Architecture and Implementation of a High Frame-Rate Stereo Vision System

by

Jamin Islam, B.Eng

Ryerson University, Toronto, Canada, 2006

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2008

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Jamin Islam

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Jamin Islam

Abstract

Architecture and Implementation of a High Frame-Rate Stereo Vision System

Jamin Islam

Master of Applied Science

Graduate Program of Electrical and Computer Engineering

Ryerson University

2008

For the purpose of autonomous satellite grasping, a high-speed, low-cost stereo vision system is required with high accuracy. This type of system must be able to detect an object and estimate its range. Hardware solutions are often chosen over software solutions, which tend to be too slow for high frame-rate applications. Designs utilizing field programmable gate arrays (FPGAs) provide flexibility and are cost effective versus solutions that provide similar performance (*i.e.*, Application Specific Integrated Circuits). This thesis presents the architecture and implementation of a high frame-rate stereo vision system based on an FPGA platform. The system acquires stereo images, performs stereo rectification and generates disparity estimates at frame-rates close to 100 fps; and on a large-enough FPGA, it can process 200 fps. The implementation presents novelties in performance and in the choice of the algorithm implemented. It achieves superior performance to existing systems that estimate scene depth. Furthermore, it demonstrates equivalent accuracy to software implementations of the dynamic programming maximum likelihood stereo correspondence algorithm.

Acknowledgments

There are many people that I would like to thank for making this thesis possible. First, my deepest gratitude goes out to my co-supervisors, in no particular order, W. James MacLean and Lev Kirischian. Thank you both for taking me as your student, I was lucky to have you as mentors. James, I enjoyed our weekly meetings and learning about computer vision. I am in debt to you for all the comments and advice that you had to share. Without you, this work would not be as well written as it is now. Lev, I appreciated all your efforts as our *manager* at the Embedded and Reconfigurable Systems Lab. Your stories and enthusiasm were extremely helpful and motivating when things didn't seem to go as they should. I hope that I am closer to becoming that ripe red tomato! I'd like to further extended my gratitude towards Professors Michael Greenspan and Vadim Geurkov.

I'd also like to thank the supporting cast of the FastTrack team: Valeri Kirischian, Peter (Pil Woo) Chun, Siraj Sabihuddin, Sergei Zhelnakov and Michael Belshaw. My experience on this project truly allowed me to learn what *team-work* means. This experience was great, I learned a lot from each one of you. Furthermore, I'd like to acknowledge the Ontario Centres of Excellence (CITO), MDA Space Missions, CMC Microsystems and the Department of Electrical and Computer Engineering at Ryerson University for their financial contributions with this work.

Finally, I'd like to thank my family for supporting and encouraging me to pursue this degree. This thesis is dedicated to them, especially to my dad, who gave us all a great scare with his health. There were times when I didn't know if I would be able continue, however, I'm glad that you're okay now and I hope we enjoy the many years ahead.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer vision is an important technology for autonomous space robotics. Consider the following scenarios: 1) When satellites are in orbit they face the risk of colliding with debris that can cause structural damage. It is often more economical to repair the damages that occur from these types of events, rather than to deploy brand new equipment into orbit. New emerging research allows developers to design modules that can be added to the satellites to collect more meaningful information. However, in order to perform repairs or upgrades, the satellite, much like a car being serviced on a hoist, must be linked to a base station. Due to the relative lack of humans in outer space, the satellites are required to have the base stations rendezvous and dock autonomously to them. 2) When satellites are launched in space, they are supplied with enough fuel to keep them in orbit for a number of years before the reserve depletes. When the fuel reserve does deplete, the satellites begin to move into the atmosphere where they being to corrode and become debris. This situation can be avoided, while increasing the lifetime of the device by a routine refueling process. Similar to the first scenario, the satellites are required to have the base stations, containing the fuel, rendezvous and dock autonomously to them. 3) While satellites are in orbit, preventative maintenance steps can be taken to ensure that the device does not fail (*e.g.*, replacing the gyroscope, thrusters, *etc.*). Much like the first and second scenario, the satellite and base

station required docking.

Each of the scenarios described above requires using a three-dimensional (3-D) computer vision system. The stereo vision system is needed to detect a target satellite from a distance, and in an arbitrary orientation, to guide the service module to the docking port of the satellite.

1.1 Motivation

This thesis represents part of a project, known as FastTrack, to develop a stereo vision system to track and follow moving objects (*e.g.*, satellites) at fast speeds, equivalent to 200 frames per second (fps). The deployment of this system is targeted for space applications, but many other applications exist. This 3-D computer vision system requires having high frame-rate image capturing to measure the motion of the satellite in very small increments, while preventing motion blur, which often occurs on frame-rate (*i.e.*, 30 fps) capturing systems when objects are moving at high-speeds. As an example, light detection and ranging (LiDAR) technology is currently used on the Canadarm2 to guide and control spacecraft docking. This technology works well for objects in long ranges, however its accuracy is poor for shorter ranges (between zero to five metres). The system proposed in this thesis can be used to resolve the accuracy problem for short ranges.

1.2 Objectives

This project has been divided among three research groups, where each group is responsible for designing a portion of the system. The research team from Ryerson University is responsible for the design and development of the high frame-rate stereo vision camera, while research teams from the University of Toronto and Queen's University are responsible for the hardware implementations of stereo correspondence and object tracking algorithms, respectively. Each portion of the system is to be designed, tested and verified on a field pro-

grammable gate array (FPGA). These FPGAs contain arrays of reconfigurable logic blocks that allow for rapid hardware prototyping at relatively low costs. Other methods, such as application specific integrated circuits (ASICs), often perform better and consume less power, but are compromised by large costs and lengthy design periods. The complete system is to be implemented on one FPGA-based platform. This requires interfacing and integrating the stereo camera and the hardware algorithms to the processing platform.

One objective of this thesis is to develop the architecture of a high frame-rate stereo image rectification module, which places corresponding points (features) on the same scanline in both images. Another objective of this work is to combine the individual portions of the system together, to form the high frame-rate stereo vision system.

1.3 Contributions

In this work, we develop a 3-D vision system that achieves benchmarks that no other state-of-the-art frame-rate vision system accomplishes. This system presents novelty in its performance, as it achieves superior frame-rates (>100 fps) to existing systems that perform stereo matching in an attempt to estimate depth in the scene. Furthermore, the system is also novel in its quality as compared to other frame-rate systems, as the stereo correspondence algorithm used is globally optimal over a scanline [1], while existing frame-rate implementations use local algorithms. The best stereo matching results are obtained using graph cuts algorithms [2], however, they are extremely difficult, if not impossible, to implement efficiently on FPGAs. For example, a graph-cuts algorithm could be designed using a soft-processor on an FPGA, however, its likely that it wouldn't even approach frame-rate performances. The contributions made to this project, which are discussed in later stages of this thesis, include:

- the on-chip design of a high frame-rate stereo image rectification module,
- interfacing the video-acquisition system with the stereo image rectification module,
- assisting in hardware modifications to the stereo extraction module,

- developing the interface between the stereo image rectification module and the stereo extraction module,
- the integration of the stereo vision system on the FPGA-based platform and analysis of results.

1.4 Thesis Organization

This document begins in Chapter 2 with an introduction to the human vision system, camera systems, FPGAs and a review of recent literature. Following this, Chapter 3 explores the hardware architecture and implementation of our state-of-the-art stereo vision system. The chapter describes a high-speed implementation of a stereo image rectification module. It also overviews the video-acquisition system and our processing platform. Furthermore, the chapter describes the integration of the system with an FPGA implementation of a dynamic programming maximum likelihood (DPML) stereo matching algorithm developed by Sabihuddin *et al.* [3]. In Chapter 4, the results obtained from the hardware implementations are evaluated. Chapter 5 concludes with possible directions for future work associated with this project.

Chapter 2

Theory and Related Works

2.1 Introduction

This chapter contains background information for many of the concepts and ideas discussed in later chapters of this thesis. Section 2.2 explores how the brain and eyes interact to form the human vision system. Sections 2.3 and 2.4 introduce a background on camera modeling and stereo vision systems, which are used to model the human vision system. Section 2.5 briefly discusses FPGAs and VHDL. Section 2.6 presents a literature review of previous hardware implementations of image rectification, stereo correspondence and stereo vision systems. Finally, Section 2.7 summarizes the chapter's contents.

2.2 The Human Vision System

This section is included in this thesis as an attempt to explain the contents in the rest of the chapter in a more meaningful way. The study of the human vision system gives us an idea of how images of the scene are captured by the eye and processed by the brain. This knowledge will allow us to use machines to implement similar functionalities. Furthermore, knowledge of how the human vision system uses the scene information to estimate depth will give us an idea of how to implement this type of functionality with a machine. This study

also gives us an idea of the complexity of performing this sort of information.

The human vision system is composed of interconnections between the brain and the eyes. The human brain is a very complex organ in nature, as humans have the ability to see, breathe, cough, sneeze, vomit, mate, swallow, urinate and think. We can also do arithmetic, speak, write, sing and compose poems. We even have the ability to play sports and play musical instruments. The brain contains approximately 10^{12} cells, which is an astronomical number compared to any other organ in the body. Better evidence of the brain's complexity is seen by the number of interconnections it possesses. A typical nerve cell in the brain receives information from hundreds or thousands of other nerve cells and in turn transmits information to hundreds or thousands of other cells. The total number of interconnections in the brain should therefore be somewhere around 10^{14} to 10^{15} [4]. A limited amount of the brain's functionality is understood, however a good understanding about the machinery of the visual system and how the brain interprets this information exists.

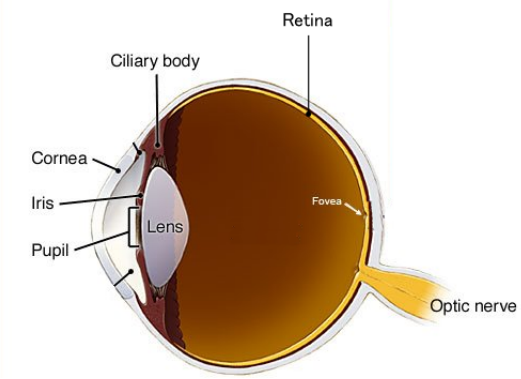


Figure 2.1: The human eye [5]

The design of our eyes allows us to have high resolution vision, as a result of the 125 million receptors in the retina. The light focuses onto these receptors, which are known as rods and cones. These receptors are nerve cells specialized to emit electrical signals when photons hit them. The task of the rest of the retina and the brain is to properly make sense of these signals and to extract information that is useful to us. An illustration of the human eye is shown in Figure 2.1. The function of the non-retinal parts of the eye is to keep a clear,

focused image of the outside world stabilized on the two retinas. The cornea and the lens together form the equivalent of the camera lens. Two-thirds of the eyes' focusing power is supplied at the air-cornea interface, while the remaining one-third is supplied by the lens. The main job of the lens is to make adjustments and focus on objects at various distances. For cameras, the focus is achieved by changing the distance between the lens and the film. We focus our eyes, not by changing the distance between the lens and the retina, but by changing the shape of the lens using *cilliary muscles*. The iris muscle changes the diameter of the pupil in the eye, thus it adjusts the amount of light allowed in. The blinking of the eyelids and the lubrication from the tear glands serves as a self-cleaning mechanism for the front cornea.

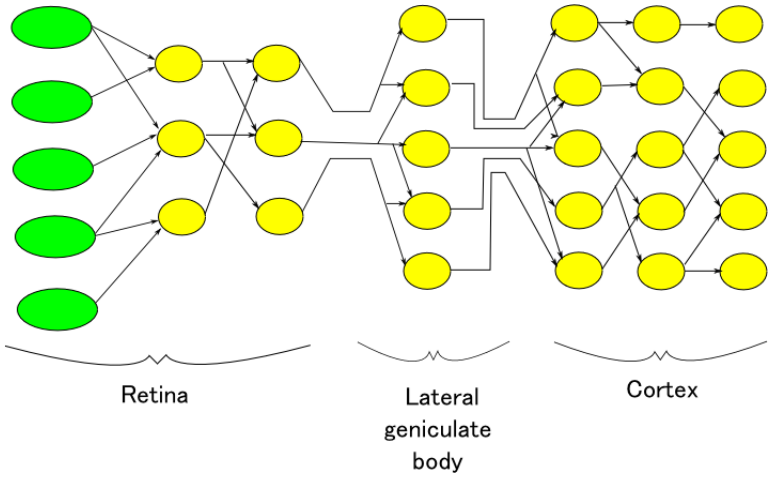


Figure 2.2: The visual pathway

The region of the brain which deals with the visual system is known as the *primary visual cortex*, or *striate cortex*. This part is perhaps the best understood part of the brain today, as it is known what each of its nerves are doing at most times in a person's everyday life, and what each nerve contributes to the analysis of the visual information. The primary visual cortex is a serially connected system of neurons. This visual pathway is shown in Figure 2.2. The retina of each eye consists of a plate having three layers of cells, one of which contains the light-sensitive receptor cells. The two retinae send their output to two peanut-sized nests of cells deep within the brain known as the *lateral geniculate body*. These structures in turn

send their fibres to the striate cortex. After being passed from layer to layer through several sets of synaptically connected cells, the information is sent to several neighbouring higher visual areas; each of these sends its output to several yet-higher cortical areas.

Stereopsis is a strategy used for judging the depth by comparing the images on our two retinas. The images cast on our retinas are two-dimensional, however we look out onto a three-dimensional world. Having a sense of how far away things are from us is important for many of the things we do in our everyday lives. We judge depth in many ways. One way is when we roughly know the size of something, such as a person, tree or cat, we can judge its distance from its apparent size in the image. Another way is if one object is partly in front of another and blocks its view, we judge the front object as closer. A powerful indicator of depth (known as perspective) occurs when the image of parallel lines, like railroad tracks, draw together as they go off into the distance. Stereopsis is perhaps the most important mechanism for assessing depth and is dependent on the use of the two eyes together. In any scene with depth our eyes receive two slightly different images. The brain compares them and estimates relative depths with great accuracy.

The term *disparity* is the computed difference in retinal position between corresponding points. When we stare at an object it is imaged on the fovea of each eye, which is responsible for high resolution vision. Scene points which are imaged on the same point of both retinae (*i.e.*, foveae) are said to have zero disparity. Closer and more distant objects are imaged on different parts of each retina and are said to have retinal disparity. The retinal disparity can be either positive or negative. Positive disparity means that a scene point images further to the right in the right image than it does in the left image. Negative disparity means that a scene point images further to the left in the left images than it does in the right image. Objects further away from the fixated scene point are said to have a positive disparity, while closer objects are said to have a negative disparity.Figure 2.3 illustrates the relative position of given scene points on the retinae. It can be noticed at point P_a, the relative position on the retinae are equal and this represents zero disparity ($P_a^R - P_a^L = 0$). The relative

position on the retinae of point P_b are less than zero and this represents a negative disparity ($P_b^R - P_b^L < 0$). Furthermore, point P_b is comparatively closer than point P_a. Finally, when we look at point P_c, we notice that it is further away than

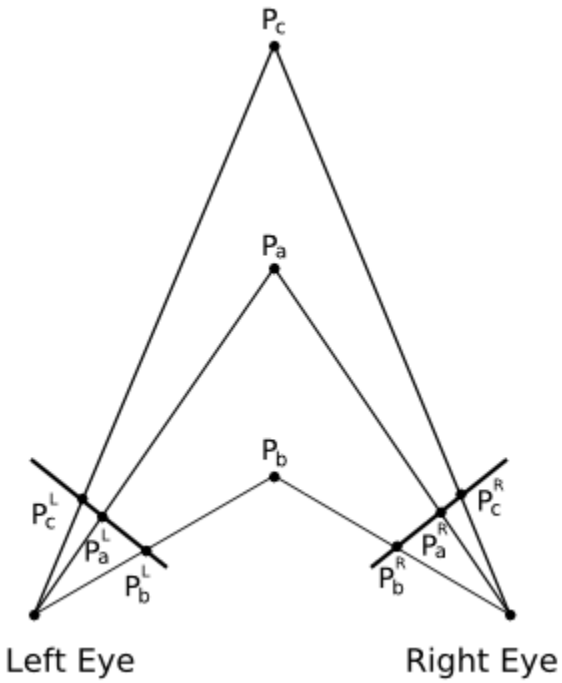


Figure 2.3: The relative position of given scene points on the retinae

point P_a and accordingly has a positive disparity. The relative position of the point P_c on the retinae is greater than zero ($P_c^R - P_c^L > 0$). If we extract this information about retinal disparity we then have valuable information about depth relative to where we are looking. Things with a negative disparity are closer and those with positive disparities are further away. Furthermore, the greater the disparity, the closer or further away the object will be [6].

The physiology of stereopsis uses the information presented about retinal disparity as a major cue to depth. We should therefore expect to find cells within the brain that are sensitive to retinal disparity. The first place where information from both eyes come together is in the striate cortex and it has been shown that many of these cells are sensitive to retinal disparity of the images in the two eyes [4]. Figure A.1 (see Appendix A) shows the firing rates

of four cells as a function of the retinal disparity. The first column in this figure represents the position of two objects and defines the disparity of the test area. We can see that the cell illustrated in the second column responded vigorously to small disparities (particularly, slightly negative ones) but not to large disparities. These types of cells are known as *tuned excitatory*. Other cells fire only to large disparities and are termed *tuned inhibitory*, as seen in the third column in the illustration. Some cells fired most when the test stimulus was further away from the fixation point (*i.e.*, positive disparity) and this type of response is shown in the fourth column and is termed a *far* cell. There are similar cells which respond well to negative disparities, shown in the last column and are termed *near* cells. Please refer to the work published by Hubel in [4] for more information about the visual interaction of the brain and the eyes.

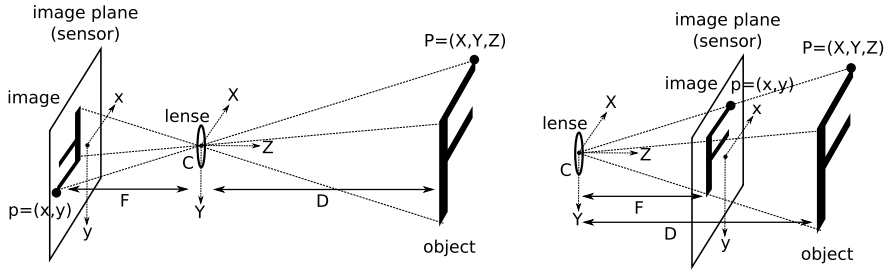
2.3 Camera Modeling

In computer vision systems we must rely on cameras to capture images of the scene at hand and thus must understand the image formation process. Section 2.3.1 discusses the geometry of a single-view pinhole camera model. Section 2.3.2 describes a calibration algorithm which is used for obtaining the camera parameters mentioned in Section 2.3.1. Section 2.3.3 describes using the Camera Calibration Toolbox for Matlab which is a common tool used for obtaining the camera parameters. Section 2.3.4 discusses a stereo camera model, which is important for solving the problem of stereo correspondence (see Section 2.4.2). Finally, Section 2.3.5 discusses a special case of the Camera Calibration Toolbox for Matlab which deals with stereo calibration.

2.3.1 The Camera Model: Single-view Geometry

Modeling cameras is commonly done using the *pinhole camera model*. The pinhole camera requires no lens, but uses instead a very small aperture. Such cameras require excessively long

exposure times in reality, but provide a good model for perspective projection. Perspective projection is the process of transforming the three-dimensional visual information received by the camera into a two-dimensional (2-D) image. Figure 2.4a shows an illustration of the pinhole camera model. From this, it can be seen that rays of light pass through a lens and intersect with an image plane. The image plane is usually a light sensitive film, or more recently, digital image sensors. It is common to model the image plane as being in front of the lens for mathematical convenience, as shown in Figure 2.4b.



(a) A pinhole camera model

(b) A modified pinhole camera model

Figure 2.4: The pinhole camera model

In Figure 2.4, f represents the focal length of the camera lens, while D represents the distance of the object. The focal length and distance measurements are typically relative to the camera centre, C. The optic axis of the lens assembly is perpendicular to the image plane and passes through the camera centre. The image is then projected on to the image plane. It should be noted that the image plane is described in reference to the 2-D image coordinate system. The point of intersection of the optic axis with the image plane is called the principle point or image centre. The perspective transformation of a 3-D point, $P = (X, Y, Z)^T$, on to a 2-D image, $p = (x, y)^T$ is given by Equation 2.1.

$$-\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.1}$$

This equation can be re-written as a set of equations in homogeneous coordinates (with $\overline{p} = \left[p^T|1\right]^T$ and $\bar{P} = \left[P^T|1\right]^T$) as shown in Equation 2.2. It should also be noted that

$\lambda = 1/Z$ in Equation 2.2.

$$\overline{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \lambda K \left[I_3 | 0 \right] \overline{P} \tag{2.2}$$

The pinhole camera model gives us a good starting point, however it deviates from real camera systems. The pinhole camera model assumes that the optic axis and the image plane are perfectly perpendicular in the alignment of the lens assembly. However, in reality the alignment of the lens can never be exact and the lack of perpendicularity causes skew distortions [7]. Other examples of distortions appearing from the lens assembly are scale, radial and tangential distortions. Assuming that there is no radial or tangential distortions, the pinhole camera model maps lines to lines, but does not preserve lengths, ratios or angles.

The distortions that may appear due to the lens assembly can be corrected for by using parameters that modify Equation 2.2. These parameters are modeled in Equation 2.3 and are known as the *intrinsic* parameters of the camera. The matrix $\overline{K}$, forms what is known as the camera calibration matrix. This includes the skew parameter, s, camera principal point (x_o, y_o) and scaling factors k_x and k_y (in millimeters). The radial distortion, $L(\bar{r})$, presented by the lens assembly, is a non-linear function of the radius, $\overline{r}$, where $\overline{r}$ is relative to the principal point. The parameters of this distortion model are considered part of the intrinsic parameters. The tangential distortion component is omitted from the model as it is typically not a significant contributor to distortion in the resulting image.

$$\overline{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda L(\overline{r}) \overline{K} \left[I_3 | 0 \right] \overline{P} = \lambda L(\overline{r}) \begin{bmatrix} k_x f & s & x_o \\ 0 & k_y f & y_o \\ 0 & 0 & 1 \end{bmatrix} \left[I_3 | 0 \right] \overline{P} \tag{2.3}$$

The 3-D point coordinates, $P = (X, Y, Z)^T$, are defined in terms of a camera centric-

coordinate system. It is possible to transform the camera centric-coordinate system to some world coordinate system, $\tilde{P} = (\tilde{X}, \tilde{Y}, \tilde{Z})^T$, by knowing the *extrinsic* parameters of the camera. The extrinsic parameters include a rotation, R, and a translation, $\overrightarrow{T}$. These allow one to perform an Euclidean transformation to align the two coordinate systems by forming the relationship seen in Equation 2.4. The extrinsic parameters are defined by the matrix τ. The combination of the intrinsic and extrinsic parameters into a *projection matrix*, P_r, is shown in Equation 2.5.

$$\overline{P} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & \overrightarrow{T} \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{bmatrix} = \tau \begin{bmatrix} \tilde{P} \\ 1 \end{bmatrix} \tag{2.4}$$

$$\overline{p} = \lambda L(\overline{r}) \overline{K} \left[I_3 | 0 \right] \tau \begin{bmatrix} \tilde{P} \\ 1 \end{bmatrix} = P_r \begin{bmatrix} \tilde{P} \\ 1 \end{bmatrix} \tag{2.5}$$

A good reference for more in-depth theory associated with camera modeling and single-view geometry can be found in work by Harley and Zisserman in [8].

2.3.2 Camera Calibration

Camera calibration is the process of estimating the camera parameters of Equations 2.3 and 2.4 through simultaneous measurement of corresponding coordinate pairs $\left\{ \bar{p}_i, \tilde{P}_i \right\}_{i=1}^{N}$. In order to estimate the coordinates $\tilde{P}_i$ of 3-D scene points, camera calibration methods rely on one or more images of a calibration pattern. The calibration pattern is a 3-D object of known geometry, possibly located in a known position in space and generating images features that can be located accurately. A common calibration pattern is a checkerboard. The use of camera calibration procedures opens up the possibility of using a wide range of existing algorithms for 3-D reconstruction and recognition, all relying on the knowledge of

the camera parameters. One method we can use to retrieve the camera parameters is by estimating the projection matrix (see Section 2.3.1), P_r, without solving explicitly for the intrinsic and extrinsic parameters, assuming that $L(\overline{r}) = 1$. This calibration method consists of two sequential stages:

1. estimate the projection matrix linking the world and image coordinates
2. compute the camera parameters as closed-form functions of the entries of the projection matrix

Work by Hartley and Zisserman in [8], recommend "renormalizing" the observed data of both $\tilde{P}_i$ and $\bar{p}_i$. For points in the world coordinate system, they recommend creating a matrix, U, such that for $\ddot{P}_i = U\tilde{P}_i$, then $\frac{1}{N}\sum_i \ddot{P}_i = [0,0,0,1]^T$ and $\sqrt{\frac{1}{N}\sum_i(\ddot{P}_{i1}^2 + \ddot{P}_{i2}^2 + \ddot{P}_{i3}^2)} = 1/\sqrt{2}$. Similarly, for points in the camera coordinate system, they recommend creating a matrix, W, such that for $\ddot{p}_i = W\bar{p}_i$, then $\frac{1}{N}\sum_i \ddot{p}_i = [0,0,1]^T$ and $\sqrt{\frac{1}{N}\sum_i(\ddot{p}_{i1}^2 + \ddot{p}_{i2}^2)} = 1/\sqrt{2}$. As seen in Equation 2.5, the matrix P_r is defined up to an arbitrary scale factor and therefore has only 11 independent entries, which can be determined through a homogeneous linear system formed by writing Equations 2.6 and 2.7 for at least 6 world-image point matches.

$$\bar{p}_1 = \frac{\tilde{p}_{r11}\tilde{P}_1 + \tilde{p}_{r12}\tilde{P}_2 + \tilde{p}_{r13}\tilde{P}_3 + \tilde{p}_{r14}}{\tilde{p}_{r31}\tilde{P}_1 + \tilde{p}_{r32}\tilde{P}_2 + \tilde{p}_{r33}\tilde{P}_3 + \tilde{p}_{r34}} \tag{2.6}$$

However, by using the calibration patterns, many more correspondences and equations can be obtained. This allows estimating using least-squares techniques.

$$\bar{p}_2 = \frac{\breve{p}_{r21}\tilde{P}_1 + \breve{p}_{r22}\tilde{P}_2 + \breve{p}_{r23}\tilde{P}_3 + \breve{p}_{r24}}{\breve{p}_{r31}\tilde{P}_1 + \breve{p}_{r32}\tilde{P}_2 + \breve{p}_{r33}\tilde{P}_3 + \breve{p}_{r34}} \tag{2.7}$$

If we assume that we are given N matches for the homogeneous linear system we have $A\breve{p}_r = 0$ (see Equations 2.8 and 2.9), where $\breve{p}_r$ is the normalized vector of the projection matrix P_r. Since A has rank 11, the vector $\breve{p}_r$ can be recovered from singular value decomposition (SVD) techniques as the column of V corresponding to the zero (in practice the smallest) singular

value of A, with $A = UDV^T$. It is required to de-normalize the subsequent matrix to get $P_r = W^{-1}\breve{P}_r U$.

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_NX_N & -x_NY_N & -x_NZ_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_NX_N & -y_NY_N & -y_NZ_N & -y_N \end{bmatrix} \tag{2.8}$$

$$\breve{p}_r = \begin{bmatrix} \breve{p}_{r11}, & \breve{p}_{r12}, & \ldots, & \breve{p}_{r33}, & \breve{p}_{r34} \end{bmatrix}^T \tag{2.9}$$

With the projection matrix, P_r, recovered we can use an "RQ" decomposition on the first three columns of P_r to compute the camera parameters, $\overline{K}$. Here "R" is an upper-triangular matrix, representing the intrinsic parameters. It should be noted that to get proper scaling, the first two rows will have to be divided by $\overline{K}_{33}$. The "Q" matrix is an orthogonal matrix and represents the rotation, R .

This matrix is guaranteed to be orthogonal (within the limits of numerical precision) due to the properties of the "RQ" decomposition. Once the rotation, R, and the intrinsic parameters are known, we can easily recover the translation, $\overrightarrow{T}$. When $L(\overline{r}) \neq 1$, a further step involving non-linear optimization is required to refine parameter estimates. For further details on performing the SVD and "RQ" decomposition, please refer to [9] and [10], respectively.

2.3.3 The Camera Calibration Toolbox

We utilize the Camera Calibration Toolbox for Matlab, developed by Jean-Yves Bouguet of the California Institute of Technology [11]. The toolbox provides the user with a simple yet

powerful approach for tackling the camera calibration algorithm seen in Section 2.3.2. The only requirements needed for this process is that a set of calibration images are captured from the camera system. The toolbox returns the intrinsic and extrinsic parameters of the camera system to the user. The main calibration procedure is for extracting the intrinsic parameters. The design of the toolbox does not only allow the extraction of data for just one camera, it also allows calibration of binocular stereo systems. Furthermore, the toolbox allows using its calibration engine on calibration data that was previously produced using other tools. The calibration data sets include: Zhengyou Zhang [12], Heikkila [13], Bakstein and Halir [14]. The extraction of the intrinsic and extrinsic parameters is an offline process and is explained in Sections 2.3.3.1 and 2.3.3.2.

2.3.3.1 Intrinsic Parameters

In order for the Camera Calibration Toolbox to extract the intrinsic parameters of the camera system, a set of calibration images are required. The computer vision community generally uses a standard checkerboard pattern for the calibration images. An example of the checkerboard pattern used is shown in Figure 2.5. Generally, a set of 10-15 calibration images are required to extract accurate values.

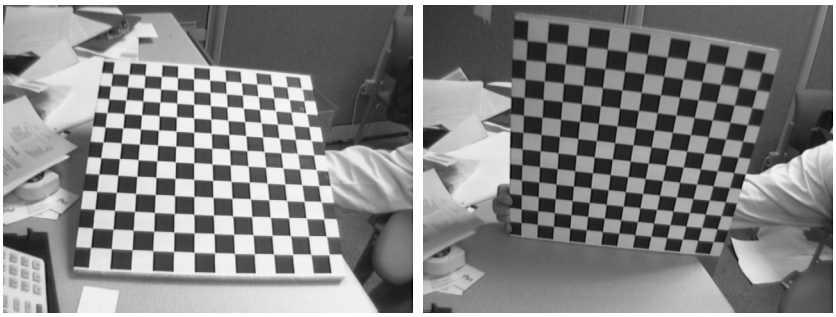


Figure 2.5: Camera calibration checkerboard images

The process in Matlab begins by running the Matlab script *calib_gui.m*. This opens up a window which allows selecting the mode of operation. The two modes consist of Standard and Memory efficient. The Memory efficient mode is useful on older computers where CPU

resources are often low. Here, the calibration images are loaded one by one, rather than loading the full set. Regardless of which mode is chosen, a new window will appear with all the calibration options. The different options for calibration include Load, Save, Read images, Extract grid corners, Calibration, and Comp. Extrinsic, among others as seen in Figure 2.6.

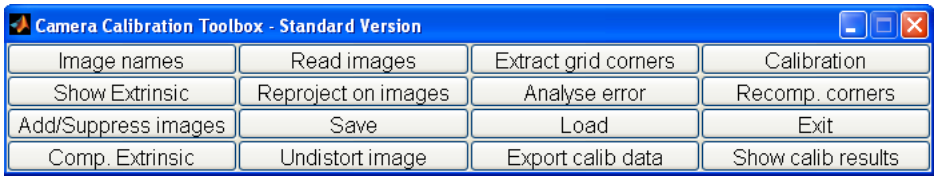


Figure 2.6: Camera calibration user interface

Loading the calibration images for processing purposes is achieved by selecting the *Read images* option. Once the set of images are known to the toolbox they can be used for determining the coefficients needed for solving the equations presented in Section 2.3.2. Solving these equations is simplified by using the *Extract grid corners* option. This allows the user to manually select the four outer corners of the checkerboard, enter the number of rectangles in the x and y directions, followed by the height and width of each rectangle in millimetres. An illustration of this is shown in Figure A.2 (see Appendix A), where O, represents the origin and {X, Y}, represent the directions of the rectangle. With this information provided, a Matlab routine automatically estimates and shows the locations of corners on the checkerboard as seen in Figure A.3 (see Appendix A), where O, represents the origin and {dX, dY}, represent the size of the sides of each rectangle. Some images may be effected by lens distortion produced by the camera, there is an option for entering a distortion coefficient to improve the corner extraction accuracy. These steps must be repeated for each image in the set.

The completion of the corner extraction procedure allows us to begin the main calibration step, this is started by selecting the *Calibration* option from the GUI. Calibration is done in two steps, these steps include initialization and optimization. The initialization

step computes a sub-optimal linear closed-form solution (as described in Section 2.3.2) for the calibration parameters. This is followed by an optimization step which minimizes the total reprojection error over all calibration parameters by iterative gradient descent with an explicit (closed-form) computation of the Jacobian matrix [11].

2.3.3.2 Extrinsic Parameters

In order for the Camera Calibration Toolbox to extract the extrinsic parameters of the camera system, one new calibration image that was not used in the main calibration procedure (of Section 2.3.3.1) is needed to provide a world coordinate system origin.

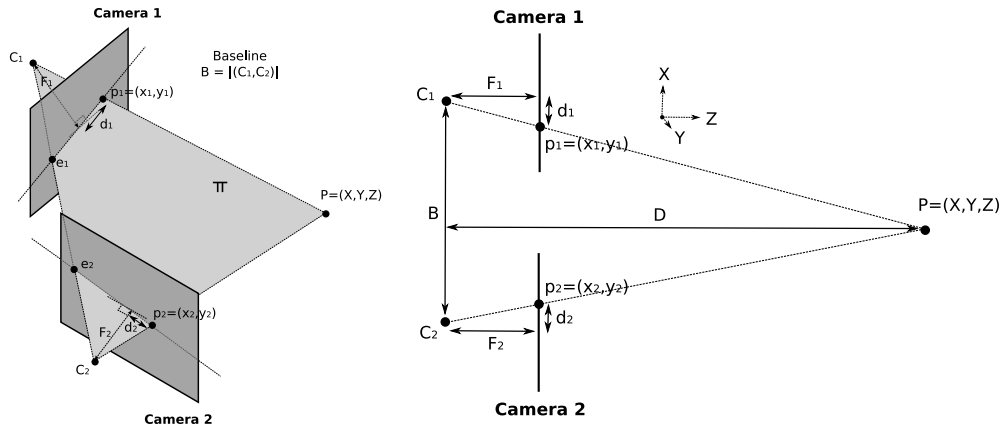
The process for extracting these values is quite similar to the main calibration procedure. To begin this procedure, the *Comp. Extrinsic* option must be chosen from the GUI. Then, the four extreme corners are selected by the user on the calibration image. This is then followed by entering the number of rectangles in the x and y directions. The height and width of each rectangle are also entered for an accurate estimate of the corners in the image. The toolbox then returns the computed values for the rotation and translation. In order to further utilize this information along with the intrinsic parameters, they must be saved by selecting the *Save* option found on the GUI. The file will be named $< Calib\_Results.mat >$.

2.3.4 The Stereo Camera Model: Two-view Geometry

A two-camera view of the scene presented can reduce the depth estimation to a problem of triangulation. Triangulation relies on the fact that most points in each of the two images can be identified as representing the same 3-D position. Figure 2.7a presents a binocular stereo vision camera model. Each camera is represented by the pinhole camera model. The points e_1 and e_2 are the epipoles of the camera model and are given by the intersection of the baseline with the image planes of each camera. The baseline, B, is the distance of separation between the principle points of the two camera (*i.e.*, $B = |c_1 - c_2|$) . A scene point, $P = (X, Y, Z)^T$, is projected onto each camera at points $p_1 = (x_1, y_1)^T$ and $p_2 = (x_2, y_2)^T$ such that they lie

on epipolar lines defined by (e_1, p_1) and (e_2, p_2), respectively. The plane represented by these two lines is known as the epipolar plane, π. The epipolar plane contains the scene point and the corresponding centres of the two cameras. Given a point in one image, the epipolar line in the other image contains the matching (corresponding) point.

Being able to match point correspondences can allow us to determine the depth, Z, of a point in the scene, given two different views. The problem of stereo correspondence can be reduced to a one-dimensional (1-D) search once the epipolar lines are known. If we know the principle points of the two cameras (c_1 and c_2) and the orientation of the image planes (p_1 and p_2), then knowing the location of the image of a scene point in one image completely specifies the entire epipolar line (along with the matching scene point) in the other image. This can be easily seen from Figure 2.7.



(a) Generalized Stereo Setup (b) Rectified Stereo Setup

Figure 2.7: A binocular (two-view) stereo camera model

2.3.5 Stereo Calibration

A special case of the Camera Calibration Toolbox for Matlab, discussed in Section 2.3.3, can be used to perform stereo calibration. Two-camera stereo calibration requires obtaining the left and right camera parameters. Once the camera parameters are obtained, the saved files should be re-named to *Calib_ Results_* ***left****.mat* and *Calib_ Results_* ***right****.mat* for each

associated view. Following this step, the stereo calibration process begins by running the Matlab script file *stereo_gui.m*. The Stereo Calibration user interface is shown in Figure 2.8. Performing stereo calibration is straight forward. Once the camera parameters are loaded, an

Figure 2.8: Stereo calibration user interface

initial calibration is executed which gives a rough estimation of the stereo rig. A better, more accurate estimation of the stereo rig can be found by selecting the *Run stereo calibration* option on the GUI. This performs a global stereo optimization procedure which recomputes the intrinsic and extrinsic parameters for the cameras. The stereo calibration results can be saved by choosing the *Save* option. The file will be named $< Calib\_Results\_stereo.mat >$.

2.4 Stereo Vision System

Having knowledge of the camera models, this section looks further into using computer vision approaches to interpreting the captured information. Section 2.4.1 introduces stereo image rectification, which is used to place points and features of two different images of the same scene on corresponding scanlines. This is done in an attempt to extract depth information from the scene in a less complex manner. Section 2.4.2 introduces stereo correspondence and a dynamic programming solution which allows machines to perform stereopsis much like we do. A brief literature review is found in Section 2.6, which discusses previous hardware implementations of image rectification, stereo correspondence and stereo vision systems.

2.4.1 Stereo Image Rectification

In stereo camera geometry, matching points are constrained to lie on epipolar lines in both the left and right image. The main goal of stereo image rectification is to transform the stereo images into a camera geometry where the optic axes are parallel, so that the images of corresponding 3-D scene points will lie on the same scanline in both images, as illustrated in Figure 2.7b. Misalignments in the assembly of the stereo camera image sensors may cause slight rotations and translations between the image pairs. This may result in features that appear in both images being placed on different scanlines in the two images. The importance of performing stereo image rectification is to help reduce the complexity of stereo correspondence problem. Typically, images that are un-rectified require searching along a non-horizontal epipolar line, which is complex. The complexity arises as different epipolar lines would be needed for every point and due to image resampling, among other things. However, after performing stereo image rectification, the problem of matching corresponding points is reduced to a 1-D search.

2.4.1.1 Algorithm

Assuming that the stereo camera has been calibrated and the stereo parameters are known, two homography matrices can be defined in order to transform the original images. The transformation rotates the (virtual) image planes about their optical centres and re-projects the scene on to them. This makes the epipolar lines of the image planes collinear and horizontal. Equivalently, this means that the epipoles (e_1 and e_2) are at infinity. The following steps from [7] describe how the rotation matrix, R_{rect}, can be computed assuming we know the relative translation (baseline), $\overrightarrow{T} = c_1c_2$, and rotation, R, of the right camera with respect to the left camera.

1. Choose: $\overrightarrow{e_1} = \dfrac{\overrightarrow{T}}{\left\|\overrightarrow{T}\right\|}$ to make the epipole of the left camera perpendicular to the optic axis

2. Choose: $\overrightarrow{e_2} = \overrightarrow{e_1} \times \hat{z} = \frac{[-T_y, T_x, 0]^T}{\sqrt{T_x^2 + T_y^2}}$ to make $\overrightarrow{e_2}$ perpendicular to both $\overrightarrow{e_1}$ and the optic axis ($\hat{z}$)

3. Cross-product: $\overrightarrow{e_3} = \overrightarrow{e_1} \times \overrightarrow{e_2}$

4. Create the rotation matrix: $R_{rect} = \begin{bmatrix} \overrightarrow{e_1^T} \\ \overrightarrow{e_2^T} \\ \overrightarrow{e_3^T} \end{bmatrix}$

It should be noted that $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$ are not the epipoles. The images are resampled via a homography, H, which is also known as a projective collineation. A homography is a 3×3 non-singular matrix, and can be used to transform an image I to create an image I' ($H : I \rightarrow I'$). Remapping the points in the left image is achieved by setting the left homography $H_L = R_{rect}$. In a similar manner, the points in the right image are remapped by setting the right homography $H_R = R_{rect}R^T$. Each image location, $\overrightarrow{x_L}$ and $\overrightarrow{x_R}$, in each of the left and right image is mapped to $\overrightarrow{x_L}' = H_L\overrightarrow{x_L}$ and $\overrightarrow{x_R}' = H_R\overrightarrow{x_R}$. A problem here exists since $\overrightarrow{x_L}'$ and $\overrightarrow{x_R}'$ may not be integral pixel locations. Thus we perform an inverse map for each (integral) image location $\overrightarrow{x_L}'$ and $\overrightarrow{x_R}'$, we compute $\overrightarrow{x_L} = H_L^{-1}\overrightarrow{x_L}'$ and $\overrightarrow{x_R} = H_R^{-1}\overrightarrow{x_R}'$. The image data must be "normalized" if the assumption of $x_o = y_o = 0$ is untrue. This requires $H_L^{-1} = \overline{K}_L H_L^{-1} \overline{K}_L^{-1}$ and $H_R^{-1} = \overline{K}_R H_R^{-1} \overline{K}_R^{-1}$. The intrinsic parameter of each view is represented by matrix $\overline{K}$ (see Section 2.3.1). It will typically be the case that $\overrightarrow{x_L}$ or $\overrightarrow{x_R}$ will not represent an integral image location. We can however, use the exact value of $\overrightarrow{x_L}$ or $\overrightarrow{x_R}$ to interpolate the intensity values of neighbouring pixel locations. This is achieved using various interpolation methods; we chose bilinear interpolation as it provides a simple yet accurate estimate of pixel intensities.

2.4.2 The Depth Estimation Problem: Stereo Correspondence

In Figure 2.7b, the values of d_1 and d_2 allow the computation of the disparity, $d = d_1 - d_2$. The disparity of corresponding pixels is the displacement of the images of a scene point with

respect to each other, measured relative to the principal points. The process of rectification allows this to be possible because the principal points of the two images are aligned to the same image coordinates. The depth, D, of a 3-D world point that corresponds to the matched feature pair can be computed using Equation 2.10, where $f = f_1 = f_2$ (assumption of rectification) represents the focal length of the two cameras, while d, B and α represent the disparity, baseline length and scale factor ($pixel/mm$), respectively. Often, the scale factor is set to 1 (*i.e.*, $\alpha = 1$) if the focal length is measured in pixels. It can be noticed that the depth is inversely proportional to disparity, up to some scale factor.

$$D = \frac{f\alpha B}{d} \tag{2.10}$$

A technique used for identifying corresponding points in each of the two images from a stereo camera system is known as *stereo matching*. As mentioned earlier, having knowledge of point correspondences provides a means to perform triangulation. Methods used for identifying matches among pairs of points across images include similarity metrics, cost functions or even likelihood functions, to name a few. The computer vision community has studied the stereo correspondence problem extensively and many algorithms exist for determining matching points. Such algorithms are grouped into two classes: sparse and dense. Sparse methods use feature-based matching techniques, such as corners or scale-invariant feature transform (SIFT), to identify matching points in each view of the stereo image. The disparity estimates for the matching points are used to generate sparse disparity maps. The dense method uses matching techniques which are pixel-based rather than feature-based, so the disparity values are estimated for all pixels. Matching based on a correlation window computes a correlation score between windows centred at a particular pixel (feature) in each of the two images of the stereo pair. For a given pixel (feature) in one image, the window slides over nearby coordinates in the other image to produce other correlation scores. The best scoring pixel (feature) pairs are deemed as being matched and a corresponding disparity estimate is computed from the relative shift of the windows. Figure 2.9 shows

a flow chart of sparse and dense disparity estimation methods. Equations 2.11, 2.12 and 2.13, define three commonly used cost functions that can be used for locally matching pixels or features. The Sum of Squared Difference (SSD) and Sum of Absolute Difference (SAD) functions are shown in Equations 2.11 and 2.12, where w_1 and w_2 are the windows containing pixels (features) p_1 and p_2 of camera 1 and 2, respectively. The Normalized Cross-Correlation (NCC) based function is shown in Equation 2.13. This function takes into account image statistics within a window which include the mean (μ_{w_1}, μ_{w_2}) and standard deviation (σ_{w_1}, σ_{w_2}).

$$C_{SSD} = \sum_{\forall p_{1,2} \epsilon w_{1,2}} \left[I\left(p_1\right) - I\left(p_2\right) \right]^2 \tag{2.11}$$

$$C_{SAD} = \sum_{\forall p_{1,2} \epsilon w_{1,2}} \left| I\left(p_1\right) - I\left(p_2\right) \right| \tag{2.12}$$

$$C_{NCC} = \sum_{\forall p_{1,2} \epsilon w_{1,2}} \frac{\left[I\left(p_1\right) - \mu_{w_1} \right] \left[I\left(p_2\right) - \mu_{w_2} \right]}{\sigma_{w_1} \sigma_{w_2}} \tag{2.13}$$

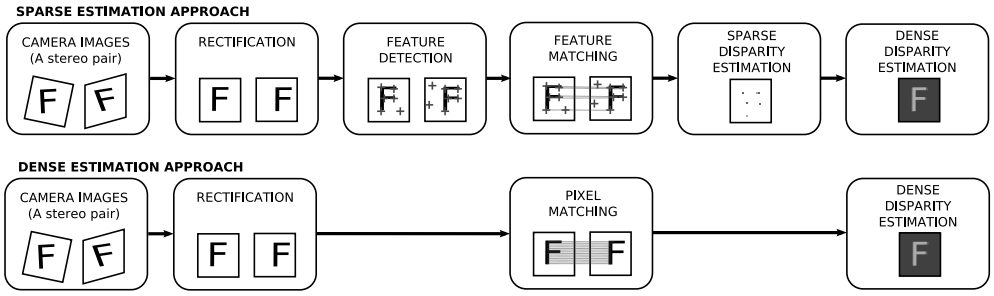


Figure 2.9: Flow chart of sparse and dense disparity estimations

2.4.2.1 Stereo Issues

A number of issues arise when performing stereo matching. The first issue deals with occlusions. Occluded pixels (features) are visible in the view of one camera, but not in the

other. During local stereo matching occlusions can result in an attempt to assign a depth estimate to a pixel (feature) that has no corresponding point for triangulation. The stereo correspondence algorithm used in this thesis was designed to take occlusions into account by including a penalty term in the cost function. Not all algorithms do this.

The second issue occurs as a result of thin objects. Typically the relative ordering of neighbouring scene points is assumed to be preserved in the two images. This is an assumption which is often made for stereo matching. However, an exception can be made for objects known as *thin occluders*. Thin occluders violate the ordering assumption mentioned previously. A good example to verify this is to consider a particular scene point which generates corresponding images points p_1 and p_2, in cameras 1 and 2, as a result of perspective projection. Suppose that a narrow object, such as a pencil, is positioned near the stereo camera at a central location. The image of the pencil in camera 1 might generate an image point, $\acute{p}_1$, to the left of p_1 while generating an image point, $\acute{p}_2$, to the right of p_2 in camera 2. It can be noticed here that the ordering assumption fails, as $\acute{p}_2$ does not appear before p_2 in camera 2.

Aside from the issues concerning occlusions and thin objects, some other issues effect the detection of pixels (features). These issues include differences in magnification between the two images, regions of low texture and changes in illumination across the cameras. It should be mentioned that magnification changes violate uniqueness.

2.4.2.2 Dynamic Programming Maximum Likelihood Stereo Algorithm

The stereo matching process imposes the epipolar line constraint as discussed earlier. Another constraint made for the search process is to assume that all objects in the scene are thick, allowing us to assume the ordering constraint. An additional set of constraints include the uniqueness constraint and disparity range constraint. The ordering constraint says that the order of neighbouring correspondences on a particular epipolar line is always preserved. The uniqueness constraint requires that each pixel in one image only maps to at most one

pixel in the other image. Lack of a match can be used to determine occlusions. Finally, the disparity range constraint indicates that corresponding pixels in one image can be found within some limited distance in the other image. This range limit can be predicted from camera and scene geometry.

An illustration of the ordering and uniqueness constraints is shown in Figure 2.10. This graphical description demonstrates that these constraints hold true given opacity and thick scene objects. It can be seen that points on the object map appear on the left and right cameras in a unique order. The point map obtained in the view of Camera 1 shows that point 5 of the object is occluded. Similarly, for the point map of Camera 2, point 2 is found to be occluded.

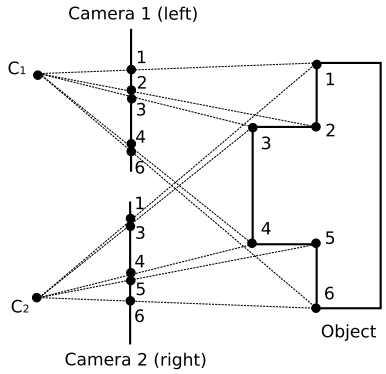


Figure 2.10: Ordering and uniqueness constraints

Given that pre-rectified images are used for the extraction of depth information, a Dynamic Programming Maximum Likelihood (DPML) optimization approach can be used to estimate disparities. We use an algorithm originally developed by Cox *et al.* in [1]. For any optimization problem to be solvable with dynamic programming, it must include optimal substructure and overlapping subproblems. Dynamic programming solves problems by combining the solutions to subproblems. If the sub-problem solutions can be reused several times, it is said to have overlapping substructure. The optimization problem is said to have optimal substructure if each subproblem can be shown as the optimal solution within the scope of its inputs. Memoization is a technique of storing values of a function, rather

than recomputing them each time the function is called. This technique is used as a variant method for taking advantage of the overlapping subproblem property [15].

The stereo correspondence algorithm developed in [1] is for cases of dense disparity estimation. The DPML optimization approach is used to determine the Longest Common Subsequence (LCS), where the sequence is defined by the pixel intensity values. Each input scanline from either view (left or right) can be viewed as two distinct subsequences. The optimal substructure of the DMPL solution means that the disparity results generated from input scanlines are globally optimal [16]. The DPML algorithm consists of two phases which execute in sequence: a forward-pass and a backward-pass. A Matlab-like outline of the forward-pass is shown in Algorithm 1 (see Appendix B). The forward-pass begins by initializing the match matrix, MM and cost matrix, CM. Here, N represents the number of pixels per scanline. The cost matrix is initialized with constant occlusion costs, OC, given by Equation 2.14, where P_d is the probability of each camera imaging a point in the scene, ϕ is the associated field of view and σ^2 represents the variance associated with camera sensor noise. Following this, left and right image pixel streams from corresponding scanlines are compared to each other sequentially up to some maximum disparity range, D_{max}. The pixel comparisons are used to produce a cost estimate given by Equation 2.15, where $I_L(x)$ and $I_R(x)$ are pixel values at position x in corresponding scanlines. Cost values are generated by comparing every pixel in the left stream to every pixel in the right stream. For each cost matrix entry, an associated entry is also made to the match matrix which stores values indicating the presence of occlusions occurring between the two subsequences, or non-occlusions when a match is deemed to exist.

$$OC(P_d, \sigma^2, \phi) = \log \frac{P_d \phi}{(1 - P_d)} \sqrt{\frac{2\pi}{\sigma^2}} \tag{2.14}$$

$$NOC(I_L(x), I_R(x+d), \sigma) = \frac{\left[I_L(x) - I_R(x+d)\right]^2 \sigma^2}{4} \tag{2.15}$$

The backward-pass initiates following the storage of all cost values to the $N \times N$ cost matrix by the forward-pass, as shown in Algorithm 2 (see Appendix B). Minimizing the cost value is accomplished by backtracking through the match matrix to reveal the lowest-cost path. The left and right pixel locations corresponding to points along the shortest path are used to compute the disparity or to indicate an occlusion [16]. It should be mentioned that dynamic programming is particularly suited to hardware implementations. For more details on this algorithm please refer to [1, 17, 16].

2.5 FPGAs

The term FPGA is an acronym for Field Programmable Gate Array. Essentially, FPGAs are digital integrated circuits (ICs) that contain configurable (programmable) blocks of logic along with configurable interconnects between them. Design engineers can configure (*i.e.*, rewire on the fly) such devices to perform a tremendous variety of tasks. All FPGAs include three major components: I/O blocks, logic blocks and programmable routing. These components can be configured to implement basic combinational logic (*e.g.*, AND, OR, NOT, NAND, NOR functions) or more complex synchronous logic (*e.g.*, a microprocessor). Xilinx FPGAs contain a large quantity of sub-circuits known as Configurable Logic Blocks (CLBs), which are made up of modules known as *slices*. The interconnection between CLBs use long wires to transfer information. Slices consist of two logic cells and are interconnected in a more localized manner using shorter wire segments. A logic cell comprises a 4-input Look-up Table (LUT), which can also act as a 16×1 RAM or a 16-bit shift register, a 2-to-1 multiplexer and a register. A 4-input LUT can synthesize any Boolean-valued function with (up to) four Boolean inputs. Figure A.4 (see Appendix A) shows an illustration of a logic cell.

FPGAs are powerful tools for realizing custom hardware solutions. Other solutions can utilize Application Specific Integrated Circuits (ASICs), Digital Signal Processors (DSPs)

or even microprocessors. General purpose processors are often too slow due to their sequential nature of processing, while DSPs are often best-suited to applications that utilize MAC (multiply-accumulate) operations. On the other hand, ASICs often provide the best performance and power consumption. However, similar to the DSPs and microprocessors, ASICs have some drawbacks. The main problem with ASICs is the long development cycle required to get from the initial design to the final product. Another problem is in the cost of developing an ASIC: often ASIC designers spend millions of dollars for specialized equipment, which requires a high degree of expertise. Furthermore, any changes in the design can be costly and may result in time-to-market delays. The typical cost of producing a single 90 nm ASIC design ranges between $20-30 million according to [18]. FPGAs are commonly used for rapid prototyping, as they have the advantage of quickly reprogramming the device as many times as required. This feature is useful when design flaws are found, as fixes can be made and verified quickly. Another advantage of the FPGA is that it is cost-effective when used for products of a limited quantity. When production is in large quantities, ASICs are often more cost-effective [19]. FPGAs also provide flexibility to the user, as the large array of configurable logic allows the implementation to be accelerated by duplicating (or parallelizing) functional units. Furthermore, some FPGAs include dynamic reconfiguration (or partial reconfiguration) capabilities [20]. This capability allows a portion of the configurable logic on the device to be loaded with a new design while the rest of the custom logic operates normally. A good reference for learning more on FPGAs is [20].

2.5.1 VHDL

The design of digital logic circuits on FPGAs is achieved by coding in a *hardware-description language* such as VHDL. This name is an acronym for VHSIC Hardware Description Language, where VHSIC stands for Very High Speed Integrated Circuit. This language is used to describe the behavior and structure of digital hardware designs. Another commonly used coding language for the design of digital circuits is Verilog.

VHDL provides support for describing concurrent events, which takes advantage of the FPGA's ability to perform multiple operations concurrently. This concurrency differentiates VHDL from other high-level languages (*e.g.*, Pascal, C and C++) which are primarily used for software design. The coding of the design allows using parameters, known as parametrization, which allows application and design specific changes to be made without having to re-code the design. Without parameters, large designs would require a significant amount of time to incorporate simple changes. Coding in VHDL also allows the behavior of complex circuits to be captured into a design system for automated synthesis or for functional simulation.

As an example, the VHDL code shown in Figure 2.11 describes a 1-bit full-adder. The design has three inputs and two outputs. The inputs consists of two addends (x and y) and a carry-in (Cin). The two outputs generated are a sum (s) and a carry-out ($Cout$).The code in Figure 2.11 highlights VHDL keywords in bold face. The ENTITY section of the code is essentially a black-box description of the design, as it contains and describes the inputs and

```
-------------------------------------
-- 1-bit full adder designed in VHDL
-------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fulladd1 IS
    PORT(Cin, x, y : IN STD_LOGIC;
         s,Cout    : OUT STD_LOGIC);
END fulladd1;

ARCHITECTURE description OF fulladd1 IS
BEGIN
    s <= x XOR y XOR Cin;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y);
END description;
```

Figure 2.11: VHDL code for a 1-bit full-adder

outputs. The ARCHITECTURE section of the code describes the internal functionality of the design, it can be seen that the outputs are assigned by a logical expression of the inputs. The code shown in Figure 2.12 describes an 8-bit full-adder. This example makes use of the

′+′ operator, which exploits the high-level nature of the language. A complete reference for VHDL and a guide for synthesis can be found in [21] and [22], respectively.

```
-----------------------------------------
-- 8-bit full adder designed in VHDL
-----------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY fulladd8 IS
    PORT(x, y : IN STD_LOGIC_VECTOR(7 downto 0);
         Cin  : IN STD_LOGIC;
         s    : OUT STD_LOGIC_VECTOR(7 downto 0);
         Cout : OUT STD_LOGIC);
END fulladd8;

ARCHITECTURE description OF fulladd8 IS
    signal tmp: std_logic_vector(8 downto 0);
BEGIN
    tmp <= x + y + Cin;
    SUM <= tmp(7 downto 0);
    Cout <= tmp(8);
END description;
```

Figure 2.12: VHDL code for a 8-bit full-adder

2.6 Literature Review

A brief overview on different hardware implementations of image rectification is given in Section 2.6.1. Similarly, previous hardware implementations of stereo correspondence algorithms are discussed in Section 2.6.2. Finally, an overview of fully integrated stereo vision systems are presented and compared in Section 2.6.3.

2.6.1 Image Rectification

A number of different hardware based image rectification modules were developed and reported in [23, 24, 25, 26, 27]. Most of these image rectification modules were developed on reconfigurable devices with the exception of the work produced by Courtney *et al.* [23], which

was developed using memory and VLSI components. The general design of the implementations share a common structure, as they all include blocks to hold the input image, compute the transformation and perform pixel intensity interpolation. The work by Jörg *et al.* [24] and Masrani *et al.* [25] use a bicubic polynomial to perform the warping transformation, where the coefficients are computed offline by camera calibration. This type of operation provides a more accurate estimation but requires considerably more hardware resources in terms of adders and multipliers when compared to our implementation which uses a 1st-order Taylor Series Polynomial model. The implementation with a 1st-order Taylor Series Polynomial works well for cameras that are fairly aligned (see Chapter 3). The work developed by Serguienko [26] and Vancea *et al.* [27] utilize look-up tables (LUTs) rather than polynomials, for obtaining the warping transformation. The LUTs are computed offline by calibration and stored in dedicated memory locations. Vancea *et al.* [27] use multiple 64 MB Synchronous Dynamic RAMs to store the LUT and the input images, while Serguienko [26] uses on-board memory to store its LUT. The image rectification module developed by Courtney *et al.* [23] used VLSI and memory components, rather than reconfigurable devices as mentioned previously. The performance achieved by the work of Vancea *et al.* [27] is 85 frames per second (fps) on 640×512 resolution images, [23, 24, 25] perform image rectification at 30 fps on 384×288 and 640×480 resolution images. [23] failed to mention the resolution of the image, while [26] failed to mention anything about its associated performance.

To date there have not been any hardware implementations of high-speed (*i.e.*, 200 fps) stereo image rectification circuits. This thesis explores an image rectification solution that achieves high frame-rates on full resolution images (*i.e.*, 640×480), while utilizing minimal resources on an FPGA device that will be used to develop a larger system.

2.6.2 Stereo Correspondence

Hardware based stereo correspondence provides for the computation of disparity maps and estimation of depth information at higher frame-rates than associated software implemen-

tations. Common approaches for the development of stereo correspondence in hardware generally use correlation and other area-based methods. The most common methods make use of SAD aggregated cost functions to compare pixel intensities, since they are easy to implement and cheap in terms of resources. However, they do not provide very accurate disparity estimates as mentioned by Scharstein *et al.* in [2]. These types of implementations utilize line buffering to align pixels in the left and right images of a stereo pair for parallel windowing computations. The stereo correspondence implementations described in Han *et al.* [28], Hariyama *et al.* [29], Mitéran *et al.* [30], Miyajima *et al.* [31] and Perri *et al.* [32] all utilize this method for parallelizing computations. Most of these implementations produce results at approximately 30 fps. Hariyama *et al.* [29] operates on image resolutions of 64×64 pixels and its design is large as it requires using multiple smaller FPGA devices to realize the implementation. Furthermore, it fails to mention the disparity range. The performance reported in the work by Han *et al.* [28] achieves approximately 60 fps on image resolutions of 640×480 with a disparity range of 128 pixels. Jia *et al.* [33] and van der Horst *et al.* [34] propose methods in their work, however they fail to make the details of their designs clear. Table 2.1 lists a summary of the SAD-based stereo implementations.

Another approach for the development of stereo correspondence in hardware follows phase based correlation methods described in [35, 36, 25]. The phase based correlation methods of [35, 25] use Freeman filters, while [36] uses Gabor filters to model the retinal cell responses described in Section 2.2. Phase based correlation has the advantage of producing significantly better results than SAD methods and compete well with DPML methods. Similar to other methods, phase based correlation uses line buffers to facilitate parallel computations. The difficulty with this method occurs in performing square root computations, as these are difficult to implement in hardware and require a large amount of resources when parallelized. Similar to the SAD based implementations, the phase based correlation implementations produce results of approximately 30 fps. Frame-rates of over 200 fps are achieved by the work by Díaz *et al.* [36], however the disparity range is compromised to only 4 pixels at image

resolutions of 640×480. Another implementation by Woodfill *et al.* [37] uses a consensus algorithm which is very simple and leads to poor accuracy. The implementation produces results of 200 fps on image resolutions of 512×480 pixels, with a maximum disparity of 52 pixels.

Table 2.1: Summary of SAD-based stereo implementations

	$N \times M$	D_{max}	FPS
Hariyama *et al.* [29]	64×64	N/A	30
Han *et al.* [28]	640×480	128	60
Mitéran *et al.* [30]	256×256	20	10
Miyajima *et al.* [31]	640×480	80	20
Perri *et al.* [32]	512×512	256	25
Jia *et al.* [33]	640×480	64	30
van der Horst *et al.* [34]	256×256	18	20

There have not been any hardware implementations of the DPML solution to date. Existing solutions for stereo correspondence produce good results at low frame-rates on images which have resolutions of 640×480 or less. This thesis looks at the modification and integration of a high frame-rate DPML solution developed by Sabihuddin *et al.* [3] which does not compromise accuracy and has little compromise on the disparity search range and resolution. It should be mentioned that correlation based approaches use SSD cost functions with the constraints mentioned to obtain a globally optimal solution.

2.6.3 Stereo Vision Systems

This section looks at a number of different stereo vision systems that have been previously developed. In general, these types of systems include a video-acquisition system, an optional video pre-processor and a stereo correspondence circuit. Reconfigurable systems have been integral for the development of these types of systems as FPGAs have had increases in capacity and speed over the last few years. The parallelization capabilities of the FPGA allows computationally intensive computer vision applications to be accelerated. The stereo vision systems reported in [35, 38, 25, 39, 40] all use FPGAs for some purpose in their

associated systems. The stereo vision system developed by Kanade *et al.* [41] is the only system reviewed that is solely composed of discrete components: a C40 DSP-array board and a real-time operating system board. The stereo vision system known as INRIA [38], uses 23 Xilinx XC3090 FPGAs and operates on 256×256 resolution images, with a maximum disparity of 32 pixels. This system, like many others, includes a frame grabber to capture the images, an image rectification circuit and also a stereo extraction unit. The performance of the system, however, is quite poor as it produces a frame rate of 3.6 fps. The PARTS [39] reconfigurable vision system consists of a 4×4 array of mesh-connected Xilinx 4025 FPGAs, which operates on 320×240 resolution images, with a maximum disparity of 24 pixels. This system does not include an image rectification circuit, but it does provide a better performance than [38] at a frame rate of 42 fps.

The next three stereo vision systems all perform at frame rate (*i.e.*, 30 fps). The CMU vision system developed by Kanade *et al.* [41], as already mentioned, does not use reconfigurable logic to implement the system. The system operates on 200×200 resolution images and includes an image rectification circuit. The maximum disparity of the system is 30 pixels. The TM-3A system developed by Darabiha *et al.* [35] uses 4 Xilinx Virtex 2000E FPGAs and does not include an image rectification circuit. It produces disparity results on 256×360 resolution images and has a maximum disparity of 20 pixels. An improvement to the system by Darabiha *et al.* [35] was accomplished by Masrani *et al.* [25]. This system produces disparity results on 640×480 resolution images and has a maximum disparity of 128 pixels. Furthermore, it includes an image rectification circuit. The system was developed using the TM-4 board which contains 4 Altera Stratix S80 FPGA devices.

Finally, a more recent stereo vision system has been developed by Woodfill *et al.* [40] that provides a much higher performance than the other described previously. The Tyzx DeepSea G2 vision system produces disparity results at 200 fps on 512×480 resolution images and has a maximum disparity of 52 pixels. It should be noted that the system includes an image rectification circuit. The system was designed on a platform that consists of a PowerPC chip

(666 MHz) running Linux, DeepSea II stereo ASIC, an FPGA and a DSP/Co-processor. The main purpose of the FPGA is to provide as a communication interface and for storage. Table 2.2 compares the characteristics of each stereo vision system discussed. From this table, $N \times M$ represents the image resolution, D_{max} is the maximum disparity, FPS represents the performance and IR represents the presence of an image rectification circuit.

Table 2.2: Summary of the reported stereo vision systems

	$N \times M$	D_{max}	FPS	IR	$Platform$
INRIA [38]	256×256	32	3.6	$\surd$	23 Xilinx XC3090 FPGAs
PARTS [39]	320×240	24	42	χ	16 Xilinx 4025 FPGAs
CMU [41]	200×200	30	30	$\surd$	C40 DSP-array board+RTOS
TM-3A [35]	256×360	20	30	χ	4 Xilinx Virtex 2000E FPGAs
TM-4 [25]	640×480	128	30	$\surd$	4 Altera Stratix S80 FPGAs
TYZX [40]	512×480	52	200	$\surd$	PowerPC+ASIC+FPGA+DSP

2.7 Summary

This chapter introduces many of the concepts that are essential to develop a stereo vision system in hardware. We began by looking at the human vision system, as a model for developing systems capable of replicating the biological function of stereopsis. Following this, we discussed imaging models such as the pinhole camera model, camera calibration, the stereo camera model and perspective projection to lead us into our description of stereo image rectification and stereo correspondence. The pinhole camera model introduces some basic principles about imaging; this model was extended to the stereo (two-view) camera model. Stereo image rectification is introduced as a method for placing corresponding points (features) on the same scanline in both images by aligning the epipolar lines. The epipolar constraint allows performing stereo matching at a reduced complexity, as corresponding points (features) lie on the epipolar lines, reducing the disparity estimation to a 1-D search problem. The stereo correspondence problem can be solved using a dynamic-programming-maximum-likelihood solution, which is based on ordering, uniqueness and disparity range

constraints. The solution uses SSD cost functions as the similarity metrics for comparing pixels. This solution is well suited for hardware implementations on FPGAs. Field Programmable Gate Arrays have been introduced as tools for rapidly prototyping digital circuit designs. They provide the benefits of low production cost, high flexibility, and excellent performance versus other digital design solutions. Finally, we presented details of existing hardware implementations of image rectification, stereo correspondence and stereo vision systems, most of which achieve frame-rates in the range of 30 fps. While there are systems that achieve frame-rates above 30 fps, they have a relatively poor accuracy or reduced image resolution. In the next chapter, the architecture and implementation of a high frame-rate stereo vision system will be the focus.

Chapter 3

Architecture and Hardware Implementation

3.1 Introduction

In this chapter we focus on the architecture and hardware implementation of an FPGA-based stereo vision system. In Section 3.2, we begin with an overview of the system architecture. Following this, we describe the on-chip design of the video pre-processor and custom-built video-acquisition system in Sections 3.3 and 3.4, respectively. These sections include two generations of the vision system, one which operates at 30 frames per second (fps) and the other at 200 fps. In Section 3.5, we explore the features of our processing platform, the Amirix AP1100. In Section 3.6 we describe the integration, modification and verification of the sub-systems which operate with the 200 fps video-acquisition system. Finally, Section 3.7 summarizes the chapter's contents.

3.2 Overview of the System Architecture

Figure 3.1 shows the general overview for our system's architecture. The core modules in the system include: the video-acquisition system, the video pre-processor, the stereo extraction

module and the object tracking module, while the stereo controller serves as an interfacing component. The purpose of the video-acquisition system is to provide a real-time binocular view of the scene for processing in 3-D. The video pre-processor is needed to perform stereo image rectification. The stereo extraction module uses the information from the video pre-processor to match corresponding pixels in the two views. As a result of the pixel matching, disparity information (*i.e.*, inverse of depth) can be obtained. Finally, the object tracking module uses the disparity information and a model of an object (*i.e.*, a satellite) to recover the pose and also to determine the location of the object with respect to the world coordinate system.

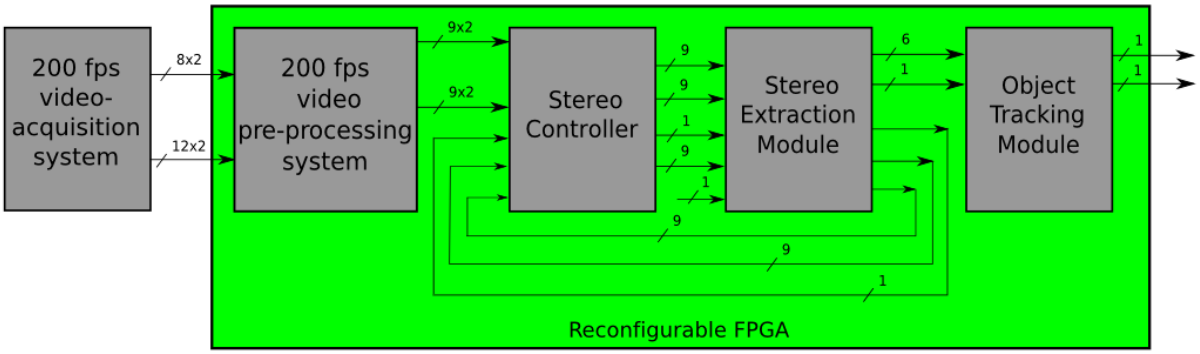


Figure 3.1: Overview of the system architecture

The implementation of the video pre-processor, the stereo extraction module and the object tracking module exist on the same reconfigurable architecture, known as an FPGA. It should be noted that the object tracking module is not yet integrated to the system, but will be included in future work.

3.3 Architecture of Video Pre-processor

The operation of the video pre-processor (VPP) follows the theory presented in Chapter 2 regarding stereo image rectification. This section describes the architecture of the video pre-processor. The design of the video pre-processor is implemented on the Xilinx Virtex-II Pro XC2VP100 FPGA found on the Amirix AP1100 Platform (see Section 3.5). The

development of the VPP is different for each of the two different camera systems, the 30 and 200 fps video-acquisition systems (see Section 3.4). The following sections introduce the major components used for the VPP and the architecture of the final product.

3.3.1 Architecture-to-task vs. Task-to-architecture Designs

Designers can be placed in different design situations. One situation is where they plan a design from the initial stages. Here the designer can come up with a set of components, links and procedures to implement the application. In the another situation, the designer is given a set of components, links and procedures. He or she must then use this to carry out the application.

The first situation described is more formally known as Architecture-to-task designs, the architecture A, of the system composed of a set of components, links and procedures, {C, L, P}; must be designed in order to meet the specifications of the given task. Similarly, the second situation is more formally known as Task-to-architecture designs. Here, the architecture A, composed of the set of components, links and procedures {C, L, P} is already established. The task must be designed in order to properly meet all specifications of the target application.

For the design of the system, a device must be selected to ensure that the required performance is achieved. The bottleneck of the system is the stereo correspondence algorithm. Executing the DPML algorithm by Cox *et al.* [1] in software, on a general-purpose processor produces very low frame rates, just above 3 fps [42]. This sequential implementation of the algorithm does not achieve the required performance, thus a different approach is needed. A parallelized approach to the DPML algorithm was introduced by MacLean *et al.* [17] which provides much higher frame rates. This approach requires using a device that has the flexibility of performing customized parallel operations at the same time. This narrows the choices down to programmable logic devices (*e.g.*, FPGAs) or application specific integrated circuits. The latter can operate using higher clock frequencies and provide better perfor-

mance, however the development costs are greatly higher. With this being said, we were provided with the Amirix AP1100 Platform (see Section 3.5) which includes a large FPGA, for which the processing tasks are to be developed in a Task-to-architecture design. The design of the video-acquisition system (see Section 3.4) took on the same approach, as it had to be tailored to work with the same processing platform.

3.3.2 Architecture of Image Warp Module

The purpose of the Image Warp Module is to generate rectified image coordinates so that the original stereo image pair can be warped to a rectified pair. The homographies needed to warp the original images are computed offline using the calibration parameters obtained using the Camera Calibration Toolbox for Matlab. Normalized image coordinates can be generated without requiring division by creating a Taylor Series approximation of the 3×3 homography matrices. This is important because performing division on hardware can be expensive in resources and lengthy in latencies.

During the assembly of the video-acquisition system, the placement of the image sensors and lens housing were mechanically adjusted to minimize the rotation between the two views. This allowed the Taylor Series Polynomial (TSP) to be of 1st-order. A poorly aligned assembly of the video-acquisition system could have resulted in higher rotational angles and larger translations. These factors alone could require 2nd- and 3rd-order Taylor Series Polynomials to achieve the desired level of accuracy. The general form of a 1st-order Taylor Series Polynomial is seen in Equation 3.1,

$$P = Ax + By + C \tag{3.1}$$

where A, B, C and P are real numbers. For this reason, we must use adders and multipliers that perform operations on numbers with fractional components (*i.e.*, real numbers). In digital systems, operations on real numbers can be performed using either floating-point

or fixed-point operators. Generally, IEEE 754 floating-point operators give more precise results over a larger dynamic range when using real numbers, the drawback for them are the long latencies for computations and large resources. An alternative is to use fixed-point operators: although they may not produce results to the accuracy of floating-point operators, they typically are faster and require less logic for implementation. With questions pertaining to the amount of resources we will have available for the end design of the complete system, we chose fixed-point operators to reduce our resource overhead as well as increase our throughput.

Our design of the Image Warp Module is parallelized as the 1st-order Taylor Series Polynomials require the least amount of multipliers and adders. We require four parallel units, to obtain (x, y) warped image plane coordinates for each of the left and right camera views. A generalized data flow graph of one Taylor Series Polynomial computation is shown in Figure 3.2, where A, B and C are fixed coefficients of the TSP. The module with four parallel TSP units uses 8 fixed-point adders and 8 fixed-point multipliers. The latency of the Image Warp module is 3 clock cycles, and the cycle time, or rate of producing the output, is equivalent to the rate of the inputs, which is either 4 clock cycles for the 30 fps system or 2 clock cycles for the 200 fps system.

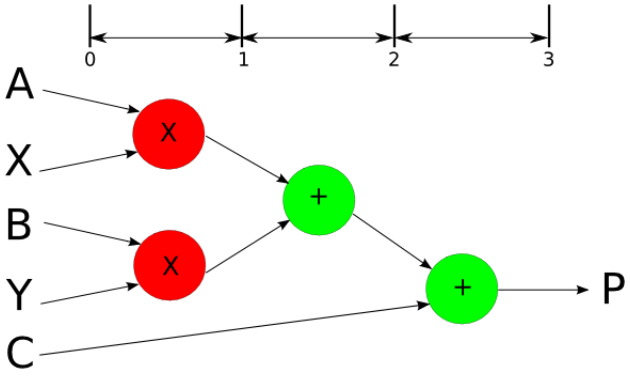


Figure 3.2: Taylor Series Polynomial computation data-flow graph

3.3.3 Architecture of Bilinear Interpolation Module

The Bilinear Interpolation Module serves as a means to estimate the pixel intensities of the rectified image from the original image given the transformed image coordinates $(x', y')^T$. In most cases the rectified pixel location may not map to an exact integer location in the original image, thus interpolating from the original image will require the pixel intensity to be computed from four neighbouring pixels. Similar to the Image Warp Module, this module is implemented using fixed-point operators. The design of the module is parallelized to maximize the throughput; two separate hardware units are implemented in parallel to process the pixel interpolation computation for the left and right views. The bilinear interpolation module was designed to implement the function shown in Equation 3.2 [43]. In this equation, $\triangle x$ and $\triangle y$ represent the fractional portions of the rectified pixel coordinates (see Section 3.3.2), while I_{old} and I_{new}represent the intensity of a pixel at a specific coordinate.

$$I_{\text{new}}(x, y) = (1 - \triangle x)(1 - \triangle y)I_{\text{old}}(x, y) + (1 - \triangle x)(\triangle y)I_{\text{old}}(x, y + 1)+$$

$$(\triangle x)(1 - \triangle y)I_{\text{old}}(x + 1, y) + (\triangle x)(\triangle y)I_{\text{old}}(x + 1, y + 1) \tag{3.2}$$

Figure 3.3 shows one hardware unit of this module, the full module contains two parallel units for the left and right views. Thus, 16 fixed-point multipliers and 6 fixed-point adders are used for the full implementation. The latency of the Image Warp module is 4 clock cycles, which represents the number of stages in the pipeline. The cycle time is equivalent to the rate of the inputs.

3.3.4 Architecture of Pixel Buffer

A portion of the incoming frame needs to be temporarily stored in order to compute correct pixel intensities, using the Bilinear Interpolation Module. The Pixel Buffer uses the addresses provided by the Image Warp Module, to output a 2×2 region of pixels. As the Image Warp Module provides different pixel addresses, only a limited region of the incoming frame will

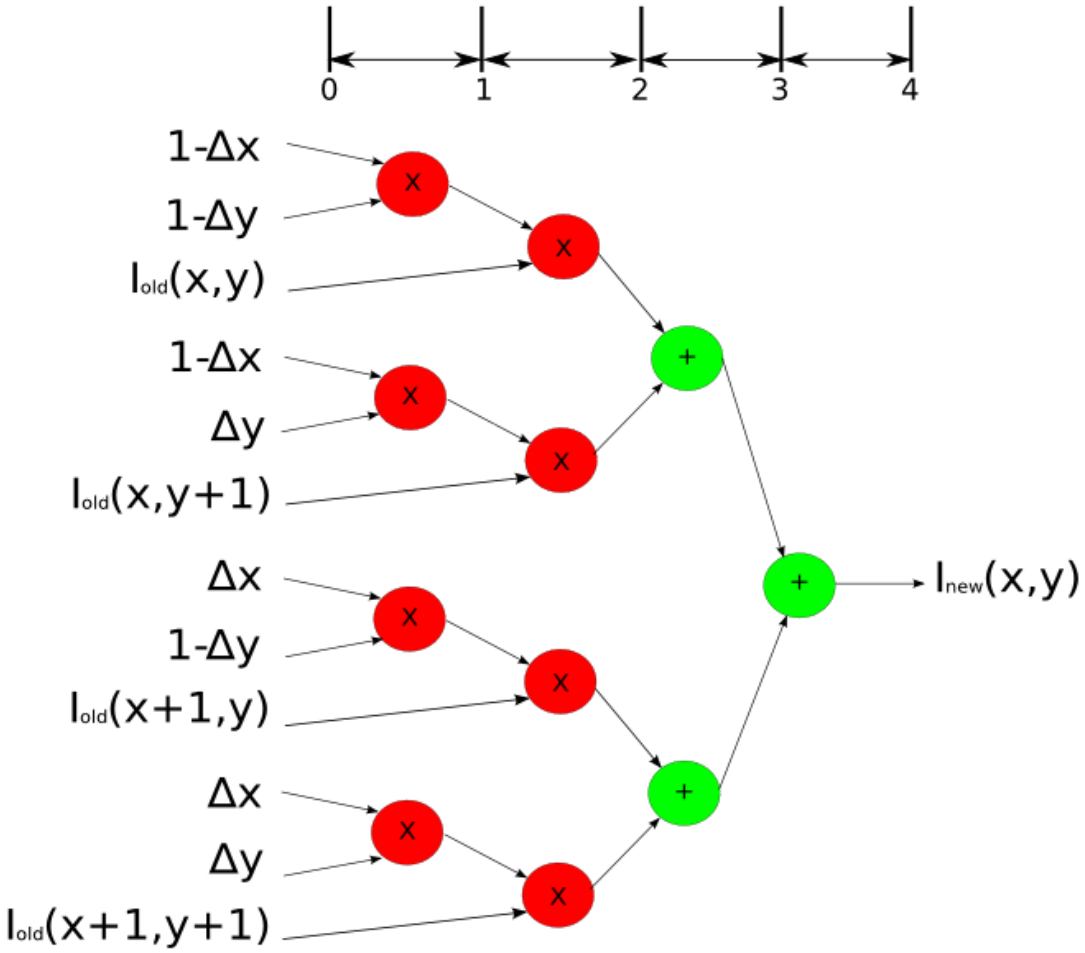


Figure 3.3: Bilinear Interpolation computation data-flow graph

be accessed. Thus, if a circular buffer is used, we can guarantee that the required portions of the incoming frame will always be available during processing. The incoming stream data is saved following a Least Recently Used strategy.

The Pixel Buffer is composed of a number of internal block RAMs (BRAMs), which are the basis for storing and retrieving pixels. The BRAMs are referenced using logic to identify which row of the frame they represent. The pixels are accessed from the BRAMs using row and column addresses supplied by the Image Warp Module. Since performing the bilinear interpolation requires a 2×2 region of pixels, subsequent BRAMs can be read in parallel.

As the video-acquisition system provides the pixels of the left and right views, the Pixel Buffer was designed to store and retrieve pixel information based on the scanline value.

Each scanline of pixels is stored into a different BRAM instance for indexing purposes. The number of BRAMs required in the architecture of the Pixel Buffer can be calculated by computing the maximum difference between the rectified Y-coordinates and the original Y-coordinates. In addition, a few extra BRAMs were instantiated to avoid ever reading and writing simultaneously to the same BRAM. The amount of BRAMs required can vary based on alignment of the images sensors and lens housing (*i.e.*, camera calibration parameters). The use of generic parameters in the VHDL code of the Pixel Buffer allows changing the size of the buffer in a simple manner. This saves the user time, by not having to manually modify portions of the code. This component can be used not only for bilinear interpolation, but for any image processing algorithm which performs processing on a 2×2 region of pixels (*e.g.*, an edge detector).

The differences between the design of the 30 fps and 200 fps Pixel Buffer are quite small. The design of the Pixel Buffer for the 30 fps system utilized single-ported BRAMs to save the contents of scanlines. Here, each BRAM would contain the contents of only one scanline with the data addressed by the column value. On the other hand, the design of the Pixel Buffer for the 200 fps system utilized dual-ported BRAMs. The dual-ports on the memory device allows multiple access of the memory at the same time. Furthermore, the dual-ports make it convenient to store multiple scanlines into each BRAM. We saved two scanlines into each BRAM by manipulating the addressing. For the first scanline being saved into the BRAM, we assign the upper most bit to '0' followed by the column value; for the second scanline being saved into the same BRAM, we assign the upper most bit to '1' followed by the column value.

3.3.5 Architecture of Stereo Rectification Module

The architecture of the video pre-processor is designed to implement the modules described in the previous sections. The architecture is fully pipelined and synchronous, which allows it to perform at the required frame rate. The implementations of the 30 fps and 200 fps Stereo

Rectification Modules (SRMs) are discussed next.

The Stereo Rectification Module interfaced with the 30 fps video-acquisition system (see Section 3.4.1) has seven main components and is shown in Figure 3.4. The Pixel Buffer, Image Warp and Bilinear Interpolator have been described previously. The other components include a Counter which provides the original (x, y) coordinates in fixed-point nomenclature. It begins counting when it receives a start signal from the Pixel Buffer. The Pixel Buffer asserts a start signal which indicates enough pixels have been buffered to correctly perform the image transformation. An Out-of-Range signal is used to identify whether the warped (x', y') coordinates exist within the range of the original frame $(0, 0) \longrightarrow (639, 479)$. It also separates the data produced by the Image Warp Module into integer and fractional portions, which are used as the read address of input pixels and the coefficients for bilinear interpolation, respectively. The 5-stage FIFO is used to synchronize coefficients for bilinear interpolation with the 2×2 region of pixels from the Pixel Buffer when they are available. Finally, the Rectification Buffer component is used to store the rectified pixels into a temporary location for video display purposes and to pass them to the stereo extraction unit.

The coefficients used after the conversion of the 3×3 homographies into the normalized 1st-order Taylor Series Polynomial are listed in Tables 3.1 and 3.2. As several 30 fps video-acquisition systems have been constructed, the coefficients mentioned pertain to the SFGv1-Ryerson board. The left view warped image plane coordinates are subscripted with X_L and Y_L, similarly the right view warped image plane coordinates are subscripted with X_R and Y_R.

Table 3.1: 30 fps TSP coefficients for left view

X		Y	
A_{XL}	1.0039	A_{YL}	-0.0059
B_{XL}	0.0057	B_{YL}	1.0015
C_{XL}	-22.8353	C_{YL}	1.9368

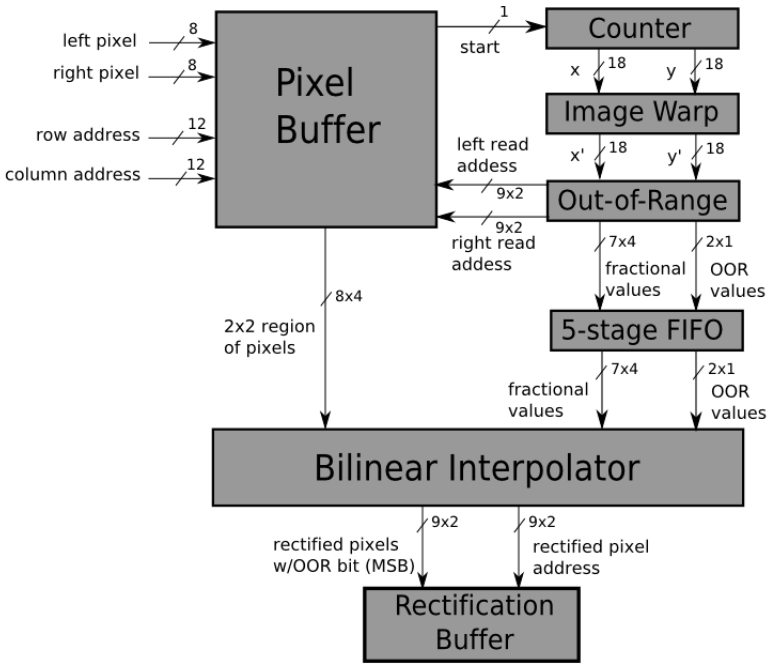


Figure 3.4: 30 fps SRM block diagram

Table 3.2: 30 fps TSP coefficients for right view

X		Y	
A_{XR}	1.0069	A_{YR}	-0.0070
B_{XR}	0.0065	B_{YR}	1.0037
C_{XR}	-47.4809	C_{YR}	5.0099

The block diagram of the Stereo Rectification Module interfaced with the 200 fps video-acquisition system (see Section 3.4.2) is shown in Figure 3.5. This SRM has six main components. Once again, the modified Pixel Buffer, Image Warp and Bilinear Interpolator have been described previously. The other components, include the Address Generator which provides the original (x, y) coordinates in fixed-point nomenclature. The Out-of-Range signal identifies whether the warped (x', y') coordinates exist within the range of the original frame $(0, 0) \longrightarrow (639, 479)$. It also separates the data produced by the Image Warp Module into integer and fractional portions, which are used as the read address of input pixels and the coefficients for bilinear interpolation, respectively. Finally, the Sync Control component is used to temporarily hold the fractional portion of the location data from the Image Warp

Module as well as the Out-of-Range (OOR) value for the current left and right pixels. The Sync Control also provides the start signal for the Address Generator, indicating that enough pixels have been buffered to correctly transform the image.

The coefficients used after the conversion of the 3×3 homographies into the normalized 1st-order Taylor Series Polynomial are listed in Tables 3.3 and 3.4. As several 200 fps video-acquisition systems have been constructed, the coefficients mentioned pertain to the SFGv2-Ryerson board. The left view warped image plane coordinates are subscripted with X_L and Y_L, similarly the right view warped image plane coordinates are subscripted with X_R and Y_R. Two 9-bit outputs are produced by the module with the same associated address. Each 9-bit result contains the OOR value on the most significant bit (MSB), while the lower 8-bits contain the rectified pixel intensity.

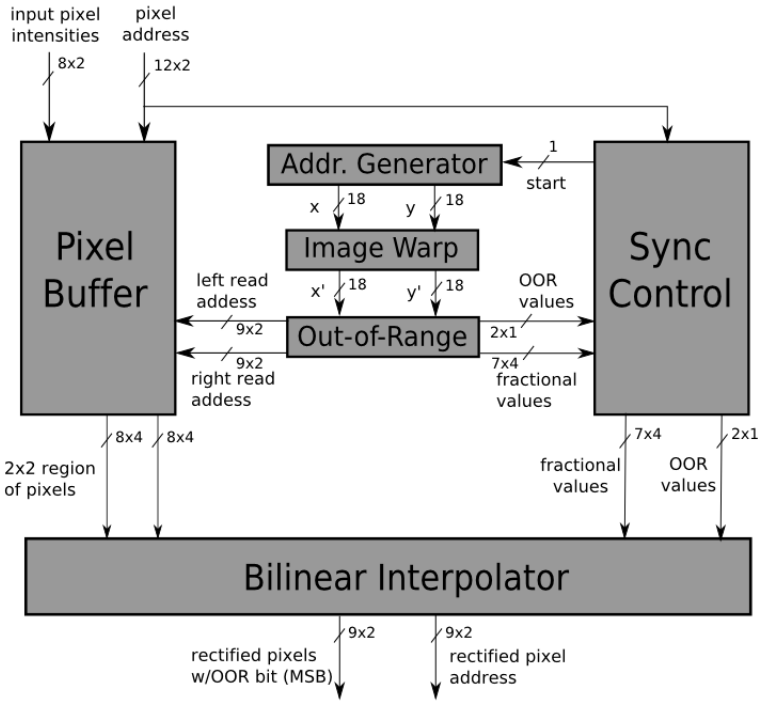


Figure 3.5: 200 fps SRM block diagram

Table 3.3: 200 fps TSP coefficients for the left view

X		Y	
A_{XL}	0.9985	A_{YL}	-0.0206
B_{XL}	0.0184	B_{YL}	0.9942
C_{XL}	7.4212	C_{YL}	8.1546

Table 3.4: 200 fps TSP coefficients for the right view

X		Y	
A_{XR}	0.9997	A_{YR}	-0.0157
B_{XR}	0.0124	B_{YR}	0.9982
C_{XR}	-2.0650	C_{YR}	11.2222

3.3.6 Timing Analysis

The architecture of the 200 fps, explored in Section 3.3.5, will now be analyzed using timing diagrams. The operation of the SRM begins when the *start* signal is asserted following a reset of the system, as seen in Figure 3.6. The arrow in Figure 3.6 indicates when the *start* signal is asserted. The calibration characteristics of this system requires buffering 16 rows of the first frame before the operation can begin. The start signal remains high until the system is reset.

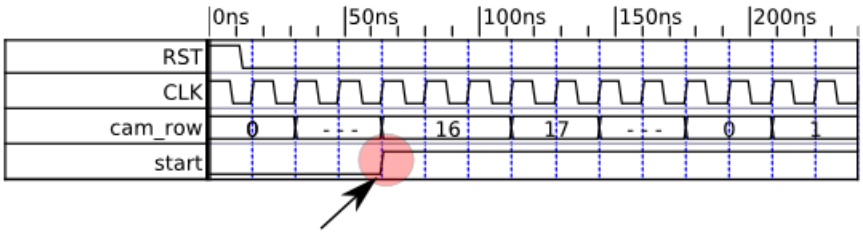


Figure 3.6: Timing diagram of the *start* signal

One clock cycle following the assertion of the *start* signal, the original (x, y) coordinates are generated by the Address Generator in fixed-point nomenclature. This is pointed out by the arrows and bubble shown in Figure 3.7. The shaded regions shown in the figure identify data that are invalid, while the valid regions are labeled to indicate the corresponding row and column values. Also seen in Figure 3.7 is the generation of the warped (x', y') coordinates

by the Image Warp Module. The Image Warp Module begins processing on the input information one clock cycle after it is produced by the Address Generator. The figure also shows a window of 4 clock cycles, which indicates the amount of time required to produce and latch the output. The outputs of the Image Warp Module are labeled to indicate the row and column values they are associated to; this labeling scheme will be used for the rest of the timing diagrams.

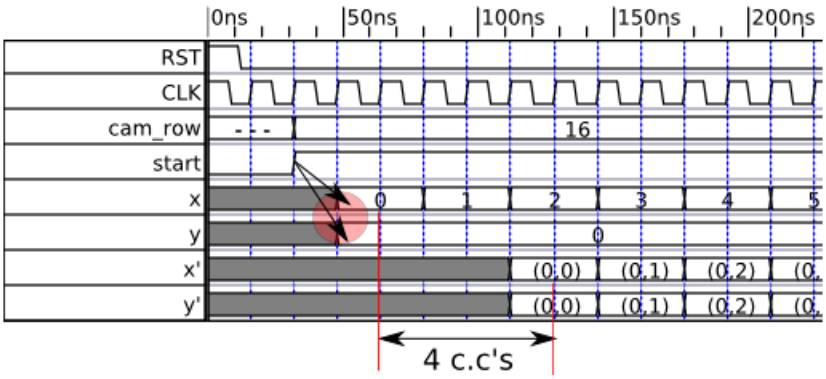


Figure 3.7: Timing diagram of the Address Generator and Image Warp Module

The generation of the out-of-range signal, *oor*, and pixel read address values are shown in Figure 3.8. The arrow indicates that only one clock cycle is required to generate the *oor* output and the pixel read addresses. The output of the Pixel Buffer, two 2×2 pixel regions, is shown in Figure 3.9. The Pixel Buffer uses the pixel read addresses generated by the Out-of-Range Module one clock cycle after they are produced. The figure shows a window of 6 clock cycles, which indicates the amount of time required to read and latch 8 pixels.

The Bilinear Interpolation Module, which is the final stage of the system, uses the 2×2 pixel regions to estimate the pixel intensity. Figure 3.10 shows a window of 5 clock cycles, which indicates the amount of time required to produce and latch the output. Finally, Figure 3.11 shows an overall timing diagram of the SRM. The figure shows that an initial 17 clock cycle latency is required to produce the first set of rectified pixels.

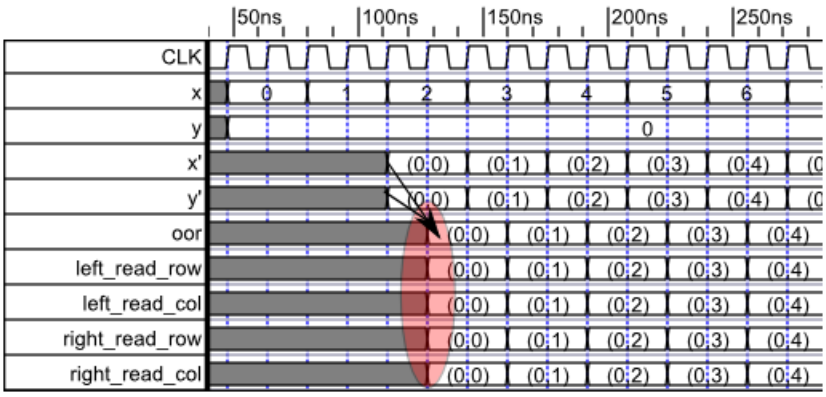


Figure 3.8: Timing diagram of the Out-of-Range Module

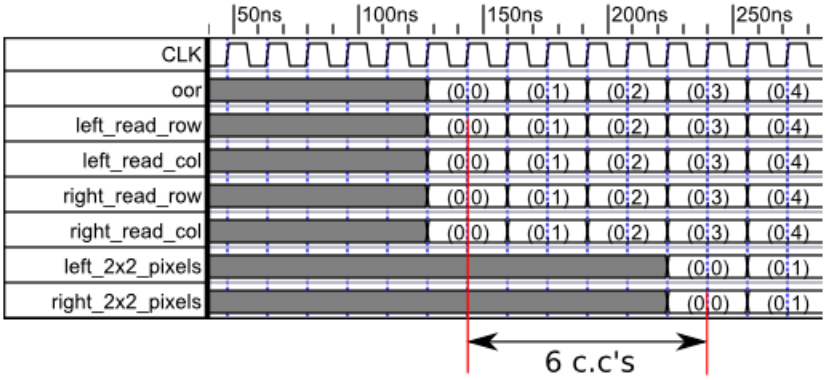


Figure 3.9: Timing diagram of the Pixel Buffer

3.4 Video-acquisition system

The purpose of the video-acquisition system is to capture a pair of images, which represent two different views of the same scene. Cameras that provide two simultaneous views of a scene for three-dimensional reconstruction are known as stereo cameras. In our case, a more accurate identification of our camera would be a *binocular* stereo camera, since it provides two views of the scene. Typically for human beings the baseline of the eyes is about 6.5 cm, and this separation allows us to determine the distance to objects in our field of view. The baseline of our stereo camera is approximately 16 cm. This system acquires data in real-time from two CMOS sensors at a resolution of 640×480 pixels. The image processing tasks involved will be discussed in Section 3.3. Initially, a low-speed video-acquisition system was developed for initial proof-of-concept; this system performs standard video-acquisition

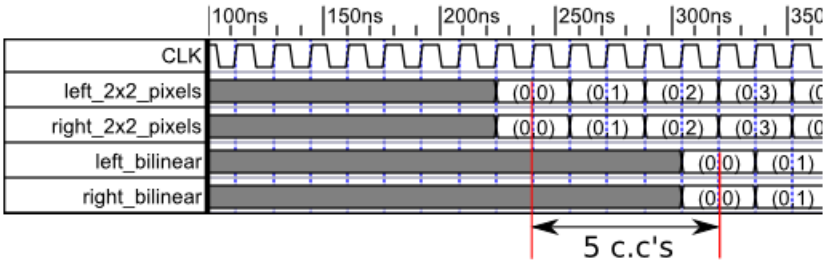


Figure 3.10: Timing diagram of the Bilinear Interpolation Module

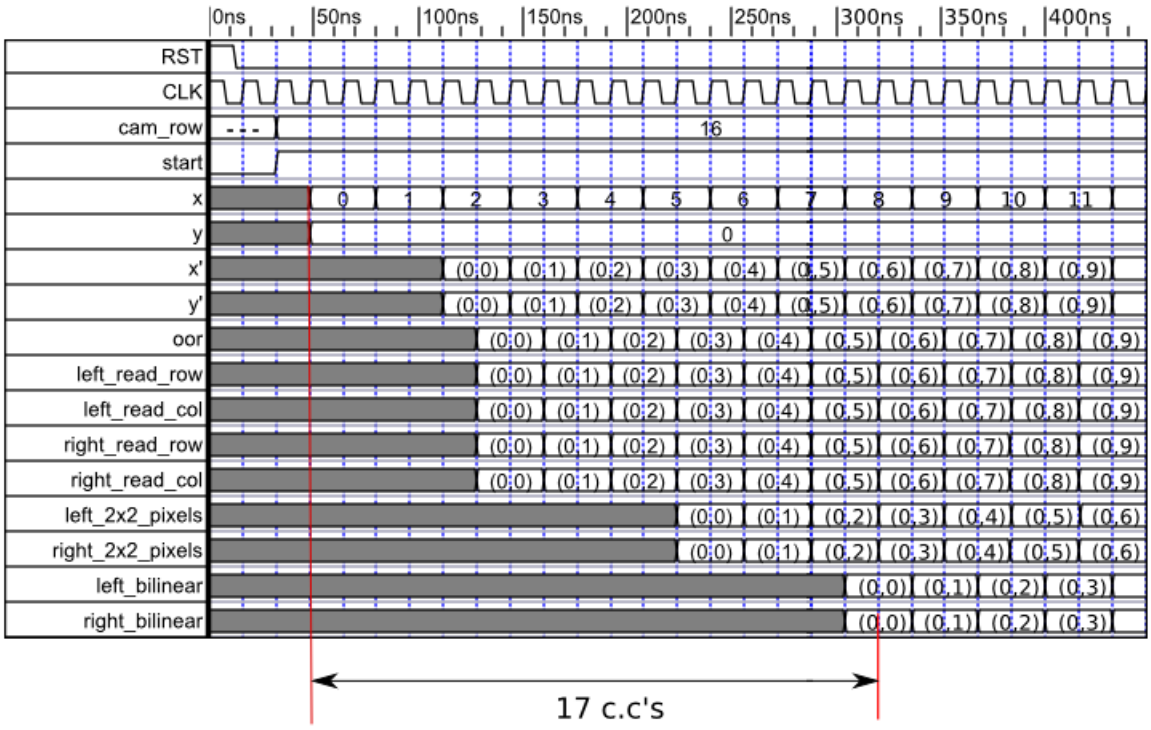


Figure 3.11: Timing diagram of the SRM

at 30 fps. The ultimate goal of the system is to achieve high frame-rates, and thus a second, high-speed video-acquisition system was developed, achieving performances up to 200 fps. The data produced by the video-acquisition system is sent to the FPGA on the Amirix AP1100 Platform (see Section 3.5) via the high-speed PCI Mezzanine Card slot as seen in Figure 3.22. The following sections describe the architecture of the two video-acquisition systems.

3.4.1 30 fps Video-acquisition

The first generation of the video-acquisition system operates at speeds of 30 fps. The interconnection of the video-acquisition system is composed of a stereo camera, also known as the Stereo Frame Grabber v.1 (SFGv1), the Amirix Daughter Card, and the Amirix AP1100 Platform (see Section 3.5).

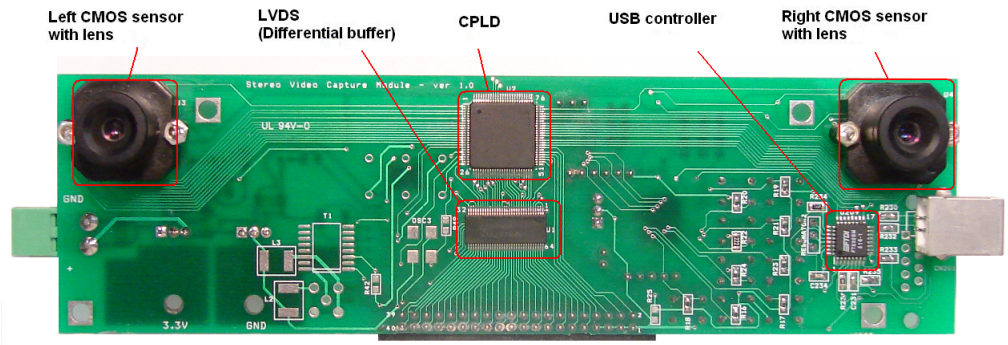


Figure 3.12: 30 fps Stereo Frame Grabber (front)

The SFGv1 and the Amirix Daughter Card were developed by Kirischian *et al.* [44] and can be seen in Figures 3.12, 3.13 and 3.14, respectively. The Amirix Daughter Card provides

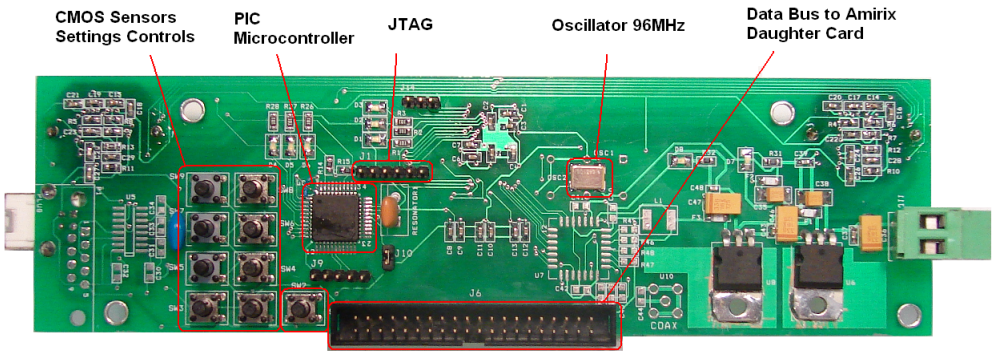


Figure 3.13: 30 fps Stereo Frame Grabber (back)

as a channel through which the data from the SFGv1 can be sent to the FPGA. This is possible since the Amirix Daughter Card has a PCI Mezzanine Card slot which plugs directly

into the Amirix AP1100 Platform.

Figure 3.13 shows the back view of the SGFv1 which contains a 20-pin port for interfacing with the Amirix Daughter Card, using a standard ribbon cable. The Amirix Daughter Card redirects the data coming from the SFGv1 through an on-board LVDS differential buffer to the PCI Mezzanine Card slot, which allows sending the SFGv1 data at high-speeds to the Amirix AP1100 Platform.

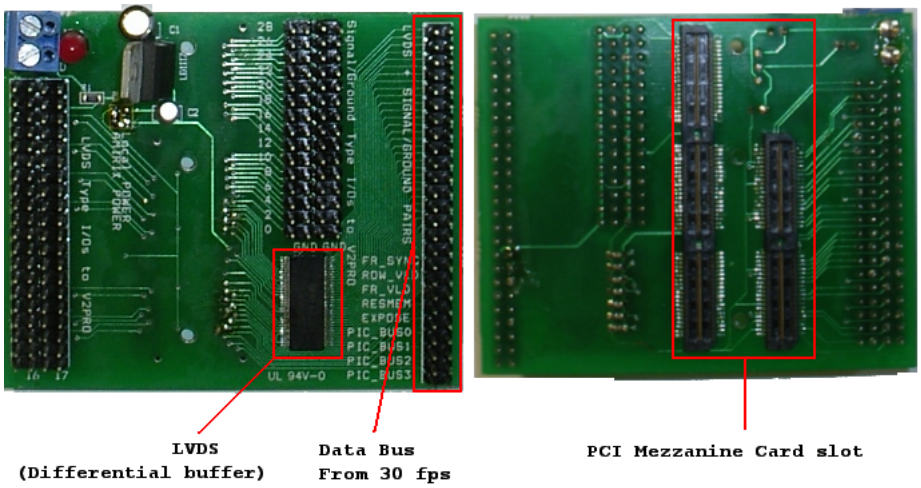


Figure 3.14: Amirix Daughter Board - front & back views

The architecture of the system includes three modules; the Image Capture Module (ICM), Video Processing Module (VPM) and the VGA Output Module. A high-level block diagram of the system is shown in Figure 3.15.

The Image Capture Module is composed of two CMOS image sensors, a Complex Programmable Logic Device (CPLD), a microcontroller, a Universal Serial Bus (USB) controller and user control buttons. The image sensors are used for capturing the scene, however they are set to operate in slave mode which allows them to operate synchronously with each other. Selecting the image sensors to operate in slave mode allows the designer to control when data should be captured via external clock and control signals. The CMOS sensors generate colour pixel data in a Bayer pattern format [45]. Full colour images usually consist of a pixel that contains at least three components, most typically red, green and blue. Raw images based on Bayer patterns have only one of the three colour components at each pixel location. The

Bayer pattern was created by Bryce Bayer and filters only one colour onto any given pixel in a square grid of photosensors. Figure 3.16 shows an example of the Bayer pattern array. It can be noticed that 50% of the array is assigned with green filters, while 25% is assigned to each of the red and blue filters. Bayer developed the array with a larger proportion of green filters to mimic the human eye's greater sensitivity to green light. The green elements in the array are known as *luminance-sensitive*, while the red and blue elements are known as *chrominance-sensitive*. The challenge has to do with the reconstruction of the two missing components at each pixel location. There are many algorithms which are used to obtain the missing colours, however a simple and effective approach is through nearest-neighbour approximation, which uses a 3×3 window of pixels to average the missing colours. This task is performed on the FPGA. Adjustments to the settings of the image sensors can be controlled by a user, by manipulating the user control buttons. The buttons allow for adjustments to the image sensors' global gain values, and gain values for each color channel - Red, Green and Blue (RGB). The changes are received by the embedded microcontroller which then passes the values through to the Complex Programmable Logic Device (CPLD). The CPLD is the controller of the two image sensors as it provides the sensors with the master clock, and other control signals to synchronize the image capture. The CPLD is also responsible for directing the captured output from the image sensors through high-speed LVDS differential buffer to the FPGA on the Amirix AP1100 platform.

The VPM is implemented on the FPGA found on the Amirix AP1100 Platform (see Section 3.5). It uses the data captured by the ICM, separates the data related to each video channel and stores it into the dedicated memory space. The VPM is composed of a Static RAM (SRAM) main controller, SRAM read controller and SRAM write controller. Essentially, the SRAM main controller produces the signals needed to read and write from the two external SRAM banks found on the Amirix AP1100 Platform. The SRAM write controller is used for sending the data and the write address to the SRAM main controller. The SRAM read controller is used for generating the read address for retrieving the pixels previously

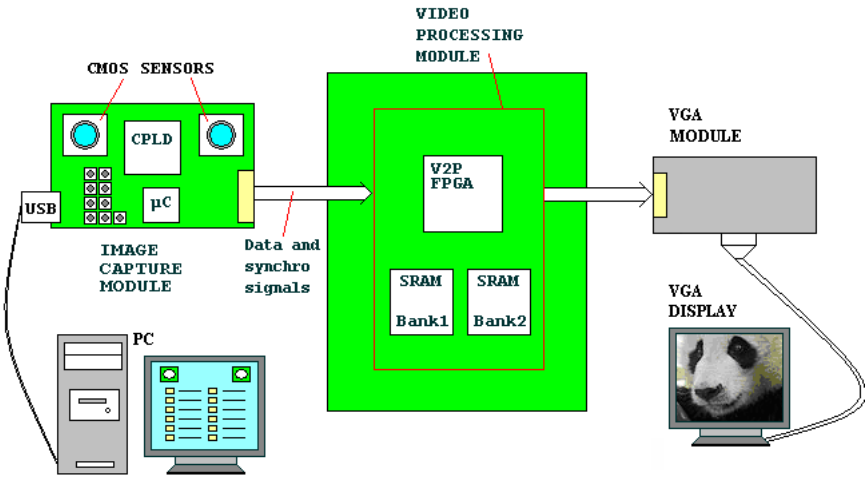


Figure 3.15: Block diagram of 30 fps video-acquisition system [46]

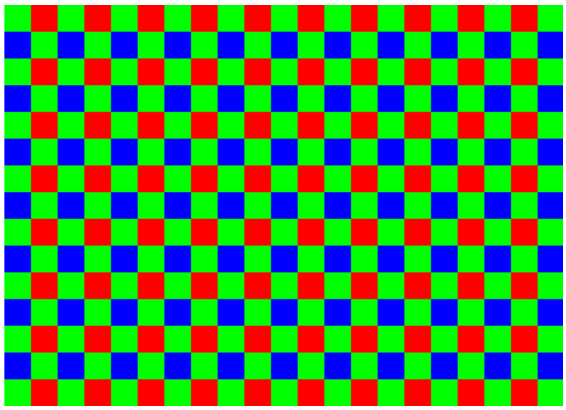


Figure 3.16: Bayer pattern array

saved in SRAM. Furthermore, it buffers the corresponding pixels from the previous scanline in order to interpolate RGB values for each pixel. The computation of the correct pixel colour intensity is sent as an output, along with the horizontal and vertical synchronization signals (HSYNC, VSYNC) for visualization via the VGA Output Module.

The VGA Output Module is a simple digital-to-analog converter (DAC) which is used to drive standard VGA analog inputs along with a VGA controller core. The VGA Output Module also contains a switch which is used to visualize the data captured from either the left or right camera of the ICM, to a standard VGA display. For additional details on the design and implementation of the 30 fps video-acquisition module, please refer to [46].

3.4.2 200 fps Video-acquisition

The second generation of the video-acquisition system operates at speeds up to 200 fps. This acquisition system can capture objects with faster motion and shorter exposure time than the 30 fps system. A comparison of the 30 fps and the 200 fps video-acquisition system can be made with a simple example. Imagine a computer cooling fan with dark blades. Suppose we were to paint one of the blades white and then start the fan. Next, let's capture images of the fan in operation using both the 30 fps and 200 fps video-acquisition systems. What would we notice? The images from the low-frame rate system would show the fan in motion, however we would not be able to distinguish the white blade apart from the rest of the blades due to motion blur. The motion blur is a function of the exposure time, while the higher capture rate allows measuring the motion in smaller increments. With the visualization of the high frame-rate system, we would be able to view the motion of the fan in slow motion. Subsequently, we would be able to distinguish the white blade apart from the others while the fan is rotating.

The second generation of the stereo camera is known as the Stereo Frame Grabber v.2 (SFGv2) developed by Chun *et al.* [47]. A number of changes have been made to this revision of the video-acquisition system. The most significant change is enhanced performance due to the change in the image sensors. The sensors contain TrueSNAP technology [48] which use electronic shutters that stop even the fastest motion with crystal clear accuracy. It was created for use in high-speed applications that allow for all pixels to be simultaneously exposed. The CMOS image sensors used are capable of operating at frame-rates reaching 400 frames per second, however at reduced resolutions. For the standard resolution of 640×480 pixels, the system operates at 200 fps. Two VGA Output ports are integrated onto the SFGv2, allowing the user to debug and verify any image processing done within the system. As an example, the two VGA Output ports can be used to display the rectified images of each view which places points (features) on the same scanline. Another important change made to the module is the direct interface to the Amirix AP1100 Platform (see Section 3.5)

using on-board PCI Mezzanine Card slots. This modification removes any cables that might have been present in the previous version. Furthermore, it allows the system to operate stand-alone independent of a personal computer (PC), by utilizing a CompactFlash card which loads designs on to the FPGA during power up. Typically, the PC-dependent Amirix AP1100 Platforms are enclosed within the desktop casing and this prevents the system being portable. The system becomes independent of a PC when it is powered by the Amirix PCI Adapter Board which provides enough current to operate the Amirix AP1100 Platform. Illustrations of the SFGv2 and Amirix PCI Adapter Board are shown in Figures 3.17 and 3.18, respectively.

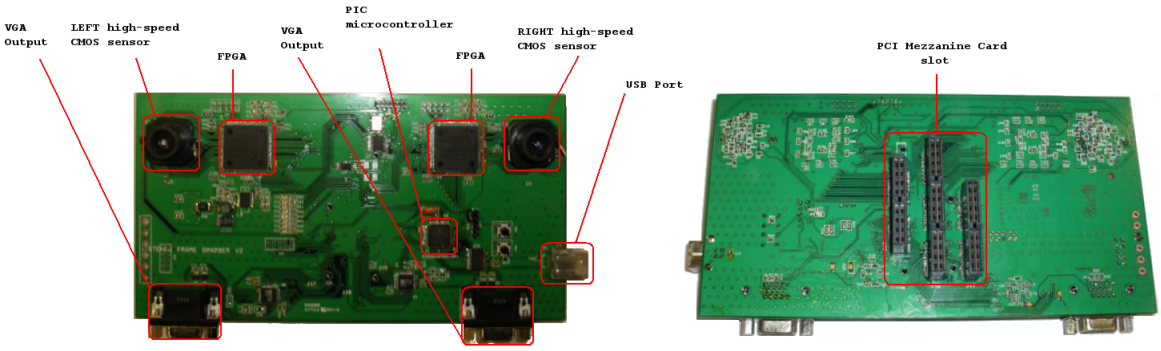


Figure 3.17: Front and back views of the Stereo Frame Grabber v.2

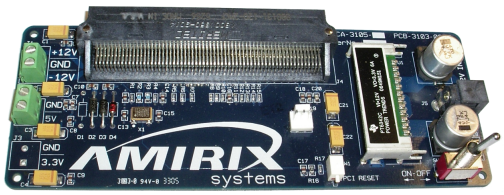


Figure 3.18: Amirix PCI Adapter Board

The main components of the SFGv2 are highlighted in Figure 3.17. They include the left and right high-speed CMOS image sensors, two FPGAs for control of the two CMOS image sensors, two VGA output ports, a microcontroller, a USB controller, and PCI Mezzanine Card slots. A block diagram of the SFGv2 is shown in Figure 3.19. The block diagram shows the interconnections between the different components on the SFGv2 and how it interfaces with the Amirix AP1100 Platform. It can be seen that the interconnection between

the SFGv2 and the Amirix AP1100 Platform occurs through the high-speed PCI Mezzanine Card (PMC) interface. The USB interface provides information from the FastTrack Image Grabber, which is the Graphical User Interface (GUI) used to adjust the image sensor parameters, such as the total gain, and gain values for three colour channels - Red, Green, Blue. More information on the GUI will be discussed shortly. Other parameters include the reference voltages supplied to the sensors. The adjustments are then sent to the image sensors via the microcontroller and FPGAs. The FPGAs have a dual-purpose in the design of the SFGv2; aside from changing the parameters of the image sensors, the FPGAs are used to interpolate the Bayer patten information received from the CMOS image sensors into the standard RGB pixel information.

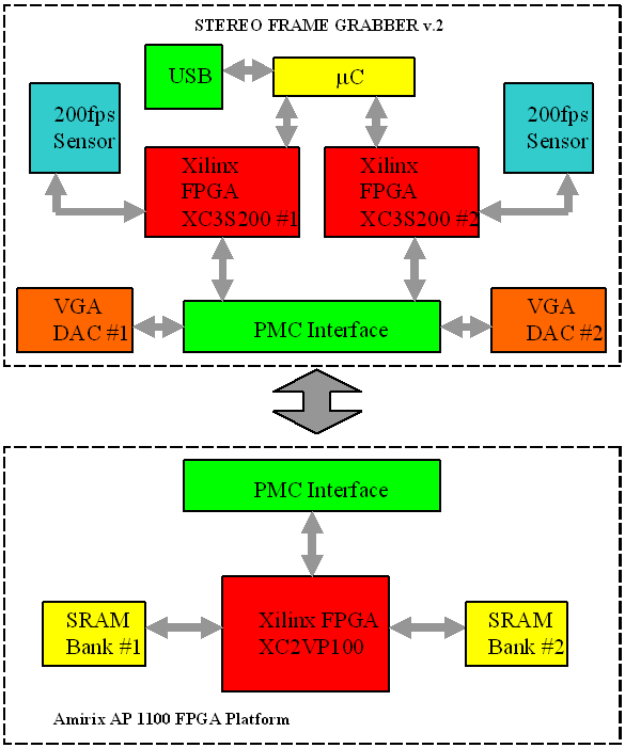


Figure 3.19: Stereo Frame Grabber v.2 Block Diagram

This section provides an overview of the features of the FastTrack Image Grabber. A screen capture of the GUI is shown in Figure 3.20. The FastTrack Image Grabber allows controls of the image sensors; all control registers described in [48] are found in the applica-

tion. Besides changing the settings of the image sensors, the GUI has buttons which initiate capturing a new frame or a stream of frames for visualization purposes. Other buttons found in the GUI initiate continuous capturing of the scene and downloading a pair of images. The hardware image download is a useful feature for capturing stereo images, which are needed to perform camera calibration to extract camera parameters. This is also useful for verifying the operation of the different modules in the system. As an example, this feature can be used to download disparity maps produced by the FPGA for comparison purposes with ground truth disparity maps. Another useful feature is interval capturing and download, which captures and downloads N images, where N is the number from the *Number of Captures* field. The numbering of the images begins with the number associated in the *Starting #* field, this number increments after each download. The pair of images downloaded are stored in files $< left\_imageN.bmp >$ and $< right\_imageN.bmp >$.

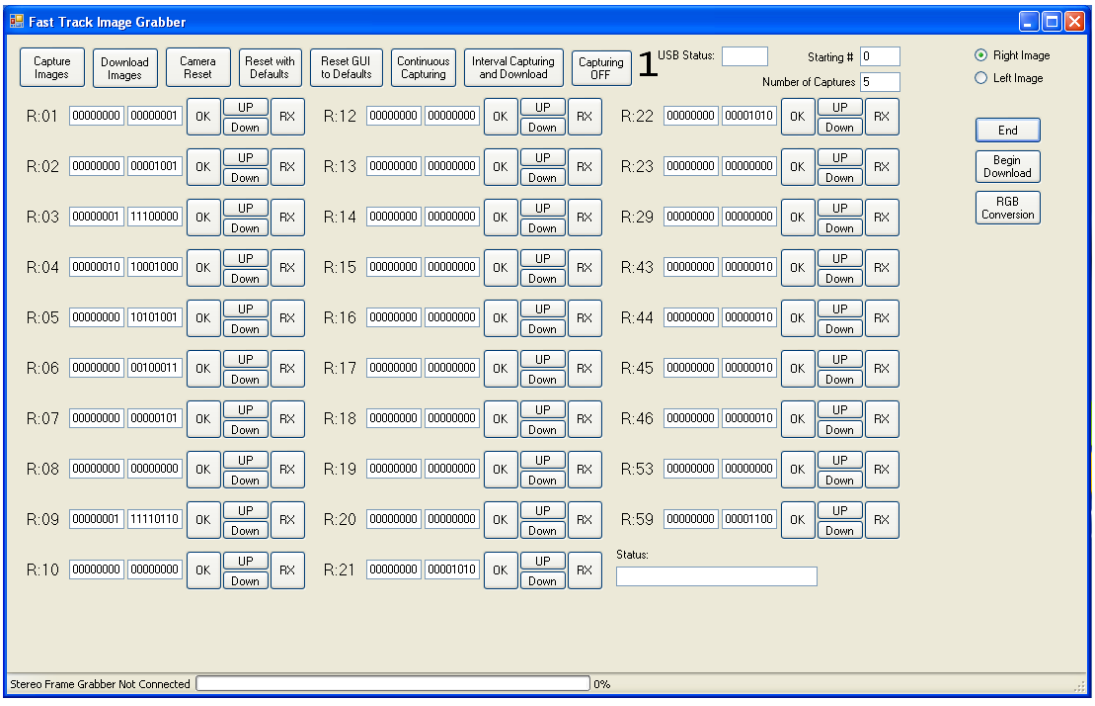


Figure 3.20: Screen Capture of FastTrack Image Grabber

We return back to the block diagram of the SFGv2. We have not yet discussed the interconnection of the SRAM banks and the VGA outputs. The SRAM banks are used in

the system to save any processed information. The 3-channel pixel image is an example of something that can be stored in SRAM. An addressing scheme is needed to simplify how the image is read back for display. Usually, row and column values for each pixel are used for this scheme. Data is written into the SRAM banks when a new frame is captured using the GUI. The information from each SRAM bank is then read using an embedded VGA controller, and the data is passed over the PCI Mezzanine Card interface to the VGA output ports on the SFGv2. Each SRAM bank contains the information which is displayed to each VGA output port.

3.5 Amirix AP1100 Platform

The flexibility of programmable logic, specifically FPGAs, allows rapid design implementations. This allows the user to explore many different design paths in a quest to find the most suitable solution and enables proof-of-concept without incurring the time of chip fabrication [49]. A reduction in cost also appears in designs for Intellectual Property (IP) and complete systems by utilizing FPGAs when compared against Application Specific Integrated Circuit fabrication. This statement is true when a limited number of units is produced for the design. When a large volume of production is required, it is often more economical to develop the IP or system as an ASIC [19].

The Amirix AP1100 PCI Platform FPGA Development Board allows developers to explore embedded system architectures in areas such as image and video processing, communications, digital signal processing, networking, industrial controls, instrumentation, testing and measurement. The platform is equipped with the following:

- Xilinx Virtex-II Pro XC2VP100 FPGA with two IBM PowerPC (PPC405) cores
- Xilinx SystemACE with CompactFlash configuration card supporting FPGA configuration

- Two banks of 64 Megabyte (MB) Double Data Rate Synchronous Dynamic Random Access Memory
- Two banks of 2 MB Static Random Access Memory
- 16 MB Program Flash
- 16 MB Configuration Flash
- 64-bit/66MHz PCI
- IEEE-P1386.1 32-bit/66MHz PCI Mezzanine Card slot for third party I/O cards
- Two 10/100/1000 BASE-T Ethernet PHYs and 10/100 Ethernet Controller
- Two RS-232D Serial Ports
- Multi-Gigabit Transceivers (aka Rocket I/O)
- Access to user switches and LEDs

A block diagram of the Amirix AP1100 PCI Platform is shown in Figure 3.21. It should be noted that this illustration shows only one IBM PowerPC (PPC405) core, however, in reality there are two. Figure 3.22 shows an image of the platform to be used.

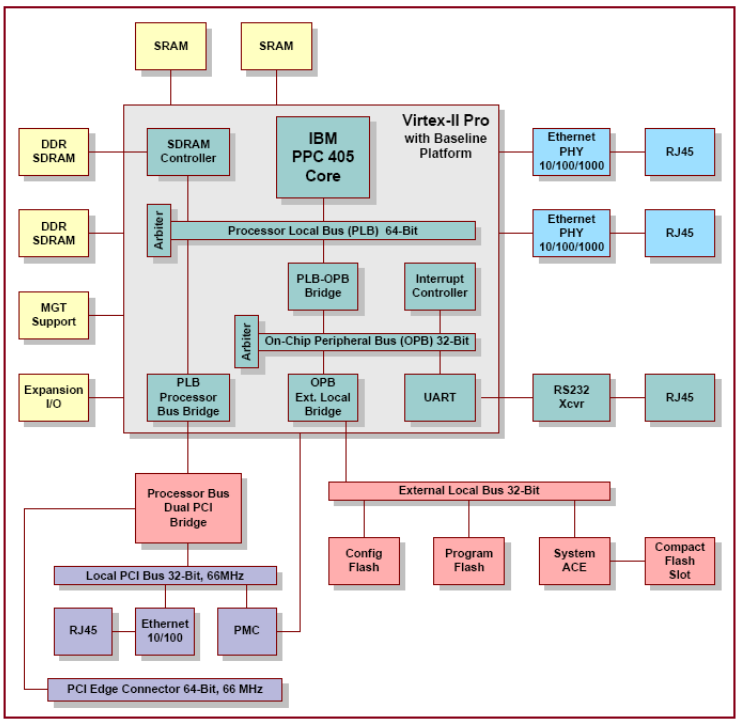


Figure 3.21: Amirix AP1100 PCI Platform block diagram

The Xilinx Virtex-II Pro FPGA included in the platform is the largest device in the Virtex-II Pro family. It contains 2 PowerPC processor blocks which are hard processors, 99216 logic cells, 44096 slices, 1378 Kb distributed RAM, 444 18×18 multiplier blocks, 444 18Kb block RAMs and 12 digital clock managers (DCMs) [50]. The device also supports loading the 32-bit MicroBlaze processor core which is a soft processor, created out of the configurable logic found on the FPGA. One special feature that is available on this particular device is know as partial reconfiguration (PR). Partial reconfiguration is useful for applications that require different designs to be loaded into the same area of a chip, or applications that require the ability to change portions of a design without having to reset or reconfigure the entire chip. Partial reconfiguration can be accomplished in either slave selectMAP mode or boundary-scan mode. Instead of resetting the device and doing a full configuration, new data is loaded into a specified area of the chip, while the rest of the chip remains in operation. More details on this FPGA can be found from the data sheets provided on [51].

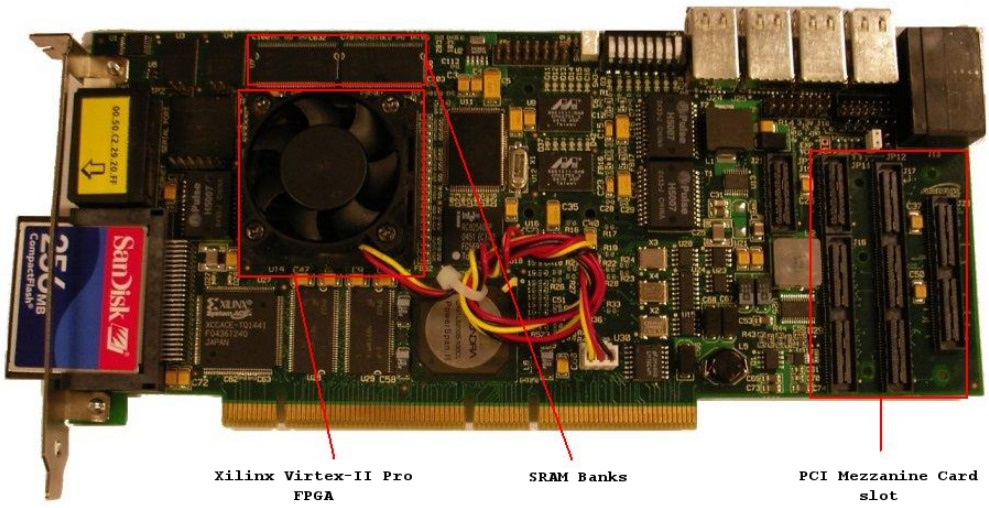


Figure 3.22: Amirix AP1100 FPGA Development Board

3.6 System Integration and Verification

One of the most daunting and time consuming tasks involved in system design is the stage of integration. The major challenge for the system integration was to create interfaces that allowed the data flow between different modules to be scheduled. Another challenge was to meet the critical timing constraints of the stereo vision system on the FPGA. A final challenge was to boost the performance of the Stereo Extraction Module. Section 3.6.1 discusses the interfacing of the video-acquisition system with the video pre-processor. Initially, an attempt was made to integrate the full system, including Stereo Extraction and Object Tracking algorithms, into the design with the 30 fps video-acquisition system. It was soon realized that timing and performance issues existed between the different stages in the system. The system was also limited in its debugging capabilities. Downloading the hardware outputs at different stages in the system was time consuming. In the end, we decided to make improvements to the algorithms and spend more time integrating the high-speed, final version of the system. This is overviewed in Section 3.6.2.

Collaborative research was done in liaison with the Vision & Image Dynamics Lab at the University of Toronto for the joint implementation of the Stereo Extraction Module. Before the joint work took place, the performance of the Stereo Extraction Module was approximately 39 fps. Improvements to the module were required to increase the performance and to ultimately meet the goals of the system (*i.e.*, operational at 200 fps). The modifications made to the Stereo Extraction Module in collaboration with Vision & Image Dynamics Lab will be discussed in Section 3.6.3. The modifications allow it to operate at 123 fps, meaning that two cores of the Stereo Extraction Module operating in parallel can process the 200 fps data produced by the video pre-processor. It should be mentioned that the Stereo Extraction Module follows the DPML stereo algorithm developed by Cox *et al.* [1] and is overviewed in Chapter 2. Verification of the different stages of the system was required to assure correct functionality. Different tools were utilized for each stage of testing and verifying. A list of these tools can be found in Table 3.5.

Table 3.5: Testing and debugging methods

Tool	Purpose
Chipscope Pro Logic Analyzer	To monitor the status of internal FPGA signals
ModelSim XE Simulator	To test and verify the operation of HDL for a component or system
HP Logic Analyzer	To view the status of signals from pins external of the FPGA
SRAM Frame Download	To verify the correct functionality of different stages in the system

3.6.1 Hardware Interfacing

The interfacing portion of the system integration required developing the video pre-processor in a Task-to-architecture design manner. The system was developed based on how often the data and control signals were produced by the video-acquisition system. The video-acquisition system's data rate was analyzed by inserting a Chipscope Pro Logic Analyzer IP-core into its design, this allowed us to verify that pixel and address information appeared every two clock cycles for the left and right camera, simultaneously. Figure 3.23 shows the interfacing of the video-acquisition system and the video pre-processor. The address information consists of the row and column values, each row and column value is represented with a 12-bit vector, while the data represents the 8-bit grayscale pixel intensity.

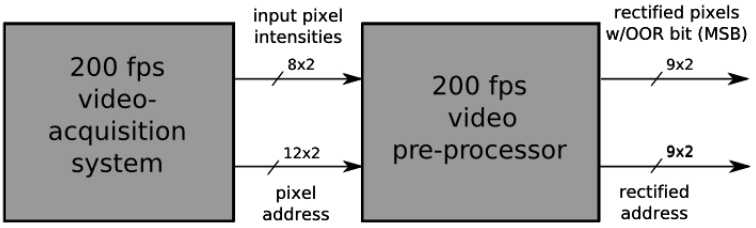


Figure 3.23: Interfacing of the video-acquisition system and video pre-processor

The video pre-processor output was verified using the SRAM Frame Download feature available on the FastTrack Image Grabber GUI. This allowed us to compare the left and right rectified images side by side, to verify corresponding points (features) appear on the same scanlines. Furthermore, another method for verifying the VPP output, although in a naïve manner, was by displaying the rectified images using the VGA Output feature found on the SFGv2.

An issue of concern was noticed during this stage, as the output of the rectified right image was skewed vertically. The reason this occurred was related to the locations of the I/O pads for the components used in the system thus far. As the system required computing at a high performance, long delay paths between components did not meet the system timing requirements of an 8 ns clock period. The issue was resolved by adding global timing constraints to the system design, which places additional restrictions to the design tool for placing routes. The addition of the constraint minimized the delay paths and routes, at the cost of longer mapping and place-and-route times. Figure 3.24 shows a comparison of the rectified right image with and without global timing constraints. A difference in the scene is noticed as the images were captured at different times.

(a) global timing constraints (b) no timing constraints

Figure 3.24: Comparison of rectified right image with timing constraints

3.6.2 Software Integration

This stage in the integration process involved utilizing the Stereo Extraction IP-core provided by the Vision & Image Dynamics Lab from the University of Toronto. The objective here was to utilize the existing hardware interface between the video-acquisition system and video pre-processor to further expand the system by integrating the Stereo Extraction Module (SEM). This allows the system to be capable of capturing stereo images, transforming the captured images to have points (features) placed on the same scanline and extracting disparity information.

The modifications made to the SEM allows it to process information at higher frame-rates. It should be mentioned that I assisted S. Sabihuddin to make the modifications. Initially, the maximum achievable clock frequency of the system was 50 MHz and resulted in a performance of 39 fps. After the modifications were made to the SEM, the maximum achievable clock frequency increased to 83.3 MHz and the performance improved to 123 fps. A discussion of the changes made to improve the performance of the SEM will be talked about in Section 3.6.3.

To allow for a simple and robust integration of the system, a reduction was made to the system clock frequency. A 62.5 MHz clock frequency was applied to the system, as it would be difficult to generate and operate the SFGv2 with a 83.3 MHz supply. The reduced clock frequency was guaranteed to allow correct functionality of all components in the system as global timing constraints were incorporated into this design. The implementation of the SEM allows processing different ranges of disparity. The disparity ranges generated by the SEM fall between 16 and 128 pixels. This is set via parameters in the design using VHDL code. The image resolution is another parameter that can be adjusted.

The integration of the SEM with the existing VPP system required an interface to provide temporary storage of a common scanline for the left and right rectified pixels from the VPP output. The temporary storage is needed as the SEM processes one scanline at a time and accesses the data much faster than the VPP produces it. Having this type of temporary memory allows each module to read and write data as they desire without any type of access hazard. The interface, known as the Stereo Controller, contains two internal BRAMs. It operates as follows: for any scanline, one of the BRAMs is set to be in read mode, which allows the SEM access to left and right view pixel information. While the other BRAM is set to be in write mode, this allows new rectified pixels for the left and right views to be saved. As the scanline finishes, the two BRAMs switch their modes, so that pixels previously used by the SEM are replaced with more recent data produced by the VPP. This double buffering approach occurs every scanline thereafter. The Stereo Controller has a dual purpose in the

interface, aside from providing temporary storage for pixels, it also provides control signals which are necessary to read computed disparity values from the SEM's disparity buffer. The control signals provide the disparity read assertion (*read*) signal and the disparity read address (*xaddr*) to the SEM. The symbols of the Stereo Controller and SEM are shown in Figure 3.25. A timeout signal was incorporated into the design of the SEM in order to synchronize the operation of the video pre-processor with the Stereo Extraction Module.

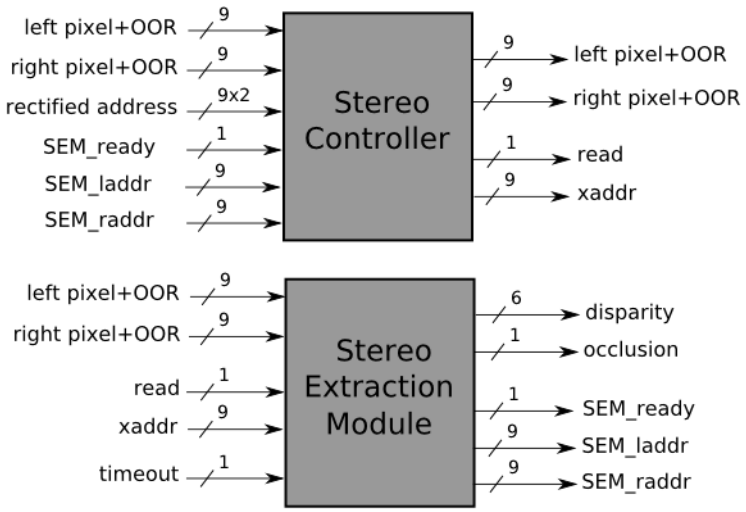


Figure 3.25: Symbols of the Stereo Controller and Stereo Extraction Module

This was required since the computation time for the SEM is based on an optimal matching path taken for each scanline. The time taken to traverse this path is variable. The timeout is designed to occur at periodic intervals, which interrupts the operation of the SEM if it has not yet finished processing.

The system was first tested by removing the video-acquisition system from the design and loading the memory components in the Stereo Controller with static scanline pairs, from a known test set, to mimic a stereo-pair from the camera. This allowed us to verify the operation of the Stereo Controller's design as it interacted with the SEM. This test also allowed us to verify the operation of the SEM with the data set used, by comparing the disparity output versus a hardware simulation for the same data set. The data used as the static scanline is a standard data set (from Tsukuba University) used for disparity algorithm

testing by Scharstein *et al.* [2]. Following this, the video-acquisition system was inserted into the design and verified for correct functionality with the SRAM Frame download and VGA Output tools. The block diagram of the system is shown in Figure 3.26.

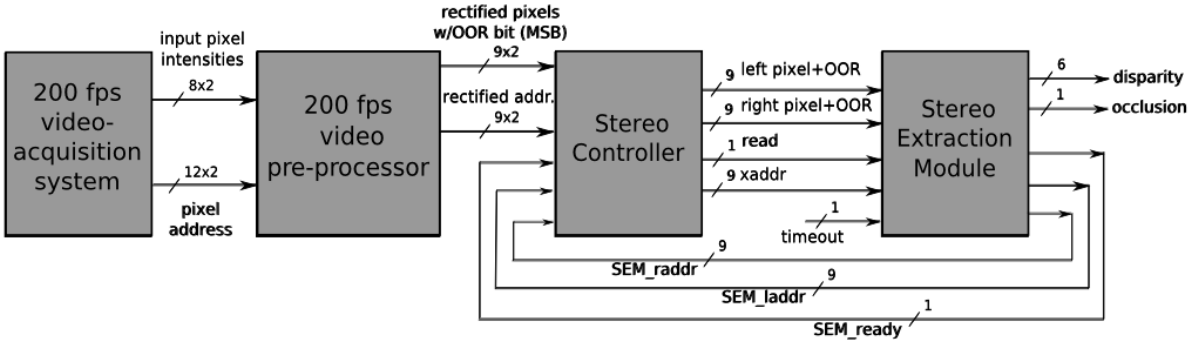


Figure 3.26: Block diagram of integrated stereo vision system

During the integration phase of the SEM a few obstacles were noticed. First, an issue arose in the displaying of the disparity information to the VGA output. The image resulting on the display appeared as though the disparity information was overlapped with the background image of the scene, see Figure 3.27. It was quickly realized that this occurred due to a problem with the timing of the SRAM module. The SEM produces a new disparity value every clock cycle (*i.e.*, 1 c.c./data), the SRAM on the other hand writes information every two clock cycles (*i.e.*, 2 c.c./data). This problem was resolved by doubling the clock frequency to the SRAM controller and SRAM chip (*i.e.*, 125 MHz). The resulting change allowed the disparity information to show as seen in Figure 3.28. Another problem which was relatively simple, but caused a great deal of headaches, occurred in the output of the SRAM Frame download. The information represented in the image did not correctly represent the disparity information produced in the hardware simulation. The problem was resolved by scaling the disparity values between 0 - 255 (*i.e.*, an 8-bit intensity). The value of the scaling factor depends on the maximum disparity, D_{max}. For example, the scaling factors for $D_{max} = 16$ and $D_{max} = 32$ would be 16 and 8, respectively.

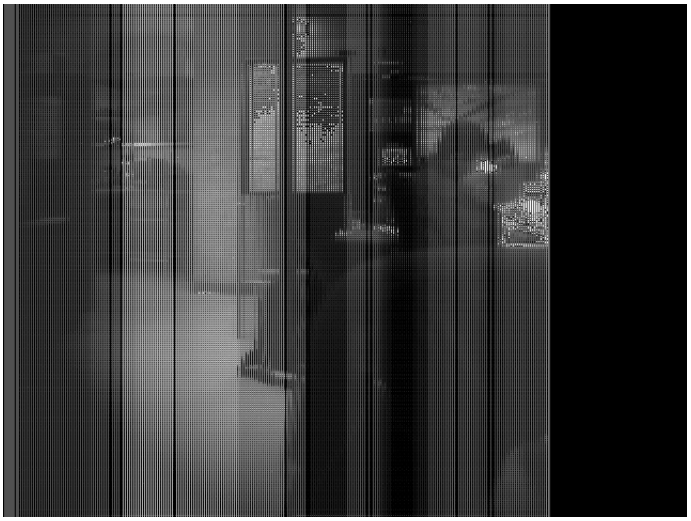


Figure 3.27: Hardware download of disparity meshed with the scene

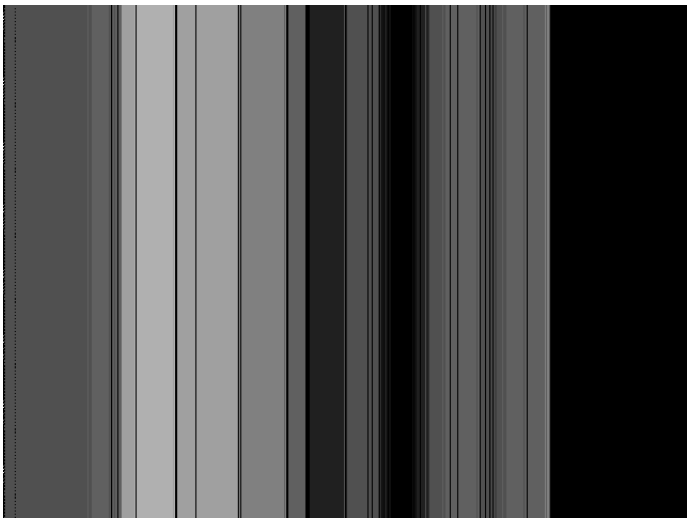


Figure 3.28: Hardware download of correct disparity information

3.6.3 Stereo Extraction Module Modifications

The development of the Stereo Extraction Module was done in an iterative process in a liaison between research groups at the University of Toronto and Ryerson University. The initial development phase of the system achieved frame-rates of approximately 39 fps on a 50MHz clock frequency. However, to meet the goals of this system, changes to the architecture of the SEM were necessary to allow processing of higher frame-rates at increased clock frequencies.

A number of problems were noticed in the initial design of the SEM. A major problem in the design was the large amount of logic resources utilized to structure the match matrix memory. The memory used for the initial design consisted of distributed RAM, which is composed of the logical blocks on the FPGA. The Virtex-II Pro FPGA, however, comes equipped with a large volume of customized memory blocks (*i.e.*, BRAMs) for such purposes. Thus, replacing the match matrix memory with BRAMs would make valuable resources on the chip free for other purposes. Furthermore, this change decreased the implementation time for the design into a bitstream file. On the downside, changes to the memory structure required modifications to be made to the hardware used for determining the optimal path. The initial implementation accessed the match matrix memory with asynchronous reads. With the replacement of the distributed RAM to the BRAM, asynchronous reads were no longer possible, as BRAMs are fully synchronous memory devices. The solution to this problem is described in the following section. Another problem was that the operational frequency of the SEM was limited by longest combinational logic path. This path was used for the computation of the non-occlusion cost and selection of the minimum cost value. This problem was solved by introducing pipelining, which inserts registers in between the combinational logic and allows using a higher clock frequency. Additional speedups to the SEM could be achieved by interleaving the forward-pass and backward-pass phases of the SEM. These modifications also resulted in changes to the state machine of the SEM. A final modification was required to the SEM to allow the data provided by the video pre-processor to effect the outcome of the cost computation (*i.e.*, if the pixel was marked Out-of-Range). These problems and modifications will be discussed in more detail in the following sections.

3.6.3.1 Backward-pass Hardware Re-design

As the match matrix memory has been replaced with BRAMs to save valuable logic resources, the issue of reading memory synchronously required design changes. The prior design, which used distributed RAM, could access the match matrix memory (MBUF) dur-

ing the backward-pass phase at will (asynchronously). The new synchronous design requires a one clock cycle delay to retrieve data from the match matrix memory. To overcome this problem, additional states were added to the state machine in order to pre-fetch data from the match matrix memory. More details on the implementation of the pre-fetch hardware can be found in [16].

3.6.3.2 Pipelining of the Forward-pass

In order to reduce the combinational delay within the longest chain of logic, pipelining was used to register data within the chain. The longest path computes the non-occlusion cost and selects the minimum cost value corresponding to the pixels compared. The process of pipelining allows increasing the throughput of a system, similar to an automobile assembly line with a small initial delay to fill the pipe. A logical register, PBUF, was inserted into the forward-pass of the SEM to disassemble the long chain of logic into two synchronous blocks. The PBUF component contains registers to temporarily buffer data and control signals, which normally would be passed through each block simultaneously. Figure 3.29 shows the modified forward-pass of the SEM. The addition of the PBUF component was instrumental in improving the performance and increasing the operational frequency of the SEM at the cost of some logic resources for its implementation. The forward-pass was tested using ModelSim XE to verify that the cost values computed in the pipelined SEM matched the cost values computed for the un-pipelined SEM using the same data set. For further information about the design of the SEM please refer to [16].

3.6.3.3 Interleaving the SEM

Achieving high performance is often related to the amount of time saved to perform tasks in an application. Further improving the performance of the SEM was made possible by interleaving between the forward- and backward-pass phases of the SEM. The initial design performed the forward-pass, which writes data into the cost and match matrix memories,

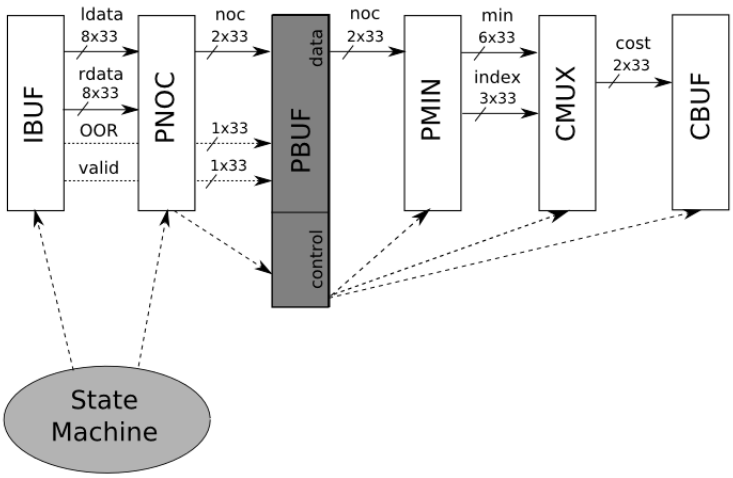


Figure 3.29: Pipelined forward-pass hardware

followed by the backward-pass which only reads data from the match memory. During the operation of the backward-pass, the hardware implemented for the forward-pass was idle and unused. Modifying the design of the SEM to have the forward- and backward-pass operating at the same time achieved the goal of interleaving. This modification required duplicating the match matrix memory, MBUF for the purpose of double buffering. The backward-pass reads previous match memory data, while the forward-pass writes current match data. An additional multiplexer is needed to alternate between the two match matrix memories to allow proper interleaving. Figure A.5 (see Appendix A) shows the schematic of the SEM after the changes for interleaving were made.

3.6.3.4 Modifications to the SEM State Machine

The modifications made to the Stereo Extraction Module for incorporating all the changes described previously also effected the design of its state machine. An illustration of the SEM state machine is shown in Figure 3.30. It should be mentioned that the circles within each state are shaded differently to distinguish between the forward- and backward-pass phases. The circles with a lighter shade represent the forward-pass, while the darker shade represents the backward-pass.

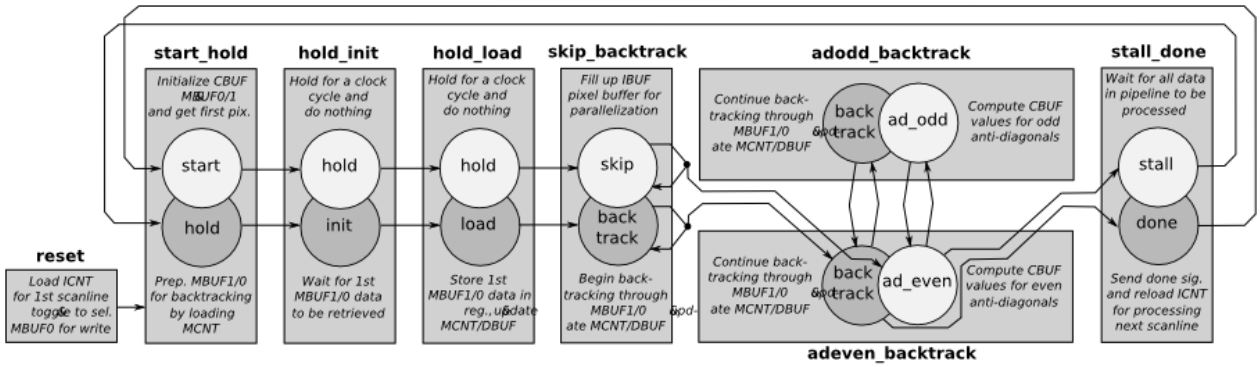


Figure 3.30: State machine of the Stereo Extraction Module

3.6.3.5 Modification to PMIN Component for OOR

As described in Section 3.3.5, the Out-of-Range flag is set high when the warped pixel is invalid (*i.e.*, it does not exist in the standard 640×480 image). Invalid pixels do not represent any meaningful information within the scene and thus should be set to a maximum cost when compared with pixels in the other view. To realize this in the SEM, one extra multiplexer was added to the PMIN component, which sets the cost for matching to an Out-of-Range pixel to be extremely high, thus having no correspondence.

3.7 Summary

In this chapter, we began by introducing the architecture of the video pre-processor, which places points (features) on the same scanlines using warping transformations. The homography matrices used to transform the original images were converted into 1st-order Taylor Series approximations, thus avoiding performing divisions, which are expensive to realize on FPGAs. Performing the warping transformations required storing portions of the original images in temporary buffers. The 200 fps version of the video pre-processor utilized dual-ported BRAMs for this temporary memory. This allows saving multiple scanlines into each BRAM, which increases the efficiency of the design compared to the 30 fps version. Following this, the 30 fps and 200 fps versions of the video-acquisition system were explored. The

200 fps video-acquisition system was designed to make debugging and verification within the system easy. For example, it included two VGA output ports for visualizing the captured or rectified images. It also included a hardware image download feature, which allowed analyzing the processed data in the form of a bitmap image. This was useful for many reasons, one important reason being analyzing the error between the hardware produced output and a known, ground truth, output. Next, we explored the features of the Amirix AP1100 Platform, which provides the basis for our implementation.

The integration of the complete system involved the video-acquisition system, the video pre-processor and the Stereo Extraction Module. Ideally, two complete versions of the system were to be developed. The first system was to be a low performance, 30 fps stereo vision system. The second system a high performance, 200 fps stereo vision system. The initial version presented many subtle issues that made it difficult to integrate and test the functionality of the system. The high performance, 200 fps stereo vision system was integrated using the high throughput video-acquisition system (SFGv2), video pre-processor (SRM) and Stereo Extraction Module (SEM).

Key modifications to the Stereo Extraction Module were then presented. A significant change in the memory structure of the match matrix allowed a vast amount of logic resources to be saved. The result of replacing the memory structure required modifying the hardware used in the backward-pass, to synchronously read the match memory and compute the correct disparity value. A prefetching system was developed and implemented to assist with the change to BRAMs. Another significant change made by pipelining the forward-pass of the design allows a higher clock frequency to be used with the system. An increase of the clock frequency itself then allows the performance of the system to increase. Finally, an additional step was used to further increase the performance of the design: by interleaving the forward- and backward-pass phases of the design, useful time is saved and this allows a higher throughput to be achieved. The modifications made to the Stereo Extraction Module were vital to meeting the performance goals of the complete system.

Chapter 4

Evaluation of Results

4.1 Introduction

In this chapter we investigate our design based on a set of metrics which include quality of output, resource utilization, clock frequency and power consumption. In Section 4.2, we evaluate the performance of the 30 fps and 200 fps Stereo Rectification Modules. Following that we evaluate the different implementations of the Stereo Extraction Module in Section 4.3. In Section 4.4, we evaluate the performance and resources utilized for the integration of the high frame-rate stereo vision system. Finally, Section 4.5 summarizes the chapter's contents.

4.2 Stereo Rectification Module

The Stereo Rectification Module was designed and implemented in order to transform the images acquired from the video-acquisition system so that each point (feature) is placed on the same scanline in the two views. The following sections evaluate the performance of the 30 fps and 200 fps implementations of the video pre-processor. The target device used for realizing the SRM was the Xilinx Virtex-II Pro XC2VP100 FPGA.

The video pre-processor is analyzed in a number of different categories. Important aspects

for the design of this module include resource utilization, performance, power consumption, precision of fixed point operators and the quality of the produced output. Each aspect is further analyzed in each of the following sections.

4.2.1 Resource Utilization

The resources of the Xilinx FPGA are broken down into slices, 4-input LUTs, PowerPCs, block RAMs, multipliers (MULT18×18) and global clocks. Furthermore, the 4-input LUTs can be divided as logic, route-through and shift registers. Table 4.1 lists the resources utilized for the initial SRM version interfaced with the 30 fps video-acquisition system. Table 4.2 lists the resources utilized for the second SRM version interfaced with the 200 fps video-acquisition system. It should be mentioned that the numbers are obtained from the Xilinx ISE Project Navigator design summary. It can be seen from Tables 4.1 and 4.2, that significant resource

Table 4.1: Resource utilization for the 30 fps SRM

Resource	**Used**	**Available**	**Percentage**
Slices	4,985	44,096	11%
4-input LUTs	6,835	88,192	7%
BRAMs	24	444	5%
Multipliers	96	444	21%
Global Clocks	3	16	18%

Table 4.2: Resource utilization for the 200 fps SRM

Resource	**Used**	**Available**	**Percentage**
Slices	1,845	44,096	4%
4-input LUTs	2,030	88,192	2%
BRAMs	20	444	4%
Multipliers	23	444	5%
Global Clocks	2	16	12%

savings have been attained with the 200 fps SRM design. The 200 fps SRM design was made more efficient than the 30 fps SRM design by utilizing dual-ported block RAMs. A comparison of the block RAM utilization between the two designs shows as very similar.

However, the 200 fps SRM was designed to buffer more scanlines before processing began than the 30 fps SRM. This was due to the camera parameters which measured the relative placement of the image sensors and lens housing. If both video-acquisition systems had the same camera parameters, the design of the 30 fps SRM would have utilized many more block RAM instances than the 200 fps version.

Generally, the amount of resources used to implement the SRM is quite reasonable. The most expensive hardware blocks found on the FPGA are the slice logic, multipliers and block RAMs. These three resources are used to under 10% of their total capacity in the 200 fps SRM design. This is economical and a good path for integrating the other components of the system as mentioned in Chapter 3.

4.2.2 Performance

Each version of the Stereo Rectification Module was designed based on the characteristics of the video-acquisition system. The 30 fps SRM design was fixed to utilize a 62.5 MHz clock frequency. This performance of the SRM is determined by the formula in Equation 4.1, where P, N, M, R, L, and F_{clk} represent the performance [fps], number of pixels per scanline $\left[\frac{pixels}{scanline}\right]$, number of scanlines [$scanlines$], pixel time $\left[\frac{clock\,cycles}{pixel}\right]$, latency [$clock\,cycles$] and operational frequency [MHz], respectively. The 30 fps SRM uses a 992×480 image, 4 clock cycles/pixel, 17 clock cycle latency, and 62.5 MHz system clock which provides a performance of 32.8 fps.

$$P = \left[\frac{(NMR + L)}{F_{\text{clk}}}\right]^{-1} \tag{4.1}$$

The 200 fps video-acquisition system was designed to utilize a 125 MHz clock signal. In order to keep up with the high frame-rate data capture, the 200 fps SRM was also designed to use this clock signal. The performance of this version of the video pre-processor was also computed using Equation 4.1. The 200 fps SRM uses a 640×480 image, 2 clock

cycles/pixel, 17 clock cycle latency, and 125 MHz system clock. An advantage of the high frame-rate video-acquisition system allows the performance to be changed via system clock parameters. Table 4.3 shows the performance of the SRM using the different system clock frequencies.

Table 4.3: Performances of the high frame-rate SRM

System Clock (MHz)	**Performance (fps)**
125	203.4
62.5	101.7
31.25	50.8
15.625	25.4

4.2.3 Power Consumption

The power consumed by FPGAs can be broken down into two components, dynamic and static (quiescent) power. The dynamic power consumption of a digital design is a function related to the supply voltage, capacitance and switching activity [52]. For FPGAs, the power consumption is also dependent on the amount of time the resource is being utilized. Equation 4.2 shows the FPGA power as a function of capacitance C, supply voltage V_{DD}, resource utilization U, and switching activity f. Now for each architectural resource i, the total switch capacitance is the product of the effective capacitance C_{eff-i}, the number of instances utilized in the design U_i, and the average switching frequency across all instances f_i. Unused elements found on the FPGA have a base rate of power consumption identified by the static power.

$$P = V_{DD}^2 \sum_{i=0}^{N} C_{eff-i} U_i f_i \tag{4.2}$$

The power consumed by the Stereo Rectification Module was estimated using XPower, which is a utility available with the Xilinx ISE package. XPower uses gate level simulations of the design to estimate the average switching activity of the resources and power consumption

over time. The simulations allow XPower to estimate the dynamic power consumption.

The Stereo Rectification Module operating at 30 fps utilizes the resources seen in Table 4.1 and utilizes the power shown in Table 4.4. This same table indicates the amount of power consumed by the 200 fps Stereo Rectification Module, and its resources are shown in Table 4.2. The efficiency in the design of the later SRM utilized less resources on the FPGA and also played a major factor in decreasing the power consumption. It can be seen from Table 4.4 that the later SRM required less power in all three categories.

Table 4.4: Power consumption of the SRM

	Total Power	**Dynamic Power**	**Static Power**
30 fps SRM	463.16 mW	258.79 mW	204.38 mW
200 fps SRM	432.94 mW	232.07 mW	200.87 mW

4.2.4 Fixed-point Precision

The use of fixed-point operators versus floating-point operators was chosen due to the fact that they perform arithmetic operations faster and consume less area for their implementations. The precision of the fixed-point operator is based on the number of bits used for the fractional portion of the operator. For our implementation of the fixed point operator, we required 1-bit for the sign, 10-bits for the magnitude and n-bits for the fraction. The value of n is decided by evaluating the amount of resources required and precision of the operator. Since the operations of addition and multiplication use dedicated arithmetic blocks on the FPGA, the total size of the fixed-point number would have an impact on the amount of resources (area) required for its implementation. Table 4.5 shows the resources required for the implementation of one fixed-point multiplier and obtained using Xilinx Project Navigator. The resources for each x-bit operator, where $x = 1 + 10 + n$, are broken down into MULT18×18s, slices, slice flip-flops (FFs) and 4-input LUTs. The table also indicates the precision of the x-bit fixed-point multiplier with its associated percentage error. The error computed indicates the amount of error for the fractional portion of the fixed-point value.

The Figure 4.1 shows a plot of the precision *vs.* area for the fixed-point operators. The plot shows the comparison of the precision with area for the multiplication operation, as the accuracy of this operation is more vital than for addition.

Table 4.5: Resources used by x-bit fixed-point multipliers

x	**MULT18×18s**	**Slices**	**Slice FFs**	**4-input LUTs**	**% Error**
32	4	137	160	120	0.01
30	4	126	150	108	0.01
28	4	114	140	95	0.01
26	4	103	130	81	0.01
24	4	91	120	67	0.01
22	3	79	110	54	0.01
20	3	69	100	44	0.13
18	1	52	90	18	0.16
16	1	46	80	16	1.91
14	1	40	70	14	11.4
12	1	34	60	12	50.0

In Figure 4.1, the precision of the fixed-point operator is indicated by the dashed line, while the resources utilized is indicated by the solid line. The figure shows that as the number of bits for the fixed-point operator increases, the precision of the operator also increases (*i.e.*, less error). However, when the operator exceeds a certain amount of bits it can be noticed that the precision reaches a threshold and begins to stabilize. Thus, we chose 18-bit fixed-point operators for our implementation of adders and multipliers, as they provided computations with reasonably low errors and very low consumption of logical resources. It should be pointed out that a "spike" appears in the plot of the resource utilization, this occurs due to the jump from 1 MULT18×18 to 3 MULT18×18s which requires additional glue logic.

4.2.5 Quality: Hardware *vs.* Software

The verification of the Stereo Rectification Module operating at 30 fps was difficult. The system did not have the means to download the produced output into a readable manner. One naïve way to verify the output of the 30 fps SRM was by displaying the images onto

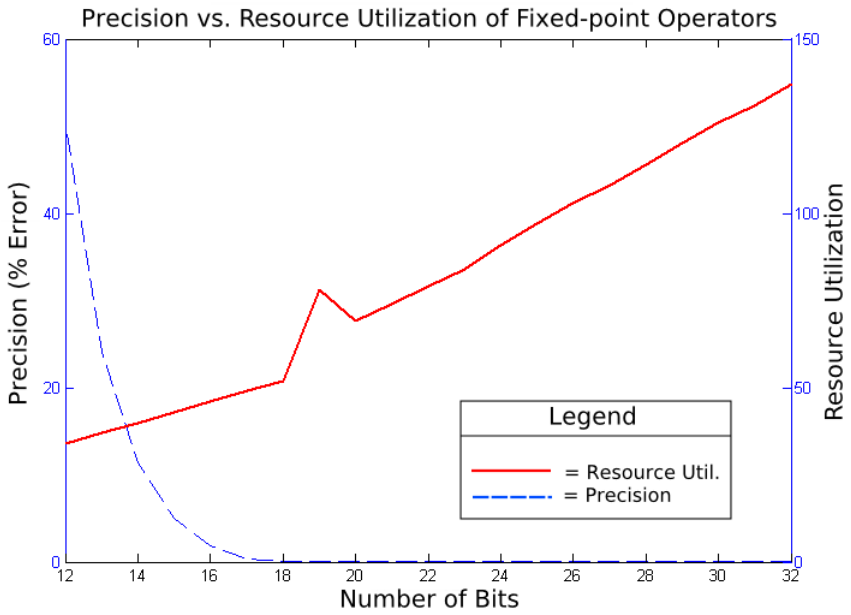


Figure 4.1: Plot of Precision vs. Area for the fixed-point operator

the VGA monitor. We were able to visually inspect that corresponding points from the two views were placed on the same scanline using a switch on the VGA output module. The information displayed on the VGA monitor was only capable of sending 2-bits of information for each colour channel due to the amount of I/O pads available from the Amirix Daughter Board. Normally, 8-bits of information are displayed for each channel. This accounted for some degradation in the images displayed. Figure 4.2 shows the left and right view rectification output as seen on the VGA monitor. It can be noticed that a green bar appears to be going down the left side of the each image. The green bar represents pixels which were computed to be Out-of-Range.

The design of the 200 fps video-acquisition system made verifying the operation of the high frame-rate Stereo Rectification Module much more convenient. The hardware download and visualization features developed for the video-acquisition system allowed the two rectified images to be saved as bitmap files and viewed simultaneously on two VGA monitors. Figure 4.3 shows the raw (original) images captured by the 200 fps video-acquisition system. By visually inspecting these images, one can notice that points in left image are slightly higher

(a) left view (b) right view

Figure 4.2: 30 fps SRM outputs displayed on a VGA monitor

than in the right image, while the artifacts that appear are due to the Bayer pattern and high regions of saturation. Figure 4.4 shows the left and right view rectification outputs produced by the high frame-rate SRM. The images produced by the high frame-rate SRM produces, and displays, a higher quality image than the previous version (*i.e.*, 30 fps SRM). The monochrome information uses 8-bits to represent 256 different shades. A comparison of the rectified images produced by the hardware can be compared pixel by pixel against the software simulation. Figure 4.5 shows the left and right view rectification outputs produced in software.

An analysis of the hardware and software implementations for stereo image rectification can be made using intensity histograms. Intensity histograms are used to tabulate and display the frequency of occurrence for each level of brightness. They are also used to determine if certain images can be enhanced, compressed or segmented. On the other hand, error histograms show the error distribution of the hardware output as compared to a reference implementation in software. The error histograms can be described as follows: $E(x,y) = P_{rect}^{FPGA}(x,y) - P_{rect}^{SW}(x,y)$, where $E(x,y)$ is the error, $P_{rect}^{FPGA}(x,y)$ and $P_{rect}^{SW}(x,y)$ represent the pixel intensity of the rectified hardware (FPGA) and software images, respectively, at a particular pixel location (x,y). Figure 4.6a shows the error histogram for the left rectified image of Figures 4.4a and 4.5a. Similarly, Figure 4.6b shows the intensity histogram for the right rectified image of Figures 4.4b and 4.5b. The cause of the errors are due to

boundary effects. The boundary effects are caused by the

(a) left view (b) right view

Figure 4.3: Original images captured by the Stereo Frame Grabber v.2

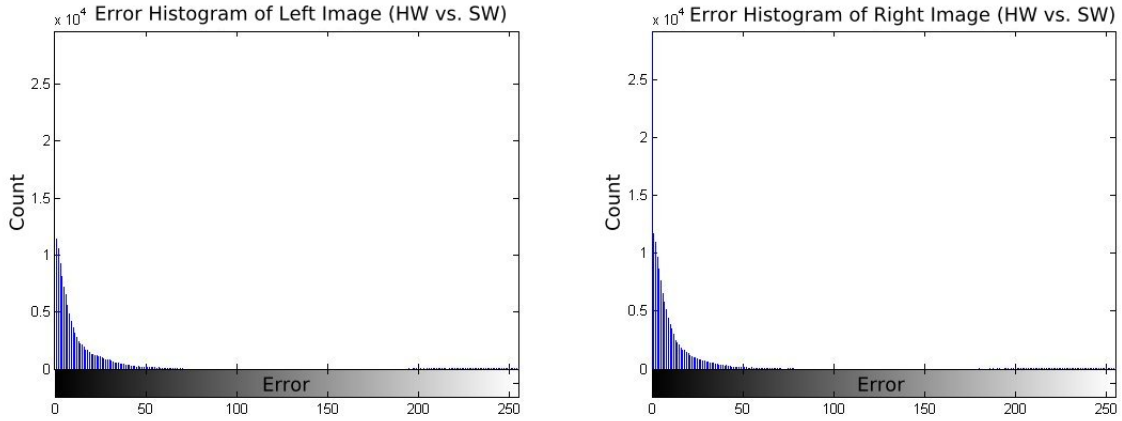
(a) left view (b) right view

Figure 4.4: Outputs of the 200 fps Stereo Rectification Module

(a) left view (b) right view

Figure 4.5: Outputs generated by software

limited precision of the fixed-point operators in the calculations of the Image Warp and Bilinear Interpolation modules. The root mean squared error (RMSE) of the error histograms are computed using Equation 4.3 and the results are shown in Table 4.6.



(a) left image

(b) right image

Figure 4.6: Error histograms

$$RMSE = \sqrt{\sum_{(x,y)}^{N} \frac{E^2(x,y)}{N}} \tag{4.3}$$

Table 4.6: RMSE of the error histograms

Error Histogram	**RMSE**
Left	6.8381
Right	7.5596

4.3 Stereo Extraction Module

The Stereo Extraction Module is designed to obtain depth information from the scene in view. It uses the data provided by the Stereo Rectification Module as input to perform a 1-D search for pixel correspondence. The Stereo Extraction Module's performance is measured in a number of different categories. Important metrics for this module include performance

in frames per second, which is further analyzed in Section 4.3.1, resource utilization discussed in Section 4.3.2 and the quality of the output, found in Section 4.3.3. The target device used for realizing this module is the Xilinx Virtex-II Pro XC2VP100 FPGA.

4.3.1 Performance

The initial implementation of the Stereo Extraction Module provided low-speed performance. The throughput of this implementation was calculated with Equation 4.4. The total performance achieved using $D_{max} = 128$ pixels, an image resolution of 640×480 and $F_{clk} = 50$ MHz, produced a frame rate of 39.38 fps [17, 42]. This performance was too slow to meet the end requirements of the system, thus modifications were carried out to improve the performance. The changes to the initial implementation consisted of pipelining and interleaving.

Pipelining the Stereo Extraction Module allowed breaking down the longest combinational path for the computation of the non-occlusion cost and selecting the minimum cost into smaller blocks. This resulted in the operational frequency, F_{clk}, increasing to 83.3 MHz. Furthermore, with the same maximum disparity and image resolution, the frame rate increased to 65.60 fps.

$$FPS = \left[(4N + \frac{D_{max}}{2} - 1) \frac{M}{F_{clk}} \right]^{-1} \tag{4.4}$$

A further modification was made to allow the interleaving of the forward- and backward-pass phases of the SEM. This allowed additional boosts to the module's performance. The performance of the interleaved SEM is measured using Equation 4.5. The total performance achieved using $D_{max} = 128$ pixels, 640×480 image and F_{clk}= 83.3 MHz, produced a frame rate of 124.49 fps. Table 4.7 shows the performance of the interleaved SEM using 640×480 images at different disparity levels.

$$FPS = \left[\frac{2N + (2N + \frac{D_{max}}{2} - 1)M}{F_{clk}} \right]^{-1} \tag{4.5}$$

Table 4.7: SEM performance with varying disparity levels on 640 × 480 images

Pixels of Disparity	Performance (fps)
16	134.39
32	133.29
64	131.14
128	124.49

It it worth mentioning that the performance of the SEM can be further improved given an FPGA device with more block RAM resources. The performance increase would be achieved, hypothetically, by duplicating the current hardware so that two parallel SEMs process subsequent scanlines. This would enable us to meet and exceed the goal of computing disparity maps at 200 fps.

4.3.2 Resource Utilization

In this section, the amount of logic resources used for the different implementations of the Stereo Extraction Module will be investigated. First, a comparison of the resources utilized for the distributed RAM and block RAM versions of the initial 39 fps SEM implementation will be analyzed. Following this, the interleaved SEM will be investigated for images of 640 × 480 resolution, at disparity ranges of 16, 32, 64 and 128 pixels.

4.3.2.1 Distributed RAM vs. Block RAM Implementation

In order to develop a complete system consisting of a chain of modules, it is absolutely necessary that the design of each module is efficient with respect to resources needed to implement them. Two implementations of the initial Stereo Extraction Module processing D_{max} = 128 pixels on 640 × 480 images are evaluated. The first implementation uses distributed RAM (composed of logic resources on the FPGA) to create the match matrix buffer (MBUF), while the second implementation uses dedicated memory resources on the FPGA, known as block RAM, to create MBUF. Table 4.8 shows the amount of slices, 4-input LUTs, BRAMs and MULT18×18s required for each implementation, obtained from Xilinx Project

Navigator. From the table it can be seen that the number of slices and 4-input LUTs used in the BRAM implementation decreases by approximately 20% compared to the distributed RAM version. On the other hand, the amount of BRAMs needed increases by 60%. This increase in the BRAM resource is acceptable, as it reduces the complexity of the routing and also allows a small increase in the operating frequency of the module.

Table 4.8: Resources utilized by the distributed and block RAM implementations (D_{max} = 128) of the 39 fps SEM

	Slices	**4-input LUTs**	**BRAMs**	**MULT**18×18**s**
Distributed RAM	22047 *(49%)*	44049 *(49%)*	0	65 *(14%)*
Block RAM	12956 *(29%)*	25911 *(29%)*	269 *(60%)*	65 *(14%)*

4.3.2.2 Interleaved Implementation

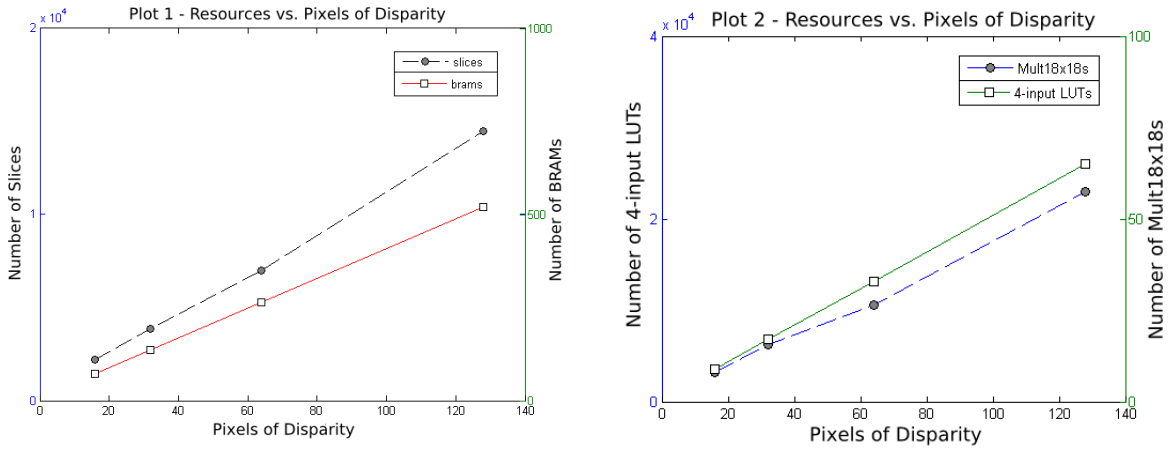
Table 4.9 shows the amount of resources required for each implementation of the Stereo Extraction Module. The resources are categorized into slices, 4-input LUTs, BRAMs and MULT18×18s and obtained via Xilinx Project Navigator. The Stereo Extraction Module operates on 640 × 480 resolution images to produce depth estimates by first estimating disparity. From Table 4.9 it is obvious that as the Stereo Extraction Module processes higher disparity ranges, the amount of resources needed for the implementation on the Xilinx XC2VP100 also increases. Looking at the D_{max} = 128 pixel version, it can be noticed that the amount of BRAMs is over mapped. Thus, it would be impossible to have a working implementation on the current FPGA. Having a larger FPGA with larger BRAM capacities would allow disparities of 128 pixels to be achieved. Plots of the data presented in Table 4.9 are shown in Figure 4.7. The plots show that the amount of resources required increases roughly linearly with the disparity range.

4.3.3 Quality: Hardware *vs.* Software

In this section, the quality of the results produced by the Stereo Extraction Module will be evaluated. First, the simulated hardware implementations of the distributed and block RAM

Table 4.9: Interleaved SEM resource utilization for varying disparity levels on 640 × 480 resolution images

Pixels of Disparity	Slices	4-input LUTs	BRAMs	MULT18×18s
16	2209 *(5%)*	3249 *(3%)*	72 *(16%)*	9 *(2%)*
32	3857 *(8%)*	6313 *(7%)*	136 *(30%)*	17 *(3%)*
64	7000 *(15%)*	10576 *(11%)*	264 *(59%)*	33 *(7%)*
128	14488 *(32%)*	23027 *(26%)*	520 *(117%)*	65 *(14%)*



(a) Resources: Slices and BRAMs

(b) Resources: 4-input LUTs and Mult18x18s

Figure 4.7: Plots of Resources *vs.* Pixels of Disparity

versions of the module will be compared to a reference implementation in software, for D_{max} = 128 pixels on 640 × 480 images. Next, the hardware implementation of the interleaved version of the Stereo Extraction Module will be compared with the software implementation. For this the hardware and software implementations will be analyzed for disparity ranges of 16, 32, 64 and 128 pixels.

4.3.3.1 Distributed RAM vs. Block RAM Implementation

Figure 4.8 shows the comparison of the outputs produced by the distributed and block RAM versions of the Stereo Extraction Module operating on 640 × 480 images at 128 pixels of disparity. The Tsukuba data set is used as the basis for this comparison, as it is widely used in the computer vision community and has ground truth disparity and occlusion maps. Our results are compared with published results based on quality metrics including RMSE

and percent bad matching pixels, metric proposed by Scharstein *et al.* [2, 53]. The results produced by the distributed RAM version are compared to the block RAM version using root mean squared error (RMSE) and percent

$$RMSE = \sqrt{\frac{1}{K}\sum_{(x,y)}^{K}(D_{est}(x,y) - D_{true}(x,y))^2} \tag{4.6}$$

bad matching pixels. The RMSE is computed using Equation 4.6, while the percentage of bad matching pixels is computed using the Heaviside step function seen in Equation 4.7. In these equations, $D_{est}(x,y)$ and $D_{true}(x,y)$ represent the estimated and true disparity values at a particular pixel location (x,y), respectively. Furthermore, K represents the number of pixels in the image.

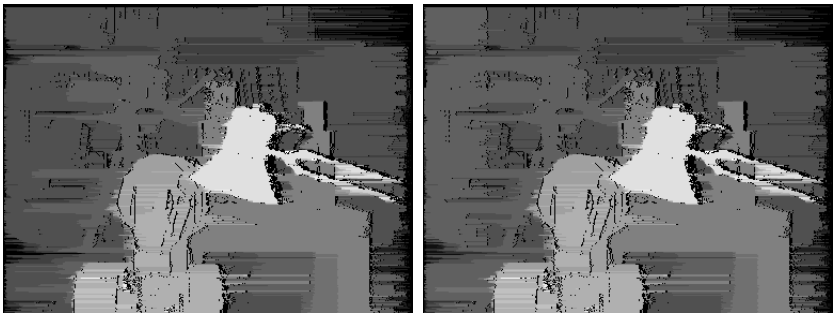
$$BAD = \frac{1}{K}\sum_{(x,y)}^{K}H(|D_{est}(x,y) - D_{true}(x,y)| - \delta_D) \tag{4.7}$$

These metrics are used to measure stereo correspondence results by the computer vision community. Table 4.10 indicates the amount of error of the block and distributed RAM outputs when compared against Figures 4.12b and 4.12d, the software generated outputs. The block RAM implementation gives slightly different results due to the inability to initialize the memory's contents on the fly. As a

Table 4.10: Accuracy of the distributed RAM and block RAM outputs

	RMS Error	**% Bad Pixel Match**
Distributed RAM	0.0183	0.0080
Block RAM	0.0255	0.0080

result, the backward-pass may initially traverse along a different minimum cost path, however, this path eventually converges to the same path as the distributed RAM implementation.



(a) distributed RAM: disparity map

(b) block RAM: disparity map

(c) distributed RAM: occlusion map

(d) block RAM: occlusion map

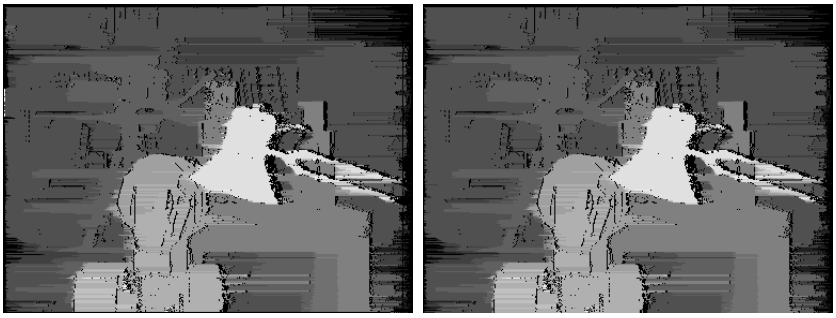
Figure 4.8: Outputs of the distributed and block RAM versions of the SEM

4.3.3.2 Interleaved Implementation: Hardware *vs.* Software

The results produced in the hardware and software implementations of the interleaved Stereo Extraction Module are compared in Figures 4.9, 4.10, 4.11, 4.12 for disparity ranges of 16, 32, 64 and 128 pixels, respectively. The Middlebury Stereo Vision Page [54] contains data sets that have different disparity ranges. The Tsukuba, Venus and Teddy data sets were used. Table 4.11 indicates the amount of error for each stereo correspondence.

Table 4.11: Accuracy of the interleaved SEM

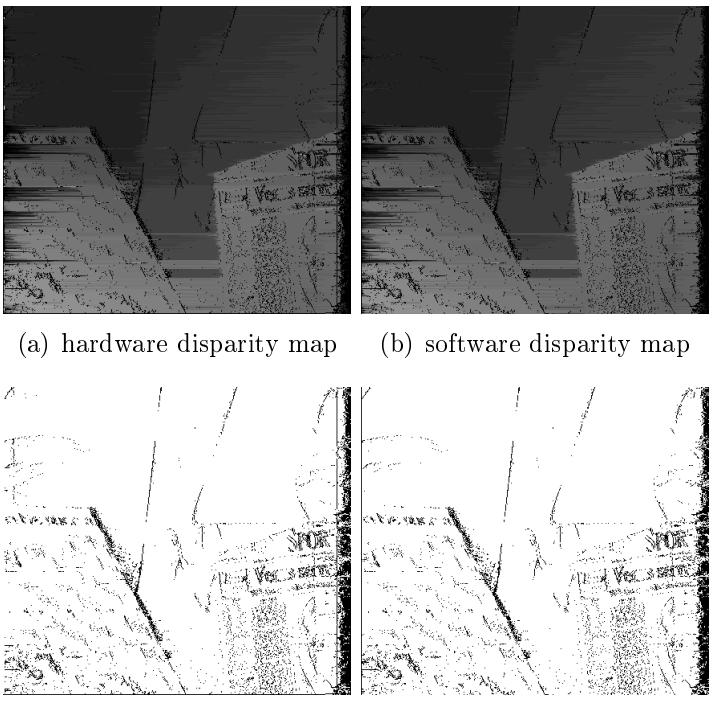
$\mathbf{D}_{max}$	**RMS Error**	**% Bad Pixel Match**
16	0.1747	0.1613
32	0.2043	0.1762
64	0.2401	0.1849
128	0.2787	0.1977



(a) hardware disparity map (b) software disparity map

(c) hardware occlusion map (d) software occlusion map

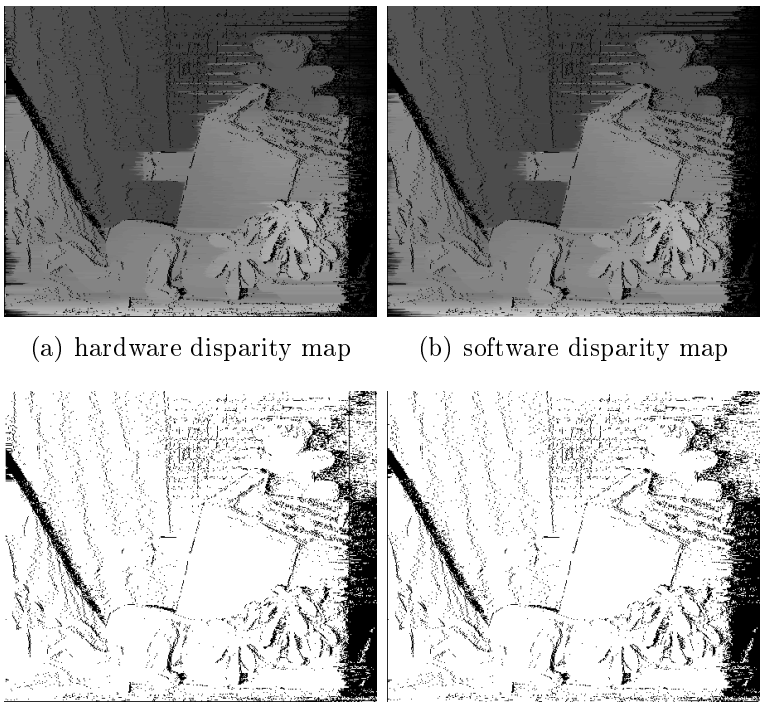
Figure 4.9: Tsukuba: results of the interleaved SEM, D_{max} = 16 pixels



(a) hardware disparity map (b) software disparity map

(c) hardware occlusion map (d) software occlusion map

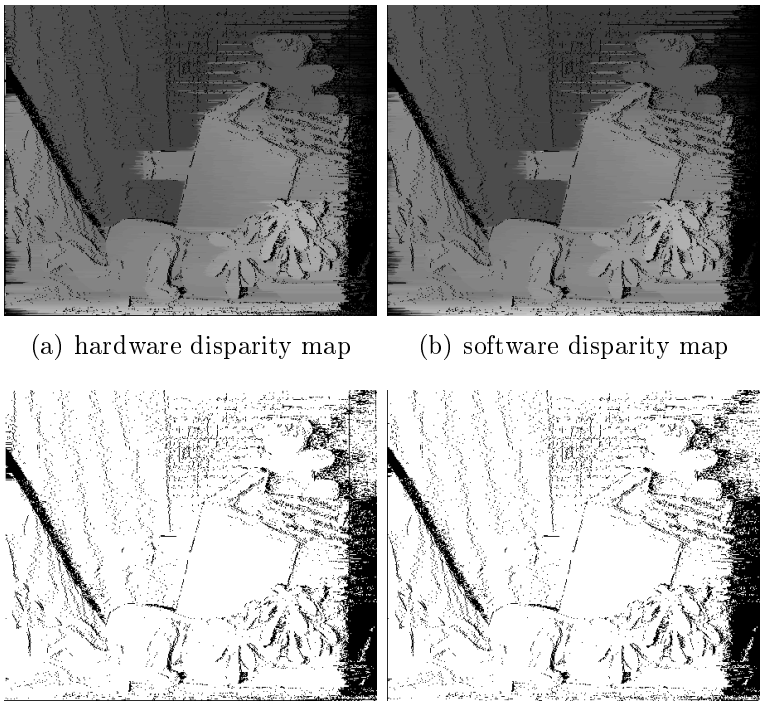
Figure 4.10: Results of the interleaved SEM, D_{max}= 32 pixels



(a) hardware disparity map (b) software disparity map

(c) hardware occlusion map (d) software occlusion map

Figure 4.11: Results of the interleaved SEM, $D_{max} = 64$ pixels



(a) hardware disparity map (b) software disparity map

(c) hardware occlusion map (d) software occlusion map

Figure 4.12: Results of the interleaved SEM, $D_{max} = 128$ pixels

4.4 System Integration

This section discusses different metrics used for evaluating the integrated stereo vision system. As mentioned in Chapter 3, to perform a simple and robust system integration it was necessary to reduce the clock frequency of the system to 62.5 MHz. The parameters of the system now consist of capturing and rectifying scene data at 100 fps, for images of 640×480 resolution. The performance of the full resolution interleaved SEM ($D_{max} = 64$ pixels) was degraded likewise due to the decrease in the system clock. It achieved a frame rate of 98.39 fps. This was computed using Equation 4.5.

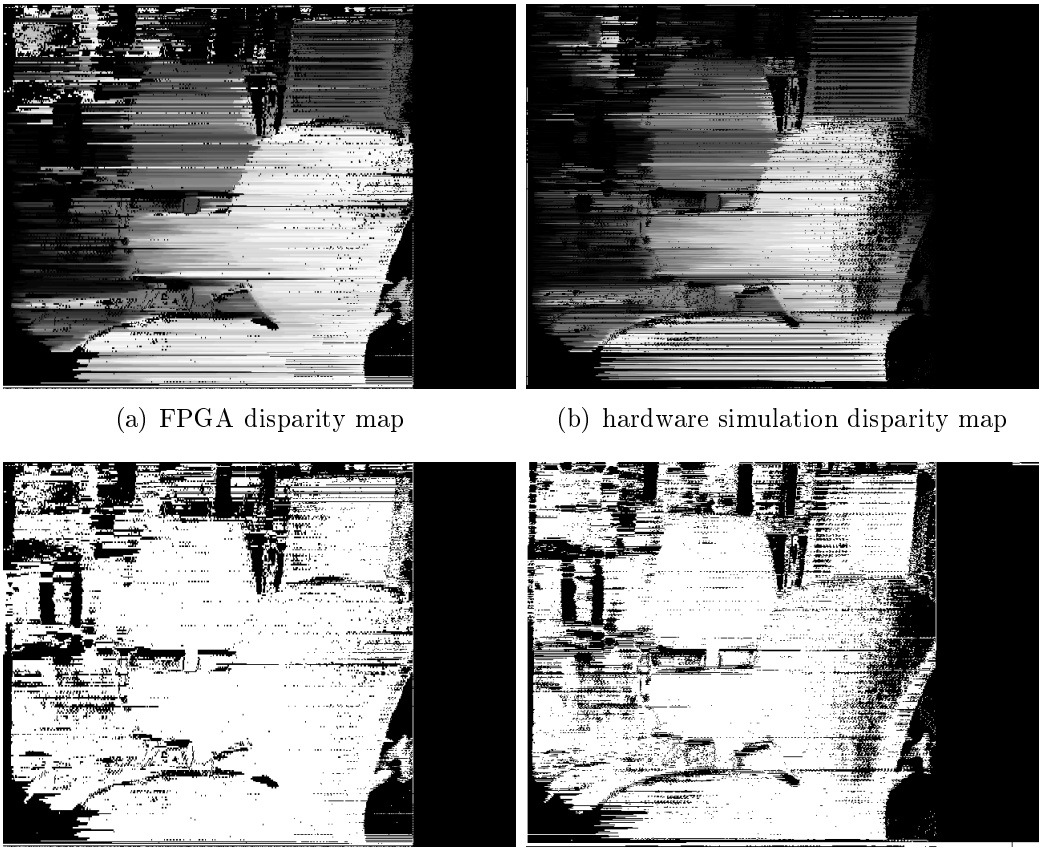
The resources required for the fully integrated system were obtained from Xilinx Project Navigator and are shown in Table 4.12. From this it can be seen that the largest occupied resource on the FPGA is the block RAM. Approximately 36% of the BRAM instances are unused on the FPGA, of which approximately 30% would be consumed by the final component of the system, the Object Tracking module [55]. It is often problematic when the block RAM usage reaches close to 90% capacity, as it is difficult for the place-and-route engine to route the design. A similar approach as in Section 4.3.3 was taken to evaluate the accuracy of the

Table 4.12: Resource utilization for the integrated stereo vision system

Resource	**Used**	**Available**	**Percentage**
Slices	8717	44,096	19%
4-input LUTs	17433	88,192	19%
BRAMs	286	444	64%
Multipliers	47	444	10%
Global Clocks	4	16	25%

high frame-rate stereo vision system. Figure 4.13 shows the results obtained from the FPGA and the hardware simulation. The outputs are generated using a stereo image pair that was captured using the 100 fps video-acquisition system. The results from software simulation served as the ground truth for results produced by the FPGA and hardware simulation. Table 4.13 shows the accuracy of the FPGA and hardware simulation results. From this table and through visual inspection of the occlusion maps it can be noticed that some errors exist.

A major contributor to the errors can be associated with the timing delays within the FPGA. Due to the large design, the length of many routes exceed the requirement and cause deviated results. As an experiment, the design was testing using a more recent and faster device. The FPGA chosen was the Xilinx Virtex-5 XC5VLX330T. The synthesis results indicated that the clock period would decrease from 20.525ns to 13.105ns, which would ensure that all routes would meet the timing requirements. The synthesis results were obtained from Xilinx Project Navigator's synthesis report file. Another possible reason for the difference can be due to Bayer interpolation. Some issues that arose during the integration and which may



(a) FPGA disparity map (b) hardware simulation disparity map

(c) FPGA occlusion map (d) hardware simulation occlusion map

Figure 4.13: Results of the integrated stereo vision system, $D_{max} = 64$ pixels

have lead to slightly poorer results include connection problems on the PCI Mezzanine Card slot which added signal interference to the input image. Another issue was the movement of the fixed-focal length lenses which caused the camera parameters to change and hence alter

the correctness of the rectification. Further necessary improvements to the fully integrated stereo vision system include interpolating the Bayer pattern for the raw (original) image and using C-mount lenses, which would provide improved stability and durability over the current fixed-focal length lenses. In comparison to the stereo vision systems introduced in

Table 4.13: Quality of the hardware simulation and FPGA hardware download

	RMS Error	**% Bad Pixel Match**
hardware simulation	0.2091	0.0153
FPGA download	1.0688	4.1008

Section 2.6.3, this system is the only stereo solution that uses an algorithm that gives a per-scanline global optimization of the cost function. Furthermore, the existing systems do not have additional constraints like the ordering constraint. Also, it should be mentioned that this is only stereo system that has published results based on the standard test sets from [54].

4.5 Summary

In Chapter 4, we began by measuring the performance, area, power consumption and accuracy of the two versions of the Stereo Rectification Module. We observed that the 200 fps SRM performed faster, used less resources on the FPGA and also consumed less power than the original 30 fps SRM. A comparison was performed to select the appropriate number of fractional-bits needed for the fixed-point operators. In the end, 18-bit fixed-point operators proved as the best choice with respect to precision and logic utilization. Furthermore, the improvements made to the SFGv2 allowed analyzing the warped images, by downloading them using the SRAM frame download feature, to estimate their accuracy.

Next, we evaluated the different implementations of the Stereo Extraction Module. The metrics used for the evaluation included performance, resource utilization and quality. The comparisons demonstrated that the interleaved hardware implementation of the Stereo Extraction Module vastly out-performed the initial and intermediate versions of the Stereo

Extraction Module with respect to speed, while producing results with comparable accuracy to the best existing stereo correspondence algorithms on standard stereo data sets [16].

Finally, the fully integrated high frame-rate stereo vision system was presented. The limitations of the block RAM resources on the Xilinx XC2VP100 FPGA did not allow SEM implementations over $D_{max} = 64$ for images of 640×480 resolution. Improvements to the results would be achieved by performing Bayer interpolation on the un-processed input image, and also by utilizing an FPGA with larger block RAM resources.

Chapter 5

Conclusion and Future Work

This thesis presents the implementation of a high frame-rate stereo vision system, which includes a high frame-rate stereo camera (video-acquisition system), a stereo image rectification module (video pre-processor) and a stereo extraction module. The novelty of this system, compared to other state-of-the-art systems, is noticed in its high frame-rate and quality of output. The implementation demonstrates that high frame-rate disparity estimations can be obtained while maintaining a high degree of accuracy. It is the only stereo system that has published results based on the standard test sets from [54]. An evaluation of this system indicates that disparity estimations are produced at frame-rates of 98.39 fps on 640×480 resolution images with a maximum disparity range of 64 pixels. These figures can easily be extended to 200 fps on 640×480 resolution images with a maximum disparity range of 128 pixels, but utilizing an FPGA with larger resources. Existing solutions perform well at video rate (*i.e.*, 30 fps), while others with higher performances are compromised by their accuracy.

The implementation of the system includes a stereo image rectification module which places corresponding points (features) in the two views on the same scanline, via a collineation computed from the estimated epipolar geometry based on camera calibration. This collineation allows for an easier correspondence search between pixels in the left and right images, for

the task of triangulation. The hardware design of the stereo image rectification module is parallelized in such a manner that both the left and right view pixels are produced simultaneously, to maximize the throughput. The design of the stereo extraction module exploits massive parallelization of functional units in order to achieve high frame-rate performance. The hardware implementation of the DPML stereo algorithm in [17, 3] was modified by introducing pipelining in two phases of the unit to decrease long combinational logic chains and to accommodate reading synchronous memories. The integration of the stereo vision system required creating an interface between the video-acquisition system and the stereo image rectification module. A second interface was required between the stereo image rectification module and the stereo extraction module.

Future Work

The initial goal of this system was to track and follow objects at high-speeds, equivalent to 200 fps, however resource limitations on the Xilinx Virtex-II Pro FPGA allowed for speeds closer to 100 fps. It should be mentioned that the object tracking module has been developed by Belshaw [55], and is currently in the process of being integrated with the existing system. The system has shown successes in many different aspects compared to other state-of-the-art hardware stereo systems. However, further improvements can be made to achieve and exceed the target speed of 200 fps. The first issue at hand, to make this possible, is to select an FPGA with larger resources. Specifically, one with a greater density of internal memories such as BRAMs, as this was the bottleneck in the implementation of the DPML stereo matching algorithm. The selection of a larger FPGA will allow replicating the stereo extraction module, in order to perform operations on two successive scanlines in parallel, thus achieving and exceeding the targeted 200 fps throughput. At the same time, the use of a larger FPGA will also help in improving the quality of the results. It will allow implementing the stereo extraction module with a 128 pixel disparity range on a 640×480 resolution image. Additionally, the quality of the results can be improved by performing Bayer interpolation

to obtain a better estimation of the pixel intensities. This change will help improve pixel matching for the estimation of depth in the scene. A final improvement to the system involves replacing the current fixed-focal length lenses, which are susceptible to moving, with rigid C-mount lenses which have an enclosure with more support and stability. This will make the camera more rigid and thus the validity of the camera calibration will improve.

Appendix A

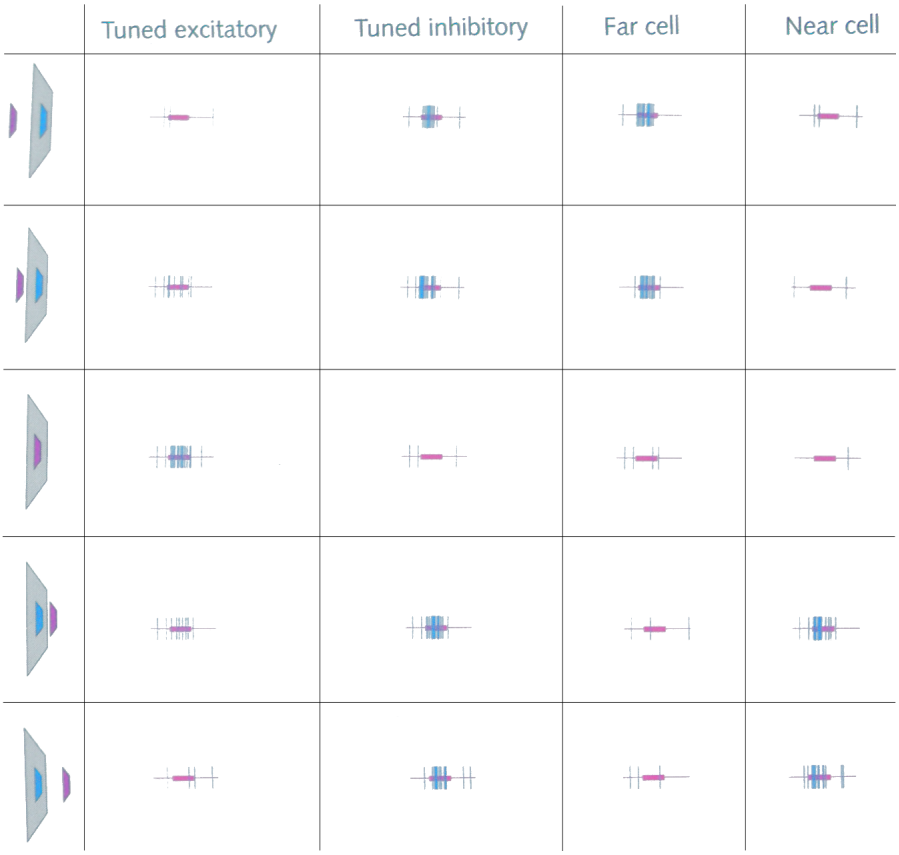


Figure A.1: Response of cells to different binocular disparities [6]

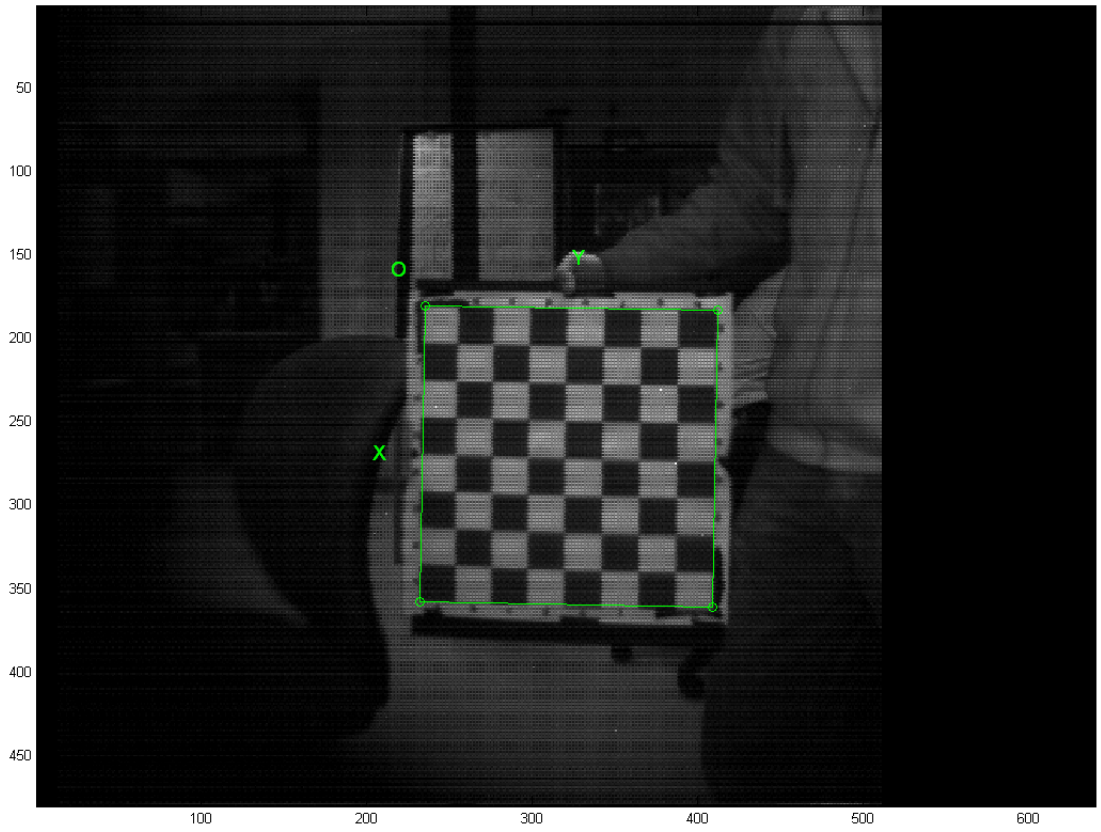


Figure A.2: Selection of the four extreme corners of the checkerboard pattern

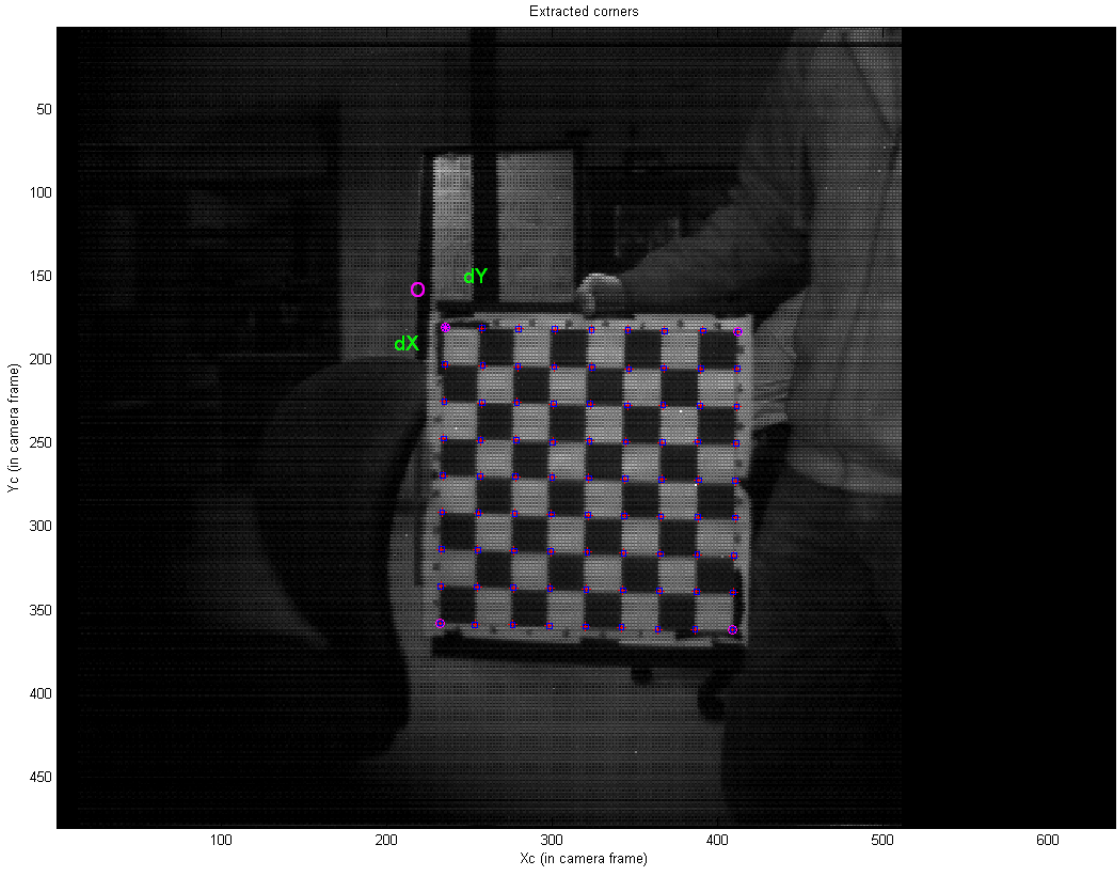


Figure A.3: Extracted corners of the checkerboard pattern

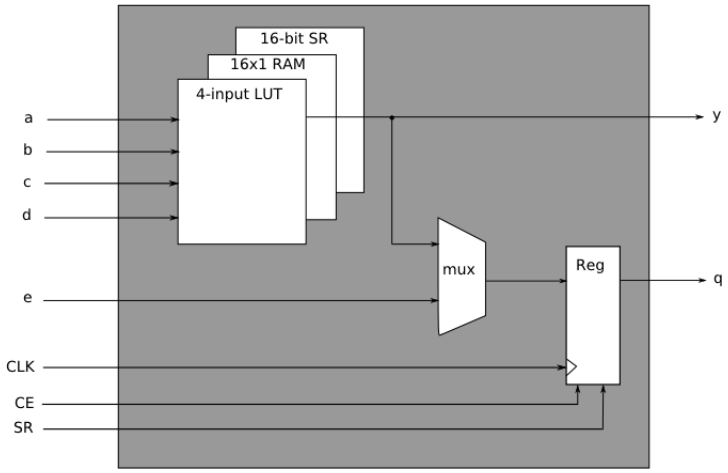


Figure A.4: A Xilinx logic cell

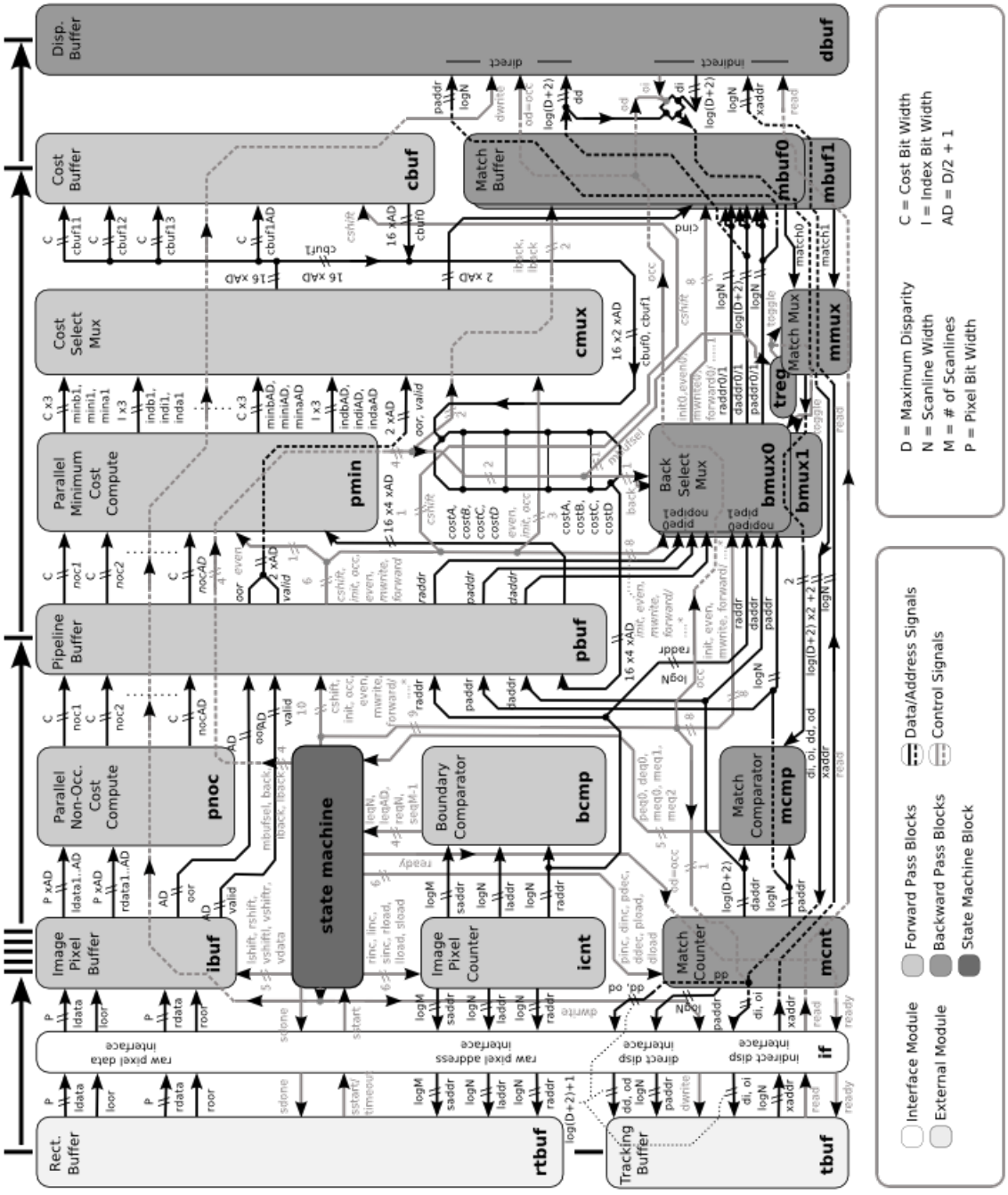


Figure A.5: Schematic of the interleaved Stereo Extraction Module

Appendix B

Algorithm 1 Forward-pass phase for DPML stereo correspondence formulation developed by Cox *et al.* [1]

```
% Initialize match and cost matricies
for c = 1 to N do
   MM(c,1:c) = 2;
   MM(c,(c+1):N) = 1;
   CM(c:N,c) = (((c-1):(N-1))*OC);
   CM(c,c:N) = (((c-1):(N-1))*OC);
end for;
% For each pixel in a row compute NOC
for l = 2 to N do
   for r = l to (l-Dmax) do
      min1 = CM(r-1,l-1) + NOC;
      min2 = CM(r-1,l) + OC;
      min3 = CM(r,l-1) + OC;
      CM(r,l) = min(min1,min2,min3) = cmin;
      if min1 == cmin then
         MM(r,l) = 0; %No occlusion
      else if min2 == cmin then
         MM(r,l) = 1; %Right occlusion
      else if min3 == cmin then
         MM(r,l) = 1; %Left occlusion
      end if
   end for
end for
```

Algorithm 2 Backward-pass phase for DPML stereo correspondence formulation developed by Cox *et al.* [1]

```
p = N; q = N;
while p != 1 and q != 1 do
    if MM(p,q) == 0 then
        DISP(q) = abs(p-q);
        OCC(q) = 0;
        p --;
        q --;
    else if MM(p,q) == 1 then
        OCC(q) = 1;
        p --;
    else if MM(p,q) == 2 then
        OCC(q) = 1;
        q --;
    end if
end while
```

Bibliography

[1] Ingemar J. Cox, Sunita L. Hingorani, Satish B. Rao, and Bruce M. Maggs. A Maximum Likelihood Stereo Algorithm. *Computer Vision and Image Understanding (CVIU)*, 63(3):542–567, May 1996.

[2] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)*, 47(1–3):7–42, April 2002.

[3] Siraj Sabihuddin, Jamin Islam, and W. James MacLean. Dynamic Programming Approach to High Frame-Rate Stereo Correspondence: A Pipelined Architecture Implemented on a Field Programmable Gate Array. In *21st Canadian Conference on Electrical and Computer Engineering*, pages 1461–1466, 2008.

[4] David H. Hubel. *Eye, Brain, and Vision.* Scientific America Library, 1988.

[5] American Health Assistance Foundation: Macular Degeneration Research. Anatomy of the Eye, http://www.ahaf.org/macular/about/AnatomyEye.htm. June 2008.

[6] Robert Snowden, Peter Thompson, and Tom Troscianko. *Basic Vision: An introduction to visual perception.* Oxford University Press, 2006.

[7] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[9] William Press, Saul Teukolsky, William Vetterling, and Brain Flannery. Numerical Recipes, http://www.nr.com. July 2006.

[10] Susan Blackford. Generalized RQ Factorization, http://www.netlib.org/lapack/lug/node47.html. July 2008.

[11] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj. August 2006.

[12] Zhengyou Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. *IEEE International Conference on Computer Vision (ICCV)*, 1:666–673, 1999.

[13] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[14] Hynek Bakstein and Radim Halir. Camera Calibration Toolbox, http://terezka.ufa.cas.cz/hynek/toolbox.html. July 2008.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 2001.

[16] Siraj Sabihuddin. Dense Stereo Reconstruction in a Field Programmable Gate Array. Master's thesis, University of Toronto, 10 King's College Road, Toronto, Ontario, M5S 3G4, 2008.

[17] W. James Maclean, Siraj Sabihuddin, and Jamin Islam. Leveraging Cost Matrix Structure for Dynamic Programming in Hardware. *Submitted for review: Computer Vision and Image Understanding (CVIU)*, June 2008.

[18] Ed Clarke. FPGAs and Structured ASICs: Low-Risk SoC for the Masses, http://www.design-reuse.com/articles/12360/fpgas-and-structured-asics-low-risk-soc-for-the-masses.html. July 2008.

[19] Pil Woo Chun, Jamin Islam, Valeri Kirischian, and Lev Kirischian. Improving Cost-Effectiveness Using a Micro-level Static Architecture for Stream Applications. *IEEE International Workshop on Electronic Design, Test and Applications (DELTA)*, 0:368–373, 2008.

[20] Clive Maxfield. *The Design Warriors Guide to FPGAs*. Newnes, 2004.

[21] IEEE standard VHDL language reference manual. *ANSI/IEEE Std 1076-1993*, Jun 1994.

[22] Xilinx Inc. *Xilinx Synthesis Technology (XST) User Guide 9.2i*. Xilinx Inc., 2007.

[23] Patrick Courtney, Neil A. Thacker, and Chris R. Brown. A Hardware Architecture for Image Rectification and Ground Plane Obstacle Detection. *11th IAPR International Conference on Pattern Recognition*, 4:23–26, 1992.

[24] Stefan Jorg, Langwald Langwald, and Mathias Nickl. FPGA based Real-Time Visual Servoing. *International Conference on Pattern Recognition (ICPR)*, 01:749–753, 2004.

[25] Divyang K. Masrani and W. James MacLean. A Real-Time Large Disparity Range Stereo-System using FPGAs. *IEEE International Conference on Computer Vision Systems (ICVS '06)*, pages 13–13, January 2006.

[26] Anton Serguienko. Evaluation of Image Warping Algorithms for Implementation in FPGA. Master's thesis, Linkopings Universitet, 2008.

[27] Cristian Vancea and Sergiu Nedevschi. LUT-based Image Rectification Module Implemented in FPGA. *2007 IEEE International Conference on Intelligent Computer Communication and Processing*, pages 147–154, 2007.

[28] Dongil Han and Dae-Hwan Hwang. A Novel Stereo Matching Method for Wide Disparity Range Detection. In *International Conference on Image Analysis and Recognition (ICIAR)*, pages 643–650, 2005.

[29] Masonori Hariyama, Yasuhiro Kobayashi, Haruka Sasaki, and Michitaka Kameyama. FPGA Implementation of a Stereo Matching Processor Based on Window-Parallel-and-Pixel-Parallel Architecture. In *Midwest Symposium on Circuits and Systems (MWSCAS)*, volume 2, pages 1219–1222, 2005.

[30] Johel Miteran, Jean-Philippe Zimmer, Michel Paindavoine, and Julien Dubois. Real-Time 3D Face Acquisition Using Reconfigurable Hybrid Architecture. *EURASIP Journal on Image and Video Processing*, 2007.

[31] Yosuke Miyajima and Tsutomu Maruyama. A Real-Time Stereo Vision System with FPGA. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 448–457, 2003.

[32] Stefania Perri, Daniela Colonna, Paolo Zicari, and Pasquale Corsonello. SAD-Based Stereo Matching Circuit for FPGAs. In *International Conference on Electronics, Circuits and Systems (ICECS)*, pages 846–849, December 2006.

[33] Yunde Jia, Xiaoxun Zhang, Mingxiang Li, and Luping An. A Miniature Stereo Vision Machine (MSVM-III) for Dense Disparity Mapping. In *Internationl Conference on Pattern Recognition (ICPR)*, volume 01, pages 728–731, 2004.

[34] Jan van der Horst, Rien van Leeuwen, Harry Broers, Richard Kleihorst, and Pieter Jonker. A Real-Time Stereo SmartCam, using FPGA, SIMD and VLIW. In *Workshop on Applications of Computer Vision*, May 2006.

[35] Ahmad Darabiha, W. James MacLean, and Jonathan Rose. Reconfigurable Hardware Implementation of a Phase-Correlation Stereo Algorithm. *Machine Vision and Applications*, 17(2):116–132, March 2006.

[36] Javier Diaz, Eduardo Ros, Silvio P. Sabatini, Fabio Solari, and Sonia Mota. A Phase-Based Stereo Vision System-On-A-Chip. *Biosystems*, 87(2–3):314–321, July 2006.

[37] John Iselin Woodfill, Gaile Gordon, and Ron Buck. Tyzx DeepSea High Speed Stereo Vision System. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, volume 3, pages 41–45, 2004.

[38] Olivier Faugeras, Bernard Hotz, Herve Mathieu, Thierry Vieville, Zhengyou Zhang, Pascal Fua, Eric Theron, Laurent Moll, Gerard Berry, Jean Vuillemin, Patrice Bertin, and Catherine Proy. Real time correlation based stereo: algorithm implementations and applications. Design Document RR-2013, INRIA, 1993.

[39] J. Woodfill and B. Von Herzen. Real-time stereo vision on the PARTS reconfigurable computer. *IEEE Symposium on FPGA-Based Custom Computer Machines (FCCM)*, pages 201–209, 1997.

[40] John Iselin Woodfill, Gaile Gordon, Dave Jurasek, Terrance Brown, and Ron Buck. The Tyzx DeepSea G2 Vision System, ATaskable, Embedded Stereo Camera. *Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 126–133, 2006.

[41] Takeo Kanade, Atsushi Yoshida, Kazuo Oda, Hiroshi Kano, and Masaya Tanaka. A Stereo Machine for Video-Rate Dense Depth Mapping and Its New Applications. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, 00:196, 1996.

[42] Siraj Sabihuddin and W. James MacLean. Maximum-Likelihood Stereo Correspondence using Field Programmable Gate Arrays. *International Conference on Computer Vision Systems (ICVS)*, March 2007.

[43] Raphael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Prentice Hall, 2 edition, 2004.

[44] Valeri Kirishchian, Pil Woo Chun, and Sergiy Zhelnakov. Stereo Frame Grabber V.1. Design document, Ryerson University, 350 Victoria Street, Toronto, Ontario, Canada M5B2K3, December 2007.

[45] Bryce E. Bayer. Color Imaging Array, Patent number: 3971065. 1976.

[46] Sergiy Zhelnakov. Development and Implementation of Interactive 3D Video Enviroment on Run-time Reconfigurable FPGA Platform. Master's thesis, Ryerson University, 350 Victoria Street, Toronto, Ontario, Canada M5B2K3, 2006.

[47] Pil Woo Chun, Valeri Kirishchian, and Sergiy Zhelnakov. Stereo Frame Grabber V.2. Design document, Ryerson University, 350 Victoria Street, Toronto, Ontario, Canada M5B2K3, January 2008.

[48] Micron Technology Inc. *MT9V403 - 1/2-Inch VGA Digital Image Sensor*, July 2005.

[49] CMC Microsystems, 210A Carruthers Hall, Kingston, Ontario, Canada, K7L 3N6. *CMC Microsystems Getting started with the Virtex-II Pro FPGA Prototyping Station V3.0*, February 2007.

[50] *Virtex-II Pro Platform FPGA Handbook*, volume 2. Xilinx Inc., 2002.

[51] Xilinx Website. http://www.xilinx.com. September 2006.

[52] Jamin Islam, Pil Woo Chun, W. James MacLean, and Lev Kirischian. Lowering Power Consumption Using Run-time Reconfiguration for Stereo Rectification. In *21st Canadian Conference on Electrical and Computer Engineering*, pages 1693–1698, 2008.

[53] Daniel Scharstein and Richard Szeliski. High Accuracy Stereo Depth Maps using Structured Light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 195–202, June 2003.

[54] Daniel Scharstein and Richard Szeliski. Middlebury Stereo Vision Page, http://vision.middlebury.edu/stereo. June 2008.

[55] Michael S. Belshaw. A High-Speed Iterative Closest Point Tracker on an FPGA Platform. Master's thesis, Queen's University, 99 University Avenue, Kingston, Ontario, K7L 3N6, 2008.

Glossary

ASIC Application Specific Integrated Circuit

BRAM Block selectRAM

CLB Configurable Logic Block

CMOS Complementary Metal-Oxide Semiconductor

CPLD Complex Programmable Logic Device

CPU Central Processing Unit

DAC Digital-to-Analog Converter

DCM Digital Clock Manager

DPML Dynamic Programming Maximum Likelihood

DSP Digital Signal Processor

FF Flip-Flop

FIFO First In, First Out

FPGA Field Programmable Gate Array

FPS Frames Per Second

GUI Graphical User Interface

I/O Input/Output

IC Integrated Circuit

ICM Image Capture Module

IP Intellectual Property

LCS Longest Common Subsequence

LED Light-Emitting Diode

LiDAR Light Detection and Ranging

LUT Look-up Table

LVDS Low-Voltage Differential Signaling

MAC Multiply-Accumulate

MB Megabyte

MHz MegaHertz

MSB Most Significant Bit

NCC Normalized Cross-Correlation

OOR Out-of-Range

PC Personal Computer

PCI Peripheral Component Interconnect

PMC PCI Mezzanine Card

PR Partial Reconfiguration

RAM Random Access Memory

RGB Red-Green-Blue

RMSE Root Mean Squared Error

RTOS Real-Time Operating System

SAD Sum of Absolute Difference

SEM Stereo Extraction Module

SFGv1 Stereo Frame Grabber v.1

SFGv2 Stereo Frame Grabber v.2

SRAM Static RAM

SRM Stereo Rectification Module

SSD Sum of Squared Difference

SVD Singular Value Decomposition

TSP Taylor Series Polynomial

USB Universal Serial Bus

VGA Video Graphics Array

VHDL Very High Speed Integrated Circuit Hardware Description Language

VPM Video Processing Module

VPP Video Pre-processor