Hardware Architectures for Real-Time Video Enhancement and their Application to an Adaptive Image Sensor

Maria E. Angelopoulou

A thesis submitted for the degree of

Doctor of Philosophy of the University of London

and for the Diploma of Membership of the

Imperial College

Department of Electrical and Electronic Engineering

Imperial College of Science, Technology and Medicine

University of London

May 2009

Abstract

The work presented in this thesis lies in the area of real-time video processing and focuses on the problem of enhancing in real-time the spatio-temporal resolution of the captured video sequence. To achieve the above, this thesis explores, proposes, and brings together, into a novel imaging system, appropriate imaging techniques related to different levels of processing: high-level video enhancement algorithms, low-level implementation on reconfigurable hardware, Field Programmable Gate Arrays (FPGA) in particular, and state of the art image sensor technology. Contrary to traditional cameras, which are passive, the proposed imaging system is dynamically configured in real-time according to the captured video data, based on the real-time interaction of an adaptive image sensor with an FPGA-based processing unit. The FPGA both configures the adaptive image sensor in a manner that maximizes the captured information, and further processes this raw data to render outputs of high resolution both in space and in time. Therefore, this work proposes possible sensor configurations and the appropriate processing methods for video enhancement under real-time constraints. These methods mainly include super-resolution and blur

identification techniques, and architectures of these methods are proposed and implemented on FPGA. The throughput that is achieved is significantly higher than the 25 fps real-time requirement, for frame sizes up to 1024×1024, and the system's performance is robust to noise for Signal to Noise Ratio (SNR) as low as 20 dB.

Contents

List of Figures

List of Tables

Acknowledgements

This PhD thesis was completed under the supervision of Professor Peter Y. K. Cheung and Dr. Christos-Savvas Bouganis. The work presented in Chapter 3 was carried out under the supervision of Dr. Konstantinos Masselos. I sincerely thank my supervisors for their insightful guidance and support. I have immensely enjoyed our cooperation during all these years, and I feel very privileged to have worked with them.

During my PhD, I had the opportunity to collaborate with researchers and academics who are experts in their fields, whom I want to thank. I thank Dr. George A. Constantinides for his help with Chapter 5, and Dr. Yiannis Andreopoulos for his help with Chapter 3. I also thank Dr. Anil A. Bharath and Dr. Thomas J. W. Clarke for providing feedback on several aspects of this work at my transfer examination. I want to thank Professor Maria Petrou for her helpful suggestions regarding image processing issues.

I also thank the examiners of this thesis, Professor Nishan Canagarajah and Dr. Oskar Mencer, for their useful comments that have contributed to the clarity of the text.

Finally, I would like to thank our group administrator, Mrs Wiesia Hsissen, for her help in dealing with numerous administrative issues during my PhD studies.

I dedicate this thesis to my mother Eva, my father Evangelos, my grandmother Marika, and my beloved fiancé Alexandros. Their love and support kept me going through the PhD years.

List of Notation

Notation	Meaning
S_h	The width of the elementary pixel of the sensor.
$\vec{d}$	Motion vector.
f_j	j^{th} video frame.
$\vec{h}$	Vector of unknown HR pixels (in the super-resolution problem).
$\vec{l}$	Vector of known LR pixels (in the super-resolution problem).
A	Matrix that contains the relative contribution of each unknown HR pixel to each known LR pixel (in the super-resolution problem).
$\mathbb{N}$	The set of all positive integers.
σ_f	The standard deviation of the noise-free image.
σ_n	The standard deviation of the noise.
Lo^k	The kth observed low resolution frame.
$\delta_0(k)$	The standard deviation of the noise of Lo^k.
Ls^k	The kth simulated low resolution frame in the Iterative Back Projection Super Resolution approach.
L_j	Low-frequency output of the 1-D DWT applied horizontally on the image rows at decomposition level j.
H_j	High-frequency output of the 1-D DWT applied horizontally on the image rows at decomposition level j.
LL_j	Low-frequency output of the 2-D DWT at decomposition level j.
LH_j, HL_j, HH_j	High-frequency outputs of the 2-D DWT at decomposition level j.

List of Terms and Acronyms

ACF Discrete Autocorrelation Function.

Adaptive Image Sensor A sensor that can be configured in real-time to locally form larger pixels.

BB Block-Based 2-D DWT computation schedule.

BIV Blur Identification and Validation scheme.

BRAM On-chip Block RAM memory.

Control path The path of the circuit through which control signals travel.

Critical path The longest path of the circuit between storage elements, *e.g.* flip-flops.

Data path The path of the circuit through which data travel.

DRAM Dynamic Random Access Memory.

DSP Digital Signal Processor/Processing (context dependent).

DWT Discrete Wavelet Transform.

FIFO First-In-First-Out buffer. A buffer from which values are read in the order in which they arrived.

FPGA Field Programmable Gate Array.

fps Frames per Second.

GPU Graphics Processing Unit.

Ground-truth Image The real-world image without any degradations. Used as a reference to determine the reconstruction quality.

Handel-C An hardware-oriented extension of ANSI C that includes structures that control hardware parallelism and instantiation.

HR High spatial and low temporal resolution, referring to pixels or frames (context-dependent).

Isotropic Invariant with respect to direction.

LB Line-Based 2-D DWT computation schedule.

LR Low spatial and high temporal resolution, referring to pixels or frames (context-dependent).

LUT Look-Up Table. An element of the circuit with multiple inputs and a single output whose value is definite for each set of inputs.

Non-Isotropic Direction-dependent.

Pipelining The process of inserting registers in the circuit to reduce the clock period.

PSF Point Spread Function.

RC Row-Column 2-D DWT computation schedule.

RMSE Root Mean Squared Error.

Slice A fine-grain component on an FPGA. The component contains two 4-LUTs with optional output registers, and additional logic for efficient carry operations. Used as a unit for area measurement.

SNR Signal to Noise Ratio.

SR Super-Resolution.

SRAM Static Random Access Memory.

Synthesis The automated process of mapping an algorithm to a set of primitive hardware blocks on the targeted device.

TI Total Intensity.

VHDL Very high speed integrated circuit Hardware Description Language.

Word-length The width of a stored value or a signal in bits.

ZBT Zero Bus Turnaround.

Chapter 1

Introduction

1.1 Motivations and Objectives

Enhancing in real-time the resolution of video data is a critical issue in a number of popular applications, such as biomedical applications and surveillance. For instance, when a camera is inserted into a patient's body, it experiences motion blur both due to its own motion and due to the movements of the human organs. In surveillance applications, high resolution is critical on local regions of interest, such as human faces or number plates.

Traditional cameras are passive, being based on a static image sensor. Instead this thesis proposes a dynamic imaging system that is based on the real-time interaction of an adaptive image sensor with a processing unit. In particular, this thesis explores and proposes state of the art techniques related to different levels of processing: video enhancement algorithms, implementation on reconfigurable hardware, and image sensor technology, and finally proposes

an imaging system that maximizes the spatio-temporal resolution of the output by bringing together the appropriate techniques from the above individual fields. The proposed system allows the real-time resolution enhancement of the captured video sequence both locally on selected regions and globally on the entire frame, based on the requirements of the given application.

The adaptive image sensor is a state of the art sensor, which is no longer subject to the constraint of time and space invariant pixel size [Fov08, CDT06]. Elementary pixels can be grouped together to form a larger pixel where and when necessary. Utilizing the above technology and taking advantage of the advances in the area of reconfigurable hardware, this work explores how a Field Programmable Gate Array (FPGA) can dynamically configure a time-variant image sensor, so as to maximize the raw information collected from the environment. Appropriate processing methods and their FPGA-based architectures are proposed to further process that information and reconstruct a final output of both high temporal and high spatial resolution. By exploiting the parallelism, pipelining and data reuse possibilities offered by reconfigurable hardware, real-time performance, *i.e.* at least 25 fps for resolutions up to 1024×1024, is achieved.

In the proposed video enhancement system, the role of the FPGA is twofold, as illustrated in Fig. 1.1. The FPGA processes the raw information in real-time and, also, decides how to configure the adaptive image sensor in a way that maximizes the raw information that is captured, according to the collected data.

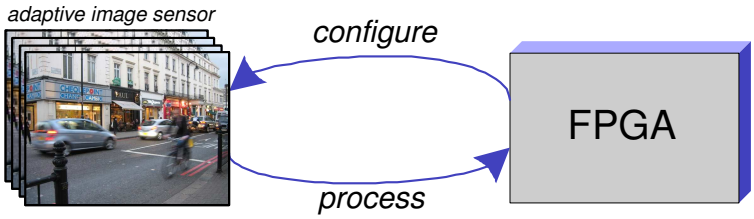


Figure 1.1: Bidirectional interaction between FPGA and adaptive image sensor.

1.2 Overview

As will be explained in the chapters that follow, the proposed video enhancement system relies on the reconfiguration property of an adaptive image sensor and comprises processing blocks that aim to reconstruct a final output of high spatio-temporal resolution. The processing mainly contains modules that execute blur identification, motion estimation, and reconstruction with Super-Resolution techniques.

The overview of the design flow is shown in Fig. 1.2. Each block of Fig. 1.2 is related to the chapters of the thesis as indicated. The proposed FPGA-based video enhancement system utilizes the reconfiguration property of an adaptive image sensor to maximize the raw information that is captured. The captured information is further processed by the blocks that are enclosed in the dashed rectangle (Fig. 1.2) for the reconstruction of the system output.

The thesis begins in Chapter 2 by examining some of the previous and concurrent work in computational photography and video, which is the broad area of interest of this thesis. Moreover, Chapter 2 presents some common background that is required throughout the thesis.

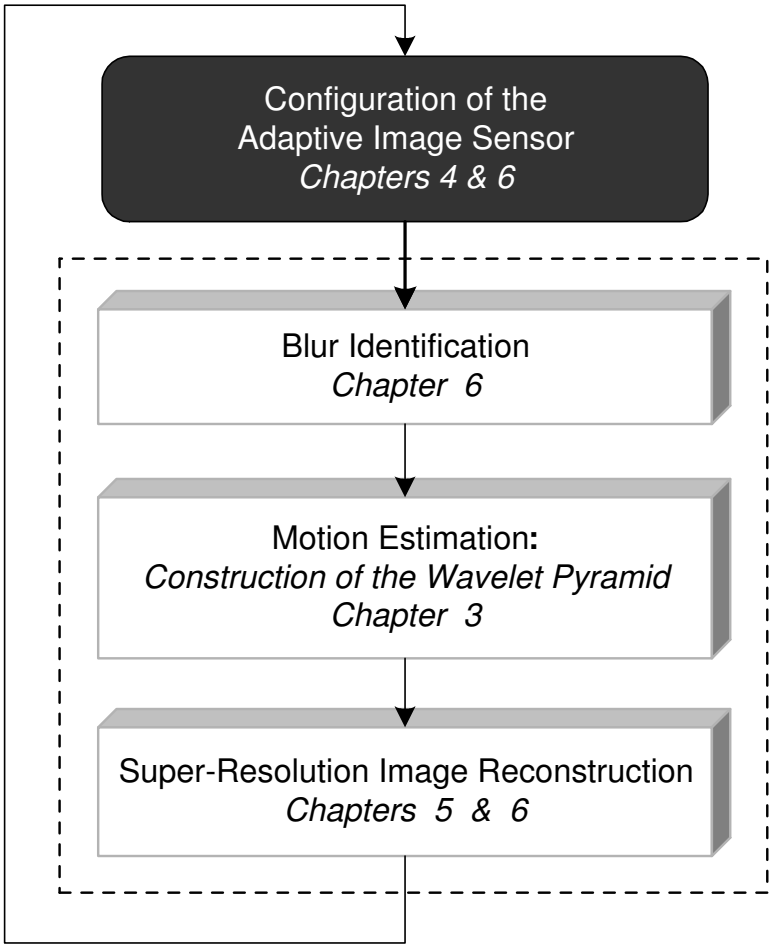


Figure 1.2: Overall design flow and relation with the chapters of the thesis.

Chapter 3 investigates the implementation of the main 2-D DWT computation schedules on FPGA. The construction of the wavelet pyramid is a key operation in a variety of applications that are based on multiresolution decomposition, such as image compression. Multiresolution decomposition also comprises a critical issue for robust motion estimation. The importance of motion estimation in the flow of Fig. 1.2 derives from the fact that the output frames should be registered before their pixels are fused with Super-Resolution techniques.

Chapter 4 explores the configuration space of an adaptive image sensor. The aim of the sensor configuration is to maximize the raw information collected from the environment. Two different types of configuration are presented and for each configuration, appropriate methods are proposed to further process that information and reconstruct a final output of both high temporal and high spatial resolution. Thus, two different approaches are proposed, namely a SR-based and a deconvolution-based approach.

Chapter 5 investigates the implementation of Super-Resolution (SR) on FPGA, which enables the real-time reconstruction of the system output with high resolution both in space and time. By including information for the noise statistics in the reconstruction process, the robustness of the traditional Iterative Back Projection SR algorithm is increased. In Chapter 5, the intra-frame motion of the SR inputs is considered to be negligible.

Contrary to Chapter 5, in Chapter 6, the motion blur in the frames at the input of the reconstruction block is considered to be non-negligible, as would

be required in the case of very fast motion. Thus, the intra-frame motion information of the SR inputs should be identified and incorporated in the SR reconstruction process. When the frames are degraded by heavy motion blur, the PSFs are highly non-isotropic, which further complicates their estimation. The ill-posed nature of blur identification is usually addressed using the assumption of linear and uniform motion. However, in real-life systems, this may deviate significantly from the actual motion blur. To resolve the above, this work proposes combining a scheme that validates the initial motion assumption with the real-time reconfiguration property of an adaptive image sensor. If the linearity and uniformity assumption is invalid for a given motion region, the sensor is locally reconfigured to larger pixels that produce higher frame-rate samples with reduced blur. Once the appropriate configuration that gives rise to a valid initial motion assumption is applied, highly accurate PSFs are estimated, resulting to an improved SR reconstruction quality.

Notation is introduced in each chapter as required. However, some basic notation, terms and acronyms that are used throughout the thesis are given in p. 24. Moreover, a list of terms and acronyms is given in p. 25.

1.3 Statement of Originality

The main contribution of the work presented in this thesis can be summarized as follows. This thesis explores and proposes state of the art techniques related to different levels of processing: video enhancement algorithms, im-

plementation on reconfigurable hardware, and image sensor technology, and finally proposes an imaging system that maximizes the spatio-temporal resolution of the output by bringing together the appropriate techniques from the above individual fields.

More specific contributions of this thesis lie in four main areas, which are the following:

1. The configuration space of an adaptive image sensor is investigated with respect to the quality of the final output. Two configuration schemes are proposed, which both provide rich spatio-temporal sampling for the real-world scene. Each configuration scheme is then combined with the appropriate processing block that further enhances the raw captured data, in order to provide a system of high reconstruction quality. Thus an SR-based and a deconvolution-based system are proposed, each associated with a particular type of sensor configuration and data processing. The performance evaluation of the two systems demonstrates that the SR-based outperforms the deconvolution-based system.

2. A robust to noise hardware-based Super-Resolution (SR) reconstruction scheme is proposed that is based on the Iterative Back Projection (IBP) SR approach. The IBP approach is shown to be very appropriate for hardware implementation, inherently offering the possibility to maximize the data reuse and parallelism in the data processing operations. The original IBP scheme is modified in order to account for the presence

of noise, by incorporating the noise statistics in the reconstruction process. This results in a robust to noise system with high reconstruction quality under noise levels in the range of 10 to 70 dB, as the system evaluation demonstrates. To meet the strict timing constraints required for real-time video capturing, a high-throughput architecture of the above scheme is proposed and implemented on an FPGA. The proposed architecture achieves in real-time reconstruction of high quality, which has been quantitatively evaluated, under noise levels within the range of 10 and 70 dB.

3. A joint linear blur identification and assumption validation (BIV) scheme is proposed that is combined with the reconfiguration property of an adaptive image sensor to enable accurate reconstruction. Since blur identification is an ill-posed problem, traditional blur identification methods utilize the linear and uniform initial motion assumption. In real-life systems, this assumption may deviate significantly from the actual motion, impairing subsequent restoration. To address this problem, while traditional blur identification is limited to the estimation of the blur parameters, the proposed scheme goes further and also evaluates the validity of the initial assumption, while calculating the blur parameters. If the linearity and uniformity assumption is invalid for a given motion region, the adaptive sensor is locally reconfigured to larger pixels that produce higher frame-rate samples with reduced blur. Results demonstrate that

once the appropriate sensor configuration, which gives rise to a valid motion assumption, is applied, highly accurate PSFs are estimated, resulting to an improved SR reconstruction quality. To target real-time restoration, a high-throughput hardware architecture of the proposed BIV scheme is presented and is implemented on a Field Programmable Gate Array (FPGA). The implementation requirements are given for different sets of parameters, and the system performance is evaluated under noise levels within the range of 10 and 70 dB. The blur parameters that are associated with each SR input are incorporated in the SR reconstruction process, and the reconstruction quality is evaluated under the above noise levels.

4. The major 2-D DWT computation schedules are compared and evaluated on FPGA. The evaluation demonstrates that the choice of a particular architecture for the execution of the 2-D DWT depends on the given specifications. These are related to the targeted throughput, area, and power consumption. The comparative analysis that is carried out indicates that the Line-Based (LB) approach achieves the highest throughput and the lowest energy consumption, while the Row-Column (RC) approach is related to the lowest area. The importance of this work derives from the wide range of image and video processing applications that are based on the 2-D DWT, such as image registration and compression. The investigation of the 2-D DWT computation schedules can act as an insight

on which schedule is most suitable for the specifications of the given application.

1.4 Publications

Parts of the work that is presented in detail in this thesis have been published in the following literature. In the list of publications, every reference is followed by the number of the chapter of the thesis that covers the corresponding material.

Journal publications:

1. Maria E. Angelopoulou, Konstantinos Masselos, Peter Y.K. Cheung and Yiannis Andreopoulos, "Implementation and Comparison of the 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs", Journal of Signal Processing Systems for Signal, Image, and Video Technology *(formerly the Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology)*, vol. 51, no. 1, pp. 3-21, April 2008. ***(Chapter 3)***

2. Maria E. Angelopoulou, Christos-Savvas Bouganis, Peter Y.K. Cheung and George A. Constantinides, "Robust Real-Time Super-Resolution on FPGA and an Application to Video Enhancement", to appear in ACM Transactions on Reconfigurable Technology and Systems. ***(Chapter 5)***

Conference publications:

1. Maria E. Angelopoulou, Christos-Savvas Bouganis and Peter Y.K. Cheung, "Video Enhancement on an Adaptive Image Sensor", in Proc. IEEE International Conference on Image Processing (ICIP), pp. 681-684, October 2008. ***(Chapter 4)***

2. Maria E. Angelopoulou, Christos-Savvas Bouganis, Peter Y.K. Cheung and George A. Constantinides, "FPGA-based Real-time Super-Resolution on an Adaptive Image Sensor", in Proc. Applied Reconfigurable Computing (ARC), pp. 125-136, March 2008. ***(Chapter 5)***

3. Maria Angelopoulou, Konstantinos Masselos, Peter Cheung and Yiannis Andreopoulos, "A Comparison Of 2-D Discrete Wavelet Transform Computation Schedules on FPGAs", in Proc. IEEE International Conference on Field Programmable Technology (FPT), pp. 181-188, December 2006. ***(Chapter 3)***

4. Maria E. Angelopoulou, Christos-Savvas Bouganis and Peter Y.K. Cheung, "A Sensor-Based Approach to Blur Identification and Super-Resolution for Real-Time Video Restoration", IEEE International Conference on Image Processing (ICIP), November 2009, to appear. ***(Chapter 6)***

Journal paper pending review:

1. Maria E. Angelopoulou, Christos-Savvas Bouganis and Peter Y.K. Cheung, "Blur Identification with Assumption Validation for Sensor-Based

Video Reconstruction and its Implementation on FPGA", IET Computers & Digital Techniques, submitted June 2009. ***(Chapter 6)***

Chapter 2

Background

2.1 Introduction

This chapter examines some of the previous and concurrent work in computational photography and video, which is the broad area of interest of this thesis. Moreover, this chapter presents some common background that is required throughout the thesis. Background theory associated with individual chapters will be given at the beginning of each chapter.

The chapter begins in Section 2.2 by presenting the topology of a traditional digital imaging system and the forward model that describes the formation of the output frames. In computational photography and video, the traditional topology is extended, as described in Section 2.3.1, to accommodate computational modules that produce new forms of visual information. In the remainder of Section 2.3, the main fields of computational photography and video are introduced, along with some representative techniques that are used to improve

the quality of the final output. The diversity of these techniques does not only concern the method of implementation, but also the level of application. Thus, apart from the computational level, some of the approaches that are described concern the sensor and optics level, and also the incident illumination that is employed.

The work of this thesis belongs to the field of computational video that aims at increasing the spatio-temporal resolution of the final output. Section 2.3.8 provides the background for this field, discussing the fundamental space-time trade-off and presenting the related literature of systems that aim at surpassing this trade-off. Section 2.3.8 finally introduces the system that is proposed in this thesis.

The proposed system is based on an adaptive image sensor. The unconventional properties of the adaptive image sensor are explained in Section 2.4.

Apart from the adaptive sensor, the proposed system also includes a computational processing unit that further processes the raw data to produce a final output of high spatio-temporal resolution. As mentioned in Chapter 1, this work targets real-time video capturing. Real-time video processing applications impose strict timing constraints being at the same time significantly computationally expensive. Therefore, for this type of applications, software solutions are often inadequate, as has been demonstrated in a number of recent publications. In [GNVV04], Guo et al. present a detailed quantitative analysis, which demonstrates the low throughput that general-purpose processing platforms, such as Pentium 3, VLIW, and MIPS, render when executing image

and video processing applications, and the speedup that can be achieved by employing reconfigurable hardware instead. For the 2-D convolution operation, which constitutes a very common block in video processing applications, Cope et al. demonstrate that both FPGAs and GPUs give a speedup which is at least an order of magnitude higher over a Pentium 4 3.0 GHz [CCLW05]. In [BMC09], it is shown that when implementing the Extended Kalman Filter algorithm to process 2-D maps containing up to 1.8 k features on FPGA, a 3-fold improvement is achieved over a Pentium M 1.6 GHz, while a 13-fold improvement is achieved over an ARM920T 200 MHz.

Considering the strict performance constraints of real-time video processing applications and the low throughput that general-purpose processing platforms give when executing such data intensive algorithms, it is highly important to choose an appropriate accelerator for the computationally intensive video processing tasks. The remainder of the chapter introduces the hardware technology that is available for this kind of acceleration. Specifically, Section 2.5 discusses the variety of hardware solutions that exist, and explains the choice of utilizing reconfigurable hardware. Section 2.6 focuses in Field Programmable Gate Arrays (FPGA) and briefly discusses FPGA technology and programming issues. Incorporating an FPGA-based processing unit into the computational imaging system, offers a flexible platform that can be customized depending on the user's needs, with relatively low power consumption, area costs, and high speed.

Finally, an FPGA-based system typically includes an external memory

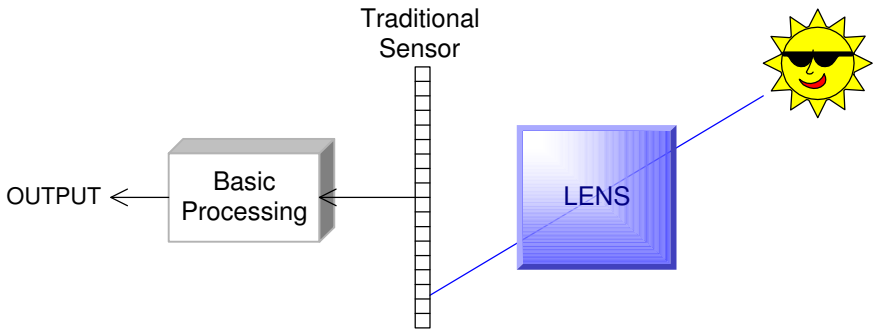


Figure 2.1: Traditional imaging system.

block for the storage of the input and the output data. Section 2.7 thus discusses the available options regarding the technology of the off-chip memory and the type of memory that is employed in this thesis.

2.2 Traditional Imaging Systems

The topology of a traditional digital imaging system is presented in Fig. 2.1. The main components of a traditional digital camera are: a lens, which captures the rays of light passing through the aperture, and an image sensor. The image sensor is a 2-D array of pixels that convert incident light to an array of electrical signals. Today, the most common type of sensor is the CMOS image sensor [GE05].

The topology of a CMOS image sensor is illustrated in Fig. 2.2. The CMOS image sensor is a 2-D array of pixels that converts the incident light to an array of electrical signals. Each one of these pixels contains a photodiode, which converts light into photocurrent. The percentage of the area of the

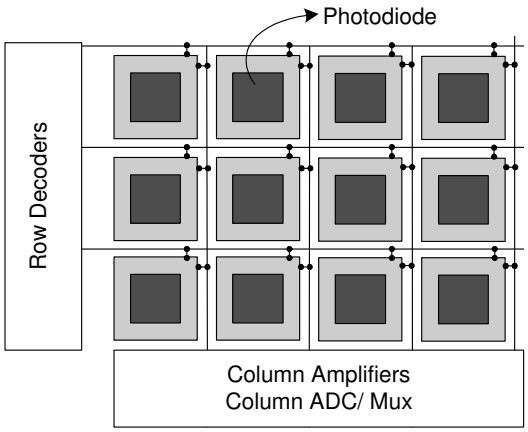


Figure 2.2: A hypothetical 3×4 CMOS image sensor.

sensor's pixel that is occupied by the photodiode is known as the fill factor. The higher the fill factor, the more sensitive is the sensor. The rest of the sensor's pixel contains transistors. These transistors need to be as small as possible, in order to increase the fill factor, so that more light will be absorbed by the photodetector, and as a result the integration of the light will give a valid value for that pixel faster. With the scaling of the CMOS technology, the fill factor of CMOS sensors increases. As it can be observed in Fig. 2.2, the architecture of the sensor is close to that of a RAM: row and column select circuits are used, and the charge voltage signals are read out one row at a time. The random access readout of a CMOS image sensor, contrary to the serial readout of CCD sensors [GE05], allows the sensor to operate at very high frame rates.

It should be noted that the basic processing unit illustrated in Fig. 2.1 is solely responsible for converting the sensed pixel values into an image.

2.2.1 The Observation Model

The output of the sensor is formed after a series of degradations are applied on the real-world scene. The real-world scene comprises the ground-truth frame whose resolution is high both in space and time. The forward model that describes the formation of the output of the imaging system is known by the term *observation model* [FREM04, PPK03]. Thus, the high-resolution pixels of the ground-truth 'input' undergo a series of degradations. The degradation stages that comprise the observation model are briefly described below.

The low spatial resolution (LR) degradation channel associated with each LR frame comprises a series of degradations, as shown in Fig. 2.3. 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 succeed the atmospheric blur and that are directly related to the imaging system. 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). If the cause of motion is due to camera shake, as in Fig. 2.3 (c), this convolution spans the entire frame. On the other hand, if the cause is due to a moving object, as in the images of Fig. 2.4, then 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,

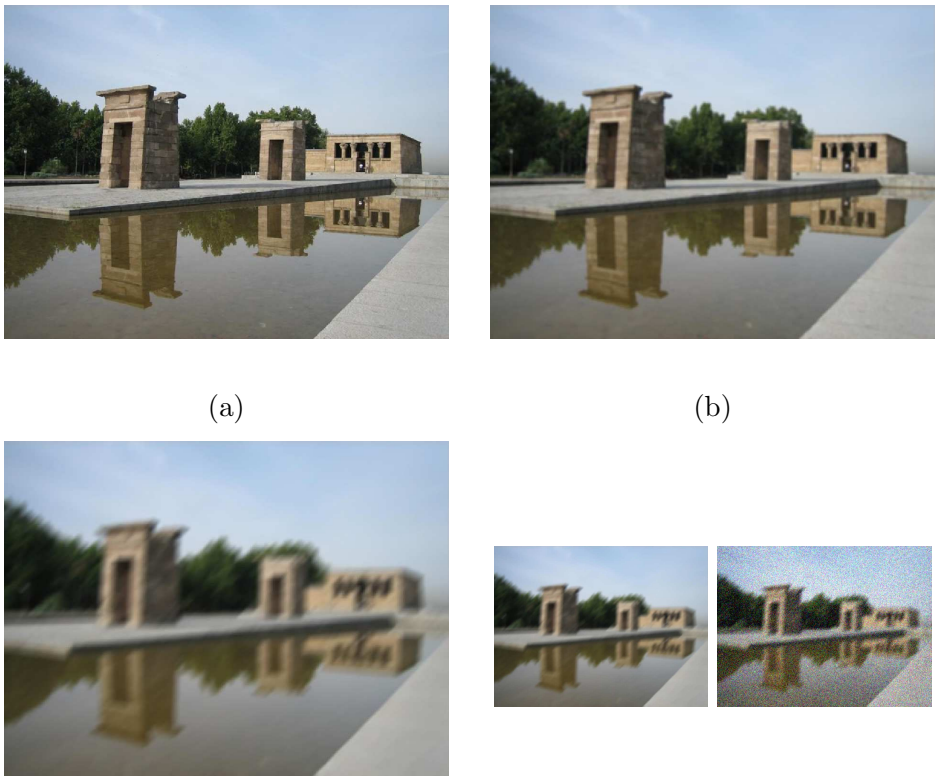


Figure 2.3: The degradation stages of the observation model that describes the generation of the LR output. (a) High resolution input (b) Atmospheric blur (c) Motion blur (d) Camera pixel blur (*i.e.* subsampling of the original pixel grid) (e) Additive system noise

(a) (b)

(c) (d)

Figure 2.4: The effect of motion blur on real-life images.

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 which 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, 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 compose an additive noise factor that reduces the pixel signal fidelity. Sources of such noise include shot noise, quantization noise, readout noise, fixed pattern noise, and reset noise [TFG01, MKG+97]. 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), *i.e.* the ratio of the signal power to the noise power.

2.3 Computational Photography and Video

Computational photography and video comprise an emerging new field that combines innovative image sensors and computational methods. Its objective is to surpass the limitations of traditional digital photography and video. The role of imaging systems has moved from merely capturing raw pixels, to generating richer forms of visual information that comprise a more meaningful

representation of the real-world scene [Nay06, RTM+06].

Due to the novelty of the field, there is not yet a clear definition for computational photography, as its limits are still unknown. The essence of computational photography is very elegantly captured in Brian Hayes' apposite remark [Hay08]: "Neuroscientists have recognized that the faculty of vision resides more in the brain than in the eye; what we "see" is not a pattern on the retina but a world constructed through elaborate neural processing of such patterns. It seems the camera is evolving in the same direction, that the key elements are not photons and electrons, or even pixels, but higher-level structures that convey the meaning of an image."

2.3.1 Topology of a Computational Imaging System

The difference between the digital system of Fig. 2.1 and an old film-based camera system mainly lies in the substitution of the film with a silicon image sensor. The remainder of the optical system is still almost the same, based on the traditional principle of *camera obscura* [New82].

Contrary to the simple digital camera, a computational imaging system extends the functionality of the modules of Fig. 2.1 in order to produce new, richer types of information [Nay06, RTM+06]. Fig. 2.5 shows the topology of a computational imaging system. As stated in [RTM+06], computational photography techniques generalize the basic elements of digital photography: namely, the optics, the image sensor, the processing that converts the sensed values into an image, and the illumination. The sections that follow examine

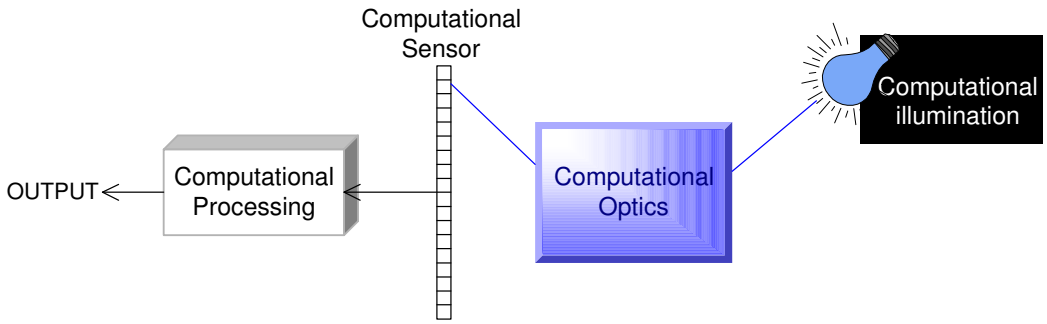


Figure 2.5: Computational imaging system.

some of the previous and concurrent work in extending the above elements beyond the capabilities of traditional digital photography.

2.3.2 Dynamic Range

Digital cameras typically employ 256 levels of brightness in each color channel. These are often not enough to capture the large variations of brightness in a real-world scene. The effect of limited dynamic range is obvious if a person sits in a dark room in front of an open window. In that case, if the exposure time is long, the person is well imaged but the window is saturated. On the other hand, if the exposure time is short, the outdoor scene appears well lit, but the person is too dark.

In the literature, various methods have been proposed for high dynamic range imaging, either employing computational processing or employing a specific sensor technology. The main idea of high dynamic range computational processing methods is to capture multiple images with different expo-

sure times and use further processing to combine the best parts of the captured images [RBS99, Gos05].

At the level of the sensor, several techniques are proposed. One approach employs logarithmic sensors [KDS+00]. Another technique proposes applying a checkerboard pattern with varying sensitivities on the sensor array [NM00]. Other approaches include well-capacity adjusting [DMBS98] and self-reset [McI01].

2.3.3 Field of View

The majority of imaging systems, either artificial or biological, capture only a part of the 360^o field that surrounds them. The vision capability of a given imaging system can thus be significantly improved if the captured field of view is increased.

At a computational processing level, the field of view is increased by employing panorama stitching, also known as image mosaicking. The first significant work on panorama stitching was done by Mann and Picard in 1994 [MP94]. Recently, Brown and Lowe proposed in [BL03] a fully automatic technique for the construction of panoramas.

Alternative solutions for increasing the field of view utilize unconventional optics, such as the catadioptric camera of [NP99]. The catadioptric camera is formulated by applying in front of a conventional camera a particular type of lens, which includes a relay lens and a paraboidal mirror.

2.3.4 3-D Imaging

The process of extracting the scene's 3-D structure is referred to as 3-D imaging or depth sensing. This is very useful for applications such as object and face recognition, tracking, and 3-D computer games.

Various computational processing 3-D imaging methods can be found in literature, such as those that are based on light detection [ZTCS99, BB04]. Recently, depth sensors have been proposed, such as the time-of-flight sensor of [GYB04], and the array of avalanche diodes presented in [NRBC04]. Also, particular topologies for computational optics have been proposed for the extraction of depth information, such as the multiview radial imaging system of [KN06]. The latter employs a hollow cone, which is mirrored from the inside and is applied in front of a conventional camera.

2.3.5 Image Matting

Another important method of computational photography is digital image matting. Image matting is the process of extracting a foreground object from the background along with the foreground opacity ("alpha matte") for each pixel of the object. Matting is a highly ill-posed problem, which is normally simplified by employing a certain degree of user interaction. Thus matting methods normally require the image to be accompanied by a a rough segmentation of the image into three regions: foreground, background and unknown [RT00, CCSS01, LLW08]. Recently, Levin and colleagues proposed

spectral matting, which computes a set of fuzzy matting components in an unsupervised manner [LRAL08].

2.3.6 Computational Illumination

Incident illumination can be controlled in a structured manner so as to enable post-capturing processing, such as virtual relighting or the fusion of multiple frames into a single output of higher quality.

Debevec, Wenger, and colleagues have done extensive work in this field, with the main objective of synthetically relighting a human performance with arbitrary lighting environments after the capturing process [WGT+05, DHT+00]. Their work is based on the linearity of light: any possible lighting environment can be expressed as a linear combination of single-light images. A basis of single-light images is formed by capturing an actor's appearance under single rays of light originating at (x, y, z) and emitting in direction (θ, ϕ), and relighting is achieved by linear combinations of the basis images.

A simpler version of the multiple illuminations scheme is to combine a pair of images captured with and without flash. The technique is known as flash-no flash and is presented in [ED04, PSA+04]. The no-flash image includes the large-scale illumination effects, but contains high levels of noise. On the other hand, the flash image has lower noise, contains high frequency components, but fails to capture the large-scale illumination characteristics. The good properties of the two images are then blended together to produce an output of low noise and more details.

2.3.7 Computational Sensors

A very interesting type of unconventional sensor that was recently developed, is the Foveon sensor [Fov08]. Specifically, the Foveon sensor has the following characteristics.

1. The sensor avoids the Bayer filter by utilizing the property of silicon to absorb different wavelengths of light at different depths. Therefore, on each pixel three layers are formed, each corresponding to a different spectral band, R, G, or B. As a result the sensor captures all spectral bands at every pixel location, thus reducing post-processing and avoiding demosaicking-related artifacts.

2. The full-color property of the Foveon pixels makes feasible the formation of full-color super-pixels, by grouping together multiple elementary pixels. Larger pixels have higher SNR. Therefore, under low illumination the sensor grid is globally configured to larger pixel sizes. This significantly reduces the image noise in low-light conditions.

While in [Fov08] the sensor is configured *globally* based on the global illumination conditions, the reconfiguration property is *locally* applied by the adaptable foveating vision chip of [CDT06]. The bio-inspired chip of [CDT06] imitates the functioning of the human eye by employing high spatial resolution on the foveal regions. In the surrounding regions, elementary pixels are grouped together in a similar manner as in [Fov08].

In the video enhancement system that is proposed in this thesis, the local reconfiguration capability of an adaptive sensor is utilized for a different purpose, *i.e.* for the local motion deblurring of motion regions. Thus, more discussion on the sensor's reconfiguration property and the space-time trade-off, which is related to the pixels' resolution, follows in Section 2.3.8.

Another type of computational sensor is the 'gradient camera' of [TAR05]. Instead of measuring static intensities, the gradient camera, measures static gradients. This increases the dynamic range, as image gradients are not saturated as much as image intensities.

As has been mentioned in Section 2.3.4, computational sensors have been also proposed for depth sensing [GYB04, NRBC04].

2.3.8 Space-Time Resolution

This section first discusses the trade-off between the spatial and temporal resolution of imaging systems. It then presents methods that have been proposed to address the above trade-off. Finally, the section introduces the main principles of the video enhancement system that is proposed in this thesis.

The Space-Time Trade-off

Two types of resolution determine the quality of information collected by an image sensor: the spatial and the temporal resolution. The spatial resolution depends on the spatial density of the photodiodes and their induced blur. The spatial density of the photodiodes of the sensor not only determines the

spatial, but also the temporal resolution. This will be explained in detail in the paragraphs that follow.

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 integration time is required for each photodiode to achieve an adequate signal to noise ratio [GE05, FXK06, CCGW00].

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. Fig. 2.7 demonstrates a scenario with zero relative motion between the scene and the imaging system. Figures 2.7(a, b, c) illustrate the LR sequences that consist of the samples of a still scene that are produced during the HR integration, for different pixel sizes. Pixel sizes 1×1 S_h, 2×2 S_h, and 4×4 S_h are employed, respectively. Since no relative motion exists, all LR frames within the same group contain exactly the same information. In Fig. 2.7(d) and Fig. 2.7(e), the output is reconstructed on the HR grid after applying bicubic interpolation on a single LR frame of the LR sequences of Fig. 2.7(b) and Fig. 2.7(c), respectively. Clearly, when no motion is involved 1×1 S_h gives a better result.

In real-life systems, relative motion between the imaging system and the scene normally exists: either the camera is shaking or/and objects are moving

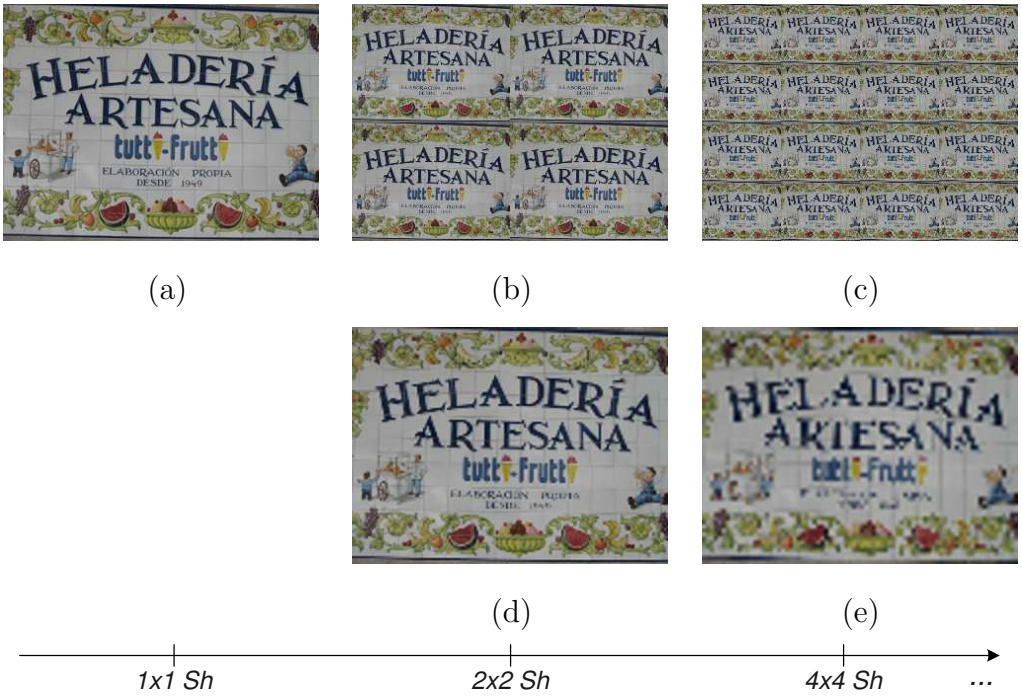


Figure 2.6: (a, b, c) Samples of a still scene produced during the HR integration for pixel sizes 1×1 S_h, 2×2 S_h, and 2×2 S_h. (d, e) Bicubic interpolation of a single frame of the LR sequences of (b) and (c). Clearly, when no motion is involved 1×1 S_h gives the best result.

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. 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.*

The fundamental trade-off between space and time is visualized in Fig. 2.7. This demonstrates how a scene affected by global motion, deriving from a shaking camera, would be captured by image sensors of different pixel sizes. Clearly, as the pixel size increases in the direction of the arrow, the motion blur is reduced, thus the temporal resolution is increased, but also the spatial resolution is reduced as well. Figures 2.7(b) and 2.7(c) illustrate the captured LR sequences for pixel sizes 2×2 S_h and 4×4 S_h. To reconstruct the LR output on the HR grid of Fig. 2.7(a), bicubic interpolation is applied on a single frame of the corresponding LR sequence. The spatial resolution of the reconstructed frames decreases in the direction of the arrow, as demonstrated in Figures 2.7(d) and 2.7(e).

In the real-life images of Fig. 2.4, the exposure time was too long for the fast moving objects to be captured. These images were captured with a commercial hand-held camera, a Canon IXUS 50. The effect of motion blur is clearly visible. Fig. 2.4(a), for instance, presents a moving bus as opposed to a still bike. The first object creates motion blur, whereas the second is well captured.

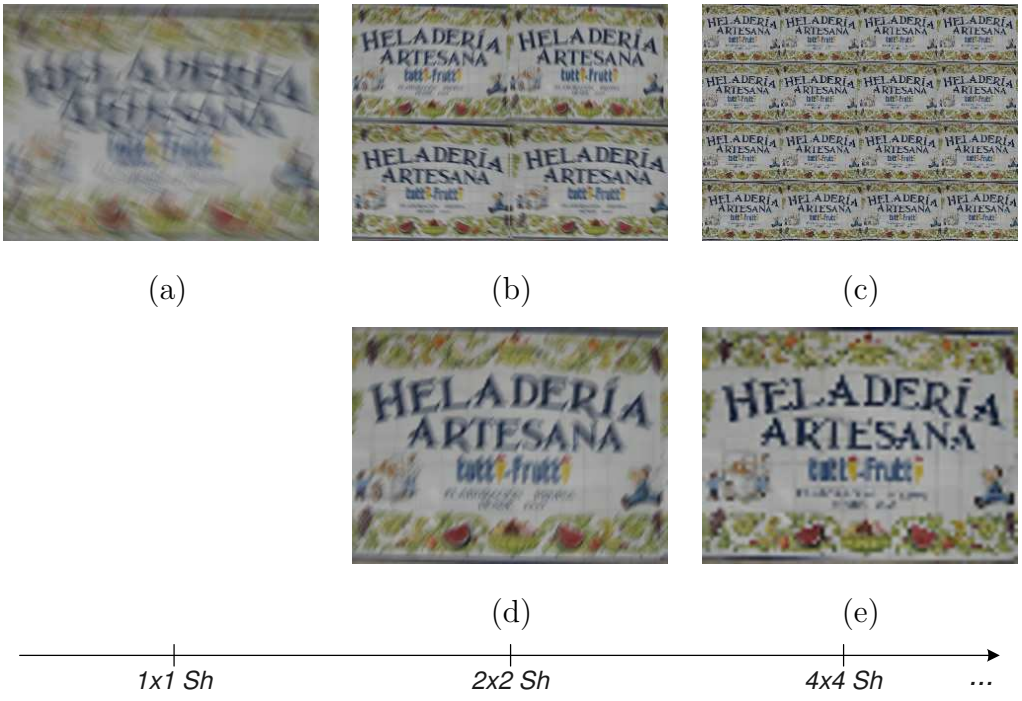


Figure 2.7: (a, b, c) Time samples produced during the HR integration for pixel sizes 1×1 S_h, 2×2 S_h, and 4×4 S_h, where S_h denotes the size of the elementary pixel of the sensor (d, e) Bicubic interpolation of a single frame of the LR sequences of (b) and (c) gives insufficient spatial resolution. SR methods can be used instead for the reconstruction on the HR grid.

Surpassing the Space-Time Trade-off: the Literature

A large number of methods that aim at increasing either the spatial or the temporal aspect of resolution exist in the literature: blur identification methods estimate the blurring function so as to reconstruct an image of high temporal resolution, while traditional super-resolution (SR) techniques create an output of high spatial resolution. Various imaging system topologies have been proposed to acquire the set of LR samples that are needed for SR fusion. Mainly, two types of topologies exist that aim at increasing the spatial resolution of the output: those that utilize multiple cameras [WJV+05, SCI05] and those where a single vibrating camera is employed [BEZN05, BEZN04]. The work presented in this thesis focuses on the problem of enhancing *both* the spatial and the temporal resolution of the output of an imaging system. The following paragraphs discuss systems that have been proposed to increase spatio-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 [SCI05]. The main strength of the algorithm in [SCI05] 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 [ZF03, Bro92]. 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 [BEN04] the proposed system consists of a high-resolution (HR) and a low-resolution (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 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 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 temporal resolution 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 [BEN04].

Very recently, sensor-based solutions have been proposed that locally deal with object motion. In particular, the sensor of [CHK+07] employs on-chip motion detection circuits that indicates regions of interest that are then configured to larger pixels, thus rendering high-frame-rate samples with reduced spatial resolution. The work presented in this thesis moves a step further. Thus, after the high-frame-rate samples of the motion regions are produced, they are fused together in order to enhance the spatial resolution of the output. This results in an output of high resolution both in space and time. Such a scheme imposes strict real-time constraints for the real-time computation of the enhanced frames, as will be discussed in the sections that follow.

Surpassing the Space-Time Trade-off: the Proposed System

The work presented in this thesis aims at resolving the space time trade-off by developing an imaging system that employs reconfigurable modules interacting in an autonomous fashion. The system addresses the problem of both local and global motions. The work presented in this thesis is concerned with the parts of Fig. 2.1 that are presented in Fig. 2.8, namely the sensor and the processing unit. The aim is to develop an unconventional video enhancement system where the sensor and the processing unit interact and modify each other according to information that they exchange. In this manner, the entire system can adapt to the motion properties of the actual scene and thus produce the best spatio-temporal representation of the real-world data. To achieve the above objective, a reconfigurable sensor is employed along with

an appropriate computational block that processes the captured raw data and aims to maximize the captured raw information.

The reconfigurable image sensor, which will be referred to as 'adaptive image sensor', is a novel type of sensor, where elementary pixels can be grouped together to form a larger pixel where and when necessary [Fov08, CDT06]. The local sensor reconfiguration property, which can be realized as explained in [CDT06], is utilized. The local sensor reconfiguration enables to treat independently local motion regions, according to the corresponding local motions. The reconfiguration property of the sensor is illustrated in Section 2.4, while possible configuration schemes are investigated in Chapter 4.

The raw information that is collected by the sensor will be further enhanced by the processing unit that reconstructs a final output of both high temporal and high spatial resolution. The system targets real-time video capturing and, thus, the necessity for real-time computation imposes strict performance constraints. Specifically, to achieve real-time performance, the system throughput should be at least 25 fps. Such requirements render software processing inadequate, due to the high computational complexity associated with the required pixel-level processing. As will be discussed in Sections 2.5 and 2.6, employing reconfigurable hardware, and Field Programmable Gate Arrays (FPGAs) in particular, can give maximum flexibility and performance under strict real-time constraints. By exploiting the parallelism, pipelining and data reuse possibilities offered by FPGAs, I will demonstrate that the above objectives are feasible, while the required area and power consumption of the circuits are

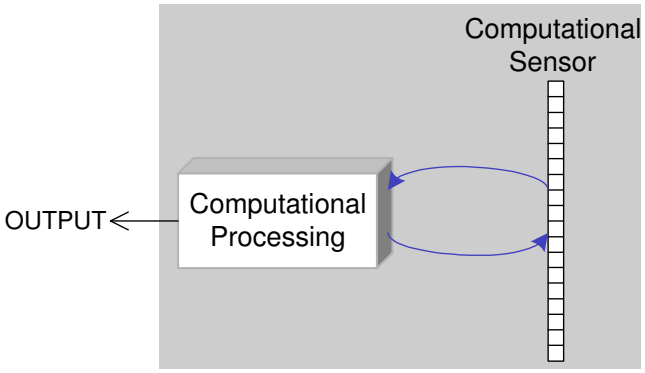


Figure 2.8: The part of the computational imaging system with which this thesis is concerned, and the proposed automated bidirectional interaction between the modules.

low. These issues will be made clear in the chapters that follow.

2.4 Dynamic Configuration of an Adaptive Image Sensor

The work presented in this thesis aims to surpass the limits imposed by the space-time trade-off and enhance in real-time the captured video sequence. To achieve this, this thesis looks beyond the conventional topology of uniform, time-invariant pixel sensors. Specifically, a real-time video enhancement system is proposed, which is based on two cornerstones: the real-time processing ability of reconfigurable hardware, FPGA in particular, and the reconfiguration property of a state-of-the-art adaptive image sensor. The current section introduces the benefits offered by the technology of the adaptive image sensor.

The state of the art in imaging technology has produced sensors that

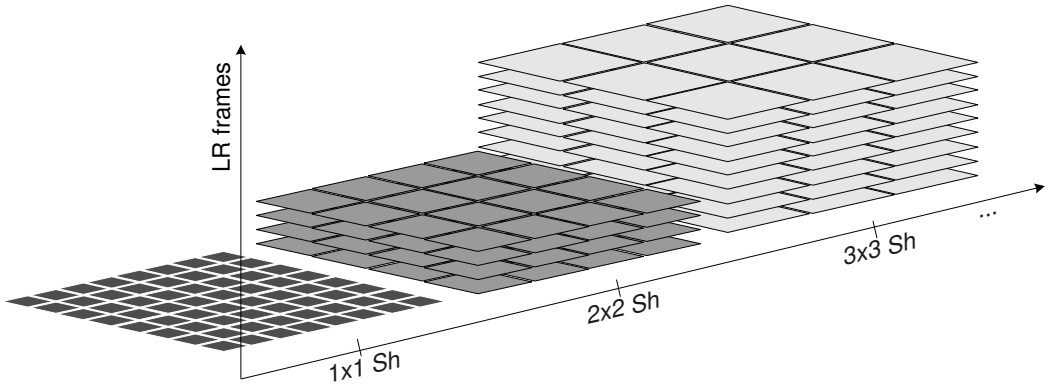


Figure 2.9: LR time samples produced during the HR integration for different configurations of an adaptive image sensor.

are no longer subject to the constraint of time and space invariant pixel size [Fov08, CDT06]. 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 on-line, real-time video enhancement. The sensor's configuration speed depends on the technology of the given sensor. The architectures that are proposed in this thesis can be combined with sensors of various configuration speeds, as will be made clear in the chapters that follow.

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 be reconfigured to larger pixel sizes and thus include

pixels whose size is larger than S_h. In that case, according to the space-time trade-off, this area produces 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 in the 3-D space should remain $m \times n$ for the given area. In one end, there is 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 Fig. 2.9. 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 2×2 S_h, the LR spatial resolution is 4 times lower than the HR resolution, while the temporal resolution is 4 times higher. In other words, 4 LR time samples are produced for Q during the HR integration, as demonstrated in the second configuration of the sensor of Fig. 2.9. If the spatial relation is 3×3, 9 LR samples are produced, as shown in the third configuration of Fig. 2.9.

2.5 Accelerator Devices

The algorithms related to video processing applications are cues of the most computationally challenging problems in the area of Digital Signal Processing (DSP). When real-time constraints are added to such applications, traditional software processing is inadequate, and other solutions should be employed. The choice between alternative accelerators depends on the specification of the target application regarding throughput, area, and power.

At one end, dedicated *Application-Specific Integrated Circuits (ASICs)* are undoubtedly the best solution if the objective is maximum performance for a fixed application. At the other end, *Digital Signal Processors (DSPs)* require significantly shorter design times than ASICs, provide considerably lower manufacturing costs, especially when the production scale is small, and offer the ability to be easily upgraded. To provide such benefits, DSPs trade-off performance and power consumption, being a lot slower and considerably more power-hungry compared to ASICs.

Between the two ends, *i.e.* ASICs and DSPs, lie the *Field Programmable Gate Arrays (FPGA)*. FPGAs are flexible semiconductor devices, in the sense that they can be programmed and configured after manufacturing. FPGAs offer a variety of advantages when it comes to the implementation of video processing algorithms. The same hardware can be used for the implementation of different algorithms or different versions of the same algorithm. Moreover, even if the specifications of a particular algorithm might not yet be complete,

the design on the FPGA can begin at once and be upgraded according to the final specifications, when these are ready. Such flexibilities are not offered by ASICs, and, compared to digital signal DSPs, FPGAs provide implementation efficiency, being much faster and less power-hungry.

Recently, *Graphics Processing Units (GPUs)* have developed from simple graphics rendering devices for personal computers into powerful general purpose processors [CCLW05]. The processing power of GPUs is recognized in various application domains, such as super-computing [KHDO06]. However, when compared to FPGAs, the performance of GPUs is considerably lower, mainly due to the following:

1. GPUs employ solely floating-point arithmetic representation. Floating-point representation can be used to accommodate values with highly varying dynamic range. This comes at the price of increased power consumption and circuit area. Contrary to GPUs, FPGAs offer the possibility of fine tuning the word-lengths of the circuit. By limiting the word-lengths to shorter bit-widths, more compact circuits are produced, thus reducing area and power consumption and increasing speed. The above trades-off system compactness and output quality. Based on the precision requirements of the given application, the designer can determine the word-length that corresponds to the optimal point with regards to the quality-compactness trade-off, or completely avoid the decrease in the quality. Extended work on word-length optimization for DSP sys-

tems is done by Constantinides [Con01].

2. Contrary to GPUs, FPGAs offer the possibility of custom pipelining. Increasing the number of pipelining levels increases the throughput, at the price of increasing the latency, area and power consumption. The increase in the latency is insignificant when the system produces long streams of outputs. Custom pipelining is a powerful weapon when specific system requirements are targeted.

3. The current heat dissipation of GPUs is considerably higher than FPGAs. In fact, moderate FPGAs do not use a fan to remove the heat as GPUs do.

2.6 Field Programmable Gate Arrays

2.6.1 The Device

The Field Programmable Gate Array (FPGA) was invented by a co-founder of Xilinx Inc. [Xil08], Ross Freeman, in 1984 [Xil04]. Nowadays, Xilinx Inc. leads the FPGA market along with Altera Corporation [Alt08].

An FPGA is mainly composed by a large number of routing channels and Configurable Logic Blocks (CLBs). The routing channels connect arbitrary CLBs together through switch-boxes. The CLB of a Xilinx Virtex-4 FPGA device contains four interconnected slices [Xil]. All slices contain two 4-input Look-Up Table (4-LUT), two storage elements, wide-function multiplexers,

carry logic, and arithmetic gates. In addition, two out of the four slices also include distributed RAM and 16-bit shift registers. In the most recent Xilinx device, the Virtex-5 device, 6-input LUTs have substituted the 4-input LUTs.

On-chip memories are extremely useful for temporary storage, as they increase the memory access locality and reduce the power consumption and I/O costs that are related to external memory accesses. Thus, in addition to distributed RAM, FPGAs include a large number of Block RAM memories (BRAMs). Virtex-4 devices feature 18 Kb dual-port BRAMs, whose number depends on the size of the device [Xil]. Each port has its own address, data in, data out, clock, clock enable, and write enable. Therefore, in every cycle, data may be written and read from either or both ports. As will be discussed in detail in the sections of this thesis that deal with implementation issues, the presence of dual-port BRAMs on FPGA can be utilized to maximize data reuse and increase parallelism.

2.6.2 Programming Languages and Tools

A number of programming tools and languages is available to facilitate the design process for FPGA implementation. These are associated with different levels of abstraction:

1. *Gate Level:* A designer can draw by hand the circuit schematic. This involves choosing elementary building blocks, *i.e.* logic gates, flip-flops and multiplexers, and graphically connecting them.

2. *Register Transfer Level:* Using a Hardware Description Languages (HDL), the designer can describe hardware at Register Transfer Level (RTL). In RTL the circuit is described as a set of registers and functions that determine the flow of data between the registers. The most popular HDLs are VHDL and Verilog.

3. *System Level:* Extensions to the C programming languages are nowadays developed, that allow the designer to focus at the system level and ignore the low level circuit. The most common among these are Handel-C [Cel05] and SystemC [Ini03].

When design complexity increases, moving from the gate level to the system level significantly reduces the design time. However, high level synthesis is still relatively new; thus, the system performance achieved with C-based languages cannot compete the performance obtained when HDLs are used in the design process. In this thesis, VHDL and Handel-C have been used. Specifically, VHDL [Ash95] has been used for the implementation of the 2-D DWT computation schedules in Chapter 3, while Handel-C [Cel05] has been employed for the more complex processing blocks of Chapters 5 and 6. Due to the fact that VHDL is a lower-level hardware description language than Handel-C, a lower level of description is adopted in Chapter 3, while the hardware blocks of Chapters 5 and 6 are described in a higher level and thus less detail.

2.7 External Memory

An FPGA-based system that implements a data-intensive algorithm requires an external memory block for the storage of the input and the output data. This off-chip memory needs a memory controller that coordinates the communication with the on-chip system. The control logic implemented by the controller depends on the technology of the external memory. A number of options are available, as far as the memory technology is concerned.

Two major technologies are Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM). The difference between these technologies is in the type of memory cell. In SRAM, each data bit is stored in a latched storage cell whose charge lasts as long as the memory remains powered, whereas the DRAM memory cell uses a capacitor for dynamic storage, and thus its content needs to be refreshed periodically. The controller of an SRAM is simpler than a DRAM controller, mainly due to the fact that no refresh cycles need to be considered.

Another criterion for categorizing memories is whether actions are initiated by a clock. Based on this criterion, memories can be either synchronous or asynchronous. Synchronous memories associate data and control signals to clock edges, whereas for asynchronous memories the data flow is solely controlled by address transition.

Standard synchronous memories require turnaround cycles between read and write operations. These idle cycles can be avoided by using Zero Bus

Turnaround (ZBT) memory devices. ZBT memories have zero latency between read and write cycles, thus maximizing the available bandwidth.

In this thesis, two types of memory have been considered. In Chapter 3, a standard Synchronous DRAM (SDRAM) is considered for the purposes of the power/energy estimation. This decision is based on the fact that DRAMs are more power-hungry than SRAMs, and thus power/energy results are provided for the worst-case scenario. In Chapters 5 and 6, the Celoxica ADMXRC4SX board [Cel08] has been employed for the FPGA implementation of the proposed architectures. This board offers 4 ZBT Synchronous SRAM (ZBT SSRAM) banks, and a built-in memory controller, which is accessed via hardware macros that are offered by the manufacturer [Cel08]. Therefore, the memory controller has been treated in this work as a black box, and issues related to its internal structure are out of the scope of this thesis. More information on such issues can be found in [Cel08, BHNC05, Lat06].

2.8 Summary

This chapter has provided a background of the previous, concurrent, and ongoing work in the area of computational photography and video. Various techniques are briefly presented, each related to one or more of the four basic elements of the imaging system, namely, the sensor, the optics, the processing unit, and the incident illumination. The chapter has focused on the problem of increasing the spatio-temporal resolution of the output frames, which is the

main interest of this thesis. The space-time trade-off related to image sensors has been discussed, along with techniques that have been proposed to surpass the trade-off. The proposed video enhancement system addresses this trade-off. As has been explained, the proposed system employs a reconfigurable sensor and a reconfigurable processing unit, which interact in an autonomous fashion, based on real-world information. The aim of the proposed system is to maximize the raw captured data and, with the appropriate processing, maximize reconstruction quality of the final output. The reconfiguration property of the adaptive image sensor [Fov08, CDT06], which makes feasible the above scheme, has been discussed in detail. Also, the different hardware options that could be employed for the processing have been presented, and the choice of FPGAs has been justified.

As has been mentioned in this chapter and will be demonstrated in the chapters that follow, FPGAs are excellent processing platforms for computational imaging systems. This is not only due to their flexibility and high speed, which enables real-time performance, but also due to their low area and power costs, that give low-power portable devices.

The first step of video reconstruction, which is based on multiple samples of low spatial resolution, is the registration of these samples. Reliable registration is a prerequisite to achieving high quality reconstruction with the subsequent reconstruction block. Therefore, before exploring the appropriate video enhancement techniques and presenting the proposed system, registration-related issues are discussed in the next chapter. In particular, the focus of Chapter 3

is on the pyramidal structure, which allows a multi-resolution approach to the registration problem. The work presented in Chapter 3 is a thorough investigation of low level hardware issues related to the implementation of the major two-dimensional Discrete Wavelet Transform computation schedules on FPGA, such as throughput, area, and energy dissipation. Readers who are mainly interested in the video enhancement that is proposed in this thesis may want to skip Chapter 3 and move on to Chapter 4.

Chapter 3

FPGA Implementation of the 2-D Discrete Wavelet Transform

The two-dimensional Discrete Wavelet Transform (2-D DWT) is nowadays established as a key operation in image and video processing. The wide range of applications that utilize the DWT-based multiresolution decomposition framework includes image compression and motion estimation. In this thesis, motion estimation is of major importance, as it enables the registration and, subsequently, the correct fusion of the high frame-rate samples that are produced at the LR regions of the sensor. Since the performance of motion estimation is highly dependent on the multiresolution decomposition step, this chapter is devoted to the 2-D DWT that effectively implements the decomposition scheme. In particular, the chapter evaluates different architectures for the efficient implementation of the 2-D DWT on FPGA, and provides a detailed analysis of the above implementations. Readers who are mainly interested in

the video enhancement system that is proposed in this thesis may want to skip this chapter and go directly to Chapter 4.

The chapter begins with a discussion on the role of the DWT in multiresolution motion estimation. Specifically, Section 3.1 introduces the concept of motion vectors, which are the unknowns of the motion estimation problem. Then, in Sections 3.2 and 3.3, the concepts of multiresolution motion estimation and the wavelet pyramid are introduced.

The remainder of the chapter focuses on the DWT and investigates the implementation of the major 2-D DWT computation schedules on FPGAs. Section 3.4 introduces the main concepts related to the DWT. Section 3.5 presents implementation decisions that are, for comparison reasons, common in the different FPGA implementations of the different 2-D DWT computation schedules. This decisions mainly concern the off-chip storage and the filterbank that implements the 1-D DWT. Sections 3.6.1, 3.6.2 and 3.6.3 describe the FPGA-based architectures of the three main 2-D DWT computation schedules. Section 3.8 evaluates the different FPGA-based architectures, with respect to the throughput, the number of FPGA slices, and the memory and energy requirements. Finally, Section 3.9 briefly discusses what type of optimizations are more appropriate for each schedule.

Parts of this chapter, related to the implementation of the 2-D DWT computation schedules on FPGAs, have been published in [AMCA08] and [AMCA06].

3.1 Motion Vectors: Definition

A motion vector is the projection of the three-dimensional motion of an object onto the two-dimensional plane. The group of motion vectors for each frame of a given video sequence is also known as the *optical flow field* or the *image velocity field*. Formally, a motion vector is defined as follows.

Definition 1. *Let f_1 and f_2 denote two grayscale adjacent video frames, and let $f_1(\boldsymbol{x}) = f_1(x, y)$ and $f_2(\boldsymbol{x}) = f_2(x, y)$ be the grayscale pixel values at point $\boldsymbol{x} = [x\ y]^T$. Let $\boldsymbol{u} = [u_x\ u_y]^T$ denote a point of f_1, and let $\boldsymbol{v} = [u_x + d_x\ u_y + d_y]^T$ denote the location of $\boldsymbol{u}$ in f_2. Then, the motion vector $\vec{d} = [d_x\ d_y]^T$ is defined as the vector that minimizes the following residual function ϵ:*

$$\epsilon(\vec{d}) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (f_1(x, y) - f_2(x + d_x, y + d_y))^2 \tag{3.1}$$

where $\omega_x, \omega_y \in \mathbb{N}$.

The positive integers ω_x and ω_y thus determine the neighborhood of pixels on which the minimization function of (3.1) is applied. This neighborhood is also known as the integration window. The size of the integration window is, according to (3.1), equal to $(2\omega_x + 1) \times (2\omega_y + 1)$. Typical values for ω_x and ω_y are values between 2 and 7 pixels [Bou02].

3.2 Multiresolution Motion Estimation

The process of determining the motion vectors describing the motion of points between adjacent video frames is known as *motion estimation*. The points on which motion estimation is applied are usually referred as features. Therefore, the motion estimation problem is also known as *feature tracking*.

For the feature tracker to render an accurate solution, choosing the features that participate in the tracking process is an important step. It is thus common practice to identify the "good" features to track before the actual motion estimation stage. "Good" features correspond to physical points of high textural content. The high textural content of a feature usually signifies good tracking performance. However, if a feature derives from the image properties and not from the actual object properties, then the tracker will be disoriented. Therefore, "bad" features include points of high texture that do not actually exist in the real world, such as the intersection of a foreground object with the background. The problem of selecting good features, is addressed by Shi and Tomasi in the popular paper of [ST94]. The textural information of a given feature is evaluated based on eigenvalue computation, whereas the concept of "dissimilarity" is used to identify features that do not correspond to actual physical points [ST94]. A monitoring process is employed for "dissimilarity" to be identified, based on the concept that non-physical features experience large variations in time. The problem of selecting discriminative tracking features is also addressed in the more recent work of [CLL05b]. A very robust algorithm

for the detection of features is the Scale Invariant Feature Transform (SIFT) algorithm [Low04]. SIFT allows the extraction of features from images that correspond to substantially different affine distortions, viewpoints, levels of additive noise, and levels of illumination. However, in this thesis, the application of interest involves adjacent video frames. Between such frames the changes in scale, rotation, illumination, and noise are limited. Experiments have shown that the additional computational cost that makes SIFT robust to such issues would be redundant for the case of adjacent video frames, and that the Shi and Tomasi algorithm provides instead a good trade-off between algorithmic complexity and performance. Therefore, in the remainder of the thesis, the Shi and Tomasi algorithm is employed for feature selection.

As explained in [Bou02], a good feature tracker should be both accurate and robust. The accuracy of the tracker affects the local subpixel accuracy of the identified motion vector $\vec{d}$. For the accuracy of $\vec{d}$ to be high, the size of the integration window, determined by ω_x and ω_y, should be small [Bou02]. On the other hand, the robustness refers to the ability of the tracker to track a feature regardless of various changes, such as changes in the object size, the motion magnitude or the lighting conditions. Therefore, for the tracker to be robust, the integration window should be relatively large. Therefore, there is a trade-off between accuracy and robustness, which is directly related to the size of the integration window. Specifically, decreasing the window size, increases the local subpixel accuracy, but decreases the robustness of the tracking process.

The accuracy-robustness trade-off is normally tackled by incorporating

multiresolution analysis in the tracking process [Bou02]. In the multiresolution analysis, a frame is represented as a set of frames with different resolutions corresponding to different frequency bands. The motion activities at different resolutions represent the *same* motion structure at *different* scales. Therefore, the estimated motion vectors, which are computed in low resolution approximations of the frame, form the initial guess for the computation of motion vectors at a higher resolution, where they are refined to increase the accuracy. In other words, the optical flow output at each level is propagated and refined at the higher level, up to the highest resolution frame, which is in fact the original frame. A multiresolution "pyramid" is illustrated in Fig. 3.1. The base of the pyramid corresponds to *Level 0*, the highest level of decomposition that contains the original frame.

In summary, the overall pyramidal motion estimation occurs as follows. Let ℓ_i denote the i^{th} level of the decomposition pyramid, where ℓ_0 is the highest level, *i.e.* the original frame. First, the motion vectors V_m are computed at the lowest resolution, which corresponds to level ℓ_m. Vectors V_m are then propagated to the next level, *i.e.* level ℓ_{m-1}, in the form of an initial guess. That initial guess is refined at level ℓ_{m-1}, and thus the motion vectors V_{m-1} are computed. Vectors V_{m-1} are then propagated to level ℓ_{m-2}, where the refined vectors for ℓ_{m-2} are computed, and so on. The motion vectors are thus propagated to higher levels, where they are refined, and the above process is repeated for each level, up to level ℓ_0.

By adopting the pyramidal tracking algorithm, a residual motion vector is

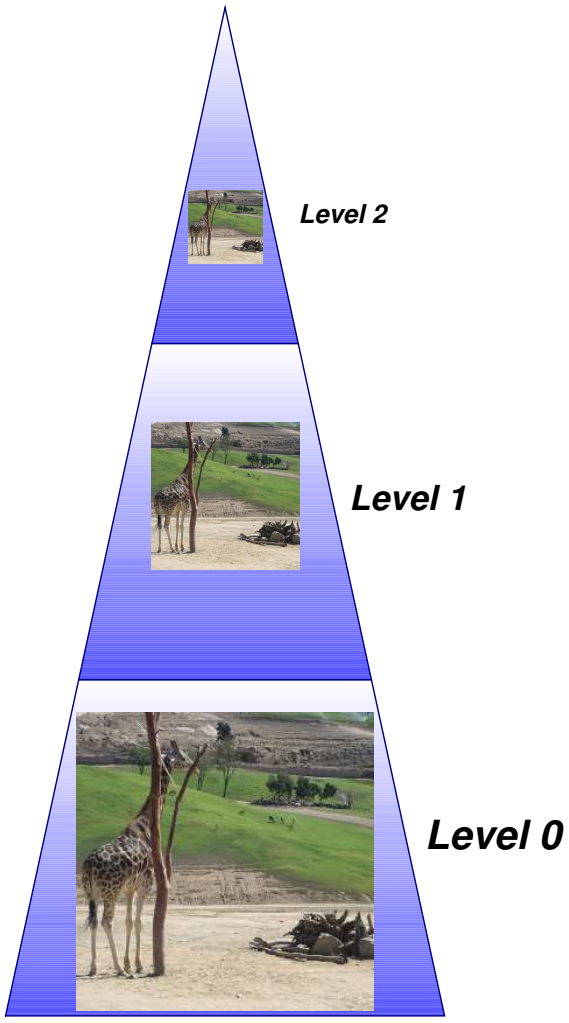


Figure 3.1: The multiresolution pyramid.

calculated at each level of the pyramid. Let $\ell + 1$ and ℓ denote two adjacent levels of the pyramid. Then, the residual motion vector that is calculated at ℓ is defined as follows.

Definition 2. *Let f_1^ℓ and f_2^ℓ denote two grayscale adjacent video frames at the resolution corresponding to level ℓ, and let $f_1^\ell(\boldsymbol{x}) = f_1^\ell(x, y)$ and $f_2^\ell(\boldsymbol{x}) = f_2^\ell(x, y)$ be the grayscale pixel values at point $\boldsymbol{x} = [x\ y]^T$. Let $\boldsymbol{u}^\ell = [u_x^\ell\ u_y^\ell]^T$ denote a point of f_1, and let $\boldsymbol{g}^\ell = [g_x^\ell\ g_y^\ell]^T$ denote the initial guess at level ℓ, which is available from computation executed from level ℓ_m to level $\ell+1$. Then, the residual motion vector $\vec{d}^\ell = [d_x^\ell\ d_y^\ell]^T$ is defined as the vector that minimizes the following residual function ϵ^ℓ:*

$$\epsilon^\ell(\vec{d}^\ell) = \sum_{x=u_x^\ell-\omega_x}^{u_x^\ell+\omega_x} \sum_{y=u_y^\ell-\omega_y}^{u_y^\ell+\omega_y} (f_1^\ell(x, y) - f_2^\ell(x + g_x^\ell + d_x^\ell, y + g_y^\ell + d_y^\ell))^2 \qquad (3.2)$$

where $\omega_x, \omega_y \in \mathbb{N}$.

It should be noted that in multiresolution motion estimation the size of the integration window remains constant, equal to $(2\omega_x + 1) \times (2\omega_y + 1)$, for all the levels of the pyramid, *i.e.* for all values of ℓ. By using the initial approximation $\mathbf{g}^\ell$ at level ℓ, the residual motion vector $\vec{d}^\ell$ is small and thus easy to compute [Bou02].

3.3 The Multiresolution Pyramid

In multiresolution analysis, the smoothed versions of the original frame at different resolutions form a pyramidal structure. An example of a multiresolution pyramid is the Gaussian pyramid. The Gaussian pyramid is formed by applying the Gaussian function as the smoothing function at every level. However, the Gaussian pyramid includes large amount of redundancy between different levels of decomposition [ZAS06].

A more systematic approach for the construction of the pyramid is provided by the wavelet theory [ZAS06, ZSA01, ZZ92, CCA+07, MK98]. Wavelet decomposition provides an ideal trade-off for the processing of video frames [ZAS06, ZSA01]. In particular, at high frequences the wavelet transform is sharper in time[1], while at low frequences it is sharper in frequency. This is in contrast to the Gabor and Fast Fourier Transforms, which use a fixed resolution at all locations in the space/time frequency plane. Thus, the wavelet domain is very appropriate for the incorporation of the block matching technique [HCHT04, CLS+08, LCT+08], which is the popular approach to motion estimation. The good qualities of the wavelet decomposition with regards to motion estimation, render the wavelet transform extremely appropriate for multiresolution motion estimation techniques [ZAS06, ZSA01].

In the literature, motion estimation, registration, optical flow, and tracking

[1]The spatial domain and not the time domain is employed in this thesis. However, for consistency with the theory of the 1-D DWT the term "time" is used, referring to the horizontal axis on which the 1-D signal is propagated.

algorithms have been exhaustively discussed, both with respect to algorithmic performance and hardware implementation. Thorough surveys of image registration methods are presented in [Bro92, MV98, ZF03]. In [YJS06, Avi04, Avi07, PO06], robust tracking algorithms have been proposed. A method for highly accurate motion estimation is presented in [BW05], while [dBNS96, GS03] discuss techniques for robust optical flow estimation. When it comes to the hardware implementation of registration and motion estimation methods, a number of hardware architectures, targeting FPGAs in particular, have been proposed in [JLR03, McE06, DRP+06, DS07, MCL02]. Thus, the current chapter focuses on the critical process of constructing the wavelet pyramid, on which multiresolution motion estimation is based, and presents a systematic evaluation of alternative hardware architectures for the 2-D wavelet transform.

3.4 Introduction to the 2-D DWT

The two-dimensional Discrete Wavelet Transform (2-D DWT) is nowadays established as a key operation in image processing. In the area of image compression, the 2-D DWT has clearly prevailed against its predecessor, the 2-D Discrete Cosine Transform. This is mainly because it achieves higher compression ratios, due to the subband decomposition it involves, while it eliminates the "blocking" artifacts that deprive the reconstructed image of the desired smoothness and continuity [VH92]. The high algorithmic performance of the 2-D DWT in image compression justifies its use as the kernel of both the

JPEG-2000 still image compression standard [ISO00, ITU08] and the MPEG-4 texture coding standard [ISO98].

3.4.1 The dyadic decomposition

The 2-D DWT can be considered as a "chain" of successive levels of decomposition as depicted in Fig. 3.2. Because the 2-D DWT is a separable transform, it can be computed by applying the 1-D DWT along the rows and columns of the input image of each level during the horizontal and vertical filtering stages. Every time the 1-D DWT is applied on a signal, it decomposes that signal in two sets of coefficients: a low-frequency and a high-frequency set. The low-frequency set is an approximation of the input signal at a coarser resolution, while the high-frequency set includes the details that will be used at a later stage during the reconstruction phase.

This procedure, presented in Fig. 3.2, is known as the *dyadic decomposition* of the image, and its impact upon the image's pixels can be presented by the diagram of Fig. 3.3, for the case of three decomposition levels. The shaded areas in Fig. 3.3 represent the low-frequency coefficients that comprise the coarse image at the input of each level.

Let us briefly describe the steps of this decomposition. The input of level j is the low-frequency 2-D subband LL_j, which is actually the coarse image at the resolution of that level. In the first level, the image itself constitutes the LL image block (LL_0). The coefficients L (H), produced after the horizontal filtering at a given level, are vertically filtered to produce subbands

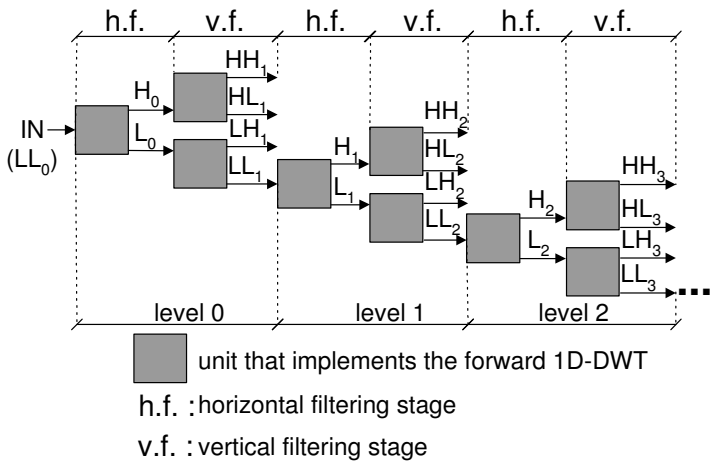


Figure 3.2: The 2-D DWT decomposition as a 'chain' of successive levels.

LL and LH (HL and HH). The LL subband will either be the input of the horizontal filtering stage of the next level, if there is one, or will be stored, if the current level is also the last one. All LH, HL and HH subbands are stored, to contribute later in the reconstruction of the original image from the LL subband.

3.4.2 The 1-D DWT

In Fig. 3.2, the units that implement the 1-D DWT are depicted as black boxes. These are now considered in detail. Two main options exist for the implementation of 1-D DWT: the traditional convolution-based implementation [Mal89] and the lifting-based implementation [DS98, ACA02].

Convolution-based 1-D DWT: The conventional convolution-based 1-D DWT of [Mal89] is presented in Fig. 3.4(a). As shown in Fig. 3.4(a), this consists of two analysis filters, h (low-pass) and g (high-pass), followed by

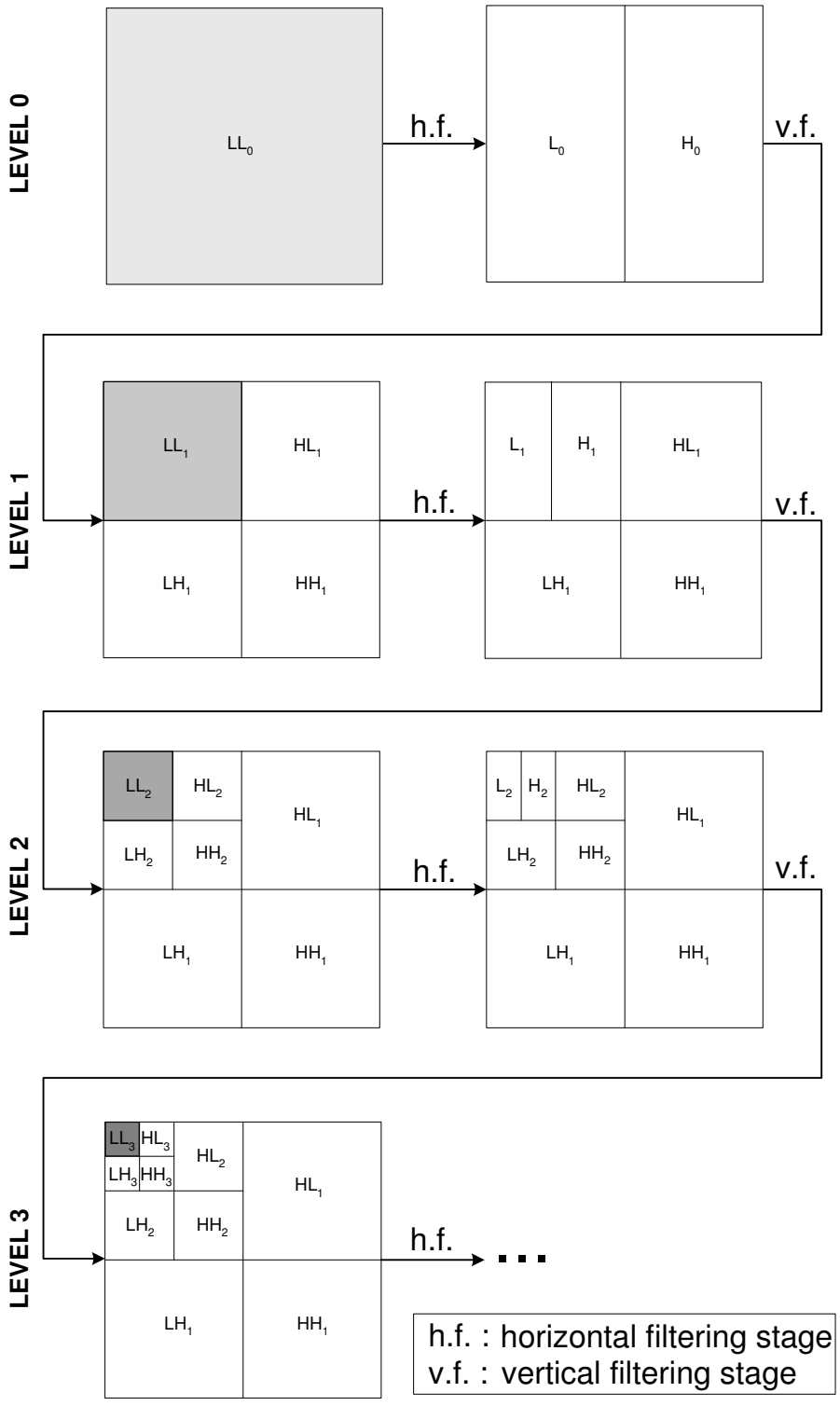
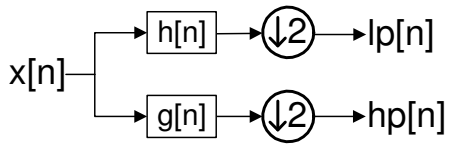
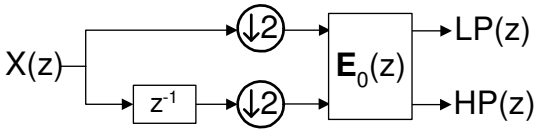


Figure 3.3: Diagrammatic representation of the dyadic decomposition for three decomposition levels.



(a)



(b)

Figure 3.4: The convolution-based implementation of the forward 1-D DWT. (a) The conventional filtering-and-downsampling structure. (b) Using the polyphase matrix of the analysis filter-bank.

subsampling units. The signal $x[n]$ is decomposed into the approximation (low-frequency) signal $lp[n]$ and the detail (high-frequency) signal $hp[n]$. Note that in the structure of Fig. 3.4(a) the downsampling is performed after the filtering has been completed. This is clearly inefficient since, in this case, half of the calculated coefficients are redundant, and the filtering is realized at full sampling rate.

Early research on filter-bank design proved that the execution of 1-D DWT can be accelerated by using the polyphase matrix of the filter-bank, instead of the conventional filtering-and-downsampling structure of Fig. 3.4(a). As Fig. 3.4(b) shows, the signal is split into two signals (polyphase components) at half of the original sampling rate. The downsampling is now performed prior to the actual filtering, thereby avoiding the calculation of coefficients that will

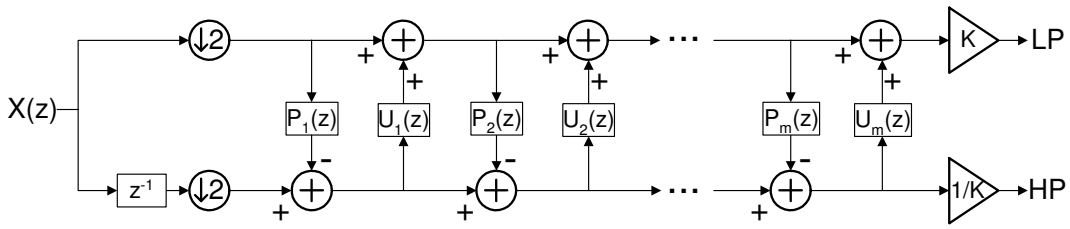


Figure 3.5: The lifting-based implementation of the forward 1-D DWT.

later be discarded. The polyphase components of the signal are filtered in parallel by the corresponding filter coefficients, producing the same result as if the downsampling was performed as described in [Mal89].

The analysis polyphase matrix for Fig. 3.4(b) is defined (in the Z-domain) as:

$$\mathbf{E}_0(z) = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix}, \tag{3.3}$$

where $H_e(z)$ and $H_o(z)$ denote the Type-I even and odd polyphase components of the corresponding low-pass analysis filter, and $G_e(z)$ and $G_o(z)$ denote the Type-I even and odd polyphase components of the corresponding high-pass analysis filter.

Using the analysis polyphase matrix of (3.3), the wavelet decomposition can be written (in the Z-domain) as:

$$\begin{bmatrix} LP(z) \\ HP(z) \end{bmatrix} = \mathbf{E}_0(z) \begin{bmatrix} X_e(z) \\ X_o(z) \end{bmatrix}, \tag{3.4}$$

where $LP(z)$ denotes the approximation at the coarser resolution, $HP(z)$ denotes the detail signal, and $X_e(z)$ and $X_o(z)$ denote the Type-I even and odd polyphase components of the signal $X(z)$.

The convolution-based 1-D DWT suffers from high computational complexity and high memory utilization requirements [ZAS+01].

Lifting-based 1-D DWT: The *lifting scheme* reduces the transform computational requirements by factorizing the polyphase matrix of the DWT into elementary matrices.

The principles of the forward lifting will be briefly discussed. More information on the lifting scheme can be found in [DS98, ACA02]. As it is proven in [DS98], if (H, G) is a perfect-reconstruction filter-pair, the following factorization of matrix $\mathbf{E}_0(z)$ of (3.3), into lifting steps, is feasible:

$$\mathbf{E}(z) = \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & U_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -P_i(z) & 1 \end{bmatrix}, \tag{3.5}$$

where K is a constant and m is the number of predict-and-update steps. They both depend on the type of filter-pair that is used.

Note that the lifting factorization of (3.5) is not unique. That is, for a given wavelet transform, multiple ways of factorizing its polyphase matrix can often be found.

The numerical factorization of (3.5) is represented by the schematic diagram of Fig. 3.5. The forward lifting scheme consists of the following steps:

1. The *splitting step*, where the signal is separated into even and odd samples.

2. The *prediction steps*, associated with the predict operator $P_i(z)$.

3. The *update steps*, associated with the update operator $U_i(z)$.

4. The *scaling step*, indicated by the scaling factors $1/K$ and K.

The inverse transform is realized by traversing the schematic of Fig. 3(b) from right to left (reverse signal flow) and switching the signs of the predict and update operators as well as the scaling factors. Because of the complete reversibility of the lifting scheme even if non-linear predict and update operators are used in the schematic of Fig 3(b), a rounding operator can be used to ensure integer form for all the data produced, throughout the execution of the 2-D DWT. In this case, the 2-D DWT is completely reversible, and, therefore, lossless. Such transforms are known as integer-to-integer transforms and are extremely useful in lossless coding. To build an integer version of a given wavelet transform, if scaling is present, then the scaling step should be either split further [CDSY98, DC97] or omitted. In fact, the scaling performed at the end of each decomposition level in the conventional decomposition can be skipped altogether [AK00], or incorporated into the subsequent encoding or processing stage of each bitplane and as a result it is not explicitly considered in this chapter.

Due to the lower computational cost, the flexibility offered in the transform factorization, and the perfect reconstruction property, the lifting-based

implementation is generally preferable for the design of optimized systems.

3.4.3 The 2-D DWT: Literature Review

Several computation schedules have been proposed, to implement the 2-D DWT. In practical designs, the most commonly used computation schedules are: the *row-column* (RC) [Mal89], the *line-based* (LB) [CO00] and the *block-based* (BB) [LNB+99]. The simplest of these is RC, which adopts the level-by-level logic of Fig. 3.2. However, such an approach necessitates the use of large off-chip memory blocks as the only source of the filter's inputs. Contrary to RC, both LB and BB involve an on-chip memory structure that operates as a cache for the original image, minimizing the accesses of the large memory blocks. Thus, memory utilization and memory-access locality are improved. The main difference among LB and BB concerns the way the original image is traversed. Specifically, in LB, non-overlapping groups of lines are processed, whereas, BB operates using non-overlapping blocks of the image.

Implementations of 2-D DWT computation schedules can be found in [Cas08] and [Amp08]. In [DGLC03] a combined lifting line-based FPGA implementation for the single-level 5/3 and 9/7 2-D DWT is presented.

In [ZAS+01], [WB03] and [CVO96], 2-D DWT computation schedules have been compared on a theoretical basis. In [ASL+03] and [MAS06], they are compared on programmable architectures and on a VLIW DSP, respectively. Even though the above comparisons are particularly enlightening, none of them is based upon hardware implementations, which would take advantage of the

implementation efficiency and the parallelism in data processing that hardware could offer. In addition, the vast majority of comparisons of the different alternatives focuses on convolution-based realizations and lifting is not considered. A discussion of VLSI architectures which also considers lifting-based DWT modules is presented in [CTHC05]. However, very few performance figures are presented for the multilevel lifting 2-D DWT, which is the case of interest in image coding applications.

In the remainder of the chapter, the three major 2-D DWT lifting-based computation schedules are implemented on FPGA-based platforms and compared in terms of performance, area and energy requirements. The computation schedules are compared for different image sizes (M×M) and number of levels (L) of the transform. This comparison will give significant insight on which schedule is most suitable for given values of the relevant algorithmic parameters, such as the size of the frames and the number of decomposition levels.

3.5 Common Implementation Decisions and Assumptions

The comparison of the main 2-D DWT computation schedules is performed on the basis of some common implementation decisions and assumptions. These are concerned with the memory architecture used to store the image (*image memory*) and the filtering structure used to compute the DWT.

3.5.1 Image memory

In this comparison, a single-port image memory is considered. The image memory is usually off-chip. However, it may also be on-chip, when dealing with small frames and/or using large FPGA devices. Considering the fact that the image memory can sometimes be on-chip, the comparison is made as generic as possible by using a common clock for the image memory and the rest of the system.

Employing a single-port image memory in the analysis adds flexibility as far as the available memory is concerned, as many boards don't include multi-port large memory blocks. Also, since the image memory is usually off-chip, the fact that single-port off-chip RAMs consume less energy per access than multi-port ones is something to consider [ZAS+01, CTHC05]. In data-intensive algorithms, such as the 2-D DWT, memory accesses are highly frequent. Thus, when the image memory is off-chip, from an energy perspective single-port RAMs are ideal, trading off the higher performance of multi-port RAMs [CTHC05].

Traditionally, two memory blocks are used in image processing systems: one to store the original image, and one for the outputs. To avoid the second block, the in-place mapping scheme is used: the filter's outputs are written over memory contents that are already consumed and no longer needed. To adopt this scheme, each memory location should have the same bit-width as the outputs of the transform.

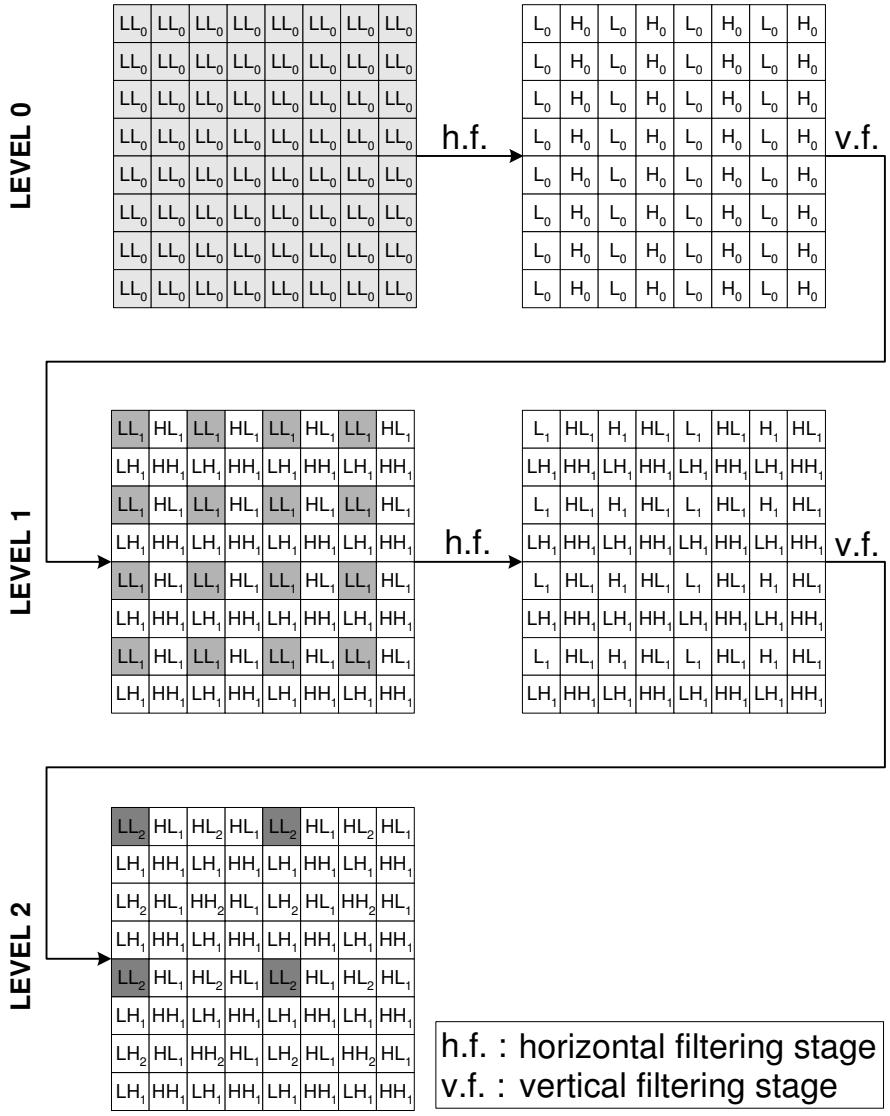


Figure 3.6: The in-place mapping scheme. The dyadic decomposition is applied on a hypothetical 8×8 original image.

The in-place mapping scheme is illustrated in Fig. 3.6. The dyadic decomposition is applied on a hypothetical 8×8 original image and the outputs of each stage are written in memory locations that have already been read and filtered. The shaded areas, in Fig. 3.6, represent the pixels of the coarse image at the input of each level. It might be interesting for the reader to compare Fig. 3.6 with Fig. 3.3, to see how the pixels of the different 2-D subbands, of the dyadic decomposition diagram, are distributed over the memory array. Note that in Fig. 3.6 every distinct square represents one pixel and bares the name of the 2-D subband in which it belongs. The decomposition of the hypothetical 8×8 original image cannot go beyond level 2, as there are not enough pixels in the LL_2 subband to proceed with the filtering operations.

3.5.2 Filter implementation

A single filter is used in all three implementations, introducing the minimum hardware cost. That is, the same single filter is *shared* among all levels and also among the vertical and the horizontal filtering stages within each level. The choice of using a single filter is mostly appropriate for an RC architecture that uses a single-port RAM.

The 5/3 lifting-based filter-pair: As for the type of filter, considering the advantages that the lifting scheme offers, a lifting-based filter is used, instead of a traditional convolution-based filter. Two types of lifting-based filter sets are mainly used in implementations of the DWT, the 5/3 lifting

filter-pair [ACA02] and the 9/7 lifting filter-pair [DS98]. The 5/3 lifting filter-bank is used, as it is more hardware efficient: for the same number of pipeline stages, it has a significantly smaller critical path than the 9/7 lifting filter-bank, while occupying significantly smaller area. This is due not only to the smaller number of lifting steps required, but also to the simplicity of the multiplying units: only two simple multiplications are involved (by (1/2) and by (1/4)), that can be implemented with simple shifters.

For the conventional 5/3 filter-pair, the analysis low-pass filter h (Fig. 3.4) has 5 coefficients, while the analysis high-pass filter g (Fig. 3.4) has 3 coefficients. The filter coefficients are the following:

- Low-pass filter h: $\{-1/8, 2/8, 6/8, 2/8, -1/8\}$
- High-pass filter g: $\{-1/2, 1, -1/2\}$

The factorization of the polyphase matrix of the conventional 5/3 filter-bank renders two elementary matrices. Therefore, the lifting version of the 5/3 filter-bank will include only one predict-and-update step, or equivalently m will equal to 1 (see Section 3.4.2). For each pair of input samples, $2n$ and $2n+1$, the lifting equations of this filter-bank are the following:

$$HP[2n+1] = X[2n+1] - \left\lfloor \frac{X[2n] + X[2n+2]}{2} \right\rfloor \tag{3.6}$$

$$LP[2n] = X[2n] + \left\lfloor \frac{HP[2n-1] + HP[2n+1] + 2}{4} \right\rfloor \tag{3.7}$$

where X are the signal samples, HP is the high-frequency output coefficient,

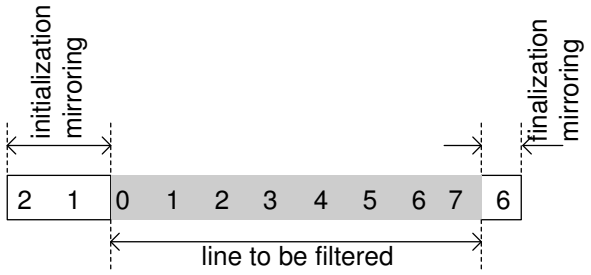


Figure 3.7: Mirroring at the borders of an 8-pixel incoming line, for a 5/3 lifting filter-bank.

LP is the low-frequency output coefficient, and $\lfloor . \rfloor$ represents the *floor* operator. The floor operator ensures an integer-to-integer lossless transform.

Hardware implementation of the 5/3 lifting filter-pair: In the hardware implementation of the filter, a three-word FIFO stores inputs $X[2n+1]$, $X[2n]$, $X[2n+2]$, and a register stores $HP[2n-1]$, which was calculated in the previous lifting step. According to the above equations, for a new pair of output coefficients to be computed, two (and not one) new filter inputs are needed. Thus, a filtering operation will take place, and a pair of output coefficients will be produced, every two cycles.

The hardware implementation of the 5/3 lifting filter-pair is shown in Fig. 3.8. Registers r1, r2 and r3 constitute the FIFO. Observing Fig. 3.9, one concludes that the filter's behavior is determined according to whether the initialization phase is over. The *initialization_signal* controls multiplexers m1 and m2, which determine if the data flows with a step of one or a step of two. Thus, the filter-bank might behave in one of the two following ways:

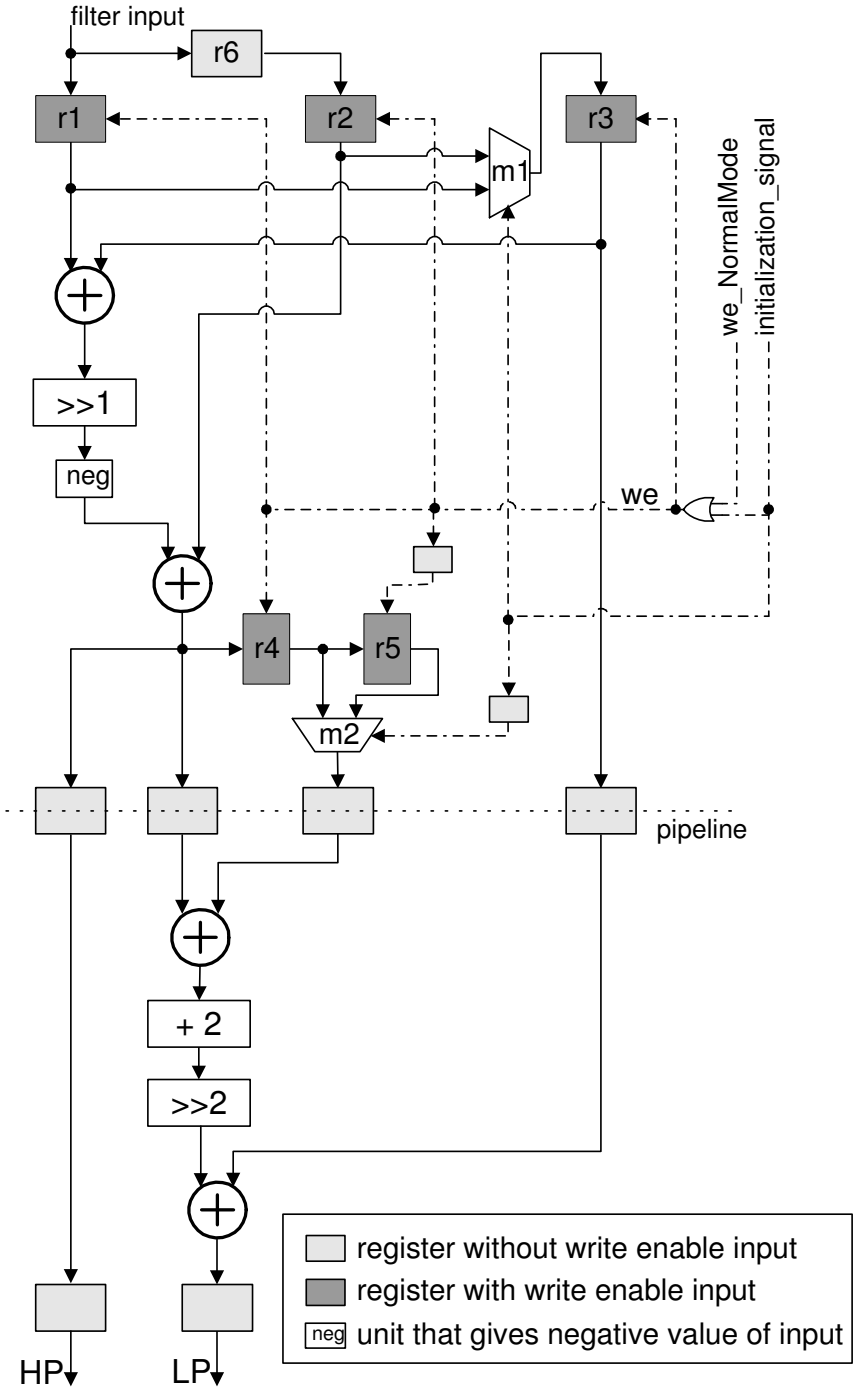


Figure 3.8: Hardware implementation of the 5/3 lifting filter-pair, designed to perform the 1-D DWT. The write enable signal (we) determines if the registers with write enable inputs will be written.

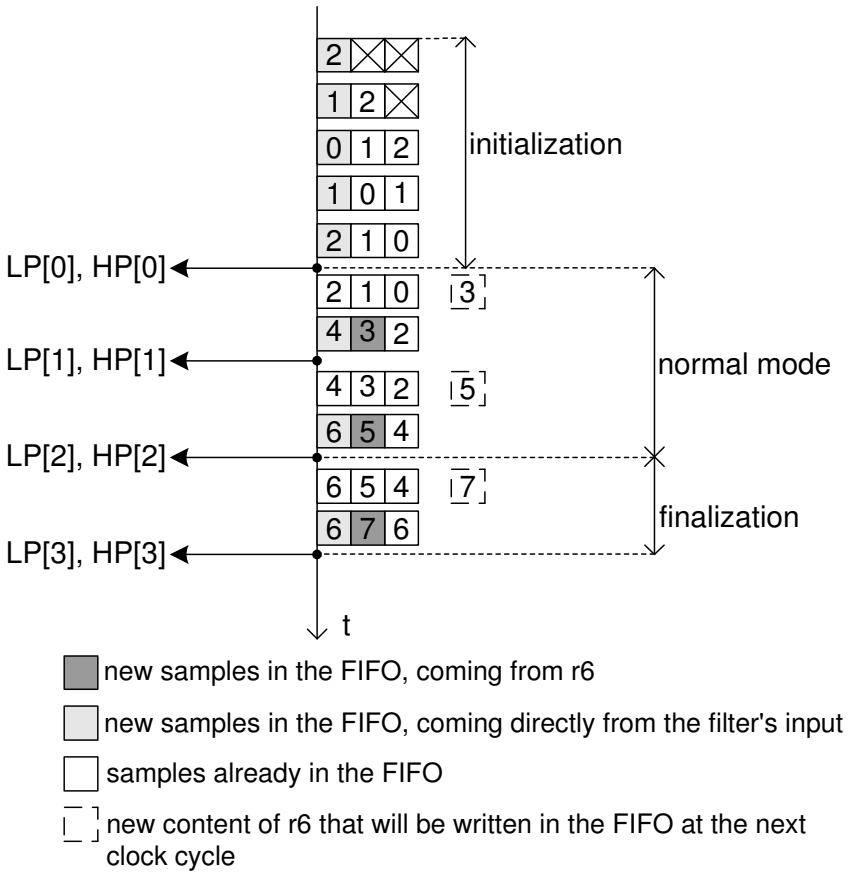


Figure 3.9: The contents of the FIFO in respect to time for the filtering of an 8-pixel line.

1. The *initialization mode*, which corresponds to the initialization phase. In this mode, the data flows with a step of one. Two samples are mirrored around the first sample, and a simple shifting takes place in the FIFO, as shown in Fig. 3.7.

2. The *normal mode*, where the data flows with a step of two. After the initialization phase has been over, the *we_NormalMode* signal will be forced to high when the FIFO should be written. Since the filtering operation should take place every two cycles, the *we_NormalMode* signal gets high every second cycle. Thus, the pattern shown in Fig. 3.9 is achieved. The same pattern applies for the finalization mirroring to be executed (Fig. 3.7), but the source of r1's input is now the FIFO itself, eliminating the need to access memory during this step.

In RC, only one input enters the filter-bank per cycle, since the single-port image memory is the only source of input coefficients. Thus, for RC, the filter-bank of Fig. 3.8 is ideal. On the contrary, as will be demonstrated in the following sections, both LB and BB involve multi-port on-chip buffers, that can supply the filter-bank with more than one inputs per cycle. In order to make the best possible use of the parallelism offered, a few small changes were made to the filter-bank of Fig. 3.8, to incorporate a multiple-input function (Fig. 3.10). Now, two or three inputs can be inserted in the filter-bank at a single cycle. However, during the horizontal filtering at level 0, the samples will be drawn from the single-port memory, just like in the case of RC. Thus,

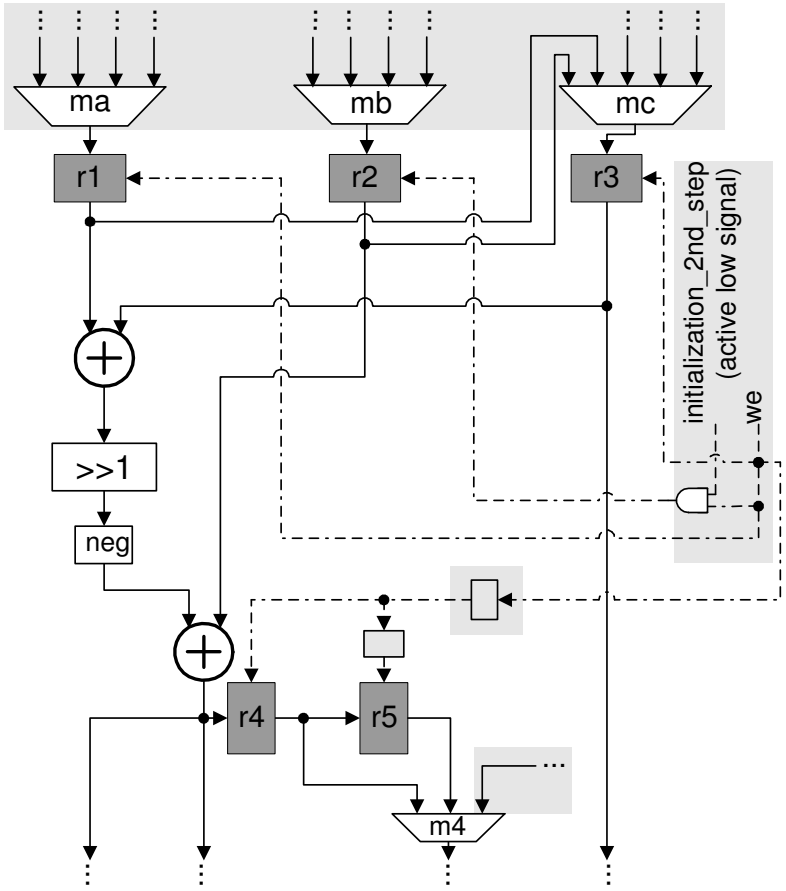


Figure 3.10: This filter-bank derives from that of Fig. 3.8, by applying a few changes (the shaded areas), to incorporate a multiple-input function. The lower part is not shown, as it remains the same.

during the horizontal filtering at level 0, the new filter-bank behaves in a single-input mode, apart from the multiple-input mode in which it functions in any other case.

3.6 Implementing the Computation Schedules

In this section, the implementation of the RC, LB and BB 2-D DWT computation schedules is described. Fig. 3.11 presents a generic block diagram of a 2-D DWT FPGA-based system that uses an off-chip image memory (which is the most common case). The dashed line indicates that on-chip buffers are not used in all of the three schedules.

It should be noted that, given the off-chip memory limitations, the implementations of the three schedules have been realized targeting the optimal use of the memory bandwidth. Thus the implementations obtain maximal throughput, for the given design parameters. Performance issues, including throughput and off-chip memory accesses, are discussed in detail in the performance evaluation section.

3.6.1 Row-column Implementation

The RC is implemented by applying the forward 1-D DWT in both the horizontal and the vertical direction of the image, for a chosen number of levels, in the way shown in Fig. 3.3. Specifically, in any given level, in order to proceed to the vertical filtering, of the current level's LL image block, the horizon-

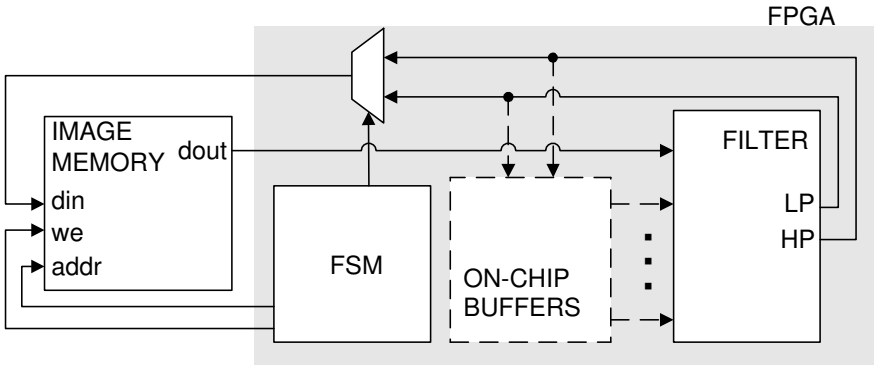


Figure 3.11: Generic block diagram of a system which executes the 2-D DWT using an off-chip image memory.

tal filtering should be complete. In addition, in order to proceed to the next level, the filtering at the previous level should be finished. Figures 3.12 and 3.13 present the flowchart and the block-diagram of RC, as implemented. Related to the generic block-diagram of Fig. 3.11, in the block diagram of RC (Fig. 3.13) the on-chip buffering block is omitted.

The RC architecture is the one with the simplest control path. The parallelism achieved during the filtering operations depends on the number of ports of the image memory. Its major disadvantage is the lack of locality, due to the use of off-chip large memory blocks. This decreases the performance and increases the power consumption, as it will be demonstrated in the performance evaluation section.

3.6.2 Line-based Implementation

Fig. 3.14 presents the flowchart of the low-level LB algorithm, as implemented. In Fig. 3.14 r denotes the current row of the original image and j denotes the

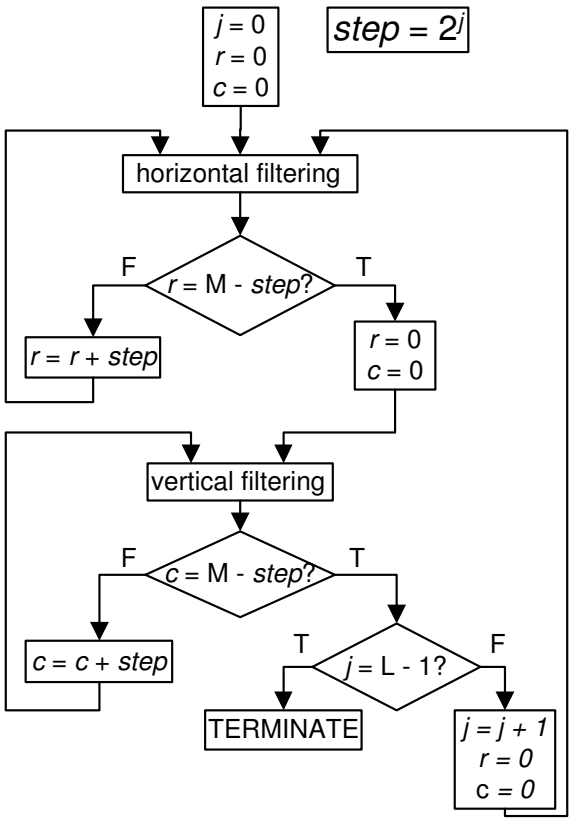


Figure 3.12: Flowchart of the RC algorithm as implemented (r/c = current row/column, j = current level).

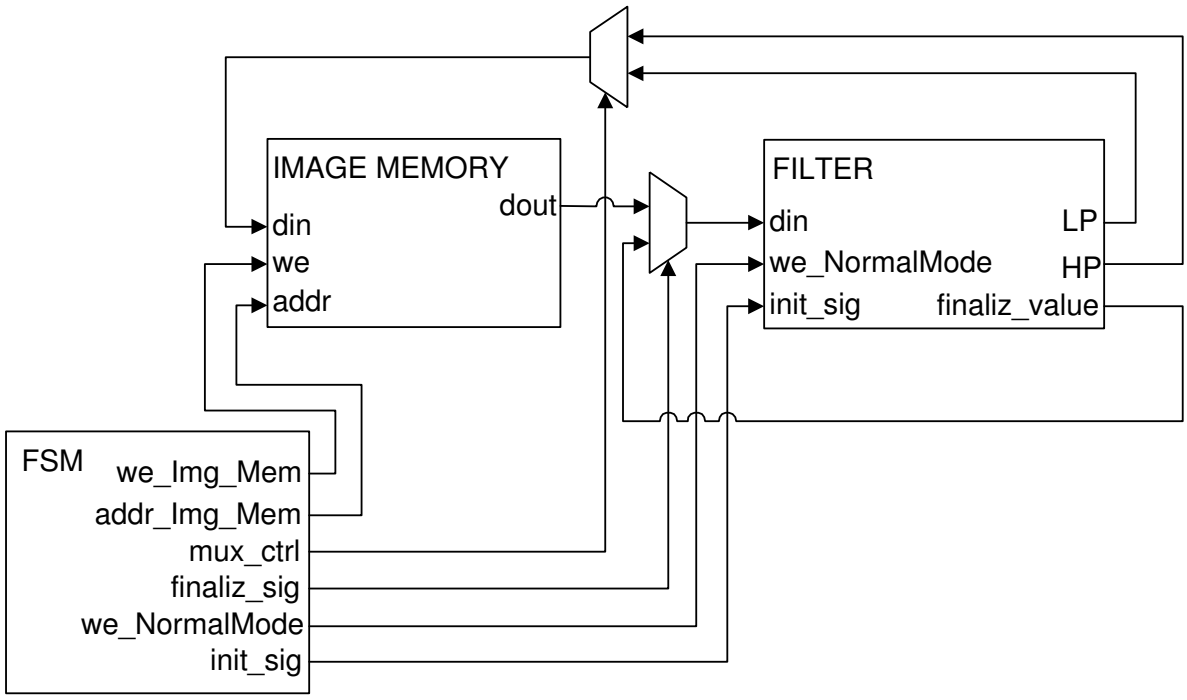


Figure 3.13: Block diagram of the RC architecture.

current level of processing. The counter $COUNT(j)$ determines (a) if there exists sufficient information for a vertical filtering to occur at level $j+1$ and (b) which is the filtering mode of the next vertical filtering at level $j+1$. Specifically, if $COUNT(j) = 4$, a vertical filtering will occur at level $j+1$ in initialization mode, whereas if $COUNT(j) = 2$, it will occur in normal mode. If $COUNT(j) = 3$ or $COUNT(j) = 1$, there is not sufficient information for a vertical filtering to occur at the current stage, but as soon as enough coefficients are produced a vertical filtering will occur in initialization or normal mode, respectively.

Fig. 3.15 shows the block diagram of the LB architecture. Contrary to the RC, the LB uses on-chip buffers. The on-chip buffers used in all of the

levels are included in the shaded area of Fig. 3.15. In Fig. 3.16 the buffers of level j are isolated. The LB uses these on-chip buffers, to store coefficients of intermediate levels which will be used at subsequent levels. This improves the memory-access locality compared to RC, and, hence, improves the performance. Moreover, contrary to RC where the single-port image memory imposes a serial nature to the filtering operations, these buffers may be multi-port to increase parallelism.

A group of lines is processed up to the final level, and a filter's output is stored in image memory only if it will be used as it is during reconstruction and not again during decomposition. Thus, LH, HL and HH coefficients are stored after the vertical filtering at any level, whereas the LL coefficients are stored only at the last level (Fig. 3.14).

After the completion of a vertical filtering at level $j-1$, the resulting LL_j coefficients are written in R(j). Buffer R(j) will then be horizontally filtered and the resulting coefficients will be written, depending on the current stage of level j's vertical filtering, in one of the following: C(j), R(j) (implementing the in-place mapping scheme) or buf1(j) (only during the initialization phase of the vertical filtering at level j, when buf1(j) is still empty).

Using the still empty buf1(j) during vertical initialization, the extra line buffer, which would store the extra information needed for the initialization mirroring to occur, is eliminated. When initialization is over, that is during the normal-mode vertical filtering, the value of the FIFO's first register (r1) and the high-frequency output of the current lifting step, are stored in buf1(j)

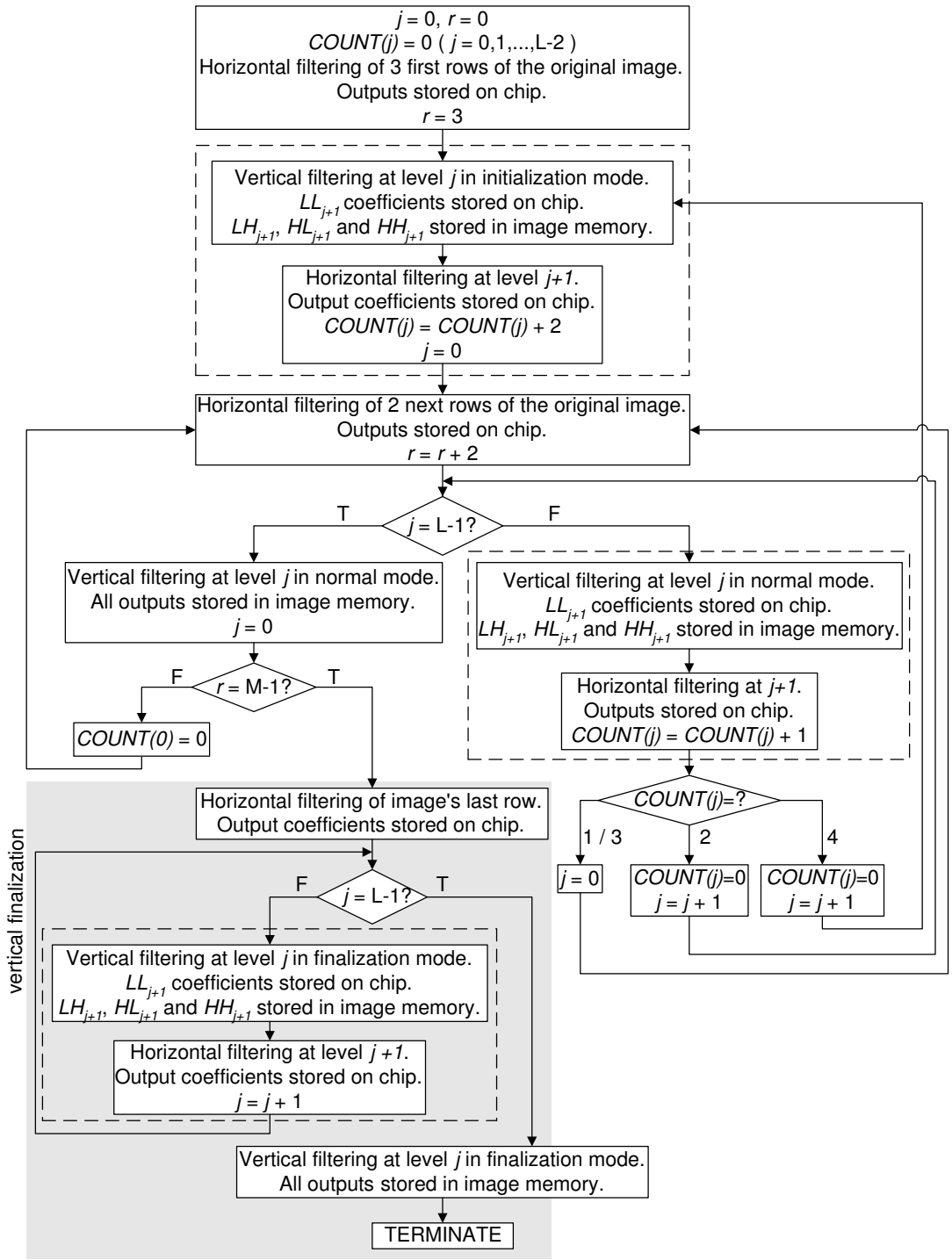


Figure 3.14: Flowchart of the LB algorithm as implemented (r = current row, j = current level).

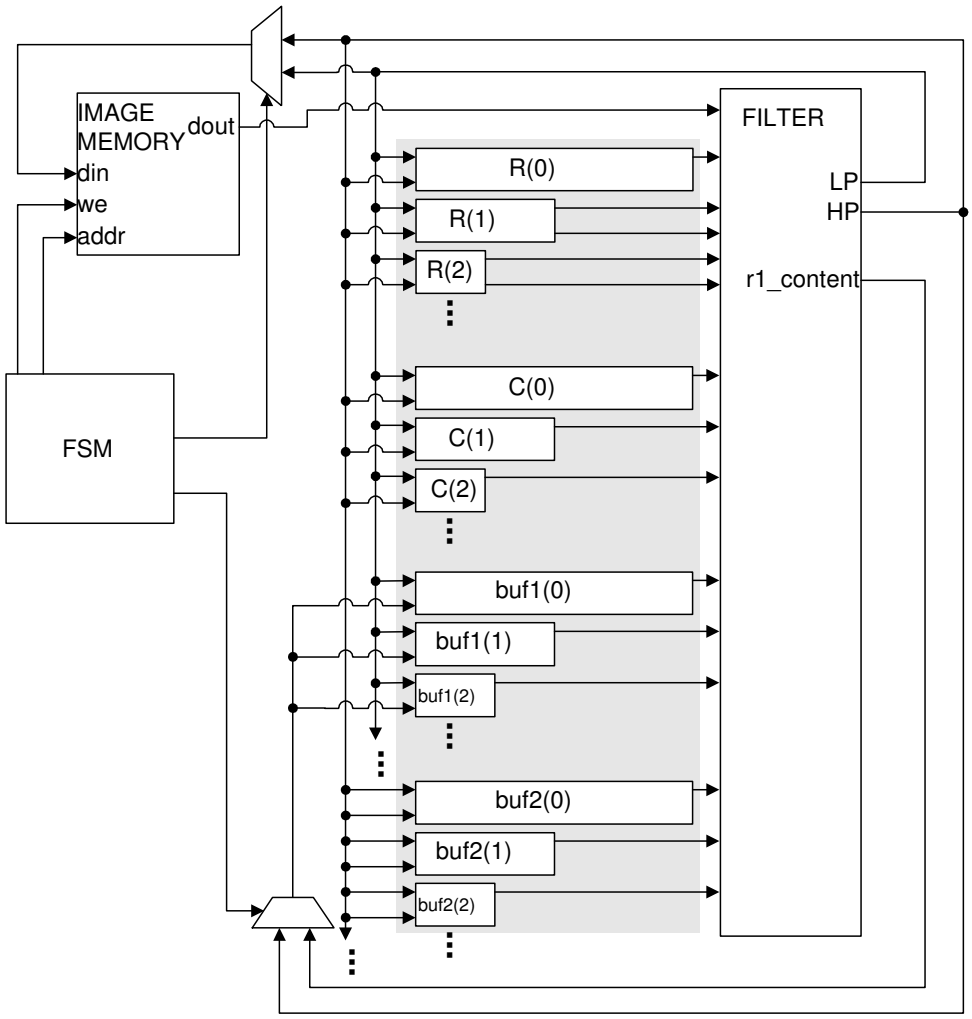


Figure 3.15: Block diagram of the LB architecture. The shaded area includes the on-chip buffers used in the architecture.

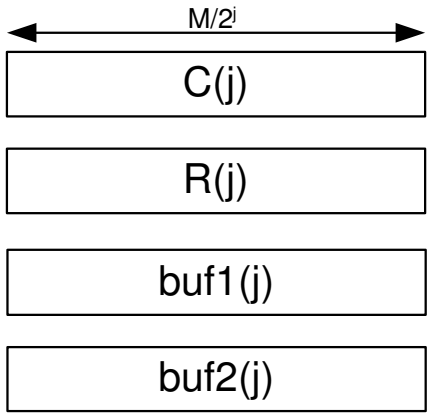


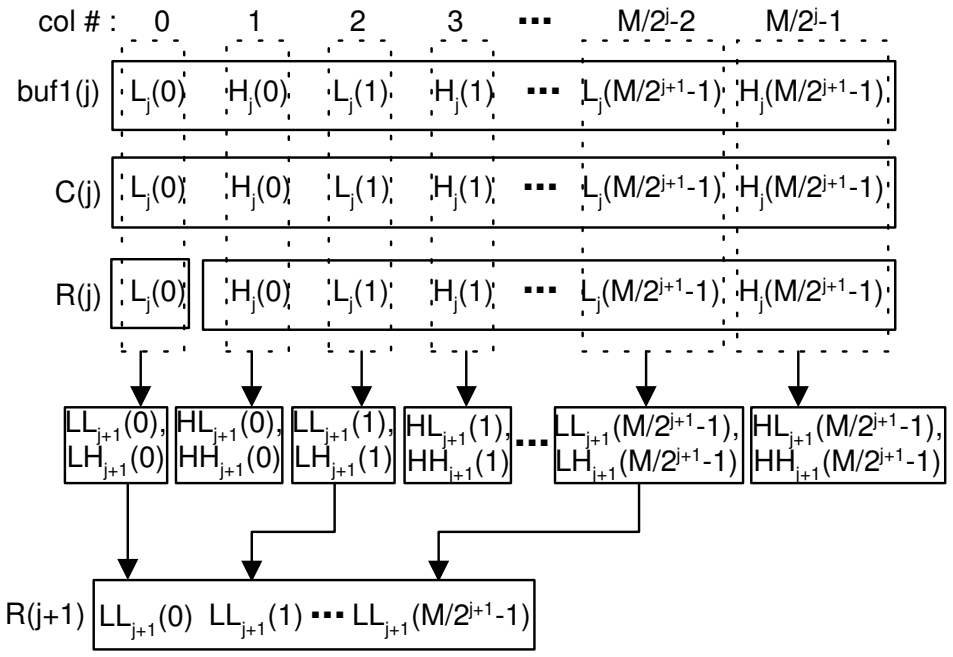
Figure 3.16: On-chip line buffers of level j, used in LB.

and buf2(j), respectively. Thus, the coefficients of the preceding lifting step are retrieved at the current step from buf1(j) and buf2(j). In this way, a continuity in the vertical filtering of each column is created, since in LB the vertical filtering, contrary to the horizontal filtering, is *not* inherently continuous. During vertical finalization, the values of buf1(j) is also the samples that are mirrored.

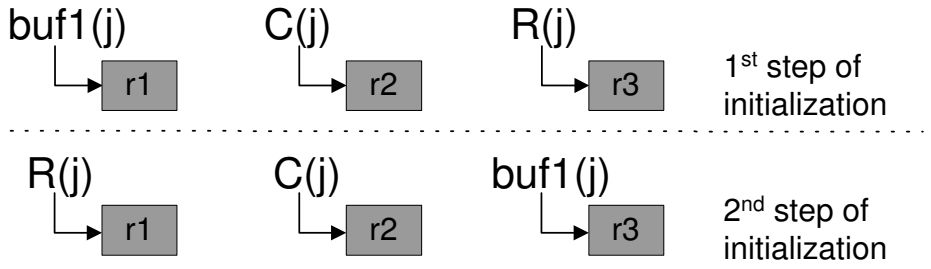
The filter used in LB, is that of Fig. 3.10, which incorporates a multiple-input mode to take advantage of the multi-port nature of the on-chip RAMs. The initialization and normal mode vertical filtering of each column is described in Figures 3.17(a) and 3.18(a). The multi-port filter's behavior that corresponds to each processing operation is illustrated in Figures 3.17(b) and 3.18(b).

3.6.3 Block-based Implementation

The BB is implemented bringing on chip blocks of the original image. Traditionally, the size of these blocks is equal to $2^L \times 2^L$, to allow the generation of



(a)



(b)

Figure 3.17: Vertical filtering in initialization mode in the LB architecture. The second step of initialization can be simplified avoiding the overwriting of r2 with the same value it already has, by forcing its *we* input to low. Also at the second step, r3 will be written with the previous value of r1.

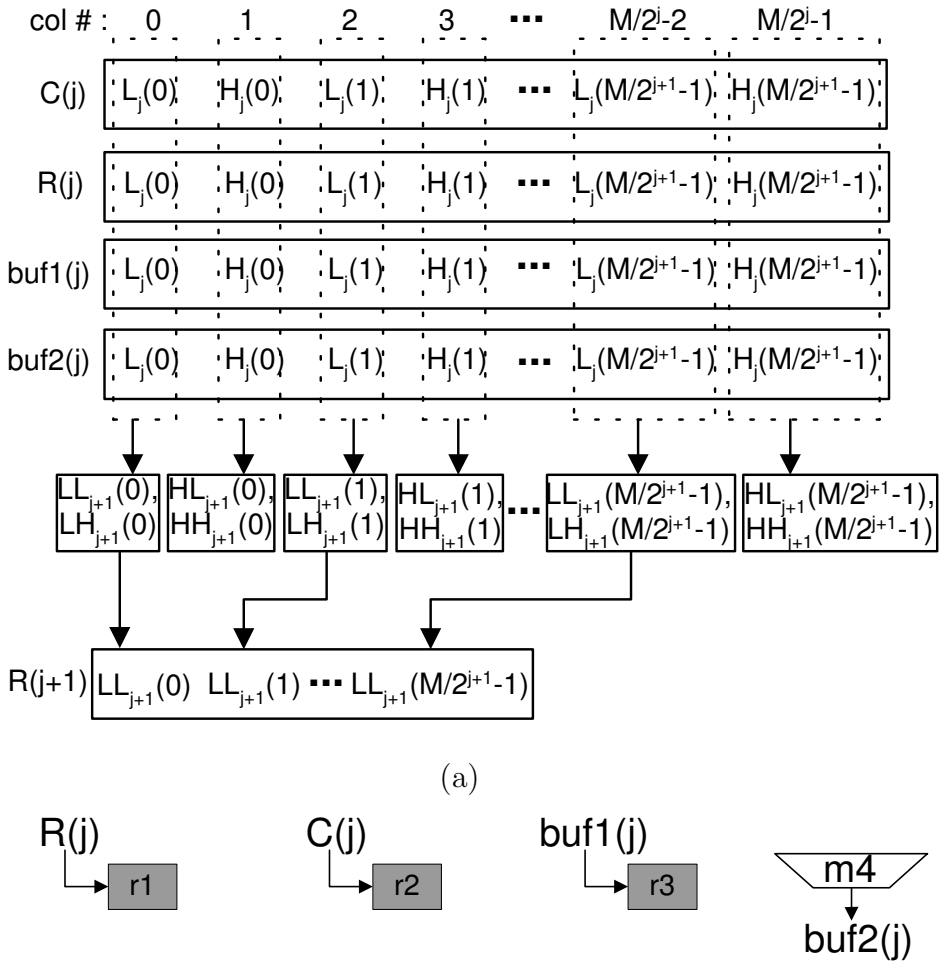


Figure 3.18: Vertical filtering in normal mode in the LB architecture. Three inputs are loaded in parallel into the FIFO, while buf2(j) passes through multiplexer m4 of the filter-bank.

either an LL_L/LH_L or an HL_L/HH_L pair, L denoting the final level. Thus, an on-chip memory of equal size should be used, where blocks are temporally stored. This memory is known as Inter-Pass Memory (IPM) [LNB+99]. The RC algorithm is then applied on the block, up to the last level, the decomposition of the block is written back to image memory, and the next block is brought on chip. The traditional version of BB demands complicated control and addressing [ZAS+01], is not effective in streaming applications and imposes high memory requirements (for six levels of decomposition the size of the IPM would be 4096 words regardless of the image size).

The BB version that is implemented is the one that requires the minimum size of local buffers and, also, simplifies as much as possible the control and the addressing. Each level has its own IPM, where coefficients LL are stored to be filtered horizontally. The size of IPM(j) is such that allows the generation of an L_j/H_j pair at level j. As have been demonstrated, for a new L_j/H_j pair to be generated, two new input coefficients should enter the filter-bank. Thus, the size of IPM(j), where $j = 1, 2, ..., L-1$, will be only 2 words. No IPM is needed for level 0, as the filter's inputs come straight from image memory.

In the previous section, it was shown that the vertical filtering is not inherently continuous in LB. In order for the vertical filtering to be executed correctly, a group of line-buffers was used. In BB, the horizontal - and not only the vertical - filtering is also deprived of inherent continuity. Therefore, apart from line buffers that will be used to create vertical continuity, in order to create continuity in the horizontal filtering, two double-word registers will

now be used in each level. Specifically, IPM(j) will store the input samples, that will enter the filter's FIFO, and bufH(j) will be needed to store the two intermediate results of horizontal filtering. So, the total storage place needed to create horizontal continuity is much smaller than that needed for the vertical filtering to be continuous. This is because the input image is traversed first horizontally and then vertically.

The block diagram of the BB architecture is shown in Fig. 3.19. The on-chip memory needed for level j is illustrated in Fig. 3.20. The control logic, implemented with the FSM, is a lot more complicated in the case of BB, compared to that of LB. Mainly, it differs from the control logic presented in Fig. 3.14 in the following:

1. Successive columns of L_j and H_j coefficients are no longer filtered vertically in a successive manner. After a vertical filtering of an even column occurs, if the current level is not also the final one, LL_{j+1} is either written in bufH(j+1) or in IPM(j+1), depending on the current stage of horizontal filtering at level $j+1$. A single step of horizontal filtering at level $j+1$ may occur (depending on the horizontal filtering stage) interrupting the vertical filtering of level j.

2. The horizontal filtering is no longer continuous. After a discrete step of horizontal filtering at level $j+1$, it should be decided if a vertical filtering at level $j+1$ can occur. If it *does* occur, a single step of horizontal filtering at level $j+2$ might follow, and so on. This domino effect in the

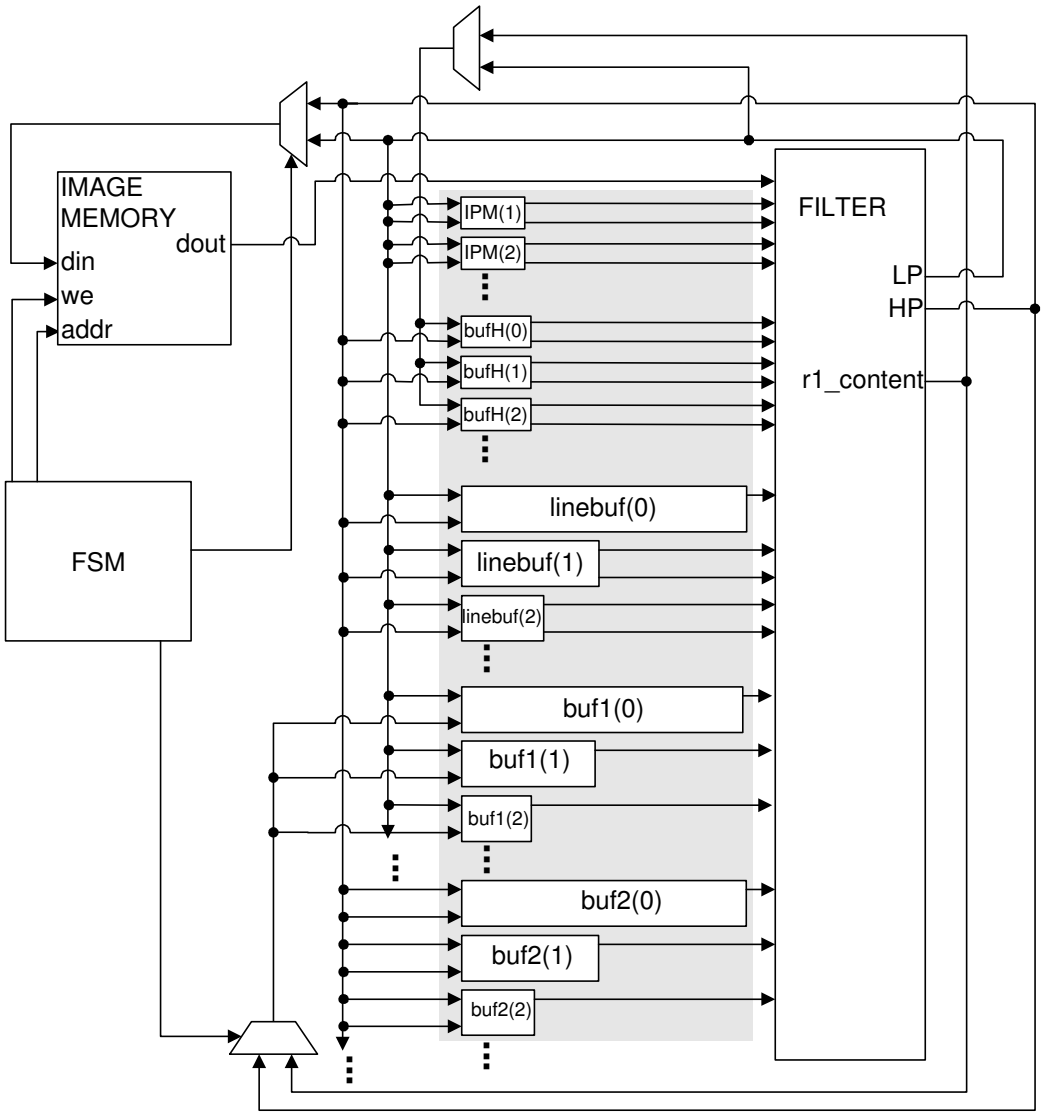


Figure 3.19: Block diagram of the BB architecture. The shaded area includes the on-chip buffers used in the architecture.

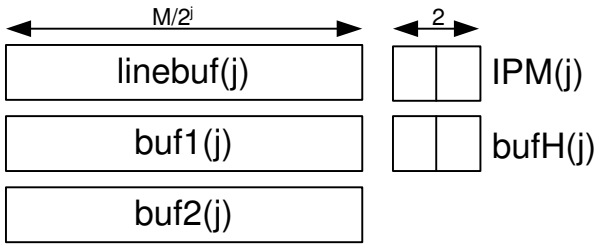


Figure 3.20: On-chip memory of level j, used in BB.

worst case reaches the final level, imposing highly frequent interchanges between successive levels, that complicate the control logic.

As in LB, the filter-bank used in BB should make full use of the parallelism multi-port buffers offer. Thus, the version of Fig. 3.10 is used, so that the filter-bank operates in a single-input mode during the horizontal filtering at level 0 and functions in a multiple-input mode in any other case.

In BB, during the vertical filtering stage, line buffers buf1(j) and buf2(j) will be used in exactly the same way as in LB.

An L_j/H_j pair, produced after the horizontal filtering of IPM(j), is either stored in a line-buffer (for the odd rows of level j) or consumed at once (for the even rows of level j) to produce either an LL_{j+1}/LH_{j+1} or an HL_{j+1}/HH_{j+1} pair (Fig. 3.21). Hence, in the case of even rows, a vertical filtering action is undertaken after the generation of every L_j/H_j pair, and no supplementary line-buffer is needed. On the contrary, in Fig. 3.18, in the even rows of vertical filtering, a whole row of L_j and H_j coefficients had to be written in R(j), so that continuous horizontal filtering would be applied on it. Only after the completion of the horizontal filtering of the even row of level j, the vertical

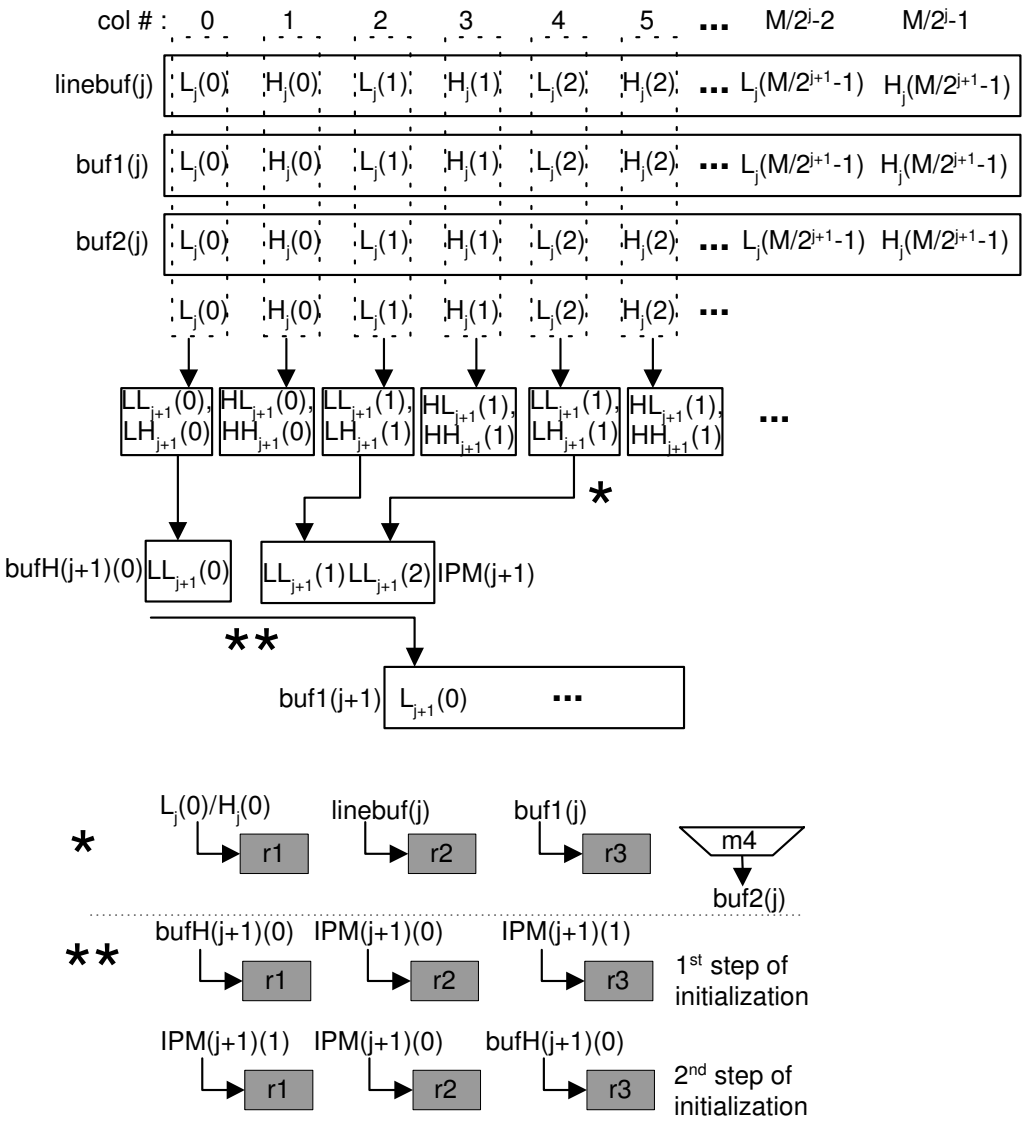


Figure 3.21: The normal-mode vertical filtering at level j, in BB, is followed by initialization-mode horizontal filtering at level $j+1$. Being at the beginning of the vertical initialization stage at level $j+1$, L_{j+1} is written in buf1(j+1).

Figure 3.22: Lena 512×512. The original image.

filtering would begin. As a result, the number of line buffers of BB is reduced by one, compared to that of LB.

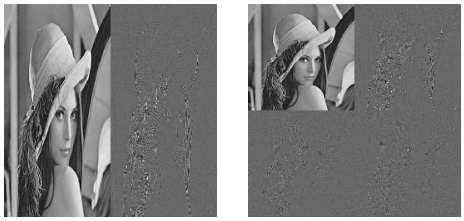
3.7 Output

Throughout the execution of the 2-D DWT, the decomposition is carried out as described in Section 3.4.1. The 2-D subbands that are produced during decomposition, for the chosen number of levels, are the same for all computation schedules. This is because the choice of a different schedule only affects the order in which the coefficients that form the subbands are produced and not their values. Their values solely depend on the type of filter used to execute the 1-D DWT. Since the same filter, the 5/3 filter-pair, is used in every schedule, the subbands produced are the same for the three schedules.

In the simulations, original images of sizes 256×256, 512×512, and 1024×1024 have been used. In this section, the 512×512 Lena image (Fig. 3.22) is

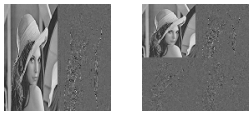
(a) (b)

Figure 3.23: Decomposition level 0. (a) Subbands L_0 and H_0. (b) Subbands LL_1, LH_1, HL_1 and HH_1.



(a) (b)

Figure 3.24: Decomposition level 1. (a) Subbands L_1 and H_1. (b) Subbands LL_2, LH_2, HL_2 and HH_2.



(a) (b)

Figure 3.25: Decomposition level 2. (a) Subbands L_2 and H_2. (b) Subbands LL_3, LH_3, HL_3 and HH_3.

used, to visualize the subbands produced during the decomposition executed by the hardware designs. This visualization is done in Figures 3.23-3.25. The decomposition is executed for 3 levels. The layout of Figures 3.23-3.25 follows that of Fig. 3.3, but only the new subbands produced at each level are now presented. Figures 3.23(a)-3.25(a) consist of the L_j and H_j subbands produced after the horizontal filtering of LL_j. Figures 3.23(b)-3.25(b) consist of the four subbands produced after filtering vertically L_j and H_j: subbands LL_{j+1}, LH_{j+1}, HL_{j+1} and HH_{j+1}. Specifically, Fig. 3.23(a) contains subbands L_0 and H_0, which are the outputs of the horizontal filtering of the original image. The vertical filtering of these renders subbands LL_1, LH_1, HL_1 and HH_1, which are presented in Fig. 3.23(b). Among these four, only subband LL_1 will be further processed during image decomposition, while the other three will be stored to contribute at a later stage in the reconstruction of the image. Thus, LL_1 will be horizontally filtered to render L_1 and H_1 (Fig. 3.24(a)), and so on.

When proceeding to a subsequent level, the resolution is decreased by a factor of 4. Therefore, since the input of level 0 (LL_0) is a 512×512 image (the original image), the input of level 1 (LL_1) is a 256×256 coarse approximation of the original image. Similarly, the inputs of levels 2 (LL_2) and 3 (LL_3), are of size 128×128 and 64×64, respectively.

3.8 Results and Comparisons

The architectures were implemented in VHDL, synthesized with Synplify Pro 7.7, and placed and routed on Xilinx Virtex 4 XC4VLX15 FPGA, using Xilinx ISE v.8.1. In all of the implementations, the image memory shares the same clock with the rest of the system. This simplification renders the comparison as generic as possible, since the image memory is sometimes on-chip (e.g. in large FPGA devices). Under this assumption, it should be noted that the computation schedules with larger traffic towards the image memory are favored, and no interface circuit is needed.

The current section presents a comparative analysis of the implementations, in terms of throughput, number of FPGA slices, memory requirements and energy consumption. These measurements, which are relative to the image size (M) and the number of decomposition levels (L), span a large parameter space. In order to deliver this information clearly, in all of the following subsections, a particular way of presenting the results is employed:

- A 3-D graph is used to present the measurements for each computation schedule, relative to a given pair of parameters, M and L. On that graph perpendicular slices are drawn at M=256, M=512 and M=1024.

- The 2-D information corresponding to each slice is displayed underneath with 2-D subgraphs. Because, in most cases the results vary a lot for different schedules, this 2-D information is organized in three graphs, one for each schedule. Note that these 2-D subgraphs are independent from

each other, using non-uniform scaling of the y-axis, in order to deliver accurately the associated non-uniform numerical variations.

- A common graphic notation is used in all 3-D graphs and 2-D subgraphs (i.e. ● is used for RC, △ for LB and ○ for BB).

3.8.1 Throughput

The RC, the LB and the BB operate on the XC4VLX15 device at 172, 113 and 117 MHz respectively. The three schemes have similar data paths; however, the frequency varies because the critical path lies in the control path.

The LB obtains the highest throughput among the three schedules. This is due to the small number of clock cycles it requires to complete the 2-D DWT (Fig. 3.26); it starts from 150,024 cycles (M=256, L=3), and reaches 2,374,740 cycles (M=1024, L=6). The throughput of LB reaches 757 frames/sec (M=256, L=3) and drops to 47 frames/sec (M=1024, L=6).

There is no great difference between the number of cycles needed for RC and BB. Specifically, RC needs 347,651-5,607,174 (Fig. 3.26), while BB requires 354,827-5,767,246 clock cycles (Fig. 3.26). However, because the RC operates in higher frequency, its throughput is improved, resulting in the difference observed in Fig. 3.27. If the image memory operates in a smaller frequency than the rest of the system, BB will outperform RC.

The larger number of cycles in RC, compared to LB, is due to the fact that a single-port memory is used as the only source of inputs for RC. Thus,

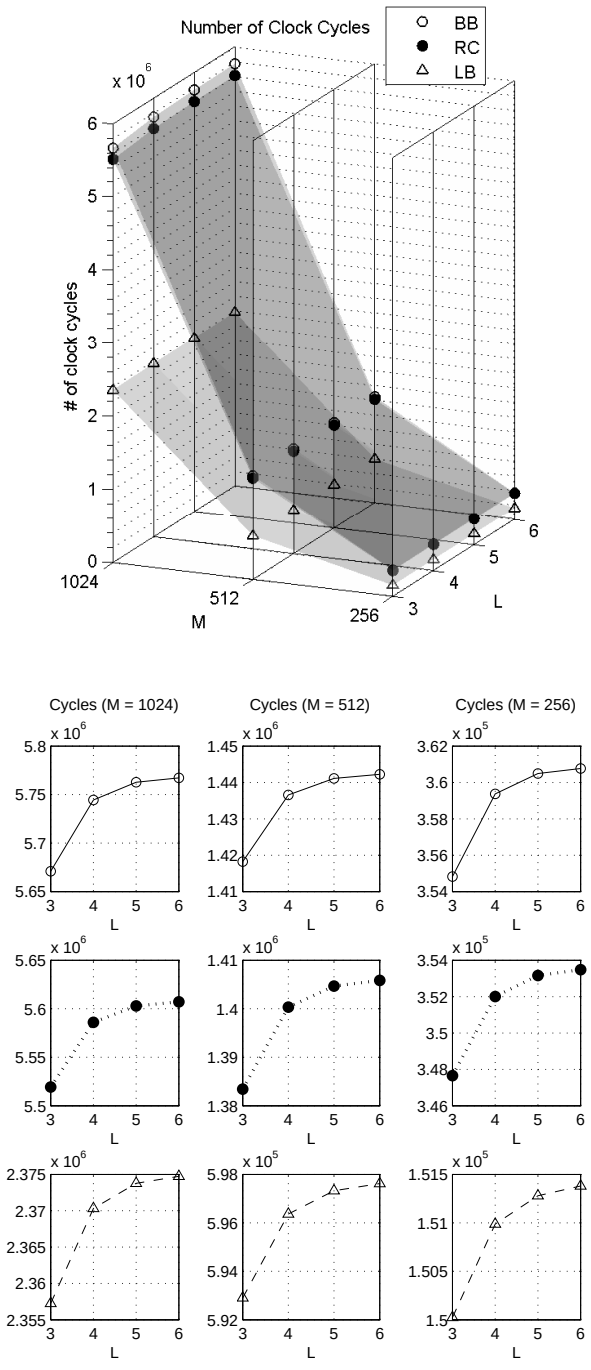


Figure 3.26: Number of cycles. Note that the values are multiplied by 10^6 in the 3-D graph, the 2-D subgraphs of all schedules for M=1024, and the 2-D subgraph of BB and RC for M=512. A different scaling is used in the 2-D subgraph of LB for M=512 and the 2-D subgraphs of all schedules for M=256: the values are multiplied by 10^5.

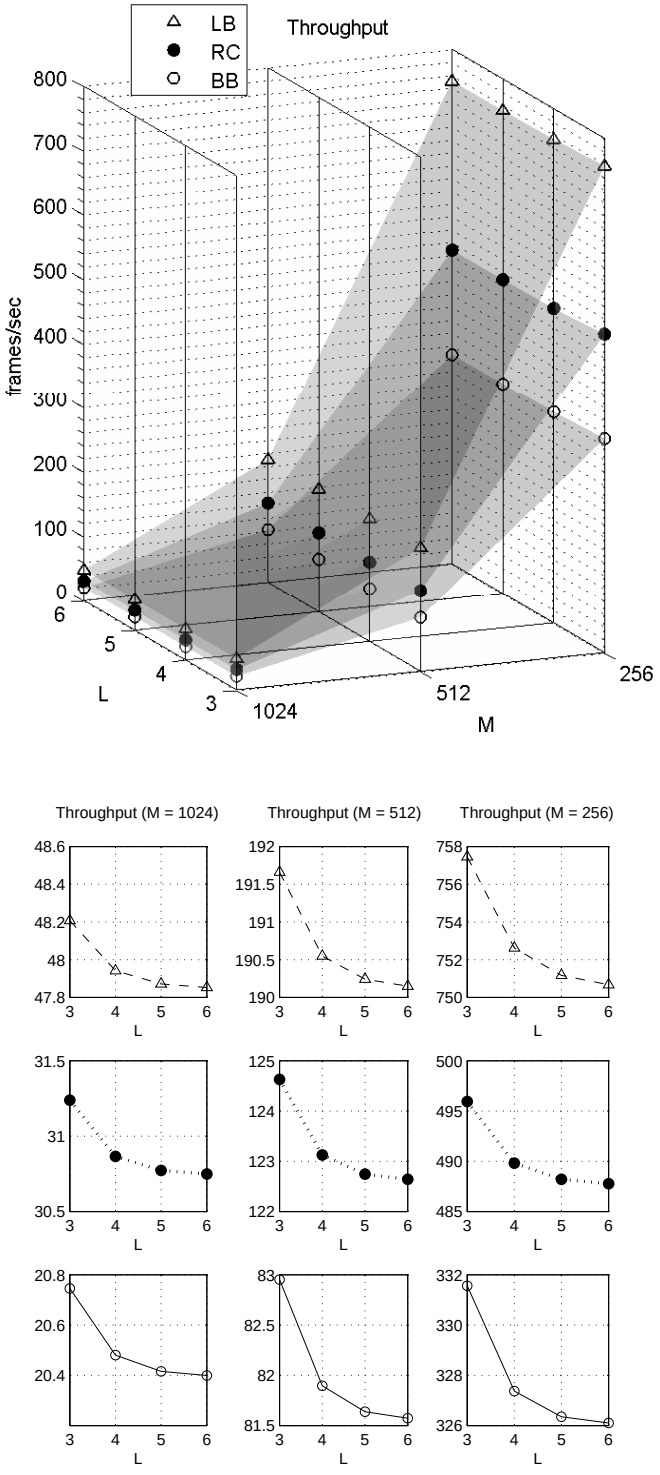


Figure 3.27: Throughput results (in frames/sec).

inputs enter the filter-bank in a serial manner and no parallelism is involved in the filtering operations. On the contrary, in LB and BB, two or three inputs might enter the filter-bank in parallel. Also, in RC the filter's output pair cannot be written in image memory in a single cycle; one of the outputs should be buffered to be written at the next cycle. The results prove that using multi-port buffers, even with a single filter-bank, halves the number of cycles for the LB architecture. But this is not the case for the BB, even if multi-port buffers are *also* used to increase memory-access locality. This is due to the streaming nature of the operations taking part in LB, which is no longer the case for BB. Due to that, in LB at many points the next action is predetermined, for example the horizontal filtering is continuous and successive columns are successively filtered. As a result, many low level actions can occur in parallel, as it is pre-decided that they wouldn't affect each other. Things are different for BB, since the control is deprived of such a streaming behavior and many options should be considered at a specific point, instead of following a predetermined route. As a result, in the case of BB, the number of cycles is increased, compared to LB.

Observing Fig. 3.27, one concludes that all three architectures can handle efficiently the image processing of still images, with minimal hardware cost - just one filter-bank with only one pipeline stage for the whole structure.

In order for a system to carry out video processing, the whole video-processing chain should operate at a speed of 30 frames/second. Therefore, a safety margin should be considered, to judge the video-processing capability of

a system based on the results of Fig. 3.27. Observing Fig. 3.27, one can safely conclude that all three architectures can handle the video-processing of image sizes 256 and 512. The high throughput for such images would also allow an efficient operation of all three systems in a low-power mode by reducing the clock frequency. However, when it comes to image size 1024, even though the throughput of the LB architecture still allows video-processing, this is no longer the case for the other two.

3.8.2 FPGA slices

The FPGA slices used in RC are much fewer than in LB and BB (Fig. 3.28). This is due to the simplicity of the control associated with the RC algorithm. The number of slices for RC covers a range from 280 (M=256, L=3) up to 329 slices (M=1024, L=6). For LB and BB this range is 2659-3001 and 2646-3597 slices, respectively.

3.8.3 Memory issues

The RC does not involve any on-chip buffers, contrary to LB and BB. Thus, in RC the image memory is the only source of inputs for the filter-bank. As a result, the number of image memory accesses is significantly larger in the RC case (Fig. 3.29). In the cases of LB and BB this number is the same, and does not vary when the number of levels varies, as it is, in both cases, equal to $2\times M^2$.

In LB and BB, the on-chip local memory of each level is accommodated in

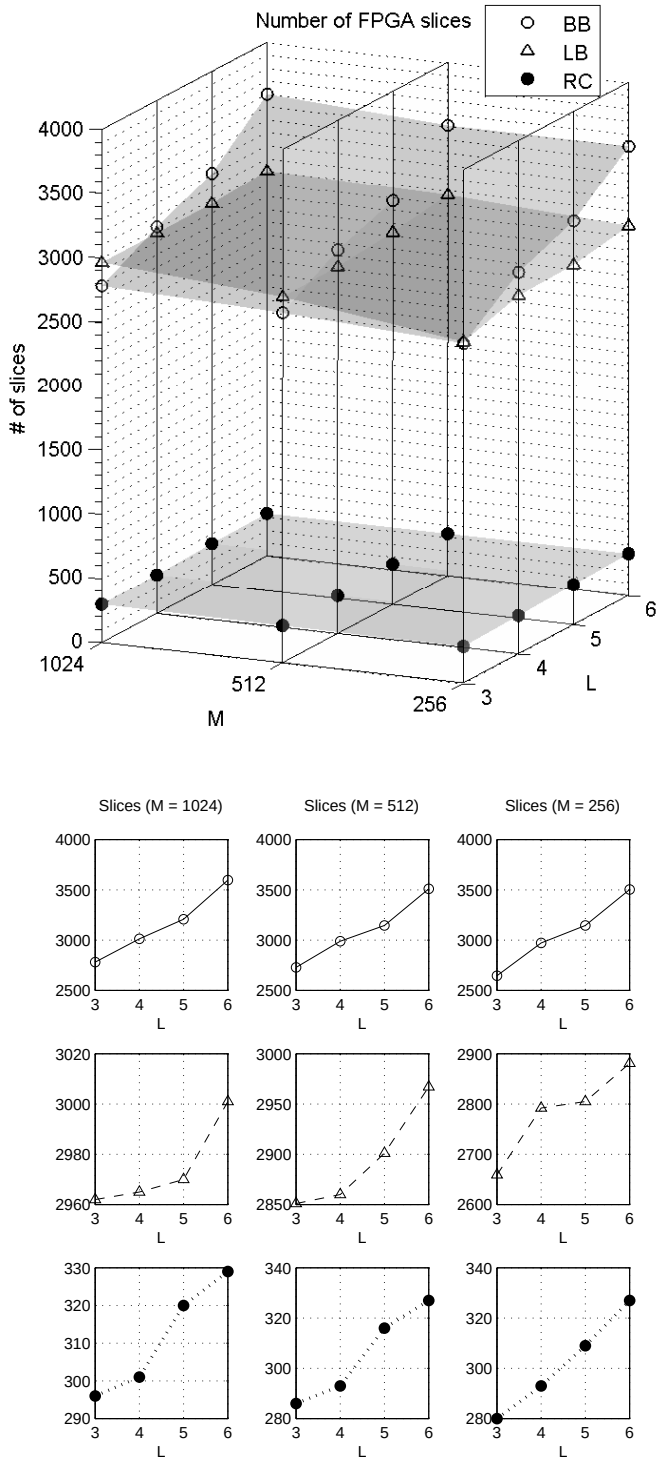


Figure 3.28: Number of FPGA slices.

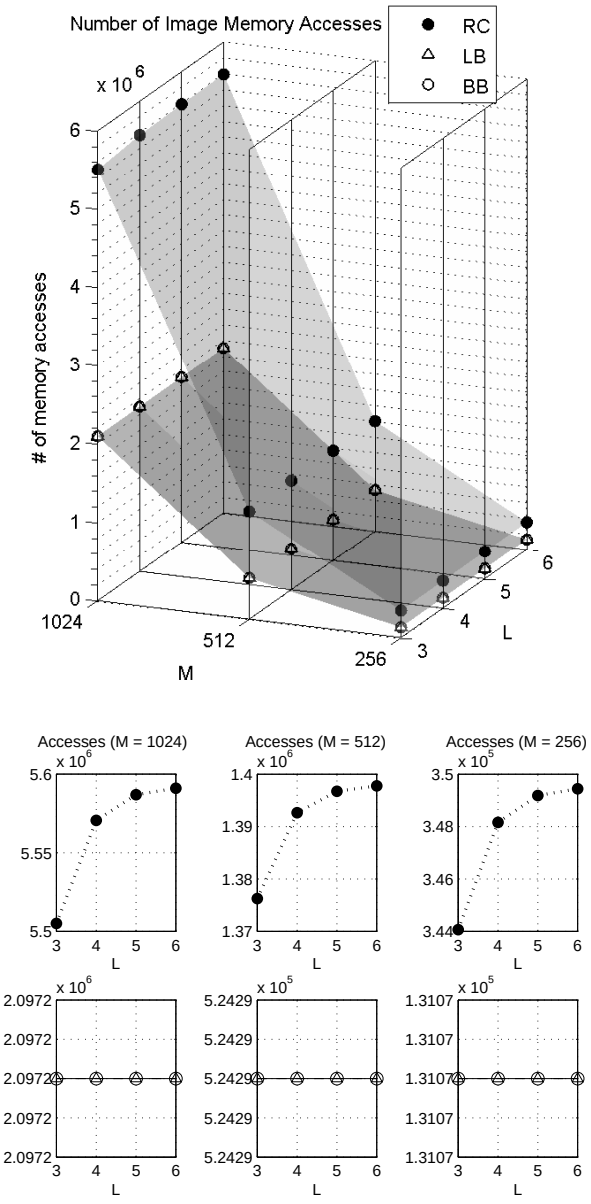


Figure 3.29: Total number of accesses (contains both read and write accesses) of the image memory. For LB and BB this number is the same, and does not vary as L varies, since it is, in both cases, equal to $2\times M^2$. The values of the vertical axis are multiplied by 10^6 in the 3-D graph, in the 2-D subgraphs of all schedules for M=1024, and in the 2-D subgraph of RC for M=512. A different scaling is used for the vertical axes in the 2-D subgraph of LB/BB for M=512 and the 2-D subgraphs of all schedules for M=256: the values are multiplied by 10^5.

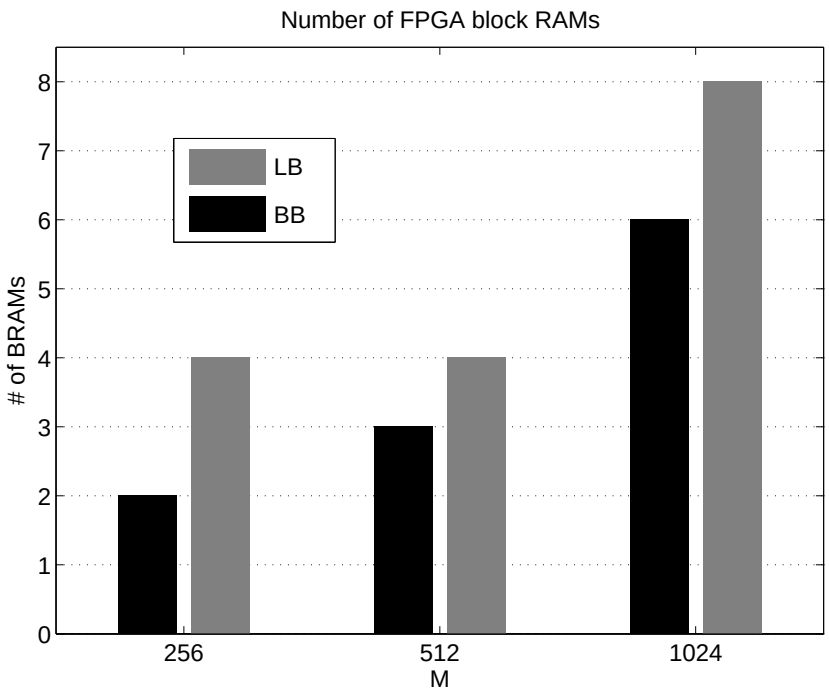


Figure 3.30: Number of BRAMs.

BRAMs (Fig. 3.30), generated by the Xilinx ISE Coregenerator, and registers. The bit-width used is 16 bits. The Virtex-4 BRAMs used are 18 K dual-port BRAMs.

To obtain maximum throughput, the buffers of the same type and of different levels can be grouped together in a single BRAM, if the space provided is enough. This way, during the vertical filtering of successive columns, R(j), C(j), buf1(j) and buf2(j) can feed the filter-bank simultaneously. At the same time, locations of buf1(j) and buf2(j), that have already been read, are overwritten with the new intermediate results, to be read at the next lifting step. Moreover, R(j) can be read while R(j+1) is written with the output of a previous column's vertical filtering. Therefore, four dual-port BRAMs should be

used in the cases of image sizes 256 and 512. Four additional dual-port BRAMs will be needed for size 1024, to accommodate the larger buffers of level 0. The number of BRAMs remains the same for 3, 4, 5 and 6 levels, to guarantee high throughput and enough space for the larger buffers of the lower levels.

In BB, buffers IPM and bufH of each level are implemented as two-word registers. Contrary to LB, the vertical filtering of successive columns is no longer successive. Thus, the constraints that guarantee maximum throughput are not so strict for BB. During vertical filtering, three filter inputs should be read simultaneously, thus, at least two dual-port BRAMs should be used. To provide enough memory space, 2, 3 and 6 BRAMs should be used for image sizes 256, 512 and 1024, respectively. These choices, which also respect the minimum of two BRAMs, remain the same for 3, 4, 5 and 6 levels, since the larger buffers of the lower levels are the ones that determine the memory space needed.

3.8.4 Energy and Power Consumption

In this section, energy measurements for the implementations are presented. First, on-chip energy results are provided, with the off-chip image memory being excluded. Afterwards, assuming an off-chip image memory and considering off-chip power estimates, the total energy, required for the completion of the transform, is calculated.

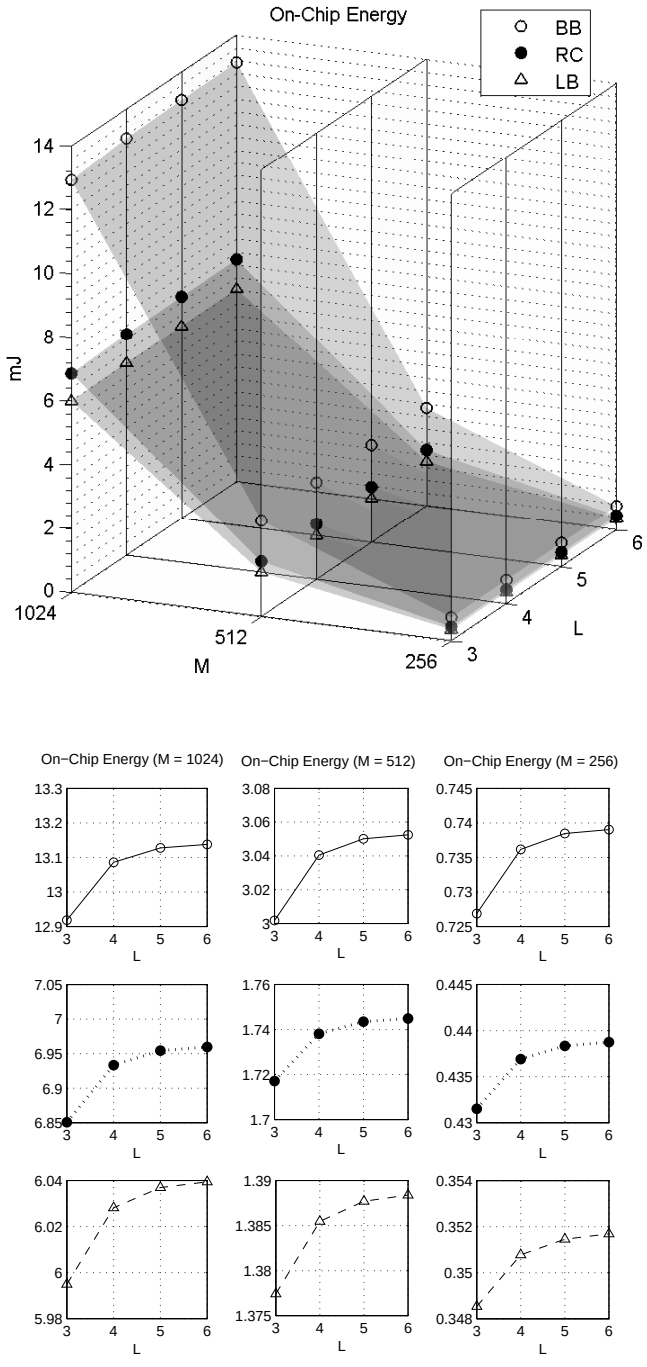


Figure 3.31: On-chip energy consumption (in mJ).

On-chip energy

To estimate the on-chip energy and power consumption of each schedule, the XPower tool, offered by Xilinx ISE v.8.1, has been used. The estimated on-chip power dissipation of the RC is 214 mW, which does not notably alter as M or L vary. This is due to two factors: (a) the RC doesn't involve on-chip buffers, whose number would depend on the given parameters M and L, and (b) the variation in the number of slices for different pairs of these parameters is very small (Fig. 3.28). This is no longer the case for the other two, where BRAMs are used, their number being directly dependent on M (Fig. 3.30), and where the number of slices varies to a larger extent, relative to the given parameters (Fig. 3.28). Thus, in the case of BB, the power consumption starts at 241 mW (M=256) and stretches to 268 mW (M=1024). The power consumption of LB is at relatively higher levels, starting at 264 mW (M=256) and reaching 289 mW (M=1024). The smaller number of slices the LB uses, compared to BB, is overshadowed by the requirement for more BRAMs, and, therefore, the LB ends up with a higher power consumption.

The on-chip energy consumed for the execution of the transform, relative to M and L, is presented in Fig. 3.31. At a first glance, one notices the considerably larger amount of energy related to the BB, compared to the other two. This is due to the large number of cycles associated with the BB (Fig. 3.26). The number of cycles is once more the dominant factor in the energy calculations for the LB. That is, the significantly smaller number of cycles related

to LB (Fig. 3.26) results in lower energy than RC, even though the RC is the less power-hungry of all.

Total energy

The power and energy results, which were presented in the previous section, were calculated without considering the power associated with the image memory accesses. This power depends on whether the memory is on-chip or off-chip. Using an off-chip image memory is normally the only option, especially when working with large images or/and not very large FPGA devices. In that case, the power cost per memory access is significantly high, and depends on technology-related characteristics of the given memory.

A Synchronous DRAM (SDRAM) will be considered as the off-chip image memory. Note here that the energy consumed at the off-chip interconnect is not included in the calculations of total energy. The energy of the off-chip interconnect would normally be proportional to the distance between the SDRAM and the FPGA device [ZAS+01].

The SDRAM system power calculator available from Micron Technologies [Mic08], has been used, to calculate the off-chip power associated with a Micron 64 Mb x16 SDRAM. This tool bases the power calculation on a combination of SDRAM device specifications and usage conditions in the system environment. It, therefore, considers reliable estimates of the static and dynamic power, to calculate the total power dissipation of the memory.

The choice of SDRAM with respect to its clock frequency is based on

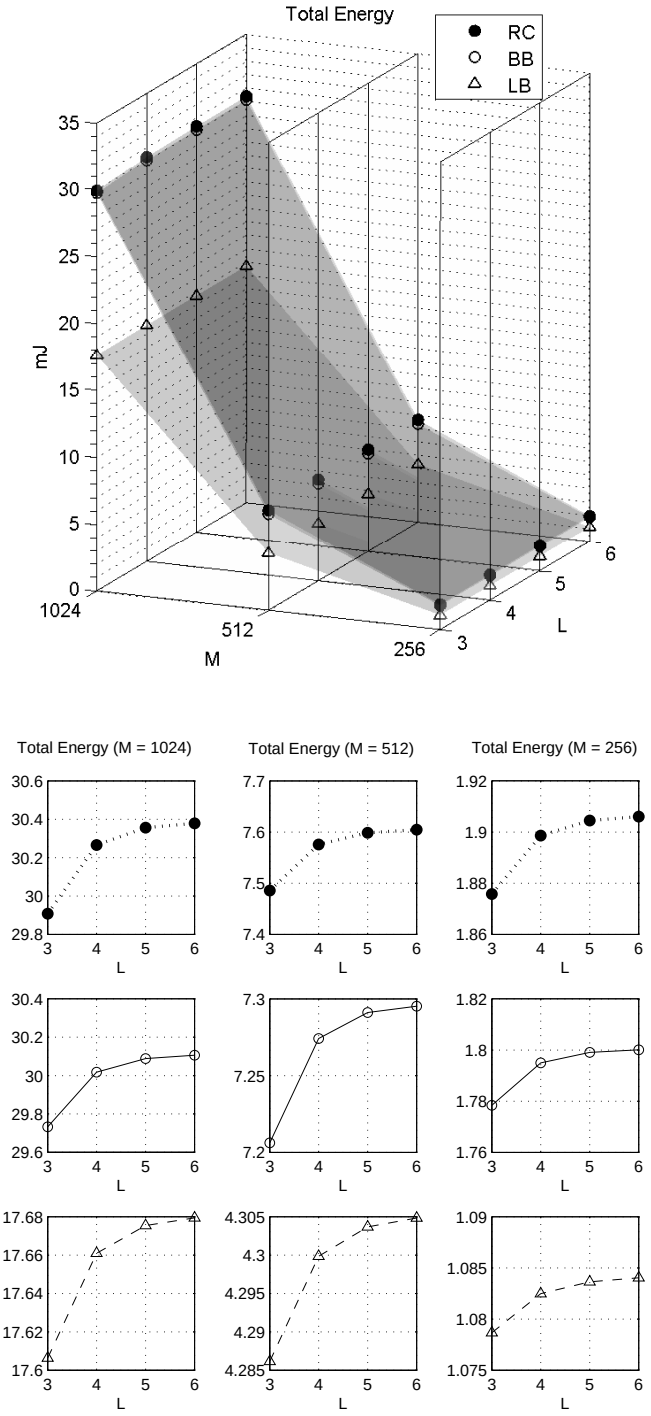


Figure 3.32: Total energy consumption (in mJ). An off-chip SDRAM is considered. The total energy required for BB is very close to that of RC (for RC it is slightly higher).

the following two assumptions. Firstly, it has been assumed that the image memory will operate at the same frequency as the rest of the system. As have been mentioned, this is done in order to make the comparison as generic as possible, since the image memory is sometimes on-chip. Secondly, the entire system operates at the frequency dictated by the on-chip design. Therefore, an available SDRAM with clock frequency that is greater or equal to the frequency of the on-chip system can be used.

As has been stated in Section 3.8.1, the RC, the LB and the BB have frequencies of 172.4, 113.6 and 117.6 MHz respectively. Therefore, an SDRAM of clock frequency 183 MHz (speed grade = -55) has been considered for the case of RC, and 133 MHz (speed grade = -75) for LB and BB.

According to the estimates of the system power calculator, the off-chip power varies from 715.7 mW to 720.3 mW for the case of RC, as the percentage of memory write/read cycles vary, for different values of L and M. Note that the variation is not as large, for different values of M, as one would expect observing Fig. 3.29. This is because, as M increases, the total number of cycles increases as well (Fig. 3.26), so the above percentages are maintained at similar levels. Due to the high cost per memory access that characterizes the off-chip case, the large number of accesses of the RC (Fig. 3.29), renders this schedule extremely demanding in terms of power.

Compared to the RC, the off-chip power of the LB and the BB is expected to be at lower levels, due to the lower number of memory accesses (Fig. 3.29). The off-chip power estimates vary between 549.7 mW and 559.7 mW for LB,

and between 346 mW and 348.8 mW for BB. The lower off-chip power associated with the BB, compared to LB, is due to the higher percentage of memory write/read cycles over the total execution cycles. This is due to the fact that the number of clock cycles is much larger for BB (Fig. 3.26), while both the BB and the LB need M^2 read memory accesses and M^2 write memory accesses. Therefore the distribution of memory accesses is rather dense in LB, compared to BB. However, due to the significantly smaller number of clock cycles, required for its completion, the LB maintains the smallest energy when it comes to total consumption, as Fig. 3.32 shows.

The large number of memory accesses of the BB will keep the total energy consumption of BB at high levels, as it was the case with the on-chip energy. As a result, the total energy required for the BB to be executed is very close to that of the RC (Fig. 3.32), even though the power is significantly lower in BB.

Using an on-chip image memory would be in favor of the RC, since the power associated with every memory access will be much less. In this case, the BB will most certainly consume a lot more total energy than RC, as was the trend in Fig. 3.31. The LB will still maintain the lowest total energy.

3.9 Appropriate Optimizations

The performance of the three schedules has been investigated under the assumption of using a single-port image memory and a single filter-bank which

is shared among all levels and stages of the transform. This section briefly discusses which optimizations would be appropriate for each schedule to further improve its performance. Specifically, the objectives would be to increase (a) parallelism in data processing and (b) locality of data accesses.

As has been discussed, in RC the lack of parallelism is due to the fact that the single-port image memory is the one and only source of the filter-bank's inputs. Therefore, using a multi-port image memory would multiply the performance by a factor equal to the number of memory ports. On the contrary, just increasing the number of filter-banks while sticking with a single-port memory would not help, since there would not be enough on-chip information to process. As far as the locality of data accesses in RC is concerned, it could be improved by incorporating in the traditional bufferless architecture of Fig. 3.13 a single on-chip line-buffer. The inputs of the filter-bank would be read from this line-buffer in all levels. Therefore, its size should be equal to the largest dimension of the original image, to accommodate the large number of coefficients of the first level.

In the cases of LB and BB, the locality of data accesses is already maximized, since image memory accesses are restricted to reading the original image and writing the final outputs. As far as the parallelism is concerned, since so much information exists in the on-chip buffers, the more filter-banks available to process it, the better. Thus, in this case increasing the number of filter-banks would vastly increase the parallelism in data processing, even if the image memory remains single-port.

3.10 Summary

This chapter has addressed the critical task of constructing the wavelet pyramid, which is the basis of the majority of multiresolution motion estimation techniques. In particular, the three major 5/3 lifting 2-D DWT computation schedules have been implemented on on the Virtex-4 FPGA family. The implemented architectures have been compared in terms of throughput, area, memory and energy requirements. In summary, the conclusions of the work presented in this chapter are the following.

The RC has by far the lowest hardware cost. Not only does it involve no on-chip buffering, but also the FPGA slices used are significantly fewer than in the cases of the other two. Due to its simplicity, it enjoys the highest frequency. However, the insufficient level of parallelism, associated with RC, increases the number of cycles required for the schedule to be completed. As a result, when it comes to throughput, it is outperformed by LB (but not by BB). Moreover, the reduced memory access locality, offered by RC, increases the number of memory accesses and the total energy to the highest level.

The LB requires the lowest number of cycles among the three schedules, thanks to, not only the parallelism achieved by using multi-port on-chip buffers, but also its streaming behavior. Due to the small number of cycles, this schedule also enjoys the highest throughput and the lowest on-chip and total energy consumption. Due to the use of on-chip buffers, the LB increases the memory-access locality, compared to RC, minimizing the number of image memory

accesses.

The BB increases the memory-access locality, compared to RC, and achieves the same number of memory accesses as LB. To do this, the BB uses a smaller number of BRAMs than LB. However, it consumes more FPGA slices than LB for most sets of parameters. The BB is associated with the highest complexity, due to the lack of streaming behavior in the filtering operations. As a result, a significantly larger number of clock cycles is needed for the BB schedule to be completed. The large number of cycles results in high energy requirements: it is by far the most expensive when it comes to on-chip energy, and it is almost as expensive as RC when the total energy is considered, even though the memory accesses of BB are considerably fewer. The BB has the lowest throughput, due to control complexity, as well as to the frequency in which it operates, which is not as high as in the RC case.

In conclusion, all three architectures that have been proposed in this chapter achieve real-time performance for the execution of the 2-D DWT. The evaluation that has been presented gives insight on which computation schedule is the most suitable for the specific requirements of the given application.

The next chapter begins the discussion of the proposed video enhancement techniques, which is the main area of interest of this thesis. In particular, in the next chapter, possible configurations of the adaptive sensor will be investigated and combined with the appropriate processing, so as to increase the spatio-temporal resolution of the output frames.

Chapter 4

Video Enhancement by Exploring the Configuration Space of an Adaptive Image Sensor

This chapter explores how the elementary pixels of an adaptive image sensor [Fov08, CDT06] can be configured, so as to maximize the raw information collected from the environment. For each configuration, appropriate methods are proposed to further process that information and reconstruct a final output of both high temporal and high spatial resolution. In particular, an adaptive image sensor is used for motion deblurring. Motion deblurring is locally performed by configuring the motion regions to larger pixel sizes that produce high frame-rate samples. Two types of configuration schemes and the

appropriate processing methods for the reconstruction of the final output are proposed. These are namely a deconvolution-based and a SR-based approach.

In summary, the contributions of this chapter are the following:

1. A super resolution (SR) system is proposed for the real-time spatio-temporal enhancement of video. This SR-based approach employs a uniform grid of large pixels on the motion areas. The motion magnitude, and corresponding motion blur, determines the pixel size that gives the appropriate spatio-temporal trade-off. The effect of motion magnitude on that pixel size is thus explored.

2. A deconvolution-based system is proposed for the real-time spatio-temporal enhancement of video. Contrary to the SR-based approach, where each motion region is configured to a single, uniform pixel size, the deconvolution-based approach employs spatially multiplexed pixels of different sizes on the motion regions.

3. A detailed comparison of the proposed SR-based and deconvolution-based video enhancement systems is presented, using various motion blur and static scenarios.

The rest of the chapter is organized as follows. Section 4.1 illustrates the main concept of reducing motion blur by utilizing the reconfigurability of an adaptive image sensor. Different sensor configuration schemes are explored, which are employed by the two video enhancement approaches that are proposed. These approaches, namely the SR-based and the deconvolution-based

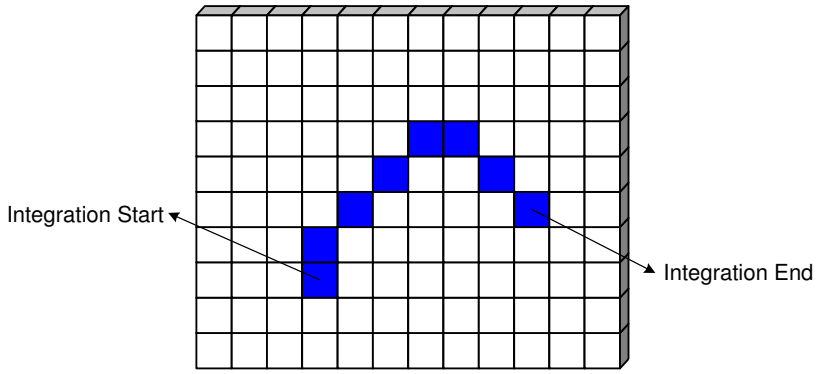
approach, are described in Sections 4.2 and 4.3, respectively. A comparative discussion of the two schemes is carried out in Section 4.4, while Section 4.5 presents reconstruction results achieved by the two approaches. In Section 4.6, the extension of the proposed methods to non-rigid objects is briefly discussed. Finally, the conclusions of the evaluation process are presented in Section 4.7.

A part of the work that is presented in this chapter has been published in [ABC08].

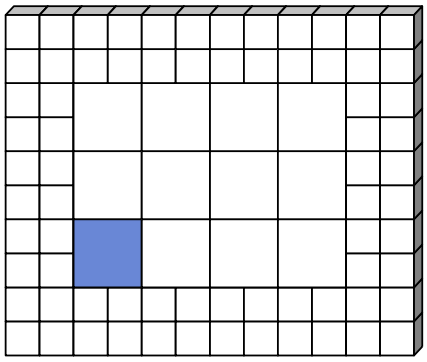
4.1 Employing Larger Pixels to Reduce Motion Blur

Before investigating how different configuration schemes may increase the spatio-temporal information that is captured, it is important to clarify how the reconfiguration property of the adaptive sensor can be utilized to reduce motion blur. This concept is visualized in Fig. 4.1, which demonstrates the blur resulting from the motion of a hypothetical one-pixel moving object. In Fig. 4.1, two different configurations are thus considered to compare the outputs of the sensor that would be produced for each scenario during the HR integration interval. This interval corresponds to the integration time required by the elementary pixels of the sensor.

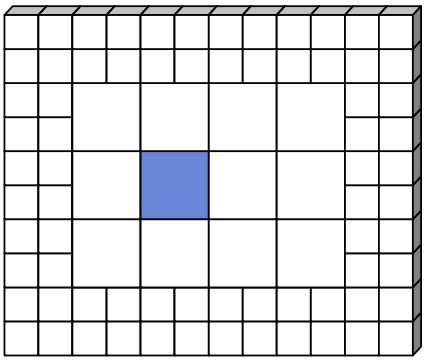
The first configuration employs elementary pixels on the entire sensor, thus a traditional, time-invariant pixel grid. In that case, a single sample of the motion region is produced during the HR integration interval which is significantly



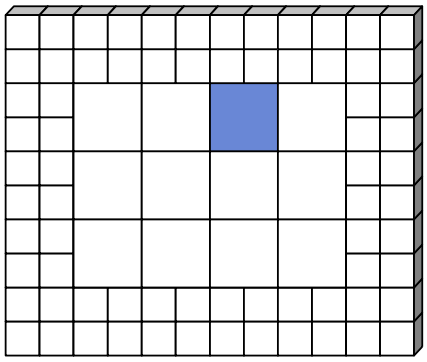
(a) Configuration I: Motion-blurred output for HR.



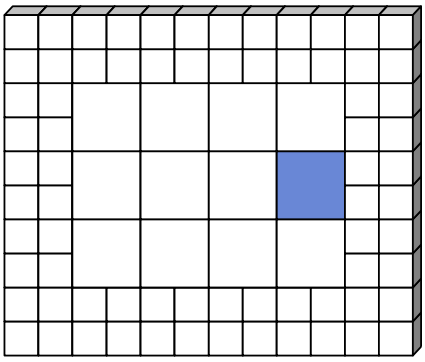
(b) Configuration II: First LR sample.



(c) Configuration II: Second LR sample.



(d) Configuration II: Third LR sample.



(e) Configuration II: Fourth LR sample.

Figure 4.1: The outputs of two different sensor configurations. (a) The HR configuration renders a single, motion-blurred output during the HR integration. (b,c,d,e) Configuring the motion area to 2×2 HR produces 4 time samples during the HR integration, each containing a fragment of the trajectory of (a).

blurred. This is illustrated in Fig. 4.1(a).

If the 2×2 HR configuration was employed on the motion area instead, then 4 LR frames would have been produced for that area during that integration interval. These are illustrated in Figures 4.1(b), 4.1(c), 4.1(d), and 4.1(e). Each one of these samples contains a short part of the trajectory of Fig. 4.1(a) and thus reduced motion blur. Compared to the output of the first configuration, these four frames have four times higher temporal resolution, but also four times lower spatial resolution. One should therefore employ methods to enhance the spatial resolution of the high-frame-rate samples, in order to produce an output of both high spatial and high temporal resolution. The remainder of the chapter explores ways of configuration and processing that can be employed to address this issue.

It should be noted that real-time video enhancement is targeted. The real-time performance constraints will be extensively evaluated in the chapters that follow.

4.2 SR-Based Motion Deblurring

This section presents the first video enhancement scheme that is proposed. The proposed method utilizes SR techniques for the reconstruction of the final output on the HR grid. 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 at all are configured to HR pixels.

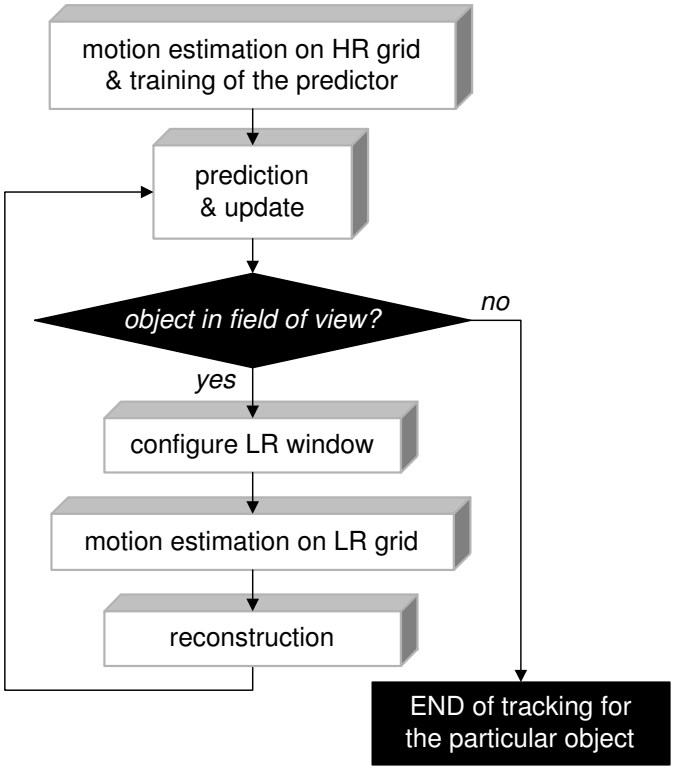
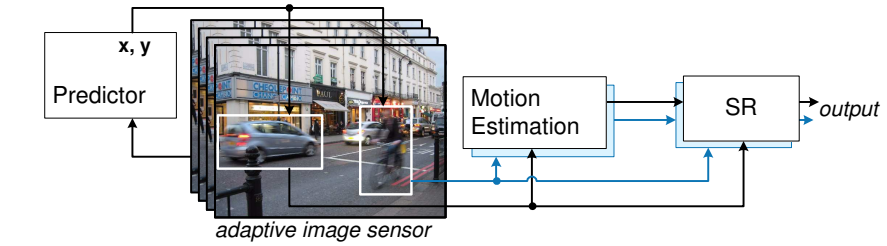


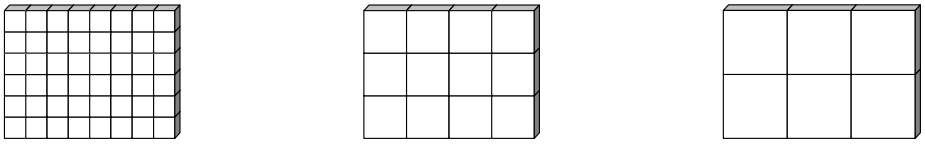
Figure 4.2: Flowchart describing the operation of the SR-based system for each independent moving object.

Ideally, the adaptive sensor would be reconfigured at every new HR integration. In reality, the reconfiguration frequency can be lower, depending on the technology of the given sensor. Thus, the proposed system operates as follows. At every new HR integration interval, newly appearing moving objects are detected. This initial object detection could be done either by employing a particular sensor technology [CHK+07] or by applying a motion estimation algorithm on the HR grid between adjacent HR samples [Bou02]. The second option has been used. Once the detection process is completed, for every individual moving object that is detected, the control of Fig. 4.2 is employed. The *configure LR window* block is executed only in those HR integration intervals that include sensor reconfiguration, whereas the rest of the blocks of the loop are executed in every HR integration interval. A predictor is employed to determine the position of the object in the next HR integration interval. Experiments have shown that the Kalman filter [WB] is appropriate for the given application. When the next sensor reconfiguration occurs, the position predicted by the predictor determines the location of the LR area, *i.e.* the motion region that includes the moving object, which is configured to larger pixels. During each 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 [PPK03, BK02, FREM04, IP91]. Thus, motion-deblurring is locally executed on the dynamic regions of the scene. At every new HR integration interval, the control starts at the second block of Fig. 4.2. The loop ends when

the particular object exits the field of view.



(a)



(b) HR grid (c) LR = 2×2 HR (d) LR = 3×3 HR

Figure 4.3: (a) SR-based motion-deblurring on an adaptive image sensor. (b) HR grid covering the static background. (c,d) Uniform LR pixel grids formed on the motion regions. The LR pixel size increases with the local motion magnitude.

The block diagram of the proposed SR-based system is shown in Fig. 4.3. The *Predictor* block locates areas for LR configuration. The pixel size of these areas is proportional to the motion magnitude, as Figures 4.3(c) and 4.3(d) indicate. The *Motion Estimation* block reads the sequence of LR frames produced at each LR area and returns the motion vectors, *i.e.* the displacements between each LR frame and the reference LR frame. This information is used by the SR unit to enhance the spatial resolution. More information on SR reconstruction is given in the chapters that follow. To identify the pixels of the static background and exclude them from the SR reconstruction process, a dense vector field is considered. Thus, the pixels with zero motion are treated

as background pixels. Error values rendered by *Motion Estimation* [Bou02] are used to weight the contribution of different LR samples. Also, to increase the spatial information available to the SR block, LR samples before and after the integration interval of interest contribute in SR with adjustable weights. The values of these parameters are given in the evaluation section of the chapter.

The spatial resolution and the frame-rate of the output are those of the HR sequence, where the motion blur has been removed in the LR regions. In the special case where the motion blur is due to camera shaking, a single LR area spans the entire sensor.

In the chapters that follow, the SR block will be further investigated; both the SR framework and implementation issues of SR on reconfigurable hardware will be discussed in detail.

4.3 Deconvolution-Based Motion Deblurring

In the deconvolution-based approach the sensor is configured to hybrid regions, containing both elementary and LR pixels. The subset of LR pixels gives high frame-rate samples, which are used in order to recover the motion information. Therefore, a motion PSF is constructed for every moving object. Then a deconvolution is applied separately on each object to produce a deblurred output for the part of the frame occupied by that object.

The flowchart of the deconvolution-based system is shown in Fig. 4.4. The first block of Fig. 4.4 estimates the motion information on the frame. With

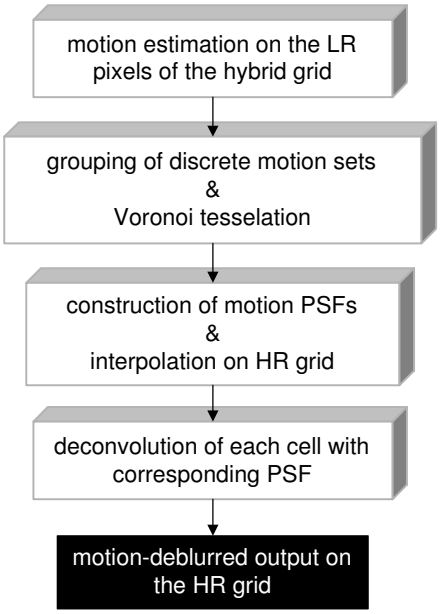


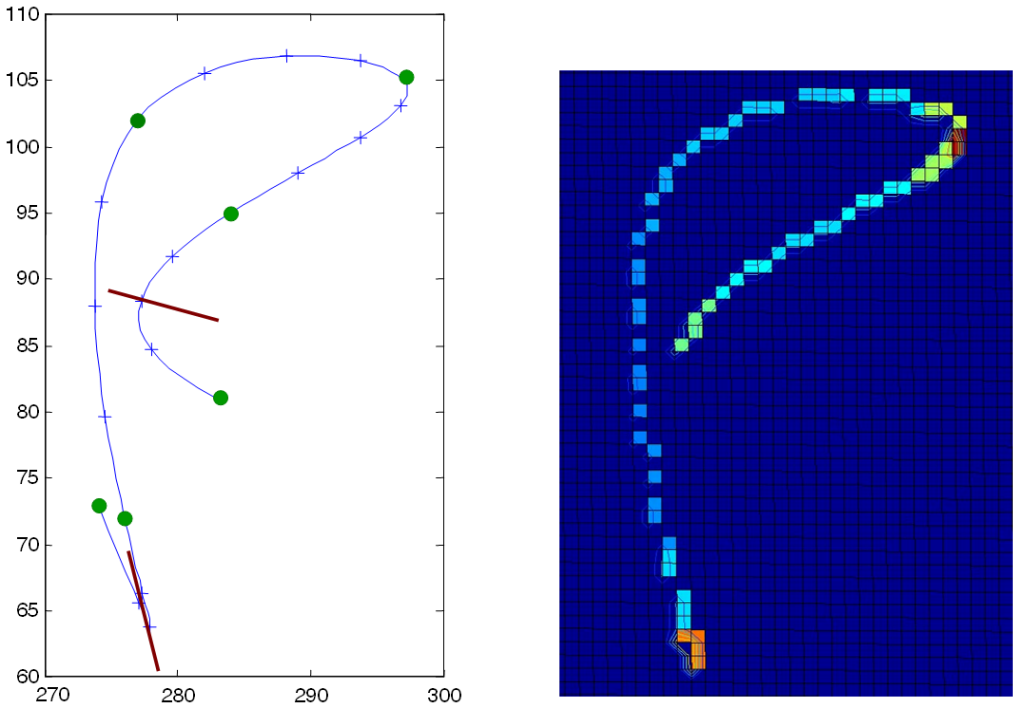
Figure 4.4: Flowchart for the deconvolution-based system.

the processing executed by the second and the third block of Fig. 4.4, each individual moving object is associated with a particular motion PSF. Finally, deconvolution is applied on each moving object to produce a deblurred output for the part of the frame occupied by that object. More information on the dataflow of the deconvolution-based approach is provided in the following paragraphs.

The block diagram of the deconvolution-based system is shown in Fig. 4.6(a). In order to achieve deblurring via deconvolution using a single sensor, a hybrid grid consisting of spatially multiplexed HR and LR pixels is used. Different sizes of LR pixels can be configured as shown in Fig. 4.6(b, c), so that larger LR pixels will be formed in areas with larger blur. The time samples captured by the LR pixels are the inputs to the *Motion Estimation* block. This

block renders a set of 2-D points (one point per LR sample) belonging to the continuous motion trajectories of individual features during the HR integration. We refer to this discrete subset of the actual motion trajectory by *discrete motion set*. Neighboring features with similar discrete motion sets are grouped together and a center of mass is determined on the frame for each group. Voronoi tessellation [OBSC00] is applied on the set of these centers of mass. For each Voronoi cell a motion PSF is constructed on the HR grid by applying spline interpolation on the points of the associated discrete motion set and calculating the "energy" along the PSF as in [BEN04].

An example of the construction of a motion PSF of a feature using spline interpolation is illustrated in Fig. 4.5. The thick dots in Fig. 4.5(a) are the points of the discrete motion set, indicating the position of the particular feature at each one of the LR time samples produced during the HR integration interval. The interpolated PSF corresponds to the part of the curve that is bounded by the two intersecting line segments. In particular, for the case of Fig. 4.5, 4 time samples are produced during the HR integration interval. The 2 time samples that lie on the outside correspond to the adjacent LR samples, which are produced before and after the HR integration interval of interest, and are needed to determine the shape of the entire PSF. In order to preserve the temporal order of the measured points, the motion path is parameterized with respect to time, and thus time increases along the curve, in the direction of the motion. Moreover, depending on how long the particular feature stayed on each pixel of the image sensor during the HR integration interval, the "energy"



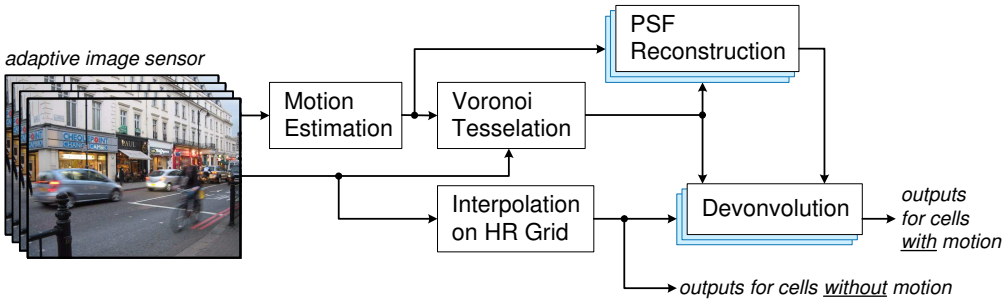
(a) (b)

Figure 4.5: The interpolated motion PSF. (a) Spline interpolation on the discrete motion set of a particular feature. (b) Discretization on the HR grid and PSF weights.

is determined along the PSF. In Fig. 4.5(b), the PSF is discretized on the HR grid, and the different colors indicate different PSF weights, determined by the calculated "energy" along the PSF.

Once the PSFs are formed, deconvolution is applied separately on the HR grid of each cell. The input of each deconvolution block consists of the raw HR pixels of the hybrid grid and the interpolated HR information formed at the LR pixels of the grid. The interpolated HR pixels are computed by solving a linear system formed by weighting the surrounding raw HR pixels and the underlying

LR information. For the cells where no motion exists, the reconstructed output is the output of this interpolation. The final reconstructed frame consists of the individual outputs produced for every cell.



(a)

(b) LR = 2×2 HR (c) LR = 3×3 HR

Figure 4.6: (a) Deconvolution-based motion-deblurring on an adaptive image sensor. (b, c) Multiplexed LR and HR pixels of a hybrid grid.

To demonstrate the control flow in the deconvolution-based approach, let us consider a simple visual example, where the entire sensor is covered with the 2×2 hybrid grid of Fig. 4.6(b). The image of Fig. 4.7(a) is the input of the deconvolution-based processing block. This comprises the raw HR pixels of the hybrid grid and the interpolated HR pixels on the regions of the sensor that are configured to LR pixels. Motion estimation is applied on the LR samples of the hybrid grid, and the discrete motion sets are identified. Voronoi tessellation is then applied on the entire grid to isolate the areas corresponding to different

motion sets. Fig. 4.7(b) demonstrates the resulting Voronoi cells, where each cell is associated with a discrete motion set. Finally, deconvolution is applied individually on each Voronoi cell. Deconvolution is implemented with the Lucy-Richardson [Luc74] method. The final output is reconstructed on the HR grid, as shown in Fig. 4.7(c).

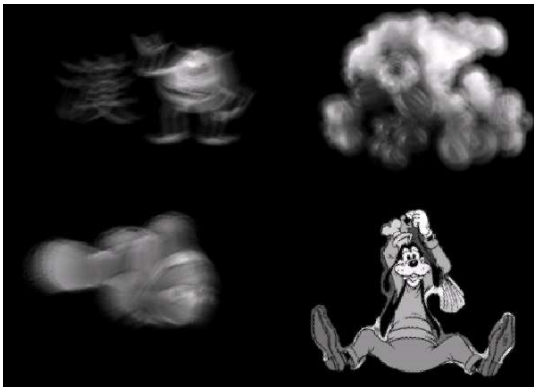
As different degrees of blur may exist on the same frame, at every configuration of the sensor, LR areas of certain pixel size may be formed on independent motion areas, as in the SR approach. Then, at every HR integration, each LR area is tesselated, and different subregions are related to different PSFs.

4.4 Comparing the Two Approaches with Respect to Sensor Specifications

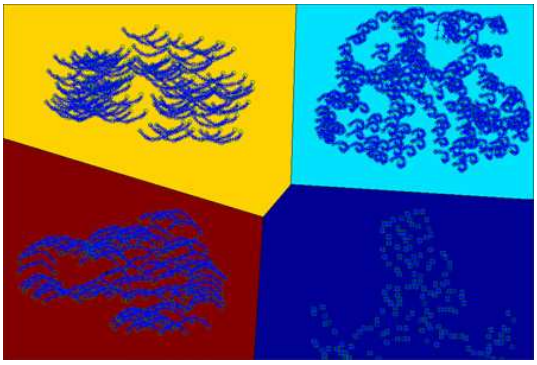
Depending on the technical specifications of the sensor itself, we distinguish two cases:

- *Sensor specifications (a):* The sensor can be configured in every HR integration.
- *Sensor specifications (b):* A sparser configuration can only be performed, *i.e.* every N HR frames.

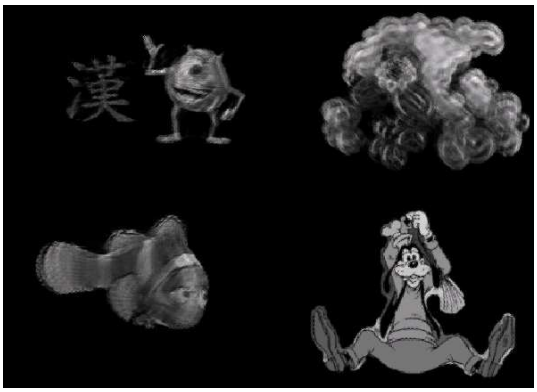
The difference between the above cases, sensor specifications (a) and (b), lies in the accuracy in locating motion regions that will be configured to larger pixel sizes to form LR areas.



(a)



(b)



(c)

Figure 4.7: (a) The input of deconvolution. The LR holes are reconstructed on the HR grid based mainly on the surrounding HR pixels. (b) Voronoi tessellation. Each Voronoi cell is associated with a discrete motion set. (c) Deconvolution is applied individually on each Voronoi cell, to reconstruct the final output on the HR grid.

Ideally, all pixels of the scene belonging to the static background would be configured on the sensor with HR pixels and would, therefore, *not* belong to any LR area. Due to the need of high accuracy in locating motion regions, the case of sensor specifications (a) is closer to that ideal situation than the case of sensor specifications (b). Therefore, for sensor specification (a), LR areas include only few background pixels and mainly consist of dynamic parts.

For the reconstruction of dynamic parts, the SR-based approach is the most effective of the two proposed approaches, due to the following reasons:

1. In the SR-based approach, the LR samples at the input of the SR block experience reduced motion blur. Therefore, the blending, between the moving objects and the static background, is reduced. An extensive discussion of the problem of the blending between objects and background is presented in [BEN04]. The importance of this problem lies in the fact that it complicates background extraction. The blending problem increases for larger motion magnitudes, and is clearly visible in Fig. 4.8(b). In the deconvolution-based approach, the input of the deconvolution block comprises an HR frame of low temporal resolution, where that blending is large for large motions. This complicates background extraction compared to the SR-based approach [BEN04].

2. As blur increases, the size of LR pixels increases to produce an output with sufficiently reduced blur. Larger LR pixels require more LR samples for SR reconstruction [PPK03, BK02, FREM04, IP91]. These

can be obtained, since the number of produced samples increases with the pixel size, due to the space-time trade-off, as explained in Chapter 2. Therefore, in this case the SR-based approach is effective. On the other hand, in the deconvolution-based approach, the quality of the high spatial-resolution information, which is interpolated in the regions configured to LR pixels, degrades as the LR pixel gets larger. This interpolated information is part of the input of deconvolution (Section 4.3), thus affecting the quality of the system's output, as will be demonstrated in the evaluation section. Thus, the SR-based approach is more effective for extensive blur.

Ideally, all static regions of the sensor would be covered with HR pixels. However, in the case of sensor specification (b), LR areas are not formed as often on motion regions, as the sensor is configured sparsely. Thus, motion regions are not identified as often on the frame, and thus larger LR areas should be formed to make sure that the moving objects remain within those areas until the next configuration of the sensor takes place. Inevitably, large motion regions not only include the moving objects, but also contain significant parts of the static background. In the worst case scenario, for very sparse configuration, a single LR area spans the entire frame. Therefore, when sensor specification (b) is employed, the situation may diverge significantly from the ideal scenario that would employ only HR pixels on the static regions of the scene. In such cases, it is critical for the configuration of LR areas to allow high-quality re-

(a) *junction*

(b) *ice-skating*

Figure 4.8: Motion blur decreases the temporal resolution of frames.

construction of both moving and static parts. If the SR-based approach is applied, all the LR channels will contain exactly the same information for the parts that are covered with static pixels. Therefore, since uniform LR areas are employed in the SR-based approach, the quality of the reconstructed static regions cannot surpass that of interpolation applied on the uniform LR grid of a reference frame. On the other hand, if the deconvolution-based approach is used, a hybrid grid would be employed, where 50% of the sensor is configured to HR pixels. Therefore, 50% of the grid gives the actual ground-truth pixel

values. In this case, interpolation is limited on the LR pixels of the grid and is optimized using the raw HR information of the surrounding HR pixels. As configuration gets sparser, LR areas contain more background pixels, and the hybrid grid of the deconvolution approach is more suitable.

To conclude, the SR approach is the most suitable for the reconstruction of the dynamic parts of the scene and prevails when, due to frequent configuration, the LR areas mainly consist of non-static parts. The hybrid grid of the deconvolution method allows effective reconstruction of static parts and is suitable for sparse configuration, where LR areas also include large static parts.

4.5 Performance Evaluation

In the performance evaluation that follows, the Iterative Back Projection (IBP) approach [IP91] is employed for SR reconstruction. This SR reconstruction method will be extensively discussed in Chapter 5, where its choice is justified with respect to the strict timing specifications of the real-time video enhancement application. The IBP is an iterative SR approach, which gradually refines the reconstructed output, as the iterations of the algorithm succeed one another. Thus, during the execution of IBP, the reconstruction error is iteratively minimized and soon converges to a minimum value. The number of iterations that can be executed depends on the throughput specifications, which derive from the real-time operation requirement. Issues related to the

convergence properties and the number of iterations of the IBP are discussed in detail in Chapter 5. For the purposes of the current chapter, the SR outputs are given for 10 and 30 iterations. Typically, the algorithm converges before 30 iterations are completed.

In the deconvolution-based approach, deconvolution is implemented with the Lucy-Richardson [Luc74] method. In both approaches, *Motion Estimation* is required. In the SR-based approach, the frames at the input of the SR block first need to be registered before the actual SR reconstruction occurs. In the deconvolution-based approach, motion information is required for the construction of the motion PSFs that are associated with the individual moving objects, as discussed in Section 4.3. In the performance evaluation that follows, motion estimation is implemented using the Lucas-Kanade optical flow [Bou02] and the Shi-Tomasi good feature extraction [ST94] algorithms.

The evaluation of the proposed approaches is carried out using semi-synthetic data. In this manner, 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 commercial hand-held digital camera, has been shifted, blurred, and downsampled, to synthetically produce the LR sequences.

In this thesis, the Root Mean Squared Error (RMSE) is employed as the error metric for quantification of the reconstruction quality. The RMSE is the quadratic root of the Mean Squared Error (MSE). The MSE is calculated as follows:

$$MSE = \frac{1}{M \times N} \sum_{i=1,j=1}^{M,N} (I_{i,j} - G_{i,j})^2 \tag{4.1}$$

where $I_{i,j}$ and $G_{i,j}$ denote the pixel with coordinates (i, j) in the reconstructed and the ground-truth frame respectively, and M and N denote the frame dimensions.

The proposed methods are evaluated on two video sequences, the *junction* and *ice-skating* (Fig. 4.8) sequences, which involve multiple moving objects. The HR frame size for both sequences is 480×640. The moving objects are isolated from the background by hand, to exclude errors associated with the background extraction method from the evaluation process. The LR sequences were synthetically produced by applying temporal and spatial blur on a dense sequence S_d, which acts as the real-world scene. The time blur of a sensor with HR pixels spans, in both experiments, 144 frames of S_d (Fig. 4.9(b), 4.12(b)). In the first experiment, the reconstruction quality obtained by the two methods is compared both on dynamic and static parts. In the second experiment, the SR-based approach, which is the most suitable for extensive blur (Section 4.4), is used to deblur a frame containing different motion magnitudes. The performance evaluation section closes with a quantitative comparison of the reconstruction quality of the two approaches. In particular, the two approaches are applied on a variety of video sequences. The statistical results that are produced (*i.e.* average and standard deviation of the reconstruction error) further verify the observations from the two aforementioned experiments

as well as the discussion of Section 4.4.

4.5.1 Comparison of the Two Approaches on Dynamic and Static Regions

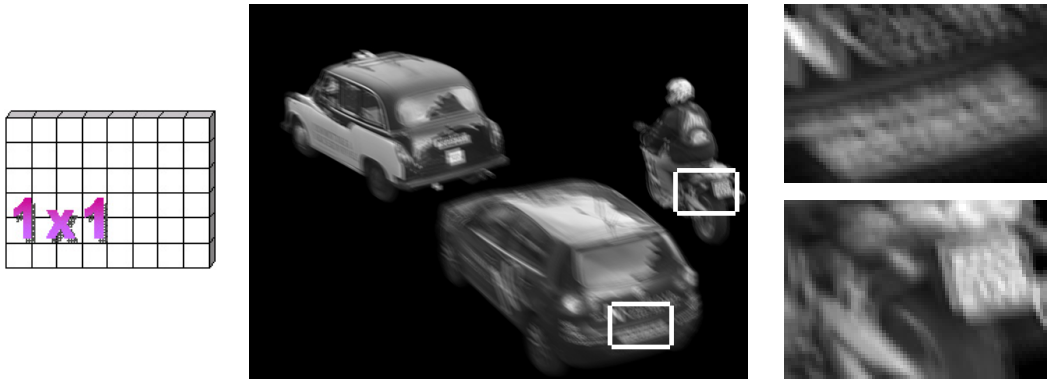
To evaluate the performance of the two approaches, we will first discuss the reconstruction of foreground moving objects and then that of the static background.

The *junction* sequence includes three moving objects with similar degrees of motion blur, as can be observed in Fig. 4.8(a). The LR sequence was generated by applying both temporal and spatial blur on the dense sequence S_d. The temporal blur was simulated by averaging 36 consecutive frames of S_d, whereas a 2×2 Gaussian was employed to simulate the spatial blur of the sensor's pixels. It should be noted that for size 2×2 the Gaussian filter is the same as a simple averaging filter.

The deconvolution-based approach (Fig. 4.9(d)) is applied assuming very sparse configuration. Thus, a single LR area with the grid of Fig. 4.6(a) spans the entire frame, which is tesselated creating a different cell for each moving object. In the SR approach (Fig. 4.9(e,f)), the three motion areas are all configured to LR pixels of size 2×2 HR. To increase the robustness of the SR approach a neighborhood of 2 LR frames at each side of the integration interval is considered.

The frames illustrated in Fig. 4.9 demonstrate the final output for different

(a) The ground-truth frame. This is the output of an ideal, non-realistic sensor that combines HR spatial resolution with LR integration time.



(b) Motion blurred output produced for HR spatial resolution and thus low temporal resolution. $RMSE = 16.8$

(c) Bicubic interpolation applied on the output of a LR sensor. Due to the space-time trade-off, the sensor gives high temporal resolution. $RMSE = 4.7$

Figure 4.9: Reconstruction of dynamic regions.

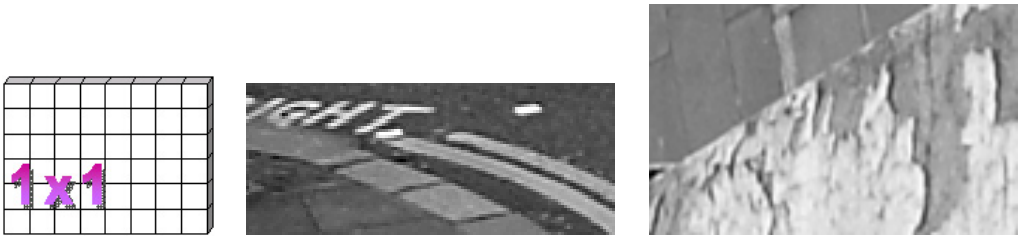
(d) Output of deconvolution-based motion deblurring. $RMSE = 6.92$. *Note: the RMSE is a poor metric for deconvolution. However, the visual results demonstrate increased readability of the number plates compared to bicubic interpolation.*

(e) Output of SR-based deblurring after 10 iterations of the IBP. $RMSE = 1.82$

(f) Output of SR-based deblurring after 30 iterations of the IBP. $RMSE = 0.85$

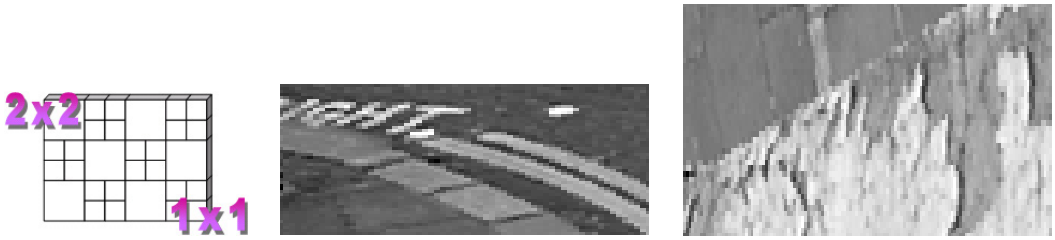
Figure 4.9: Reconstruction of dynamic regions.

(a) Regions of the static background.



(b) HR spatial resolution gives ground-truth for static regions.

(c) Reconstruction of static background for SR-based approach. This is the output of bicubic interpolation applied on the reference frame. $RMSE = 6.9$



(d) Reconstruction of static background for deconvolution-based approach. $RMSE = 4.6$

Figure 4.10: Reconstructing the static regions of the scene on the HR grid.

scenarios. The root mean square error (RMSE) is used as a metric to quantify the reconstruction quality. The RMSE values with respect to the ground-truth frame, which is illustrated in Fig. 4.9(a), are given in the captions of the individual subfigures of Fig. 4.9. The motion-blurred output of a sensor with HR spatial resolution is presented in Fig. 4.9(b). Fig. 4.9(c) illustrates the output of bicubic interpolation applied on a uniform LR pixel grid. The reconstructed output that is obtained after applying the deconvolution-based approach is shown in Fig. 4.9(d). The detail images of Fig. 4.9(d) demonstrate the increased readability that is achieved after deconvolution-based reconstruction on the number plates of the vehicles, in particular that of the car. This is not however reflected in the RMSE value, as the RMSE is not a good metric for deconvolution-based reconstruction [BIRB05]. Figures 4.9(e) and 4.9(f) illustrate the output obtained after executing 10 and 30 iterations of the IBP SR algorithm.

A visual comparison of Fig. 4.9(d) with Fig. 4.9(f) demonstrates the increased robustness of SR-based reconstruction on dynamic regions. In particular, the deconvolution-based approach achieves high readability for the first detail image, the number plate of the car, but not for the second detail image, which contains the number plate of the motor-cycle. This is mainly due to the accuracy of the spline interpolation that is employed in the deconvolution-based approach for the reconstruction of the motion PSF on the HR grid. The reconstructed motion PSF accurately describes the underlying motion of the car, but not of the motorcycle. In fact, for the case of the car, the achieved tem-

poral resolution is even higher than that of the ground-truth case, illustrated in Fig. 4.9(a). This is due to the fact that the reconstructed motion PSF, which describes the continuous motion within the HR integration interval, accurately describes the subsequent intra-LR-frame motions in the sequence of LR frames. Therefore, the LR samples are also effectively deblurred, and the reconstruction method achieves temporal resolution higher than the LR temporal resolution. However, the PSF reconstruction based on spline interpolation is not always accurate, as happens for the case of the motorcycle in Fig. 4.9(d). Therefore, the quality obtained with the deconvolution-based approach is dependent on the underlying motion, and thus the SR-based approach is preferable in the case of dynamic regions, as a more generic and robust solution.

If the configurations described in the previous paragraph are applied on the entire sensor, the static background also needs to be reconstructed on the HR grid. Considering the background of Fig. 4.10(a) without the presence of foreground objects, the root mean square error between the reconstructed and the ground-truth background is: (i) 4.6 after spatially enhancing the hybrid grid of the deconvolution-based approach and (ii) 6.9 after applying bicubic interpolation on the uniform large pixels of the SR-based approach. The lower root mean square error of the deconvolution-based approach is due to the higher fidelity of the output of the hybrid grid, relative to the ground-truth image of Fig. 4.10(b). As explained in Section 4.4, this is due to the fact that in the deconvolution-based approach 50% of the grid is covered with HR pixels. Thus, 50% of the pixels have the actual ground-truth values.

The amount of real-world information that is preserved on the static regions after employing the deconvolution-based approach is clearly higher than that obtained by the SR-based approach. Thus, the readability of the text of the first detail image is increased in Fig. 4.10(d), compared to Fig. 4.10(c). However, when it comes to the visual inspection of the output, a certain level of jagginess may be created on edges with strong directional properties. This may affect the perceived quality of the output of the deconvolution-based approach (Fig. 4.10(d)). Therefore, to improve the perceived quality of the output, a simple post-processing stage can be used, by employing multidirectional filters that will improve the image smoothness.

4.5.2 SR-Based Approach with Various Motion Magnitudes

In the *ice-skating* sequence, moving objects with different blur extents are acquainted, as it can be observed in Fig. 4.11(a). Clock-wise from left, these are: *Susan*, *Amy*, and *Jack* (Fig. 4.11(b1), Fig. 4.11(c1), and Fig. 4.11(d1), respectively). The pixel size of each one of the three motion regions is determined according to the local motion magnitude. The corresponding LR sequences are produced according to the space-time trade-off and are presented in Fig. 4.11(b2), Fig. 4.11(c2), and Fig. 4.11(d2). In the simulation process, these LR sequences are created independently, by applying on S_d a temporal blur of 36, 9 and 16, and a spatial blur of 2×2, 4×4 and 3×3, respectively.

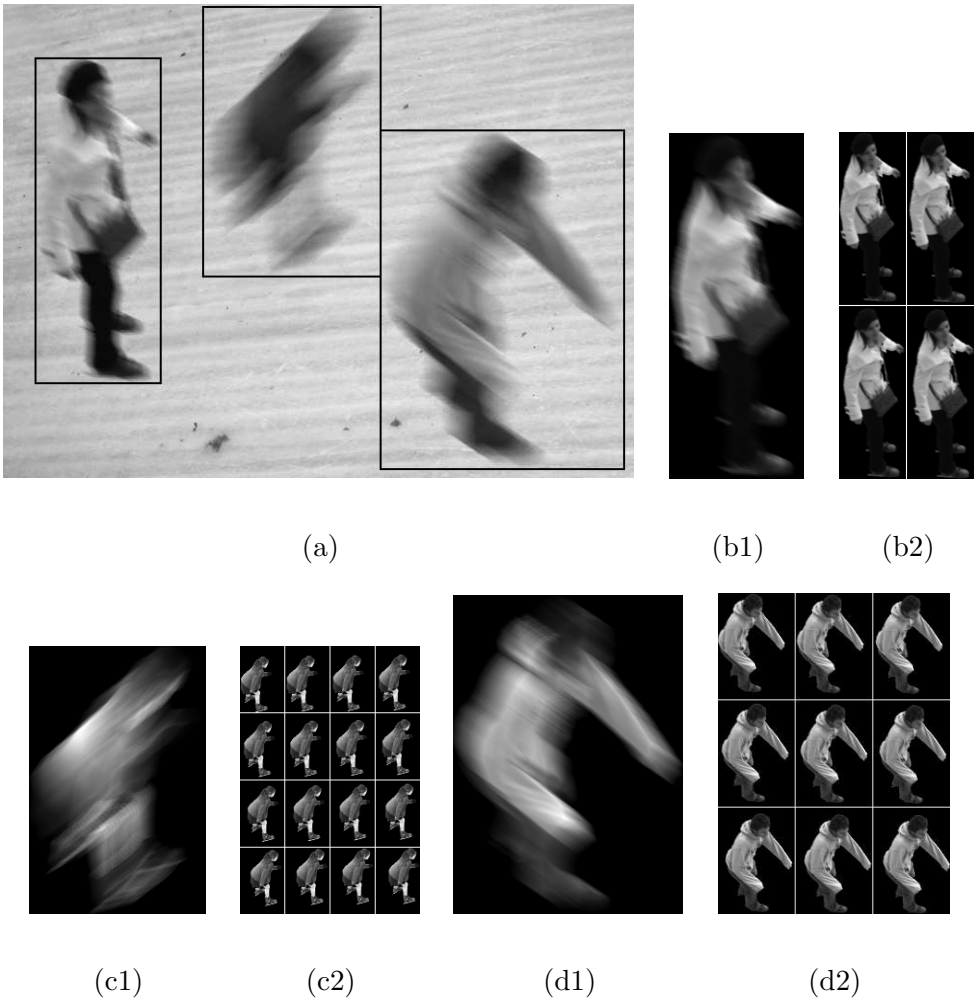


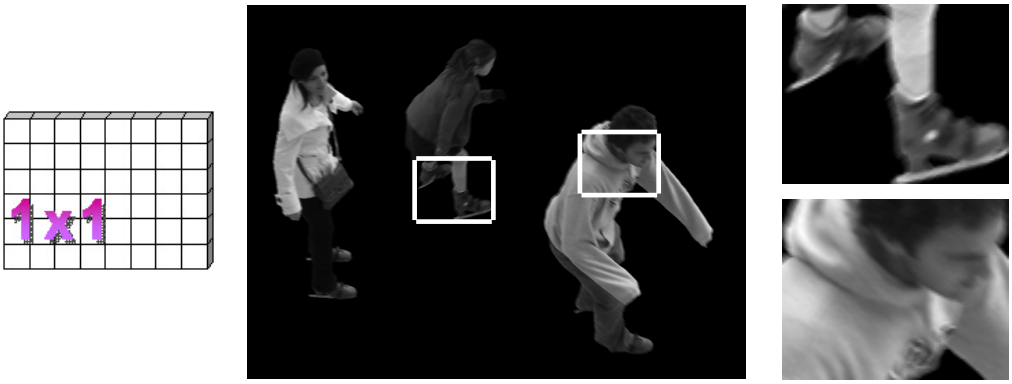
Figure 4.11: (a) A scene with three different motion magnitudes as captured by a conventional sensor. (b1, c1, d1) Motion-blurred regions. (b2, c2, d2) Determining the pixel size of each region based on the local motion magnitude, blur is reduced to a sufficient degree in the produced LR sequences. As the LR pixel size increases, more LR frames are produced, for the same time interval.

Three motion areas are formed in the SR approach, with LR pixel size 2×2, 4×4 and 3×3 HR, producing 4, 16 and 9 time samples during HR integration. Fig. 4.12(a) demonstrates the ground-truth frame, which is the output of an ideal sensor that combines HR spatial resolution with LR integration time. In reality, HR spatial resolution is associated with low temporal resolution, rendering the motion blurred output of Fig. 4.12(b), where the blur increases as the motion magnitudes increases. In Fig. 4.12(c), the output of bicubic interpolation is illustrated. The interpolation is executed considering magnification factors of 2, 4, and 3, for the independent motion regions *Susan*, *Amy*, and *Jack*, respectively. Finally, Fig. 4.12(d) presents the output of SR reconstruction, after executing 30 iterations of the Iterative Back Projection algorithm. Super-Resolution reconstruction (Fig. 4.12(d)) clearly outperforms bicubic interpolation (Fig. 4.12(c)) and successfully handles large degrees of blur. This is also clear from the quantitative comparison of the RMSE values, which are given in the captions of the individual subfigures of Fig. 4.12.

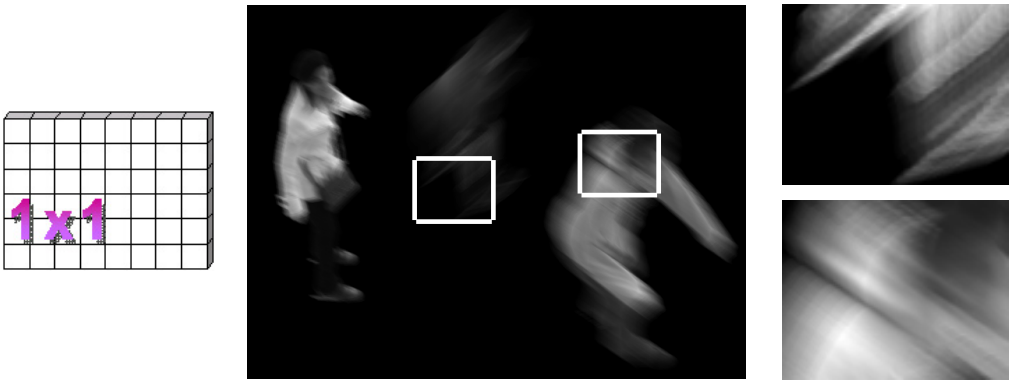
4.5.3 Quantitative Performance Evaluation

To provide generalized estimations of the reconstruction error, a variety of LR sequences has been produced, for various sensor configurations and pixel sizes. In particular, each LR sequence was synthetically produced by applying both temporal and spatial blur, as described in Sections 4.5.1 and 4.5.2, on the 8 ground-truth frames of Fig. 4.13.

The statistical quantitative results, *i.e.* the average RMSE (ϵ_μ) and the

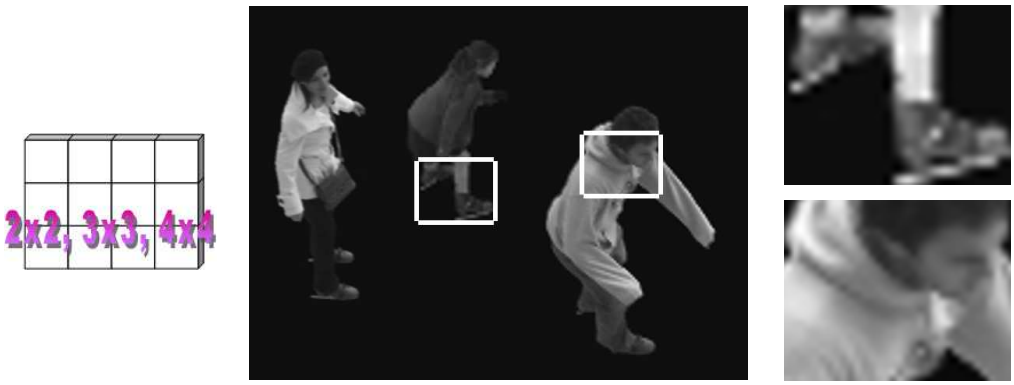


(a) Ground-truth frame. This is the output of an ideal sensor that combines HR spatial resolution with LR integration time.

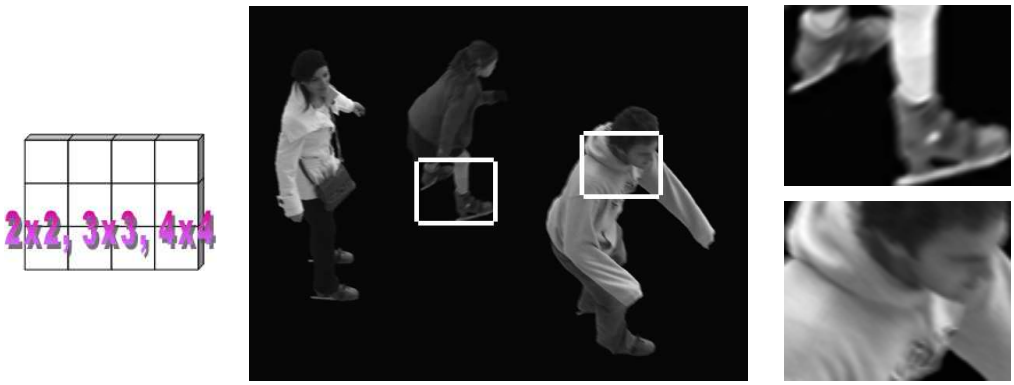


(b) Motion blurred output that would be produced if the sensor had HR spatial resolution and, therefore, low temporal resolution. $RMSE = 15.26$

Figure 4.12: Reconstruction of dynamic regions for different motion magnitudes.



(c) Bicubic interpolation applied on the output of a LR sensor, where the pixel size of each motion area is determined according to the motion magnitude. $RMSE = 2.2$



(d) Output of SR-based deblurring after 30 iterations of the algorithm. The reconstruction is executed individually on the three motion regions, and the LR pixel size is different for each region. $RMSE = 0.54$

Figure 4.12: Reconstruction of dynamic regions for different motion magnitudes.

standard deviation of the RMSE values (ϵ_σ), which describe the statistics of the outputs of the above experiments, are given in Table 4.1. Results are given both for the static and the dynamic case. For the static case, the entire sensor is configured either with a uniform or with a hybrid pixel grid, for the SR-based and deconvolution-based approaches respectively, with LR pixel size 2×2 or 3×3, as indicated in Table 4.1. For the dynamic case, a single motion area is considered, which spans the entire sensor. This is the case when global camera motion occurs. The given reconstruction errors correspond to the following scenarios: SR reconstruction after executing 10 iterations and after executing 30 iterations of the IBP, and deconvolution-based reconstruction on a hybrid grid. Note that in the case of the SR-based approach the reconstruction for the static case is executed by applying bicubic interpolation. Therefore, the reconstruction error in the static case is only different when a hybrid grid is employed, which is done when the deconvolution-based approach is used.

Table 4.1 illustrates that for the group of conducted experiments the SR-based approach renders highly accurate reconstruction for the case of dynamic regions, whereas the hybrid grid of the deconvolution-based approach performs better for the static case. The above observations are in accordance with the discussion in Section 4.4, as well as with the conclusions of Sections 4.5.1 and 4.5.2.

Figure 4.13: Ground-truth frames used for the generation of the test sequences.

LR pixel size	2×2				3×3			
Type of region	*static*		*dynamic*		*static*		*dynamic*	
Error metric	ϵ_μ	ϵ_σ	ϵ_μ	ϵ_σ	ϵ_μ	ϵ_σ	ϵ_μ	ϵ_σ
$SR-based$ 10	7.16	0.68	1.74	0.34	10.16	1.04	1.77	0.36
$SR-based$ 30	7.16	0.68	0.76	0.13	10.16	1.04	0.78	0.16
$Deconv.-based$	4.88	0.57	6.58	0.63	6.87	0.60	8.86	0.75

Table 4.1: Average (ϵ_μ) and standard deviation (ϵ_σ) of the reconstruction error for the group of experiments that are executed on the test sequences.

4.6 Extending the Proposed Methods to Non-Rigid Objects

The work presented in the previous sections assumes rigid moving objects. Thus, future work related to this chapter mainly includes extending the proposed methods to non-rigid objects. However, it should be noted that, for the particular application addressed in this work, the general problem of reconstruction using multiple frames with non-rigid motions is simplified, since the LR frames participating in the reconstruction are neighboring frames. This confines the inter-frame motions and reduces their non-rigidness.

Extending the deconvolution-based approach to non-rigid objects is straightforward in the sense that more motion PSFs will be constructed per object. This means that a single moving object will span multiple Voronoi cells, each associated with a particular PSF that describes the motion of a particular part of the object.

The extension of the SR-based method to effectively handle non-rigid objects is less straightforward than that of the deconvolution-based approach. The issue of extending the SR framework in order to accommodate non-rigid motions is thoroughly discussed in [BK99].

4.7 Summary

In this chapter, two methods of configuring an adaptive image sensor are proposed. These configurations aim to maximize the information collected from the environment. For each configuration, an appropriate system is developed to process that raw information and enhance the output. Specifically, an SR-based and a deconvolution-based approach, that execute motion deblurring on an adaptive sensor, are proposed, compared and evaluated. Results demonstrate that the SR method performs better in dynamic regions and is preferable when the sensor can be frequently configured. Moreover, this method effectively manages large blur by increasing the LR pixel size. Experiments show that the deconvolution-based approach achieves better reconstruction of static regions and is suitable for sparse configuration, as large static parts are included in LR areas.

As demonstrated in this chapter, the deconvolution-based approach is not as robust as the SR-based approach for the reconstruction of dynamic regions, in particular for cases of large motions. Therefore, in the chapters that follow, the focus is on the SR-based approach. As for the problem of deblurring the

individual LR samples, which is briefly mentioned in Section 4.5.1, this is also incorporated within the framework of the SR-based approach.

Since the resolution enhancement of dynamic regions is the main area of interest of this thesis, the next chapters deal with the SR approach, which is more appropriate for this type of regions. In particular, Chapter 5 deals with the classic SR problem, where isotropic Gaussian PSFs are appropriate to describe the blur of the LR frames, while Chapter 6 uses non-isotropic motion PSFs to describe non-negligible intra-frame motion in the LR samples. The latter is required for the reconstruction process to render accurate results in the case of considerably fast motion, due to which even the high frame-rate samples contain significant motion blur.

Chapter 5

Real-Time Super-Resolution with Isotropic PSFs

In this chapter, an FPGA architecture is proposed for the implementation of the resolution enhancement module of the SR algorithm based on the Iterative Back Projection (IBP) [IP91] approach. 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 proposed architecture comprises a general-purpose SR hardware block. The reconstruction quality of the architecture is evaluated under different noise levels. Moreover, 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 contributing in SR, the initialization of the SR iterative scheme, and the word-length of the data path. In particular, an investigation of the impact of the word-length of the data path to the convergence and the reconstruction quality of the IBP

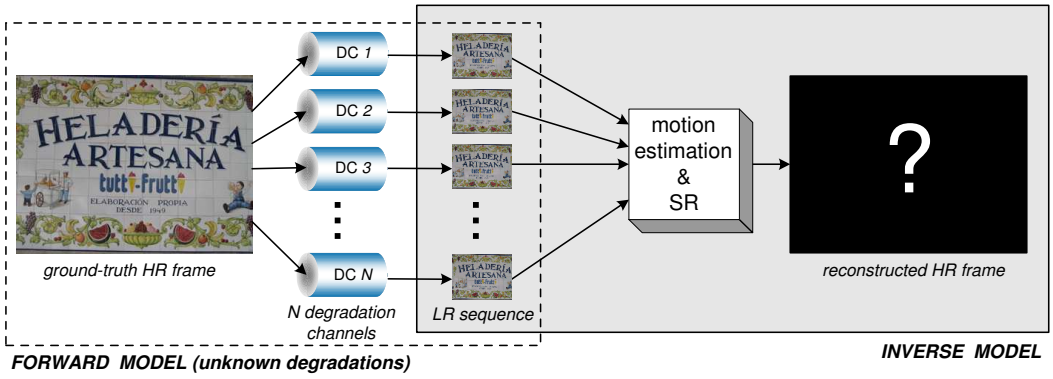


Figure 5.1: The forward model consists of a series of unknown degradations related to the imaging system. The inverse model estimates those degradations to reconstruct the missing high-resolution data.

algorithm is performed.

The structure of the chapter is as follows. Section 5.1 introduces SR, while Section 5.3 focuses in the Iterative Back Projection SR algorithm. Section 5.4 discusses how the reconstruction algorithm can account for the presence of noise to produce a more robust system. Section 5.5 describes the FPGA implementation of the SR block. In Section 5.6, the implementation requirements are discussed, and the system performance is evaluated, considering different noise levels and system parameters.

Parts of the work presented in this chapter have been published in [ABCC08] and [ABCC09].

5.1 Introduction to Super-Resolution

The forward model, which describes the generation of LR samples at the output of an imaging system, was presented in Chapter 2. The parameters of the forward model are estimated and used by the inverse model, which is algorithmically formed to reconstruct the missing high-resolution data. In the high-level diagram of Fig. 5.1, 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 Fig. 5.1. The inverse problem uses as inputs these LR samples and approximates the related degradations, so as to apply the inverse procedure and reconstruct the missing high-resolution data. The original frame at the input of the forward model (Fig. 5.1) 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 known precisely 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 aim of the inverse problem is to get an HR image from the LR channels.

The forward model of generating LR pixels is shown in Fig. 5.2. Many HR pixels are mapped on a single LR pixel, thus imitating the integration of a

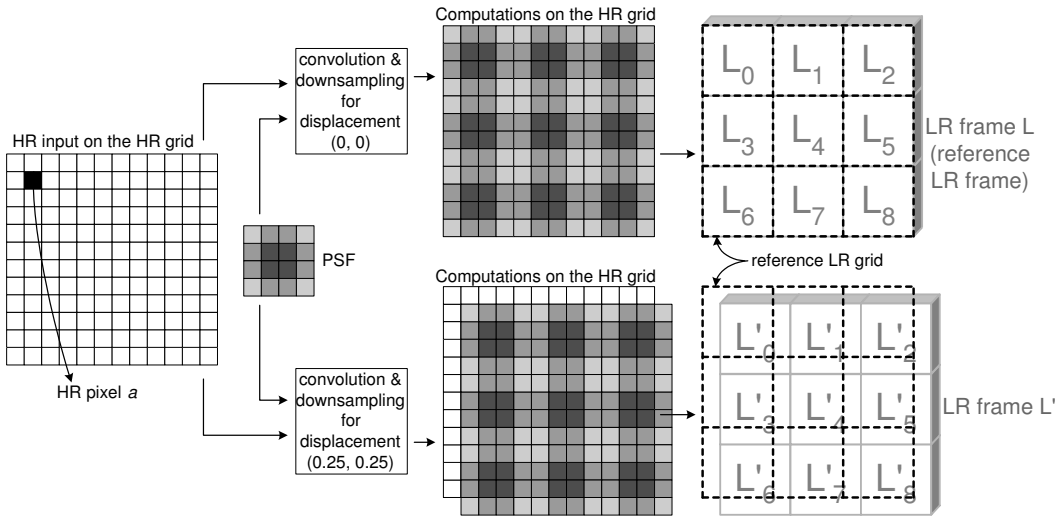


Figure 5.2: The formation of the LR output presented mathematically. A 4×4 PSF is employed. Two simulated LR frames with displacements $(0, 0)$ and $(0.25, 0.25)$ are produced.

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. 5.2. The Gaussian PSF assumption is the usual assumption for this type of problem. As mentioned in Chapter 2, the 2-D Gaussian function closely resembles the photodiode's sensitivity, which is high in the middle and decreases towards the borders with a Gaussian-like decay. Every LR pixel can be thus expressed as a weighted sum of HR pixels, and the following linear system of equations is formed:

$$A\vec{h} = \vec{l} \tag{5.1}$$

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 (Fig. 5.2). 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 (Fig. 5.2) 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 an LR frame to contribute uniquely in the system of (5.1), 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 additive noise present in the LR samples (Chapter 2) and partly due to errors in the motion estimation procedure [Bou02]. Therefore, in practice it is preferable to consider many LR frames, even if their displacements overlap.

5.2 Related Work

The SR methods found in the literature solve the SR problem either in the spatial or in the frequency domain [PPK03, BK02, FREM04]. 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, the need for long word-lengths in hardware implementations is not required. Among the spatial domain methods the Iterative Back Projection (IBP) [IP91] approach was selected because of its hardware-friendly characteristics. Instead of solving (5.1) 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 re-use, as it will be demonstrated in Section 5.5.

Most of the work on SR found in literature is concerned with the algorithmic aspects of SR [PPK03, BK02, FREM04]. The computationally demanding nature of the SR problem renders software solutions inadequate when real-time specifications are imposed. Surprisingly, the literature that is related to hardware architectures is very limited. In [CLL$^+$05a], SR is implemented with a system that uses VLIW and ARM processors. In [BB08], an FPGA architecture for the implementation of the SR algorithm of [FEM04] is proposed, whose reconstruction quality is however poor and is extremely sensitive to errors from the motion vectors. Moreover, a very large number of LR frames are required for the reconstruction process. In particular, when 25 LR frames participate in the reconstruction, the Root Mean Square Error (RMSE) of the reconstructed output equals 25 when no errors are present in the motion vectors. When half of the frames contain errors in the motion vectors, the RMSE increases to 34.

The architecture presented in this chapter achieves significantly higher reconstruction quality, rendering an RMSE which is an order of magnitude smaller than that of [BB08].

5.3 The Iterative Back Projection (IBP) Super-Resolution Approach

The 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 Fig. 5.2 is applied on 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 sequence, 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:

$$e^{(i)} = \sqrt{\sum_{k=0}^{K-1} \sum_{(x_l,y_l)} (Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l))^2} \tag{5.2}$$

where (x_l, y_l) denote 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 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. 5.2), pixel L_0 of frame L participates with a weight proportional to $PSF(1,1)$. However, in the refinement of the same HR pixel, the weight of pixel L'_0 of frame L' is proportional to $PSF(0,0)$ (Fig. 5.2). At iteration i, every pixel of the current high-resolution approximation H_i is refined as follows:

$$H_{i+1}(x_h, y_h) = H_i(x_h, y_h) + \sum_{k=0}^{K-1} \sum_{(x_l, y_l) \in Y} (Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l)) \times W(k, x_l, y_l), \tag{5.3}$$

where (x_h, y_h) denote 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 in the refinement of $H_i(x_h, y_h)$.

5.4 Increasing the System Robustness

As explained in Section 5.1, the error given by (5.2), which determines the quality of the SR algorithm, is the error between the simulated and observed LR sequences. In the ideal situation of noise-free LR samples, it can be proven that if the error function of (5.2) 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 Chapter 2. In the presence of such noise, the error function of (5.2) 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 [IP91], averaging in (5.3) 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 neighbourhood around the integration interval of interest (Section 5.1). By doing the above, the correlation in time among the two ends of the LR sequence becomes low. This may increase the level of non-rigid deformations of the moving object, which makes in turn the fusion of the relevant LR information more difficult.

5.4.1 Reducing the Noise Effect

To keep the number of LR frames within reasonable limits, so as to both avoid undesirable types of changes in the scene and execute an efficient reconstruction, we propose including in the expression of (5.2) information about the statistics of the noise. This is done in a manner inspired by the Projection onto Convex Sets (POCS) framework [SPBDK01, SO89, YW82]. Specifically, 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 of $Lo^k(x_l, y_l)$ and $Ls_i^k(x_l, y_l)$ falls within that interval, these pixels will *not* contribute in 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:

$$r_i(x_h, y_h, k, x_l, y_l) = Lo^k(x_l, y_l) - Ls_i^k(x_l, y_l) \tag{5.4}$$

then, (5.3) is modified as follows:

$$H_{i+1}(x_h, y_h) = H_i(x_h, y_h) + \sum_{k=0}^{K-1} \sum_{(x_l, y_l) \in Y} u_i(x_h, y_h, k, x_l, y_l) \tag{5.5a}$$

$$u_i(x_h, y_h, k, x_l, y_l) = \begin{cases} (r_i(x_h, y_h, k, x_l, y_l) - \delta_0(k)) \times W(k, x_l, y_l), \\ \qquad\qquad if\ r_i(x_h, y_h, k, x_l, y_l) > \delta_0(k) \\ 0, \\ \quad if\ -\delta_0(k) \leq r_i(x_h, y_h, k, x_l, y_l) \leq \delta_0(k) \\ (r_i(x_h, y_h, k, x_l, y_l) + \delta_0(k)) \times W(k, x_l, y_l), \\ \qquad\qquad if\ r_i(x_h, y_h, k, x_l, y_l) < -\delta_0(k) \end{cases} \tag{5.5b}$$

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 the 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 (5.5) is simplified to (5.3). Thus, (5.3) can be considered as a special case of (5.5), 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 [Bou02] 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 (Fig. 5.2) 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 above.

5.5 Architecture of the SR System

Figure 5.3 shows an overview of the proposed system. For every new group of LR frames, produced during a particular HR integration interval, 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 HR integration interval) is processed, and so on. The rest of the 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

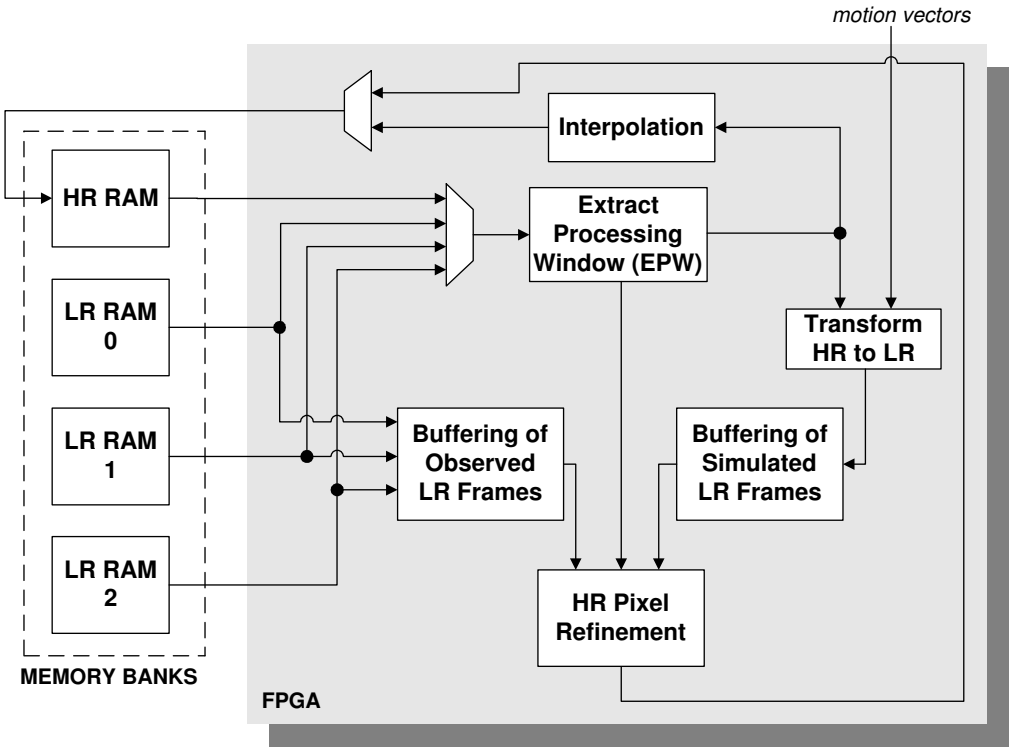


Figure 5.3: High level architecture overview. The architecture has been implemented on a Celoxica ADMXRC4SX board [Cel08], which hosts a Xilinx Virtex-4 FPGA and 4 Zero Bus Turnaround (ZBT) SSRAM banks.

the reference frame, are treated as inputs to the implemented SR system. It should be mentioned that the target system has 4 memory banks, each with a word-length of 4 bytes.

5.5.1 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, as explained in Section 5.4, 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. Fig. 5.4(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 in the processing of an LR frame neighbourhood. In Fig. 5.4(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 Fig. 5.4(a) and execute on-line processing of the LR data, two memory banks need to be read in parallel, as Fig. 5.4(b) illustrates for the case of a 2×2 PSF and four neighboring frames. For instance, in SR stage 1 (Fig. 5.4(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. 5.4(b). There are

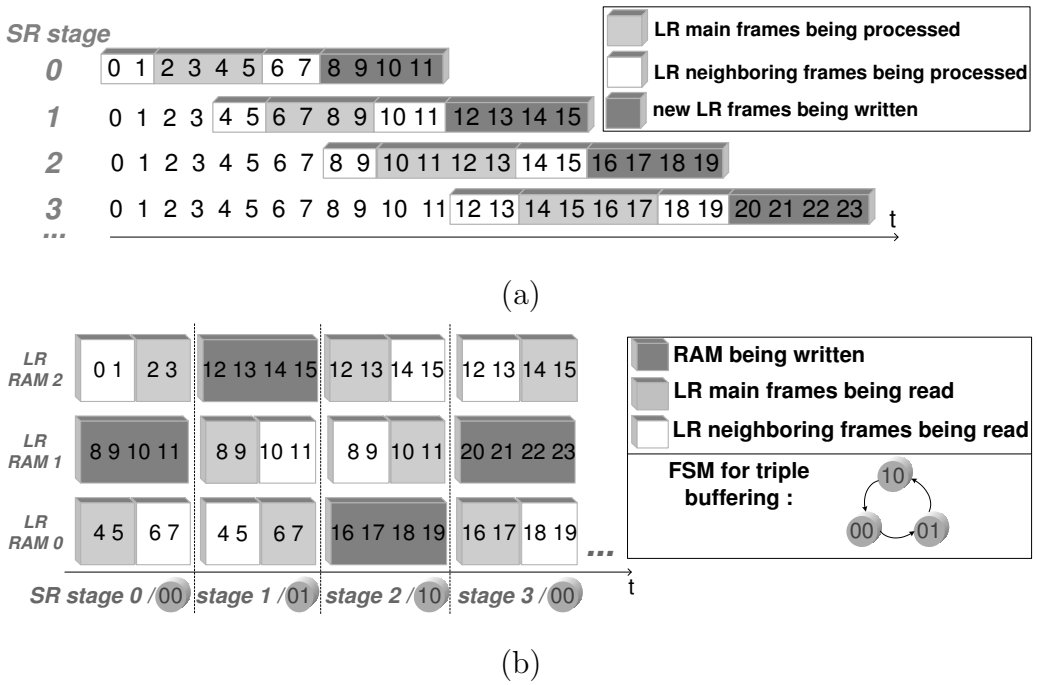
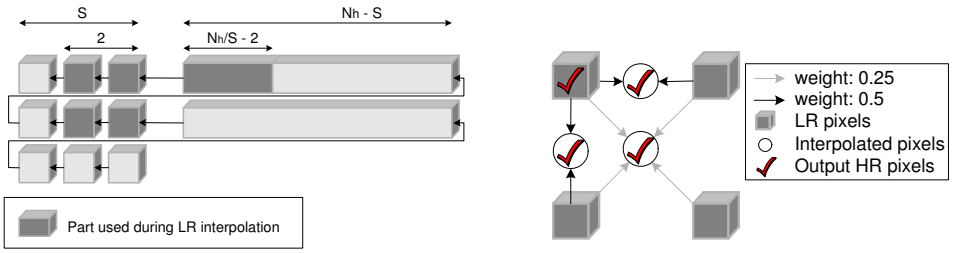


Figure 5.4: The numbers correspond to the LR frames. (a) A sliding window indicates the group of frames processed during the current SR stage. While the processing occurs on these frames, a new group of four frames is written in the memory banks. (b) Triple buffering scheme applied on the LR RAMs.

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 Fig. 5.4(b), a particular configuration appears every 3 SR stages.



(a) (b)

Figure 5.5: (a) Extract Proccessing 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

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.

5.5.2 Individual Processing Units

The *Extract Processing Window* (EPW) unit of Fig. 5.3 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 Fig. 5.5(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 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, as illustrated in Fig. 5.5(b), or to a raw pixel of this 2×2 group of LR pixels. This depends on the position of the interpolated HR pixel, as it is demonstrated in Fig. 5.5(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 5.1) will be produced by subsampling the output of the convolution of the last high-resolution 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 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 Fig. 5.6. These parallel branches meet at a final adder, which corresponds to the external summation in (5.3) or (5.5), to produce the refined version of the HR pixel that is currently under process. In [ABCC08], the iterative process was executed

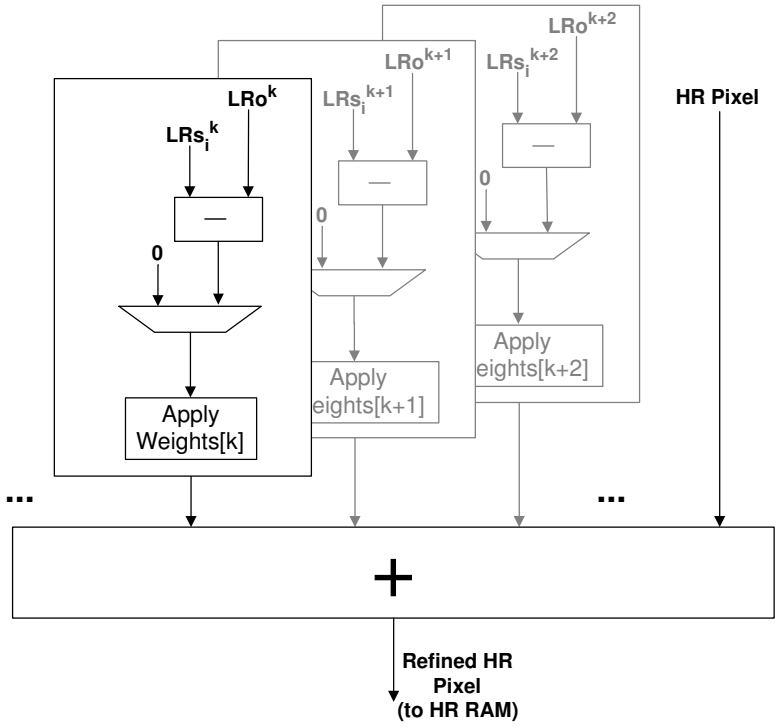


Figure 5.6: The HR Pixel Refinement Unit. LRo^k and LRs_i^k denote the contribution of each pair of LR frames Lo^k and Ls_i^k, in the refinement of the particular HR pixel.

without taking into account the noise statistics related to each LR frame, and thus (5.3) was implemented. In the current work, (5.5) is used instead. The hardware that implemented (5.3) is modified to accommodate (5.5), by including an extra multiplexer that executes the control logic of (5.5b), leading to a more robust to noise algorithm, as it is demonstrated in the performance evaluation section.

5.5.3 Data Re-use and Maximum Performance

Each HR and observed LR pixel are read from the corresponding RAM only once and remain on-chip until they are no longer needed. Thus, data re-use is maximized. Also, for maximum performance, one iteration requires the number of cycles imposed by the HR convolution (Section 5.5.2). 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 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* 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 Fig. 5.7(a) pointing at one HR pixel at every cycle. This is the HR pixel that is currently

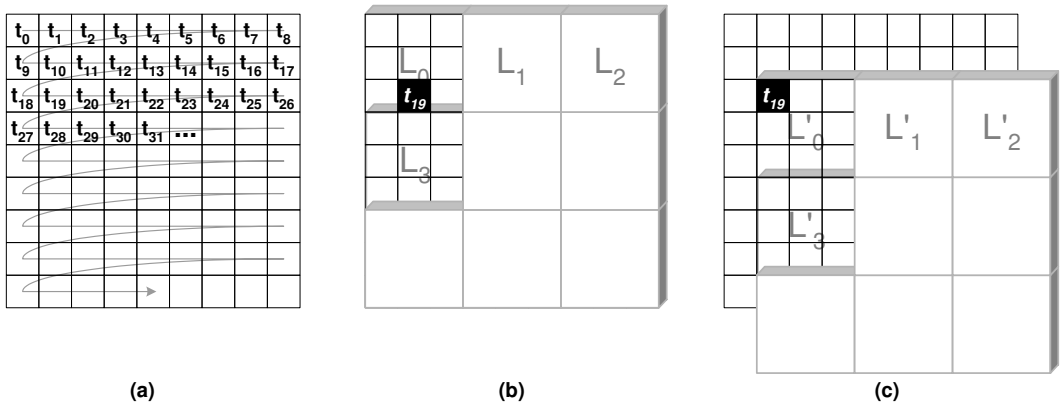


Figure 5.7: Temporal Aspect of HR Grid (a) At every clock cycle the cursor moves one position on the HR grid indicating the currently processed HR pixel. (b) The reference LR frame (c) An LR frame with displacement (2,1)

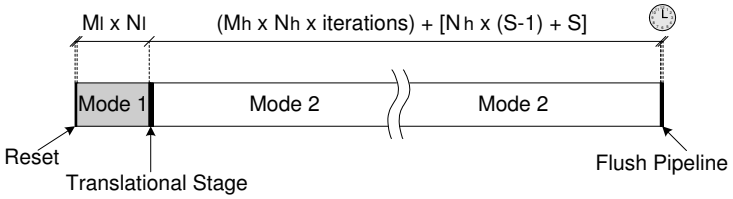


Figure 5.8: Time diagram with execution cycles.

under process. Therefore, HR pixel h_0 would correspond to time-slot t_0 and so 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 (Fig. 5.7(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 5.1.

The time diagram with the total number of cycles required for the above specifications is shown in Fig. 5.8. In Fig. 5.8, 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 Fig. 5.8 are the two operating modes of the *Extract Processing Window* unit, which are described in Section 5.5.2.

5.5.4 On-chip Memory

The units *Buffering of Simulated LR Frames* and *Buffering of Observed LR Frames* of Fig. 5.3 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 not be used anymore at the current iteration.

Ls buffers

The width of the Ls buffers is equal to that of the LR frames. Note here that since several LR frames might have the same subpixel displacement, the output of the *Transform HR to LR* unit might be written simultaneously to multiple Ls buffers.

Lo buffers

These buffers need to be synchronized with buffers Ls. As mentioned in the previous paragraph, multiple Ls buffers might be written simultaneously in the case of LR frames with identical displacements. An equivalent scenario would

be impossible in the case of the *Lo* group of buffers, as it would be accessing in parallel multiple locations of the single-port off-chip LR RAM. The problem increases as the number of LR frames increases, and in particular if many of these frames have the same displacements. We cannot just "move" the request to a later time slot, as this would probably be occupied by another LR frame.

We should, therefore, increase the size of the *Lo* buffers, so that they would store more pixels than the width of the LR frames. While a particular line is being read from the on-chip buffers, the next line, or part of the next line, will be read from the RAM and written on-chip. In the current version of the system polling is used, by assigning slots dedicated to each of the *Lo* buffers. Also the capacity of these buffers is chosen to be twice the width of the LR frames. An alternative would be to use requests from the *Lo* buffers to the RAM, when the content of a location is no longer needed.

Since the pixels in the LR RAMs are written in groups of four, we have used a word of 32 bits for these buffers, to store four pixels at a time.

5.6 Results

5.6.1 Implementation Requirements

The design targets a Celoxica ADMXRC4SX board [Cel08]. The DK5 Handel-C compiler has been used, and the implementation has been placed and routed using Xilinx ISE v.9.1. The ADMXRC4SX board hosts a Xilinx Virtex-4 FPGA [Xil08] and 4 ZBT SSRAM banks [Cel08]. The operating frequency of

Size	64×64	128×128	256×256	240×320	512×512	480×640	1024×1024
Iter.	780	195	48	41	11	10	2

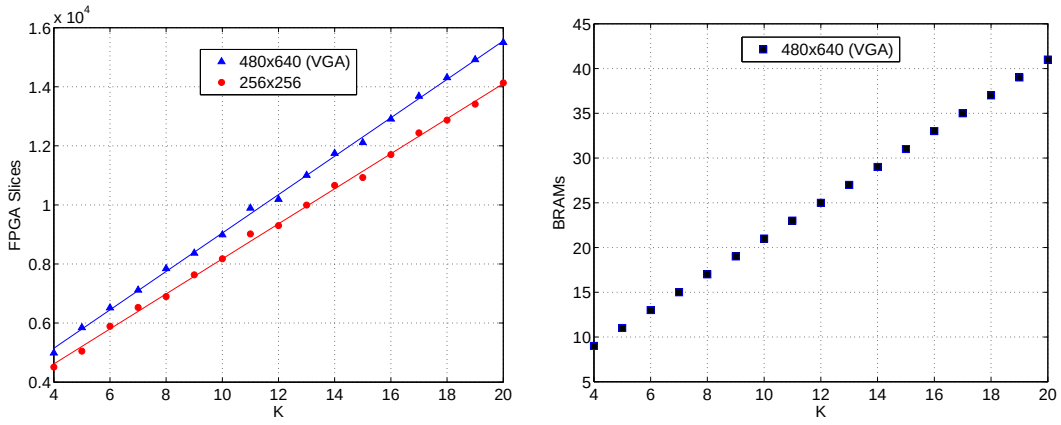
Table 5.1: Iterations for real-time performance for different HR sizes.

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 Fig. 5.4. To meet real-time requirements, the system should achieve 25 fps. As shown in Fig. 5.8, the required number of cycles for SR reconstruction is:

$$C = R + M_l \times N_l + M_h \times N_h \times I + [N_h \times (S-1) + S] + Latency, \quad (5.6)$$

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 $\lceil log_2(K+1) \rceil$, *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 permitting 25 fps derives from (5.6) and is given in Table 5.1.

The number of FPGA slices is mainly affected by K, *i.e.* the number of LR frames, and does not significantly vary for different frame sizes, as Fig. 5.9(a) demonstrates for 256×256 and 480×640 HR size. Linear least squares fitting is applied on the samples of Fig. 5.9(a), giving the following linear equations



(a) (b)

Figure 5.9: 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 5.1.

for the number of FPGA slices with respect to K:

$$S_{VGA} = 650 * K + 2546 \tag{5.7}$$

$$S_{256} = 593 * K + 2241 \tag{5.8}$$

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 Fig. 5.9(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 Fig. 5.9, is due to the parallel processing of the different

LR frames. This has been discussed in Section 5.5. Thus, as it is demonstrated in Fig. 5.6, 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 in parallel pixels from all of the LR frames and, thus, achieve a throughput of one HR pixel per cycle, as explained in Section 5.5.4.

5.6.2 Performance Evaluation

The performance of the implemented algorithm has been investigated using different sets of evaluation parameters. Such 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 above word-length corresponds to the word-length of the output of each IBP iteration that is propagated to the next iteration. The evaluation is done using semi-synthetic 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 white Gaussian 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 & Kanade optical flow [Bou02] and Shi & Tomasi good feature extraction [ST94] 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, which range from 10 to 70 dB, with a step of 10 dB.

The SNR level is defined as follows.

Definition 3. *Let σ_f denote the standard deviation of the noise-free image, and σ_n denote the noise standard deviation. Then, the SNR of the noisy image, which is contaminated with the above noise, is:*

$$SNR = 10 \log \frac{\sigma_f^2}{\sigma_n^2} \tag{5.9}$$

The value σ_n of (5.9) is the noise standard deviation $\delta_0(k)$ of (5.5b).

The noise-level of the captured frames is determined by the level of illumination and the sensor quality [TFG01]. Under moderate illumination, most cameras give an SNR between 50 and 60 dB, which may decrease to 30 dB for lower illumination levels. The SNR under moderate illumination is thus related to the technology of the image sensor and is typically given by the camera manufacturer in the camera datasheet. An example camera datasheet including information about the SNR is given in Appendix A.

The evaluation process has been repeated for different numbers of LR samples. Specifically, 4, 8, and 12 LR frames are used. The output of every iter-

ation, which comprises the input of the next iteration, has been 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 has been developed for bicubic interpolation.

To demonstrate the benefits of SR, double precision floating-point bicubic interpolation has been 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 the output of a single LR channel, using solely 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 (5.5), with the exception of a comparative analysis between (5.3) and (5.5), which is presented for the second group of experiments to prove the increased robustness of (5.5) with respect to noise.

The Classic SR Problem

In the first group of experiments, the natural image of Fig. 5.10(a) was used to produce a group of LR samples, the *drinks* LR sequence. Random displace-

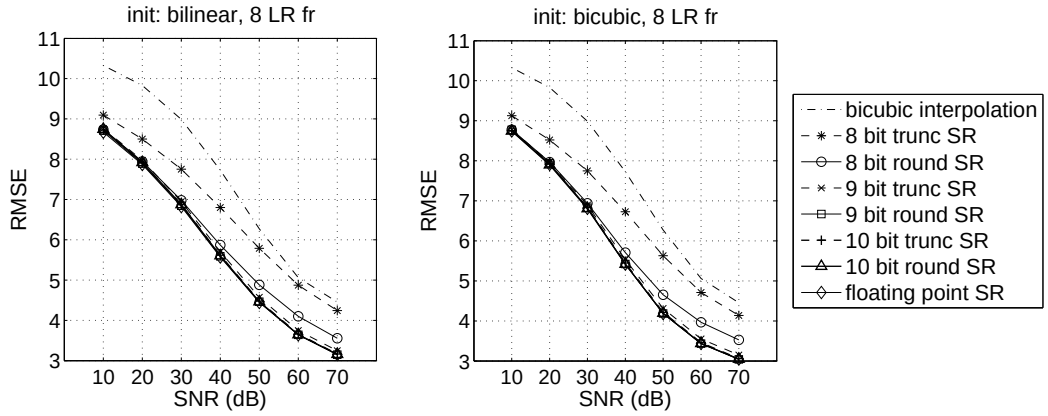
(a) (b)

Figure 5.10: "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.

ments 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.

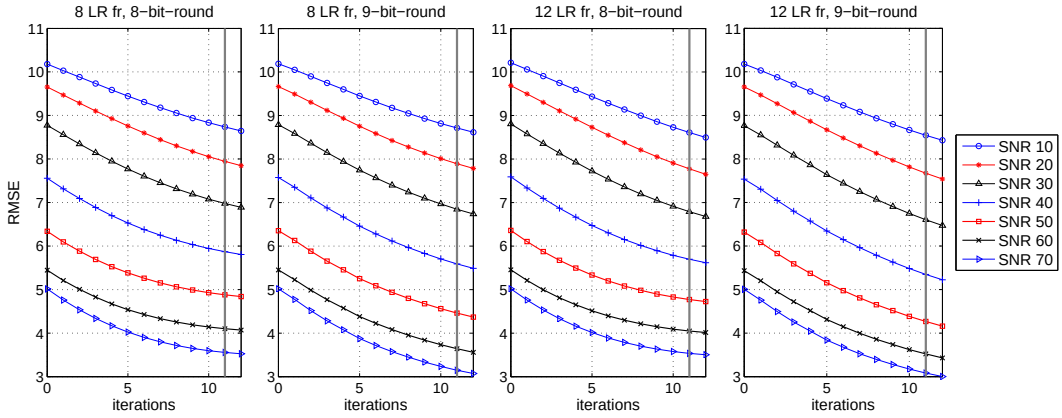
The graphs of Fig. 5.11 demonstrate the RMSE values obtained when using 8 LR frames of the *drinks* sequence with different word-lengths and SNR values, after executing the number of iterations indicated in Table 5.1 for 512×512 HR. The difference among these two graphs lies in the type of interpolation used as the initial guess: bilinear interpolation is employed for Fig. 5.11(a) and bicubic for Fig. 5.11(b). These graphs are very similar, which proves the good convergence properties of the algorithm of (5.5). Bilinear interpolation is thus preferable, as its leads to similar results with lower computational cost.

The graphs of Fig. 5.12 demonstrate the decrease in the RMSE as the iter-



(a) (b)

Figure 5.11: 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.



(a) (b) (c) (d)

Figure 5.12: 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.

ations 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 up 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 with 8 LR frames, shown in Fig. 5.12(a), gives outputs of similar quality to both larger word-lengths and longer LR sequences (Fig. 5.12(b-d)).

In Fig. 5.13, visual results of the reconstructed output are demonstrated for the set of parameters of Fig. 5.12(a), after executing the number of iterations for real-time performance. Each row of Fig. 5.13 corresponds to the SNR value indicated on the left. The part of the frame shown in Fig. 5.13 corresponds to the "ground-truth" part of Fig. 5.10(b). The first column of Fig. 5.13 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 spatial resolution and LR temporal resolution is achieved, leading to superior results.

The Motion Deblurring Problem

In the second experiment, a motion area employing pixels of 2×2 S_h is considered, which produces 4 time samples during the HR integration. If very fast motion is involved, the LR frames are blurred themselves. To incorporate this

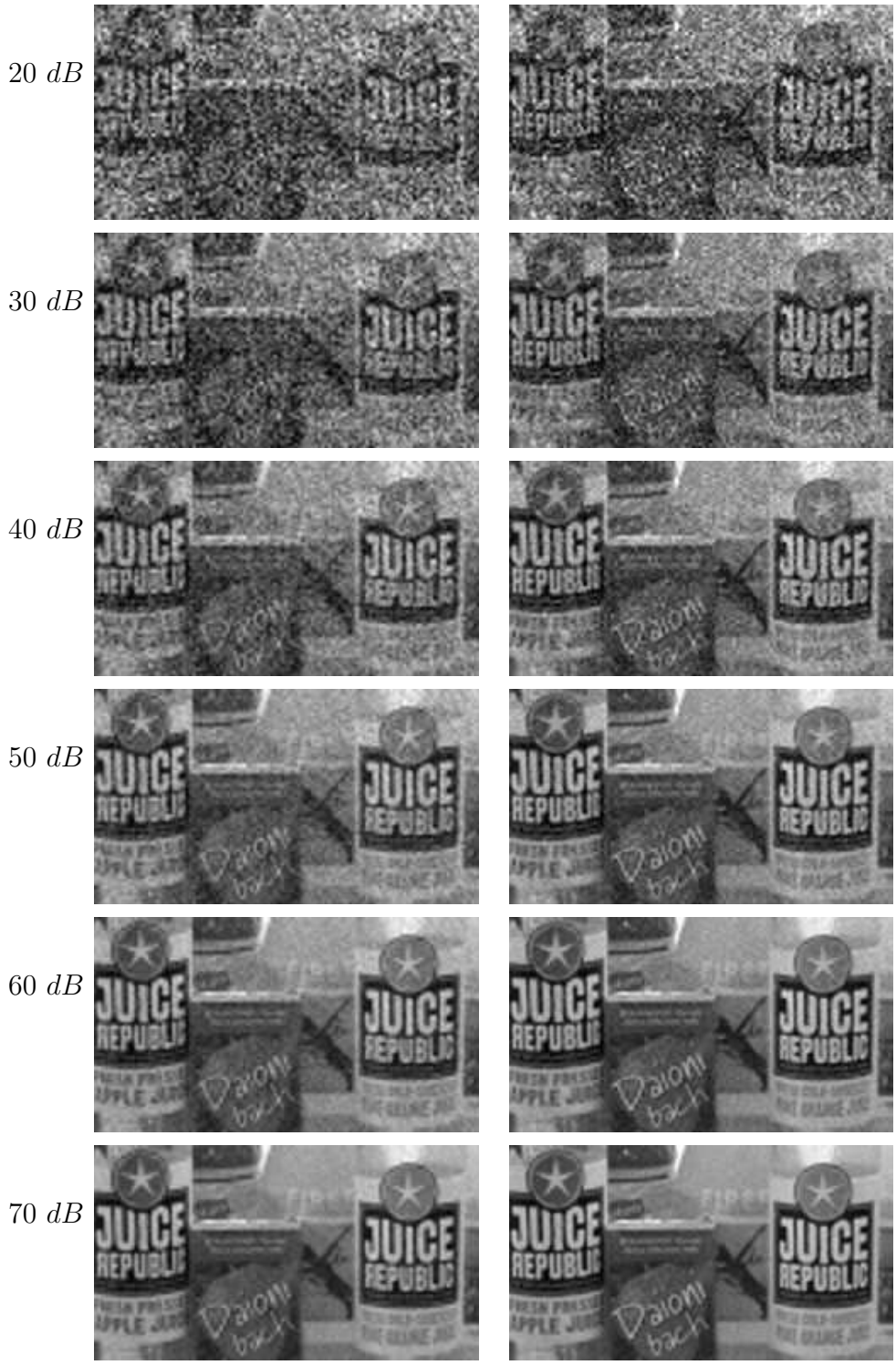


Figure 5.13: Reconstruction of *drinks* for various SNRs. The first column shows the output of floating-point bicubic interpolation using a single LR frame, while the second one demonstrates the higher quality obtained by the real-time SR FPGA implementation.

(a) (b)

Figure 5.14: "Ground-truth" frame for the *car* sequence, *i.e.* the output of an ideal sensor combining HR spatial resolution with LR temporal resolution.

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 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 Gaussian PSF was then applied on sequence A (for size 2×2, the Gaussian PSF is actually a uniform 2×2 PSF). Then, 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 Fig. 5.14.

The graphs of Fig. 5.15(a,b) describe exactly the same scenarios as those of Fig. 5.11(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 5.1. As

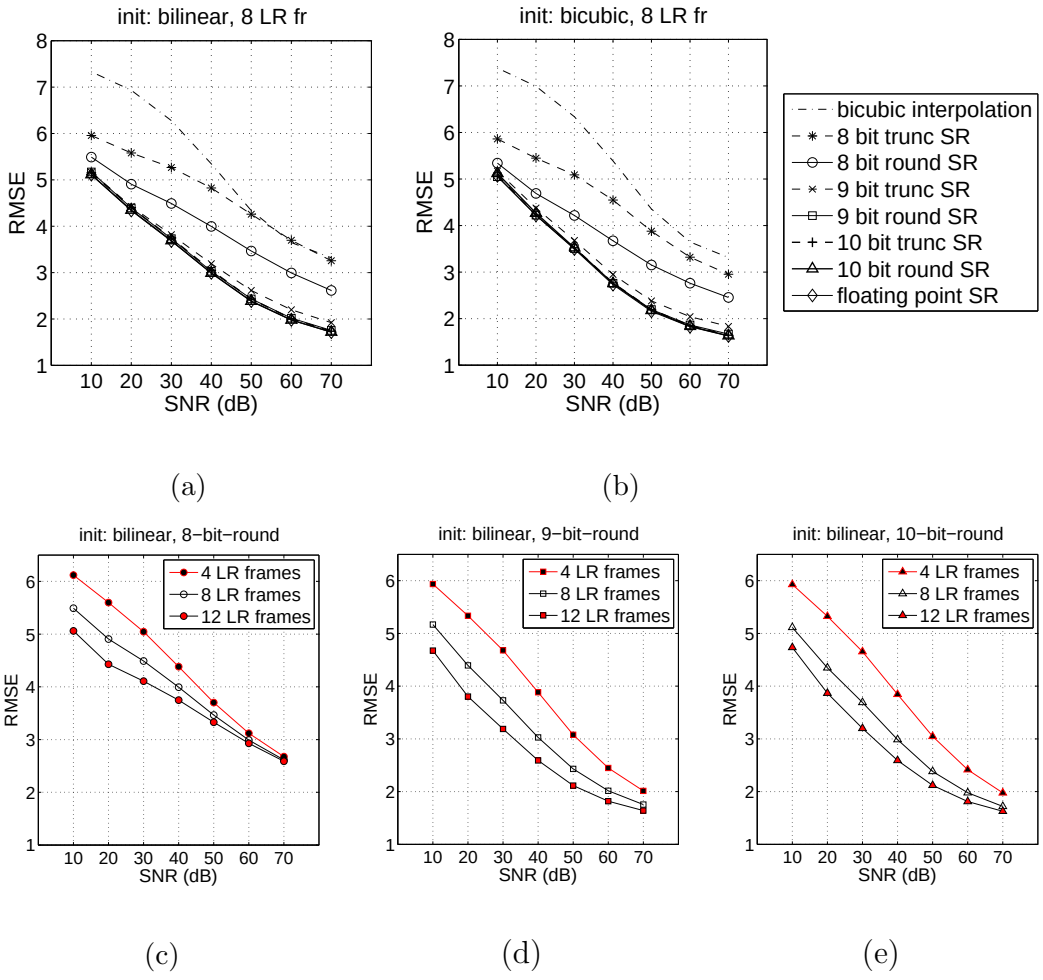


Figure 5.15: RMSE values for real-time SR on the *cars* sequence. (a, b) Employing 8 LR frames, different word-lengths and SNR values, and using (a) bilinear and (b) bicubic interpolation as the initial guess. (c, d, e) RMSE values obtained for the given word-lengths and the indicated numbers of LR frames.

in Fig. 5.11(a,b), it is also observed here that, due to the good convergence properties of the algorithm, bilinear interpolation is preferable as its leads to similar results to bicubic interpolation but with lower computational cost.

The graphs of Fig. 5.15(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 (5.5) with (5.3), to demonstrate the increased robustness of (5.5) with respect to noise.

Robust SR in the Presence of Noise

The graphs of Fig. 5.16 demonstrate a comparison between the RMSE values obtained by (5.3) (Fig. 5.16(a-d)) and (5.5) (Fig. 5.16(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 different data bit-widths, number of LR frames, and SNRs, as indicated in the figure.

As explained in Section 5.4, the original IBP algorithm of (5.3) can be considered as a special case of (5.5) 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 (5.5) is simplified to (5.3). The results reported in [ABCC08] indicate that in

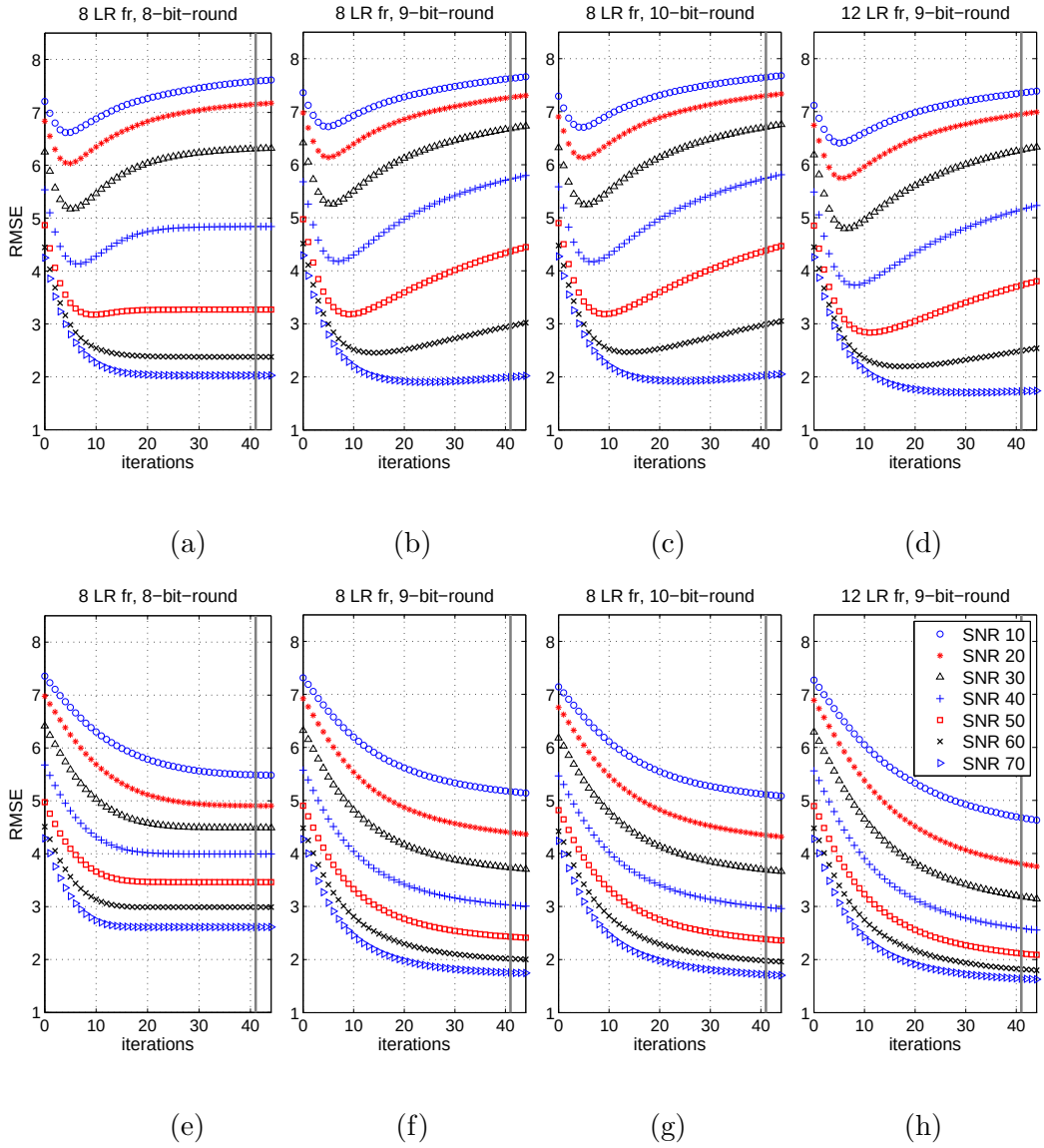


Figure 5.16: Comparing the convergence properties of (5.3) (a-d) and (5.5) (e-h), for the *cars* sequence. By accounting for the noise statistics in the LR frames, (5.5) gives robustness with respect to noise.

the noise-free situation (5.3) minimizes the error between the "ground-truth" frame and the reconstructed output, as the iterations proceed. Let us call this error *reconstruction error*. However, in the presence of noise, (5.3) no longer minimizes the reconstruction error, as Figures 5.16(a-d) demonstrate. As explained in Section 5.4, 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 (5.2) but *not* of the reconstruction error. For low noise levels, such as $SNR = 70$ dB (Figures 5.16(a-d)), (5.3) renders a behavior close to that reported in [ABCC08]. However, when the noise level increases, *i.e.* the SNR decreases, the reconstruction error diverges more from the ideal behavior of the noise-free case.

The above problems are resolved by employing (5.5) instead of (5.3). The value of $\delta_0(k)$ of (5.5b) is actually the noise standard deviation σ_n of (5.9), and is calculated based on the given SNR and the image statistics according to (5.9). Figures 5.16(e-h) demonstrate the good convergence properties of the algorithm of (5.5). The reconstruction error is now always minimized, regardless of the given system parameters. Two main trends, affecting the rate of convergence, can be observed: (i) 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 Fig. 5.16(f) with that of Fig. 5.16(h). (ii) 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 Fig. 5.16(e)

with Fig. 5.16(f).

The above observations can be interpreted as follows. For low SNRs, the reliability of the LR samples is low. Therefore, additional LR frames further contribute in the reconstruction on the HR grid, as long as the level of the related non-rigid deformations remains low (Section 5.4). 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 more the reconstruction quality 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 Fig. 5.16(f) to that of Fig. 5.16(g), the decrease of the error is trivial, indicating that a tenth bit is redundant. Thus, 9 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 Fig. 5.16(f), after the number of iterations giving real-time performance, is visualized in Fig. 5.17. The part of the frame shown in Fig. 5.17 corresponds to the "ground-truth" part of Fig. 5.14(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

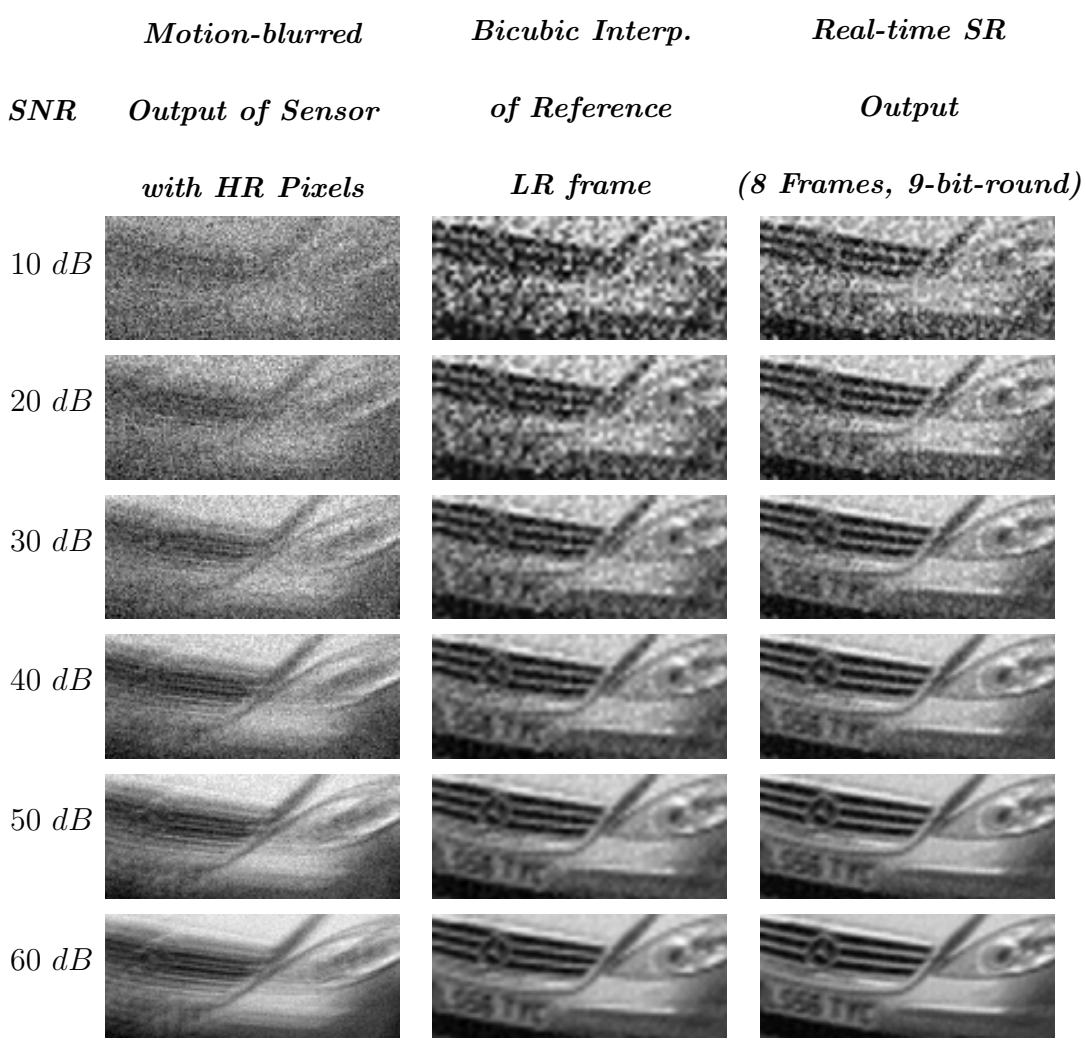


Figure 5.17: Reconstruction of *car* for different SNRs. The SR outputs visually demonstrate the higher quality obtained by the implemented real-time algorithm, compared to the motion-blurred output of a traditional HR sensor and the bicubic interpolation of the output of an LR sensor.

SR output, where HR spatial resolution and LR temporal resolution are successfully combined.

Conclusions of the Performance Evaluation

In the above paragraphs, multiple design options and algorithmic parameters have been evaluated and discussed. The general conclusions deriving from this evaluation are the following:

1. Taking into account the noise statistics in the reconstruction process dramatically improves the convergence properties of the iterative process in the presence of noise, resulting in a more robust reconstruction algorithm (Fig. 5.16).

2. 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.

3. The minimum word-length for maximal performance increases as the number of iterations increases, thus as the frame size decreases (Table 5.1). The above word-length is that of the pixels propagated from one iteration of the IBP algorithm to the next.

4. Bilinear interpolation is preferable for initialization purposes, as its leads to similar results to bicubic interpolation but with lower computational cost.

5.7 Summary

In this chapter, the SR-based reconstruction is employed to compensate for the low spatial resolution of LR areas. In particular, the IBP algorithm is implemented on an FPGA, after being modified to account for the additive noise in the LR samples. The number of FPGA resources of the proposed architecture scales linearly with the number of LR frames that contribute in the reconstruction process.

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 of the system. 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 (Fig. 5.16).

Interesting observations derive from the evaluation process of the implemented algorithm of (5.5): 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 mainly affected for low SNRs, being noticeably decreased. On the other hand, for a given number of LR frames, using larger word-lengths mainly affects the reconstruction quality 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.

In the current chapter, the contribution of each LR pixel in SR reconstruction was modelled with a gaussian PSF shifted on the HR grid by the displacement corresponding to the particular LR frame (Fig. 5.2). For large motion magnitudes, the above model is inaccurate, since considerable extent of motion blur also appears in the LR samples rendering the LR PSF significantly non-isotropic. This would degrade the reconstruction quality of the SR block. In the following chapter, blur identification is employed to reveal the intra-LR-frame motion information, which is then incorporated in SR for the effective reconstruction of the output. Moreover, by incorporating intra-LR-frame motion information in the reconstruction process, the temporal resolution of the reconstructed output is no longer limited by that of the LR samples.

Chapter 6

Blur Identification and Super-Resolution with Non-Isotropic PSFs

6.1 Introduction

Super-resolution methods are largely affected by the accuracy of the estimated PSFs that are related to the input frames [BEN04]. When the frames are degraded by heavy motion blur, these PSFs are highly non-isotropic, which further complicates their estimation. The ill-posed nature of blur identification is normally addressed using the assumption of linear and uniform motion [YK97, TKW86, CY06, MJ07, CTE91, Can76]. However, in real-life systems, this may deviate significantly from the actual blur.

To address the above, this chapter proposes a joint blur identification and

classification scheme that classifies the underlying motion with respect to the validity of the linear and uniform motion assumption. The proposed scheme, which employs the autocorrelation-based blur identification framework [YK97], evaluates the validity of the linearity and uniformity assumption, while calculating the blur parameters. The joint blur identification and classification scheme is combined with the real-time reconfiguration property of an adaptive image sensor. If the assumption is invalid for a given motion region, the sensor is locally reconfigured to larger pixels that produce higher frame-rate samples with reduced motion blur. Once the appropriate configuration that produces a valid assumption is applied, highly accurate PSFs are estimated, improving the SR reconstruction quality.

An efficient real-time hardware architecture of the proposed method is presented, which is implemented on Field Programmable Gate Array (FPGA). The implementation requirements are presented for different sets of parameters, and the system performance is evaluated under different noise levels. Moreover, a software evaluation of the SR reconstruction quality for various PSF assumptions is presented for a large set of motion types.

The chapter is organized as follows. Section 6.2 introduces the blur identification problem and its relation to subsequent restoration, to present the motivation for the work presented in this chapter. Section 6.3 presents the detailed algorithm of the proposed joint blur identification and classification scheme. Section 6.4 explains how the scheme is effectively incorporated into a video enhancement system that is based on an adaptive image sensor. In Section 6.5,

the effect of intra-frame motion on the registration of LR frames is dealt with. Then, Section 6.6 presents a software evaluation of the SR reconstruction quality for different PSF assumptions and various motion types, and demonstrates that accounting for intra-frame motion significantly improves the reconstruction quality. The hardware architecture of the proposed blur identification and classification scheme is described in Section 6.7. Section 6.8 presents the experimental results. Specifically, Section 6.8.1 discusses the throughput and hardware requirements of the proposed architecture for different sets of parameters. Finally, Section 6.8.2 evaluates the system performance for various parameters and different levels of noise.

Parts of the work discussed in this chapter have been presented in [ABC09] and [ABC].

6.2 Blur Identification and Subsequent Restoration

For the image reconstruction process to be effective, the PSFs of the input frames should be accurately estimated, since they contain information regarding the contribution of each frame in the reconstruction [SPBDK01, IP91, ABCC09]. In Chapter 5, the intra-frame motion of the LR frames is considered to be negligible. Therefore, traditional SR is employed, where the uniqueness of the LR channel that is associated to each LR frame is based solely on the corresponding subpixel shift. In that case, the gaussian PSF

assumption is adequate. Thus, once the subpixel shifts of the LR frames are computed by the preceding registration block, all the information required for the actual reconstruction is available.

For large motion magnitudes, considerable extent of motion blur appears in the LR samples. Such intra-frame motions render inaccurate the gaussian PSF assumption that is employed by the traditional approaches of [SPBDK01, IP91, ABCC09]. In that case, the corresponding PSFs are no longer isotropic. Thus, for different LR frames, the PSF shapes and weights vary. In that case, the intra-frame motion information of each LR sample should be identified and included in the reconstruction process. Therefore, apart from registration, blur identification should also precede the actual reconstruction. As will be demonstrated in the sections that follow, the quality of the reconstructed output is highly dependent on the accuracy of the identified blurring function.

Blur identification is a critical task when motion blur degrades the quality of the output frames. In real-time video capturing, fast moving objects produce heavy blur, locally on the region of the frame that the motion spans, and are thus related to significantly non-isotropic PSFs [BEN04], whose identification is a highly ill-posed problem [YK97]. The ill-posed nature of the blur identification problem and the required high computational cost [TKW86, CY06] pose a bottleneck in the overall system performance, in particular when real-time applications are targeted. In the literature, the blur identification process is simplified by adopting the assumption that the motion PSF is linear and uniform [YK97, TKW86, CY06, MJ07, CTE91, Can76]. This reduces the iden-

tification task to the estimation of two parameters, namely the direction and the extent of the underlying motion, thus resolving the ill-posed nature of the problem and decreasing the related computational load. However, in real-life systems, the actual blurring function might deviate significantly from an ideal linear and uniform PSF [BEN04], rendering the above assumption unrealistic. Moreover, traditional software solutions are inadequate when the given system should operate in real-time, *i.e.* at least 25 fps for the given image size.

The current chapter addresses the problems and limitations that are described in the previous paragraphs by proposing a joint blur identification and validation scheme, which both estimates the motion parameters and validates the initial linearity and uniformity assumption. The scheme employs the simple linearity and uniformity assumption, but also identifies cases where the initial assumption is invalid and thus the PSF estimation is inaccurate. In this manner, the blur identification problem is tackled in a more robust manner, while the required computational cost remains low, as will be demonstrated in the sections that follow. The proposed system employs the autocorrelation framework for blur identification [YK97] and is implemented on FPGA, in order to target real-time operation. By exploiting the parallelism, pipelining, and data reuse possibilities offered by FPGA, high throughput is achieved that meets the strict performance constraints of real-time applications.

The proposed joint blur identification and validation scheme is independent from the subsequent reconstruction process. Therefore, the scheme can be used for general purpose blur identification and can be combined with any

restoration method [KH96, SPBDK01, IP91]. Cases of negative classification could be tackled in various ways, such as fitting a second order polynomial to the blurred region, which is included in the future work, or employing an adaptive image sensor [Fov08, CDT06], which is the approach that has been employed in this chapter.

The latter is the author's motivation for incorporating the assumption validation in the blur identification process and is, therefore, thoroughly investigated in this chapter. The interaction with the adaptive sensor is based on the following concept. When the initial linear and uniform motion assumption is found invalid, the adaptive image sensor is configured to larger pixel sizes that produce higher frame-rate samples with reduced motion blur. In these samples, the motion trajectory is broken into shorter, more linear parts. Once the appropriate pixel size is employed, for which the assumption is found valid, the blurring function is accurately estimated, as will be demonstrated in the sections that follow.

By incorporating in the reconstruction process accurate estimations of the PSFs that correspond to the LR frames, the contribution of each frame is estimated with high accuracy, as will be shown in the following sections, and the LR data are correctly fused together. Moreover, in that case, the temporal resolution of the reconstructed output is no longer limited by that of the LR samples, since the intra-frame motion effect is also incorporated in the reconstruction process. This is different to the case where simple gaussian PSFs that contain no intra-frame motion information are employed, as was the case

in Chapters 4 and 5.

In summary, the contributions of this chapter are the following:

1. A joint blur identification and validation method is proposed. The method utilizes the autocorrelation-based blur identification framework of [YK97].

2. A hardware architecture of the blur identification and validation scheme is presented. The proposed architecture is implemented on an FPGA and its performance is evaluated. To the best of the author's knowledge, this is the first hardware approach to the blur identification problem that is reported in the literature.

3. The interaction of an adaptive image sensor with a joint blur identification and validation scheme is proposed, which increases the accuracy in the estimation of the PSFs related to the SR inputs.

4. The complete algorithm for real-time video restoration is presented, including both the sensor configuration framework and the processing of the captured data. The system performance is evaluated for linear and non-linear motion types under various noise levels.

6.3 Description of the Algorithm

The proposed joint blur identification and validation scheme utilizes the spatial domain blur identification framework of [YK97]. The decision of using

a spatial domain instead of a frequency domain method is based on both algorithmic and hardware-related criteria. Specifically, frequency domain blur identification methods [CTE91, Can76, MJ07] are restricted to blurring functions that exhibit a periodic pattern of spectral zeros, which is not true in various cases [YK97]. When it comes to hardware, as has been already mentioned in Chapter 5, a spatial method reduces the required hardware cost by avoiding the transformations between the two domains. Moreover, spatial domain analysis 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 blur identification methods [TKW86, YK97, CY06] the autocorrelation-based method of [YK97] was selected due to its potential for maximum parallelism and data reuse, as it will be demonstrated in Section 6.7.

The detailed algorithm of the proposed joint blur identification and validation scheme is presented next. The notation that is used in the presentation of the algorithm is summarized in Table 6.1.

Notation	Meaning
c	The classification flag.
θ	The estimated motion direction.
L	The estimated motion extent.
f	The input image.
Δ_ϕ	The directional image derivative operator in direction ϕ.
$I(\phi)$	The normalized total intensity of $\Delta_\phi(f)$.
s	The step in degrees for calculating $\Delta_\phi(f)$ and $I(\phi)$.
g	The horizontal derivative of the derivative vertical to θ.
M	The number of rows of g rotated by θ.
K	The number of columns of g rotated by θ minus 1.
$m_i[n]$	The i^{th} line of g that derives after interpolation along θ, where $m_i[n] = 0$ for $n \notin [0, K]$.
R_i	The discrete set of autocorrelation coefficients for $m_i[n]$.
$\bar{R}$	The mean of the discrete autocorrelation coefficients of the M lines of g along θ.
μ	The mask for isolating the moving object from the background.
p_1	The threshold for identifying considerably small values of L that are due to motion nonlinearities.
P, N	The number of dominant positive and negative lobes in $\bar{R}$.
p_2	The threshold for $I(\theta)$.

Table 6.1: The notation that is used in the proposed blur identification and validation algorithm.

Using the notation of Table 6.1, the proposed blur identification and validation algorithm is formulated as follows.

Inputs: f, μ, s, p_1, p_2

Outputs: c, θ, L

// ————————- blur identification: ————————- //

for $\phi = 0^o : s : 180^o - s$ **do**

  calculate $\Delta_\phi(f)$ and $I(\phi)$

find $\theta \in [0^o, 180^o)$: $I(\theta) = \min(I)$

$g = \Delta_\theta(\Delta_{\theta+90^o}(f))$

$g = g \cdot \mu$

for $i = 1 : M$ **do**

  for $k = -K : K$ **do**

    $R_i[k] = \sum_{n=-K}^{K} m_i[n] m_i[n-k]$

$\bar{R} = (\sum_{j=1}^{M} R_j)/M$

find $L \in [0, K]$: $\bar{R}[L] = min(\bar{R})$

// ————————- classification: ———————— //

if $L < p_1$ **then**

  $c \leftarrow 0$; **return** c

else

  count positive(P) and negative(N) dominant lobes in $\bar{R}$

  if $P \neq 1$ or $N \neq 2$ **then**

    $c \leftarrow 0$; **return** c

  else if $I(\theta) > p_2$ **then**

20: $c \leftarrow 0$; **return** c

21: **else**

22: $c \leftarrow 1$; **return** c, θ, L

6.3.1 Blur Identification

The blur identification part, in the above algorithm, employs the framework of [YK97], which is based on the calculation of the mean discrete autocorrelation function (ACF) along the motion direction.

The discrete autocorrelation function is defined as follows.

Definition 4. *Let x_n be a discrete signal. Then the discrete autocorrelation R for signal x_n at lag j is:*

$$R(j) = \sum_n x_n \overline{x}_{n-j} \tag{6.1}$$

This subsection discusses the approach of [YK97], and how this was modified and extended to achieve local, computationally efficient, and robust to noise blur identification.

In lines 2-11 of the blur identification and validation algorithm that is presented above, the linear motion parameters (θ, L) are identified employing the autocorrelation-based framework of [YK97]. The method of [YK97] is based on the fact that along the motion direction, image smoothness is higher and pixels are correlated. As demonstrated in the algorithm, a series of filtering operations precedes the autocorrelation calculations. First, the image derivatives are calculated in various directions, and the total intensities of the derivatives

are computed (lines 2-3).

Definition 5. *Let $\Delta_\phi(f)$ be the directional derivative of image f, and let U and V denote the number of rows and columns of image $\Delta_\phi(f)$. Then the total intensity $I(\phi)$ of $\Delta_\phi(f)$ is:*

$$I(\phi) = \sum_{i=1}^{U-1} \sum_{j=1}^{V-1} |\Delta_\phi(f(i,j))| \tag{6.2}$$

The minimum of the total intensities, along directions $[0^o, 180^o)$ with step s, indicates the motion direction θ. An optimized derivative g, which is 'optimized' in the sense that it is, to a large extent, 'freed' from object properties and thus mainly contains motion properties, is then generated as follows. First, the derivative of the input frame is calculated in the direction vertical to θ, which removes object-related image properties and emphasizes motion-related properties. On this output, the derivative in direction θ is then applied (line 5), resulting to image g. Image g is then traversed along direction θ, and the autocorrelation coefficients are calculated on the lines of pixels that are interpolated along that direction (lines 7-9). As will be explained in the sections that follow, nearest-neighbor interpolation has been employed for this purpose. Finally, the mean of these autocorrelation outputs is calculated (line 10), and the lag of the minimum mean autocorrelation coefficient gives the motion extent along θ (line 11).

In [YK97], the directional filters, for the calculation of $\Delta_\phi(f)$ and g, derive from the [-1 1] kernel rotated and interpolated to fit each direction. The small support of these filters renders them particularly sensitive to additive

noise [YK97]. This reduces the robustness of the method of [YK97]. Thus, for $SNR < 35$ dB, [YK97] suggests avoiding the calculation of g and computing the ACF directly on $\Delta_\theta(f)$. In that case, the ACF is significantly affected by the object properties. To resolve the above, Sobel filters have been used in this work for the derivative calculations, replacing the [-1 1] kernels of [YK97]. Moreover, for $SNR < 40$ dB, the system robustness is further increased by applying an adaptive Wiener filter [Lim90] on the input frame.

In [YK97], the frames that are considered involve global motions, which normally would derive from camera motion. To deal with object motion as well, a masking stage should be included (line 6). The mask should be applied after the optimized derivative g has been calculated. In this manner, all derivatives are computed on the unmasked object, so as to preserve the image smoothness and avoid wrong derivative values, which would otherwise be calculated between the border of the object and the zero pixels that would surround it after the masking. The mask is generated by using median filtering and by considering a pair of frames before and after the current frame. This avoids the blending regions at the borders of the object and minimizes the interference from the static background [BEN04].

The computational load of the blur identification process can be significantly reduced by considering fewer directions for calculating $\Delta_\phi(f)$ and also fewer ACF lags. Moreover, by rotating f by θ and then calculating g on the rotated frame, only two filters are required for the generation of g: a vertical and a horizontal filter. Such issues are further discussed in Section 6.7 and

Section 6.8.1, where different design options are evaluated with respect to system's area and throughput, while Section 6.8.2 discusses their impact on the system's performance.

6.3.2 Classification

Linear and uniform motions can be identified using two types of data: the normalized total intensities $I(\phi)$ and the ACF coefficients, *i.e.* the coefficients of $\bar{R}$. In [YK97], the above data are utilized in blur identification. This section demonstrates how these data can also be utilized for motion classification purposes, with respect to whether the linearity and uniformity assumption is valid. To demonstrate their properties for different motion types, the above data are calculated on the motion blurred frames deriving after the application of different motion PSFs on the 240×320 ground-truth image of Fig. 6.1. Four motion PSFs are applied, each associated with a different motion type: (i) a linear and uniform PSF with an extent of 15 pixels, which is applied in the vertical direction (Fig. 6.2(a)), (ii) the nonuniform linear PSF of Fig. 6.2(b), which is also vertically applied, (iii) the nonlinear and uniform PSF of Fig. 6.2(c), and (iv) the nonlinear and nonuniform PSF of Fig. 6.2(d). The experiments demonstrated in Fig. 6.3 are executed after adding Gaussian noise that results to an SNR of $50dB$, since $50dB$ is the typical SNR of digital cameras. The intensities $I(\phi)$ and coefficients $\bar{R}$, which are calculated for each blurred frame, are illustrated in the graphs of Fig. 6.3.

The mean ACF along the motion direction normally has a particular shape

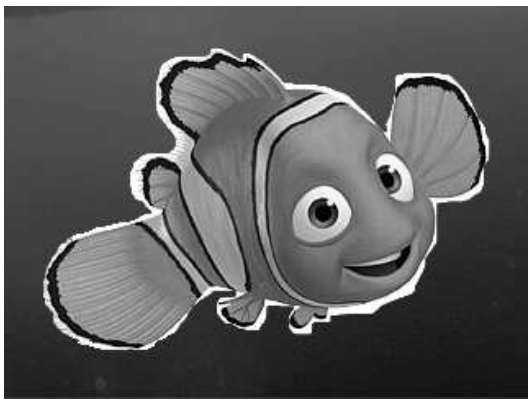
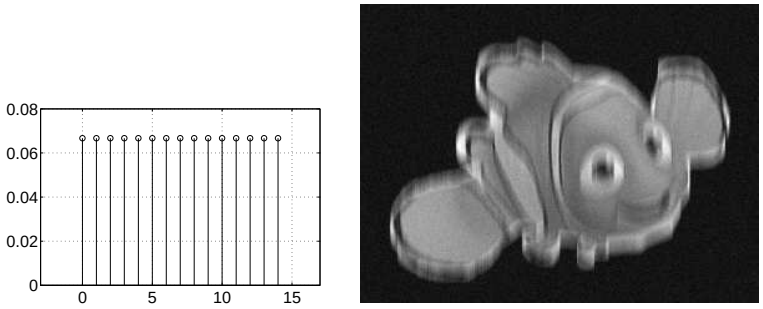


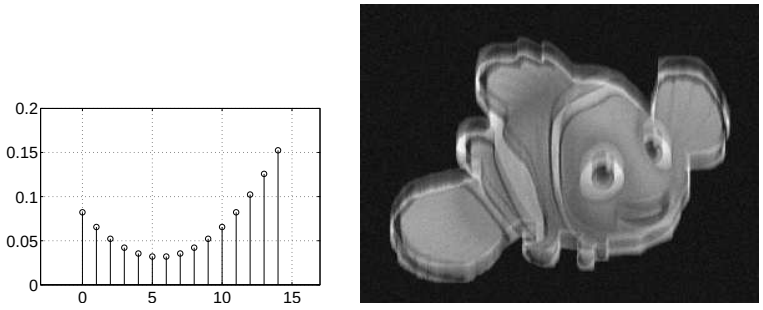
Figure 6.1: The ground-truth image.

for the case of linear and uniform motion, which is similar to that of Fig. 6.3(a). Specifically, it contains three dominant lobes: a positive lobe at lag 0 and two negative lobes that are symmetrical with respect to the y axis. The lag of the minimum coefficients of the two symmetrical negative lobes indicates the motion extent. In Fig. 6.3(a), for example, the minimum ACF coefficients lie at lag ± 15. Therefore, the estimated motion extent is equal to 15. The relation between the minimum ACF coefficients and the linear motion extent is explained in detail in [YK97]. Experiments have shown that the more the given ACF diverges from the form that is described above, the more does the corresponding motion diverge from the ideal linear and uniform case. Therefore, negative classification occurs in the following cases:

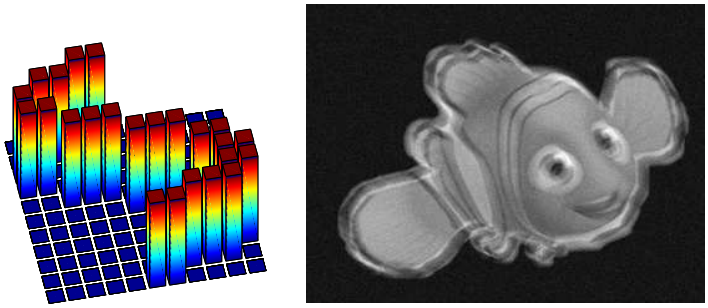
1. The total numbers of positive and negative main lobes is different than those stated above, *i.e.* *one* positive lobe at lag 0 and *two* symmetrical negative lobes. This is the case for high-frequency temporal vibrations that lead to nonuniform motion, where more lobes are formed.



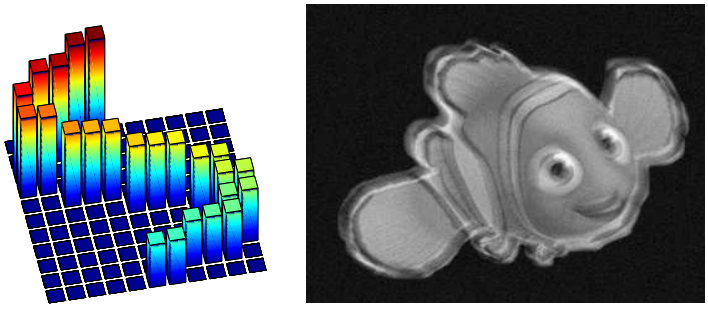
(a) Linear and uniform motion.



(b) Linear and nonuniform motion.



(c) Nonlinear and uniform motion.

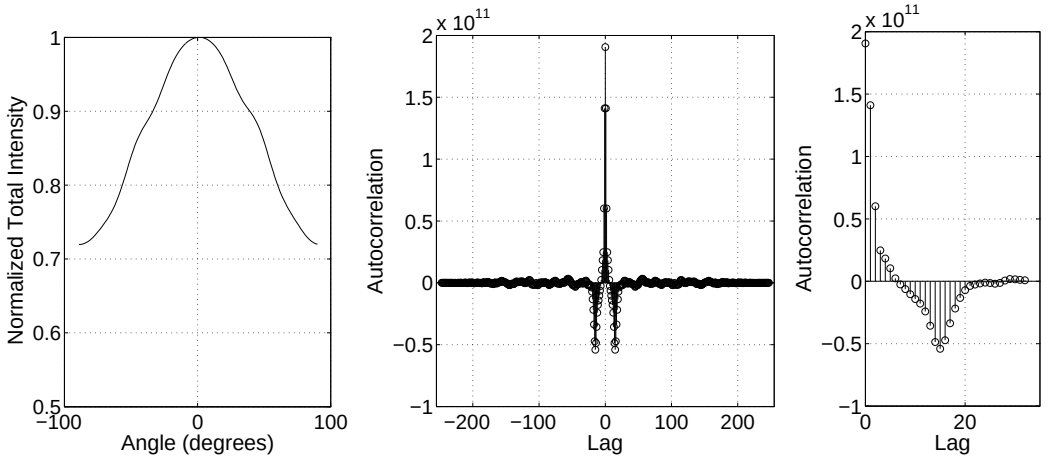


(d) Nonlinear and nonuniform motion.

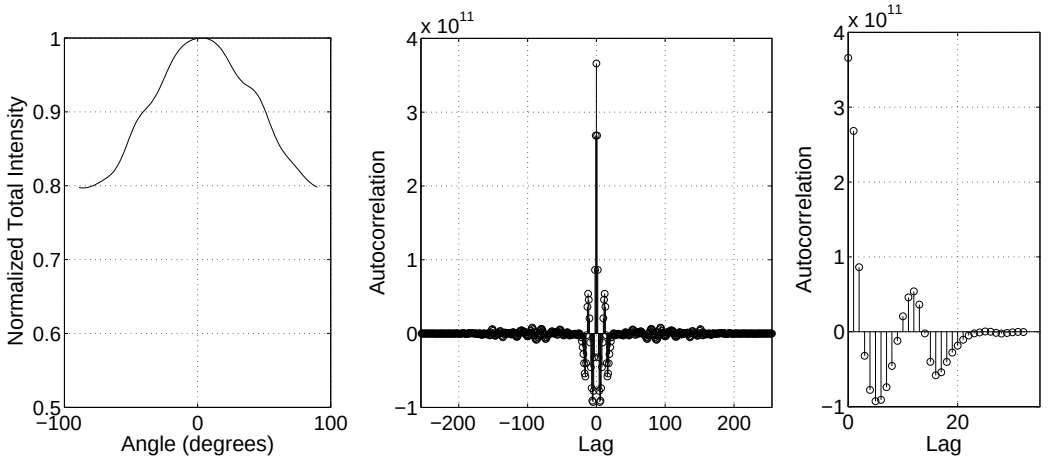
Figure 6.2: Various motion PSFs and the corresponding motion-blurred images.

Specifically, in that case, at each side of lag 0, there is a positive lobe surrounded by two negative lobes, as can be observed in Fig. 6.3(b). Nonlinear motions, both uniform and nonuniform, may as well generate multiple irregular lobes, as illustrated in Fig. 6.3(c) and Fig. 6.3(d).

2. The lag of the minimum ACF coefficient, *i.e.* the position of the mini-

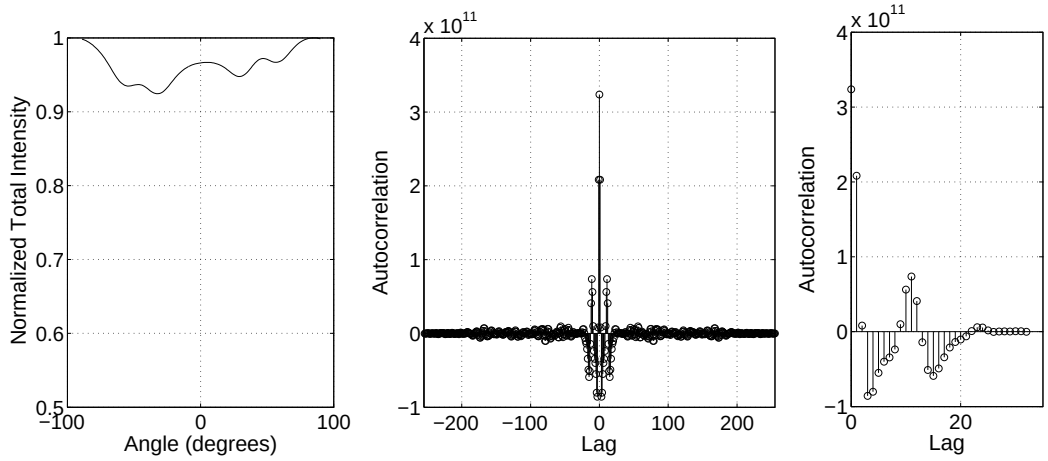


(a) Linear and uniform motion.

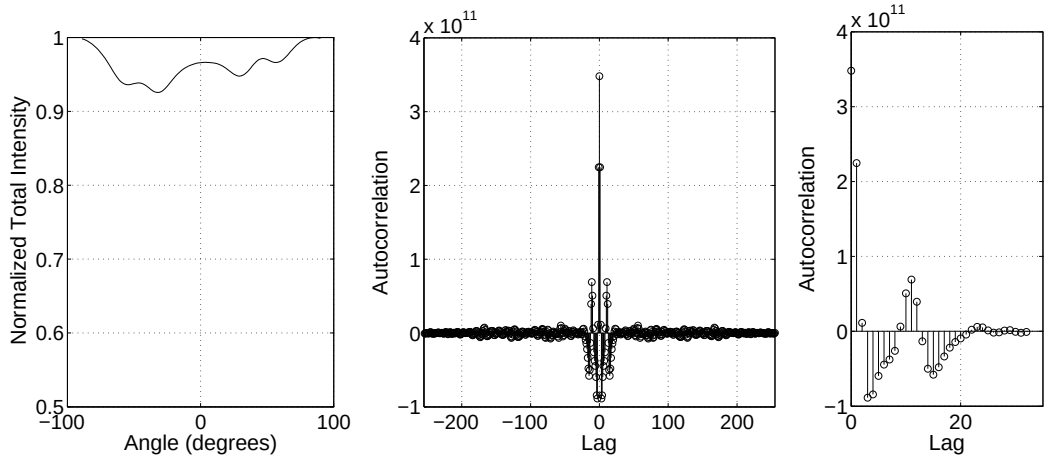


(b) Linear and nonuniform motion.

Figure 6.3: Calculated normalized total intensities $I(\phi)$ and autocorrelation coefficients $\bar{R}$ for the indicated motion types. The figure continues at the next page.



(c) Nonlinear and uniform motion.



(d) Nonlinear and nonuniform motion.

Figure 6.3: Calculated normalized total intensities $I(\phi)$ and autocorrelation coefficients $\bar{R}$ for the indicated motion types.

mum ACF coefficient on the x axis of the ACF graph, is very close to lag 0. This indicates a motion extent that is too small to reflect the actual PSF. This is normally observed in the case of nonlinear motions, both uniform and nonuniform, as shown in Fig. 6.3(c) and Fig. 6.3(d).

When a frame is linearly blurred along a particular direction, there is a clear minimum in the total intensity graph, which indicates the motion direction (Fig. 6.3(a,b)). In the case of nonlinear motions, this minimum is not clear (Fig. 6.3(c,d)). Therefore, for nonlinear motions, the minimum *normalized* total intensity is significantly higher (Fig. 6.3(c,d)) than in the case of linear motions (Fig. 6.3(a,b)), and this value is used in the classification.

In addition to the above, linear and uniform motions of very large extents may as well be undesired for the given specifications, as they increase the computational cost and required precision for the subsequent restoration block. By setting a maximum lag in the ACF computations, only linear motions whose extent is smaller than that lag produce negative ACF lobes. Larger motions are automatically classified as negative.

6.3.3 Inter-frame and Intra-frame Motion

Due to the continuity of motion in subsequent frames, for a linearly moving object, the intra-frame motion PSFs are expected to be consistent, both in extent and direction, with the inter-frame motion vectors. Thus, if the intra-frame and inter-frame motion parameters are not consistent, the linear-

ity assumption is probably invalid. The above can be incorporated in the validation scheme as a further check for motion linearity, with the extra computational cost required for registration. When the subsequent reconstruction block is SR-based, registration is executed anyway for the SR method to be applied [SPBDK01, IP91, ABCC09], and thus the above check can be implemented with no additional computational cost.

6.4 Accounting for Intra-frame Motion in a System with an Adaptive Image Sensor

This section explains how the reconfigurability of the sensor is combined with the blur identification and validation scheme (BIV) to improve the reconstructed output.

The proposed system utilizes a blur detection block, which is based on the comparison of the strongest edges of the object with those of the background, to identify cases where the current configuration produces samples with negligible motion blur [TLZZ04]. In such cases, an isotropic Gaussian PSF can be employed to associate each LR pixel to the HR pixels of the underlying HR grid [ABCC09]. In all other cases, the Gaussian assumption is inadequate, and blur identification is required to estimate the motion parameters. Thus the BIV scheme is executed. If BIV finds the linear and uniform motion assumption invalid, the pixel size increases in the next sensor reconfiguration, to produce samples with reduced and more linear motion blur. In this man-

ner, the initial non-linear and/or non-uniform motion trajectory is fragmented into shorter, more linear parts. If BIV finds the initial motion assumption valid, accurate linear PSFs can be estimated, and the pixel size thus remains constant.

The pixel size of the adaptive sensor depends on the outputs of *blur detection* and *BIV* blocks, as described above. These blocks comprise a classifier, whose binary output c determines the pixel size in the next sensor reconfiguration: If $c = 1$, the pixel size remains constant, as it allows an accurate estimation of the PSF, employing either the Gaussian or the linear and uniform motion assumption. If $c = 0$, the pixel size increases in the next reconfiguration. A mechanism that reduces the pixel size every N frames can be accommodated, in order to obtain high-resolution frames.

For linear motion, the non-isotropic PSFs are expected to be consistent with the inter-frame motion vectors. In case of inconsistency, invalidity of the initial linearity assumption is indicated. The above can be used as an additional validity check, and is thus incorporated into the classifier that is described above, giving $c = 0$ in the case of inconsistency.

Ideally, the adaptive sensor would be reconfigured at every new HR integration. In reality, reconfiguration is sparser, depending on the technology of the given sensor. The proposed video enhancement system operates as follows. Moving objects are detected with a rough motion estimation on the HR grid, as indicated in Fig. 6.4. For every moving object, the control of Fig. 6.4 is employed, where s denotes the LR pixel size, with $s = 2$ corresponding to the

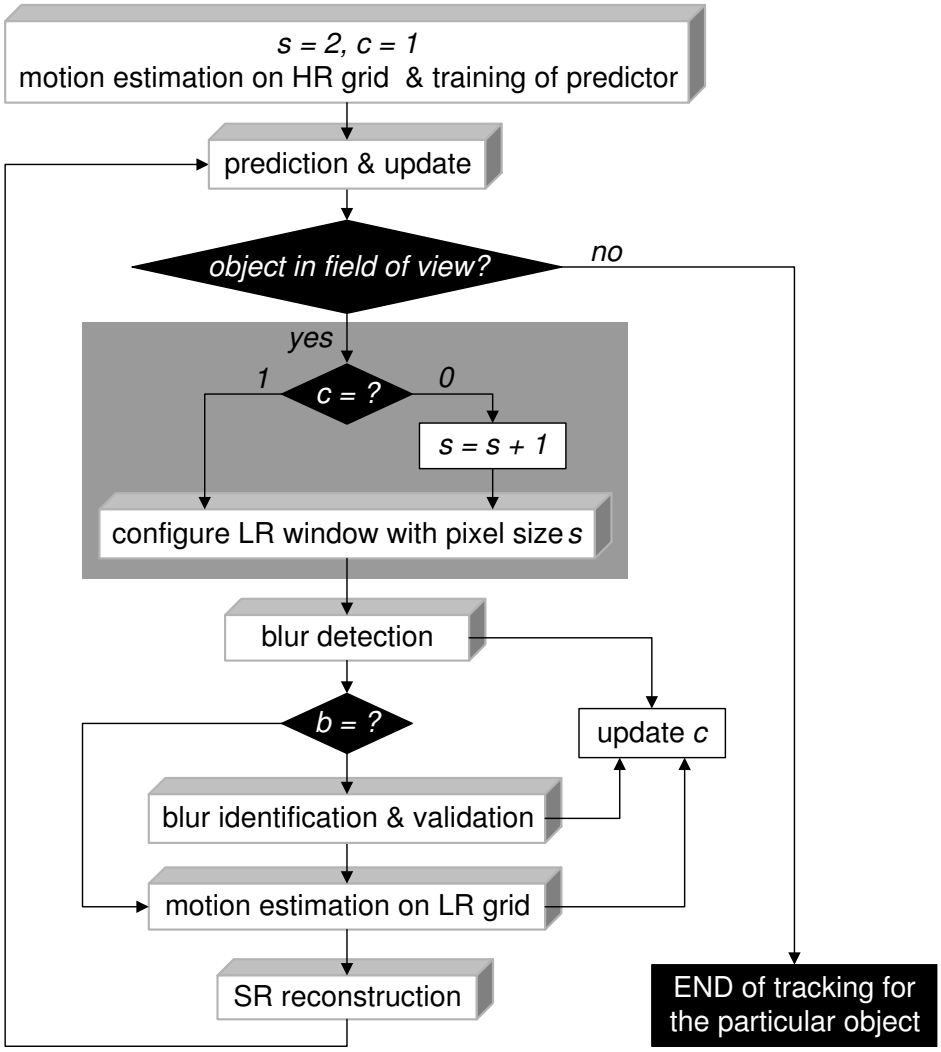


Figure 6.4: The algorithm of the system operation for a dynamic region.

smallest LR pixel size, and c denotes the binary output of the classifier that determines the change in the pixel size in the next reconfiguration. Moreover, b denotes the binary output of the *blur detection* block; $b = 0$ indicates negligible motion blur, for which the Gaussian assumption is employed and BIV is skipped, while $b = 1$ indicates that BIV is required. The part of Fig. 6.4 included in the gray rectangle is executed only in those HR integration intervals when the sensor is reconfigured. A Kalman filter predictor is employed to determine the position of the object in the next HR integration. When the sensor reconfiguration occurs, that position determines the location of the LR area, while its pixel size depends on the validity of the linearity assumption in the last HR integration, indicated by the value of c. For $c = 0$ the pixel size increases, whereas for $c = 1$ it remains the same. For each LR area, an LR sequence with reduced blur is produced, and the PSFs are estimated, based on the outputs of the *blur detection* and *BIV* blocks, as described in the previous paragraphs. The LR samples are registered using a *motion estimation* block, and the corresponding PSFs are used by the *SR* block that executes the reconstruction on the HR grid [ABCC09]. This produces for each LR area an output with high resolution both in space and time, and thus motion deblurring is locally executed on the dynamic regions of the scene. At every new HR integration, the control starts at the second block of Fig. 6.4. The loop ends when the particular object exits the field of view.

6.5 Registration of Frames Blurred with Different PSFs

The ground-truth features are blurred in each LR frame by the frame's PSF. Since different frames have different PSFs, the resulting features are also different. This decreases the registration accuracy, and thus affects subsequent reconstruction. The obvious solution would be to apply deconvolution on each frame before registration. However, this would introduce deconvolution-related errors and significantly increase the computational cost of a hardware implementation. A more efficient approach is thus proposed.

Let f_1 and f_2 denote two frames, related to the ground-truth g as follows:

$$f_1 = g \otimes r_1 \tag{6.3}$$

$$f_2 = g \otimes r_2 \tag{6.4}$$

where r_1 and r_2 denote the corresponding PSFs. If the PSFs are cross-applied on f_1 and f_2 as follows:

$$f_1{}' = f_1 \otimes r_2 \tag{6.5}$$

$$f_2{}' = f_2 \otimes r_1 \tag{6.6}$$

then substituting in Equations 6.5 and 6.6 with Equations 6.3 and 6.4, it

derives that

$$f_1{}' = g \otimes r_1 \otimes r_2 \tag{6.7}$$

$$f_2{}' = g \otimes r_2 \otimes r_1 \tag{6.8}$$

Thus, both $f_1{}'$ and $f_2{}'$ are related to g with the same PSF, which equals $r_1 \otimes r_2$. Therefore, instead of applying registration on f_1 and f_2, registration is applied on $f_1{}'$ and $f_2{}'$ as in those frames the blurred features are the same.

6.6 The Effect of Blur Identification and Validation on SR Reconstruction Quality

This subsection investigates how the quality of the reconstructed output is affected by different sensor configurations and PSF assumptions, in order to demonstrate the benefits from the utilization of joint blur identification and validation prior to SR.

Different parameters are employed, including the type of PSF assumption, the LR pixel size, and the noise level. White Gaussian noise is applied, with SNRs ranging from 10 to 50 dB. The iterative SR approach of [ABCC09] is used, and 30 iterations are executed for each estimation. To exclude any evaluation errors due to the blending of the object with the background, SR is applied on the isolated foreground objects. The number of frames produced during HR integration for each pixel size is subject to the space-time trade-

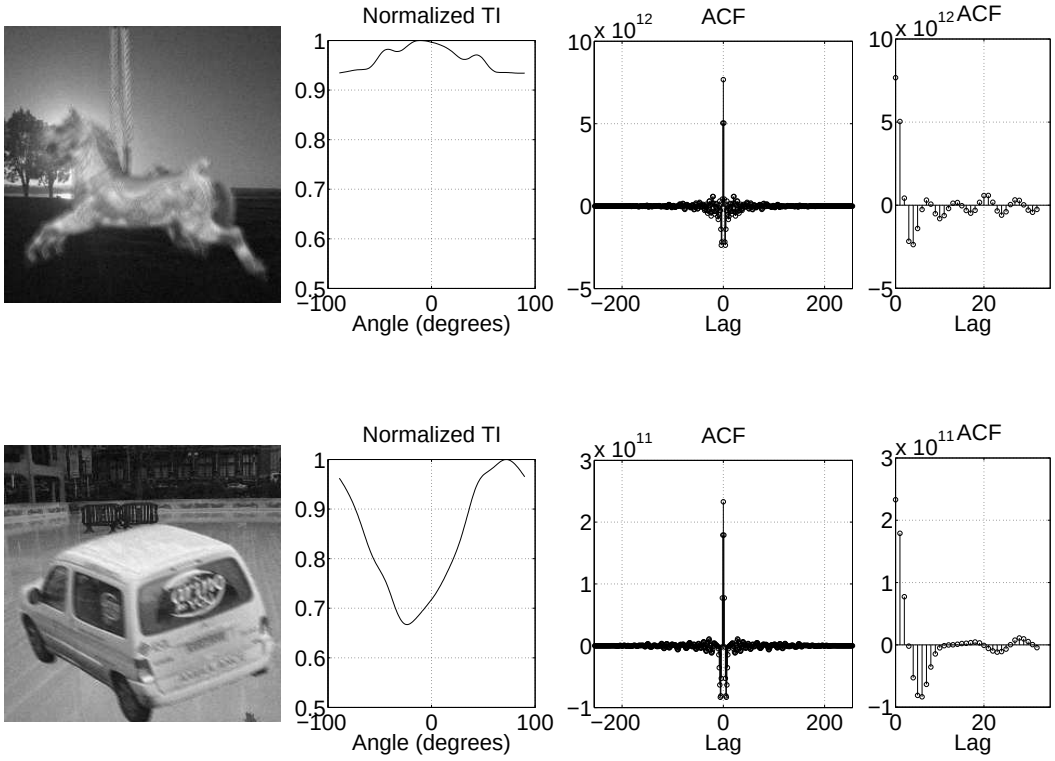


Figure 6.5: TI and ACF for *carousel* (top) and *ambulance* (bottom row). On the right, a detail of the ACF is presented, for lags 0 to 33.

off [ABC08]. If this number is k for the current configuration, k additional neighboring frames are used in SR, for increased robustness [ABCC09].

Fig. 6.5 shows the TI and ACF outputs of BIV for two moving objects with non-negligible motion blur ($b = 1$). Both cases employ 2×2 configuration and 256×256 LR resolution. The frame on the left is one of the 4 samples generated during HR integration at 50 dB SNR. According to the validation criteria described in Sect. 6.3, the TI and ACF outputs of Fig. 6.5 indicate that for the 2×2 configuration the linearity and uniformity assumption is valid for *ambulance*, and invalid for *carousel*. Indeed, the actual motion of *carousel*, is not sufficiently linear for a 2×2 configuration, as the trajectory shows in

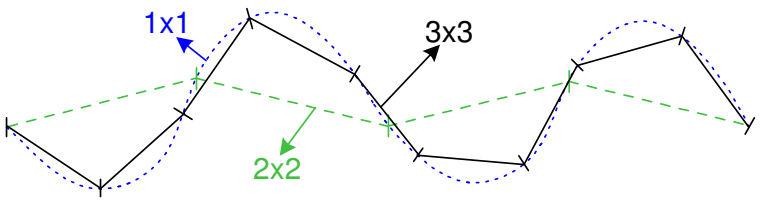


Figure 6.6: The intra-frame motion of the time samples for the three configurations.

1×1
$50\ dB$

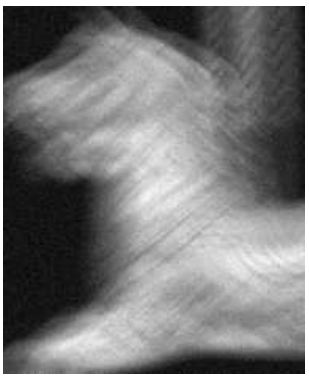


Figure 6.7: The blurred output for configuration 1×1.

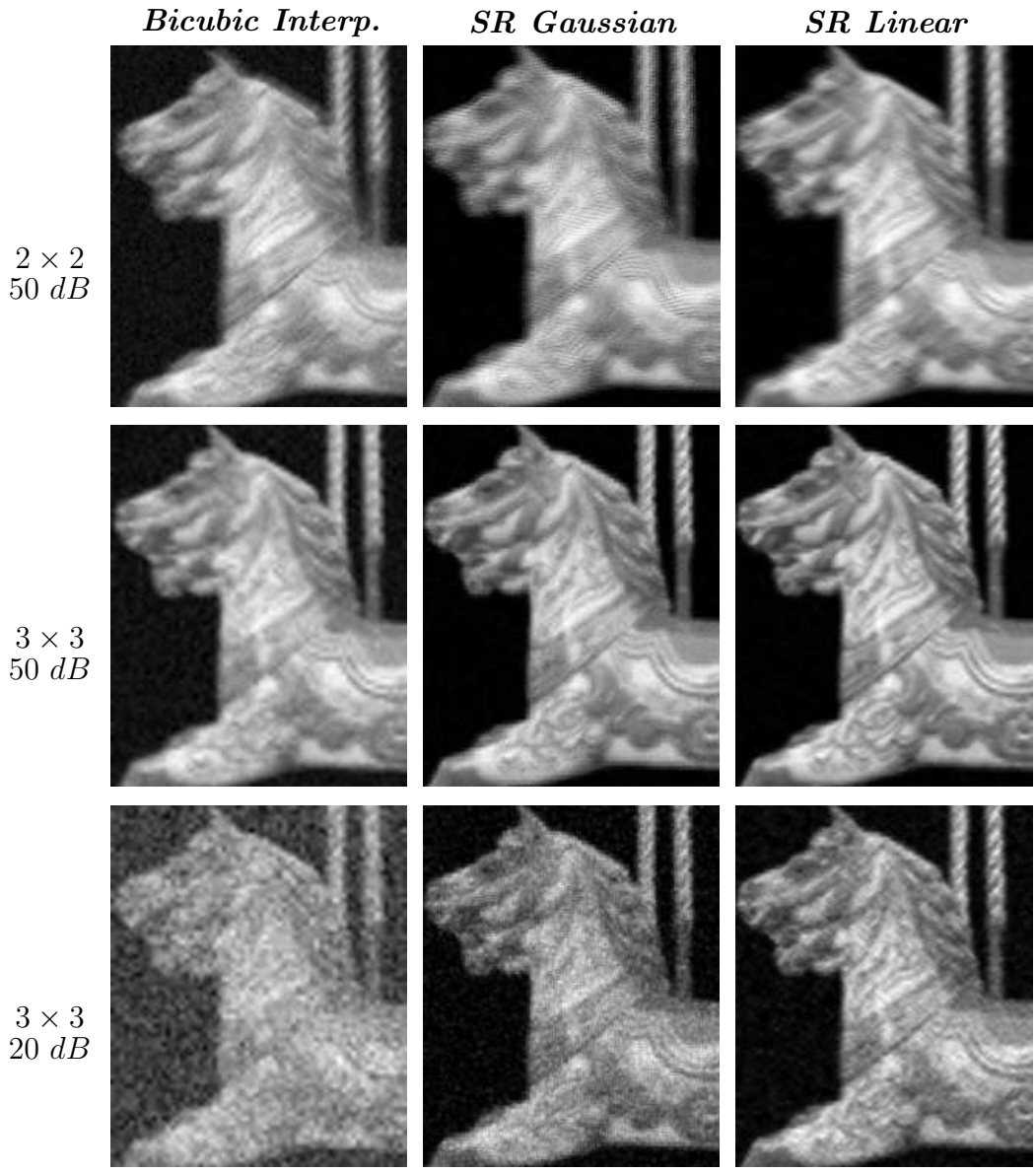


Figure 6.8: The reconstructed outputs for the SNR values and reconstruction methods that are indicated in the figure.

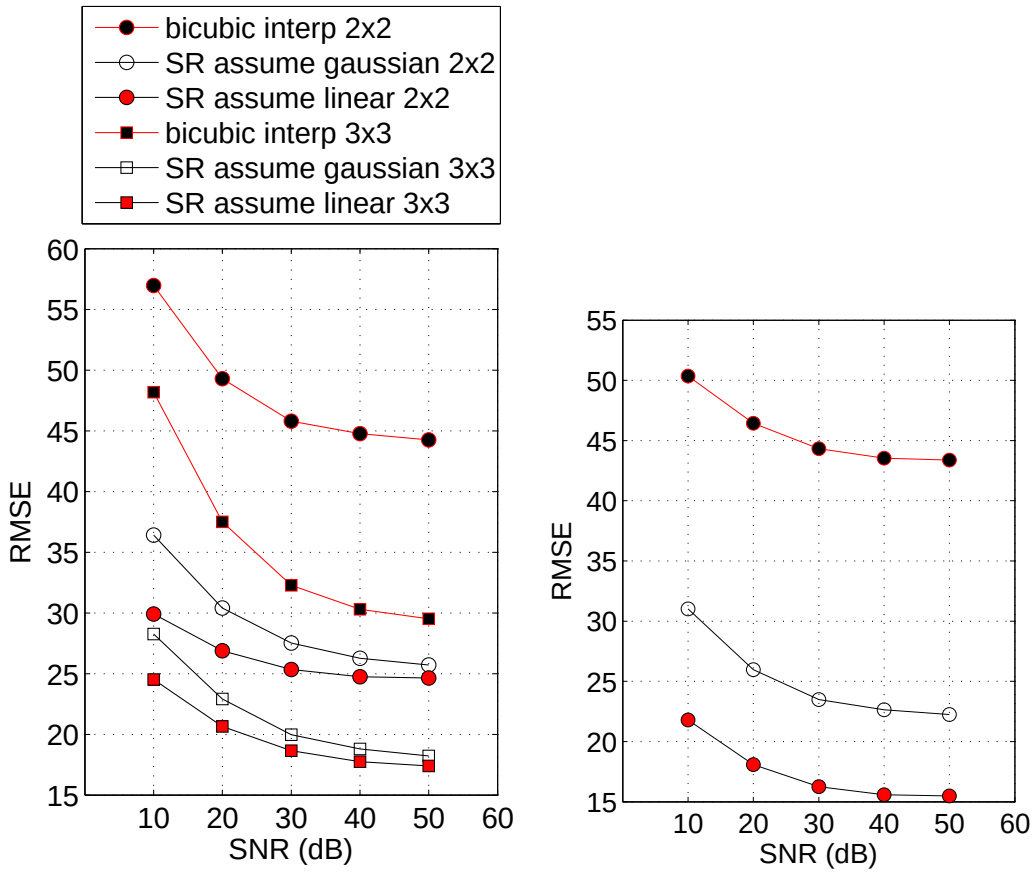
(a) (b)

Figure 6.9: The ground-truth frames for the two sets of experiments.

Fig. 6.6. Thus a 3×3 configuration is employed next.

The detailed images of Fig. 6.8 show the SR reconstructed output for *carousel*, for 2×2 and 3×3 configurations. For reference purposes, bicubic interpolation is applied on a single LR frame, with magnification factors 2 and 3, respectively. The SR output is given both for Gaussian PSF approximations, whose support corresponds to the LR pixel size [IP91, ABCC09], and linear PSFs estimated by BIV. The system output is that of the 3×3 configuration with SR that uses linear PSFs (Fig. 6.8).

The last row of Fig. 6.8 demonstrates the system robustness, presenting the reconstructed outputs for significantly noisy LR samples (SNR = 20 dB). Fig. 6.10(a) quantifies the evaluation giving the RMSE values with respect to the ground-truth, for the above scenarios and SNR from 10 to 50 dB.



(a) (b)

Figure 6.10: Errors for the two sets of experiments, for various SNRs. The legend applies to both graphs, with (b) containing only the 2×2 configuration values.

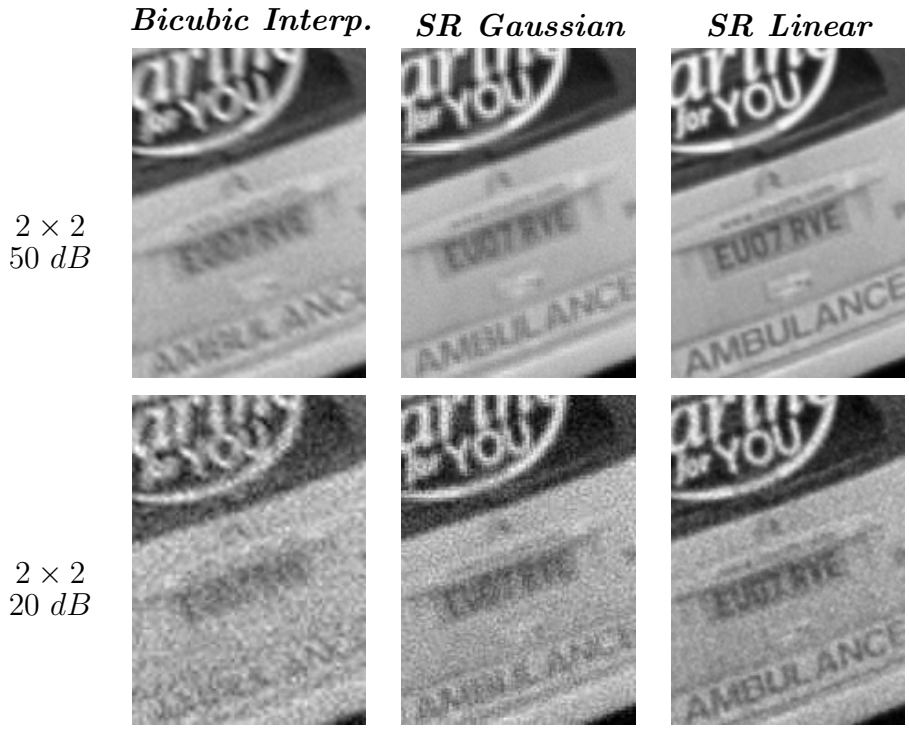


Figure 6.11: Reconstructed outputs for *ambulance*.

Contrary to *carousel*, *ambulance* passes the validity check for 2×2 configuration (Fig. 6.5); thus the pixel size remains at 2×2. Fig. 6.11 presents the indicated outputs for 50 and 20 dB. The system output is that of SR with the linear PSF approximation. The ground-truth is illustrated in Fig. 6.9(b), and the RMSE values with respect to the ground-truth are given in Fig. 6.10(b).

The above software evaluation demonstrates that when the linear PSF is estimated, the SR output improves dramatically, compared to the use of a Gaussian PSF. In the remainder of the chapter, the hardware architecture of the proposed blur identification and classification scheme will be presented and evaluated.

6.7 Hardware Architecture

The block diagram of the proposed FPGA-based architecture, which implements the proposed algorithm, as presented in Section 6.3, is presented in Fig. 6.12. The proposed architecture is fully pipelined and optimized with respect to throughput, area, and data reuse. The input frames and the corresponding binary mask are read from a single-port off-chip RAM, the *Image Memory*. The processing is mainly divided in two stages, each associated with certain system blocks and input stream, as explained in the following subsections. The first processing stage identifies the motion direction θ, and the second stage both calculates the motion extent L and implements the validation scheme.

6.7.1 Directional Filters and Minimum Total Intensity Block

During the first processing stage, the input frame is horizontally traversed, and the directional image derivatives $\Delta_\phi(f)$ and corresponding total intensities $I(\phi)$ are calculated (Section 6.3). To maximize data reuse, an *Extract Processing Window* block, produces a processing window in every clock cycle. In every cycle, the current processing window is fed in parallel to the directional filters, for the computation of the directional image derivatives, thus maximizing the parallelism in the derivative calculations.

The Sobel approximation to the derivative is employed, which is in 0° the

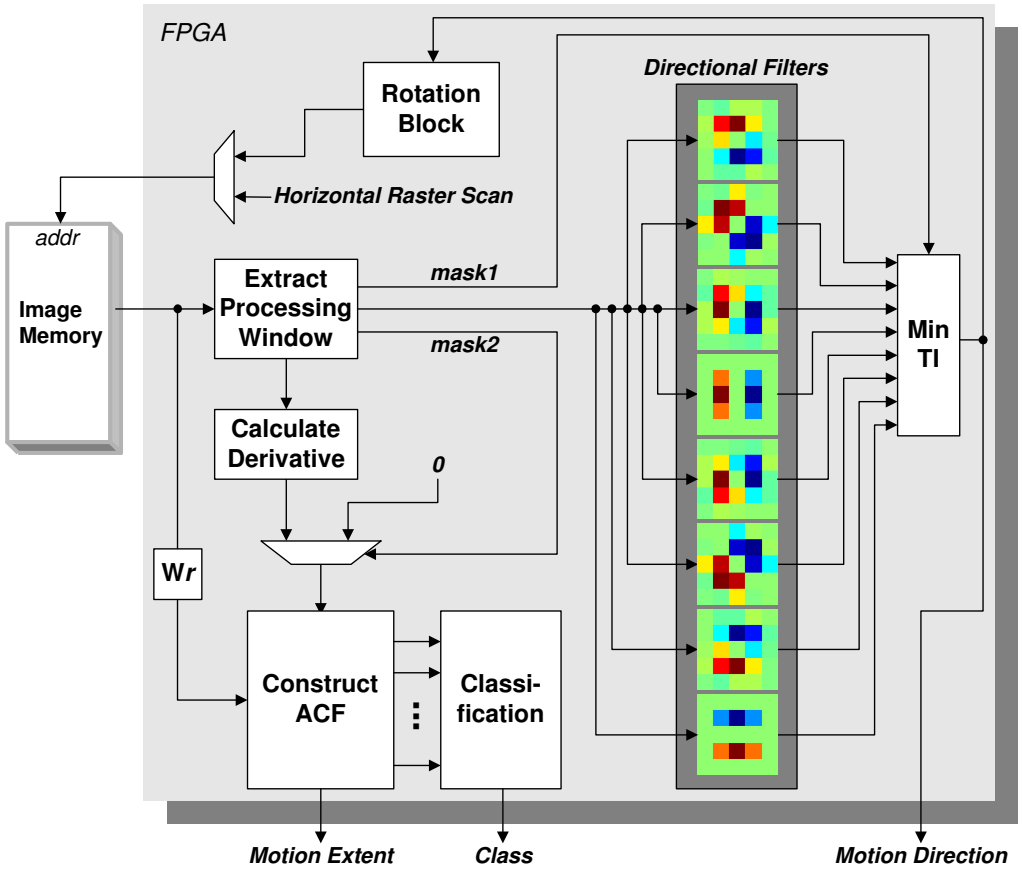


Figure 6.12: An overview of the joint blur identification and validation system, as implemented on FPGA.

following:

$$S_{0^\circ} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \tag{6.9}$$

In directions other than the horizontal and vertical, the Sobel approximation requires 5×5 coefficients, to accommodate the corresponding rotations of the derivative kernel.

As explained in Section 6.3, the image derivatives should be calculated on the *umasked* frames. After the derivatives have been calculated, the *Minimum Total Intensity (MinTI)* block computes for each derivative the sum of absolute intensities only of pixels with mask bit 1. In this manner, the static background is excluded from the intensity calculations. The *MinTI* block finally returns the direction θ corresponding to the minimum total intensity. This does *not* necessarily coincide with a filter's direction. That is, if the difference between the minimum total intensity and the total intensity corresponding to an adjacent filtering angle are similar, then the block returns a weighted average direction. The ACF will then be computed by the subsequent blocks along that direction.

Only the motion line matters in the above computations, and not which way the object moves along that line. Therefore, the rotation angles included in the filter-bank need to span only half of the cartesian plane. Fig. 6.12 demonstrates a filter-bank employing eight different angles, thus a step of 22.5° is used for the filter formation.

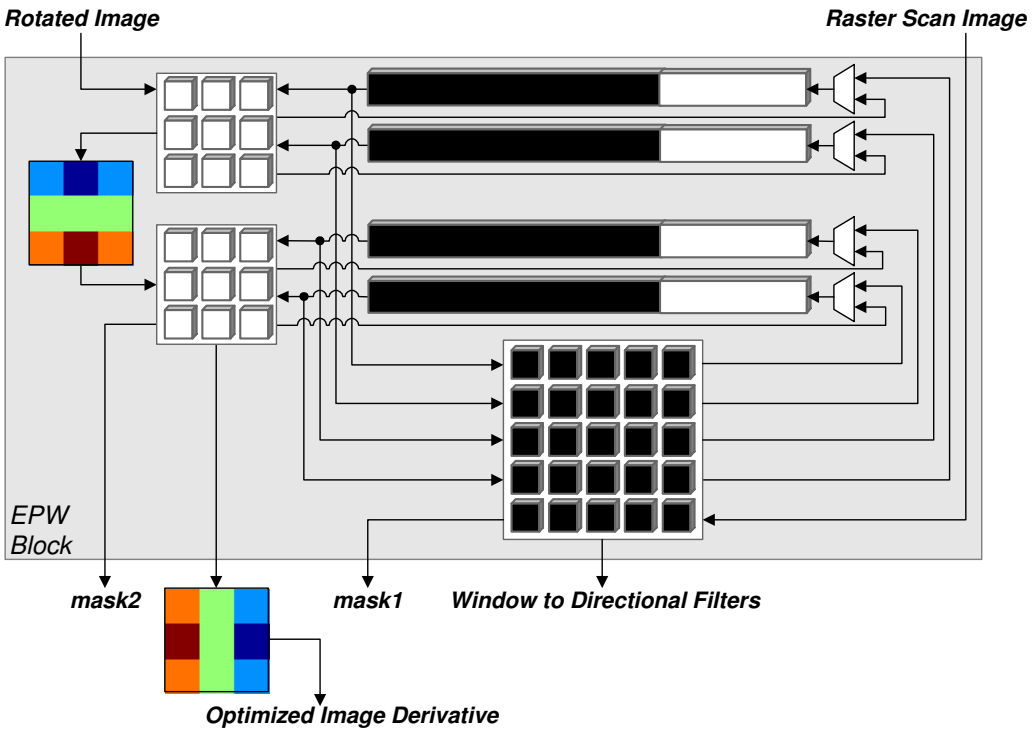


Figure 6.13: The *Extract Processing Window (EPW)* block operates in two different modes, which are distinguished in the figure with the black and white shades. Each mode is associated with a processing stage.

6.7.2 The Rotation Block

The mean ACF should be computed on the optimized derivative g in direction θ (Section 6.3). To do this, instead of combining horizontal scanning with non-horizontal processing, it is computationally more efficient to do the opposite: traverse the frame along θ, thus horizontally scan the rotated frame, and process the corresponding rows. In addition, by doing the above, only two directional filters are required for the generation of g: a vertical and a horizontal filter.

The rotation is implemented using the nearest-neighbor interpolation scheme,

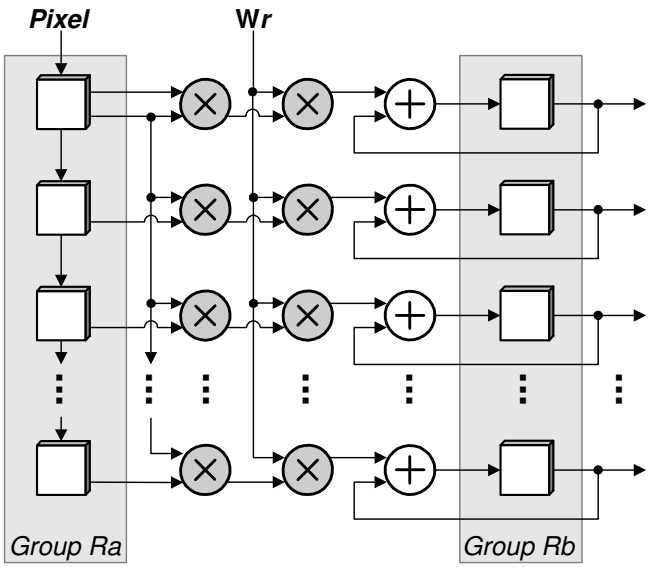


Figure 6.14: The *Construct ACF* block. The number of registers comprising groups *Ra* and *Rb* equals the number of lags under consideration.

which transforms the rotation problem to the simpler problem of generating the corresponding stream of RAM addresses. Thus, the *Rotation Block* takes the angle θ as its input, and generates the streaming addresses for the horizontal raster scan of the rotated by θ frame. Lookup tables return the sine and cosine values corresponding to θ, for the formation of the cartesian rotation equations:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.10}$$

6.7.3 The Extract Processing Window Block

This block increases data reuse by producing in every cycle a convolution window. The block employs FIFOs and registers in the circular buffering scheme demonstrated in Fig. 6.13. FIFOs are implemented with on-chip dual-

port BRAMs. The block operates in two modes, associated with the two main processing stages, thus saving hardware resources.

The first mode, *Mode 1*, which is denoted in Fig. 6.13 with black shade, is associated with the generation of the directional image derivatives $\Delta_\phi(f)$, as has been mentioned in Section 6.7.1. In *Mode 1* the input frame f is read into the buffers in horizontal raster scan order. Thus, the active length of the FIFOs equals the length of the rows of f. All four FIFOs feed the same window of registers, whose size is 5×5, *i.e.* the maximum required filter size (Section 6.7.1). As mentioned in Section 6.7.1, in every cycle, the processing window is read in parallel by the directional filters of Fig. 6.12.

The second mode, *Mode 2*, is associated with the calculation of the optimized image derivative g. Therefore, in *Mode 2*, the buffers of Fig. 6.13 are fed with f rotated by θ, and an active FIFO width equal to the diagonal of frame f is required to accommodate the longest possible row. Two independent buffering structures, related to different filtering operations, are formed. Each includes a FIFO pair and a separate window, denoted in Fig. 6.13 with a white shade. The first filter feeds a vertical Sobel filter that calculates the vertical image derivative $\Delta_{\theta+90^o}(f)$, to remove object edges as explained in Section 6.3. The vertical derivative is directly consumed by the second buffering structure, which feeds a horizontal Sobel filter, and the optimized derivative g is produced. Therefore, only 0° and 90° derivatives are involved in *Mode 2*, and a 3×3 window size suffices for the Sobel approximation.

As mentioned in Section 6.7.1, the derivatives are applied on the unmasked

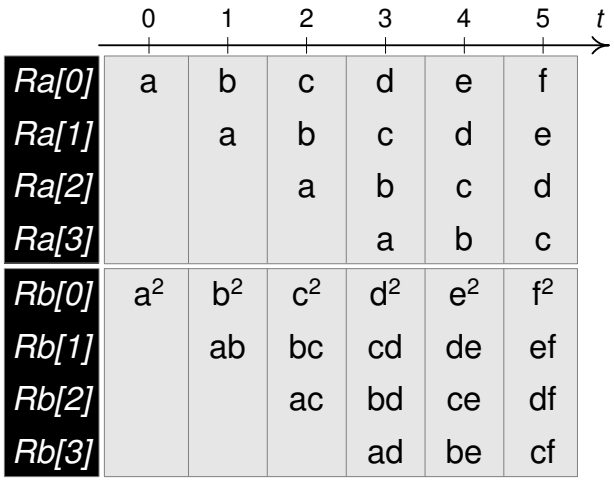


Figure 6.15: Time diagram of the ACF computation for a hypothetical 6-pixel row "abcdef". The t axis indicates the clock cycles.

frame, thus the mask bits are propagated to the next processing levels along with the corresponding pixels.

6.7.4 The Construct ACF and Classification Blocks

The ACF calculations are executed on the rows of the optimized image derivative, which are generated in a streaming manner–1 pixel/cycle–by the structure of Fig. 6.13. The current pixel is propagated to the *Construct ACF* block only if the accompanying mask bit is 1, *i.e.* the pixel belongs to the unrotated input frame f and is also a part of the moving object. To minimize the latency and resources, redundancies involved in the ACF computation are removed. Therefore, the ACF coefficients are computed only for half of the cartesian plane, since the ACF is symmetrical with respect to axis y (Fig. 6.3). In addition, since the expected intra-frame motion extent is smaller than the row length,

only ACF coefficients up to a predefined maximum lag are calculated.

The *Construct ACF* block of Fig. 6.14 consumes 1 pixel/cycle. To clarify its functionality, Fig. 6.15 demonstrates the dataflow for a hypothetical 6-pixel row, using a maximum lag 3. Each register of the two groups of Fig. 6.14 corresponds to a particular lag (0 to 3), as indicated by the indexing of Fig. 6.15. The upper rectangle displays the contents of registers *Ra* at every cycle, while the lower rectangle shows the values that are *added* to each *Rb* register. To obtain the mean ACF, the block processes in this manner the entire image, accumulating information from all rows in registers *Rb*. Thus, at the beginning of a new row only the *Ra*, and not the *Rb* registers, are reset. Each row contributes in the mean ACF with weight *Wr*, which is proportional to the number of non-zero mask bits of the particular row. In this manner, rows containing insignificant part of the moving object do not bias the output. The *Wr* weights should be available at the beginning of each row's processing; thus, for their calculation, part of the pipeline is circumvented to fetch from an earlier level the corresponding mask bits. At the end, registers *Rb*, each related to a particular lag, contain the coefficients of the mean ACF.

The validity of the linear and uniform motion assumption is decided by the *Classification* block. This control block takes the mean ACF coefficients as its input, implements the control logic described in Section 6.3, and renders a classification bit that indicates if the assumption is valid or not.

6.8 Experimental Results

The current section presents an evaluation of the proposed blur identification and validation scheme, within the context of the system of Fig. 6.4. The section is mainly divided in two parts:

1. The first part presents a hardware-oriented evaluation for the FPGA-based blur identification and validation block. Specifically, Section 6.8.1 discusses the throughput and hardware requirements of the proposed architecture for different sets of parameters. Then, Section 6.8.2 evaluates the blur identification and validation performance of the proposed hardware block for various system parameters and different levels of noise.

2. The second part of the section includes a software-oriented evaluation of the overall reconstruction process. Specifically, Section 6.6 investigates how incorporating the above scheme into the SR-based video enhancement system of Fig. 6.4 affects the final output. The SR reconstruction quality is evaluated for different configurations of the adaptive sensor and types of PSF approximation.

6.8.1 Implementation Requirements

The design targets a Celoxica ADMXRC4SX board [Cel08], which hosts a Xilinx Virtex-4 FPGA [Xil08] and 4 ZBT SSRAM banks [Cel08]. The DK5 Handel-C compiler [Cel08] has been used, and the implementation has been placed and routed using Xilinx ISE v.9.2.01i [Xil08]. The operating frequency

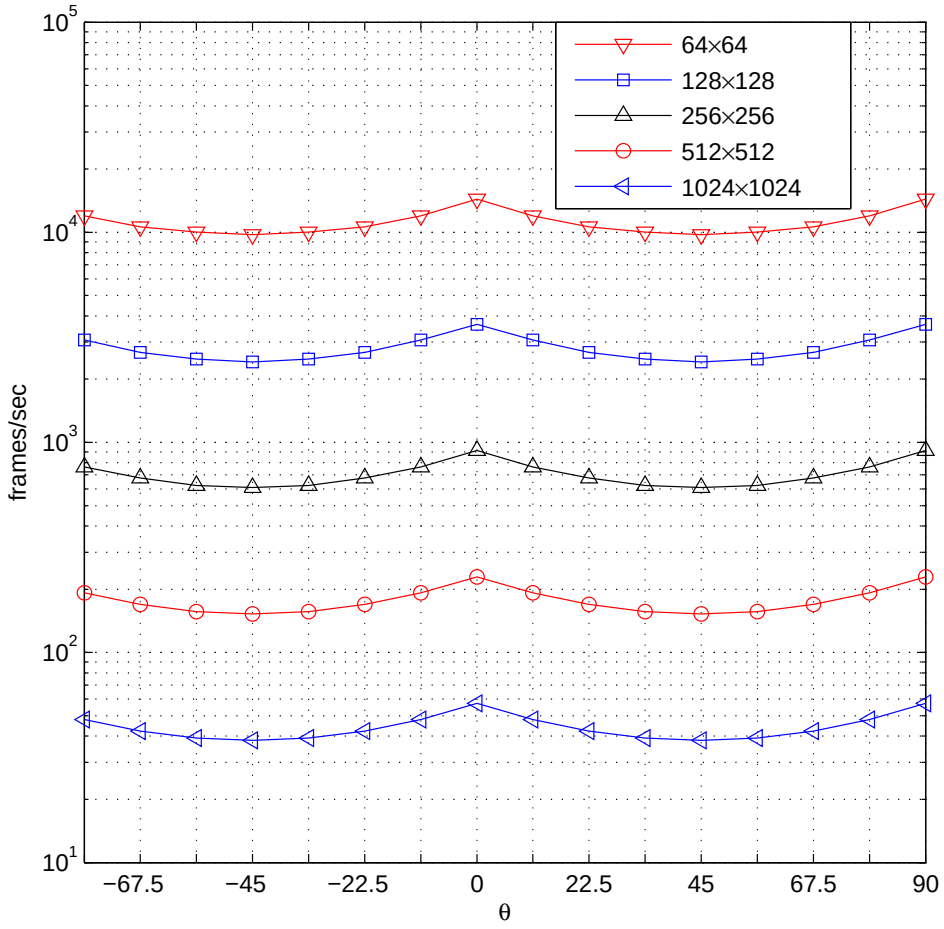


Figure 6.16: System throughput for different frame sizes and motion directions. The vertical axis is in logarithmic scale.

of the design on ADMXRC4SX is 120 MHz. The critical path of the circuit lies in the control logic of the *Rotation Block* (Section 6.7).

The total number of clock cycles required to process a single frame is as follows:

$$Cycles = 80 + (V \times U) + (V_r \times U_r) + (2 \times G) \tag{6.11}$$

where V and U denote the number of rows and columns of the original frame, V_r and U_r denote those of the rotated frame (which depend on θ for given V and U), and G is the number of lags that are considered in the ACF calculations. Also, a constant equal to 80 is the sum of the reset and latency cycles of the circuit. Thus, the total number of cycles is mainly affected by the frame size and the motion direction (indicated by θ). Specifically, the term $V \times U$ corresponds to the first processing stage when the original frame is horizontally scanned, while the term $V_r \times U_r$ is related to the second stage, when horizontal scanning is applied on the rotated frame. The number of lags also contributes in a small extent in Eq. 6.11, determining the number of cycles for the *Construct ACF* and *Classification* blocks. On the contrary, the number of directional filters does not affect the number of cycles, as these filters operate in parallel.

The system throughput is plotted in Fig. 6.16, with respect to the frame size and the motion direction. The vertical axis of Fig. 6.16 is in logarithmic scale. To meet real-time requirements, the system should achieve at least 25 fps. All throughput values in Fig. 6.16 keep a large 'safety margin' above that minimum, the smallest being 38 fps for frame size 1024×1024 and $\pm 45°$. Due to its high throughput, the proposed architecture is extremely appropriate for low power applications, as it can be operated in lower frame-rates. The number of external memory accesses is $2 \times V \times U$ per frame, since each frame is read twice from the off-chip RAM (the *Image Memory* in Fig. 6.12), once for every processing stage, according to the raster scan order that corresponds to that

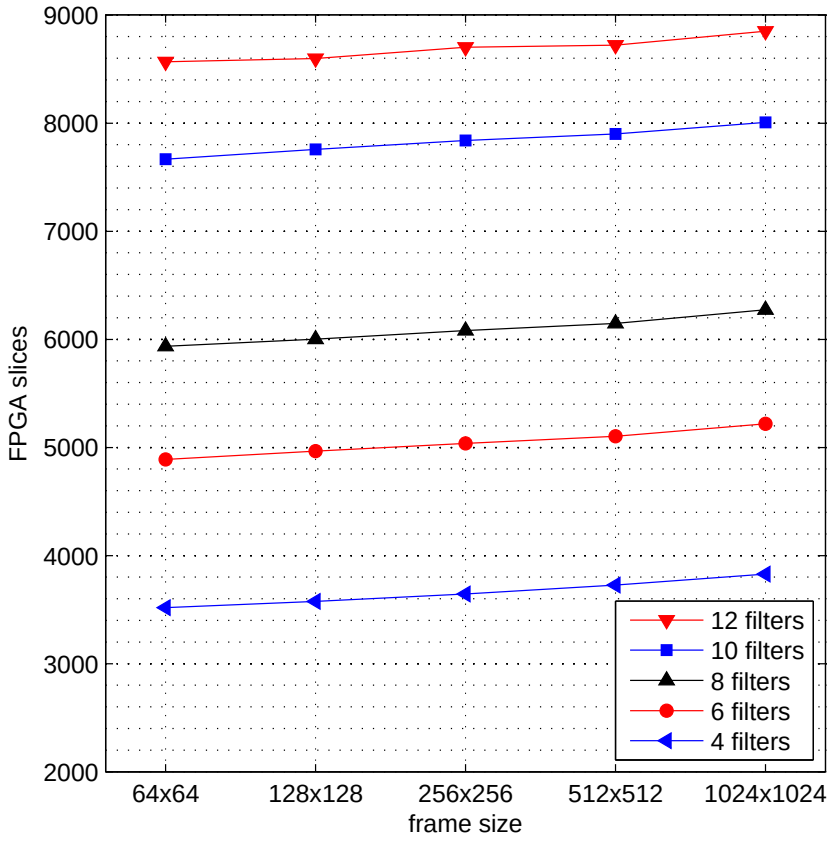


Figure 6.17: Number of FPGA slices for different number of directional filters and frame sizes. The number of considered ACF coefficients equals 30.

stage.

The number of FPGA slices mainly depends on the number of directional filters and is slightly affected by the frame size, as Fig. 6.17 demonstrates for $G = 30$. The effect of variations of G on the number of slices is relatively insignificant.

The number of BRAMs depends on the size of the derivative kernel that is used. For a 3×3 derivative filter, whose rotated version requires up to 5×5 coefficients, the required number of BRAMs equals 4, for the frame sizes reported in Fig. 6.17, as for these sizes, the maximum required length of line

buffers does not surpass the BRAM length.

6.8.2 Performance Evaluation for the Blur Identification and Validation System

The performance evaluation is done using semi-synthetic data. That is, a real image, captured with a simple hand-held digital camera, has been convolved with a motion PSF and contaminated with noise, to synthetically produce the blurred frames. In this way, the system performance is accurately evaluated, since the actual validation class and motion parameters are known, and are thus compared with the validation output and estimated parameters. The input frames have been contaminated with white Gaussian noise with various SNRs, which range from 10 to 70 dB.

In Fig. 6.18, a ROC curve is used to evaluate the performance of the classifier under different noise levels and employing different numbers of directional filters for the image derivative computations. The input frames used for the generation of the ROC were produced after applying various linear, nonlinear, uniform and nonuniform motion PSFs on the ground-truth frames of Fig. 6.19. Specifically, 50 different motion PSFs of varying spatio-temporal shapes and magnitudes were applied on each ground-truth, locally on the foreground objects of the four leftmost images and globally on the entire rightmost image. Therefore, in the last case the associated binary mask has ones on the entire area of the unrotated frame f. The two first images of Fig. 6.19 have reso-

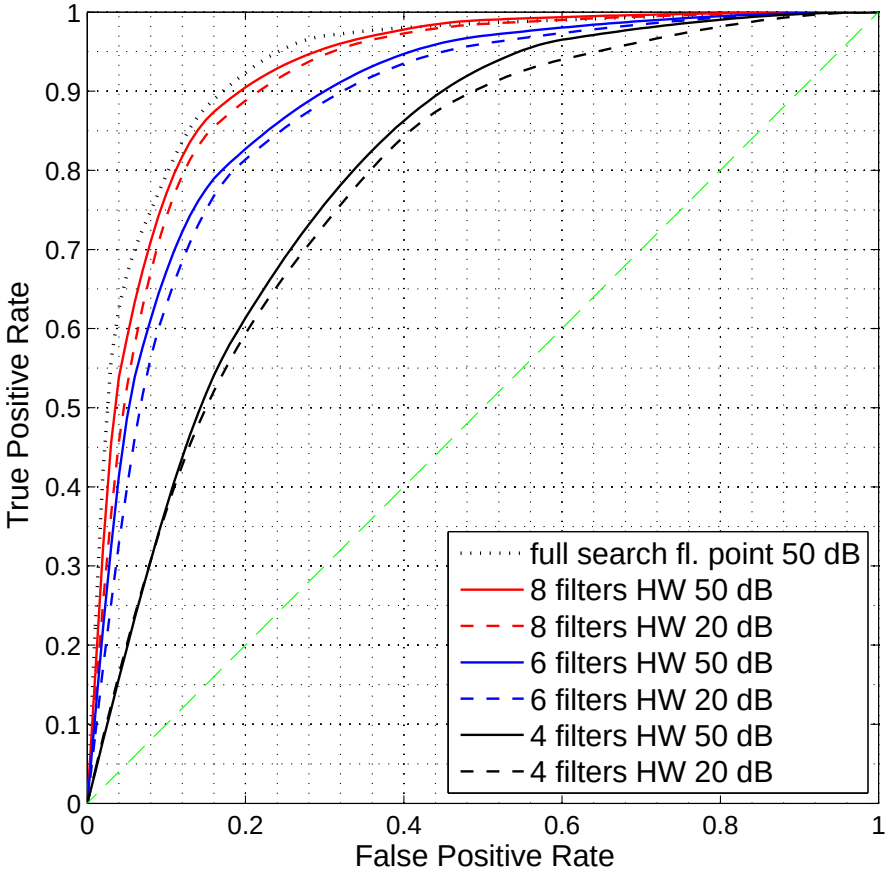


Figure 6.18: ROC curves for the indicated number of filters and noise level.

Figure 6.19: Ground-truth frames used for the generation of the test sequences.

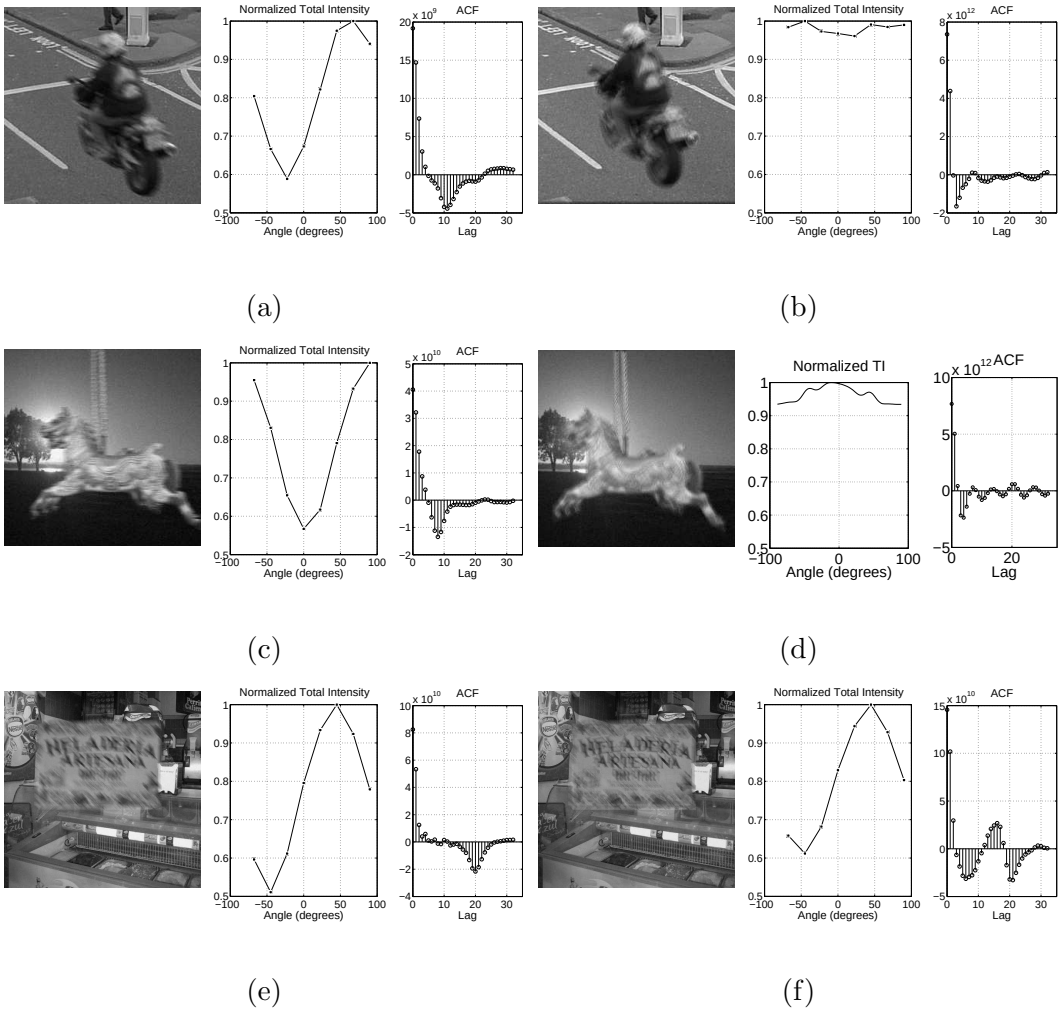


Figure 6.20: The linear and uniform motion assumption is valid for the blurred frames on the left (a,c,e), but not for those on the right (b,d,f). In particular, the motion is nonlinear for (b,d) and linear but nonuniform in (f). For each experiment, the calculated total intensity graph and the mean autocorrelation coefficients are demonstrated in the figure.

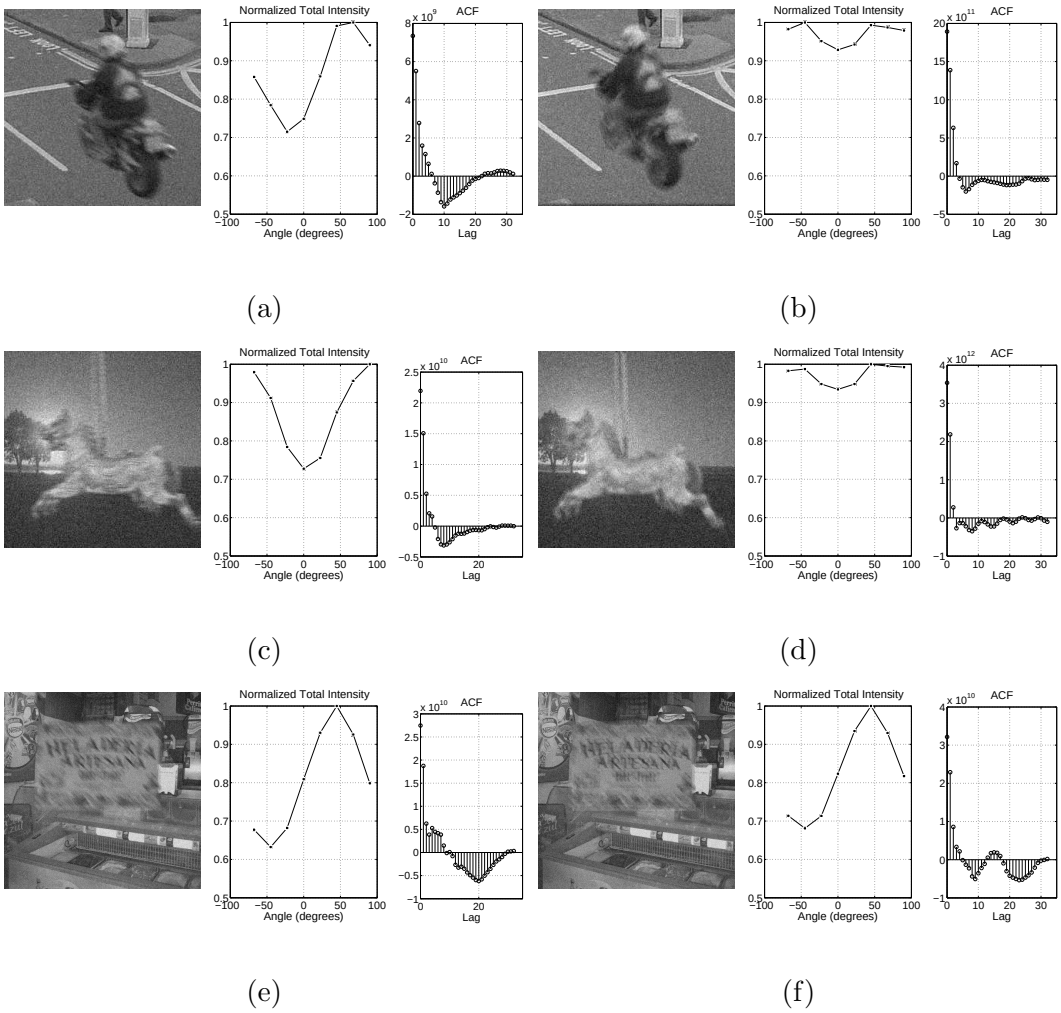


Figure 6.21: The same scenarios as Fig. 6.20, but under significant noise: SNR = 20 dB.

lution 256×256, while the last three have 512×512 resolution. The entire experiment has been repeated considering 4, 6 and 8 directional filters and with SNR 50 and 20 dB. In all experiments, 33 lags have been considered for the autocorrelation calculations, fully accommodating the maximum extent of linear motion that has been used, which is 27 pixels.

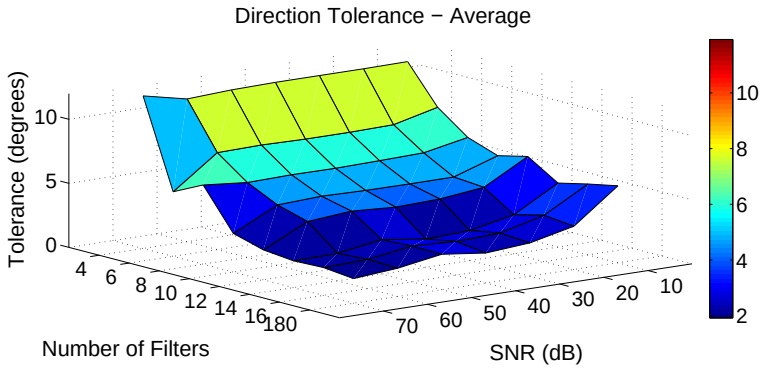
The dotted curve in Fig. 6.18 presents the output of floating point validation, employing full search for the minimum total intensity, thus a step of 1^o and 180 directions. The floating point scenario is implemented in Matlab. The rest of the curves present the outputs of the hardware implementation for numbers of directional filters and noise levels that are indicated in the legend. As can be observed in Fig. 6.18, the classifier's performance significantly improves when the number of filters increases from 4 to 6, and from 6 to 8. For 8 directions or more, the achieved performance is similar to the best possible floating point full search scenario. As demonstrated in Fig. 6.18, the variation of performance for a given number of filters under different SNRs is not significant, which proves the increased robustness of the classifier with respect to noise.

Six representative examples, taken from the set of experiments used in the above evaluation, are demonstrated in Fig. 6.20. The scenarios of Fig. 6.20 involve 50 dB SNR and 8 directional filters, while six different motion types are employed: three that are linear and uniform, in Figures 6.20(a,c,e), and three that are not, in Figures 6.20(b,d,f). In particular, Figures 6.20(b,d) contain nonlinear sinusoidal motion, whereas Fig. 6.20(f) contains nonuniform

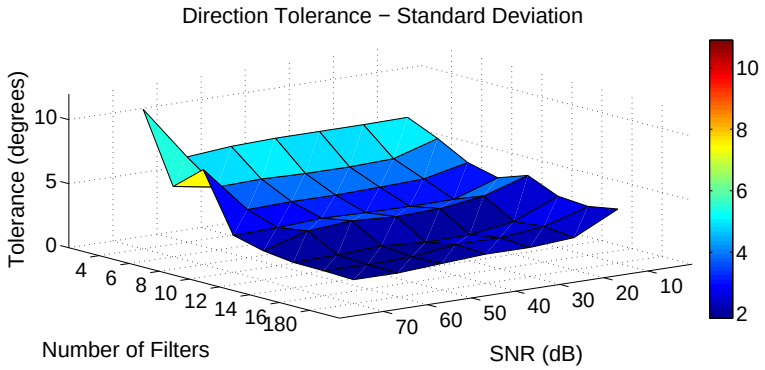
motion resulting from high frequency temporal vibrations (Section 6.3). The blurred frames are correctly classified; thus, for Figures 6.20(a,c,e), the validation output is positive, whereas for Figures 6.20(b,d,f) it is negative. For positive validation, the linear motion parameters are also accurately identified. The actual motion parameters of Figures 6.20(a,c,e) are: extents of 11, 8, and 20 pixels, and directions of -25^o, 0^o and 45^o, respectively. As observed in Fig. 6.20, these parameters are accurately identified with the lag of the minimum autocorrelation coefficient and the minimum total intensity, respectively.

Fig. 6.21 presents the outputs of the system for the same scenarios as Fig. 6.20, but under a high level of noise. In particular, an SNR of 20 dB is employed, and the corresponding noisy frames are shown in Fig. 6.21. For the examples of Fig. 6.21, both the classification outputs and the estimated motion direction are unaffected by the heavy noise. A few very small divergences are observed for the estimated motion extents in the case of positive classification. Specifically, for Fig. 6.21(a), the estimated motion extent is now 10 pixels instead of 11, which is the actual value and which was accurately identified at 50 dB. In Fig. 6.21(c), the extent of 8 pixels is identified. However the high level of noise makes the negative lobe of the ACF less steep and thus the minimum less clear. A more systematic evaluation of the accuracy of the identified linear parameters with respect to the level of noise is presented in Figures 6.22 and 6.23, where a large test set is considered.

In Figures 6.22 and 6.23, the accuracy of the identified linear parameters is evaluated with respect to the number of directional filters and the level of

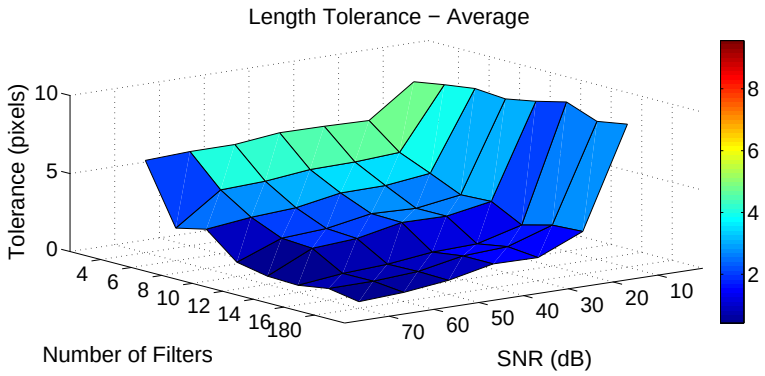


(a)

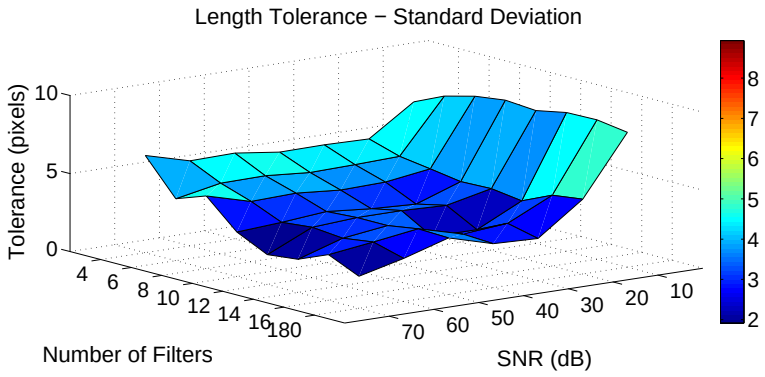


(b)

Figure 6.22: The average and standard deviation values of the errors between the actual and estimated motion direction, for linear and uniform motion.



(c)



(d)

Figure 6.23: The average and standard deviation values of the errors between the actual and estimated motion extent, for linear and uniform motion.

noise, for a set of linearly blurred frames. To produce this set, 50 different linear motions, of random angles and extents between 7 and 27 pixels, were applied on the ground-truth frames of frames of Fig. 6.19. The graphs of Fig. 6.22 present the average and standard deviation values of the errors between the actual and estimated motion direction, whereas the graphs of Fig. 6.23 present the same values for the estimated motion extent. The value 180 on the axis with the numbers of filters indicates the output for the full-search, floating point scenario, where a step of 1^o covers all integer directions, calculating 180 derivatives. Contrary to the validation block, which achieves maximal performance with 8 directional filters (Fig. 6.18), for the identification block to obtain accuracy similar to the full-search floating point scenario, more than 8 filters need to be used. This is expected, since validation is a binary problem, but blur identification is not. Therefore, the choice of the number of filters for joint blur identification and validation depends on the required accuracy in the estimation of motion direction and extent, which is imposed by the given application and mainly the subsequent reconstruction block that utilizes these parameters.

6.9 Summary

This chapter proposes a joint blur identification and validation scheme, which not only estimates linear blur, but also validates the initial linearity and uniformity assumption. This keeps the computational cost of blur identification

low, while identifying cases where the assumption is invalid and thus the PSF estimation is inaccurate.

To tackle cases of invalid initial assumption, this chapter proposes combining the blur identification and validation scheme with the reconfiguration property of an adaptive image sensor. The sensor grid is adapted to the local motions depending on the validity of the linear motion assumption. Once the appropriate configuration is applied, the appropriate PSF approximation, which is optimal for the given type of blur, is employed to reconstruct the final output.

The proposed joint blur identification and validation system is implemented on an FPGA and is evaluated for different sets of parameters. The system throughput, which mainly depends on the frame size, is significantly higher than the 25 fps real-time requirement. The area of the circuit mainly depends on the number of directional filters used for the derivative calculations, while the required number of on-chip Block RAMs is determined by the size of the derivative kernel. The proposed method is robust to noise, and accurately validates the initial assumption with 8 directional filters.

Experimental results show that when applied to a SR-based video enhancement system, the proposed scheme surpasses limits related to traditional SR, demonstrating the significance of adapting the sensor grid to the given motion for the efficient processing of the raw samples.

Chapter 7

Conclusions and Future Work

7.1 Summary of the Thesis

This thesis proposes an FPGA-based real-time video enhancement system that targets an adaptive image sensor, that is, a sensor whose grid can be reconfigured in real-time to form larger pixels where and when it is needed. The proposed system belongs to the broad area of computational photography and video. The blocks of the system implement functions, such as super-resolution, blur identification, and multi-resolution image decomposition, which are key operations in the general area of image and video processing, and can also be seen as stand-alone processing blocks for general-purpose applications. The work presented in this thesis spans both the algorithmic and the hardware design domains. In the algorithmic domain, robust to noise methods have been developed to execute super-resolution and blur identification. The data intensive nature of the system's processing blocks and the strict real-time con-

straints impose the requirement for fast and effective video processing. Thus, in the hardware level, the architectures have been designed to allow for high throughput and high data reuse.

An important step in multiframe fusion is registration, whose robustness is achieved by employing a multiresolution decomposition. Thus, the discussion of the work presented in this thesis started with Chapter 3, by presenting the FPGA implementation of the major 2-D DWT computation schedules that compute the discrete wavelet pyramidal decomposition. The performance of the architectures was evaluated with respect to issues such as throughput, area, and energy dissipation. Experiments showed that the Line-Based (LB) computation schedule achieves the highest throughput and lowest energy dissipation. The Block-Based (BB) approach, which has the same memory access locality as LB, is however associated with the highest control complexity, which results in the lowest throughput and the highest energy dissipation. As for the Row-Column (RC) approach, it has the lowest hardware cost.

The main discussion of the proposed video enhancement system started with an exploration of the configuration space of an adaptive image sensor in Chapter 4. Two approaches were proposed, namely an SR-based and a deconvolution-based approach. Experiments showed that the SR-based approach outperforms the deconvolution-based approach on dynamic regions. Therefore, the rest of the thesis focused in the SR-based multiframe fusion approach, being the most suitable option for handling highly motion intense regions, since motion-deblurring is the main problem that this work targets.

In the SR-based approach, motion areas are locally treated, by configuring motion regions to LR areas that produce high-frame-rate samples of reduced blur, whereas areas of no motion or very small motion are configured to elementary pixels. The high-frame-rate samples of the LR areas are fused under the SR framework, to construct an output of high resolution both in space and time. Super-resolution reconstruction was then investigated, in Chapters 5 and 6, with respect to both algorithmic and hardware performance. Specifically, in Chapter 5, the traditional Iterative Back Projection SR approach was modified to account for noise statistics, and its implementation was evaluated under different noise levels. To handle large motion magnitudes, non-isotropic PSFs should be incorporated in the SR fusion to allow for accurate SR reconstruction. Thus a blur identification and validation scheme has been proposed in Chapter 6 for the estimation of the non-isotropic intra-frame PSFs. The above scheme validates the initial linear and uniform motion assumption while calculating the blur parameters. Whenever the initial assumption is found invalid, the blur identification and validation block interacts with the adaptive image sensor by increasing the pixel size in the next sensor reconfiguration cycle. Larger pixels require faster integration times and therefore produce more linear and uniform motion PSFs. This increases the accuracy in the blur identification process and improves the reconstruction quality of the subsequent SR block.

7.2 Conclusions

The general conclusion that derives from the presented work is that in the context of the video enhancement system that utilizes an adaptive image sensor, which is proposed in this thesis, FPGA technology offers a flexible processing platform that allows real-time performance, while having several advantages that are not provided by ASIC technology. These are mainly the reconfigurability and ease of prototyping.

More specific conclusions are associated with the individual chapters of the thesis and are the following.

- The comparison of the major 2-D DWT computation schedules, which is presented in Chapter 3, demonstrates that the choice of a particular architecture for the execution of the 2-D DWT depends on the given specifications. These are related to the targeted throughput, area, and power consumption. The Line-Based (LB) approach achieves the highest throughput and the lowest energy consumption, while the Row-Column (RC) approach is related to the lowest area.

- Chapter 4 explored the configuration space of the adaptive image sensor with respect to the spatio-temporal resolution that can be achieved after processing the captured data. An SR-based approach and a deconvolution-based approach were proposed, each related with a particular sensor configuration. The evaluation of the approaches demonstrated that the SR-based outperforms the deconvolution-based scheme on dynamic re-

gions.

- Chapter 5 tackles the problem of robust to noise, real-time SR-based reconstruction. The chapter demonstrates that incorporating the noise statistics into the traditional SR framework of the Iterative Back Projection (IBP) SR approach results in a robust to noise system with high reconstruction quality under noise levels in the range of 10 to 70 dB. A high-throughput architecture of the above scheme is proposed and is implemented on an FPGA achieving real-time performance.

- Chapter 6 aims at improving the reconstruction quality of the SR block by employing blur identification prior to the actual reconstruction stage. In this manner, the temporal resolution of the output is no longer limited by the temporal resolution of the frames that are fused (as was the case in Chapter 5). To further improve the final reconstruction quality, Chapter 6 aims to increase the accuracy of blur identification by incorporating assumption validation in the linear blur identification process. The blur identification and validation block that is proposed identifies cases where the initial linearity and uniformity assumption is invalid, as it has been demonstrated in the performance evaluation of Chapter 6. The proposed scheme interacts with the adaptive sensor in the manner that is proposed in Chapter 6, and once the appropriate pixel size is employed that gives rise to a valid motion assumption, highly accurate PSFs can be estimated. As has been demonstrated in the performance

evaluation of Chapter 6, this improves the quality of subsequent SR reconstruction. To target real-time performance, an efficient FPGA-based architecture is proposed that implements BIV in real-time.

7.3 Future Work

The work presented in this thesis could be expanded in various ways both in the algorithmic and in the hardware level. The current section discusses some suggestions and possibilities for future work, presenting both the short term plans and the long term goals that are related to this work.

7.3.1 Short Term

As explained in Chapter 6, the proposed blur identification and validation scheme interacts with the adaptive sensor to increase the accuracy of blur identification by adapting the pixel size of the sensor to the given motion. As a result, the accuracy of blur identification and the subsequent reconstruction quality depend on the configuration frequency of the sensor. Therefore, the short term plans for future work include the investigation of the reconfiguration rate of an adaptive image sensor, where the reconfiguration is executed in order to address the problem of non-linear motion, as stated in Chapter 6. The effect of this reconfiguration rate on SR reconstruction will be evaluated.

As stated above, the configuration rate of an adaptive sensor is limited by the sensor technology. On the other hand, as discussed in Chapter 4,

the LR motion region should be configured on the region of the scene where the moving object is expected to move until the next sensor reconfiguration occurs. It may thus be effective to determine the direction of expansion of the LR area, with respect to the direction of motion. Specifically, this direction will be perpendicular to the object motion, and the LR region will span the area of the sensor, which lies ahead of the moving object, being expanded towards that direction.

The linear blur identification and validation system that is proposed in Chapter 6 is based on the linear, *i.e.* first order polynomial, motion assumption. The proposed scheme interacts with the adaptive sensor by configuring it, so as to achieve linear and uniform intra-frame motion. As stated above, the configuration frequency of the sensor is a limiting factor for the proposed system, and therefore, for a slow reconfiguration rate adaptive image sensor, it may be preferable to employ a second order, instead of a first order motion blur approximations. The exploration of this is included in the short term plans for future work. In particular, the effect of using second order approximation will be investigated with respect to the SR reconstruction quality, and the HW requirements for the blur identification block and the SR block will be evaluated.

The investigation of the 2-D DWT on FPGA, which is presented in Chapter 3, was performed by considering a single-port image memory and a single filter-bank which is shared among all levels and stages of the transform. This choice was made based on the fact that the off-chip memory is usually single-

port and also based on the aim of employing the minimum hardware for the main 1-D DWT processing, that is a single filter-bank. The importance of this work is due to the wide range of image and video processing applications that are based on the execution of the 2-D DWT, such as video compression and image registration. The investigation of the 2-D DWT computation schedules can act as an insight on which schedule is most suitable for the specifications imposed by the given application. As future work, this investigation will be broadened by extending the range of system parameters, such as the number of ports of the off-chip image memory or the number of filters that execute the 1-D DWT.

7.3.2 Long Term

Throughout the thesis, the assumption of rigid moving objects has been used. Extending the system to handle as well non-rigid object deformations is a challenging task. Several approaches for handling non-rigid deformations have been proposed [BK99]. Tailoring the methods to the particular video enhancement application could produce a better determined problem and simplify the processing. Specifically, since the computations are applied on adjacent video frames, the non-rigidness of the related motions is somewhat confined, due to the proximity of the temporal samples of the continuous motion. Therefore, when extending the system to non-rigid object motions, additional motion constraints can be employed, to decrease the set of possible motions and reduce the complexity of non-rigid reconstruction. Moreover, the system can be

extended so as to focus on a particular type of moving objects, such as cars or human faces. This would require a training and a detection stage for the particular class of objects [ACS08, JLD06, LB08], but would improve the reconstruction quality, due to the additional constraints imposed by the object class.

In the work presented in this thesis, it has been assumed that no motion irregularities, such as motion discontinuities and occlusions are present in the captured video sequence. However, in real-life video sequences, the presence of motion discontinuities and occlusions may affect the final system performance. Such irregularities were out of the scope of this thesis, but have been addressed in the literature with various methods that have been proposed [PHZ08, PVW06, ZWYF08]. Identifying and incorporating the appropriate method in the proposed system, so as to handle such irregularities and maintain the high reconstruction quality under such circumstances, is a part of the future work. Moreover, the HW requirements associated with the above will be investigated.

The reconfiguration property of the grid of an adaptive sensor has been employed to reduce motion blur, locally for object motion and globally for camera motion. The blocks of the proposed system have been endowed with robust to noise mechanisms and tested under different noise levels. However, the reconfigurability of the sensor has not been exploited to address high dynamic range applications. In fact, a reconfigurable sensor is extremely appropriate for increasing the dynamic range of a camera [Fov08, RBS99, Gos05]. This

is due to the fact that larger pixels can be formed under low illumination conditions, in order to integrate faster the incident light. On the other hand, saturated frames can be avoided by decreasing the pixel size. When employing an adaptive image sensor, the sensor reconfiguration property can be locally utilized to increase the dynamic range of certain regions, that are either too dark or too bright (saturated). Combining the motion deblurring capability with high dynamic range functionality seems to be a promising idea that may render a significantly competent video enhancement system. Dealing with the motion deblurring problem and the dynamic range problem at the same time, will inevitably result in some conflict. If, for instance, a fast moving object is located in a saturated region, the motion criterion requires a decrease in the pixel size, while the dynamic range criterion demands the opposite. It is thus important to address such trade-offs, in order to employ the best pixel size for the given conditions and application requirements. Determining the pixel size requires an evaluation based on the reconstruction quality of the final output under various illuminations, varying both globally and locally in the scene, in addition to the various motions, such as those employed in this thesis.

The sensor reconfiguration scheme that is presented in this thesis is limited to the formation of square pixels. Employing other shapes for the sensor's pixels and investigating the impact of different shapes on the system's output is part of the future work. The use of hexagonal pixels has recently been proven to be a viable alternative to the traditional grid of square pixels in various image processing applications [TGP02, Jia08]. This is mainly due to

the fact that hexagonal pixels provide high rotational symmetry and a closely packed structure [TGP02]. The latter signifies that the sampling of the real-world scene by the camera's sensor will be denser and thus more accurate. This may further increase the reconstruction quality of the final output.

To conclude, while this thesis has provided a thorough framework for the real-time enhancement of the spatio-temporal resolution of video, a wide variety of possibilities exist for future work, to stimulate further research in various directions and levels of application.

Appendix A

Example Camera Datasheet

The typical camera datasheet given by the camera manufacturer includes information regarding the image sensor, such as the sensor size, the number of pixels in the horizontal and the vertical direction, and the Signal to Noise Ratio (SNR) of the output frames.

An example camera datasheet is given in Fig. A.1, which describes the specifications of the GE DI-XR2-VF3 camera model [Gen09]. As can be observed in Fig. A.1, the SNR of the particular camera is 50 dB, which is a typical value for digital cameras.

DI-XP2-VF3

1/3inch UltraView dome Colour XP2 3-8mm 0.6lux Fixed Iris 24Vac/12Vdc

Specifications

Camera
Image sensor:
- 1/3 in. sensor with GE Xposure technology

Dynamic range:
- 95 dB typ./120 dB max.

Scanning system:
- 525/60 (NTSC); 625/50 (PAL)

Synchronization:
- Internal / line-lock AC

Pixels:
- 720 H x 540 V

Resolution:
- 470 TVL horizontal/460 TVL vertical

Sensitivity:
- 0.6 lux @ f1.2, 30 IRE

Signal-to-noise ratio:
- 50 dB

Automatic gain control:
- Max. 30 dB

Gamma compensation:
- 0.45

Lens options:
- 3 to 8 mm varifocal, 9 to 22 mm varifocal, or 3 to 8 mm autoiris

Physical
Weight:
- 703g

Construction, housing and adapters:
- Aluminum

Construction, dome:
- 3.3 mm polycarbonate (clear or smoke)

Finish, housing: Powder coat
Electrical
Input voltage:
- 12 VDC or 24 VAC

Power:
- 12 VDC, 455 mA; 24 VAC, 160 mA

Environmental
Operating temperature:
- -10 to 45°C

Diagram

Ordering Information

DI-XP2-VF3	1/3inch UltraView dome Colour XP2 3-8mm 0.6lux Fixed Iris 24Vac/12Vdc

www.gesecurity.net

Figure A.1: An example camera datasheet.

Bibliography

[ABC] M. E. Angelopoulou, C.-S. Bouganis, and P. Y. K. Cheung. Blur Identification with Assumption Validation for Sensor-Based Video Reconstruction and its Implementation on FPGA. *IET Computers & Digital Techniques, submitted June 2009.*

[ABC08] M. E. Angelopoulou, C.-S. Bouganis, and P. Y. K. Cheung. Video Enhancement on an Adaptive Image Sensor. In *IEEE International Conference on Image Processing (ICIP)*, pages 681–684, October 2008.

[ABC09] M. E. Angelopoulou, C.-S. Bouganis, and P. Y. K. Cheung. A Sensor-Based Approach to Blur Identification and Super-Resolution for Real-Time Video Restoration. In *IEEE International Conference on Image Processing (ICIP), to appear*, November 2009.

[ABCC08] M. E. Angelopoulou, C.-S. Bouganis, P. Y. K. Cheung, and G. A. Constantinides. FPGA-based Real-time Super-Resolution on an

Adaptive Image Sensor. In *International Workshop on Applied Reconfigurable Computing (ARC)*, pages 125–136, March 2008.

[ABCC09] M. E. Angelopoulou, C.-S. Bouganis, P. Y. K. Cheung, and G. A. Constantinides. Robust Real-Time Super-Resolution on FPGA and an Application to Video Enhancement. *ACM Transactions on Reconfigurable Technology and Systems, to appear*, 2009.

[ACA02] K. Andra, C. Chakrabarti, and T. Acharya. A VLSI architecture for lifting-based forward and inverse wavelet transform. *IEEE Transactions on Signal Processing*, 50(4):966977, April 2002.

[ACS08] N. Alt, C. Claus, and W. Stechele. Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments. In *Design, Automation and Test in Europe (DATE)*, pages 176–181, March 2008.

[AK00] M. D. Adams and F. Kossentini. Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis. *IEEE Transactions on Image Processing*, 9(6):1010–1024, June 2000.

[Alt08] Altera Corporation, San Jose. http://www.altera.com, December 2008.

[AMCA06] M. Angelopoulou, K. Masselos, P. Cheung, and Y. Andreopoulos. A Comparison of 2-D Discrete Wavelet Transform Computation

Schedules on FPGAs. In *IEEE International Conference on Field Programmable Technology*, pages 181–188, December 2006.

[AMCA08] M. E. Angelopoulou, K. Masselos, P. Y. K. Cheung, and Y. Andreopoulos. Implementation and Comparison of the 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs. *Journal of Signal Processing Systems*, 51(1):3–21, April 2008.

[Amp08] Amphion Semiconductor, Ltd. http://www.amphion.com, December 2008.

[Ash95] P. J. Ashenden. *The Designer's Guide to VHDL*. Morgan Kaufmann Publishers, December 1995.

[ASL+03] Y. Andreopoulos, P. Schelkens, G. Lafruit, K. Masselos, and J. Cornelis. High-level cache modeling for 2-D discrete wavelet transform implementations. *VLSI Signal Processing (special issue on Signal Processing Systems)*, 34(3):209–226, July 2003.

[Avi04] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, August 2004.

[Avi07] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, February 2007.

[BB04] C.-S. Bouganis and M. Brookes. Multiple Light Source Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):509–514, April 2004.

[BB08] O. Bowen and C. S. Bouganis. Real-time image super resolution using an FPGA. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 89–94, September 2008.

[BEN04] M. Ben-Ezra and S. K. Nayar. Motion-based motion deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):689–698, June 2004.

[BEZN04] M. Ben-Ezra, A. Zomet, and S.K. Nayar. Jitter Camera: High Resolution Video from a Low Resolution Detector. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 135–142, June 2004.

[BEZN05] M. Ben-Ezra, A. Zomet, and S.K. Nayar. Video Super-Resolution Using Controlled Subpixel Detector Shifts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):977–987, June 2005.

[BHNC05] A. Burchardt, E. Hekstra-Nowacka, and A. Chauhan. A real-time streaming memory controller. In *Design, Automation and Test in Europe (DATE)*, volume 3, pages 20–25, March 2005.

[BIRB05] M. Balasubramanian, S.S. Iyengar, J. Reynaud, and R.W. Beuerman. A ringing metric to evaluate the quality of images restored using iterative deconvolution algorithms. In *International Conference on on Systems Engineering (ICSEng)*, 2005.

[BK99] S. Baker and T. Kanade. Super resolution optical flow. Technical Report CMU-RI-TR-99-36, Robotics Institute, Carnegie Mellon University, October 1999.

[BK02] S. Baker and T. Kanade. Limits on Super-Resolution and How to Break Them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, September 2002.

[BL03] M. Brown and D.G. Lowe. Recognising Panoramas. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1218–1225, October 2003.

[BMC09] V. Bonato, E. Marques, and G. A. Constantinides. A floating-point extended kalman filter implementation for autonomous mobile robots. *Journal of Signal Processing Systems*, 56(1):41–50, July 2009.

[Bou02] J.-Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm. Microprocessor Research Labs, Intel Corporation, 2002.

[Bro92] L. G. Brown. A Survey of Image Registration Techniques. *ACM Computing Surveys (CSUR)*, 24(4):325–376, December 1992.

[BW05] A. Bruhn and J. Weickert. Towards ultimate motion estimation: combining highest accuracy with real-time performance. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 749–755, October 2005.

[Can76] M. Cannon. Blind deconvolution of spatially invariant image blurs with phase. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(1):58–63, February 1976.

[Cas08] Cast, Inc., Woodcliff Lake, NJ. http://www.cast-inc.com, December 2008.

[CCA+07] M. Cagnazzo, F. Castaldo, T. Andre, M. Antonini, and M. Barlaud. Optimal motion estimation for wavelet motion compensated video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7):907–911, July 2007.

[CCGW00] T. Chen, P. Catrysse, A. E. Gamal, and B. Wandell. How Small Should Pixel Size Be? In *SPIE Sensors and Camera Systems for Scientific, Industrial and Digital Photography Applications*, volume 3965, pages 451–459, January 2000.

[CCLW05] B. Cope, P. Y. K. Cheung, W. Luk, and S. Witt. Have GPUs made FPGAs redundant in the field of video processing? In

IEEE International Conference on Field-Programmable Technology (FPT), pages 111–118, December 2005.

[CCSS01] Y.-Y. Chuang, B. Curless, D.H. Salesin, and R. Szeliski. A Bayesian approach to digital matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271, December 2001.

[CDSY98] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet Transforms that Map Integers to Integers. *Journal of Applied Computational Harmonics Analysis*, 5(3):332–369, July 1998.

[CDT06] T. G. Constandinou, P. Degenaar, and C. Toumazou. An Adaptable Foveating Vision Chip. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3566–3569, May 2006.

[Cel05] Celoxica. *Handel-C Language Reference Manual*. 2005.

[Cel08] Celoxica. http://www.celoxica.com, December 2008.

[CHK+07] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon. A spatial-temporal multi-resolution cmos image sensor with adaptive frame rates for moving objects in the region-of-interest. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 502–618, February 2007.

[CLL+05a] G. M. Callico, S. Lopez, J. F. Lopez, R. Sarmiento, and A. Nunez. Low-cost implementation of a super-resolution algorithm for real-time video applications. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.

[CLL05b] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, October 2005.

[CLS+08] G. Callico, S. Lopez, O. Sosa, J. F. Lopez, and R. Sarmiento. Analysis of fast block matching motion estimation algorithms for video super-resolution systems. *IEEE Transactions on Consumer Electronics*, 54(3):1430–1438, August 2008.

[CO00] C. Chrysafis and A. Ortega. Line-based, reduced memory, wavelet image compression. *IEEE Transactions on Image Processing*, 9(3):378–389, March 2000.

[Con01] G. A. Constantinides. *High Level Synthesis and Word Length Optimization of Digital Signal Processing Systems*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, September 2001.

[CTE91] M. M. Chang, A. M. Tekalp, and A. T. Erdem. Blur identification using the bispectrum. *IEEE Transactions on Signal Processing*,

39(10):2323–2325, October 1991.

[CTHC05] P.-C. Tseng C.-T. Huang and L.-G. Chen. Analysis and VLSI Architecture for 1-D and 2-D Discrete Wavelet Transform. *IEEE Transactions on Signal Processing*, 53(4):1575–1586, April 2005.

[CVO96] C. Chakrabarti, M. Vishwanath, and R. M. Owens. Architectures for wavelet transforms: A survey. *Journal of VLSI Signal Processing*, 14(2):171–192, November 1996.

[CY06] L. Chen and K.-H. Yap. Efficient Discrete Spatial Techniques for Blur Support Identification in Blind Image Deconvolution. *IEEE Transactions on Signal Processing*, 54(4):1557–1562, April 2006.

[dBNS96] A. del Bimbo, P. Nesi, and J. L. C. Sanz. Optical flow computation using extended constraints. *IEEE Transactions on Image Processing*, 5(5):720–739, May 1996.

[DC97] S. Dewitte and J. Cornelis. Lossless Integer Wavelet Transform. *IEEE Signal Processing Letters*, 4(6):158–160, June 1997.

[DGLC03] G. Dillen, B. Georis, J.-D. Legat, and O. Cantineau. Combined Line-Based Architecture for the 5-3 and 9-7 Wavelet Transform of JPEG2000. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(9):944–950, September 2003.

[DHT+00] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In

Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 145–156, July 2000.

[DMBS98] S. J. Decker, R. D. McGrath, K. Brehmer, and C. G. Sodini. A 256x256 CMOS imaging array with wide dynamic range pixels and column-paralleldigital output. *IEEE Journal of Solid-State Circuits*, 33(12):2081–2091, December 1998.

[DRP+06] J. Diaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota. FPGA-based real-time optical-flow system. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):274–279, February 2006.

[DS98] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4(3):247–269, May 1998.

[DS07] O. Dandekar and R. Shekhar. FPGA-Accelerated Deformable Image Registration for Improved Target-Delineation During CT-Guided Interventions. *IEEE Transactions on Biomedical Circuits and Systems*, 1(2):116–127, June 2007.

[ED04] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics (TOG)*, 23(3):673–678, August 2004.

[FEM04] S. Farsiu, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, October 2004.

[Fov08] Foveon, Inc. Foveon X3 Image Sensor. http://www.foveon.com, December 2008.

[FREM04] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Advances and Challenges in Super-Resolution. *International Journal of Imaging Systems and Technology*, 14(2):47–57, August 2004.

[FXK06] J. Farrell, F. Xiao, and S. Kavusi. Resolution and Light Sensitivity Tradeoff with Pixel Size. In *SPIE Electronic Imaging '06 Conference*, volume 6069, pages 211–218, February 2006.

[GE05] A. E. Gamal and H. Eltoukhy. CMOS image sensors. *IEEE Circuits & Devices Magazine*, 21(3):6–20, May-June 2005.

[Gen09] General Electric, Brussels, Belgium. http://www.gesecurity.net/, July 2009.

[GNVV04] Z. Guo, W. Najjar, F. Vahid, and K. Vissers. A quantitative analysis of the speedup factors of fpgas over processors. In *International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 162–170, February 2004.

[Gos05] A. A. Goshtasby. Fusion of multi-exposure images. *Image and Vision Computing*, 23(6):611–618, June 2005.

[GS03] D. Gibson and M. Spann. Robust optical flow estimation based on a sparse motion trajectory set. *IEEE Transactions on Image Processing*, 12(4):431–445, April 2003.

[GYB04] S.B. Gokturk, H. Yalcin, and C. Bamji. A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 35–43, June 2004.

[Hay08] B. Hayes. Computational Photography. *American Scientist*, 96(2):94–99, March-April 2008.

[HCHT04] W.-J. Hwang, J.-F. Chen, Y.-C. Huang, and T.-Y. Tsai. Layered video coding based on displaced frame difference prediction and multiresolution block matching. *IEEE Transactions on Communications*, 52(9):1504–1513, September 2004.

[Ini03] Open SystemC Initiative. *SystemC 2.0.1 Language Reference Manual*. 2003.

[IP91] M. Irani and S. Peleg. Improving Resolution by Image Registration. *CVGIP: Graphical Models and Image Proc*, 53(3):231–239, May 1991.

[ISO98] ISO/IEC JTC1/SC29/WG11, FCD 14496-1. Coding of moving pictures and audio, May 1998.

[ISO00] ISO/IEC FCD15444-1: 2000. JPEG 2000 image coding system, May 2000.

[ITU08] ITU-T Recommentation T.800. JPEG2000 Image Coding System Part I, ITU Std, July 2002. [Available from] http://www.itu.int/ITU-T/, December 2008.

[Jia08] Q. Jiang. FIR Filter Banks for Hexagonal Data Processing. *IEEE Transactions on Image Processing*, 17(9):1512–1521, September 2008.

[JLD06] H. Jiang, Z.-N. Li, and M. S. Drew. Detecting human action in active video. In *IEEE International Conference on Multimedia and Expo*, pages 1497–1500, July 2006.

[JLR03] J. Jiang, W. Luk, and D. Rueckert. FPGA-based computation of free-form deformations in medical image registration. In *IEEE International Conference on Field-Programmable Technology (FPT)*, pages 234–241, December 2003.

[KDS+00] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts. A logarithmic response CMOS image sensor with on-chip calibration. *IEEE Journal of Solid-State Circuits*, 35(8):1146–1152, August 2000.

[KH96] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, May 1996.

[KHDO06] E. J. Kelmelis, J. R. Humphrey, J. P. Durbano, and F. E. Ortiz. High-performance computing with desktop workstations. In *WSEAS International Conference on Applied Mathematics*, pages 83–88, November 2006.

[KN06] S. Kuthirummal and S. K. Nayar. Multiview Radial Catadioptric Imaging for Scene Capture. *ACM Transactions on Graphics (TOG)*, 25(3):916–923, July 2006.

[Lat06] Lattice Semiconductor Corporation. Developping High-Speed Memory Interfaces. February 2006.

[LB08] T. H. Le and L. T. Bui. A hybrid approach of adaboost and artificial neural network for detecting human faces. In *IEEE International Conference on Research, Innovation and Vision for the Future (RIVF)*, pages 79–85, July 2008.

[LCT+08] S. Lopez, G. Callico, F. Tobajas, J. Lopez, and R. Sarmiento. A flexible template for h.264/avc block matching motion estimation architectures. *IEEE Transactions on Consumer Electronics*, 54(2):845–851, May 2008.

[Lim90] J. S. Lim. *Two-Dimensional Signal and Image Processing*, page 548. Prentice Hall, 1990.

[LLW08] A. Levin, D. Lischinski, and Y. Weiss. A Closed-Form Solution to Natural Image Matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, February 2008.

[LNB+99] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens. Optimal Memory Organization for Scalable Texture Codecs in MPEG-4. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(2):218–243, March 1999.

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, January 2004.

[LRAL08] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral Matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1–14, October 2008.

[Luc74] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79(6):745–754, June 1974.

[Mal89] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(7):674–693, July 1989.

[MAS06] K. Masselos, Y. Andreopoulos, and T. Stouraitis. Performance comparison of two-dimensional discrete wavelet transform computation schedules on a VLIW digital signal processor. *IEE Proceed-*

ings Vision, Image and Signal Processing, 153(2):173–180, April 2006.

[McE06] M. McErlean. An FPGA Implementation of Hierarchical Motion Estimation for Embedded Object Tracking. In *IEEE International Symposium on Signal Processing and Information Technology*, pages 242–247, August 2006.

[McI01] L. G. McIlrath. A low-power low-noise ultrawide-dynamic-range CMOS imager with pixel-parallel A/D conversion. *IEEE Journal of Solid-State Circuits*, 36(5):846–853, May 2001.

[MCL02] W. J. C. Melis, P. Y. K. Cheung, and W. Luk. Image Registration of Real-Time Video Data Using the SONIC Reconfigurable Computer Platform. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1148–1151, September 2002.

[Mic08] Micron Technology, Inc. SDRAM System-Power Calculator. [Available from] http://www.micron.com/support/designsupport/tools/, December 2008.

[MJ07] M. E. Moghaddam and M. Jamzad. Motion blur identification in noisy images using mathematical models and statistical measures. *Pattern Recognition*, 40(7):1946–1957, July 2007.

[MK98] J. Magarey and N. Kingsbury. Motion estimation using a complex-valued wavelet transform. *IEEE Transactions on Signal Processing*, 46(4):1069–1084, April 1998.

[MKG+97] S. Mendis, S. Kemeny, R. Gee, B. Pain, C. Staller, Q. Kim, and E. Fossum. CMOS Active Pixel Image Sensors for Highly Integrated Imaging Systems. *IEEE Journal of Solid-State Circuits*, 32(2):187–197, February 1997.

[MP94] S. Mann and R.W. Picard. Virtual bellows: constructing high quality stills from video. In *IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 363–367, November 1994.

[MV98] J. B. A. Maintz and M. A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, March 1998.

[Nay06] S. K. Nayar. Computational Cameras: Redefining the Image. *Computer*, 39(8):30–38, August 2006.

[New82] B. Newhall. *History of Photography*. Bulfinch, 1982.

[NM00] S. K. Nayar and T. Mitsunaga. High dynamic range imaging: spatially varying pixel exposures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 472–479, June 2000.

[NP99] S. K. Nayar and V. Peri. Folded catadioptric cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 217–223, May 1999.

[NRBC04] C. Niclass, A. Rochas, P.-A. Besse, and E. Charbon. A CMOS single photon avalanche diode array for 3D imaging. In *IEEE International Solid-State Circuits Conference*, volume 1, pages 120–129, February 2004.

[OBSC00] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons, 2000.

[PHZ08] J. Pan, B. Hu, and J. Q. Zhang. Robust and Accurate Object Tracking Under Various Types of Occlusions. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(2):223–236, February 2008.

[PO06] E. Polat and M. Ozden. A nonparametric adaptive tracking algorithm based on multiple feature distributions. *IEEE Transactions on Multimedia*, 8(6):1156–1163, December 2006.

[PPK03] S. C. Park, M. K. Park, and M. G. Kang. Super-Resolution Image Reconstruction: A Technical Overview. *IEEE Signal Processing Magazine*, 20(3):21–36, May 2003.

[PSA+04] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (TOG)*, 23(3):664–672, August 2004.

[PVW06] P. Peursum, S. Venkatesh, and G. West. Observation-switching linear dynamic systems for tracking humans through unexpected partial occlusions by scene objects. In *International Conference on Pattern Recognition (ICPR)*, volume 4, pages 929–934, August 2006.

[RBS99] M. A. Robertson, S. Borman, and R. L. Stevenson. Dynamic range improvement through multiple exposures. In *IEEE Conference on Image Processing (ICIP)*, volume 3, pages 159–163, October 1999.

[RT00] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 18–25, May 2000.

[RTM+06] R. Raskar, J. Tumblin, A. Mohan, A. Agrawal, and Y. Li. EUROGRAPHICS 2006 STAR State of the Art Report Computational Photography, 2006.

[SCI05] E. Shechtman, Y. Caspi, and M. Irani. Space-Time Super-Resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–545, April 2005.

[SO89] H. Stark and P. Oskoui. High-resolution image recovery from image-plane arrays, using convex projections. *Journal of the Optical Society of America A*, 6(11):1715–1726, November 1989.

[SPBDK01] A. Stern, Y. Porat, A. Ben-Dor, and N. S. Kopeika. Enhanced-resolution image restoration from a sequence of low-frequency vibrated images by use of convex projections. *Applied Optics*, 40(26):4706–4715, September 2001.

[ST94] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, June 1994.

[TAR05] J. Tumblin, A. Agrawal, and R. Raskar. Why i want a gradient camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 103–110, June 2005.

[TFG01] H. Tian, B. Fowler, and A. E. Gamal. Analysis of Temporal Noise in CMOS Photodiode Active Pixel Sensor. *IEEE Journal of Solid-State Circuits*, 36(1):92–101, January 2001.

[TGP02] G. Tirunelveli, R. Gordon, and S. Pistorius. Comparison of square-pixel and hexagonal-pixel resolution in image processing. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2002.

[TKW86] A. Tekalp, H. Kaufman, and J. Woods. Identification of image and blur parameters for the restoration of noncausal blurs. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):963–972, August 1986.

[TLZZ04] H. Tong, M. Li, H. Zhang, and C. Zhang. Blur detection for digital images using wavelet transform. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, pages 17–20, June 2004.

[VH92] M. Vetterli and C. Herley. Wavelets and filter banks: theory and design. *IEEE Transactions on Signal Processing*, 40(9):2207–2232, September 1992.

[WB] G. Welch and G. Bishop. An introduction to the Kalman filter, July 2006. [Available from] http://www.cs.unc.edu/ welch/media/pdf/kalman_intro.pdf, December 2008.

[WB03] M. Weeks and M. Bayoumi. Discrete Wavelet Transform: Architectures, Design and Performance Issues. *Journal of VLSI Signal Processing*, 35(2):155178, September 2003.

[WGT+05] A. Wenger, A. Gardner, C. Tchou, J. Unger, T. Hawkins, and P. Debevec. Performance relighting and reflectance transforma-

tion with time-multiplexed illumination. *ACM Transactions on Graphics (TOG)*, 24(3):756–764, July 2005.

[WJV+05] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Transactions on Graphics (TOG)*, 24(3):765–776, July 2005.

[Xil] Xilinx, Inc. Virtex-4 FPGA User Guide.

[Xil04] Xilinx, Inc. Celebrating 20 Years of Innovation. *Xilinx Xcell Journal*, 48:14–16, 2004.

[Xil08] Xilinx, Inc., San Jose. http://www.xilinx.com, December 2008.

[YJS06] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, December 2006.

[YK97] Y. Yitzhaky and N. S. Kopeika. Identification of Blur Parameters from Motion Blurred Images. *Graphical Models and Image Processing*, 59(5):310–320, September 1997.

[YW82] D. C. Youla and H. Webb. Image Restoration by the Method of Convex Projections: Part 1-Theory. *IEEE Transactions on Medical Imaging*, MI-1(2):81–94, October 1982.

[ZAS+01] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos, and C. E. Goutis. Evaluation of design alternatives for

the 2-D-discrete wavelet transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12):1246–1262, December 2001.

[ZAS06] J. Zan, M. O. Ahmad, and M. N. S. Swamy. Comparison of wavelets for multiresolution motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):439–446, March 2006.

[ZF03] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, October 2003.

[ZSA01] J. Zan, M. N. S. Swamy, and M. O. Ahmad. Wavelet filters in multi-resolution motion estimation. In *Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 1321–1326, May 2001.

[ZTCS99] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, August 1999.

[ZWYF08] W. Zhang, Q. M. J. Wu, X. Yang, and X. Fang. Multilevel Framework to Detect and Handle Vehicle Occlusion. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):161–174, March 2008.

[ZZ92] Y.-Q. Zhang and S. Zafar. Motion-compensated wavelet transform coding for color video compression. *IEEE Transactions on Cir-*

cuits and Systems for Video Technology, 2(3):285–296, September 1992.