Robust Real-Time Super-Resolution on FPGA and an Application to Video Enhancement

MARIA E. ANGELOPOULOU, CHRISTOS-SAVVAS BOUGANIS, PETER Y. K. CHEUNG, and GEORGE A. CONSTANTINIDES
Imperial College London

The high density image sensors of state-of-the-art imaging systems provide outputs with high spatial resolution, but require long exposure times. This limits their applicability, due to the motion blur effect. Recent technological advances have lead to adaptive image sensors that can combine several pixels together in real time to form a larger pixel. Larger pixels require shorter exposure times and produce high-frame-rate samples with reduced motion blur. This work proposes combining an FPGA with an adaptive image sensor to produce an output of high resolution both in space and time. The FPGA is responsible for the spatial resolution enhancement of the high-frame-rate samples using super-resolution (SR) techniques in real time. To achieve it, this article proposes utilizing the Iterative Back Projection (IBP) SR algorithm. The original IBP method is modified to account for the presence of noise, leading to an algorithm more robust to noise. An FPGA implementation of this algorithm is presented. The proposed architecture can serve as a general purpose real-time resolution enhancement system, and its performance is evaluated under various noise levels.

Categories and Subject Descriptors: B.6.1 [**Logic Design**]: Design Styles—*Parallel circuits*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Algorithms implemented in hardware, Gate arrays*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Reconfigurable computing, FPGA, super-resolution, motion deblurring, smart camera, real time

ACM Reference Format:
Angelopoulou, M. E., Bouganis, C.-S., Cheung, P. Y. K., and Constantinides, G. A. 2009. Robust real-time super-resolution on FPGA and an application to video enhancement. ACM Trans. Reconfig. Techn. Syst. 2, 4, Article 22 (September 2009), 29 pages.
DOI = 10.1145/1575779.1575782. http://doi.acm.org/10.1145/1575779.1575782.

Authors' address: M. E. Angelopoulou, C.-S. Bouganis, P. Y. K. Cheung, and G. 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.

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 (Figure 1(a)) [Gamal and Eltoukhy 2005]. Two types of resolution determine the quality of information collected by the sensor: spatial and temporal resolution.

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. As a result, a longer exposure time is required to achieve an adequate signal to noise ratio [Gamal and Eltoukhy 2005; Farrell et al. 2006; Chen et al. 2000].

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, that is, 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 Figure 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 article, LR denotes low spatial resolution and, thus, high temporal resolution image samples, while HR refers to high spatial and low temporal resolution. Also, S_h denotes the width of the elementary pixel of the sensor, which corresponds to resolution HR.

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 [Shechtman et al. 2005]. The main strength of the algorithm in Shechtman et al. [2005] 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 [Zitová and Flusser 2003; Brown 1992]. 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 Ben-Ezra and Nayar [2004], the proposed system consists of an HR and an 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 and focuses on solving the blur caused by the undesired global motion due to

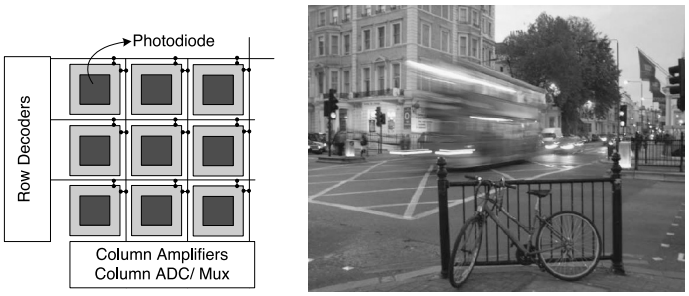

Fig. 1. (a) A hypothetical 3 × 3 CMOS image sensor. (b) A moving bus as opposed to a still bike. The first creates motion blur, whereas the second is fully captured.

camera shaking. The proposed system uses either two separate image sensors or a sensor with an 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 individual 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. As a result, the LR integration time is also fixed. Therefore, the quality of the outputs of the LR detector decreases as motion becomes faster, and beyond a certain threshold of motion magnitude, the LR samples are considerably blurred themselves. Hence, to reduce motion blur to a desired extent, the integration time of the LR device should adapt to the motion magnitude and decrease for faster motion. This issue is not resolved by the system proposed in Ben-Ezra and Nayar [2004].

Recent advances in imaging technology have produced sensors no longer subject to the constraint of time and space invariant pixel size [Foveon, Inc. 2007; Constandinou et al. 2006]. Elementary pixels can be grouped together to form a larger pixel where and when necessary. Inspired by these advances, this work proposes the marriage of an FPGA and an adaptive image sensor, aiming at producing an output of high resolution both in space and time. In Angelopoulou et al. [2008a], we investigate different ways of configuring the sensor in order to maximize the raw information collected from the environment, and propose appropriate methods to process that information and enhance the final output. A super-resolution and a deconvolution-based approach are evaluated, and the super-resolution approach is proven to be more efficient in dynamic regions. In this approach, uniform LR areas are formed on the motion regions and the LR pixel size is adapted according to local motion magnitude. In Angelopoulou et al. [2008b], an efficient FPGA architecture is proposed for the implementation of the resolution enhancement module of the Super-Resolution (SR) algorithm based on the Iterative Back Projection algorithm, and its performance is investigated. The proposed architecture can also be used as a general-purpose SR hardware block.

Our results reported in Angelopoulou et al. [2008b] present some initial performance figures; however, in this article we extend these results as follows:

(1) The noise in the image samples is taken into consideration and the SR block is modified to account for such noise, leading to a more robust system. The reconstruction quality is evaluated under different noise levels.
(2) A thorough investigation is carried out on how the system performance is affected by different decisions and parameters, such as the number of LR samples in the SR, the initialization of the SR iterative scheme, and the word-length of the data path.

The structure of the article is as follows. Section 2 presents the forward model, which describes the generation of the low-resolution output of an imaging system. Section 3 first discusses the possibilities offered by adaptive image sensors. It then presents the architecture of the proposed FPGA-based video-enhancement system, which benefits from such possibilities. Finally, it focuses on the spatial enhancement block, introducing SR and the Iterative Back Projection algorithm in particular, and discusses how the reconstruction algorithm can account for the presence of noise to produce a more robust system. Section 4 describes the FPGA implementation of the SR block. In Section 5, the implementation requirements are discussed, and the system performance is evaluated, considering different noise levels and system parameters. Finally, Section 6 concludes the article.

2. THE OBSERVATION MODEL

The proposed FPGA architecture deals with the problem of reconstructing the SR output based on LR samples. Before looking into that problem, let us briefly describe the forward model that forms these samples [Farsiu et al. 2004; Park et al. 2003]. This is known as the *observation model*. The LR degradation channel associated with each LR frame comprises a series of degradations, as shown in Figure 2. The first stage involves the atmospheric blur. A group of images that are largely affected by this type of degradation are astronomical images. This work deals with the stages of the observation model that follow the atmospheric blur. These are: the motion blur, the spatial blur, and the additive noise on the pixels of the sensor.

The effect of motion blur can be represented by a convolution with the motion point spread function (*PSF*). When the cause of motion is the shaking camera, as in Figure 2(c), this convolution spans the entire frame. On the other hand, if the cause is a moving object, as in Figure 1(a), the convolution spans only the part of the scene affected by that motion.

In the next degradation stage, the high-resolution grid of pixels is spatially subsampled. This subsampling is done by the LR pixels of the image sensor, and thus this type of blur is known as *camera pixel blur*. The generation of each LR pixel can be thought of as calculating a weighted average of all the pixels of the high-resolution input that topologically correspond to that LR pixel; therefore, applying a 2-D PSF on the high resolution pixel neighborhood. The 2-D Gaussian function is widely accepted as an appropriate sensor PSF,

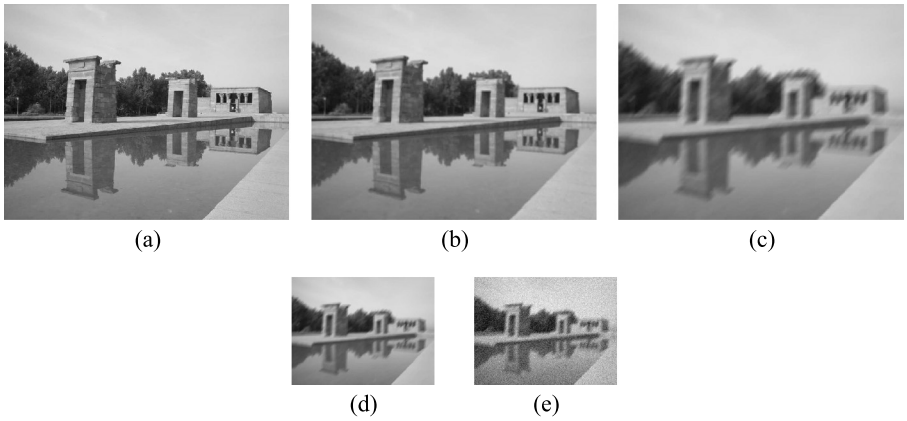

Fig. 2. The degradation stages of the observation model that describes the generation of the LR output. (a) High resolution input; (b) Atmospheric blur; (c) Motion blur; (d) Camera pixel blur (subsampling of the original pixel grid); (e) Additive system noise.

as it resembles the pixel's sensitivity: being high in the middle and decreasing towards its borders with a Gaussian-like decay.

The final degradation stage involves technology-related non-idealities of the given sensor, which is usually modeled by an additive noise factor that reduces the pixel signal fidelity [Tian et al. 2001; Mendis et al. 1997]. The level of additive noise determines the illumination range that can be detected by the sensor. The quality of the sensor's outputs, which are produced within that range, is quantified using the signal-to-noise-ratio (SNR): the ratio of the signal power to the noise power.

3. SURPASSING THE FUNDAMENTAL TRADE-OFF: OUR PROPOSAL

3.1 Dynamic Configuration of an Adaptive Image Sensor

The state of the art in imaging technology has produced sensors that are no longer subject to the constraint of time and space-invariant pixel size [Foveon, Inc. 2007; Constandinou et al. 2006]. Elementary pixels can be grouped together to form larger pixels that produce high-frame-rate samples. 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 on the motion regions, and execute online, real-time video enhancement.

Let S_h denote the size of the elementary pixel of the sensor, corresponding to resolution HR, which is 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 therefore, 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

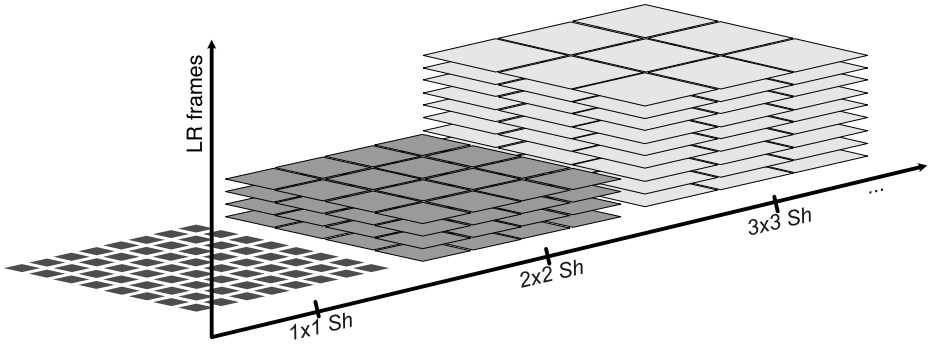

Fig. 3. LR time samples produced during the HR integration for different configurations of an adaptive image sensor.

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 in the 3-D space should remain $m \times n$ for the given area. At one end, there are the regions without motion that should be covered with HR pixels. Therefore, on these regions the distribution is $m \times n \times 1$, where m is on the x axis of the 2-D plane, n is on the y axis of the 2-D plane, and 1 is on the time axis, which gives the number of LR time samples that are produced during the HR integration. This is the case that is demonstrated in the first configuration of the sensor of Figure 3. At the other end, there is the configuration $1 \times 1 \times (m \times n)$, which would occur if all the available pixels were grouped together to form one large pixel. Thus, if the pixel size of area Q equals $2 \times 2\ S_h$, the LR spatial resolution is four times lower than the HR resolution, while the temporal resolution is four times higher. In other words, four LR time samples are produced for Q during the HR integration, as demonstrated in the second configuration of the sensor of Figure 3. If the spatial relation is 3×3, 9 LR samples are produced, as shown in the third configuration of Figure 3.

3.2 Combining an FPGA with an Adaptive Image Sensor

The adaptive image sensor should be configured in a manner that maximizes the raw information collected from the environment. Once this information is captured, it should be further processed to reconstruct a final output of both high temporal and high spatial resolution. In addition, the entire system should operate in real time, at least 25 fps, to achieve real-time capturing of the scene. Such throughput requirements render software processing inadequate, due to the high computational complexity associated with the required pixel-level processing, which scales with the number of LR samples. By exploiting the parallelism, pipelining, and data reuse possibilities offered by reconfigurable hardware, these objectives are feasible, as will be explained in the sections that follow. The role of the FPGA is twofold, as illustrated in Figure 4. It processes the raw information in real time and, also configures the adaptive sensor in a way that maximizes the raw information, according to the collected data.

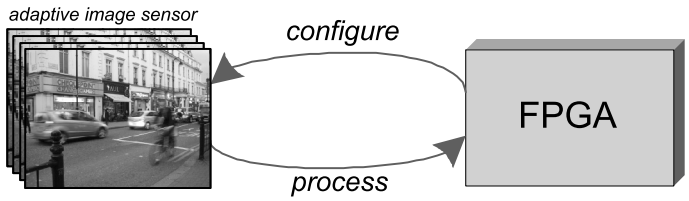

Fig. 4. Bidirectional interaction between FPGA and adaptive image sensor.

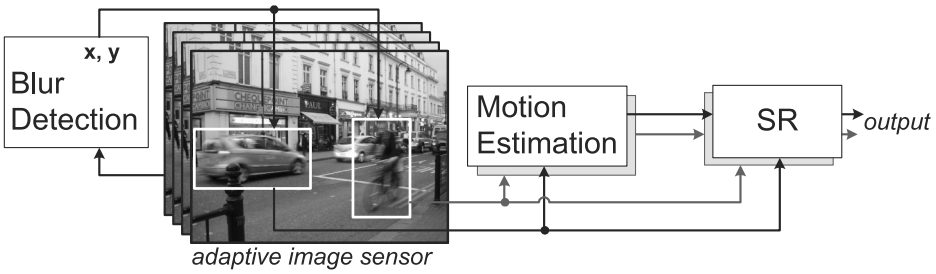

Fig. 5. The proposed FPGA-based system that employs an adaptive image sensor to execute online video enhancement. LR areas are formed on the motion regions.

An overview of the proposed video enhancement system is shown in Figure 5. Motion areas are located on the frame and are configured to larger pixel sizes forming LR areas, whereas areas with slow motion or no motion are configured to HR pixels. During the HR integration, a sequence of LR frames with reduced blur is produced at every LR area. Each LR area is spatially enhanced using SR to estimate a high resolution frame based on the LR inputs [Irani and Peleg 1991]. Thus, motion-deblurring is locally executed on the dynamic regions of the scene. The focus of this article is on the implementation of the SR block that performs the spatial enhancement of the raw outputs. For issues associated with the sensor reconfiguration, the reader is referred to Angelopoulou et al. [2008a]. The *Blur Detection* and *Motion Estimation* blocks of Figure 5 are briefly discussed in the following paragraphs before we look into the SR block and its implementation on FPGA.

The *Blur Detection* block of Figure 5 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.

The *Motion Estimation* block of Figure 5 reads the sequence of LR frames and returns the motion vectors: the displacements of selected features [Shi and Tomasi 1994] between each LR frame and the reference frame. Any frame of the LR sequence can be chosen as the reference frame. These displacements are then used by the *SR* block to enhance the spatial resolution. In the case of

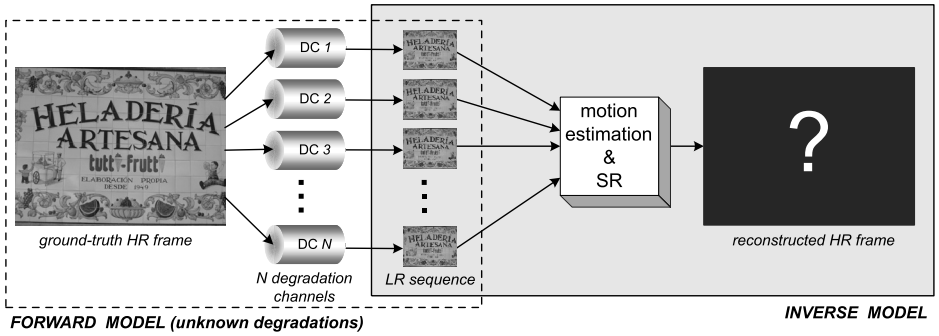

Fig. 6. The forward model consists of a series of unknown degradations related to the imaging system. The inverse model estimates those degradations to reconstruct the missing high-resolution data.

a moving object, the pixels of the static background do not participate in the SR process. The background pixels are identified based on the information from the motion vectors. The spatial resolution and the frame-rate of the system's output are those of the HR sequence, where motion blur has been removed in the LR areas.

3.3 Super-Resolution

The forward model, which describes the generation of LR samples at the output of an imaging system, was presented in Section 2. The parameters of the forward model are approximated and used by the inverse model, which is algorithmically formed to reconstruct the missing high-resolution data. In the high-level diagram of Figure 6, the forward model is succeeded by the inverse model, since the outputs of the former are the inputs of the latter. In the forward model, every LR sample, M, is produced individually after a series of degradations, which are not known but can be estimated. These degradations are presented by the individual LR channels, denoted by DC M in Figure 6. The inverse problem uses these LR samples as inputs and approximates the related degradations, so as to apply the inverse procedure and reconstruct the missing high-resolution data. The undegraded frame at the input of the forward model (Figure 6) comprises the ideal high-resolution information. This ideal frame is known as the *ground-truth frame*. Since the parameters of the forward model are not precisely known and can only be approximated, it would be naive to expect the exact ground-truth frame at the output of the inverse model. In the current section, the forward model will be mathematically stated, and the inverse problem will be formulated.

The forward model of generating LR pixels is shown in Figure 7. 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 to the formation of the particular LR pixel form a Gaussian kernel—the 2-D PSF shown in Figure 7. As mentioned in Section 2, the 2-D

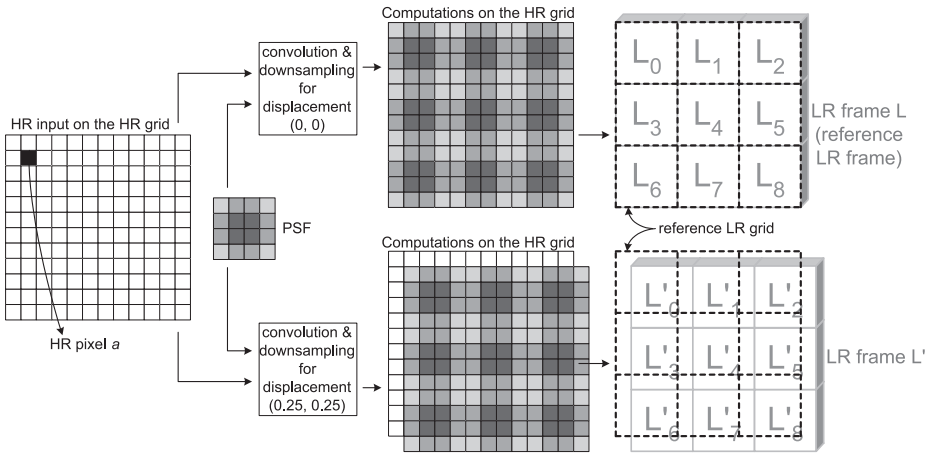

Fig. 7. The formation of the LR output presented mathematically. A 4 × 4 PSF is employed. Two simulated LR frames with displacements (0, 0) and (0.25, 0.25) are produced.

Gaussian function closely resembles the pixel's sensitivity, which is high in the middle and decreases towards the borders with a Gaussian-like decay. Every LR pixel can thus be expressed as a weighted sum of HR pixels, and the following linear system of equations is formed:

$$A\vec{h} = \vec{l}, \tag{1}$$

where $\vec{h}$ and $\vec{l}$ 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 $\vec{h}$. The HR grid on which reconstruction will occur is the HR grid underlying the LR reference grid (Figure 7). Thus, $\vec{h}$ 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 (Figure 7) by integer LR pixel units, they would all give the same set of equations since the same groups of HR pixels would form their LR pixels in the same manner. Therefore, for an LR frame to uniquely contribute in the system of Equation 1, it should be shifted by subpixel units on the LR reference grid compared to the other LR frames. However, although in theory these statements are true, in practice LR frames with the same integer displacements may give different sets of equations. This is partly due to additive noise present in the LR samples (Section 2) and partly due to errors in the motion estimation procedure [Bouguet 2002]. Therefore, in practice it is preferable to consider many LR frames, even if their displacements overlap.

The SR methods found in the literature solve the SR problem either in the spatial or in the frequency domain [Park et al. 2003; Baker and Kanade

2002; Farsiu et al. 2004]. In this work, a spatial domain method is adopted. 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, there is no need for long word-lengths in hardware implementations. Among the spatial domain methods, the iterative back projection (IBP) [Irani and Peleg 1991] approach was selected because of its hardware-friendly characteristics. Instead of solving Equation 1 for $\vec{h}$, 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 reuse, as will be demonstrated in Section 4.

3.4 Iterative Back Projection (IBP)

IBP employs an iterative refinement scheme on the HR grid, starting with an initial high-resolution approximation such as the interpolation of the reference LR frame. Then, at every iteration of the algorithm the forward model of Figure 7 is applied to the current high-resolution 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 sequences, by refining the high-resolution estimation.

Let Lo^k denote the kth observed LR frame and Ls_i^k denote the corresponding simulated LR frame at the current iteration, i. As the iterations of the IBP algorithm succeed one another, each Ls_i^k converges to the corresponding Lo^k. The error function that is iteratively minimized contains the total error from all LR frames and is the following:

$$e^{(i)} = \sqrt{\sum_{k=0}^{K-1} \sum_{(x_l, y_l)} (Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l))^2}, \tag{2}$$

where (x_l, y_l) denotes the LR coordinates, and K is the number of LR frames.

All of the observed LR pixels and the corresponding simulated LR pixels that are influenced by a particular HR pixel contribute to 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 (Figure 7), pixel L_0 of frame L participates with a weight proportional to $PSF(1, 1)$, whereas the weight of L'_0 of L' will be proportional to $PSF(0, 0)$. At iteration i, every pixel of the current high-resolution approximation, H_i, is refined as follows:

$$H_{i+1}(x_h, y_h) = H_i(x_h, y_h) + \sum_{k=0}^{K-1} \sum_{(x_l, y_l) \in Y} (Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l)) \times W(k, x_l, y_l), \tag{3}$$

where (x_h, y_h) denotes the HR coordinates, Y is the set of LR coordinates of the pixels of Lo^k and Ls_i^k that are influenced by point (x_h, y_h), and W is the weight with which $Lo^k(x_l, y_l)$ and $Ls_i^k(x_l, y_l)$ contribute to the refinement of $H_i(x_h, y_h)$.

3.5 Increasing the System Robustness

As explained in Section 3.3, the error determining the quality of the SR algorithm is the error between the reconstructed output and the ground-truth frame. In the ideal situation of noise-free LR samples, it can be proven that if the error function of Equation 2 is iteratively minimized, this is also the case for the error between the SR output and the ground-truth frame. However, in a real-world scenario, the LR outputs of the imaging system always contain a certain level of noise. This is the additive noise factor, which comprises the final degradation stage of the observation model and is connected to technology-related non-idealities of the image sensor, as discussed in Section 2. In the presence of such noise, the error function of Equation 2 is still minimized, regardless of the noise level. However this is no longer true for the error between the reconstructed output at every iteration and the ground-truth frame. The reason for this is that the simulated LR frames gradually converge to the noisy LR samples, and therefore the final reconstructed output is affected by the presence of noise. As a result, the higher the noise levels in the LR samples, the more the SR output diverges from the ideal ground-truth frame.

As explained in Irani and Peleg [1991], averaging in Equation 3, all LR pixels related to a given HR pixel, already diminishes the effect of noise to a significant extent. Therefore, increasing the number of LR frames decreases the sensitivity of the updating process to noise. If a large number of LR frames is employed, LR pixels with extreme values, within the LR pixel group related to a given HR pixel, can be identified and not used in the reconstruction, since such outliers are most likely due to noise. However, in the particular application of SR in motion-deblurring, which is the main focus of this work, increasing the number of LR frames means widening the temporal neighborhood around the integration interval of interest (Section 3.3). By doing this, the correlation in time between the two ends of the LR sequence becomes low. This may increase the level of nonrigid deformations of the moving object, and as a result, may complicate the fusion of the relevant LR information.

To keep the number of LR frames within reasonable limits, so as to both avoid undesirable types of changes in the scene and execute efficient reconstruction, we have included in the expression of Equation 2, information about the statistics of the noise. This is done in a manner inspired by the projection onto convex sets (POCS) framework [Stern et al. 2001; Stark and Oskoui 1989; Youla and Webb 1982], in which the standard deviation of the noise of each LR frame is taken into account. In practice, this metric derives from the signal-to-noise-ratio (SNR) of the corresponding frame. The standard deviation of noise determines an interval in which the variation of LR pixel values is attributed solely to the presence of noise. Thus, if the difference between $Lo^k(x_l, y_l)$ and $Ls_i^k(x_l, y_l)$ falls within that interval, these pixels will not contribute to the

refinement of the corresponding HR pixel $H_i(x_h, y_h)$. Let $r_i(x_h, y_h, k, x_l, y_l)$ denote this difference:

$$r_i(x_h, y_h, k, x_l, y_l) = Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l), \tag{4}$$

then, Equation 3 is modified as follows:

$$H_{i+1}(x_h, y_h) = H_i(x_h, y_h) + \sum_{k=0}^{K-1} \sum_{(x_l, y_l) \in Y} u_i(x_h, y_h, k, x_l, y_l) \tag{5a}$$

$$u_i(x_h, y_h, k, x_l, y_l) = \begin{cases} (r_i(x_h, y_h, k, x_l, y_l) - \delta_0(k)) \times W(k, x_l, y_l), \\ \qquad if\ r_i(x_h, y_h, k, x_l, y_l) > \delta_0(k) \\ 0, \\ \quad if\ -\delta_0(k) \le r_i(x_h, y_h, k, x_l, y_l) \le \delta_0(k), \\ (r_i(x_h, y_h, k, x_l, y_l) + \delta_0(k)) \times W(k, x_l, y_l), \\ \qquad if\ r_i(x_h, y_h, k, x_l, y_l) < -\delta_0(k) \end{cases} \tag{5b}$$

where $u_i(x_h, y_h, k, x_l, y_l)$ denotes the contribution of each pair of LR frames, $Lo^k(x_l, y_l)$ and $Ls_i^k(x_l, y_l)$, in the refinement of HR pixel $H_i(x_h, y_h)$, and $\delta_0(k)$ denotes the standard deviation of the noise of Lo^k. In the ideal case of noise-free LR frames, $\delta_0(k) = 0$, for $k \in [0, K-1]$, and thus Equation 5 is simplified to Equation 3. Thus, Equation 3 can be considered as a special case of Equation 5, corresponding to the ideal scenario of noise-free LR samples.

Apart from the additive noise, another type of error affecting the reconstruction process is that of the motion vectors, which are generated by the motion estimation process and passed to the SR block. To deal with such errors, the error information at the output of the *Motion Estimation* block of Figure 5 [Bouguet 2002] is utilized by the SR block. Thus the different LR samples are weighted, and the contribution of those with large error values is decreased.

To further increase the robustness of the system, neighboring LR samples before and after the integration interval of interest contribute in SR with adjustable weights. This technique increases the available spatial information at the input of the SR block, since a larger number of different subpixel displacements on the underlying HR grid (Figure 7) is considered, leading to a better determined SR problem. In addition, using a larger number of LR samples reduces both the effect of errors related to some of the motion vectors and the effect of additive noise on the reconstruction output, as explained in the preceding.

4. ARCHITECTURE OF THE SR SYSTEM

Figure 8 shows an overview of the proposed system. For every new group of LR frames, produced during a particular HR integration interval (Section 3.2), an SR stage occurs. At the beginning of each SR stage an initial high-resolution 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

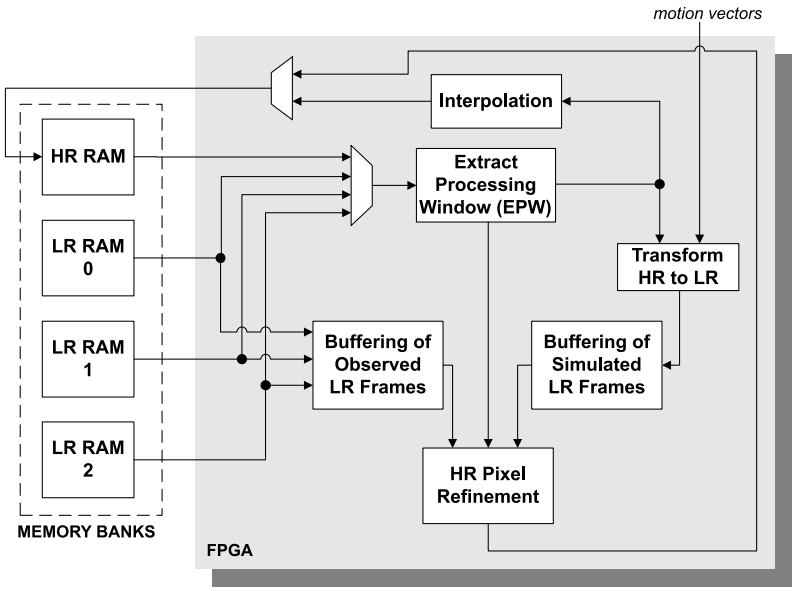

Fig. 8. Architecture overview.

HR integration interval) is processed, and so on. The rest of this section focuses on the description of the individual blocks of the proposed architecture. The motion vectors, describing the displacements of each LR frame with respect to the reference frame, are treated as inputs to the implemented SR system. It should be mentioned that the target system has four memory banks, each with a word-length of four bytes.

4.1 Off-Chip Memory Banks

4.1.1 *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, as explained in Section 3.5, in order to increase the robustness of the system, a number of neighboring LR frames are used in addition to those produced during the HR integration. Figure 9(a) demonstrates how the tasks of processing LR frames and writing new frames in the memory banks are scheduled in time, so as to include an LR frame neighborhood in the processing. In Figure 9(a), 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. To implement the scheduling of Figure 9(a) and execute online processing of the LR data, two memory banks need to be read in parallel, as Figure 9(b) illustrates for the case of a 2×2 PSF and four neighboring frames. For instance, in SR stage 1 (Figure 9(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

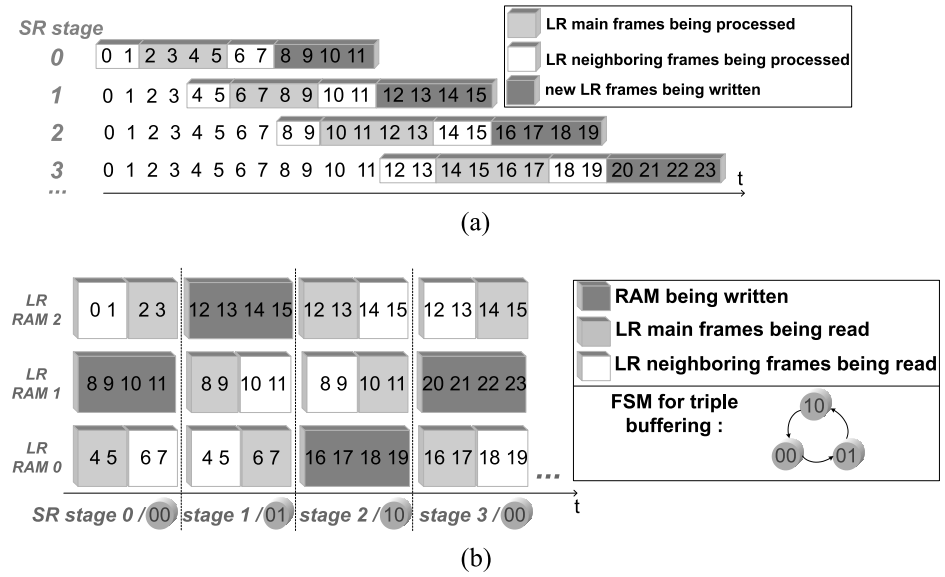

Fig. 9. The numbers correspond to the LR frames. (a) A sliding window indicates the group of frames processed during the current SR stage. While the processing occurs on these frames, a new group of four frames is written in the memory banks. (b) Triple buffering scheme applied on the LR RAMs.

in Figure 9(b). There are three possible configurations, each corresponding to a different combination of modes of the three LR RAMs, depending on which RAMs are being read and which is being written during the current SR stage. 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. Thus, as demonstrated in Figure 9(b), a particular configuration appears once in every three SR stages.

4.1.2 *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 eight bits. This allows storing HR pixels in groups of four in the 32-bit RAM, thus increasing the available memory bandwidth.

4.2 Individual Processing Units

The *Extract Processing Window* (EPW) unit of Figure 8 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, indicated by the different shades of grey in Figure 10(a). 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

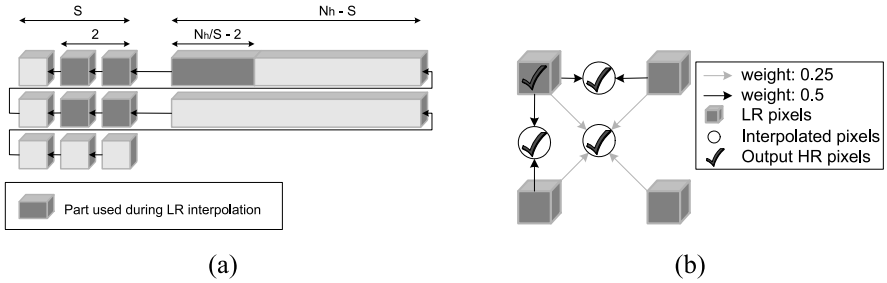

Fig. 10. (a) Extract Processing Window Unit. N_h denotes the number of columns of the HR frame, and S is the size of the PSF relating the LR to the HR grid. (b) Interpolation Unit.

LR to the HR grid. The EPW unit consists of $S-1$ FIFOs that are connected to $S \times S$ registers to form the processing window.

To compute the initial high-resolution estimate, the *Interpolation* unit executes bilinear interpolation. Each interpolated HR pixel is equal either to a weighted sum of the surrounding 2×2 LR pixels or to a raw pixel of this 2×2 group of LR pixels. This depends on the position of the interpolated HR pixel, as is demonstrated in Figure 10(b).

The *Transform HR to LR* unit multiplies each HR pixel of an $S \times S$ processing window with the weight corresponding to its location in the window. The Ls pixels of the simulated LR sequence (Section 3.3) will be produced by subsampling the output of the convolution of the last high-resolution approximation. Since all possible subpixel displacements should be covered, the HR pixels should move in the FIFOs of the EPW unit one location at every cycle. This determines the number of cycles that leads to maximum throughput. Therefore for maximum performance, the number of cycles per iteration should be equal to the number of HR pixels.

The *HR Pixel Refinement Unit* includes parallel processing branches, each one of them associated with an LR frame, as demonstrated in Figure 11. These parallel branches meet at a final adder, which corresponds to the external summation in Equation 3 or Equation 5, to produce the refined version of the HR pixel that is currently under process. In Angelopoulou et al. [2008b], the iterative process was executed without taking into account the noise statistics related to each LR frame, and thus Equation 3 was implemented. In the current work, Equation 5 is used instead. The hardware that implemented Equation 3 is modified to accommodate Equation 5, by including an extra multiplexer that executes the control logic of Equation 5b, leading to an algorithm that is more robust to noise.

4.3 Data Reuse and Maximum Performance

Each HR and observed LR pixel is read from the corresponding RAM only once and they remain on-chip until they are no longer needed. Thus, data reuse is maximized. Also, for maximum performance, one iteration requires the number of cycles imposed by the HR convolution (Section 4.2). To achieve this, the EPW unit, which produces the processing window for convolution, is

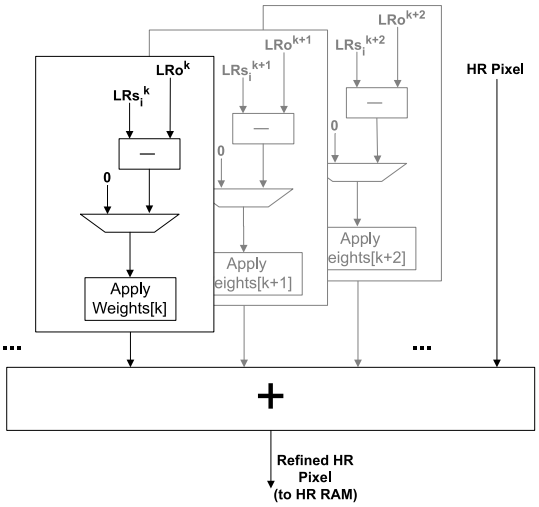

Fig. 11. The HR Pixel Refinement Unit. LRo^k and LRs_i^k denote the contribution of each pair of LR frames, Lo^k and Ls_i^k, in the refinement of the particular HR pixel.

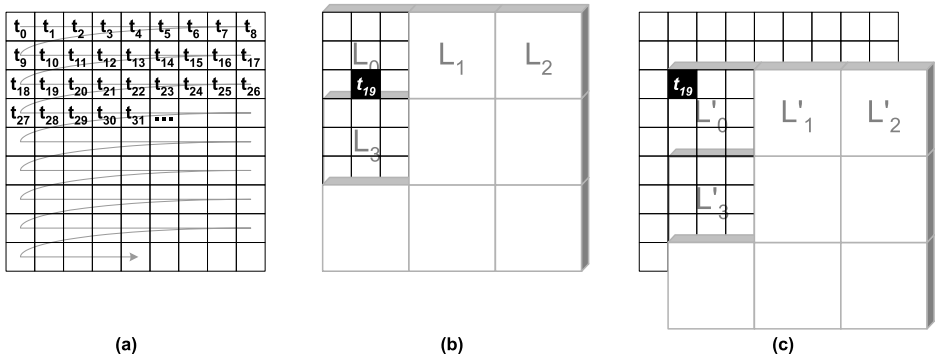

Fig. 12. Temporal Aspect of HR Grid (a) At every clock cycle the cursor moves one position on the HR grid indicating the currently processed HR pixel. (b) The reference LR frame. (c) An LR frame with displacement (2,1).

designed to produce the synchronization control signals for the entire system. When an 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*. When this happens, all the LR pixels influenced by the particular HR pixel, both observed (Lo), and simulated (Ls), should be available on-chip.

Since one high-resolution pixel is processed per cycle, it would be convenient to view the HR grid underlying the reference LR frame as a time-map of the current iteration. Imagine a cursor moving along the arrow of Figure 12(a) pointing at one HR pixel every cycle. This is the HR pixel that is currently under process. Therefore, HR pixel h_0, would correspond to time-slot t_0, and so

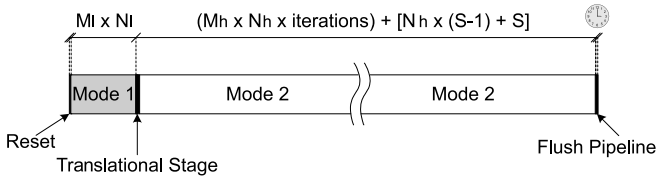

Fig. 13. Time diagram with execution cycles.

on. To find when, and also how much, every LR pixel will contribute in the refinement procedure, we can superimpose the LR grids of the LR frames on the time-map (Figure 12(b, c)). For instance, at t_{19} pixels, L_0 and L'_0 will both contribute to the refinement of HR pixel h_{19}, but with different weights, depending on their relative position with respect to h_{19}, as explained in Section 3.3.

The time diagram with the total number of cycles required for these specifications is shown in Figure 13. In Figure 13, M_h and N_h denote the number of rows and columns of the HR frame, M_l and N_l are the number of rows and columns of the LR frame, and S denotes the size of the PSF relating the LR to the HR grid. The two modes in Figure 13 are the two operating modes of the *Extract Processing Window* unit, which are described in Section 4.2.

4.4 On-chip Memory

The units *Buffering of Simulated LR Frames* and *Buffering of Observed LR Frames* of Figure 8 include the *Ls*-type and *Lo*-type groups of line-buffers, respectively, where *Ls* and *Lo* pixels are stored. In order to achieve a throughput of one HR pixel per cycle, at every cycle, pixels from all LR frames should be 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 no longer be used 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. The *Lo* buffers used are twice as wide as the LR frames, and are written using a polling scheme.

5. RESULTS

5.1 Implementation Requirements

The design targets a Celoxica ADMXRC4SX board. The DK5 Handel-C compiler was used, and the implementation was placed and routed using Xilinx ISE v.9.1. The ADMXRC4SX board hosts a Xilinx Virtex-4 FPGA. The operating frequency of the design on ADMXRC4SX is 80 MHz. The critical path of the circuit lies in the control block that implements the triple-buffering scheme that is illustrated in Figure 9. To meet real-time requirements, the system should achieve 25 fps. As shown in Figure 13, the required number of cycles for SR reconstruction is:

$$C = R + M_l \times N_l + M_h \times N_h \times I + [N_h \times (S-1) + S] + Latency, \qquad (6)$$

Table I. Iterations for Real-Time Performance for Different HR Sizes

$M_h \times N_h$	64 × 64	128 × 128	256 × 256	240 × 320	512 × 512	480 × 640	1024 × 1024
Iterations	780	195	48	41	11	10	2

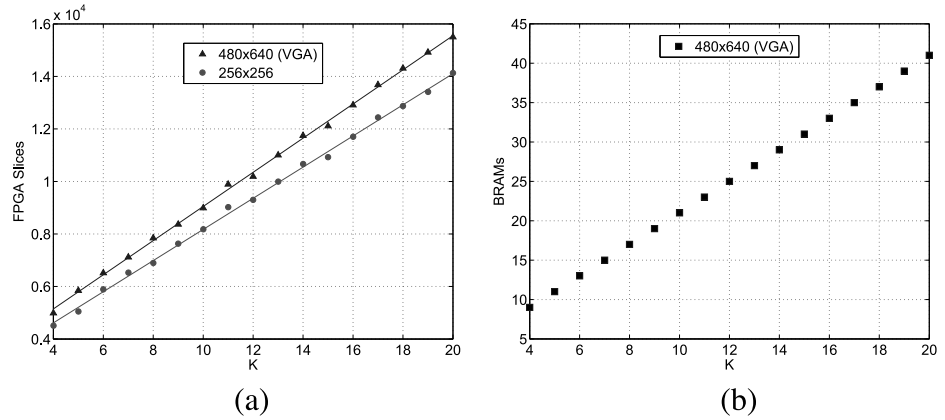

Fig. 14. The number of FPGA resources increases linearly with the number of LR frames (K). (a) Number of FPGA slices. (b) Number of BRAMs. The number of BRAMs is independent of the image size for the sizes reported in Table I.

where M_h (M_l) and N_h (N_l) denote the number of rows and columns of the HR (LR) frame, I denotes the number of iterations of the SR algorithm, R is the number of reset cycles, and S is the the size of the PSF relating the LR to the HR grid. Thus, C depends on the frame size and on the number of LR frames (K) that contribute to the *Latency* by $\lceil log_2(K+1) \rceil$—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 permitting 25 fps derives from Equation 6 and is given in Table I.

The number of FPGA slices is mainly affected by K and does not significantly vary for different frame sizes, as Figure 14(a) demonstrates for 256×256 and 480×640 HR sizes. Linear least squares fitting is applied on the samples of Figure 14(a), giving the following linear equations for the number of FPGA slices with respect to K:

$$S_{VGA} = 650 * K + 2546 \tag{7}$$

$$S_{256} = 593 * K + 2241, \tag{8}$$

where S_{VGA} and S_{256} denote the number of slices for HR sizes 480×640 (VGA size) and 256×256.

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. The graph of Figure 14(b) demonstrates the number of BRAMs for $S = 2$.

The linearity of the number of slices and BRAMs with the number of frames K, which is observed in Figure 14, is due to the parallel processing of the different LR frames. This has been discussed in Section 4. Thus, as is demonstrated in Figure 11, every LR frame is associated to a parallel processing

branch. Also, for each additional LR frame, two line-buffers are employed, one belonging to the *Ls*-type and the other to the *Lo*-type groups of buffers. This is required in order to access pixels in parallel from all of the LR frames and, thus, achieve a throughput of one HR pixel per cycle, as explained in Section 4.4.

5.2 Performance Evaluation

The performance of the implemented algorithm was investigated using different sets of evaluation parameters. The parameters are the level of noise in the LR samples, the length of the LR sequence, the type of initial approximation of the iterative scheme, and the chosen word-length of the system. The evaluation is done using semisynthetic data. In this way, the ground-truth frame is known and can be used as a reference to accurately evaluate the quality of the reconstructed output. Specifically, a real image, captured with a simple hand-held digital camera, has been shifted, blurred, downsampled, and contaminated with noise, to synthetically produce the LR sequences.

Two groups of experiments are presented. The first is concerned with the classic SR problem, where a sequence of shifted LR frames is used to reconstruct a high-resolution output. The second deals with the motion deblurring of a moving object, presenting the SR results based on time samples read from an LR motion area. To include motion estimation errors in the simulation, the OpenCV Lucas and Kanade optical flow [Bouguet 2002] and Shi and Tomasi [1994] good feature extraction algorithms are employed in both scenarios to calculate the motion vectors that are used by the SR block.

In all experiments, the LR frames have been contaminated with white Gaussian noise with various SNRs, ranging from 10 to 70 dB, with a step of 10 dB. The SNR level is defined according to the following expression:

$$SNR = 10 \log \frac{\sigma_f^2}{\sigma_n^2}, \tag{9}$$

where σ_f denotes the standard deviation of the noise-free image, and σ_n denotes the noise standard deviation.

The evaluation process was repeated for different numbers of LR samples. Specifically, 4, 8, and 12 LR frames were used. The output of every iteration, which comprises the input of the next iteration, was rounded and truncated to different bit-widths to investigate how this affects the reconstruction quality. The results corresponding to the 8 bit rounding and truncating schemes derive from the FPGA implementation of the algorithm. Apart from those, Matlab results are reported for the following scenarios: double precision floating-point version, 9 bits truncated, 9 bits rounded, 10 bits truncated, and 10 bits rounded (the last four are bit-accurate models). As for the initial approximation, bilinear interpolation was implemented on FPGA, whereas a bit-accurate Matlab model was developed for bicubic interpolation.

To demonstrate the benefits of SR, double precision floating-point bicubic interpolation was applied on the LR reference frame, to be compared with the SR output. This is the most elaborate type of interpolation that can be applied on

(a) (b)

Fig. 15. Ground-truth frame with HR spatial resolution and LR temporal resolution for the *drinks* sequence. This is the output of an ideal sensor that combines HR spatial resolution with LR integration time.

the output of a single LR channel, solely using information from that channel. The metric used to quantify the reconstruction quality is the root mean square error (RMSE) between the ground-truth frame and the reconstructed output. All results correspond to the implementation of Equation 5, with the exception of a comparative analysis between Equation 3 and Equation 5, which is presented for the second group of experiments to prove the increased robustness of Equation 5 with respect to noise.

5.2.1 *The Classic SR Problem.* In the first group of experiments, the natural image of Figure 15(a) was used to produce a group of LR samples, the *drinks* LR sequence. Random displacements were applied on the original 512×512 image and the displaced frames were blurred with a 2×2 kernel, rendering LR samples of size 256×256.

The graphs of Figure 16 demonstrate the RMSE values obtained when using 8 LR frames of the *drinks* sequence with different word-lengths and SNR values, after executing the number of iterations indicated in Table I for a 512×512 HR. The difference between these two graphs lies in the type of interpolation used as the initial guess: bilinear interpolation is employed for Figure 16(a) and bicubic for Figure 16(b). These graphs are very similar, which proves the good convergence properties of the algorithm of Equation 5. Bilinear interpolation is thus preferable, as it leads to similar results with lower computational cost.

The graphs of Figure 17 demonstrate the decrease in the RMSE as the iterations of the algorithm proceed. The solid vertical line indicates the number of iterations complying with real-time requirements. In all graphs, the SNR covers a range from 10 to 70 dB, with a step of 10 dB, while bilinear interpolation is used to initialize the algorithm. The graphs correspond to different data bit-widths and number of LR frames, as indicated in the figure. For the given frame size and corresponding number of iterations, the 8 bit rounding scenario

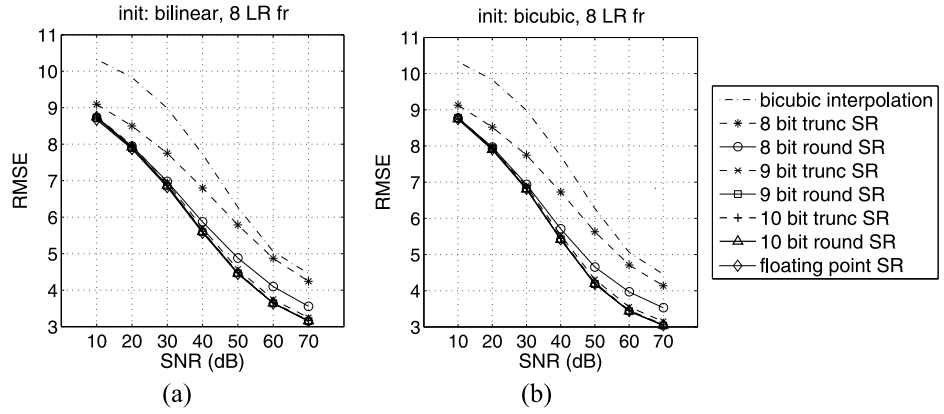

Fig. 16. RMSE values obtained for real-time SR when using 8 LR frames of *drinks* with different word-lengths and SNRs. (a) Bilinear interpolation used as initial approximation. (b) Bicubic interpolation used as initial approximation.

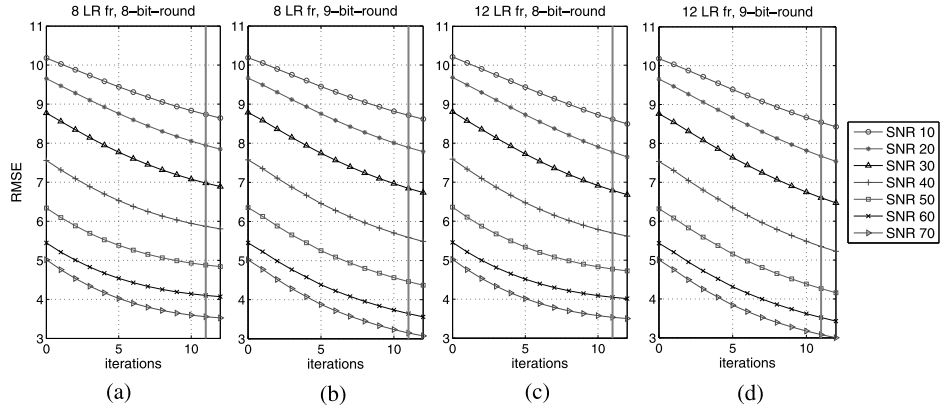

Fig. 17. RMSE as a function of the number of iterations of the IBP applied on the *drinks* sequence. The graphs correspond to different word-lengths and number of LR frames: (a) 8 frames and data rounded to 8 bits; (b) 8 frames and data rounded to 9 bits; (c) 12 frames and data rounded to 8 bits; and (d) 12 frames and data rounded to 9 bits.

with 8 LR frames, shown in Figure 17(a), gives outputs of similar quality to both larger word-lengths and longer LR sequences (Figure 17(b-d)).

In Figure 18, visual results of the reconstructed output are demonstrated for the set of parameters of Figure 17(a), after executing the number of iterations for real-time performance. Each row of Figure 18 corresponds to the SNR value indicated on the left. The part of the frame shown in Figure 18 corresponds to the ground-truth part of Figure 15(b). The first column of Figure 18 illustrates the output of floating-point bicubic interpolation applied on a single LR frame for a magnification factor of 2. The second column shows the real-time SR output. Using real-time FPGA-based SR, the combination of HR

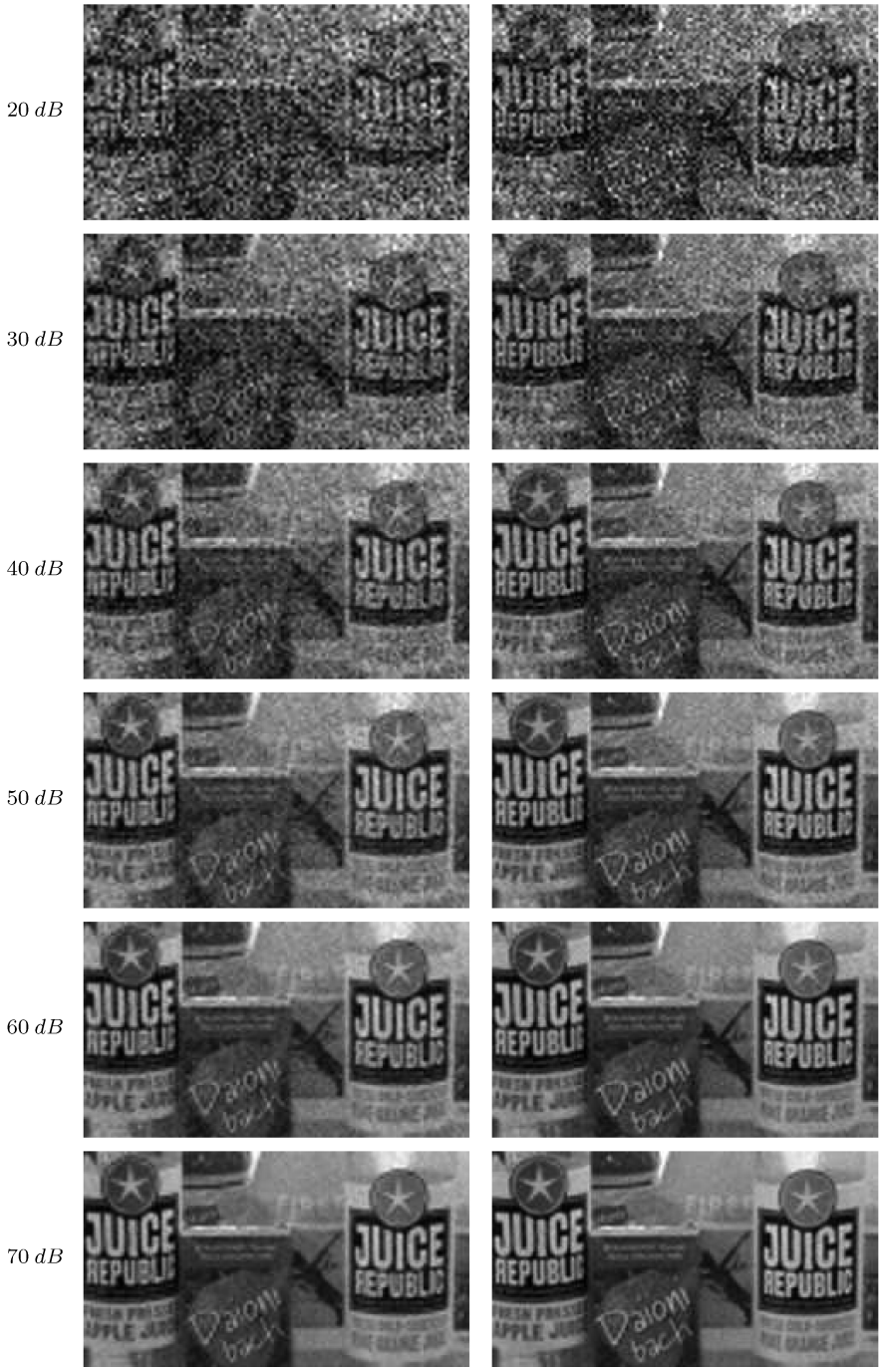

Fig. 18. Reconstruction of *drinks* for various SNRs. The first column shows the output of floating-point bicubic interpolation using a single LR frame, while the second one demonstrates the higher quality obtained by the real-time SR FPGA implementation.

(a) (b)

Fig. 19. Ground-truth frame for the *car* sequence: the output of an ideal sensor combining HR spatial resolution with LR temporal resolution.

spatial resolution and LR temporal resolution is achieved, leading to superior results.

5.2.2 *The Motion Deblurring Problem.* In the second experiment, a motion area employing pixels of 2×2 S_h is considered, which produces four time samples during the HR integration. If very fast motion is involved (as in Figure 1(b)), the LR frames themselves are blurred. To incorporate this intra-LR-frame motion, we first generated a dense HR sequence, using random HR displacements, and then created the LR motion blurred sequence in two steps. First we averaged groups of four 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, and Gaussian noise was added, to get LR sequences with SNR levels ranging from 10 to 70 dB. The desired output belongs to sequence A and is shown in Figure 19.

The graphs of Figure 20(a,b) describe exactly the same scenarios as those of Figure 16(a,b), but for the *car* sequence. The size of the HR frame is now 240×320 and thus 41 iterations can be executed, according to Table I. As in Figure 16(a,b), it is also observed here that, due to the good convergence properties of the algorithm, bilinear interpolation is preferable as it leads to similar results to bicubic interpolation but with lower computational cost.

The graphs of Figure 20(c-e) correspond to the 8-bit-round, 9-bit-round, and 10-bit-round scenarios for different lengths of the LR sequence. For a given word-length, as the SNR decreases, significant deviation of the RMSE values is observed when different lengths of the LR sequence are used.

In the following subsection, the motion deblurring experiment is employed for the comparison of Equation 5 with Equation 3, to demonstrate the increased robustness of Equation 5 with respect to noise.

5.2.3 *Robust SR in the Presence of Noise.* The graphs of Figure 21 demonstrate a comparison between the RMSE values obtained by Equation 3 (Figure 21(a–d)) and Equation 5 (Figure 21(e–h)), as the iterations of the algorithm succeed one another, for the *car* sequence. The solid vertical line indicates the iterations for real-time performance. In all graphs, bilinear interpolation is used as the initial estimate. Results are demonstrated for

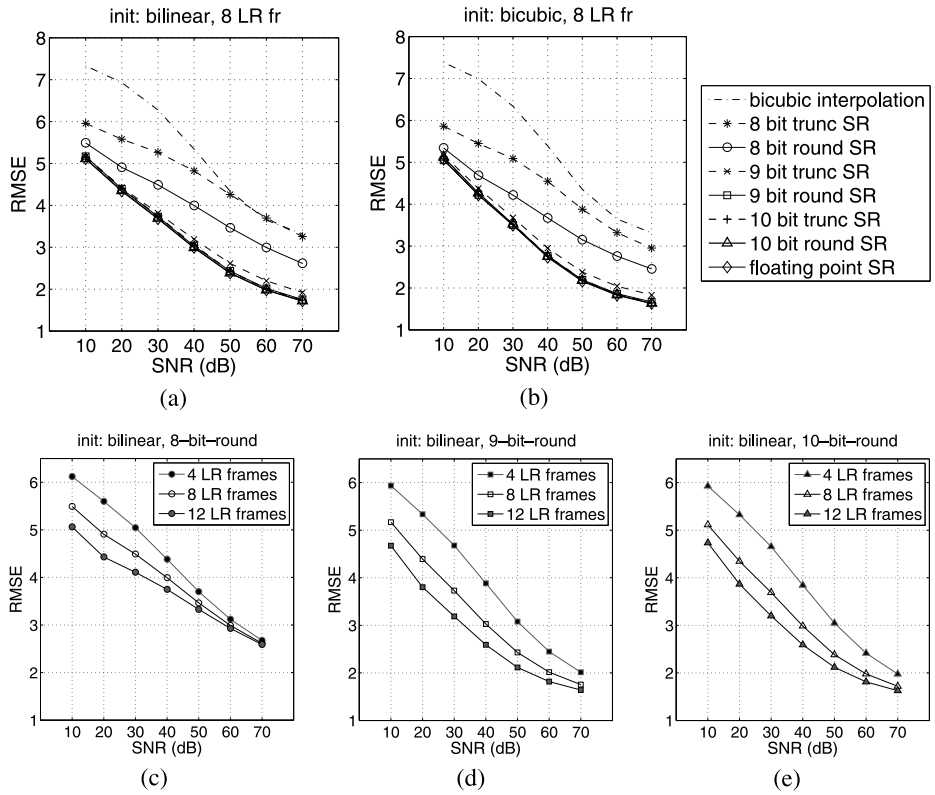

Fig. 20. RMSE values for real-time SR on the *cars* sequence. (a, b) Employing 8 LR frames, different word-lengths and SNR values, and using: (a) bilinear, and (b) bicubic interpolation as the initial guess. (c, d, e) RMSE values obtained for the given word-lengths and the indicated numbers of LR frames.

different data bit-widths, number of LR frames, and SNRs, as indicated in the figure.

As explained in Section 3.5, the original IBP algorithm of Equation 3 can be considered as a special case of Equation 5 corresponding to the ideal scenario of noise-free LR samples, since in that case $\delta_0(k) = 0$, for $k \in [0, K-1]$, and Equation 5 is simplified to Equation 3. The results reported in Angelopoulou et al. [2008b] indicate that in the noise-free situation, Equation 3, minimizes the error between the ground-truth frame and the reconstructed output, as the iterations proceed. Let us call this error *reconstruction error*. In the presence of noise, Equation 3 no longer minimizes the reconstruction error, as Figures 21(a-d) demonstrate. As explained in Section 3.5, this is due to the convergence of the simulated LR frames to the noisy LR samples, which results in the minimization of the error of Equation 2 but not of the reconstruction error. For low noise levels, such as $SNR = 70$ dB (Figures 21(a-d)), Equation 3 renders a behavior close to that reported in Angelopoulou et al. [2008b]. However, when the noise level increases, that is, the SNR decreases, the reconstruction error diverges more from the ideal behavior of the noise-free case.

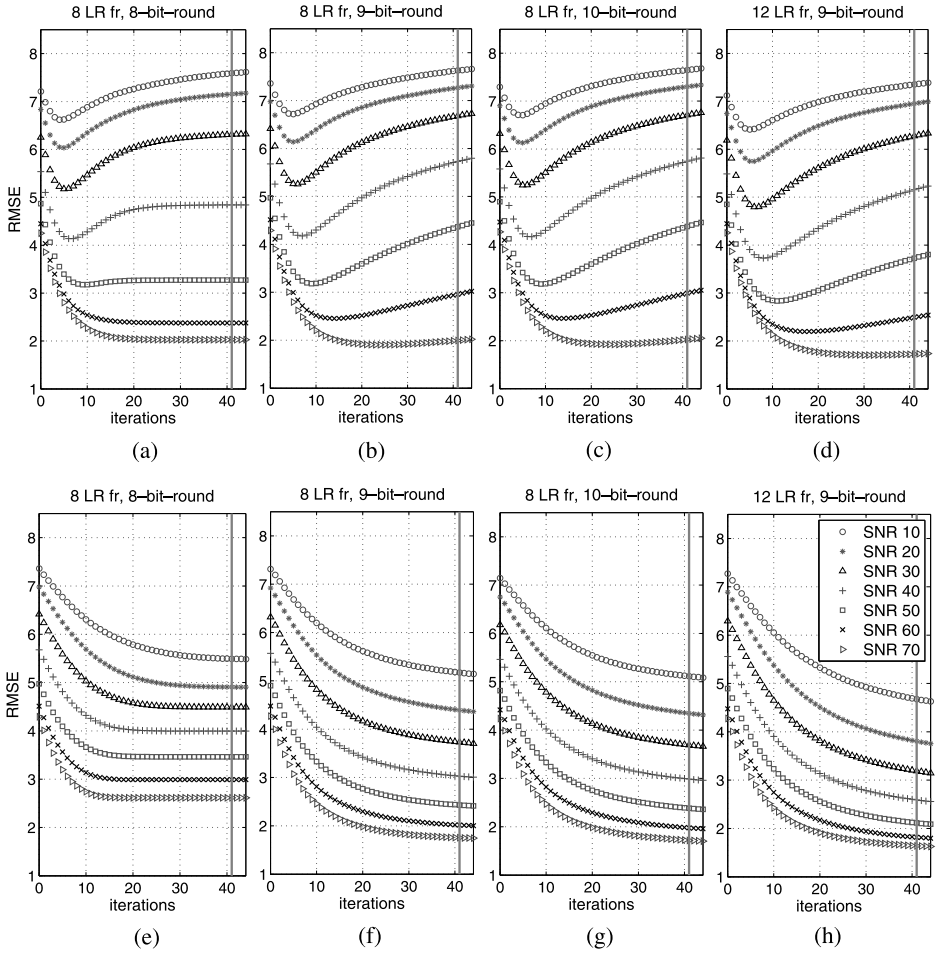

Fig. 21. Comparing the convergence properties of Equations 3 (a–d) and Equations 5 (e–h), for the *cars* sequence. By accounting for the noise statistics in the LR frames, Equation 5 gives robustness with respect to noise.

These problems are resolved by employing Equation 5 instead of Equation 3. Figures 21(e–h) demonstrate the good convergence properties of the algorithm of Equation 5. The reconstruction error is now always minimized, regardless of the given system parameters. Two main trends, affecting the rate of convergence, can be observed: (1) For a given word-length, a decrease in the error deriving from increasing the number of LR frames, becomes larger as the SNR level becomes lower. This is observed comparing the graph of Figure 21(f) with that of Figure 21(h). (2) For a given number of LR frames, the decrease of the error resulting from an increase in the word-length becomes larger as the SNR gets higher. This is obvious when comparing Figure 21(e) with Figure 21(f).

These observations can be interpreted as follows. For low SNRs, the reliability of the LR samples is low. Therefore, additional LR frames further

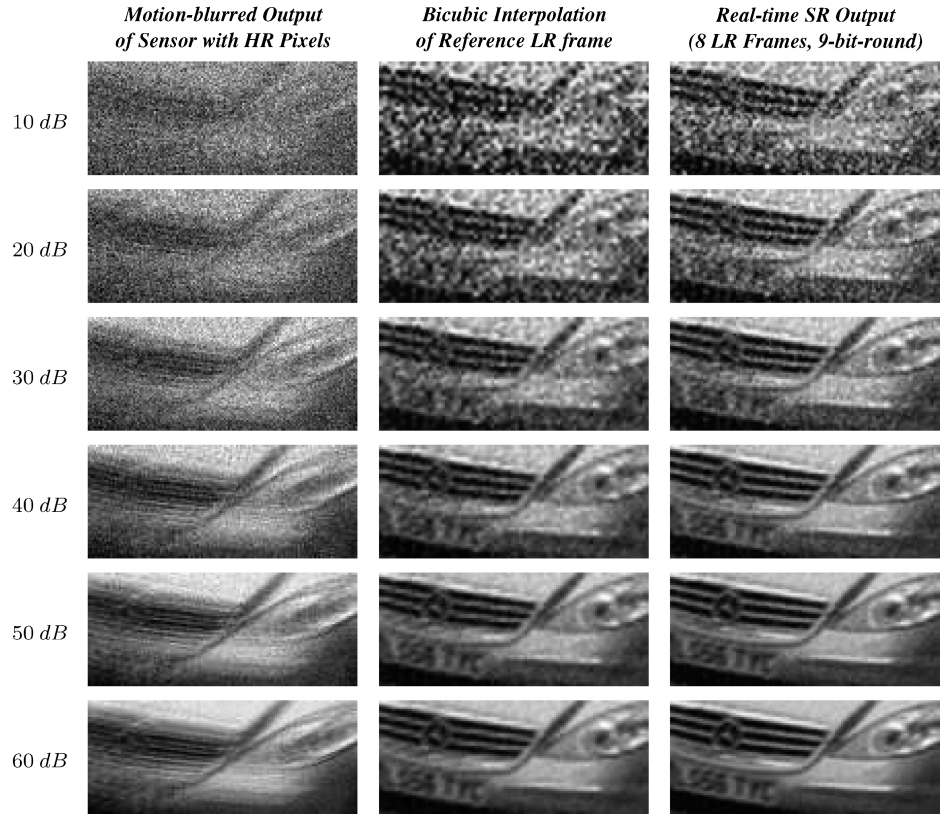

Fig. 22. Reconstruction of *car* for different SNRs. The SR outputs visually demonstrate the higher quality obtained by the implemented real-time algorithm, compared to the motion-blurred output of a traditional HR sensor and the bicubic interpolation of the output of an LR sensor.

contribute in the reconstruction on the HR grid, as long as the level of the related non-rigid deformations remains low (Section 3.5). On the other hand, for high SNRs, the LR frames are more reliable, and since they comprise samples of the same scene, they tend to overlap when their number increases. In that case, increasing the word-length affects the reconstruction quality more than considering more frames. In other words, for low SNRs more frames are generally more valuable than more bits, while for high SNRs it is the other way round.

Moving from the scenario of Figure 21(f) to that of Figure 21(g), the decrease of the error is trivial, indicating that a tenth bit is redundant. Thus, nine bits is the minimum word-length for maximal performance for the number of iterations corresponding to the given frame size.

The output obtained for the parameters of Figure 21(f), after the number of iterations giving real-time performance, is visualized in Figure 22. The part of the frame shown in Figure 22 corresponds to the ground-truth part

of Figure 19(b). Three types of results are illustrated for each noise level. The first column shows the motion-blurred output that would be produced by a traditional image sensor employing a uniform static grid of HR pixels. The second column contains the output of floating-point bicubic interpolation applied on a single LR frame. Finally, the third column illustrates the real-time SR output, where HR spatial resolution and LR temporal resolution are successfully combined.

5.2.4 *Conclusions of the Performance Evaluation.* In the previous paragraphs multiple design options and algorithmic parameters have been evaluated and discussed. The general conclusions deriving from this evaluation are the following.

(1) For low SNRs, considering more LR frames is generally more valuable than increasing the word-length of the data path, while for high SNRs it is the other way round.
(2) The minimum word-length for maximal performance increases as the number of iterations increases, thus as the frame size decreases (Table I).
(3) Bilinear interpolation is preferable for initialization purposes, as it leads to similar results to bicubic interpolation but with lower computational cost.

6. CONCLUSIONS AND FUTURE WORK

In this work, we propose an FPGA-based motion-deblurring system that uses an adaptive image sensor, which is configured with areas of large pixels on the motion regions. FPGA-based SR is used to compensate for the low spatial resolution of such areas. In particular, the IBP algorithm is implemented on FPGA, after being modified to account for the additive noise in the LR samples. The system is evaluated under various noise levels, considering different options and parameters, such as the type of initial approximation, the number of LR samples, and the word-length. Results demonstrate that including information about the noise statistics in the algorithm dramatically improves the convergence properties of the iterative process in the presence of noise, leading to a more robust reconstruction scheme (Figure 21).

Interesting observations derive from the evaluation process of the implemented algorithm of Equation 5. The low-computational-cost bilinear interpolation, when used for initialization purposes, renders similar results as the elaborate bicubic interpolation, due to the good convergence properties of the algorithm. Also, for a given word-length, when more LR frames are considered, the error is more noticeably decreased for low SNRs. On the other hand, for a given number of LR frames, increasing the word-length results in better enhancement for high SNRs. 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 following issues. (1) For large motion magnitudes, considerable extent of motion blur may also appear in the LR samples. Using SR techniques, the temporal resolution of the reconstructed output is limited by that of the LR samples. This limit could be surpassed by employing

multi-channel blind deconvolution [Sroubek et al. 2007] in the SR framework. (2) Since the system operates in real time, a crucial issue is the configuration of the adaptive image sensor for the next frame to keep the moving object within the LR window. This can be achieved by determining the location of the LR window in the next frame, after predicting the new position of the object. Candidate predictors include the popular Kalman filter [Welch and Bishop 2006; Liu et al. 2007], its extended [Thrun et al. 2005; Bonato et al. 2007], and unscented [Wan and Merwe 2000] forms; and particle filters [Arulampalam et al. 2002]. (3) Finally, future work includes the implementation of the motion estimation and blur detection blocks of Figure 5 on FPGA as well.

REFERENCES

ANGELOPOULOU, M. E., BOUGANIS, C.-S., AND CHEUNG, P. Y. K. 2008a. Video enhancement on an adaptive image sensor. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 681–684.

ANGELOPOULOU, M. E., BOUGANIS, C.-S., CHEUNG, P. Y. K., AND CONSTANTINIDES, G. A. 2008b. FPGA-based real-time super-resolution on an adaptive image sensor. In *Proceedings of the International Workshop on Applied Reconfigurable Computing (ARC)*. 125–136.

ARULAMPALAM, M. S., MASKELL, S., GORDON, N., AND CLAPP, T. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Sig. Proc. 50*, 2, 174–188.

BAKER, S. AND KANADE, T. 2002. Limits on super-resolution and how to break them. *IEEE Trans. Patt. Anal. Mach. Intell. 24*, 9, 1167–1183.

BEN-EZRA, M. AND NAYAR, S. K. 2004. Motion-based motion deblurring. *IEEE Trans. Patt. Anal. Mach. Intell. 26*, 6, 689–698.

BONATO, V., MARQUES, E., AND CONSTANTINIDES, G. A. 2007. A floating-point extended kalman filter implementation for autonomous mobile robots. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*. 576–579.

BOUGUET, J.-Y. 2002. *Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm*. Microprocessor Research Labs, Intel Corporation.

BROWN, L. G. 1992. A survey of image registration techniques. *ACM Comput. Surv. 24*, 4, 325–376.

CHEN, T., CATRYSSE, P., GAMAL, A. E., AND WANDELL, B. 2000. How small should pixel size be? In *Proceedings of the SPIE Sensors and Camera Systems for Scientific, Industrial and Digital Photography Applications*. Vol. 3965. 451–459.

CONSTANDINOU, T. G., DEGENAAR, P., AND TOUMAZOU, C. 2006. An adaptable foveating vision chip. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*. 3566–3569.

FARRELL, J., XIAO, F., AND KAVUSI, S. 2006. Resolution and light sensitivity tradeoff with pixel size. In *Proceedings of the SPIE Electronic Imaging '06 Conference*. Vol. 6069. 211–218.

FARSIU, S., ROBINSON, D., ELAD, M., AND MILANFAR, P. 2004. Advances and challenges in super-resolution. *Int. J. Imag. Syst. Tech. 14*, 2, 47–57.

FOVEON, INC. 2007. Foveon X3 Image Sensor. www.foveon.com.

GAMAL, A. E. AND ELTOUKHY, H. 2005. CMOS image sensors. *IEEE Circ. Devic. Mag. 21*, 3, 6–20.

IRANI, M. AND PELEG, S. 1991. Improving resolution by image registration. *Graph. Mod. Image Proc. 53*, 3, 231–239.

LIU, Y., BOUGANIS, C.-S., AND CHEUNG, P. Y. K. 2007. Efficient mapping of a Kalman filter into an FPGA using Taylor expansion. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*. 345–350.

MENDIS, S., KEMENY, S., GEE, R., PAIN, B., STALLER, C., KIM, Q., AND FOSSUM, E. 1997. CMOS active pixel image sensors for highly integrated imaging systems. *IEEE J. Solid-State Circ. 32*, 2, 187–197.

PARK, S. C., PARK, M. K., AND KANG, M. G. 2003. Super-resolution image reconstruction: a technical overview. *IEEE Sig. Proc. Mag. 20*, 3, 21–36.

SHECHTMAN, E., CASPI, Y., AND IRANI, M. 2005. Space-time super-resolution. *IEEE Trans. Patt. Anal. Mach. Intell. 27*, 4, 531–545.

SHI, J. AND TOMASI, C. 1994. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 593–600.

SROUBEK, F., CRISTOBAL, G., AND FLUSSER, J. 2007. A unified approach to super-resolution and multichannel blind deconvolution. *IEEE Trans. Image Proc. 16*, 9, 2322–2332.

STARK, H. AND OSKOUI, P. 1989. High-resolution image recovery from image-plane arrays, using convex projections. *J. Opt. Soc. Amer. A 6*, 11, 1715–1726.

STERN, A., PORAT, Y., BEN-DOR, A., AND KOPEIKA, N. S. 2001. Enhanced-resolution image restoration from a sequence of low-frequency vibrated images by use of convex projections. *Appl. Opt. 40*, 26, 4706–4715.

THRUN, S., BURGARD, W., AND FOX, D. 2005. *Probabilistic Robotics*. MIT Press, Cambridge, MA.

TIAN, H., FOWLER, B., AND GAMAL, A. E. 2001. Analysis of temporal noise in CMOS photodiode active pixel sensor. *IEEE J. Solid-State Circ. 36*, 1, 92–101.

WAN, E. A. AND MERWE, R. V. D. 2000. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*. 153–158.

WELCH, G. AND BISHOP, G. 2006. An introduction to the Kalman filter. http://www.cs.unc.edu/welch/media/pdf/kalman_intro.pdf.

YOULA, D. C. AND WEBB, H. 1982. Image restoration by the method of convex projections: part 1-theory. *IEEE Trans. Med. Imag. MI-1, 2*, 81–94.

ZITOVÁ, B. AND FLUSSER, J. 2003. Image registration methods: A survey. *Imag. Vis. Comput. 21*, 11, 977–1000.

Received May 2008; revised September 2008; accepted October 2008