M_h (M_l) and N_h (N_l) denote the number of rows and columns of the HR (LR) frame. Thus, C depends on the image size and on the number of LR frames (K) which contributes in $Latency$ by $\lceil log_2(K+1) \rceil$, *i.e.* the latency of the final adder of the *HR Pixel Refinement* unit. For $K \in [8, 15]$ the number of maximum iterations of the IBP leading to 25 fps is given in Table 1.

$M_h \times N_h$	64×64	128×128	256×256	240×320	512×512	480×640	1024×1024
Iterations	585	146	36	31	8	7	2

Table 1. *Iterations for real-time performance for different HR sizes.*

The number of FPGA slices is mainly affected by K and does not significantly vary for different image sizes, as Fig. 6(a) demonstrates for 256×256 and 480×640 HR size. The number of BRAMs equals $(S-1) + K \times 2$, as $(S-1)$ BRAMs are used by the EPW unit, and K are occupied by each group of LR line-buffers (Fig. 6(b)).



(a)

(b)

Fig. 6. The number of FPGA resources increases linearly with the number of LR frames (K). (a) Number of FPGA slices. (b) Number of BRAMs. The number of BRAMs is independent of the image sizes reported in Table 1.

4.2 Performance Evaluation

The performance of the system has been evaluated under two different scenarios. The first one is concerned with the classic SR problem where a sequence of shifted LR frames is used as the input to produce a HR output. The second deals with the motion deblurring of a moving object, presenting the SR results based on time samples read from a LR motion area. To incorporate motion estimation errors in the simulation process, the OpenCV Lucas & Kanade optical flow [7] and Shi & Tomasi good feature extraction [9] algorithms were used in both scenarios to calculate the motion vectors. The calculated motion vectors were inserted in the SR system.

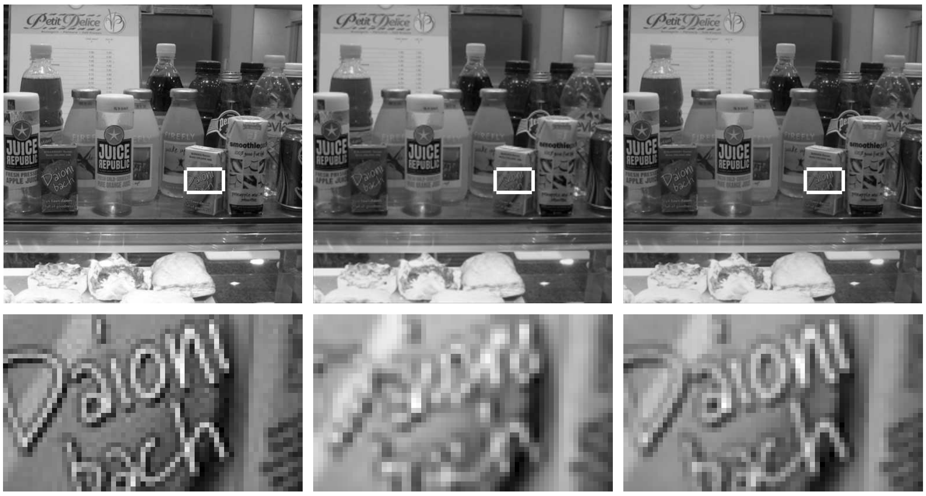
In the first experiment, a 512×512 natural image was used (Fig. 7(a)) and a sequence of 8 shifted 256×256 LR images was generated. The LR sequence was synthetically produced by first using randomly generated displacements to move the original image on the HR grid and then applying a 2×2 spatial blur kernel on that HR sequence.

The produced LR sequence was used as input to the proposed system. Fig. 8(a) shows the decrease in the Root Mean Square Error (RMSE) as the iterations of the algorithm proceed. The vertical line indicates the number of iterations which complies with real-time requirements for the given image size (Table 1). The results corresponding to the 8 bit rounding *of the output of every iteration* derive from the FPGA implementation of the algorithm. Apart from those, Matlab results are reported for the following scenarios: floating point version, floating point bicubic interpolation of the reference frame, 8 bits truncated, 9 bits truncated and 9 bits rounded (the last three are bit-accurate models). Fig. 8(a) illustrates that for large image sizes that impose a small number of iterations for real-time performance, the 8 bit rounding scenario gives outputs of similar quality as both larger word-lengths and the floating point SR, clearly prevailing against ‘8 bits truncated’. The detail images of Fig. 7 show the higher quality obtained by the FPGA implementation, after the number of iterations allowed for real-time performance, compared to floating point bicubic interpolation.

In the second experiment, a motion area employing pixels of 2×2 HR is considered, which produces 4 time samples during the HR integration. The size of the HR frame is 240×320. To increase the robustness of the system a neighborhood of 2 LR frames at each side of the integration interval is considered, so 8 LR frames are used in total.

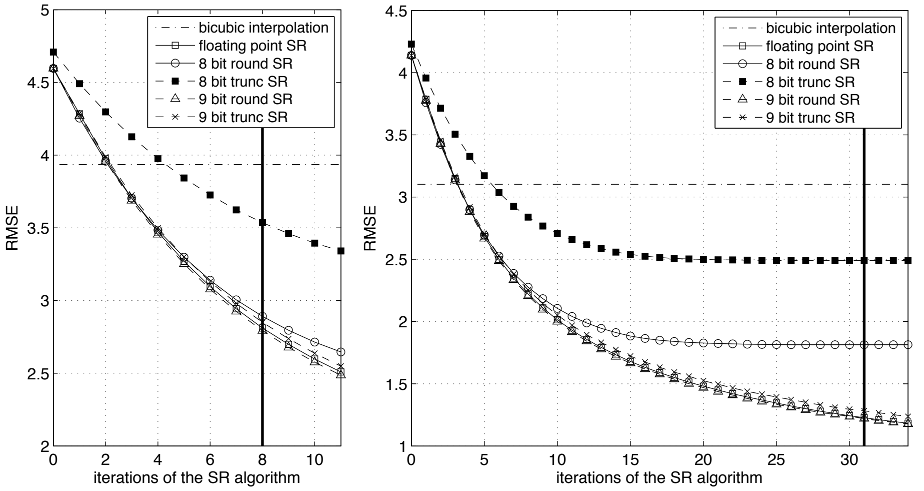
If very fast motion is involved (as in Fig. 1(b)), the LR frames are blurred themselves. To incorporate this intra-LR-frame motion, we first generated a dense HR sequence of 32 frames, using random HR displacements, and then created the LR motion blurred sequence in two steps. First we averaged groups of 4 successive frames and produced sequence A, with LR pixel temporal resolution and HR pixel spatial resolution. This would be the output of an ideal but unrealistic sensor that combines LR temporal resolution with HR spatial resolution. A 2×2 PSF was then applied on sequence A to produce the actual LR sequence.

The desired output belongs to sequence A and is shown in Fig. 9(a). Note how close the detail of the reconstructed output presented in Fig. 9(d) is to Fig. 9(a), as opposed to Figures 9(b) and 9(c). The system can be easily modified to accommodate high precision in the pixels, which is required for further improvement in the quality. This is useful when a smaller frame size is considered and, therefore, more iterations can be performed (Fig. 8(b)).



(a) (b) (c)

Fig. 7. (a) Ground-truth reference frame (*i.e.* the real-world reference frame without any degradation). (b) Floating point bicubic interpolation of the reference LR frame. (c) Reconstructed frame: Hardware output after 8 iterations (*i.e.* the number of iterations leading to real-time performance for $M_h \times N_h = 512 \times 512$).



(a) (b)

Fig. 8. RMSE as a function of the number of iterations of the IBP. The solid vertical line indicates the number of iterations allowed to obtain 25 fps for the given HR frame size ($M_h \times N_h$). (a) Experiment 1: $M_h \times N_h = 512 \times 512$ (b) Experiment 2: $M_h \times N_h = 240 \times 320$

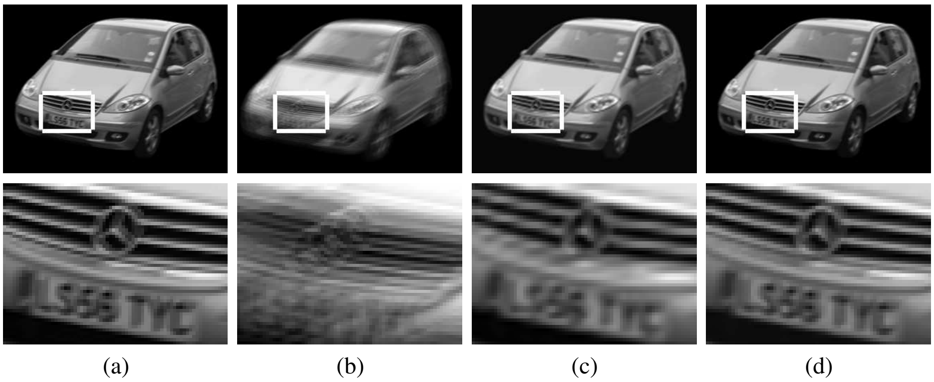


Fig. 9. (a) Ideal frame with HR spatial resolution and LR temporal resolution. This is the output of an ideal sensor that combines HR spatial resolution with LR integration time. (b) Motion-blurred output produced if the motion area had HR pixels. (c) Floating point bicubic interpolation of the reference LR frame. (d) Reconstructed frame for a motion area with LR pixels: Hardware output after the number of iterations leading to 25 fps. Using FPGA-based SR, the ideal combination of HR spatial resolution and LR temporal resolution is achieved.

5 Conclusions and Future Work

In this paper an FPGA-based system that forms areas of large pixels to cure motion blur was proposed. To compensate for the low spatial resolution of such areas FPGA-based SR is used. The reconstructed frame is of similar quality as the output of an ideal sensor with HR spatial resolution but LR temporal resolution, thus surpassing the fundamental trade-off between space and time. Future work includes the quantification of the scaling of the pixel size of the motion areas with the magnitude of motion, the use of bicubic interpolation as the initial estimation for faster convergence, and the implementation of the motion estimation, blur detection and background extraction blocks as well on FPGA.

References

1. A. E. Gamal and H. Eltoukhy, "CMOS image sensors," *IEEE Circuits & Devices Magazine*, vol. 21, no. 3, pp. 6–20, May-June 2005.
2. J. Farrell, F. Xiao, and S. Kavusi, "Resolution and Light Sensitivity Tradeoff with Pixel Size," in *SPIE Electronic Imaging '06 Conference*, vol. 6069, Feb. 2006, pp. 211–218.
3. E. Shechtman, Y. Caspi, and M. Irani, "Space-Time Super-Resolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 531–545, April 2005.
4. M. Ben-Ezra and S. K. Nayar, "Motion-based motion deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 689–698, June 2004.
5. T.G. Constandinou, P. Degenaar, and C. Toumazou, "An Adaptable Foveating Vision Chip," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006, pp. 3566–3569.
6. http://www.foveon.com
7. J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker - Description of the algorithm," Intel Corporation, Microprocessor Research Labs, Part of OpenCV Documentation, http://sourceforge.net/projects/opencvlibrary/.
8. M. Irani, and S. Peleg, "Improving Resolution by Image Registration," *CVGIP: Graphical Models and Image Proc*, vol. 53, no. 3, pp. 231–239, May 1991.
9. J. Shi, and C. Tomasi, "Good Features to Track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, June 1994, pp. 593–600.