Expanding Stereo-Disparity Range in an FPGA-system While Keeping Resource Utilisation Low

by

Divyang K. Masrani

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of The Edward S. Rogers Sr. Department of
Electrical and Computer Engineering
University of Toronto

Abstract

Expanding Stereo-Disparity Range in an FPGA-system While Keeping Resource Utilisation Low

Divyang K. Masrani

Master of Applied Science

Graduate Department of The Edward S. Rogers Sr. Department of Electrical and Computer Engineering

University of Toronto

2006

We present the design and implementation of a Field-Programmable Gate Array (FPGA) based dense stereo depth measurement system that is capable of handling a very large disparity range. The throughput of the system is 60 frames/second on 640×240 images. Image rectification and consistency check improve accuracy of the results. The system is based on the Local Weighted Phase-Correlation algorithm [?] which estimates disparity using a multi-scale and multi-orientation approach. Though FPGAs are ideal devices to exploit the inherent parallelism in many computer vision algorithms, their finite resource capacity poses a challenge when adapting a system to deal with large image sizes or disparity ranges. We utilise the temporal information available in a video sequence to design a novel architecture for the correlation unit to achieve correlation over a large range while keeping the resource utilisation very low as compared to a naive approach.

Dedication

To my family.

Acknowledgements

I am sincerely grateful to my supervisor, Professor James MacLean, for his invaluable guidance and support that was vital to the success of this thesis.

I would like to express my thanks to my colleagues and friends in the Vision and Image Dynamics Lab and at the University, who were a source of support and encouragement throughout the project. They provided for some very useful discussions both on matters related to the project and to life outside the university.

The University of Toronto, OGS, Micronet R & D and CMC are also acknowledged for their financial and equipment support.

Above all, I am deeply grateful to my family and friends who have made this journey possible for me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The goal of a *computational vision* system is to automate the task of generating a description of a given scene through an analysis of captured images of the scene. The description of the scene can consist of information such as the location of single or multiple objects in the scene, the identity of objects, or even any actions an object is performing.

The task of building a general purpose computational-vision system is a "grand-challenge" due to the compute-intensive nature of many vision algorithms. However, researchers have been successful in designing algorithms and building systems that deal with some specific tasks of the human vision system. One important feature of the human vision system is its ability to perceive depth of a viewed scene. This ability to perceive depth, known as *stereo vision*, or *stereopsis* is made possible by the difference in viewpoints of the scene when sensed by our left and right eyes. The information about depth in a scene is of great importance because it helps us navigate in a three-dimensional environment and aids us in recognising objects of interest, among other tasks.

In computer based stereo-vision systems, a *stereo-rig* is a pair of cameras placed side-by-side, much like our eyes, to capture the left and right images. The processing required to extract depth information from the image pair may seem second nature when performed by the human brain due to its immense and complex computational capabilities. In a stereo-vision system, this processing is carried out using a computing platform that can be based on software, hardware, or a mixture of the two. The depth information is encoded in the *disparity*, defined as the difference in pixel locations of corresponding points in the image pair. The disparity is inversely proportional to the distance of an object from the cameras, so the disparity increases as objects get closer to the cameras. The estimation of this disparity then becomes the primary task of a stereo-vision system.

In the simplest setup of a stereo-rig, where the optical axes of the two cameras are parallel and the vertical axes are aligned, corresponding pixels lie at the same vertical coordinate in the image pair. The search for the corresponding pixel is therefore limited to the same *scanline* in the image pair, which allows processing of each scanline as they arrive. In the more general case where the cameras are not aligned as described above, the search for corresponding pixel may span across numerous scanlines and this increases the computational load of the system. When the cameras are not in the ideal setup, *Image rectification* of input images can be performed. *rectification* is the process by which the input image pair is warped to resemble the output from an aligned stereo-rig.

Often, when viewing a scene from different viewpoints as in a stereo setup, objects visible in one image may not be visible in the other image. A foreground object hides, or *occludes*, different parts of the background in the left and right views, a phenomenon known as *occlusion.* In addition, the information present at the left edge of the image captured by the left camera is not available in the right image and vice-versa as this part of the scene falls outside the viewing area of the other camera. This further complicates the task of accurate disparity estimation because pixels visible in one image may not have a corresponding match in the other image of the pair.

Many stereo-vision applications require the depth information for every single pixel, which translates to finding a *dense disparity estimate*, the estimation of disparity for every single pixel in the image. Some of the most commonly used cameras today capture images that are 640 × 480 pixels in dimension at a rate of 30 frames per second (fps). To bridge the gap between the desired frame-rate response and the actual performance that general purpose computing platforms can provide, appropriate hardware accelerators need to be used.

Dense stereo-vision algorithms based on a technique called correlation have inherent parallelism which can be exploited to achieve significant improvements in the execution time of the algorithm. Hardware accelerators in the form of Digital Signal Processors (DSPs), reconfigurable devices such as Field Programmable Gate Arrays (FPGAs), and Application Specific Integrated Circuits (ASICs), all provide a viable alternative to take advantage of inherent parallelism as opposed to the use of expensive and large-scale parallel computers for a similar task.

DSPs are commonly used to speed up the computation of many signal and image processing algorithms. Though easy to program, DSPs have a fixed architecture that limits the kind of operations that can be performed. DSPs therefore do not provide a system-on-a-chip solution which is a drawback when space and mobility are a concern. ASICs on the other hand provide the greatest amount of flexibility in designing the architecture but suffer from a long and tedious design process. Furthermore, the high cost of designing and fabricating an ASIC can make it prohibitive to use.

Reconfigurable devices such as FPGAs provide a middle ground. The design process is shorter and cheaper than for an ASIC and they provide much greater flexibility than DSPs making it possible to develop a variety of algorithms from start to end on the FPGA. Another important advantage of FPGAs is that they are reconfigurable, a process that can be completed in a span of milli-seconds, so that the same chip can be used for different algorithms. This makes FPGAs ideal for computer-vision tasks which often require the result from multiple algorithms to accurately make a decision. Take the case of image segmentation. Methods used for image segmentation rely on information such as texture, depth, and colour. No one method will accurately perform segmentation in every scenario, but a combination of information provides the best solution. An FPGA can be easily reconfigured with these different algorithms as needed.

While designing with FPGAs is faster than designing Application Specific ICs (ASICs), it suffers from the problem of fixed resources. In an application based on a serial CPU or DSP, one can typically add memory or disk space to allow the algorithm to handle a larger version of the same problem, for example larger image sizes or increased disparity ranges in the case of stereo. System performance in terms of timing may suffer, but the advantage of such serial implementations is that the new system still runs. In the case of FPGA-based systems, there is a finite amount of logic available, and when this is exhausted the only solution is to add another device or modify the algorithm. Not only is this costly from the design point of view, but may also involve the additional design issue of how to partition the logic algorithm across several devices which is non-trivial. Apart from finding logical partitions in the algorithm, issues with transferring large amounts of data from one FPGA to another in the limited bandwidth between available between FPGAs and also meeting strict timing requirements make this a challenging task. Keeping this in mind, it is important to devise a suitable architecture for the vision system on hand for a successful implementation on FPGAs.

1.1 Thesis Objectives and Contribution

Stereo disparity estimation is a prime application for a hardware accelerated computer vision system. Since stereo can provide depth information, it has potential uses in navigation systems, robotics, object recognition and surveillance systems, just to name a few. Due to the computational complexity of many stereo algorithms, a number of attempts have been made to implement such systems using hardware [?, ?, ?, ?], including reconfigurable hardware in the form of FPGAs [?, ?, ?, ?, ?]. One of the more recent attempts at developing a stereo-vision system is described in [?]. The system is based on the Local Weighted Phase Correlation (LWPC) algorithm [?] and is implemented on the Transmogrifier-3 reconfigurable computing platform [?]. Though the system provides dense disparity estimates at 30 fps, it has several major limitations. The amount of logic resources required by the system is directly proportional to the largest disparity that the system can support. The system is capable of handling a maximum disparity of only 20 pixels, which for the particular set-up of the system does not generate accurate depth information for objects closer than 2 meters from the camera. This is a severe hinderance to the use of the system in many applications such as autonomous navigation. Furthermore, the system does not attempt to rectify the input images, which affects the accuracy of the results, and it supports an image size of only 256 $\times$ 360 pixels.

In this work, we address the specific limitations of the previous system mentioned above. The goal of this work is to develop a versatile real-time stereo-vision platform with various salient features; capability to handle very large disparities, improved accuracy by pre-processing (input image rectification), and the ability to handle larger images. The highlight of the work is the development of a novel architecture that can handle the correspondence task for scenes with very large disparities, but without increased resource usage on the FPGA, as compared to [?]. The key to achieving large disparity correspondence matches is the use of shiftable correlation windows that track the disparity estimate for each pixel over time, as well as secondary roving correlation windows that explore the correlation surface outside the range of the shiftable tracking window in order to detect new matches when the tracking window is centred on an incorrect match. In our work, the shiftable tracking window is termed the *Primary Tracking Window (PTW)* and the window that performs roving correlation as the *Secondary Roving Window (SRW)*. The basic assumption in our approach is that, in most cases, disparity values do not change radically between frames, thus allowing some of the computation to be spread over time.

The thesis is organised as follows. Chapter 2 describes the relevant background in stereo-vision and reconfigurable computing systems. In Chapter 3 we present the main focus of this thesis; development of hardware architectures for image rectification, expanded disparity range support, and consistency check of stereo disparity estimates. Chapter 4 compares the stereo-system in this work with others in the literature and disparity results from the system are also presented. We conclude in Chapter 5 with a summary and a discussion of possible directions for future work.

Chapter 2

Background

In Chapter 1 we established the focus of this work; developing a novel architecture for a stereo-vision system that is capable of supporting very large disparity range without a corresponding increase in the logic resource usage. In order to achieve this with the limited resources offered by FPGAs, an understanding of the various stereo matching algorithms and knowledge of the hardware technology that will be used is required.

This chapter first provides an overview of binocular stereo vision, the options available for stereo-matching and a justification for the selected technique for this work in Section 2.1. Section 2.2 gives an introduction to FPGA technology and the reconfigurable computing platform used in this work. Finally, in Section 2.3, a brief review of previous work in hardware-based computer vision and image processing is presented.

2.1 Theoretical Basis

Stereopsis is the process in visual perception leading to perception of depth or distance of objects. Depth from stereopsis, in particular *binocular stereopsis*, arises from the difference in the viewpoints of the two cameras that view the scene. This process is known as *triangulation* and is illustrated in Figure 2.1. In the case of binocular stereo, a *scene point* (point in the 3-D world space) is projected on the two image planes of the camera pair; these projected points are knows as image points. Given the centre of projection of the two cameras, two rays are formed that go through the the centre of projection of each camera and image point of the respective camera. The location of the scene point is at the intersection of the two rays.

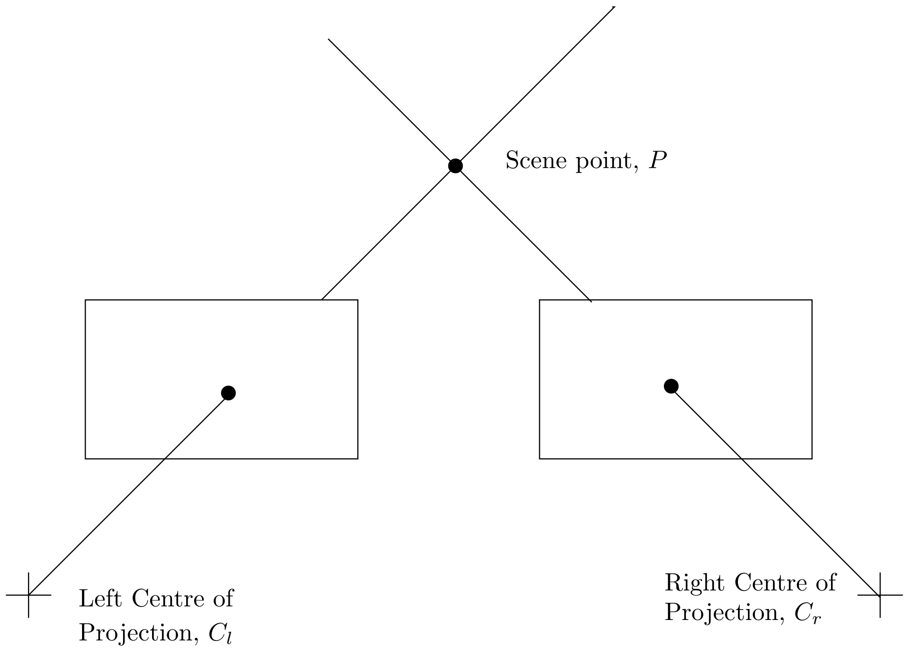

Scene point, P
Left Centre of Projection, C_l
Right Centre of Projection, C_r

Figure 2.1: Given the projections of a scene point in both the left and right image, the three-dimensional location of the scene point can be determined by triangulation.

From a computational standpoint, a stereo system must solve two problems [?]. The key to performing triangulation is to first establish point correspondences, or in other words, for each point in one image, find the point in the other image that is a projection of the same scene point if it exists. This task of solving for stereo correspondence is the first problem of stereo and because it involves searching for matching points in the binocular image pair, it is often termed *stereo matching*. The search for corresponding points need only be performed along a one-dimensional line rather than a two-dimensional search area if the stereo images have been rectified. The surety of finding the corresponding point along the line is guaranteed by the *epipolar constraint* which says that given an image point p_l, its corresponding point p_r in the other other image is constrained to lie along the *epipolar line* which is formed by the intersection of the plane formed by P, C_l, and C_r and the image plane, where P is the scene point and C_l and C_r are the camera centres of the left and right cameras as illustrated in Figure 2.2.

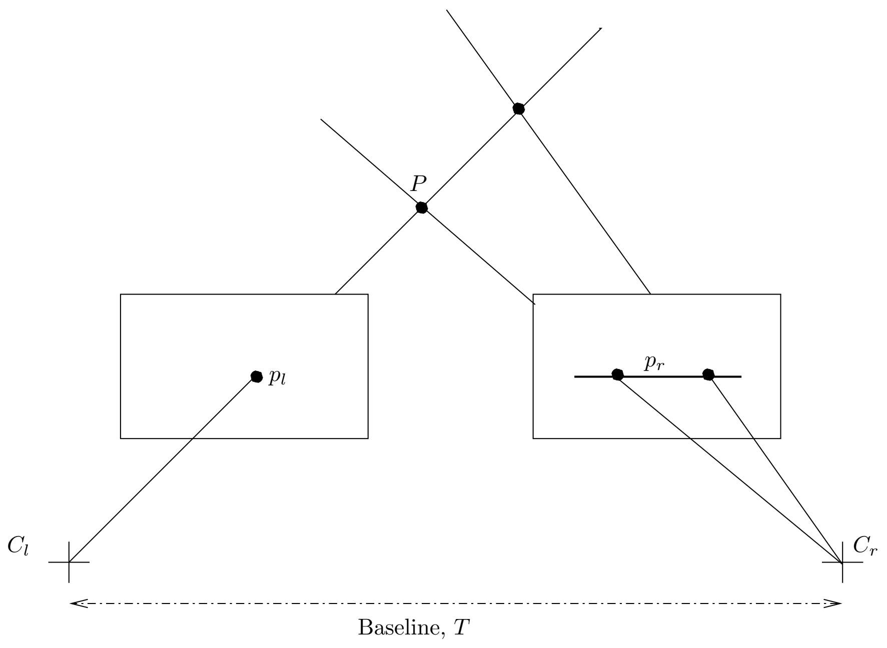


Figure 2.2: The epipolar constraint guarantees the location of a corresponding image point along the epipolar line in the image pair.

The simplest binocular stereo system is one whose optical axes are parallel, vertical axes are aligned and with each camera having identical focal lengths. In such a configuration, the epipolar lines coincide with the *scanlines* of the images, thus simplifying the search. For a scene point P having corresponding image points p_l at (u, v) and p_r at (u', v') one in each image of the pair, the vertical positions are the same, that is, $v = v'$ as shown in Figure 2.3. The difference in the horizontal locations of the corresponding image points

$$d = u' - u \tag{2.1}$$

is termed the *binocular disparity*, or simply the *disparity*. The horizontal locations, u and u' are measured relative to the the respective image centres.

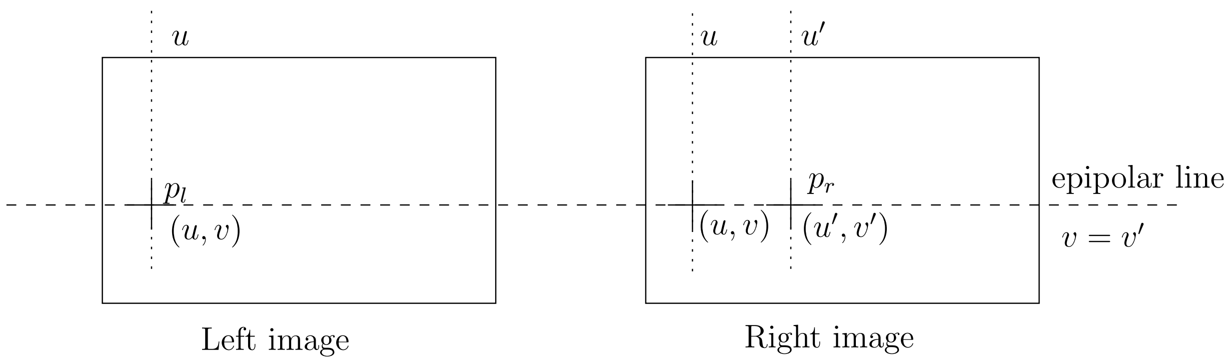


Figure 2.3: Epipolar constraint in parallel camera configuration. p_l and p_r are point correspondences of the same scene point. p_l and p_r lie along the same scanline (same vertical position).

Given the camera calibration parameters, the depth of a point P in 3-D space can be obtained using similar triangles as shown in Figure 2.4. The depth z is calculation using the equation

$$z = f\frac{T}{d} \tag{2.2}$$

where f is the focal length of the camera, and $T = \| C_l - C_r \|$ is the baseline of the stereo system. The computation of this depth is the second problem of stereo, also known as the problem of *reconstruction*. This is a more challenging problem when the cameras are in a general position.

2.1.1 Stereo Rectification

The problem of finding correspondence in an image pair taken from cameras in a general position can be simplified by rectifying the image pair before proceeding to find the matches. Rectification is the process by which the two images taken from cameras in a general position are reprojected onto a common image plane that is parallel to the baseline of the stereo-rig. This is illustrated in Figure 2.5. The rectified images can be thought of as acquired by a new stereo-rig, obtained by rotating the original cameras around their optical centres.

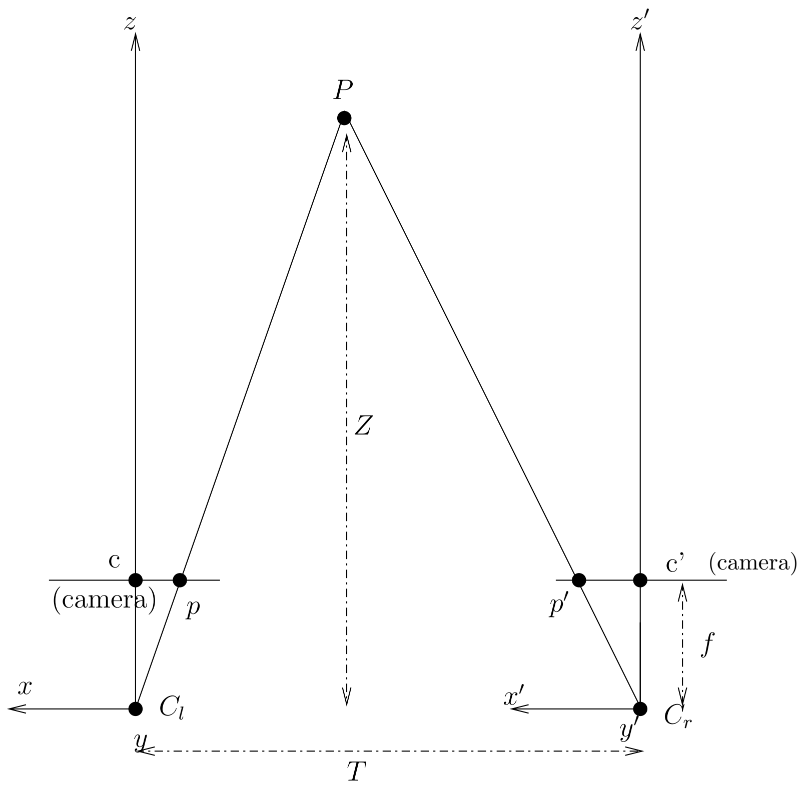


Figure 2.4: Given the stereo rig calibration parameters and the disparity measure, the depth, Z, of a scene point, P, can be determined. T is the baseline of the stereo rig and f is the focal length of the cameras.

Mathematically, the reprojection can be described by a 3×3 projection or *homography* matrix $\mathbf{H}$. The matrix $\mathbf{H}$ represents the transformation of coordinates from the original image to the reprojected image as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.3}$$

The stereo pair can then be rectified by applying two appropriate homographies $\mathbf{H}_l$ and $\mathbf{H}_r$ to the two images. $\mathbf{H}_l$ and $\mathbf{H}_r$ are computed from the position and orientation of the two cameras given that the stereo-rig has been calibrated and its intrinsic and extrinsic parameters are known. Further information on this process is given in Appendix A.

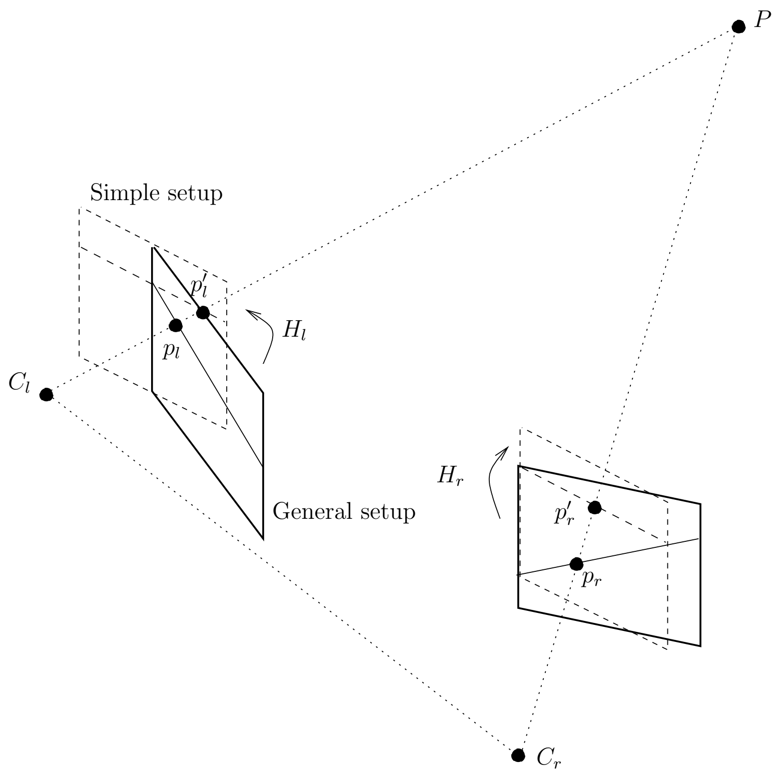


Figure 2.5: The simple stereo geometry can be derived from a general setup by reprojecting the two images onto a plane parallel to the baseline.

The actual rectification is performed using backward mapping by re-sampling the original images. For each pixel (x', y') in the rectified image, the corresponding pixel (x, y) in the original image is computed using $\mathbf{H}^{-1}$. This backward mapping produces real-valued coordinates in the original image so the intensity value of each pixel in the rectified image must be interpolated from pixels in this neighbourhood. One method of obtaining the intensity values for the rectified image is through *bilinear interpolation*, which computes the intensity value from a neighbourhood of four pixels.

2.1.2 Stereo Matching

According to [**?**], stereo matching algorithms, in general, can be broken down into the following four steps:

1. matching cost computation: some common ones include *squared intensity differences* (SD), *absolute intensity differences* (AD), normalised cross-correlation, and binary matching costs based on features such as edges. The Locally Weighted Phase Correlation (LWPC) algorithm used in this work is similar to normalised cross-correlation;

2. cost (support) aggregation: the summing or averaging of the matching cost over a *support region*, which is normally in the neighbourhood of the pixel;

3. disparity computation / optimisation: refers to selecting the best disparity estimate based on the aggregated cost and in some cases the subsequent improvement of the estimate with respect to some criteria; and

4. disparity refinement: in certain situations, there might be a need to obtain disparity estimate at sub-pixel accuracy using methods such as iterative gradient descent or fitting a curve to the matching costs at discrete disparity levels.

Specific algorithms may alter the sequence of the steps, combine steps, or skip some of the computation altogether.

Stereo matching algorithms can be classed into two categories; one that generate sparse disparity maps and another that generate dense disparity maps. Algorithms that generate sparse disparity maps rely on features such as edges or corners for matching and this matching is commonly known as *feature based matching.* A detailed survey of many sparse algorithms is available in [**?**] and will not be discussed here. Many vision applications require dense disparity estimates and we focus on this class of algorithms in this work. Dense disparity estimates can be generated from a sparse map by interpolation, but this approach requires several assumptions about the scene geometry, and is burdened with difficulties. More important is the class of algorithms that generate dense disparity maps directly without the need for interpolation.

Stereo matching algorithms that generate dense disparity maps can further be classified into *local* (window-based) or *global* depending on the optimization performed in *Step 3* of the above described steps. The computational emphasis of local methods is put on matching cost computation and on the cost aggregation steps. The optimisation is a local "winner-take-all" optimization at each pixel [?] limited to the extent of the correlation window. Despite the drawback of only local optimization, the fact that these algorithms use a *local support* area, commonly known as the *correlation window* means they exhibit regular computational structures that allow for efficient parallel implementation on hardware such as DSPs, ASICs, and FPGAs. A number of local algorithms have been implemented on various types of hardware and a survey of these can be found in Section 2.3.

The computational focus of global algorithms, on the other hand, is on the optimization step. These algorithms seek to optimise a global energy function consisting of a "data term" and a "smoothness term",

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \tag{2.4}$$

The data term, $E_{data}(d)$, gives a measure of the the likeness between the input image pair and the disparity function d. The smoothness assumptions made by the algorithm in the viewed scene are encoded in the smoothness term, $E_{smooth}(d)$. Global algorithms use several different methods to achieve minimisation of the energy function. These include simulated annealing [?], relaxation labeling [?] and non-linear diffusion of support [?]. More recently, algorithms based on graph-cuts [?] have been developed, and represent the state of the art for accuracy.

It is worth noting that though global algorithms provide more accurate dense disparity measurements compared to local algorithms, they have much higher computational costs, where the 2-D optimisation of (global equation) is often NP-hard [?], that render this class of algorithms unsuited to frame-rate applications. With current hardware technology, it is a challenge to even implement local algorithms that use complex matching techniques. It is the focus of this work to first design an efficient architecture for hardware implementation of local algorithms.

2.1.3 Locally Weighted Phase Correlation

The system implemented in this work is based on the "Local Weighted Phase Correlation" (LWPC) algorithm [?], which estimates disparity at a set of pre-shifts using a multi-scale, multi-orientation approach. A version of this algorithm was implemented in [?] but that system is limited to handling a maximum disparity of 20 pixels due to resource limitations on the FPGA, and has no input image rectification.

The LWPC algorithm has four major steps. A three-scale Gaussian pyramid is first created from the original images, sub-sampled at each level by a factor of two horizontally and vertically. Each level of the pyramid is decomposed into three orientations by applying quadrature-pair G2-H2 filters , tuned to orientations $0^o, +45^o, and -45^o$ [?]. G2 is the second derivative, $\frac{\delta^2}{\delta x^2}G(x, y; \mu, \sigma)$, of a Gaussian and H2 is the Hilbert transform of G2. G2-H2 filters of any orientation can be synthesised as linear combination of a set of "basis filters"; three basis filters for G2 and four for H2 are needed.

Correlation is then performed on each filtered pair of images at a set of pre-shifts. The correlation scores for a pair of images at each of these pre-shifts is collectively known as the *confidence measure* function. The confidence measure is then summed across all the scales and orientations so that spurious peaks are attenuated while the true peaks are re-inforced. The correlation results for the coarser scales must be interpolated to have the same resolution as the original scale before performing the summation. Finally, the peak must be located in the combined confidence measure function for each pixel. This peak is the estimate of the disparity for the particular pixel. The algorithm can be summarised as follows:

1. Create a Gaussian pyramid with total number of scales S for both left and right images. Apply spatially-oriented quadrature-pair filters [?] to each scale of the pyramid at F orientations. If $K_j(x)$ is the filter impulse response of the jth orientation, then we can write the complex-valued output of the convolution of $K_j(x)$ with each scale of the left and right images, $I_l(x)$ and $I_r(x)$, as

$$\begin{aligned} O_l(x) &= \rho_l(x)e^{i\phi_l(x)} = K_j(x) \otimes I_l(x) \quad \textit{and} \\ O_r(x) &= \rho_r(x)e^{i\phi_r(x)} = K_j(x) \otimes I_r(x) \end{aligned}$$

   in polar representation, where $\rho(x) = |O(x)|$ is the amplitude and $\phi(x) = \arg[O(x)]$ is the phase of the complex response.

2. For each scale and orientation, compute local voting functions $C_{j,s}(x,\tau)$ in a window centred at τ as

$$C_{j,s}(x,\tau) = \frac{W(x) \otimes [O_l(x)O_r^*(x+\tau)]}{\sqrt{W(x) \otimes |O_l(x)|^2}\sqrt{W(x) \otimes |O_r(x)|^2}} , \tag{2.5}$$

   where $W(x)$ is a smoothing, localized window and τ is the pre-shift of the right filter output.

3. Combine the voting functions $C_{j,s}(x,\tau)$ over all orientations, $1 \leq j \leq F$, and scales, $1 \leq s \leq S$, where F is the total number of orientations, to get the cumulative voting function

$$V(x,\tau) = \sum_{j,s} C_{j,s}(x,\tau) .$$

4. For each image position x, find the τ value corresponding to the peak in the real part of $V(x,\tau)$ as an estimate for the true disparity.

The reason for basing our work on this algorithm is two-fold. First, the LWPC algorithm is one of the more complex and recent local algorithms. The results obtained using this algorithm have greater accuracy than simpler local algorithms such as ones based on sum-of-squared-differences (SSD) [?, ?, ?] or sum-of-absolute-differences (SAD)

[?]. The second is that a previous implementation [?] of this algorithm exists, but is limited to a maximum disparity of 20 pixels, which in the set-up of that system means any objects closer than 2 meters from the camera will not have accurate disparity estimates. It is our hope that by creating a hardware stereo system based on this algorithm without significant loss of features, as compared to its software version, and at the same time providing frame-rate performance we can encourage computer scientists who develop stereo, and in general, vision algorithms, to steer their work towards hardware based algorithms. At the same time, we hope that we can also encourage hardware designers to break from a traditional approach of designing some common arithmetic components so that more of the complex algorithms have implementations on hardware.

2.2 Reconfigurable Systems

A reconfigurable system is a computing system that can be reprogrammed to perform many different tasks, often to support future upgrades and enhancements. Reconfigurable systems have at their core single or multiple interconnected FPGAs that act as the main processing unit of the system. FPGAs, by nature, are reconfigurable and thus provide this functionality to the system. In addition to the FPGA(s), a reconfigurable system may also provide various means of supplying input to the system, such as audio or video signals, and also the ability to capture the results of the system. The communication with other systems and peripherals are through interfaces such as Peripheral Component Interconnect (PCI) bus, Universal Serial Bus (USB), or even network-based.

2.2.1 Field Programmable Gate Arrays

An FPGA is an array of logic elements whose behaviour can be programmed by the end-user to perform a wide variety of logical functions, and which can be dynamically reconfigured as requirements change. FPGAs generally consist of four major components: 1) Logic blocks/elements (LB/LE); 2) I/O blocks; 3) Logic interconnect; and 4) dedicated hardware circuitry. The logic blocks of an FPGA can be configured to implement basic combinatorial logic (AND, OR, NOR, etc gates) or more complex sequential logic functions such as a microprocessor. The logic interconnect in an FPGA consists of wire segments of varying lengths which can be interconnected via electrically programmable switches. The density of logic blocks used in an FPGA depends on the length and number of wire segments used for routing.

The Altera Stratix S80 device used in this work contains a two-dimensional row- and column-based architecture to implement custom logic. The smallest unit of logic in the Stratix device is called a Logic Element (LE). Each LE contains a four-input LUT, which is a function generator that can implement any binary-valued function of four binary variables. In addition, each LE contains a programmable register and carry chain with carry select capability to facilitate fast implementation of arithmetic components. The architecture of the Stratix LE is shown in Figure 2.6. A set of 10 LEs make up a Logic Array Block (LAB) [?].

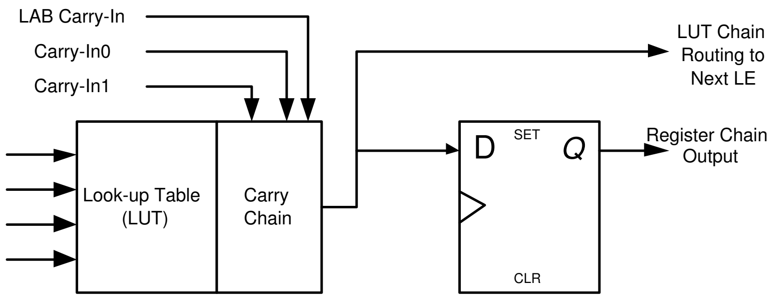


Figure 2.6: Simplified LUT architecture of Stratix chip.

Most modern FPGAs also have various dedicated circuitry in addition to the programmable logic. These come in the form of high-speed and high-bandwidth embedded memory, dedicated DSP blocks, Phase-Locked Loops (PLLs) for generating multiple clocks, and even general purpose processors. The FPGA we are using in our system, the Altera Stratix S80, comes with three different memory block sizes; 512 bits, 4 Kbits, and

512 Kbits for a maximum of 7 Mbits of embedded memory and 22 DSP blocks consisting of multipliers, adders, subtractors, accumulators, and pipeline registers. Figure 2.7 shows an overview of the Altera Stratix S80 chip [?].

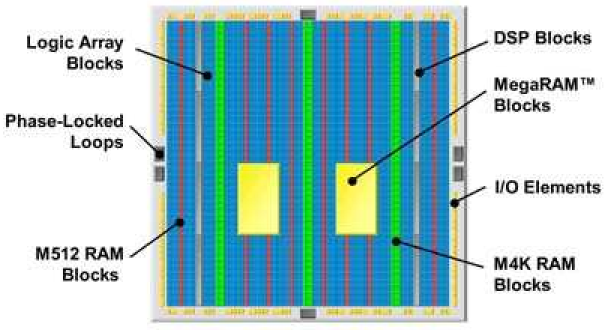


Figure 2.7: Advanced features of the Altera Stratix FPGA [?].

The combination of memory and DSP blocks make the Stratix family of FPGAs ideal for image processing applications as they can efficiently implement many common image processing tasks such as finite impulse response (FIR) filters, discrete cosine transform (DCT), colour space conversion (CSC), and MPEG-related operations among others. Each DSP block, illustrated in Figure 2.8, is optimised to interface with the specialised memory structures in Stratix devices for memory-intensive and high-precision DSP applications. Each DSP block can implement four 18×18-bit signed or unsigned multiplications using dedicated multiplier circuitry. The blocks can also be configured to support eight 9×9-bit multiplication or one 36×36-bit multiplication. In addition, the adder/subtractor/accumulator unit can switch between adder and subtractor functionality, acting as a 9-bit, 18-bit, or a 36-bit unit as necessary. In the accumulator mode, the unit acts as a 52-bit accumulator that can be used to implement operations such as convolutions.

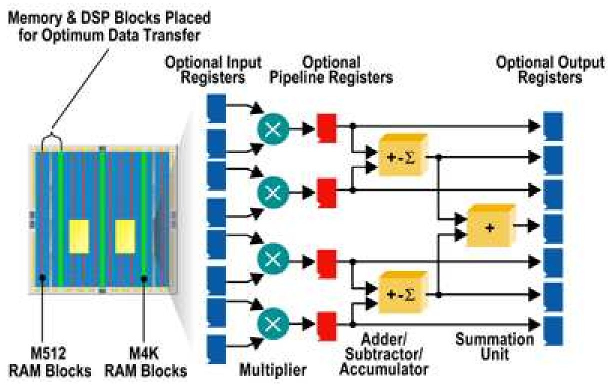

Memory & DSP Blocks Placed for Optimum Data Transfer
Optional Input Registers
Optional Pipeline Registers
Optional Output Registers
+-Σ
+
+-Σ
M512 RAM Blocks
M4K RAM Blocks
Multiplier
Adder/ Subtractor/ Accumulator
Summation Unit

Figure 2.8: DSP Block architecture of Altera Stratix S80 FPGA [?].

2.2.2 FPGA Design Options

FPGA designers have several options for implementing algorithms on the device. The circuitry can be designed by connecting logic gates to generate the desired output, a technique known as *gate-level design*. Gate-level designs result in optimised designs, but the learning curve is considered prohibitory for most engineers, and these designs also suffer from portability issues across FPGA architectures. A more common approach is to design at a higher level using Hardware Description Languages (HDLs). HDLs can efficiently describe the structure and behaviour of digital logic designs for creating ASICs or implementation on FPGAs.

HDLs provide support for describing concurrent event to take advantage of FPGAs' inherent ability to perform multiple operations concurrently. This feature differentiates HDLs from other high-level languages which are primarily intended for software design. HDLs also support inclusion of technology-specific modules, which come in the form of *cores* from FPGA or third-party vendors, for most efficient synthesis to FPGAs. IP (intellectual property) cores are generally parameterisable to suit the user's design and are often designed to provide optimised performance on a particular device or architecture. These cores allow IP to be easily distributed and help in speeding-up the design process.

A number of HDLs currently exist, some of which are proprietary, but two of the most widely used are Verilog HDL, or simply Verilog, and VHSIC (Very High Speed Integrated

Circuit) HDL, or simply VHDL. Verilog, a former proprietary language, was originally intended as a simulation language. It has since been updated to be used in synthesis of hardware designs and made into an open standard. VHDL, an open IEEE standard, first appeared in 1987 as IEEE standard 1076-87. VHDL is intended to support the design, verification, synthesis, and testing of hardware designs. A second update, IEEE 1076-93, came out in 1993, and it is currently being considered for a third update. The third update has a number of features specifically suited for implementation of arithmetic algorithms. These include explicit support for variable bit-width floating and fixed point operations, which helps a designer manage these operations in the design better.

The choice of which language to use is arbitrary and depends solely on the designer's familiarity with the language. Geographically, Verilog is more popular on the west-coast of North America, whereas VHDL is the preferred language on the east-coast of North America and in Europe. The language chosen in this work is VHDL and the design is fully compliant with the IEEE standard for VHDL.

2.2.3 Design Approach

A successful large-scale hardware design requires extensive simulation and verification before a designer can validate the circuit. Designing arithmetic components in hardware often requires that the data be converted to fixed-point format because a floating-point version of the algorithm may not fit on the device. Implementation of floating-point arithmetic on hardware takes up a lot more logic resources than fixed-point operations. However, there is a trade-off between the accuracy of the system and hardware resource usage: greater accuracy requires larger bit representation and therefore more logic resources on the device. In addition, the concurrent nature of hardware design means that the designer must also make sure that the data-flow in a pipelined design is synchronised with the clock-cycle it is intended for. This means that validation must be performed both at bit-true and cycle-true levels.

MATLAB was used extensively to facilitate the smooth flow of design and verification of bit-true results. Modifications and extensions to the LWPC algorithm were tested in MATLAB first because it is optimised for matrix operations and provides a platform for a fast implementation. The design approach is shown in Figure 2.9. The figure shows the interaction between the hardware and software phases of the design process. The software phase is on the left side and the hardware phase is on the right side of the diagram. The verification and validation phases are shown in the middle of the diagram.

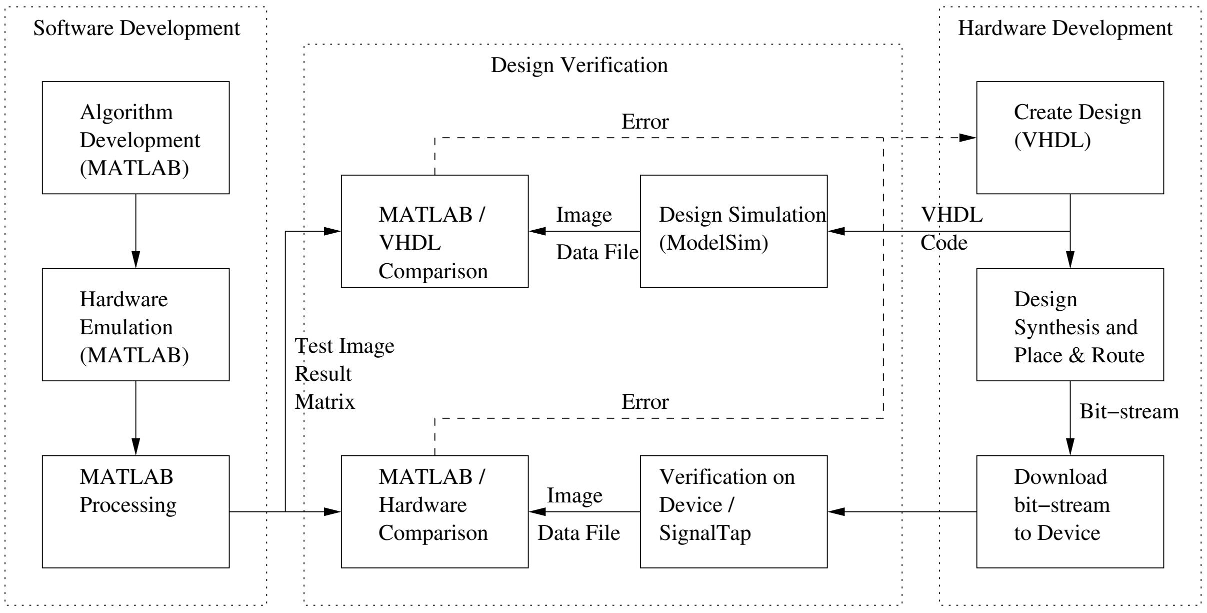

Software Development
Design Verification
Hardware Development
Algorithm Development (MATLAB)
Hardware Emulation (MATLAB)
MATLAB Processing
Error
MATLAB / VHDL Comparison
Image Data File
Design Simulation (ModelSim)
VHDL Code
Create Design (VHDL)
Design Synthesis and Place & Route
Bit−stream
Test Image Result Matrix
Error
MATLAB / Hardware Comparison
Image Data File
Verification on Device / SignalTap
Download bit−stream to Device

Figure 2.9: Hardware design approach taken in this work.

Once a satisfactory algorithm was developed, a "hardware emulation" version of the algorithm was coded in MATLAB to produce bit-true results for all components along the data-path. While a matrix-manipulation software program and a hardware implementation are fundamentally different, they should produce the exact same results, provided that care is taken in the hardware design. After completing the emulation version of the algorithm in MATLAB, the hardware design was created component by component. Each component is tested for functionality and cycle-true results using the ModelSim simulator tool. Bit-true results are checked by importing the results from ModelSim into

MATLAB and comparing with the results generated from the emulation version. When all components required for a module have been completed, the module is synthesised and a bit-stream is generated. The bit-stream is then downloaded onto the FPGA and the design is verified by providing it with the same input image pairs that were used for the emulation and ModelSim simulation versions. Any errors encountered in the simulation stage were debugged by analysing the signal waveform in ModelSim. Similarly, discrepancies found in the hardware were debugged using the SignalTap on-chip logic-analyser from Altera Corporation. The design was created and verified module by module, which were then all connected to create the complete system.

2.2.4 Transmogrifier-4 Reconfigurable System

The Transmogrifier-4 [?] is a general-purpose reconfigurable prototype board containing four Altera Stratix S80 FPGAs. The board has specific features to support image processing and computational vision algorithms; these include dual-channel NTSC and FireWire camera interfaces, video encoder/decoder chip, and 2GB of DDR RAM connected to each FPGA. Each FPGA is also connected to the other three FPGAs and an interface is provided to communicate with the board over a network. This can be used to send control signal or for debugging. The board is shown in Figure 2.10.

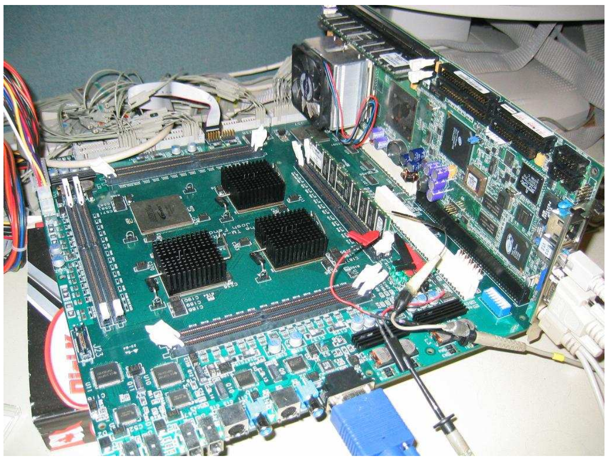


Figure 2.10: Transmogrifier-4 reconfigurable computing board.

2.3 Previous Work

A variety of reconfigurable stereo machines have been reported [?, ?, ?, ?, ?]. The PARTS reconfigurable computer [?] consists of a 4×4 array of mesh-connected FPGAs with a maximum total number of about 35,000 4-input LUTs. A stereo system was developed on PARTS hardware using the census transform, which mainly consists of bit-wise comparisons and additions [?]. Kanade *et al.*[?] describe a hybrid system using C40 digital signal processors together with programmable logic devices (PLDs, similar to FPGAs) mounted on boards in a VME-bus backplane. The system, which the authors do not claim to be reconfigurable, implements a sum-of-absolute-differences along predetermined epipolar geometry to generate 5-bit disparity estimates at frame-rate.

In Faugeras *et al.*[?], a 4×4 matrix of small FPGAs is used to perform the cross-correlation of two 256×256 images in 140 ms. In Hou *et al.*[?], a combination of FPGA and Digital Signal Processors (DSPs) is used to perform edge-based stereo vision. Their approach uses FPGAs to perform low level tasks like edge detection and uses DSPs for higher level integration tasks. In [?] a development system based on four Xilinx XCV2000E devices is used to implement a dense, multi-scale, multi-orientation, phase-correlation based stereo system that runs at 30 frames/second (fps).

It is worth noting that not all previous hardware approaches have been based on reconfigurable devices. In [?], a DSP-based stereo system performing rectification and area correlation, called the SRI Small Vision Module, is described. ASIC-based designs are reported in [?, ?, ?] and in [?] commodity graphics hardware is used.

Chapter 3

Design

In this chapter we first analyse a previous implementation of the LWPC based stereo algorithm to point out the shortcomings of the system. We then describe the design of pre- and post-processing modules in Section 3.2 and Section 3.4 respectively that improve the general accuracy of a stereo algorithm. The highlight of this work is the development of a novel architecture for performing the phase-based correlation which can support a very large disparity range without a corresponding increase in the logic resource requirements as described in Section 3.3. Finally, we show an architecture of a stereo-vision system that incorporates these modifications in Section 3.5.

3.1 Stereo Vision System on Transmogrifier-3

The previous stereo system based on the LWPC algorithm was implemented on the Transmogrifier-3 board. The TM-3 board has four Xilinx Virtex 2000E FPGAs. The system was a straight forward implementation of the LWPC algorithm described in Section 2.1.3. A high-level architecture of this system is shown in Figure 3.1.

Two key steps taken to successfully implement the algorithm on the TM-3 board were:

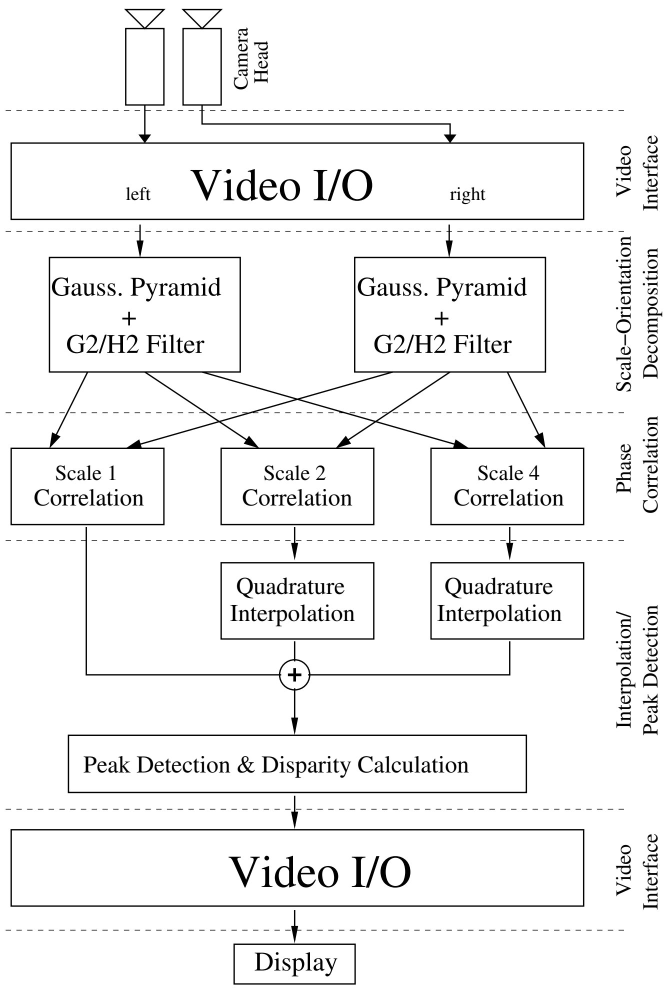


Figure 3.1: High-level architecture of LWPC based stereo-vision system on the TM-3 board.

1. Modification of the voting function (Step 2 of the LWPC algorithm) to share the normalisation operation and Gaussian filtering for all the pre-shifts τ. This reduces the number of multiplication, division, and addition operations by 65% so that the algorithm can be implemented in the FPGAs. A detailed look at this modification and its effects is given in Appendix B.

2. Conversion of the computation from floating-point to fixed-point.

The above-mentioned modifications are retained in the work. Even with these modifications, the previous system supported a disparity range of only 20 pixels. Moreover, the system was limited to using two orientations instead of the suggested three in [?].

It is important to note that the LWPC algorithm as described in [?] and its previous hardware implementation [?] compute the disparity of each frame from scratch at every frame. No attempt is made to use the fact that in most situations, the disparity of a pixel will not change drastically from one frame to the next when an image sequence is being captured at 30 frames per second. In this work, we successfully use this information and develop an architecture that is capable of handling very large disparities in the limited resources on the FPGA which we describe in Section 3.3. Section 3.2 describes the design of an image rectification unit which improves the accuracy of the disparity results, and Section 3.4 discusses the design of a post-processing unit to perform a consistency check in the disparity map.

3.2 Image Rectification Module

Rectification of an image requires the synthesis of a new image through warping of the original image. The theory of image rectification is described previously in Section 2.1.1. In practice, image rectification is achieved through an *inverse mapping* strategy—each pair of integer pixel coordinates in the rectified image is mapped to a pair of coordinates, not necessarily integer, in the original image. The pixel value for the warped image can be found through bilinear interpolation of the pixel values in the neighbourhood of the mapped coordinates.

There are two main issues to be aware of when implementing image rectification in hardware. First, since this is an inverse mapping process, we do not have prior information on what exact input is required at each time instance. This means some sort of buffering of the input image is required. In addition, bilinear interpolation requires four input pixels for each output pixel, whereas only one pixel can typically be read out at a time from the buffer. In order to keep up with the rate of incoming pixels one solution would be to store four copies of the incoming pixel stream in separate buffers. This allows all four pixels, one from each of the buffers, to be read at the same time. The incoming pixel rate for our system is about 13.5 MHz which is significantly lower than the full potential of the Stratix S80 FPGA. This gives us another option to achieve real-time performance; by designing a *multi-clock* system. In a multi-clock design, each hardware module is clocked by one of the two or more clocks used in the system which makes it possible to achieve a high-performance design. In this work, the image rectification unit is designed to run on a clock that is set to at least four times the frequency of the camera clock, so that the four pixels needed for bilinear interpolation can be read and processed in the time it takes for a single new pixel to arrive from the camera.

The second issue is efficient implementation of the inverse warping operation to compute the source pixel coordinate in the original image. The inverse warping operation (Equation 2.3) requires a matrix multiplication and two scalar division operations to conform to homogeneous coordinates and is expensive to implement in hardware. Instead, we have modeled this with a second-order polynomial which approximates the inverse homography matrix as follows:

$$\begin{aligned} x &= a_0 + a_1 x^{'} + a_2 y^{'} + a_3 x^{'2} + a_4 x^{'} y^{'} + a_5 y^{'2} \\ y &= b_0 + b_1 x^{'} + b_2 y^{'} + b_3 x^{'2} + b_4 x^{'} y^{'} + b_5 y^{'2} \ , \end{aligned} \qquad (3.1)$$

where x and y are real-valued source image coordinates and x' and y' are integer-valued coordinates of the rectified image. The coefficients a_i and b_i are computed offline.

The integer parts of x and y are used as the index of the source pixel for bilinear interpolation. The fractional parts are used as weights for bilinear interpolation. The values for the image coordinates, x and y, can be obtained by one of the following two approaches:

1. Look-Up Table - The image coordinates can be calculated in *advance* by evaluating the polynomials and storing the results in look-up tables which can be referenced

at run-time. Only one bilinear interpolation for each pixel needs to be carried out at run-time.

2. Real Time Computation - The image coordinates are computed by evaluating the polynomials at run-time and then bilinear interpolation is performed to determine the intensity of each pixel.

For an image that is 640 pixels wide, at least 10 bits are needed to index these pixels. The fractional part is represented by 6 bits bringing the total to a 16-bit representation. The Look-Up Table approach would require 600 KB of memory for storing each of the four coordinates for a total of 2400 KB to rectify both the left and right images. The Stratix S80 FPGA has 9 blocks of 64 KB of on-chip memory for a total of 576 KB. This on-chip memory is insufficient for the size of this problem. The off-chip memory on the TM-4 board can be used but it would require the design of a cache on the FPGA because it is not possible to read a value from the off-chip memory in a single clock cycle. In comparison, real-time computation of the polynomial requires 13 multiplication and 10 addition operations. We have chosen to compute these coordinates in real-time.

A fixed-point representation is employed for computing Equation 3.1. Figure 3.2 (a) shows the expected orientation of the warped image. The warped image is represented by the gray area. Figure 3.2 (b), (c), (d), and (e) show the resulting warped orientations obtained when the fractional part of the coefficients in Equation 3.1 is represented by 4, 8, 16, and 32 bits respectively. The root-mean-square error (RMSE) based on the resulting orientation is shown in Figure 3.3. There is no significant improvement in the RMSE when representing the coefficients with more than 16 bits so we chose this as the precision for the the coefficients a_i and b_i.

The architecture of the **Image Rectification Unit** is illustrated in Figure 3.4. A stereo-setup with a worst-case vertical misalignment of 16 *scanlines* between the left and right image is assumed, which requires buffering of 32 *scanlines* of both the left and right images. The *Image Buffer* stores the incoming pixels and the *Controller* keeps

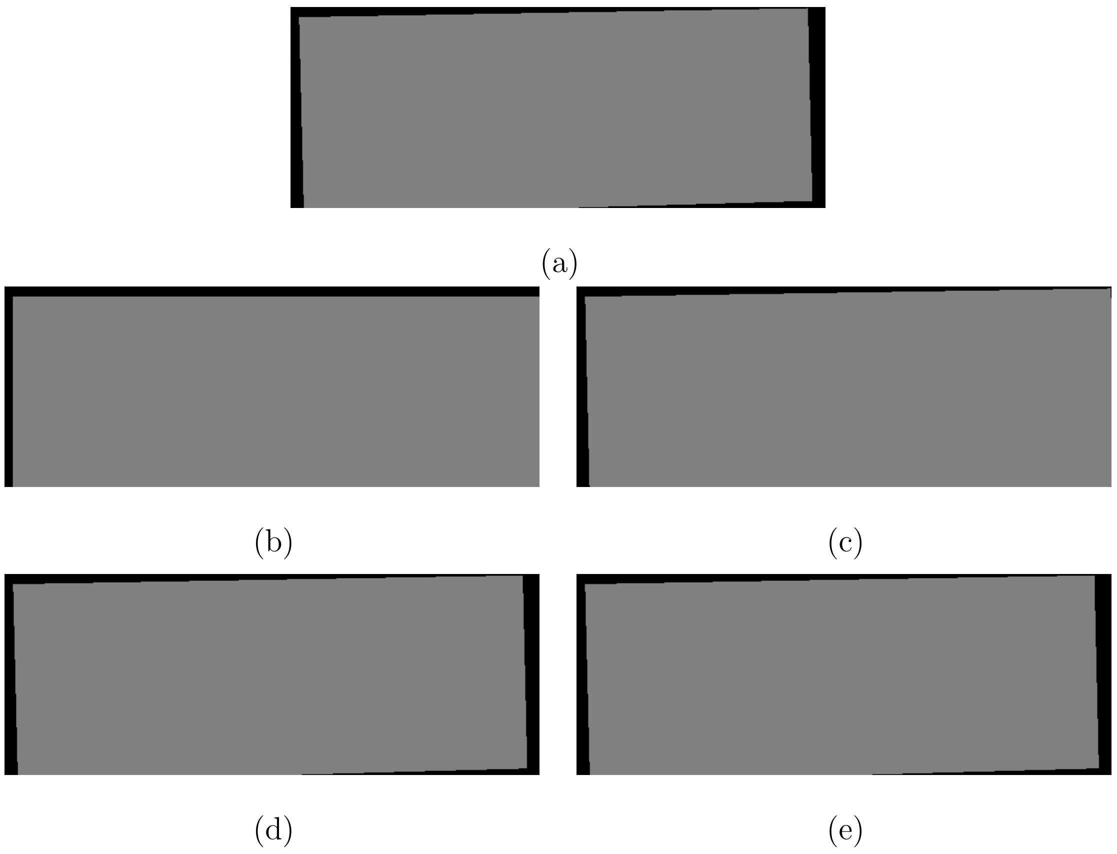


Figure 3.2: Comparison of the expected warped image (a) with the resulting warped images when computing Equation 3.1 with 4 (b), 8 (c), 16 (d), and 32 (e) bits of precision for coefficients a_i and b_i.

track of the latest source line that arrives into the buffer. If all the lines required for the next output line are available, the *Controller* generates the output indices for pixels in the scanline. The *Address Generator* then computes the source address for the four neighbouring pixels required for bilinear interpolation for each pixel in the scanline. The integer part of the source address is used to read out the required pixels from the buffer. The fractional parts of the source address are used as weights for bilinear interpolation. A warped image is not in general contained in the same region of the image plane as the original image resulting in some "missing" pixels typically at the edges of the warped image. A 1-bit flag is generated to indicate these missing or invalid pixels.

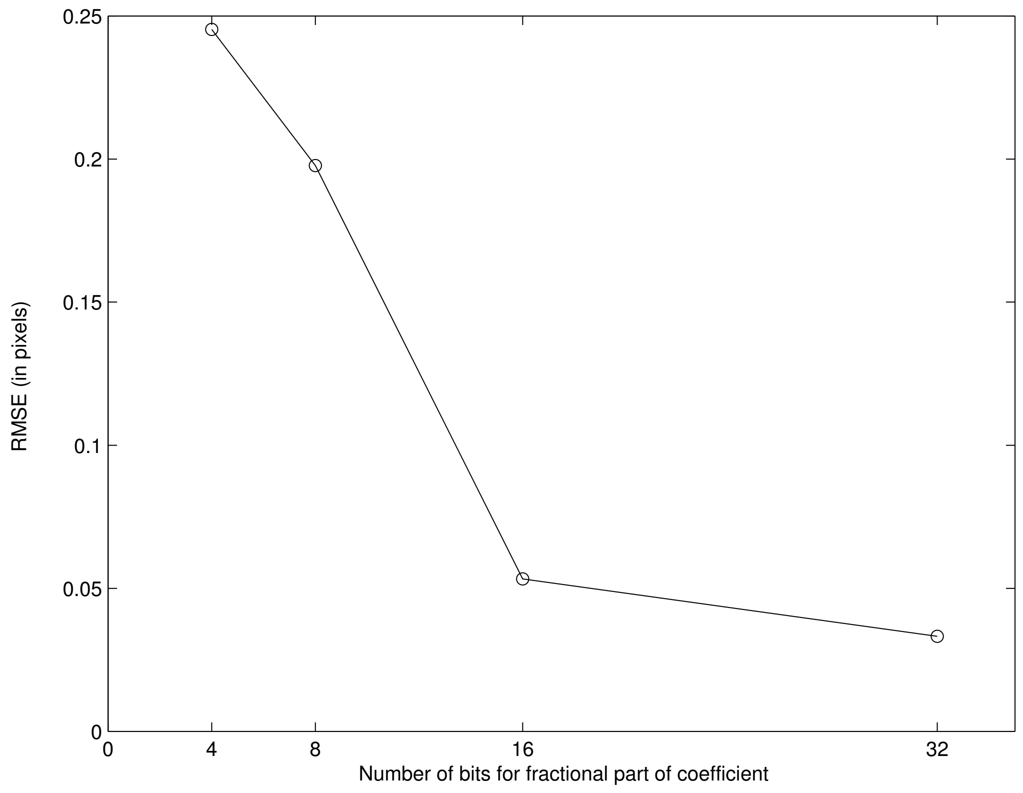


Figure 3.3: Root-mean-square error values for the warped image orientations computed with various precision bits for the coefficients a_i and b_i.

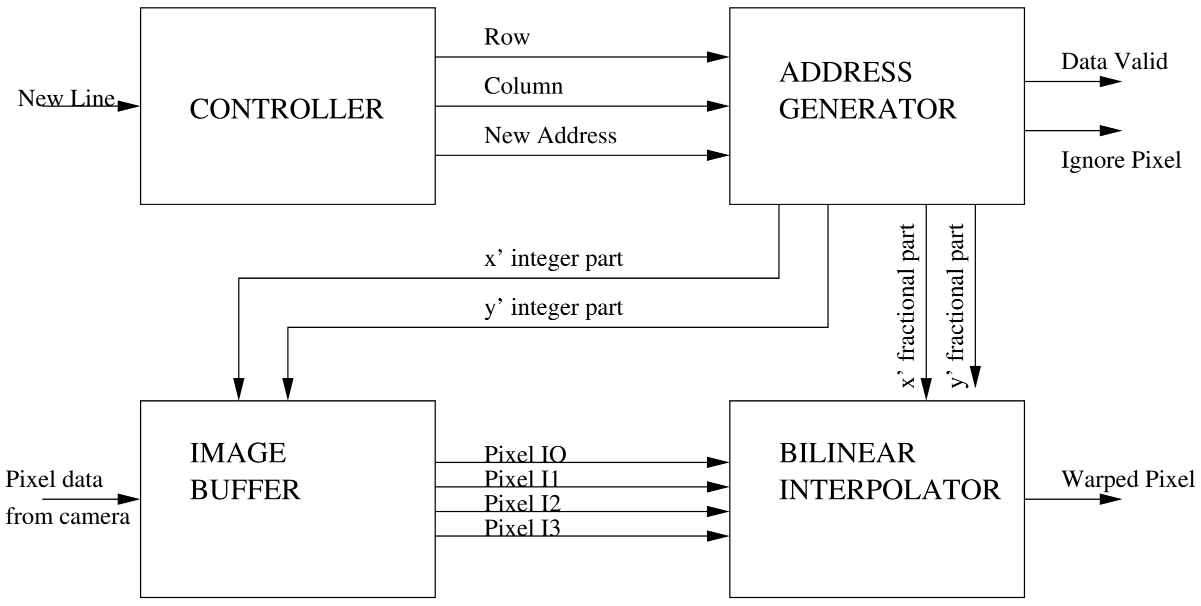


Figure 3.4: Architecture of Image Rectification Unit.

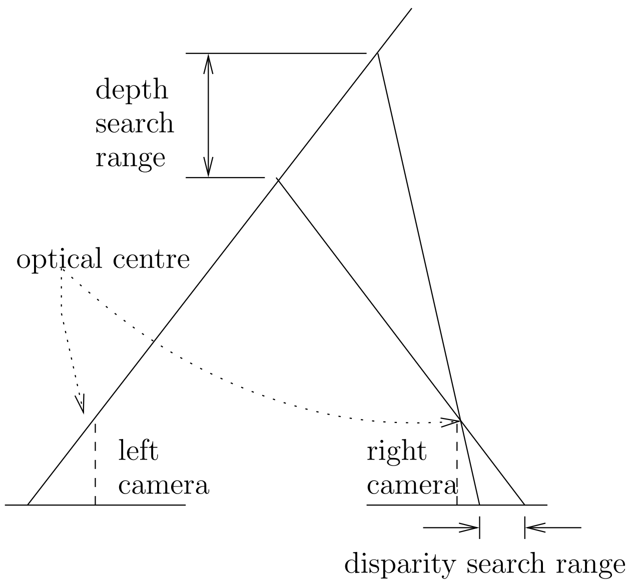


Figure 3.5: There is a direct relationship between the epipolar search band and the expected depth of objects in a scene that can be recovered.

3.3 Expanding the Disparity Range

A simple and straightforward solution to expanding the disparity range can be achieved by increasing the size of the correlation window to correlate pixels at greater disparities. This increases the size of the *epipolar search band*, which is the area along the image scanline within which a corresponding match is searched for. There is a direct relationship between the search band range and the depth in a scene that can be recovered, as illustrated in Figure 3.5. However, considering that the resource usage on the FPGAs is proportional to the size of the correlation window, it is not an optimal solution because finite device resources pose a restriction.

In addition, in [?] it is shown that the probability of an incorrect stereo match P_m^{Total} is given by :

$$P_m^{Total} = P_m^a + P_m^b + P_m^c \tag{3.2}$$

where P_m^a is the probability of mismatching a pair of features when neither feature has its correct match detected in the other image, P_m^b is the probability of mismatch when one feature has had its correct match detected and P_m^c is the probability of mismatch when both features have had their correct matches found in the other image. P_m^a, P_m^b and P_m^c are each proportional to the mean number of candidate matches, and mutually exclusive, and thus P_m^{Total} is proportional to the epipolar search area considered during matching.

3.3.1 The Tracking Correlation Window

We saw above that increasing the size of the correlation window is not a viable solution from either an algorithmic point of view or a hardware implementation point of view. However, in many practical situations, such as when an object gets closer to the cameras, it becomes necessary to be able to handle a larger disparity range.

Fortunately, the input to real-time stereo-vision systems is from cameras that stream images at a rate of 30 fps or higher. At this rate, a large amount of temporal coherence is expected in most real-life image sequences. By modeling and predicting the movement of pixels in an image, we can restrict the epipolar search to a limited area at a particular time frame. The correlation window can shift accordingly along the epipolar line (which for the case of rectified images is the same as a *scanline*) and perform localised correlation rather than performing a blind search over a much larger range, much of which has a very low probability of having the match of interest.

Keeping this in mind, we have modified the original LWPC algorithm to encapsulate the correlation algorithm within a temporal loop. This change is reflected in Step 2 of the LWPC algorithm initially mentioned in Section 2.1.3 as follows:

2. For each scale and orientation, compute local voting functions $C_{j,s}(x,\tau)$ in a window centred at τ_c as

$$C_{j,s}(x,\tau) = \frac{W(x) \otimes [O_l(x) O_r^*(x+\tau_c)]}{\sqrt{W(x) \otimes |O_l(x)|^2}\sqrt{W(x) \otimes |O_r(x)|^2}}\ , \tag{3.3}$$

   where $W(x)$ is a smoothing, localized window and τ_c is the pre-shift of the right filter output centred at the disparity of the pixel from the previous frame.

This is implemented as the *Primary Tracking Window* (*PTW*) and its hardware implementation is discussed in Section 3.3.3. The tracking algorithm is currently a very simple one; the window is centred at the estimated disparity from the previous frame for a given pixel. Since images are received at a rate of 30 frames per second or higher, we assume that the disparity of a given pixel will not change drastically from one frame to the next. This allows us to reduce the size of the correlation search area, which results in reduced hardware resources as well as a reduced probability of mismatch assuming we are close to the correct match.

In correlation-based stereo algorithms such as this one the recovered object boundaries tend to be located away from the real ones, a problem known as *boundary over-reach*. Boundary over-reach occurs when the correlation window straddles an object boundary so that part of the window is on an object at one depth and part of it is on an object at another depth. The system in this work also exhibits this phenomenon, but the shiftable nature of the window can be used to overcome this problem in a future version of the system as will be discussed briefly in Section 3.3.5.

3.3.2 System Initialisation

When propagating disparity estimates between frames, it is necessary to consider that such algorithms suffer from the risk of getting stuck in a local minima (wrong matches) [?], especially during the initial frames. This problem can be overcome by performing system initialisation using one of the following two methods:

1. Coarse-to-fine strategy.

2. Stochastic search strategy.

The idea behind a coarse-to-fine strategy is to gradually increase the resolution at which correlation is performed. This performs the initial exhaustive search that is required to give a temporal stereo algorithm a good seed point from which to start. Processing the initial few frames at a much coarser scale (say 25% or 50% of their original size) allows the correlation algorithm to search a greater range of disparity in a given time. The main advantage of using a coarse-to-fine strategy is that a valid disparity map is generally available from the first frame. This approach is ideal for a software implementation because it does not increase the computational load. In a hardware implementation though, this approach is less appealing because it requires decision making regarding which scale to use at a specific time, though its potential uses can be considered for a future addition to the system. Furthermore, using results from only a particular scale would mean losing the essence of the LWPC algorithm, which combines results across scales for improved accuracy. This is discussed later in the section.

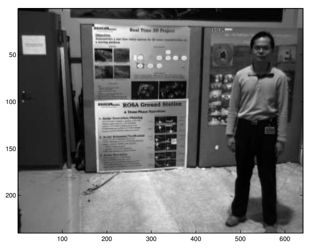
Stochastic search applies an exhaustive search over a period of time. It normally takes a few frames before complete initialisation of the scene under consideration is achieved. A second correlation window, in addition to the primary window that performs temporal correlation, is required to perform this search. In a software implementation, this means an increase in the computational load. In hardware, the secondary window can operate in parallel with the primary window so that there is no increase in the computation time. We chose to employ this method because this secondary window can also be used after the initialisation stage to help the system recover from mis-matches as discussed below. This is an advantage over the coarse-to-fine strategy. A coarse-to-fine strategy would require frequent reinitialisation and computation of the entire scence at coarser scale to recover from mis-matches whereas the stochastic search approach treats individual pixels separately and is able to recover from a mis-match without switching the computation between coarse and fine scales.

In [?] it is claimed that a coarse-to-fine strategy is preferred over an initialisation stage that uses a window to incrementally search over a wider range, but from our experiments on real image sequences we have found that a secondary correlation window that performs disparity calculations at regularly spaced intervals in successive frames, similar to the initialisation stage employed in [?], gives good results, as will be shown.

Figure 3.6 shows the results from the initialisation stage for the scene shown in Figure 3.6(a). Figure 3.6 (b) shows the disparity map that the LWPC algorithm would generate if it had no limitations on the maximum disparity. Figure 3.6 (c) to (g) show the settling of the disparity map into the expected disparity of Figure 3.6(b). The Root Mean Square Error (RMSE) between the expected disparity map and the disparity map generated by our modified algorithm steadily decreases at each progressive frames and tends to zero at the fifth frame for this particular scence as shown in Figure 3.6 (h).

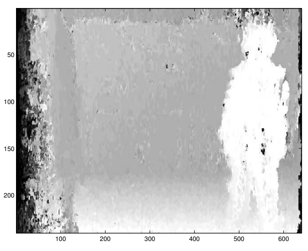
We call this second correlation window the *Secondary Roving Window* (*SRW*). The main advantage of using this approach is that the *SRW* also aids in recovery from a mismatch *after* the initialisation stage. In situations where a new object enters the scene, or a region is dis-occluded, the *SRW* will pick up this new information, typically within a few frames, and provide a disparity estimate with higher confidence value than the PTW, which can then latch on to this new estimate as illustrated in Figure 3.7. This provides better results as well as much better utilisation of the hardware resources.

There is a trade-off between the stochastic search area and the time to recovery using this approach. The further that the *SRW* has to search over, the greater number of frames it would require to recover should the PTW be stuck at a wrong match. In Figure 3.8 we show the difference in recovery time for the cases when the secondary correlation window is shifted up to a disparity of: i) 140 pixels and ii) 60 pixels. Figure 3.8 (a) shows frame 11 for case (i); the results start to deteriorate as the subject moves to the left and pixels to the right of the subject are dis-occluded. The system completely recovers by frame 15, Figure 3.8 (b). For case (ii), the results deteriorate at frame 12, Figure 3.8 (c), and are already recovered by frame 13, Figure 3.8 (d), though the results get noisier as the maximum disparity of the system increases. When the maximum disparity expected



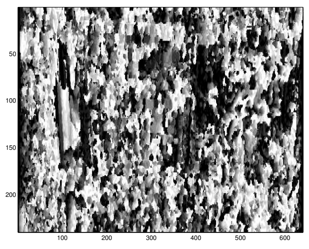
(a)

Original image



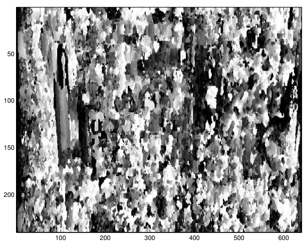
(b)

Expected disparity map



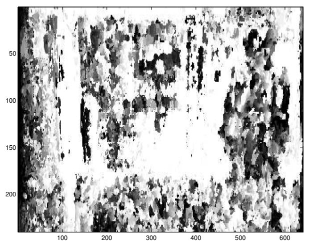
(c)

t1



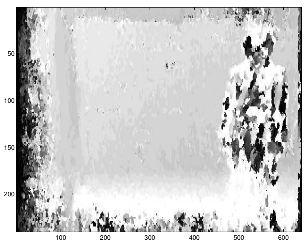
(d)

t2



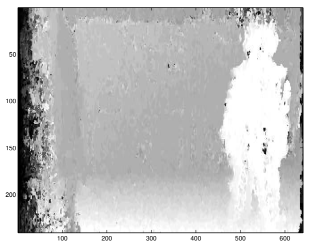
(e)

t3



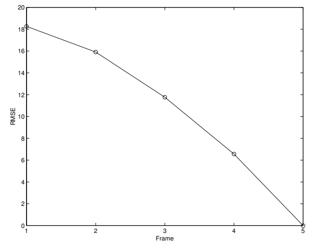
(f)

t4



(g)

t5



(h)

RMSE error

Figure 3.6: The disparity map at successive time instances are shown in (c) to (g) obtained using our modified temporal algorithm. The expected disparity map is shown in (b) and our modified algorithm reaches this expected disparity by the fifth frame (g) for this particular set-up. The error during the initialisation stage shown in (d) converges to zero.

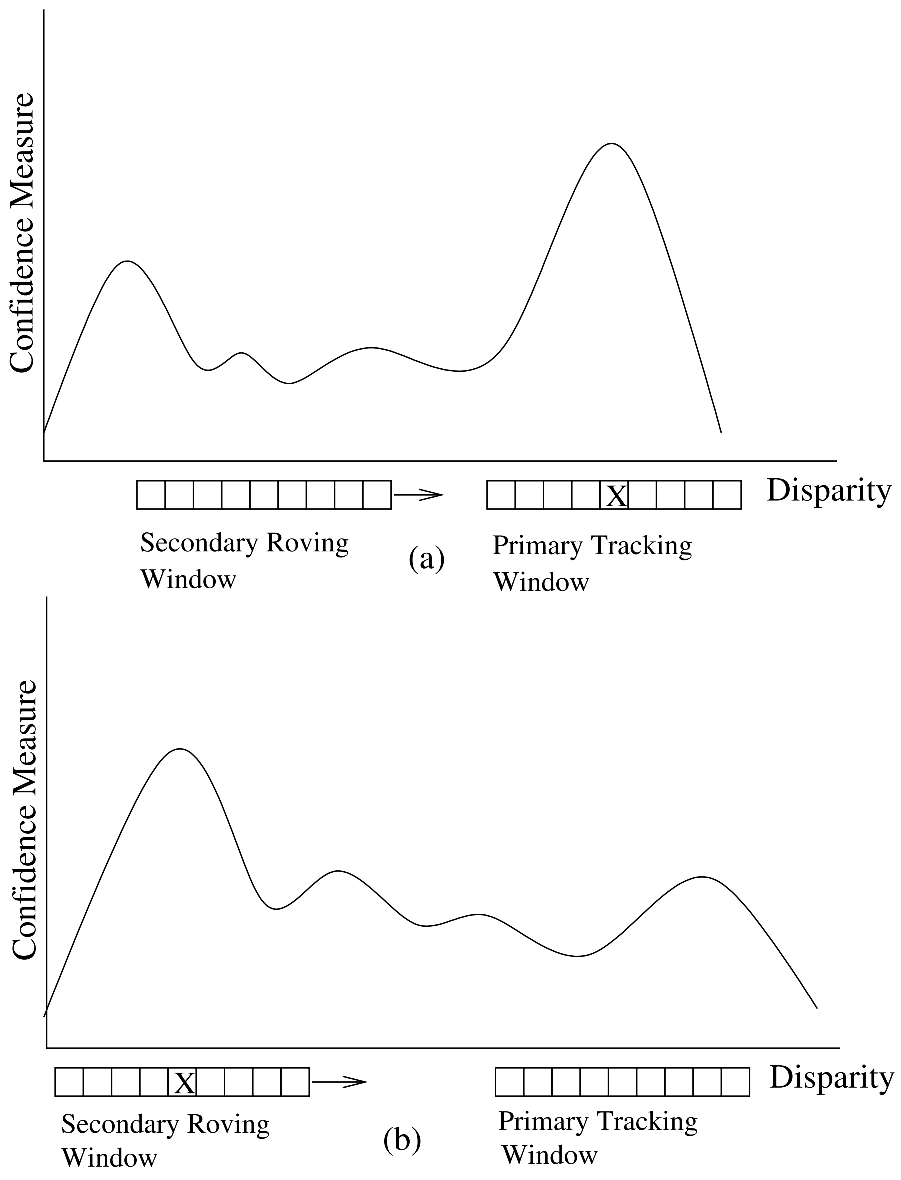
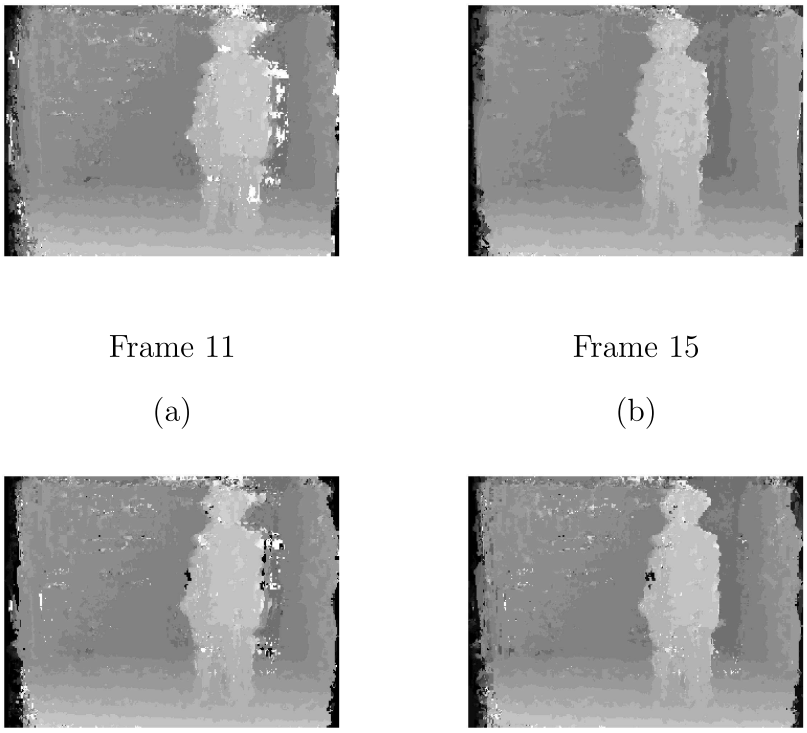


Figure 3.7: PTW is correctly tracking the peak (denoted by an **X**) in the confidence measure in (a). In (b), PTW has lost track of the peak, but SRW has picked it up. PTW will latch on to this estimate at the next frame.



Frame 11

(a)

Frame 15

(b)

Frame 12

(c)

Frame 13

(d)

Figure 3.8: The recovery time for the system with a maximum secondary shift of 140 pixels is shown in (a) and (b). This can be reduced by using a smaller maximum shift, *e.g.* 60 pixels as shown in (c) and (d). In the latter case, recovery occurs in one frame as opposed to four.

is known in a particular environment, the roving distance of the *SRW* can be restricted to minimise the recovery time and noise in the disparity map.

Furthermore, performing disparity calculations at all three scales$(1, 2, 4)$ and in three orientations (-45^o, 0^o, $+45^o$), the results of which are summed across scale and orientation, acts as a built-in error-correction feature of the LWPC algorithm. The expected interval between false peaks is approximately the wavelength of the filters applied at each scale. Thus the false peaks at different scales occur at different disparities and summation over the scales yields a prominent peak only at the true disparity [**?**] as shown in Figure 3.9.

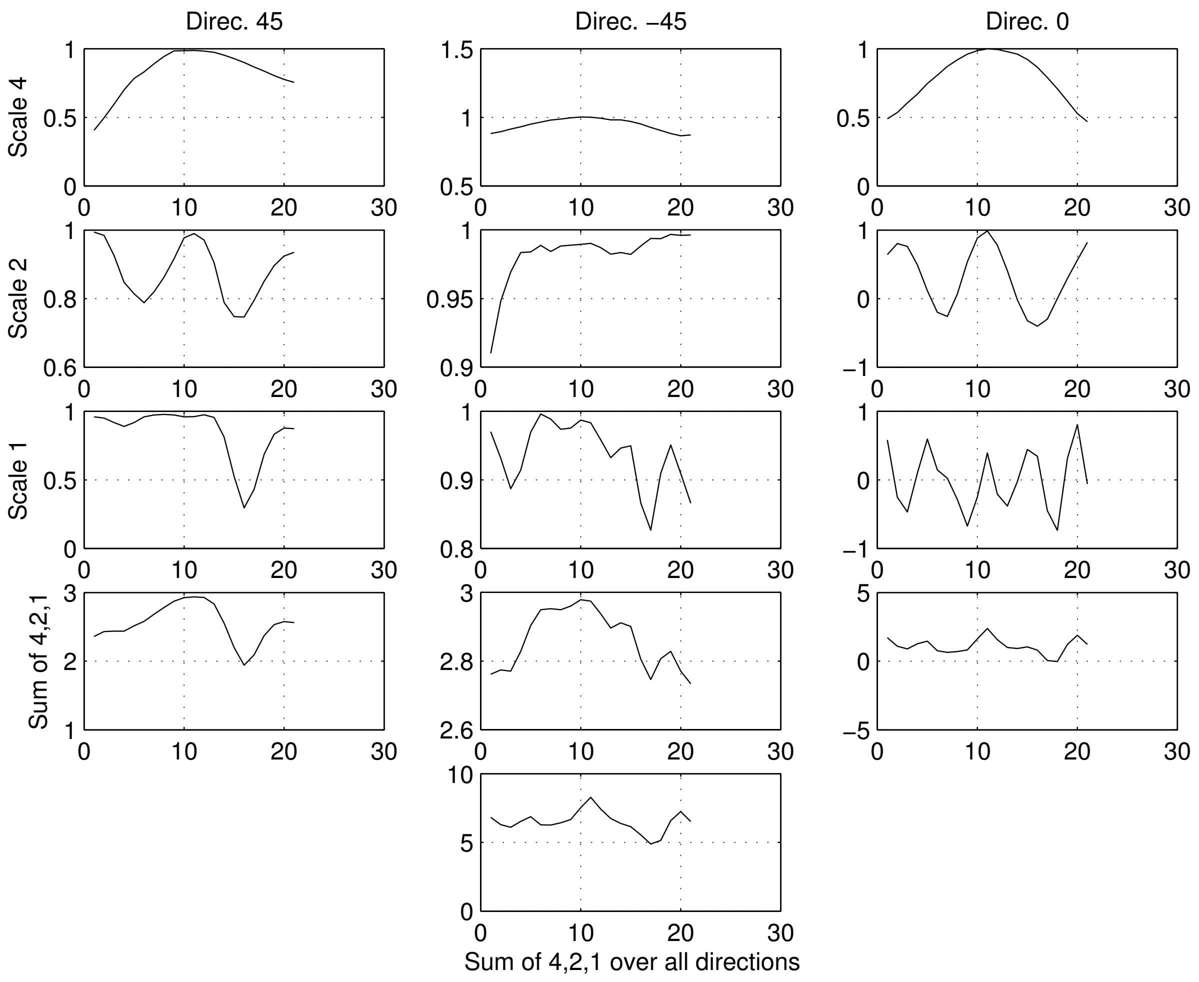


Figure 3.9: LWPC tends to cancel erroneous matches across scales, leaving only the true match after voting. The top three rows show the voting function for three different scales at orientations of 45^o, -45^o, and 0^o. The fourth row shows the voting function summed across the scales for each orientation and the final summation across orientations is shown in the fifth row.

3.3.3 Comparison of the traditional and new architecture

The use of temporal information to seed the correlation windows increases the epipolar search area while keeping the probability of mismatches and the use of hardware logic to a minimum. Nonetheless, it is a challenging task to implement the *shiftable window correlation* in hardware. FPGAs are well suited for processing data in a serial-shift or *systolic* dataflow fashion. The traditional textbook approach to designing a correlation unit, one which was followed in the the stereo-system on the TM-3 board, follow a

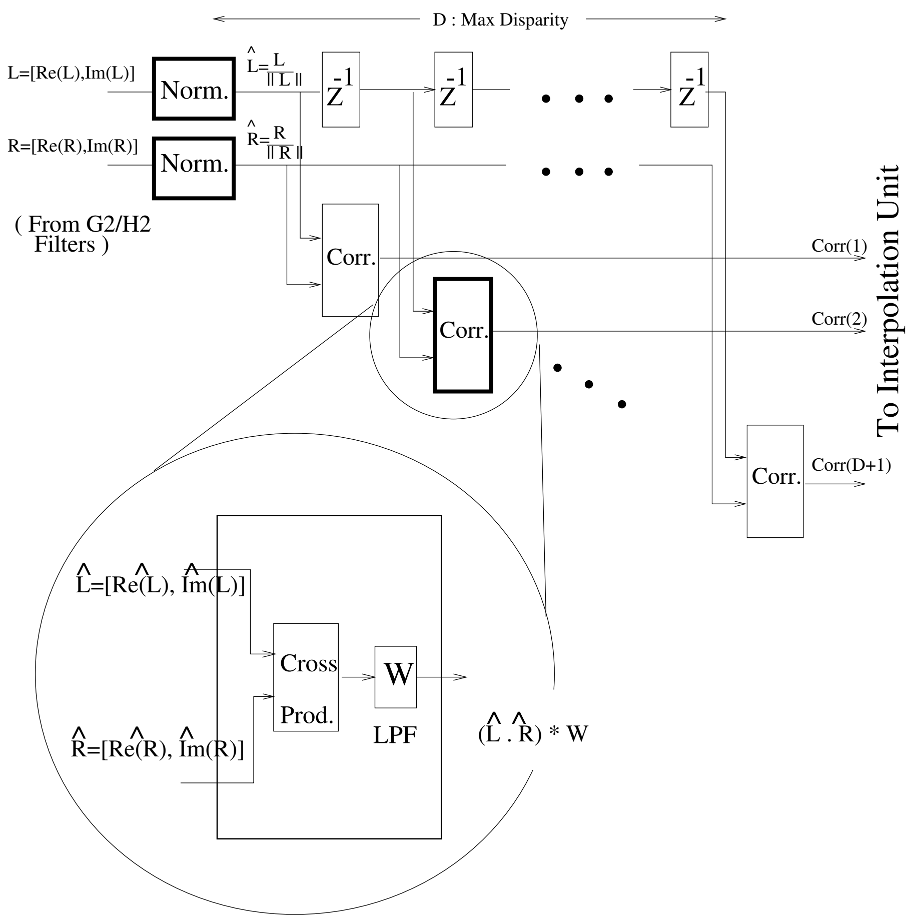


Figure 3.10: Traditional architecture of the correlation unit with fixed window.

serial-shift dataflow and this only allows for a fixed-window correlation architecture. This architecture, used in [?], is illustrated in Figure 3.10. The correlation, $W(x) \otimes [O_l(x)O_r^*(x+\tau)]$, is performed by supplying the left image input of the unit with a new pixel value every cycle and delaying the right image input by 1 to D cycles for each of the D correlation values. For $D = 20$, this results in a latency of 20 cycles for the first correlation result to appear, after which a new correlation value is generated every cycle.

In the traditional fixed-window correlation architecture the maximum disparity is equal to the actual number of *voting function / correlation* blocks used. The resource usage is linearly proportional to the maximum disparity that the correlation unit can support making it prohibitive to use in scenarios with large disparities.

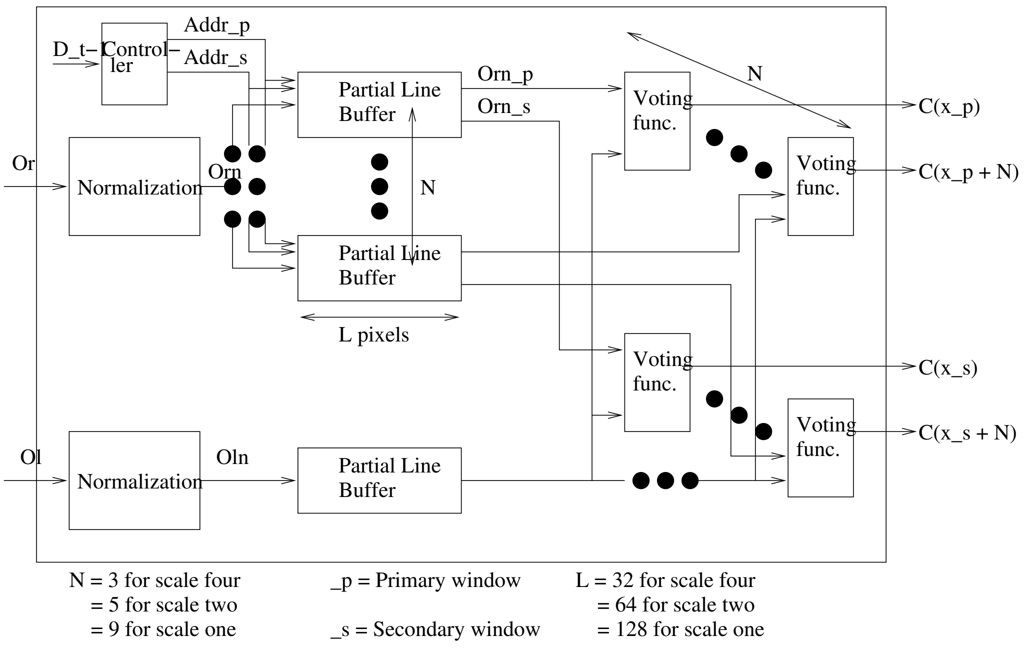


Figure 3.11: Modified correlation unit with two shiftable windows.

To efficiently implement the shiftable window correlation architecture that was discussed previously, we need to take into account the resources available on the specific hardware we intend to use. In the modified correlation algorithm, the data no longer flows in a serial-shift fashion. We do not have *a priori* information on where the window will be located at a particular time instance but we can set the maximum distance over which the correlation might be performed, which we have set to 128 for this work. This requires buffering of partial lines corresponding to the maximum search range.

The correlation is carried out at three scales for both the *PTW* and *SRW*. We have set the width of the correlation window to 9 pixels at the original scale, requiring nine voting function units. Five voting function units at 50% scale (Scale 2) and three voting function units at 25% scale (Scale 4) are required. We need to buffer n copies of one input, where n corresponds to the number of voting function units at each scale, in order to keep up with the rate of incoming pixels. High-bandwidth M4K memory blocks of the Altera Stratix S80 chip are used to buffer these incoming pixels.

In our modified architecture (Figure 3.11), one of the incoming data streams is stored in partial line buffers; the right input stream for a left-to-right correlation and the left input stream for a right-to-left correlation. A left-to-right correlation treats the left input as the reference and the right input is shifted to search for the best match, and vice-versa for a right-to-left correlation. The maximum disparity is determined by the size of these buffers whereas the number of voting function units are fixed; 9 for PTW and 9 for *SRW*. Once all the required pixels for the next correlation operation are available in the buffer, the controller generates an address to read out the required pixel for the current correlation window. To minimize the resource usage on the FPGA, the partial line buffers are implemented using true dual-port memories so that two pixel values can be independently read from the buffer each cycle, one for each of the correlation windows.

The *SRW* centres at a disparity of 9 for the first frame, and shifts in increments of $L = 9$, the correlation window length, for the successive frames until it reaches a disparity value of 128, or some other user-specified maximum. At the next frame, the window centres at 0 disparity, after which it circles back to being centred at 9 and the cycle continues. The effective range of disparity that our system can handle is 128 pixels but this can easily be increased to accommodate larger disparity. There is a tradeoff between the time to recovery from a mismatch and the maximum disparity that the system can handle, as discussed earlier. For a maximum disparity of 128 pixels with increments of 9 pixels per frame for the *SRW*, the worst-case time to recovery is 433 milliseconds corresponding to a wait of 13 frames.

A comparison of the relative resource requirements for a traditional architecture against our modified architecture is shown in Table 3.1. These are calculated for performing the correlation in three orientations at three scales. The number of *normalisation units* remains the same in both architectures. The number of *voting function units*, the core of the correlation module, required for the modified architecture is actually less than that required for a traditional architecture (102 v.s. 105) for a significantly larger

Unit	**# of Blocks**	**Altera LEs**	**On-chip M4K RAM**	**9-bit DSP Elements**
Normalisation	18 (18)	29,070	-	72
Partial Row buffers	60 (0)	-	120	-
Voting function	102 (105)	21,726	-	-
Controller	1 (0)	$\leq$ 1,000	-	-
Available resouces on S80		79,040	364	172

Table 3.1: Resource consumption of modified Phase Correlation Unit on Altera S80 FPGA. The number of corresponding blocks required for the architecture described in [?] are shown in parentheses for comparison purposes.

support of disparity (128 v.s. 20). In fact, the number of *voting function units* remains the same no matter what the maximum disparity is set to. This represents a significant savings in resource usage and opens up a wide variety of uses for the modified correlation architecture. Our modified architecture requires a significant amount of on-chip memory for buffering and ways to achieve the same or better disparity estimates with reduced on-chip memory usage are discussed in Section 5.1.

3.3.4 Architecture Limitations

The original LWPC algorithm uses a 5×5 pixel 2-D Gaussian mask to compute the voting function $C(x, \tau)$ given in Equation 3.3, but in this work $C(x, \tau)$ is computed using a 1×5 pixel 1-D Gaussian mask. The accuracy of correlation based stereo-matching algorithms is inherently dependent on the amount of texture available in the image pair. The use of a 1-D Gaussian mask reduces the texture information available for correlation

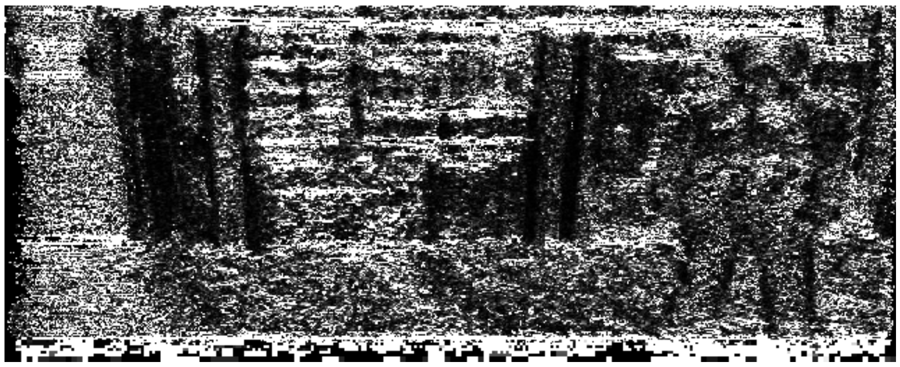


Figure 3.12: Pixels where the disparity estimate using a 1-D Gaussian mask differ from the disparity estimate using a 2-D Gaussian mask are shown in white.

as the texture information in the vertical direction is ignored causing some degradation in matching performance.

In our implementation, the amount of on-chip memory is not sufficient to buffer multiple incoming scanlines needed for a 2-D mask, as the architecture requires multiple copies of each line. The impact of the use of a 1-D mask is shown in Figure 3.12 for a sample frame. The white pixels indicate where the disparity estimates differ between the use of of 1-D mask and a 2-D mask. Though most of these estimates differ by a single value of disparity (some differ by over 20 disparity values), the difference is compounded when used in a temporal algorithm such as ours and we expect some degradation in the results. A difference of one disparity value per frame can cause the tracking window to go off-track over time. The architecture can be modified in a future version to accommodate a 2-D Gaussian mask as discussed in Section 5.1.

3.3.5 Flexibility of Modified Correlation Unit

A number of variations of the design can be achieved without having to make any changes to the correlation unit. Instead of the simple tracking algorithm that we are currently using for the *PTW*, an algorithm based on a constant-velocity motion model can be used to achieve better tracking. The velocity estimate can be obtained by taking the difference between disparities in the previous two frames, $v_t = d_{t-1} - d_{t-2}$, where v_t is the predicted disparity velocity for the current frame. Similarly, the location of the secondary window can be computed using a probabilistic likelihood estimate instead of the pre-determined roving locations.

Other options include the possibility of concatenating the two correlation windows after the initialisation stage so as to support greater movement of objects from one frame to the next. The decision of when to concatenate the windows and when to use them individually in parallel can be made by a simple count of the number of invalid disparity estimates after the validation check phase. This can be done for the whole image, region by region, or even for individual pixels. The issue of boundary overreach in correlation based algorithms [?] discussed earlier in Section 3.3.1 can also be solved by simply shifting the correlation windows by $\pm L/2$, where L is the length of the correlation window, so that the window does not cross over an object boundary. All of these modifications require the implementation of a post-processing stage that generates the appropriate input parameters for the correlation unit without having to make internal changes to the correlation unit itself.

The use of the correlation unit is not limited to a stereo-system. It can also be used in other systems such as object recognition using template matching, for *e.g.*, appearance models for object recognition. The two correlation windows can be used independently to search different regions of an image thereby speeding up the search process or they can be combined to support a larger template.

3.4 Consistency Check Unit

Dense stereo disparity algorithms based on correlation such as the LWPC algorithm used in this work generate a disparity value for every single pixel. When the two images are taken from different viewpoints there invariably are pixels that are visible in one image

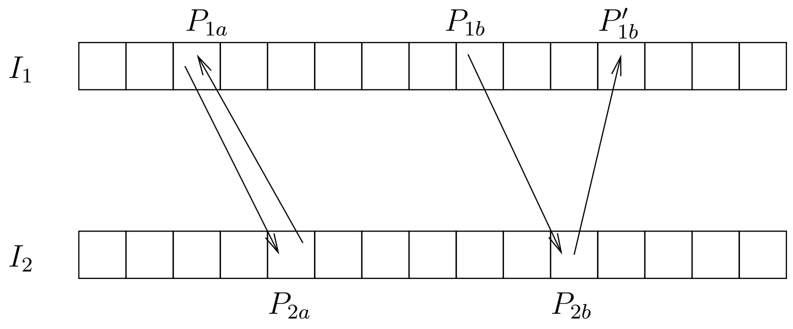


Figure 3.13: The two rows represent pixels along two epipolar lines of I_1 and I_2 and the arrows go from a point in one of the images towards the point in the other image that maximises the correlation score. The match on the left is consistent because correlation from I_1 to I_2 and from I_2 to I_1 yields the same match, unlike the matches on the right that are inconsistent.

but not in the other, a condition known as *occlusion*. The matching algorithm will still attempt a best guess as an estimate for the disparity resulting in erroneous results. We have employed a mechanism called a *left-right, right-left (LR-RL) consistency check* to identify these and other irregularities in the disparity estimates. This validation measure is illustrated in Figure 3.13 and can be defined as follows:

> Given a point P_1 in I_1, let P_2 be the point of I_2 located on the epipolar line corresponding to P_1 so that an optimal correlation score is achieved. P_1 is the reference point of correlation and the window that shifts along the epipolar line is centred on P_2. The match is valid *if and only if* the correlation score is also maximised when P_2 is the reference point of correlation and the window that shifts along the epipolar line of I_1 corresponding to P_2 is centred on P_1.

We implement this check by performing the correlation twice; namely *left-to-right* (L-R) correlation where the left image is the reference input and a match is searched for in the right image and *right-to-left* (R-L) correlation where the right image is now the reference input. Both correlations are performed in parallel and one row of disparity estimates from each of the correlation units are buffered. The size of this buffer needs

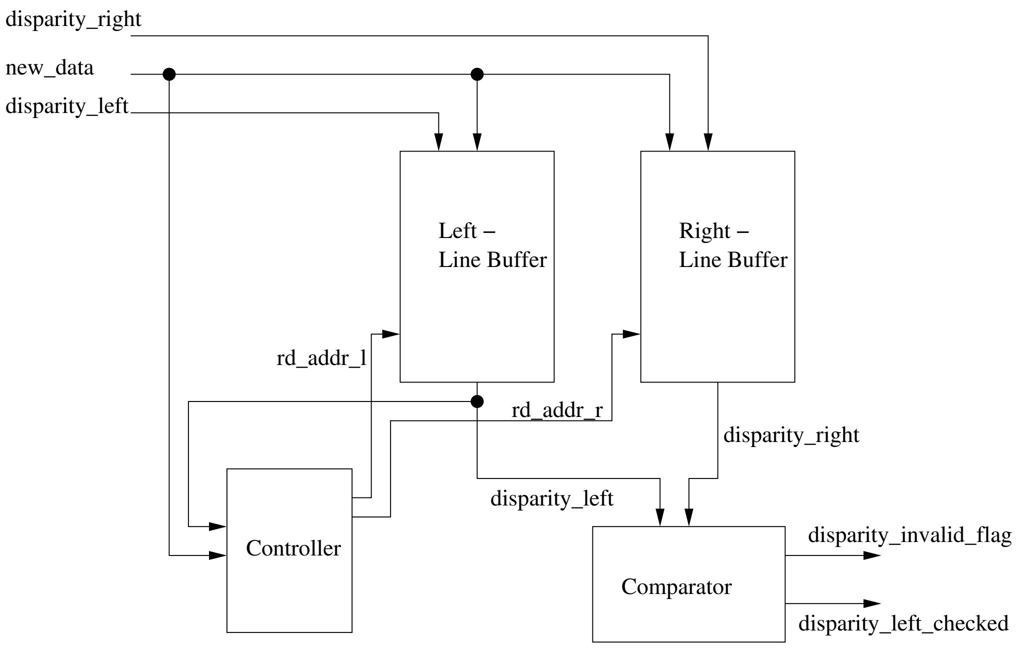


Figure 3.14: Architecture of Consistency Check Unit.

to correspond to the maximum disparity that the system is designed for, which in our case is 128 pixels. The architecture of the **Consistency Check Unit** is illustrated in Figure 3.14. A controller reads out the appropriate disparity value from the R-L correlation based on the disparity value of the L-R correlation. The values are then checked for consistency which is determined by a threshold level. The threshold level for our system is set so that a difference greater than 2 pixels values in disparity is classified as an invalid result, but this can be easily modified to a different level. A 1-bit flag is generated for each invalid disparity result.

3.5 Stereo-Vision system Architecture

In the previous sections, we discussed the design of pre- and post-processing blocks for image rectification and LR-RL consistency check respectively that improve the overall accuracy of the stereo-system as well as the development of a novel correlation architecture with shiftable windows. We now present a stereo-system based on the LWPC algorithm that utilises these developments. Hardware design techniques for implementing common signal and image processing tasks such as filtering and implementing a system across multiple FPGAs are also discussed.

The high level architecture of the proposed system is shown in Figure 3.15. It consists of six major units: Video Interface unit, Image Rectification unit, Scale-Orientation Decomposition unit, Phase-Correlation unit, Interpolation and Peak Detection unit, and Consistency Check unit.

The **Video Interface Unit** receives video signal from a stereo-rig with NTSC cameras (Figure 3.16). NTSC cameras output 640×480 images at 30 fps but the odd and even field of each frame are captured with a time difference of $(1/60)^{th}$ of a second. The even field consists of even rows (rows 0,2,4,...) and the odd field consists of odd rows (rows 1,3,5,...) of the image. This leads to a serious problem of "jagged edges" when 2-D separable filters are applied to these images. To avoid this problem, we treat the odd and even fields as separate frames, which results in the system processing 640×240 images at 60 fps. The processing capabilities and resource usage of the system in this mode exactly mimic that of the system if it were processing 640×480 images at 30 fps. The system can be easily adapted to process 640×480 images by designing a compatible video interface unit that accepts input from a progressive-scan camera.

The acquired pixel data is sent to the *Image Rectification Unit* as it arrives without any buffering. This unit runs on the same clock as the camera, called the *camera clock*, which can be different and asynchronous to the *system clock* that clocks the rest of the modules in the system.

The **Image Rectification Unit**, described previously in Section 3.2, warps both the left and right input images. The output from this unit resembles that of a stereo-rig in a simple set-up. Border pixels that are no longer part of the image due to warping as discussed earlier in Section 3.2 are indicated by a 1-bit flag.

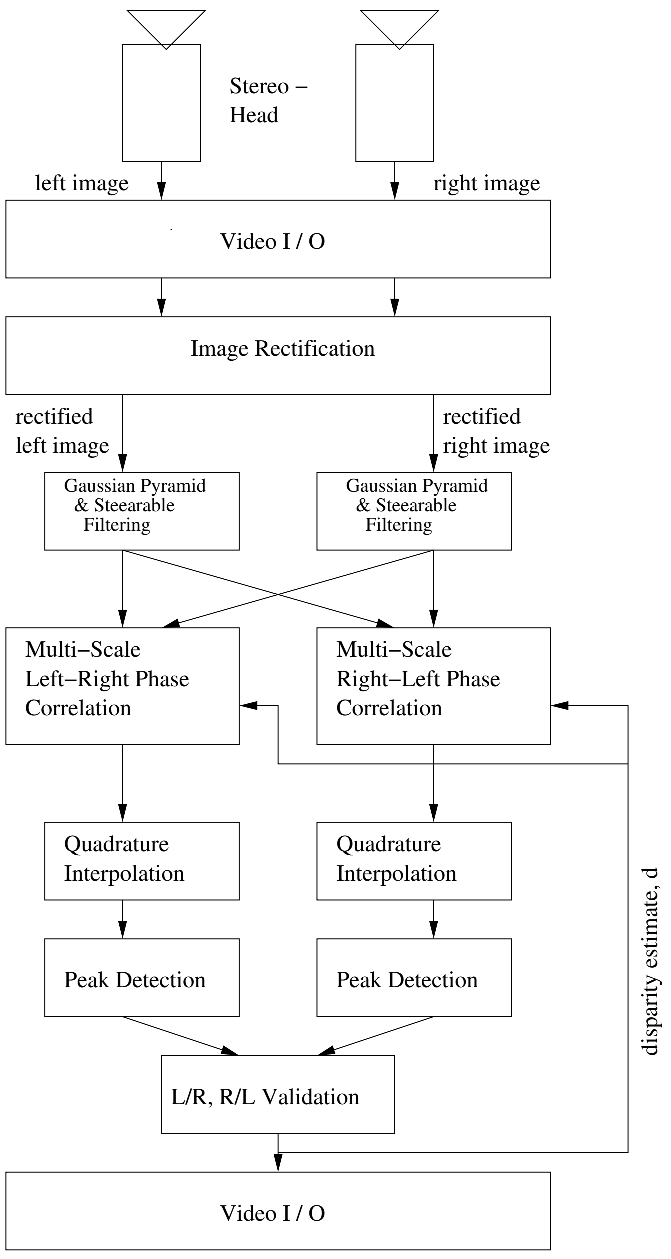


Figure 3.15: High-level architecture of the stereo-system

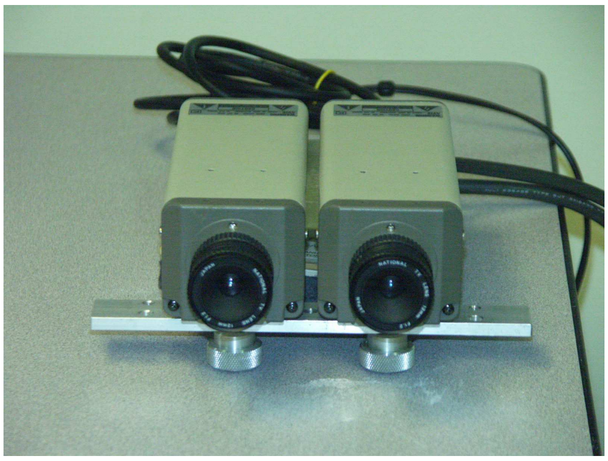


Figure 3.16: The stereo-rig with two NTSC cameras. The separation between the optical centres of the two cameras is approximately 70mm.

A *synchroniser circuit* is designed to handle glitch-free transfer of data between the Video Interface Unit, which runs on the camera clock, and the Image Rectification Unit that, like the rest of the system, runs on the system clock. The system clock is set at a higher frequency than the camera clock. When transferring a single-bit signal, such as the control signals from the *Video Interface Unit*, from a slow clock to a fast clock, two-levels of flip-flop can be used to synchronise the signal and avoid metastability issues. The synchroniser is illustrated in Figure 3.17. The transfer of pixel values, which are multi-bit signals known as a *bus*, cannot be achieved with a two-stage flip-flop. We synchronise the bus transfer by configuring the input image buffer to support dual-clock operations. The write operation from the *Video Interface Unit* to the buffer is performed on the *camera clock* and the read operation from the buffer to the *Image Rectification Unit* occurs on the *system clock*.

The **Scale-Orientation Decomposition Unit** first builds a three-level Gaussian Pyramid by passing the incoming right and left images through low-pass filters and sub-sampling. The pyramids are then decomposed into three orientations (-45^o, 0^o, +45^o) using G2/H2 steerable filters [?]. G2/H2 filtering is implemented using a set of seven basis filters. By choosing a set of proper coefficients for the linear combination of the

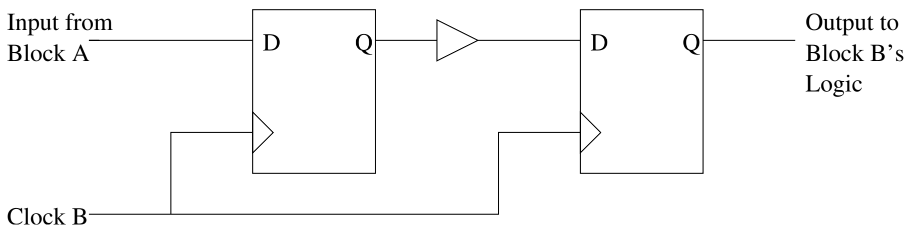


Figure 3.17: The synchroniser circuit uses two-stages of flip-flop to synchronise 1-bit signals that cross between the camera clock and the system clock.

basis filters, filter output of any arbitrary orientation can be synthesised. Since G2/H2 filters are X-Y separable, they require considerably less hardware resources than non-separable filters. The filter output is reduced to a 16-bit representation which is then sent to the Phase-Correlation unit.

Filtering is a standard signal and image processing operation. There are standard architectures [?] to realise specific filtering operation in hardware, some of which we describe here. A 2-D, X-Y separable filter can be realised by implementing it as two separate 1-D filters. This reduces the complexity of the filter to $O(N)$ whereas a regular (unseparable) 2-D filtering operation has a complexity of $O(N^2)$. When multiple filters need to be applied to an image, such as the seven basis filters to implement G2/H2 filtering, maximum efficiency in hardware can be achieved by performing the vertical (Y) filtering first followed by the horizontal (X) filtering. This allows us to share the vertical (Y) buffer across all the filters. The size of the Y buffer for an N-tap Y-filter is equal to $(N-1) \cdot W$, where W is the number of pixels in one scanline of the image. Only $N-1$ delay elements are needed for each of the horizontal filters so it is considerably less expensive to have separate buffering in the horizontal direction. The architecture of 2-D X-Y separable filter is illustrated in Figure 3.18.

Furthermore, all but one of the seven basis filters are either symmetric or anti-symmetric 7-tap finite impulse response (FIR) filters. An FIR filter with co-efficents $C_1, C_2, ..., C_7$ can be implemented in hardware as illustrated in Figure 3.19 (a). This requires 7 multipliers and 6 adders. Symmetric and anti-symmetric FIR filters with

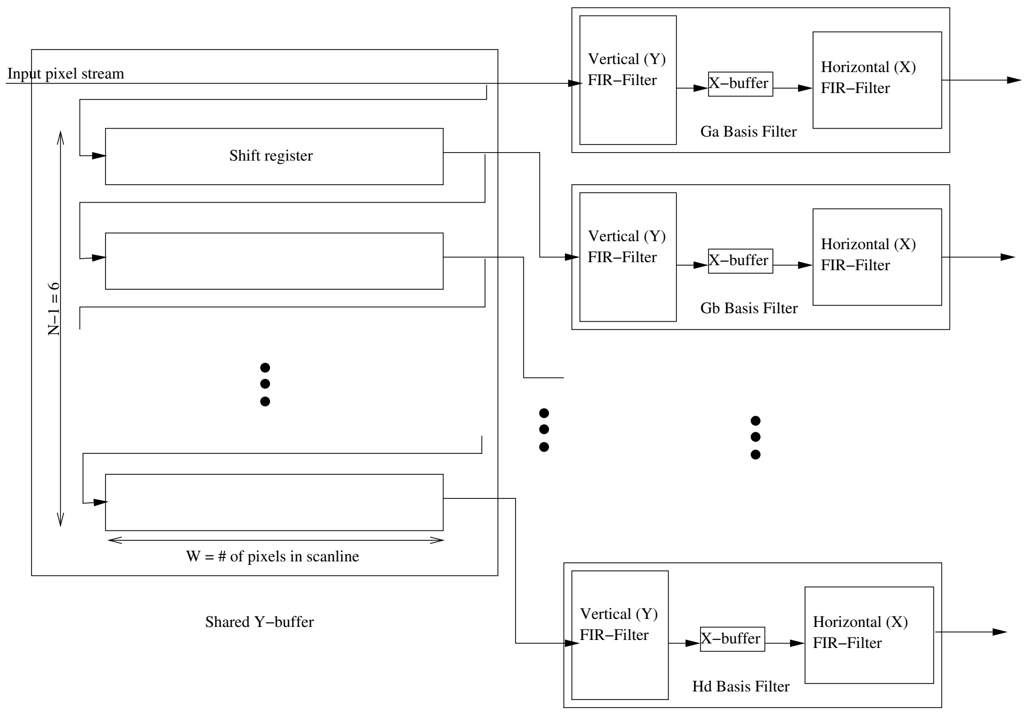


Figure 3.18: Performing vertical filtering first allows sharing of the Y-buffer for all the seven basis filters resulting in considerable resource savings.

coefficents $C_1 = \pm C_7$, $C_2 = \pm C_6$ and $C_3 = \pm C_5$ can be implemented as shown in Figure 3.19 (b) which reduces the number of multipliers required from 7 to 4. The number of adders remain the same.

The **Phase-Correlation Unit** computes the real part of the voting function $C_{j,s}(x,\tau)$ as mentioned in Eq. 3.3 for all $1 \leq s \leq S$, $1 \leq j \leq F$, $D_{min} \leq \tau \leq D_{max}$, where S is the total number of scales, F is the total number of orientations, and D is the disparity range of the correlation window. The epipolar search area for correlation is nine pixels wide for scale 1, five pixels wide for scale 2, and three pixels wide for scale 4 at any time instance.

The **Interpolation/Peak-Detection Unit** interpolates the voting function results, $C_{j,2}(x,\tau)$ and $C_{j,4}(x,\tau)$, from the two coarser scales, in both x and τ domains such that they can be combined with the results from the finest scale, $C_{j,1}(x,\tau)$. Quadratic interpolation is performed in the τ domain and constant interpolation in the x domain.

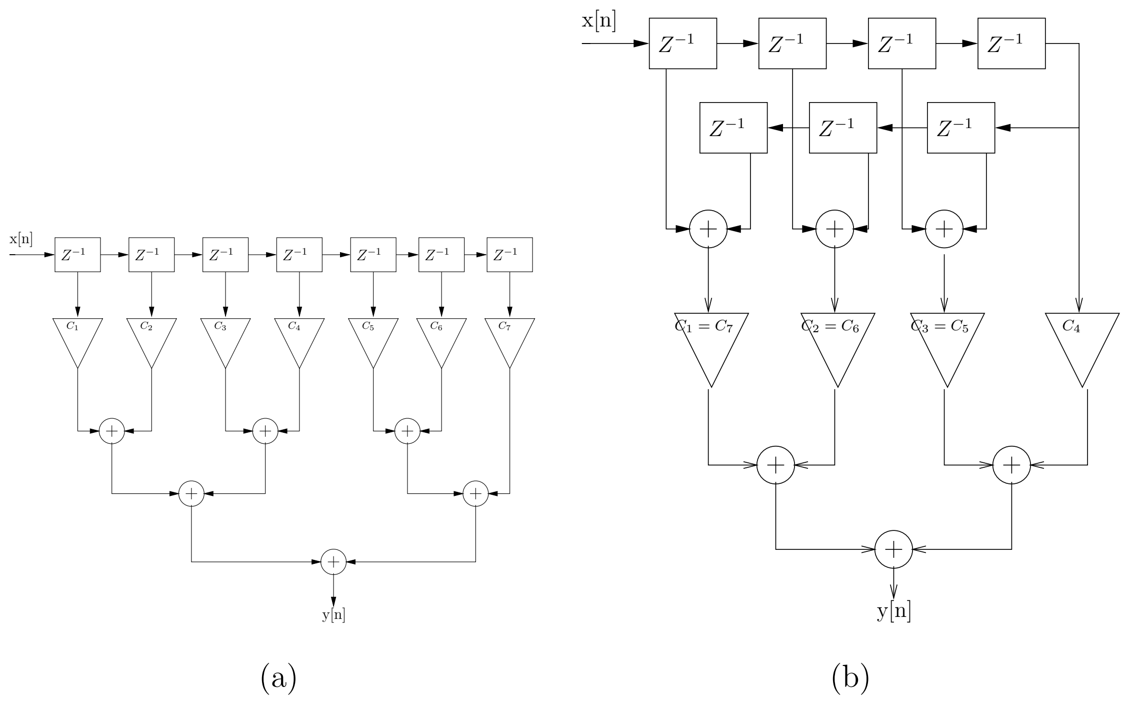


Figure 3.19: (a) Regular architecture of a 7-tap FIR filter. The number of multiplier required can be reduced for symmetric and anti-symmetric FIR filters using the architecture shown in (b).

The interpolated voting functions are then combined across the scales and orientations to produce the overall voting function $C(x, \tau)$. The peak in the voting function is then detected for each pixel as the maximum value of $C(x, \tau)$.

The **Consistency Check Unit** receives the estimated disparity results from both left-right and right-left correlations and performs a validity check on the results. The disparity value is accepted as valid if the results from the two correlation windows do not differ by more than two pixels. The checked disparity values are then sent back to the video interface unit to be displayed on a monitor or otherwise made available as output. The rejected disparity estimates are assigned a special flag for display purposes.

3.5.1 Stereo-vision system on Transmogrifier-4

The stereo-vision system described is too large to be implemented on a single FPGA. The system is partitioned across four FPGAs on the TM-4 board as illustrated in Figure 3.20.

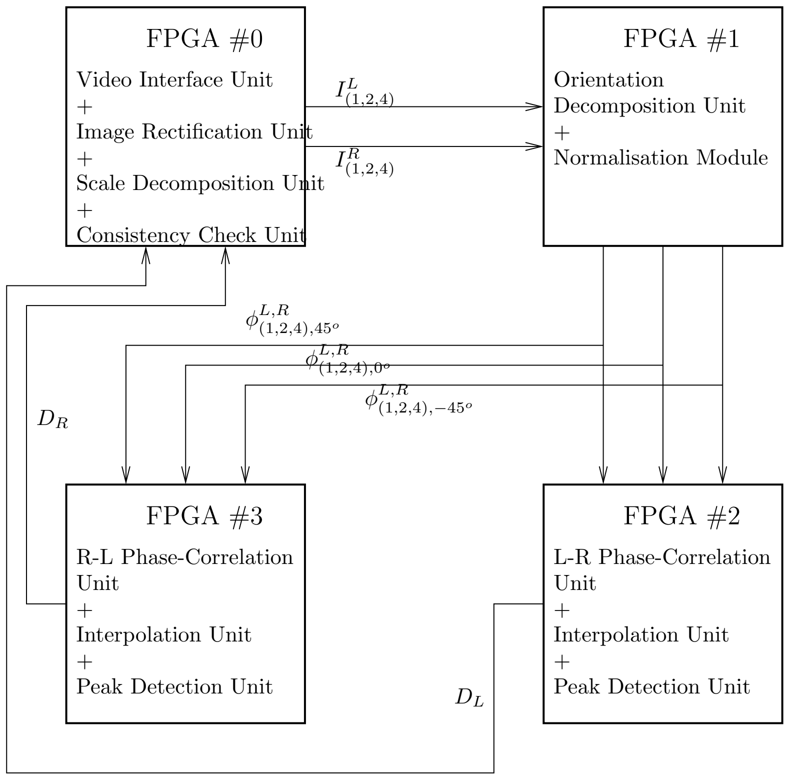


Figure 3.20: Partitioning of the algorithm and data transfer on the TM-4 board.

The partitioning of the algorithm is dependent on the resources available on each FPGA, the communication bandwidth between FPGAs, and external peripherals support on each FPGA. FPGA #0 contains the *Video Input / Output*, *Image Rectification*, and *Consistency Check* units. FPGA #1 contains the *Scale/Orientation Decomposition Unit* and the *Normalisation* module of the *Phase-Correlation Unit*. The remaining modules of the *Phase-Correlation Unit*, *Interpolation*, and *Peak Detection* modules are implemented on FPGA #2 that outputs a disparity map with the left image as the reference image. FPGA #3 is reserved to perform the same operations as FPGA #2 but with the right image as the reference. The disparity values from each of these would be checked for consistency on FPGA #0. Note that at the time of this work, the TM-4 board still has some bugs rendering FPGA #3 unusable. The complete system has been extensively simulated with real video sequences. The major units have also been tested on the TM-4 board and system integration is underway. Simulation results are shown in Chapter 4.

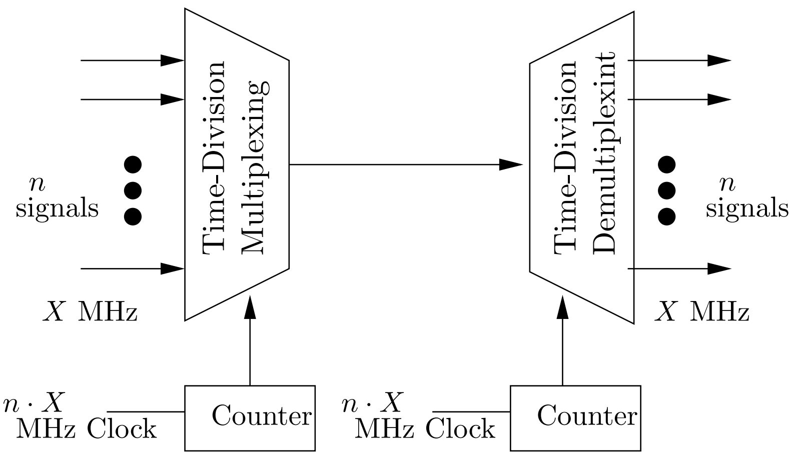


Figure 3.21: The data to be transfered is time multiplexed on the sending FPGA and de-multiplexed on the receiving FPGA, allowing large amounts of data to be transferred on a limited width data bus.

FPGA #1 is connected to FPGA #2 and FPGA #3 with a bus that is approximately 100-bits wide, but the number of bits that need to be transferred between these FPGAs is much larger. Eight bits are retained after normalisation for the real and imaginary components of the phase at each orientation for each scale. A total of 288 bits, 16 bits for each of the three orientations ($-45^o, 0^o, +45^o$) at each scale (1,2,4) for both the left and right images, need to be transferred between the FPGAs. We use Time Division Multiplexing to handle this transfer as illustrated in Figure 3.21.

3.6 Summary

Many vision algorithms are computationally expensive and require specialised hardware to run at frame-rate. The design of hardware systems for these applications require more than simply porting the software version of the algorithm to hardware due to the limited logic resources on hardware devices. In this chapter we saw that a traditional approach to implementing the correlation unit is not the most efficient solution for a hardware realisation of the LWPC stereo algorithm. A novel architecture for the correlation unit with shiftable correlation windows was developed in this work to exploit the temporal coherence present in real-life sequences. The architecture uses two correlation windows, one to perform localised correlation in an area of the image where a corresponding match is expected by tracking disparities in the time domain, and the other to perform a stochastic search over a wider search band over time. The maximum disparity can easily be increased or decreased by setting the maximum limit for the roving window without any changes in the logic resource usage except an appropriate amount of memory usage corresponding to the number of pixels that need to be buffered.

In addition, an image rectification unit was designed to warp the incoming stereo images so that the epipolar lines in the image pair are horizontal and aligned, to improve the accuracy of the disparity estimates. The design of a consistency check unit is also discussed. This consistency check invalidates erroneous disparity estimates by performing consistency checks on the left-right and right-left disparity estimates. Finally, a complete architecture of the LWPC-based vision system on the TM-4 board was presented and design techniques and architectures used to implement this system were discussed. In Chapter 4, we compare the performance of our system to others in the literature and present results from our system.

Chapter 4

Results

4.1 System Performance

The stereo system presented in this work performs multi-scale, multi-orientation disparity estimation up to 128 pixels using roughly the same amount of hardware resources as the system in [?] that is capable of handling disparities of only 20 pixels. A dense disparity map is produced at the rate of 60 fps for an image size of 240×640 pixels (which is equivalent to 480×640 pixels at 30 fps).

A common metric to measure the performance of a stereo-system in terms of throughput is the Points $\times$ Disparity per Second (PDS) defined as follows:

$$PDS = \frac{n \times m \times D}{T} \tag{4.1}$$

where $n \times m$ is the image size, D is the range of disparities evaluated and T is the time it takes to evaluate the disparities. It must be noted that the PDS metric does not take into account algorithm complexity, or accuracy of the results, but rather is a measure of all possible disparity values that are calculated by the algorithm in a specified time frame. For our system, we use $D = 18$ in the above equation because eighteen unique disparities are computed in each frame; nine disparity values are computed by the primary window

System	$n \times m$	**D**	**T (msec)**	**PDS** ($\times 10^6$)	**algorithm**	**platform**
INRIA [?]	256 x 256	32	280	7.5	Intensity Correlation	PeRLe-1 board (23 Xilinx XC3090 FPGAs
PARTS [?]	240 x 320	24	23.8	77	Census	16 Xilinx 4025 FPGAs
CMU [?]	200 x 200	30	33	36	sum of absolute difference	C40 DSP + real-time processor
UofT-TM3 [?]	256 x 360	20	33	55	LWPC (Phase-based)	TM-3A board (4 Xilinx Virtex 2000E FPGAs)
Our system	240 x 640 (480 x 640)	128	16.5 (33)	165	Temporal LWPC	TM-4 board (4 Altera Stratix S80 FPGAs)

Table 4.1: Comparison of various reported real-time stereo vision system performance.

and nine by the secondary window. The system designed in this work is capable of achieving a performance of over 330 million PDS when both the left-right and right-left correlation units are implemented, which is considerably greater than the any of the others listed [?, ?] some of which are listed in Table 4.1. Even with single directional correlation the system has a performance of 165 million PDS.

Compared to the other systems in Table 4.1, our system is capable of handling the largest disparity range without a similar increase in the resource usage over [?]. We achieved this by utilising the temporal coherence in real-life video sequences and designing a shiftable correlation window whereas the previous system in [?] uses a traditional fixed-window architecture for correlation. The system does not have a “hard limit” on the maximum disparity; this can be easily increased by setting the maximum distance for the *SRW* to a higher value. But, at the same time, there is a trade-off between time to recovery from a mis-match and maximum disparity as discussed previously in Section 3.3.2. The logic resource usage of the Phase-Correlation module remains the same except an increase in the size of the partial-line buffers. A software implementation of this algorithm in MATLAB requires over 25 minutes to generate the disparity map for a pair of images on a LINUX 3 GHz machine with 4 GB of memory. This is because the correlation windows do not follow a regular pattern so it is not possible to take advantage of MATLAB's matrix computation features. If multiple copies of the data are stored like in the hardware version, there will be a speed-up in the computation time but it will still not be close to the frame-rate performance of the hardware implementation. A C/C++ implementation may offer speed-up over the MATLAB implementation, and though no tests have been carried out, even this is not expected to match the hardware performance.

4.2 System Results

In this section, we present results from various stages in the system. First, we look at the output from the image rectification unit. Next, the output from the scale / orientation decomposition unit is presented. Finally, disparity estimates before and after performing left-right consistency check from a pre-captured and pre-rectified sequence are presented. The image size used in this work is 240 scanlines in height and 640 pixels in width. Each image is either the odd or even field of a frame captured by an NTSC camera, but no subsampling is performed in the horizontal direction so the image appear to be stretched horizontally.

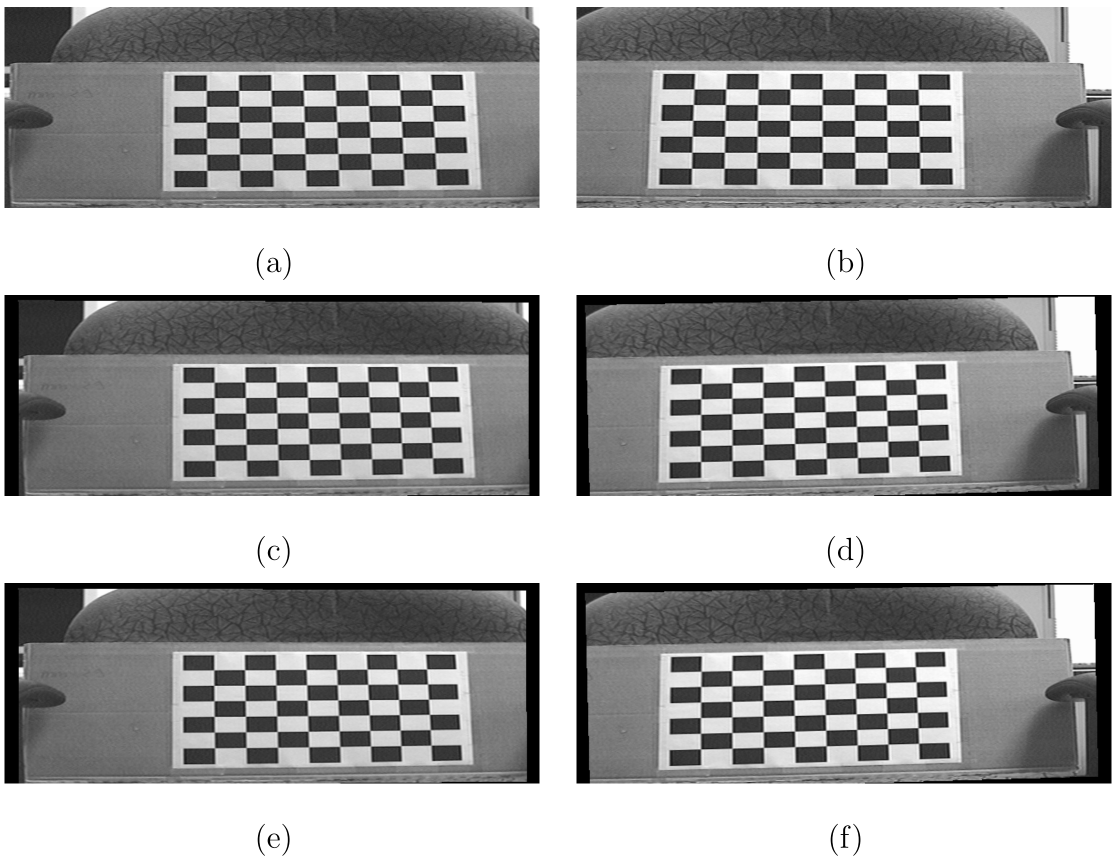


Figure 4.1: The original images from the left and right cameras are shown in (a) and (b). The rectified left and right images obtained using a second order polynomial (c) and (d) compare favourably to the expected rectified images (e) and (f).

4.2.1 Image Rectification

The incoming images from the camera are first warped so that corresponding pixels in the image pair appear on the same scanline. The original left and right input images are shown in Figure 4.1 (a) and (b) respectively. The warped left and right camera outputs from the hardware computed using a second order polynomial are shown in Figure 4.1 (c) and (d). In Figure 4.1 (e) and (f), we show the expected results obtained from a software implementation using a matrix inverse transformation.

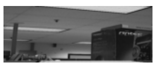
The rectified images obtained using a second order polynomial are very similar to the expected rectified images. The difference in the results is negligible at the top left of the image (pixel coordinates (0,0)), but increases with increasing pixel co-ordinates in the x- and y-axes. An error analysis of the warped image obtained using the second order quadratic with fixed-point precision was shown earlier in Section 3.2. The results can be improved by computing the coefficients for the warping polynomial with the centre of the image set as pixel coordinates $(0, 0)$ and then normalising the pixel indices so that the width of the image (pixels 1 to 640) and the height of the image (pixels 1 to 240) are each represented between $(-1, 1)$. The actual calibration of the cameras was done using [?], a freely available online calibration toolbox for MATLAB. We do not compensate for radial distortion currently but the stereo-rig can be recalibrated to compensate for radial distortion. A second order quadratic such as the one used in this work to approximate the homography is capable of compensating for radial distortion so long as pixel coordinate $(0, 0)$ is mapped to the calibrated optic centre of the camera. The stereo-rig would also have to be recalibrated if there is any disturbance to the stereo-rig setup that would change the position or orientation of one camera with respect to the other. The new co-efficients obtained from a recalibration can simply replace the existing coefficients in our system and the VHDL recompiled without any need to modify the rest of the image rectification unit.

4.2.2 Scale and Orientation Decomposition

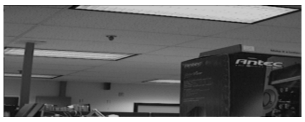
The next step in the LWPC stereo algorithm is to create a Gaussian pyramid of the input image and apply steerable G2-H2 filters to the image pyramid to obtain phase information in the images. The Gaussian pyramid generated by the scale decomposition block is shown in Figure 4.2. The H2 filter outputs at the original scale (Scale 1) tuned to -45^o, 0^o, and $+45^o$ are shown in Figure 4.3.

4.2.3 Disparity Results

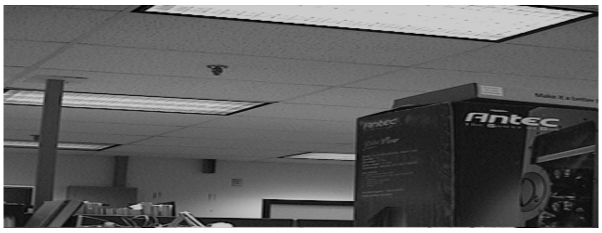
The last two major components of stereo system are the phase-correlation unit and the peak-detection unit that generates the best estimate of the disparity value for each pixel.



(a)



(b)



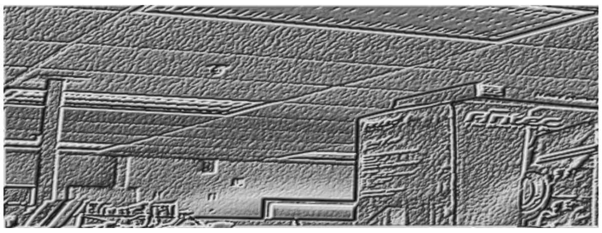
(c)

Figure 4.2: Output of the scale decomposition block is a Gaussian pyramid at Scale 4 (a), Scale 2 (b) and Scale 1 (c).

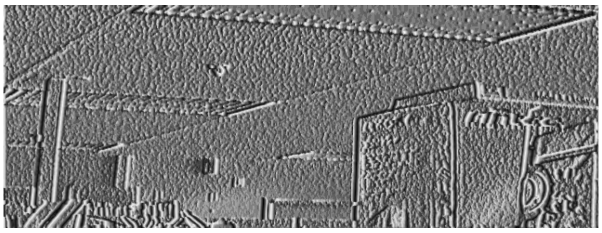
This last unit is still undergoing integration on the TM-4 board so we present results from a ModelSim simulation of the system.

A pre-rectified captured sequence is used for this simulation. Frames 1, 7, and 15 from the sequence captured by the left camera, MDR-1, are shown in Figure 4.4. The sequence consists of a person in the foreground and poster boards serving as the background in the image. The person is standing still and the camera is moving from left to right as well as slightly diagonally. Another way to think of this is that the cameras are still and the person is moving to the left.

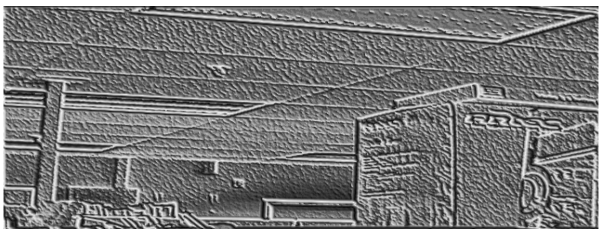
Figure 4.5 shows the disparity results using left-to-right correlation for the first 15 frames of the sequence. The maximum disparity in this sequence is around 40 pixels. The system is currently configured so that the *SRW* searches for a corresponding match in the range of 5 to 13 pixel disparities during the first frame, 14 to 22 pixel disparities during the second frame and so on. The initialisation stage for this particular sequence spans the first three frames, as illustrated in Figure 4.5 (a) to (c) and the system settles



(a)

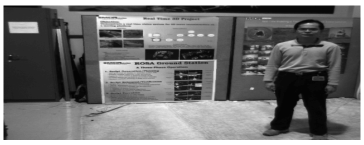


(b)

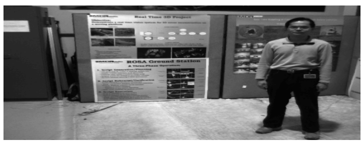


(c)

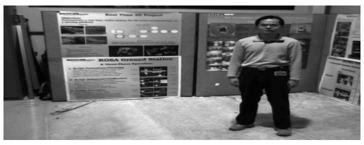
Figure 4.3: Normalised H2 filter output at -45^o (a), 0^o (b) and $+45^o$ (c) on scale 1 image of the Gaussian pyramid.



(a)



(b)



(c)

Figure 4.4: Frames 1 (a), 4 (b), and 15 (c) from the left camera of the MDR-1 pre-captured stereo sequence.

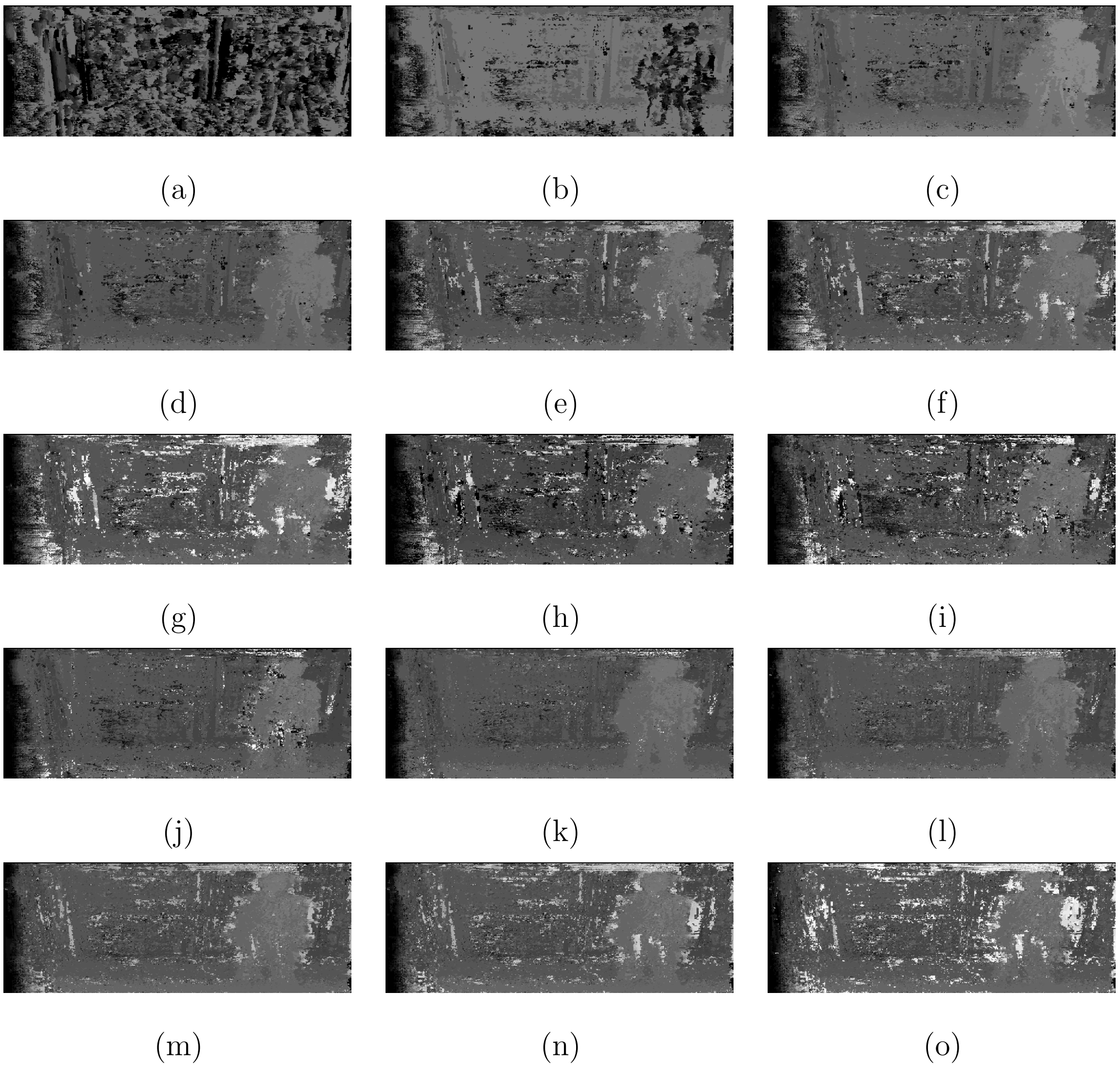


Figure 4.5: Disparity maps for frames 1 to 15 of the MDR-1 sequence using left-to-right correlation are shown in (a) to (o) respectively.

into a steady state by frame 4 as illustrated in Figure 4.5 (d). The PTW is then able to track the disparities over subsequent frames as illustrated in Figure 4.5 (e) to (o).

As the person moves to the left, parts of the background become disoccluded to the right of the person. Because there is no previous disparity estimate for these disoccluded regions, they tend to have inaccurate disparity estimates during the first frame that the regions become visible as illustrated in Figure 4.5 (g). The algorithm recovers accurate disparities for these disoccluded regions over the course of next few frames with the help

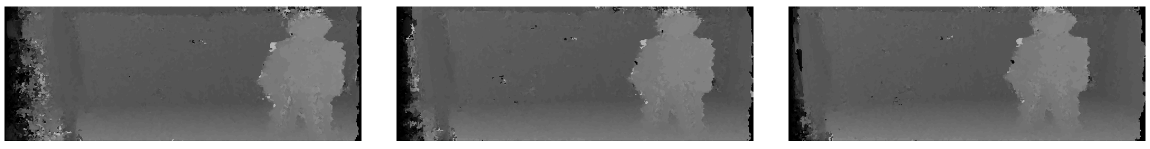


Figure 4.6: Disparity map of frames 1 (a), 11 (b), and 15 (c) of the MDR-1 sequence generated from the golden version of the LWPC algorithm running on a Linux processor.

of the *SRW* as can be seen in Figure 4.5 (j). Another important observation is the level of noise in the disparity estimate for different frames of the sequence. The disparity map for certain frames (for example frame (o)) has greater amount of noise than others (for example frame (k)). This is because we use a 1-D 1×5 pixel Gaussian mask for computing the voting function in Equation 3.3 which can handle horizontal translation, but since any depth discontinuity (in this case introduced by the small vertical motion in the sequence) disrupts tracking, the quality of the disparity estimate suffers. The architecture can be modified to accommodate a 2-D Gaussian mask (such as a 3×3 pixels mask) to improve performance, as discussed later in Section 5.1.

The disparity estimates from the golden version is shown in Figure 4.6 for comparison purposes. The golden version is a software (MATLAB) implementation of the LWPC algorithm as described in [?] using floating-point arithmetic. The golden version is configured to support the maximum disparity in the scene under consideration using a single fixed correlation window at every frame.

4.2.4 Consistency Check Results

The last step that the system performs is a consistency check of the disparity estimates. This is used to improve the accuracy of the system by comparing the disparity estimates from the left-to-right correlation and right-to-left correlation and rejecting disparity estimates that differ by more than 2 pixel values. The rejected pixels are set to black for

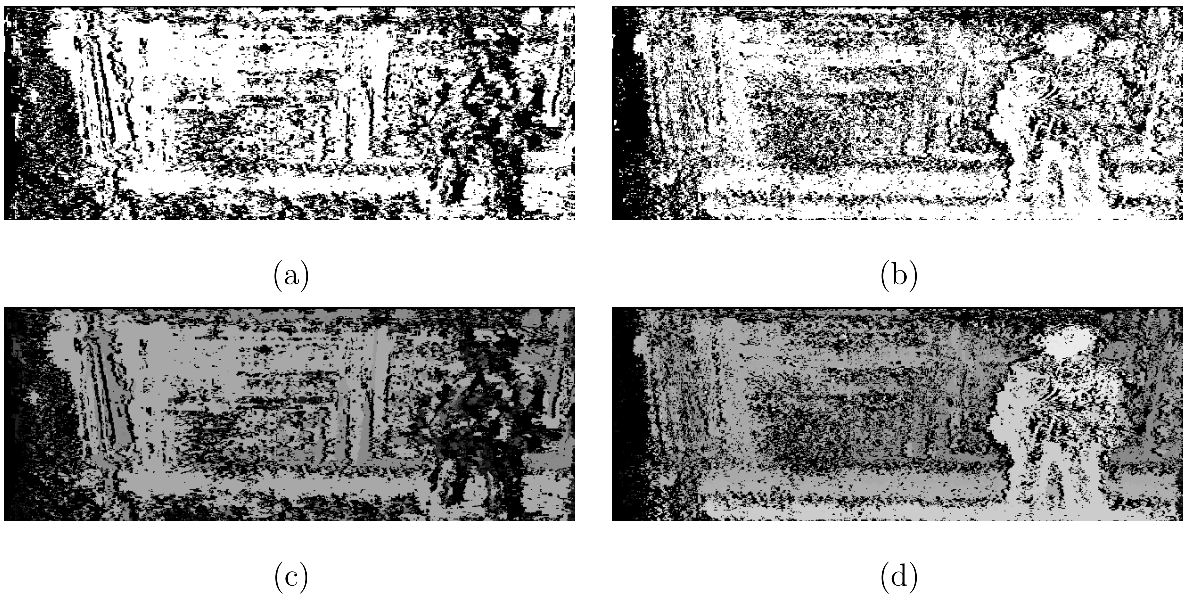


Figure 4.7: Binary map showing valid matches in white and rejected disparity estimates in black for frame 2 (a) and frame 11 (b). (c) and (d) show the disparity maps for the two frames. In (c) and (d), the black pixels do not actually have a disparity value of zero, but are rather pixels for which no good disparity estimate exists.

display purposes and the accepted disparities are assigned the disparity estimate from the left-right correlation.

A binary map obtained after rejecting invalid matches for frame 2 of the MDR-1 sequence is shown in Figure 4.7 (a). The "white" pixels represent pixels that passed the left-right consistency check and the "black" pixels represent pixels that failed the consistency check. Fewer pixels pass the consistency check test during the initialisation frames as expected. The pixels that pass the consistency check belong mainly to the image background because they have a smaller disparity which are recovered early. Once the system has settled into steady-state the accuracy of the system improves as illustrated in Figure 4.7 (b) for frame 11. The disparity maps for the two frames after rejecting invalid disparity estimates are shown in Figure 4.7 (c) and (d). The accepted disparity estimates are assigned the disparity values obtained from left-right correlation.

4.2.5 Analysis of Disparity Results

In this section, we analyse the performance of our system by comparing the disparity estimates obtained from our work with the disparity estimates from the golden version. However, before we present the analysis, it should be noted that the disparity estimates after consistency check are quite sparse as some of the estimates are rejected due to the noise in the results. In addition to the use of fixed-point arithmetic as opposed to floating-point, our system computes the voting function using a 1×5 pixel Gaussian mask instead of a 2-D $n \times n$ mask so we expect some degradation in results as we had earlier mentioned in Section 3.3.4.

The consistency check phase eliminates erroneous disparity estimates that are a result of occlusion or lack of texture in the captured image. The occluded areas of the image pair in Figure 4.7 (d), left edge of the person and a vertical band area at the left edge of the image, are rejected after consistency check. Correlation based stereo-matching have an inherent limitation in that they are able to successfully estimate disparities only in regions with texture. A 1-D mask for correlation such as the one we have used in this work does not integrate texture information in the vertical direction. This lack of texture information results in the disparity estimates of some of the background regions as well as regions in the middle of the person to be rejected.

To get a quantitative sense of the accuracy of the disparity estimates, we look at two key performance areas: i) results of the left-right consistency test, and ii) accuracy of the accepted disparity estimates in comparison to the golden version.

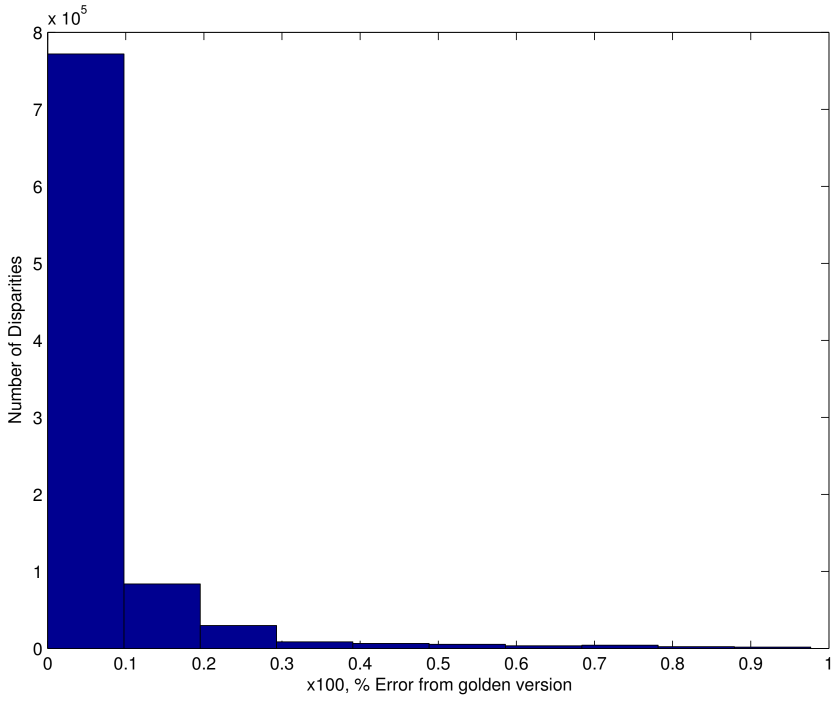
Performing a left-right consistency check is a standard method of validating disparity estimates. Pixels or other image features whose disparity estimates do not differ by more than a certain threshold between the left-right correlation and right-left correlation are treated as having accurate disparity estimates. Table 4.2 shows the percentage of accepted versus rejected pixels for the MDR-1 sequence. The total percentage of pixels that pass the consistency test for frames 4 to 15 combined is just over half the pixels

Percentage of:	**Frames 4 to 15**	**Frames 10 to 12**
Accepted	53%	62%
Rejected	47%	38%
Accepted with disparity difference $\leq$ 10% from golden version	85%	85%

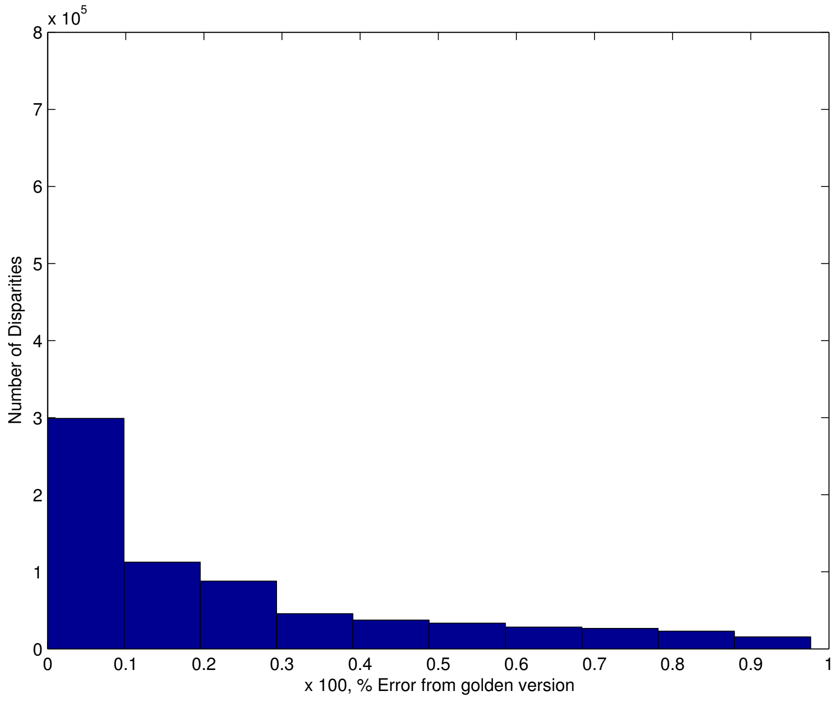
Table 4.2: Percentage of disparities accepted and rejected disparities after left-right consistency check. The last row shows the percentage of accepted pixels that have a difference of 10% or less in disparity value from the golden version.

in the image at 53%. This number increases to 62% when only frames 10 to 12 are considered which have less noise in the disparity estimates. Frames 10 to 12 do not have a significant vertical component in scene motion, hence the horizontal matching window is able to handle the situation better. For pixels that pass the consistency check, approximately 85% have disparity estimates with a difference of 10% or less from the golden version disparity estimates.

A histogram provides a better sense of the performance of the disparity estimates. Figure 4.8 shows the histogram for the difference in disparity values between the system in this work and the golden version for (a) the accepted pixels and (b) the rejected pixels for frames 4 to 15 of the MDR-1 sequence. Eighty-five percent of the accepted pixels have a difference of 10% or less from the golden version and most of the remaining accepted pixels fall within 20% of the golden version disparity estimates. The use of a 1-D Gaussian mask explains the few accepted pixels that have a greater difference as compared to the golden version. The rejected disparity estimates tend to have much greater difference as compared to the golden version. Similar observations are made for the results of frames 10 to 12, as shown in Figure 4.9.

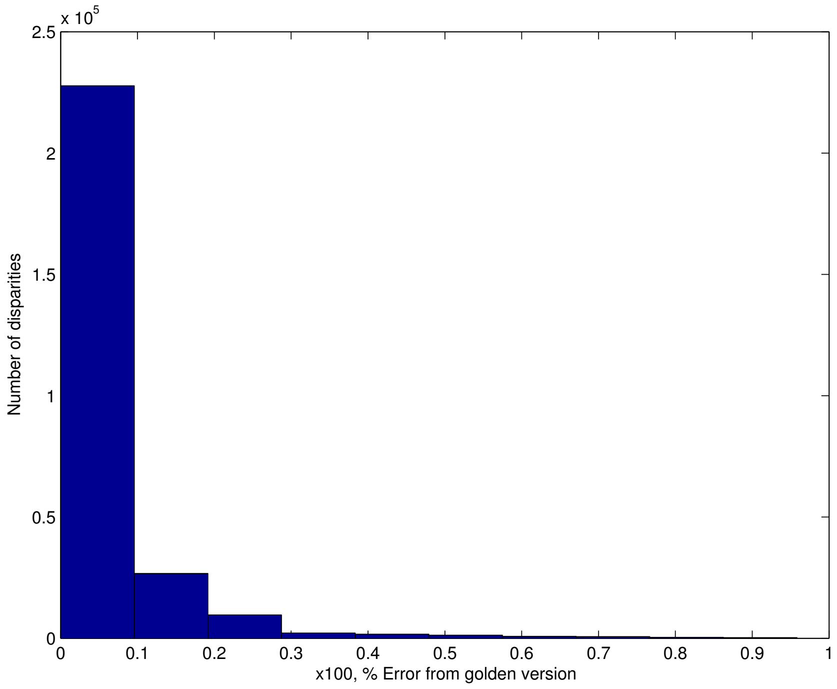


(a)

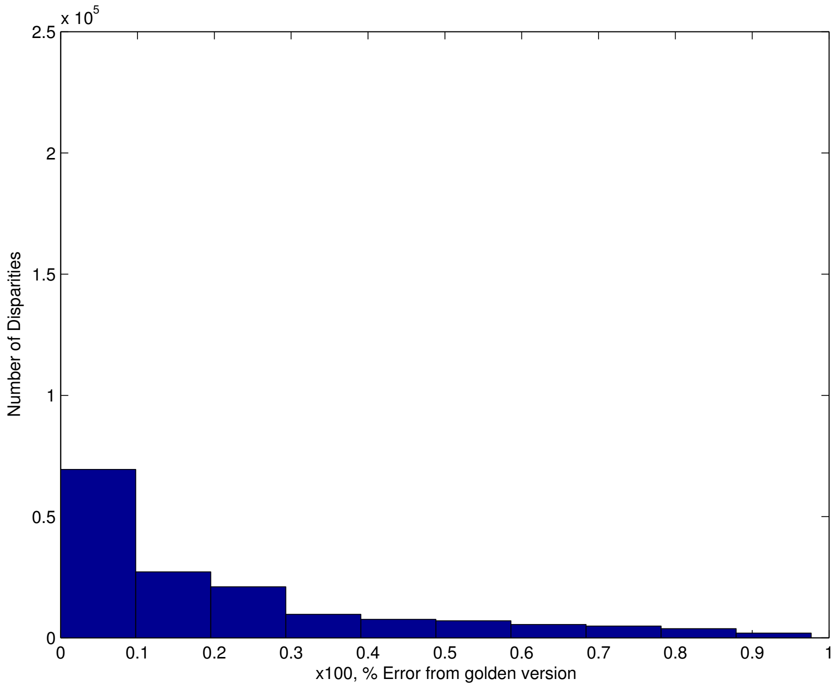


(b)

Figure 4.8: Histogram of difference in disparity estimates between our system and golden version for (a) accepted pixel and (b) rejected pixels. The values are computed using results from frames 4 to 15 inclusive.



(a)



(b)

Figure 4.9: Histogram of difference in disparity estimates between our system and golden version for (a) accepted pixel and (b) rejected pixels. The values are computed using results from frames 10 to 12 inclusive.

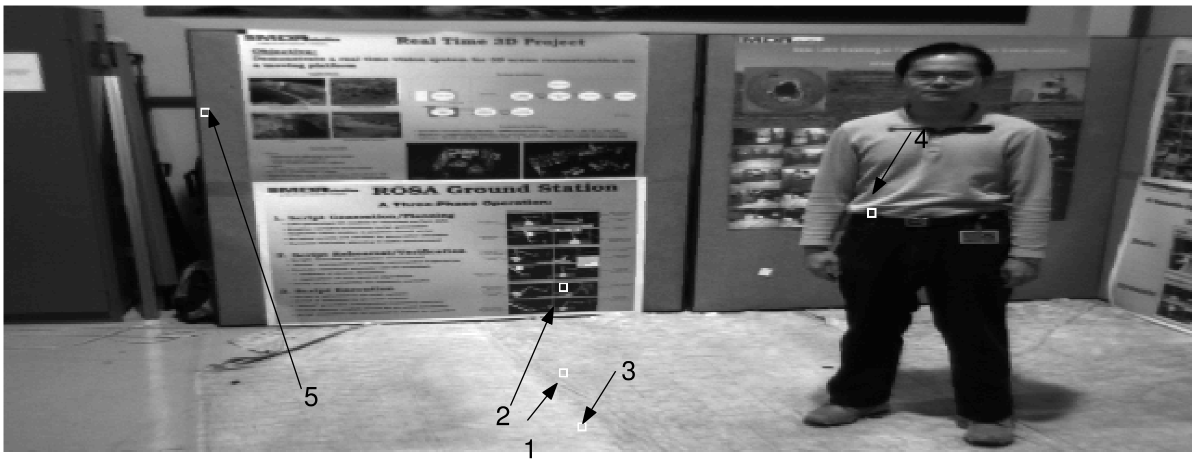


Figure 4.10: The five 5 × 5 pixel blocks used to compute the disparity estimate error in Table 4.3 are numbered and indicated by the arrow.

Block	**1**	**2**	**3**	**4**	**5**
% error	1.03	1.80	1.90	5.60	6.73

Table 4.3: Percentage of error as compared to the golden version disparities for the numbered 5 × 5 blocks showing in Figure 4.10. The blocks are shown inside the white circles.

The second area of analysis is to compare the actual difference in disparity estimates for a given pixel. The disparity estimates obtained from the golden version are compared to the disparity estimates obtained from our temporal algorithm using fixed-point arithmetic. Table 4.3 shows the average error in percent of the disparity estimates of five 5×5 pixel blocks. These blocks are shown and numbered in Figure 4.10. Blocks 1, 2, and 3 are used from frame 11 and blocks 4, and 5 from frame 15 to get a sense of the error in frames with various levels of noise in the results. Frame 15 has greater amount of noise present in the disparity map as compared to the disparity map from Frame 11. The error in disparity estimates is between 1% and 2% for frames with less noise and between 5% and 7% for noisier frames. This is comparable to the system in [?] which reports errors of between 3% and 13%.

4.3 Summary

In this chapter we first compared the performance of our system with other real-time stereo systems using the PDS metric. Then the results of image rectification, Gaussian pyramid, and G2/H2 filtering from the implementation on the TM-4 board are shown. Disparity estimates and consistency check results are provided from a ModelSim simulation of the system. Finally, a detailed analysis of the disparity estimates pointing out where the system performs well and where it requires improvements is presented.

In the next chapter, we discuss ways to to improve the accuracy of the system. This includes a modification of the line buffers used in the phase-correlation unit to accommodate a 3×3 Gaussian mask for computing the voting function as well as the use of other post-processing options to improve the disparity estimates.

Chapter 5

Conclusions and Future Work

As discussed in Chapter 2, FPGAs are ideal for many vision tasks as they allow us to take advantage of the inherent parallelism of vision algorithms. However they have limited resources. Though larger and more complex designs can be realised using devices with greater resources such as ASICs or even larger FPGAs, it may not be an ideal solution due to higher costs. Also, many algorithms require devices with much greater resources than currently available. It is a challenging task to develop complex vision systems with these resource constraints, and this differentiates the designing of hardware-based system from software-based systems. We need to take advantage of the information present in the data to be processed to develop efficient architectures for the system at hand rather than simply taking a software implementation and "porting" it to hardware.

Keeping these ideas in mind, we have presented in this work an FPGA-based, frame-rate stereo system with the following salient features:

1. Ability to handle very large disparities using limited hardware resources by designing a novel architecture for performing correlation in hardware.

2. Improved accuracy by including an image rectification unit to pre-process the images and a consistency check unit to remove invalid disparity estimates.

3. Capability to handle image sizes of 240×640 pixels at 60 frames per second. The system is also capable of handling 480×640 images at 30 frames per second if a suitable video interface is available, as discussed later in Section 5.1.

The system captures views of its surroundings using a stereo-rig and generates dense disparity maps. The results of the disparity map provide depth information and can be used to construct a 3D map of the viewed scene. This information is useful in a variety of vision tasks such as object recognition, autonomous navigation, and surveillance among others.

The highlight of this work is the design of a correlation unit with shiftable correlation windows. This a departure from the traditional fixed window correlation architecture that hardware designers are accustomed to, and we have not found a shiftable window architecture in the present literature. This allows our design to use temporal coherence to track disparities over time and perform localised correlation. Our architecture is able to support a disparity range of 128 pixels with the same amount of hardware logic resources for the correlation unit as the system in [?] which is limited to a disparity of 20 pixels. The range of our system can easily be increased but there is a trade-off between the maximum disparity that the system can support and the recovery time from a mismatch.

In addition, the correlation unit can be configured easily to accommodate various correlation search strategies as discussed in Section 3.3.5. The use of the correlation unit is not limited to a stereo-vision system. The flexibility of the correlation windows means that the unit can be used as a platform for correlation based algorithms that allow vision researchers to implement and experiment in hardware with minimal development time.

5.1 Future Work

The LWPC algorithm used in this work performs correlation in three orientations at three scales, a task requiring significant resources. Though we have been successful in developing an architecture for the correlation unit that handles large disparities with limited logic resources for the arithmetic operations, the operations were limited to a 1×5 correlation for this work. This has had an impact on the accuracy of the disparity estimates by introducing noise in the results. The main limitation of our architecture is that it needs to store multiple copies of the input image stream preventing the buffering of multiple lines at a time that are needed for correlation with an $n \times n$ Gaussian mask, such as 3×3 or 5×5 pixel masks, due to the limited on-chip memory. The simplest way to accommodate an $n \times n$ Gaussian mask is to support smaller image sizes and use only two orientations as in [?] instead of the suggested three. With an image width of 320 pixels and correlation using a 3×3 mask, a total of 240 out of the 364 M4K memory blocks are required to store multiple copies of three rows at a time. For larger image widths, the line buffers need to be modified as shown in Figure 5.1 so that three adjacent pixels from the same row can be accessed in a single clock cycle. For a 3×3 mask, nine pixels, three adjacent pixels from three rows, are required for each correlation. In addition the voting function needs to be computed at a maximum of nine locations corresponding to the epipolar search band at Scale 1. This can be achieved by designing a multi-rate system. The input stream arrives at approximately 13.5 MHz which means the correlation unit in a multi-rate design would have to run at 9 times that speed or 121.5 MHz to maintain frame-rate. Only a single copy of three rows needs to be buffered at a time using this modification. Each row requires six M4K memory blocks at Scale 1 and three M4K blocks each at Scales 2 and 4. A total of 216 out of 364 available M4K blocks are required to store a single copy of three rows at three different orientations for both the left and right images.

Another possibility is to use off-chip memory to buffer the output of the Scale/Orientation Decomposition block and design an on-chip cache to rapidly access the required pixels. The use of off-chip memory also requires the design of memory controller to interface with the off-chip memory blocks. For a complex design such as this, board-level simulation

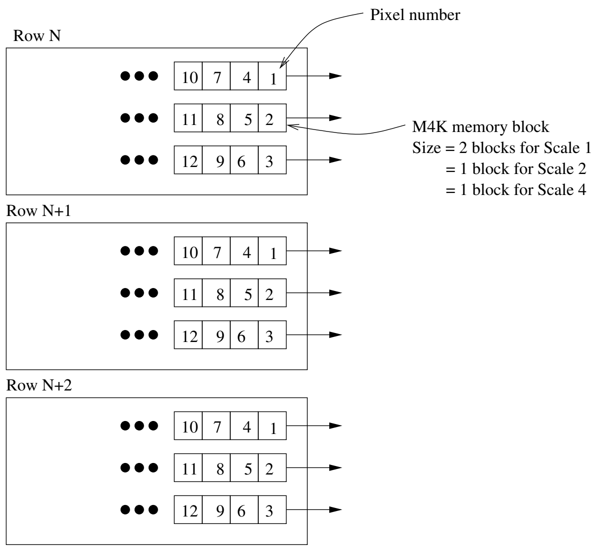


Figure 5.1: Modified buffer to store three rows of phase information. This would eliminate the need to store multiple copies of the same row when used in conjunction with a multi-rate system.

models that include the interaction between the FPGAs and off-chip memories will be needed for a rapid design process.

Other possibilities for future work include:

- Design of a de-interlacer module to eliminate the "jagged edges" issue with NTSC cameras so that the system can operate on the desired image size of 480 × 640 pixels. Alternatively, a compatible module to interface with FireWire cameras can be designed. FireWire cameras provide a progressive-scan output so the "jagged edges" are not a concern with the filter sizes used in this design.

- Instead of choosing the maximum of the voting function as the disparity estimate, information contained in the curvature of the peak can be used to determine the best disparity estimate. The curvature information can be used to differentiate two peaks with the same magnitude but different levels of confidence in the estimate. A narrow peak may correspond to greater confidence in the disparity estimate than a broader peak.

- Development of post-processing blocks to experiment with various correlation strategies as suggested in Section 3.3.5 which are mentioned briefly below:
  - Use of an elaborate tracking algorithm such as one using a constant-velocity model.
  - Computing location of *SRW* using a probabilistic likelihood estimate instead of pre-determined locations.
  - On-the-fly decision to concatenate the two correlation windows after initialisation stage.
  - Shifting the correlation windows so that they do not straddle object boundaries.

Finally, as pointed out in Chapter 2, global stereo-matching algorithms are fundamentally better in terms of accuracy and quality of results than local algorithms. The implementation of a hardware-based stereo system that uses global matching needs to be explored.

Appendix A

Stereo Rectification

In this appendix, we briefly describe the mathematical background on perspective projection and the rectification technique we use for our stereo system.

A.1 Camera model

Each camera in our stereo-rig is modeled, using the classic pinhole model, by its optical centre C and its image plane $\mathcal{R}$ and a 4×3 perspective projection matrix P.

Let $w = [x\ y\ z]^T$ be the coordinates of a 3-D world point W in the world reference frame and let its projection onto the image plane, M, have the coordinates $m = [uv]^T$ in the image plane. The mapping from 3-D coordinates to 2-D coordinates is a linear transformation in homogeneous coordinates. This is a *perspective projection* and is given (up to a scale factor) by the matrix $\tilde{P}$ as follows:

$$\tilde{m} \simeq \tilde{P}\tilde{w}, \tag{A.1}$$

where $\tilde{m} = [u\ v\ 1]^T$ and $\tilde{w} = [x\ y\ z\ 1]^T$ are the homogeneous coordinates of M and W respectively. The camera is therefore modeled by this perspective projection matrix (PPM) $\tilde{P}$, which can be decomposed into the product, $\tilde{P} = A[R|t]$ using QR factorisation.

The matrix A depends only on the intrinsic parameters of the camera and has the form:

$$A = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1, \end{bmatrix} \tag{A.2}$$

where f_x and f_y are the focal length of the camera in the horizontal and vertical directions respectively, (u_0, v_0) are the coordinates of the principal point, and γ is the skew factor that models non-orthogonal $u - v$ axes.

The extrinsic parameters that represent the camera position and orientation are given by the 3×3 rotation matrix R and translation vector t which bring the camera reference frame onto the world reference frame.

A.2 Rectification

Assuming each of the cameras in the stereo-rig has been calibrated and therefore their PPMs, $\tilde{P}_{o1}$ and $\tilde{P}_{o2}$, are known, the idea behind rectification is to define two new PPMs $\tilde{P}_{n1}$ and $\tilde{P}_{n2}$ and defining a rotation matrix R_{rect} that transforms the original image to conform with the new PPMs. The new PPMs are obtained by rotating the old ones around their optical centres so that the epipolar lines in the two image planes are parallel. In addition, to have horizontal epipolar lines as is preferred, the baseline of the stereo-rig must be parallel to the horizontal axes of the two cameras. The new cameras must also have the same intrinsic parameters to ensure that coupled points have the same vertical coordinates.

The steps for obtaining R_{rect} given the baseline of the stereo-rig $\vec{T}$ and the rotation R between the left and right camera views are as follows:

1. CHOOSE

$$\vec{e_1} = \frac{\vec{T}}{\|\vec{T}\|} \tag{A.3}$$

to make the epipole of the left camera perpendicular to the optic axis

2. CHOOSE

$$\vec{e_2} = \vec{e_1} \times \hat{z} = \frac{[-T_y, T_x, 0]^T}{\sqrt{T_x^2 + T_y^2}} \tag{A.4}$$

to make $\vec{e_2}$ perpendicular to both $\vec{e_1}$ and the optic axis.

3. $\vec{e_3} = \vec{e_1} \times \vec{e_2}$ (no choice here)

4. CREATE the rotation matrix

$$R_{rect} = \begin{bmatrix} \vec{e_1^T} \\ \vec{e_2^T} \\ \vec{e_3^T} \end{bmatrix} \tag{A.5}$$

The points in the left image are remapped by applying the transformation R_{rect} and points in the right image are remapped using $R_{rect}R^T$. The points from the original image plane will not lie in the new image plane so we need to re-apply projection of f/Z' to both image transformations.

Appendix B

Simplification of Voting Function

In this appendix, we reproduce the discussion on the simplifications made to the voting function in [?] and its effects on the disparity estimates.

The voting function in the original LWPC algorithm is computed as follows:

$$C_{j,s}(x,\tau) = \frac{W(x) \otimes [O_l(x)O_r^*(x+\tau)]}{\sqrt{W(x) \otimes |O_l(x)|^2}\sqrt{W(x) \otimes |O_r(x)|^2}} \;, \tag{B.1}$$

The voting function C at location x for a candidate disparity of τ is computed by convolving the Gaussian window $W(x)$ with the inner product of $O_l(x)$ and $O_r^*(x+\tau)$, where $O_l(x)$ is the complex-valued G2/H2 filter output for the left image and $O_r^*(x+\tau)$ is the conjugate of the right image G2/H2 filter output shifted by τ pixels horizontally. The result is then divided by square root of convolution of $W(x)$ with the square of the amplitude of both $O_l(x)$ and $(O_r(x)$.

In practice, only the real part of $C(x,\tau)$ needs to be computed because at the true disparity the real part is at its maximum and the imaginary part is close to zero. The disparity is then estimated by finding the peak in the real part of $C(x,\tau)$. The Gaussian window, w, and denominator in Equation B.1 are always real-valued. The real part of $O_l(x)(O_r^*(x+\tau))$ is computed as follows:

$$\Re[O_l(x)O_r^*(x+\tau)] = \Re[O_l(x)]\Re[O_r(x+\tau)] - \Im[O_l(x)]\Im[O_r(x+\tau)] \tag{B.2}$$

The hardware architecture for computing $\Re[C(x,\tau)]$ is illustrated in Figure B.1. It requires seven multipliers, one square root block, one divider, three adders and three Gaussian windowing blocks. A parallel implementation requires multiple implementations of this block. A phase-correlation unit that covers an epipolar search area of p pixels at a single time instance requires p copies of this block at Scale 1, $\lceil p/2 \rceil$ at Scale 2, and $\lceil p/4 \rceil$ at Scale 4 for each of the three orientations. An epipolar search band of twenty pixels would then require 105 of these voting function blocks and any savings made in the logic resource usage of the voting function block will be magnified 105 times.

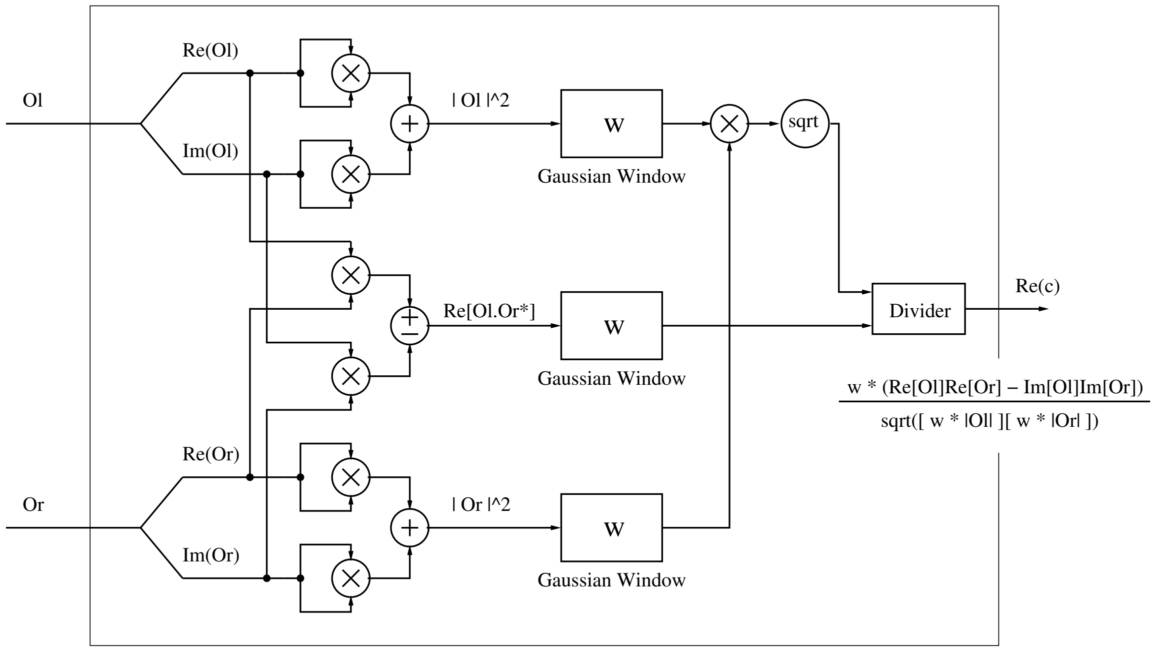


Figure B.1: Realisation of the real part of the original voting function. Courtesy of [?].

The approach taken in [?] is illustrated in Figure B.2. The Gaussian window is first moved to after the divider block resulting in one Gaussian windowing block instead of three per voting function block so that for $p = 20$, the number of Gaussian window blocks required are reduced to 105 from 315. Since Gaussian filtering is a linear operation, they further reduce the number of Gaussian window blocks required by performing Gaussian filtering on the sum of the correlation results from all three orientations. This reduces the number of Gaussian window blocks required to 35.

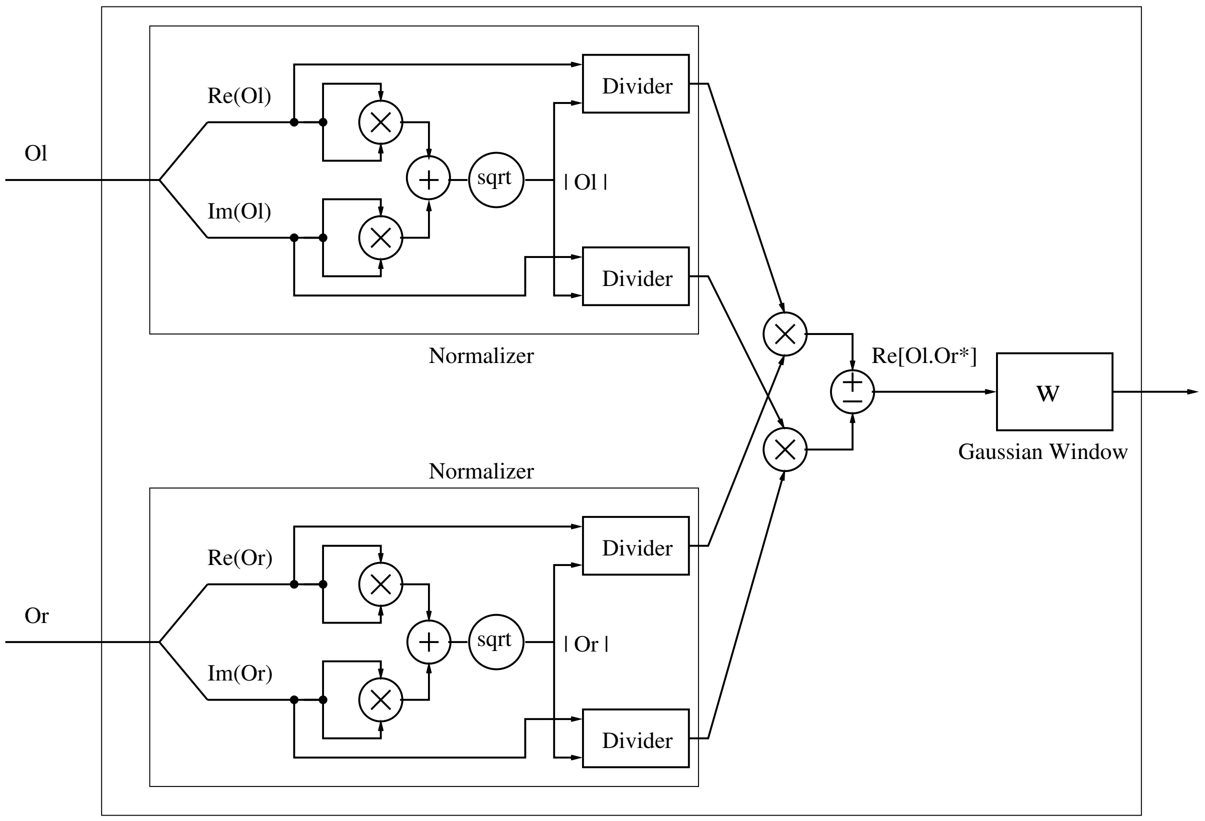


Figure B.2: Modified voting function unit with shared Gaussian window and normalisation unit. Courtesy of [?].

Normalisation is another resource intensive operation. To achieve resource savings, the normalisation block is moved outside the voting function unit so that a single normalisation unit is shared across all voting function blocks. Further, the normalisation is performed using an L_1 norm instead of an L_2 norm. The L_1 norm of a 2-D vector A is given by:

$$\|A\|_1 = |\Re(A)| + |\Im(A)| \tag{B.3}$$

and the L_2 norm of the 2-D vector A is given by:

$$\|A\|_2 = \sqrt{\Re(A)^2 + \Im(A)^2}. \tag{B.4}$$

Table B.1 compares the number of blocks required with and without these modifications. Sharing the normalisation block and changing the location of Gaussian window

	Original Architecture	**Modified Architecture**
Multipliers	735	210
Dividers	105	36
Square roots	105	-
Adders	210	123
Gaussian Windows	210	35

Table B.1: Summary of the number of basic blocks required for an epipolar search band of 20 pixels in three orientations for the original and modified voting function units. Courtesy of [?].

reduces the total number of multipliers, dividers and square roots in the correlation unit by over 65 %.

The effects of these modification on the disparity map are showing in Figure B.3. The stereo image pair is showing in Figure B.3 (a) and (b). The depth map using the original voting function unit is shown in (c) and the depth map from the modified voting function is shown in (d). In most of the regions, the two maps have the same depth values, but the depth map in (d) contains slightly more noise compared with (c).

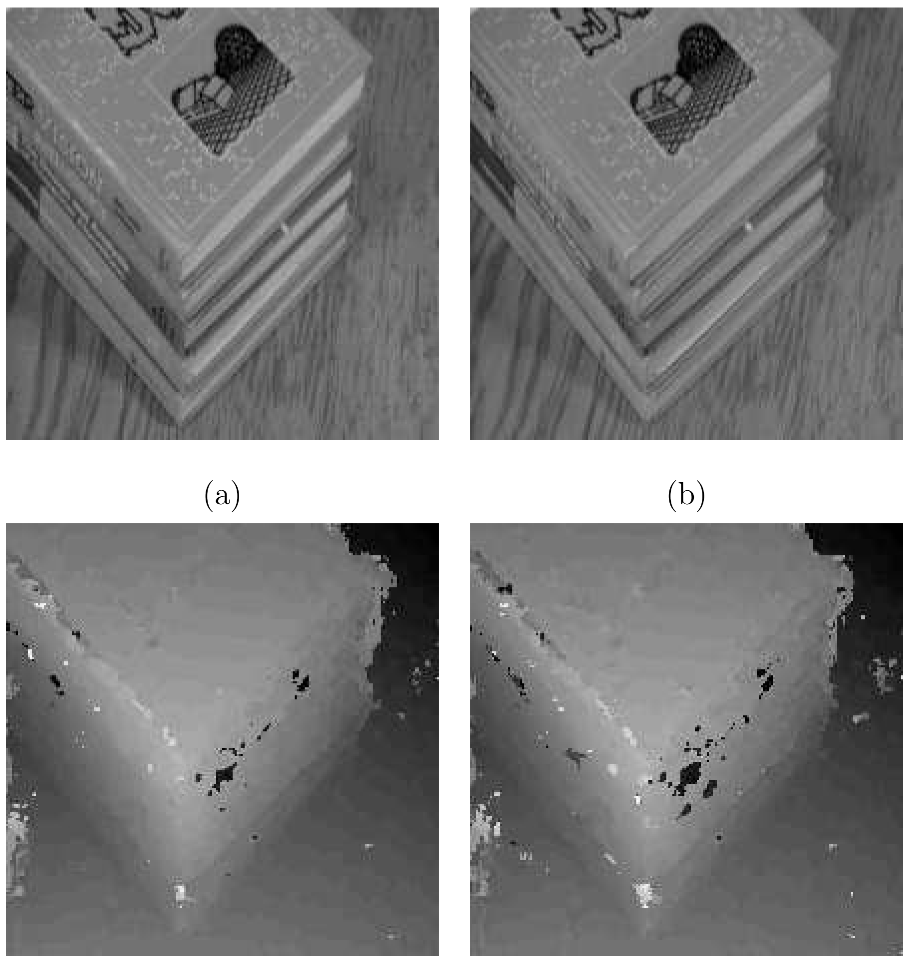


Figure B.3: Effects of using L_1 norm instead of L_2 norm, sharing the normalisation operation and changing the location of the Gaussian window on the final depth map of the ‘books’ stereo images. (a) Left image. (b) Right image. (c) Depth map from the original voting function. (d) Depth map from the modified voting function. Courtesy of [?].

References

[1] Mobileye product webpage. http://www.mobileye.com.

[2] G. Balakrishnan, G. Sainarayanan, R. Nagarajan, and S. Yaacob. A stereo image processing system for visually impaired. *International Journal of Signal Processing*, 2(2):136–145, 2005.

[3] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, August 2005.

[4] Peter J. Burt. A pyramid-based front-end processor for dynamic vision applications. *Proceedings of the IEEE*, 90(7):1188–1200, July 2002.

[5] Altera Corporation. Stratix devices. http://www.altera.com/products/devices/stratix/stx-index.jsp, 2003.

[6] S. Crossley, A. J. Lacey, N. A. Thacker, and N. L. Seed. Benchmarking of bootstrap temporal stereo using statistical and physical scene modelling. In *Proceedings of the British Machine Vision Conference*, pages 346–355, 1998.

[7] S. Crossley, N. A. Thacker, and N. L. Seed. Robust stereo via temporal consistency. In *Proceedings of the British Machine Vision Conference*, pages 659–668, 1997.

[8] Ahmad Darabiha. Video-rate stereo vision on reconfigurable hardware. Master's thesis, Department of Electrical & Computer Engineering, University of Toronto, 2003.

[9] Ahmad Darabiha, Jonathan Rose, and W. James MacLean. Video-rate stereo depth measurement on programmable hardware. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision & Pattern Recognition*, volume 1, pages 203–210, Madison, WI, June 2003.

[10] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Gérard Berry, Jean Vuillemin, Patrice Bertin, and Catherine Proy. Real time correlation-based stereo: Algorithm, implementations and applications. Technical Report Research Report 2013, INRIA Sophia Antipolis, August 1993.

[11] Josh Fender. Transmogrifier 4 preliminary information. http://www.eecg.toronto.edu/˜fender/tm4/sointroduction.shtml, August 2003.

[12] David J. Fleet. Disparity from local weighted phase correlation. In *International Conference on Systems, Man and Cybernetics*, volume 1, pages 48–54, 1994.

[13] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[14] Dave Galloway, Michael van Ierssel, Paul Chow, and Jonathan Rose. Tm-3 documentation. http://www.eecg.toronto.edu/˜tm3/, 2003.

[15] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[16] Heiko Hirschmüller, Peter R. Innocent, and Jon Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1/2/3):229–246, 2002.

[17] K. M. Hou and A. Belloum. A reconfigurable and flexible parallel 3d vision system for a mobile robot. In *IEEE Workshop on Computer Architecture for Machine Perception*, New Orleans, Louisiana, December 1993.

[18] Takeo Kanade, Atsushi Yoshida, Kazuo Oda, Hiroshi Kano, and Masaya Tanaka. A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings of the 15th IEEE Computer Vision & Pattern Recognition Conference*, pages 196–202, San Francisco, June 1996.

[19] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proc. Int. Conf. on Computer Vision, ICCV-2001*, pages 508–515, 2001.

[20] Kurt Konolige. Small vision systems: Hardware and implmentation. In *Proceedings of the Eighth International Symposium on Robotics Research (Robotics Research 8)*, pages 203–212, Hayama, Japan, October 1997.

[21] Uwe Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2004.

[22] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard M. Anner. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1/2/3):79–88, 2002.

[23] M. Okutomi and Y. Katayama. A simple stereo algorithm to recover precise object boundaries and smooth surfaces. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision—SMBV'01*, pages 158–165, 2001.

[24] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 343–350. IEEE Computer Society, IEEE Computer Society Press, 1996.

[25] D. Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002.

[26] Richard Szeliski. *Bayesian Modeling of Uncertainty in Low-level Vision.* Kluwer Academic Publishers, Boston, Massachusetts, 1989.

[27] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[28] G. van der Wal and P. Burt. A VLSI pyramid chip for multiresolution image analysis. *Int. Journal of Computer Vision*, 8:177–190, 1992.

[29] J. Woodfill and B. Von Herzen. Real time stereo vision on the parts reconfigurable computer. In *5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 201–210, 1997.

[30] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition*, pages 211–218, Madison, Wisconsin, June 2003.

[31] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the 3rd European Conference on Computer Vision*, pages 150–158, May 1994.