Design of a Real-time Image-based Distance Sensing System by Stereo Vision on FPGA

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan
Saskatoon

By
Zheng Qian

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical and Computer Engineering
College of Engineering
University of Saskatchewan
57 Campus Drive
Saskatoon, Saskatchewan
Canada
S7N 5A9

Abstract

A stereo vision system is a robust method to sense the distance information in a scene. This research explores the stereo vision system from the fundamentals of stereo vision and the computer stereo vision algorithm to the final implementation of the system on a FPGA chip. In a stereo vision system, images are captured by a pair of stereo image sensors. The distance information can be derived from the disparities between the stereo image pair, based on the theory of binocular geometry. With the increasing focus on 3D vision, stereo vision is becoming a hot topic in the areas of computer games, robot vision and medical applications. Particularly, most stereo vision systems are expected to be used in real-time applications.

In this thesis, several stereo correspondence algorithms that determine the disparities between stereo image pair are examined. The algorithms can be categorized into global stereo algorithms and local stereo algorithms depending on the optimization techniques. The global algorithms examined are the Dynamic Time Warp (DTW) algorithm and the DTW with quantization algorithm, while the local algorithms examined are the window based Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD) and Census transform correlation algorithms. With analysis among them, the window based SAD correlation algorithm is proposed for implementation on a FPGA platform.

The proposed algorithm is implemented onto an Altera DE2 board featuring an Altera Cyclone® II 2C35 FPGA. The implemented module of the algorithm is simulated using ModelSim-Altera to verify the correctness of its functionality. Along with a pair of stere image sensors and a LCD monitor, a stereo vision system is built. The entire system realizes a real-time video frame rate of 16.83 frames per second with an image resolution of 640×480 and produces disparity maps in which the objects are clearly distinguished by their relative distance information.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Kunio Takaya. Professor Kunio Takaya is a knowledgeable and nice professor. During the research, I benefit a lot from his professional knowledge on multimedia and computer software. The completeness of this thesis would not have been possible without his advice, guidance and support. I would also like to thank Professor Kunio Takaya for his reference for me when I am looking for jobs. His advice is valuable.

I would like to express my sincere gratitude to my parents, Jingshan Qian and Xiaonan Yu, for their love and endless support in my life.

I would like to thank my academic colleagues, Liu Han, Zhuo Wang, Xingxing Jin and Zhongkai Chen, for their creative opinions, valuable advice and generous helps during my research. Also, special thanks to all my friends in Saskatoon, for those happy hanging outs.

Last but not the least, I would like to acknowledge Department of Electrical and Computer Engineering and NSERC for the financial support.

This thesis is dedicated to my beloved parents.

Contents

List of Tables

List of Figures

List of Abbreviations

ADC	Analog-to-digital Converter
ASIC	Application-specific Integrated Circuit
AVI	Audio Video Interleaved
CCD	Charge-coupled Device
CMOS	Complementary Metal-oxide-semiconductor
CPU	Central Processing Unit
DAC	Digital-to-analog Converter
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
DTW	Dynamic Time Warp
FPGA	Field Programmable Gate Array
FPS	Frames per Second
GPU	Graphics Processing Unit
HDL	Hardware Description Language
I^2C	Inter-integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
LCD	Liquid Crystal Display
LEs	Logic Elements
MMX	Multimedia Extension
PC	Personal Computer
PLL	Phase-lock Loop
RAM	Random Access Memory
RMS	Root Mean Square
SAD	Sum of Absolute Differences
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
SSD	Sum of Squared Differences
VGA	Video Graphics Array

Chapter 1

Introduction

1.1 Motivation of Stereo Vision Based Distance Sensing

In the recent years the needs for 3D technology result in a dramatic development in the modeling and understanding of computer stereo vision. In computer stereo vision, multiple views are realized using two or more cameras mounted at different angles and position to capture a scene. The images captured from these cameras are called stereo images, from which the depth information can be derived so that the 3D reconstruction of a scene is possible. Specifically, a disparity map which represents the depth information can be derived from a set of stereo images. In general, stereo vision provides a precision approach to grab the depth information.

Stereo vision produces the depth information by determining the correspondences among a set of stereo images. With the fundamentals of multiple-view geometry, it has been proved that the precise orthogonal distance from an object point to the baseline of multiple views can be derived from at least two projective points of the object point among the views if enough parameters about the view points are given [9]. Given that multiple views of a scene are available, in a stereo vision problem, the first step to measure the distance of an object point is to find out its projective points in all views. The process to find out the projective points is called the search for stereo correspondences. For the depth information of an entire scene, the distance of every object point is computed by this way. In a digital image, a scene is represented by discrete pixels, so the correspondences of objective points becomes the correspondences of pixels. Once all pixel correspondences are determined, the disparity for every pixel is computed and finally a dense disparity map is produced

by all disparities for the entire scene.

Motivated by the effectiveness of stereo vision approach, a stereo vision based distance sensing system is built in our research. For extracting dense and reliable depth information from a scene, the stereo vision approach is usually computationally expensive. Meanwhile, modern applications such as robot navigation and vehicle surveillance require real-time processing. The challenge is the tradeoff between precision and processing time. Aiming to tackle this challenging task, the objective of our work is to produce the dense disparity maps in real time as video.

1.2 Common Strategies for Stereo Vision Systems

To build a stereo vision system, different strategies can be applied with regard to both stereo correspondence algorithms and implementation platforms.

Since all stereo correspondence algorithms are processing based on raster scanlines, the stereo correspondence algorithms can only be effective under an assumption that all scanlines from one stereo image must correspond to the scanlines from any other stereo image. In order to realize this assumption, there are two ways. One way is to project all pixels in the stereo images onto a new set of coordinates so that scanlines of the two images correspond to each other. The process of this projection method is called rectification. The other way is to mount the stereo cameras with their lens axes in parallel and their projective planes being coplanar so that the scanlines from any stereo image are corresponded.

Stereo correspondence algorithms are commonly categorized into global algorithms and local algorithms depending on the optimization technique used within the algorithms [21]. Typical global algorithms include dynamic programming [7][16][20][2], graph cuts [3] and belief propagation [22], while most local algorithms are window-based correlation algorithms [11][15][14][13].

With regard to the implementation platforms, most stereo vision systems were realized by plain software and operated by CPU in a PC. However, with the increasing demands for large image resolution and fast processing time, high-performance GPU is introduced to implement stereo vision system. Meanwhile, hardware implementations on FPGA and ASIC are also developed to meet these demands. Compared to

software implementations, hardware implementations are advantageous to speed up the processing with the parallelism.

1.3 Current State of Related Technologies

In order to explore the stereo vision systems in our research work, some available stereo vision systems are reviewed. The descriptions of the systems focus on several aspects, including the system platform, the basic stereo correspondence strategy, the image size, the number of evaluated disparity levels and the frame rate achieved.

Forstmann et al. [7] introduce a stereo system based on dynamic programming. The system applies a two-step dynamic programming based method, and it is implemented on an AMD AthlonXP 2800+ CPU. The system uses several techniques for speed optimization, such as the reduction of the matrix size in dynamic programming algorithm, implementing the main part of the stereo algorithm using the assembly language by integrating the Multimedia Extension commands (MMX), applying a coarse to fine strategy, and the compiler optimizations. The system can reach a frame rate of 12.3 fps at a VGA resolution with 50 disparity levels, and 58.5 fps at 320× 240 with 32 disparity levels. The system is evaluated with various numbers of disparity levels, of which the maximum is 100 levels. As the number of disparity levels increases, the frame rate gets lower.

The DeepSea G2 stereo vision system is developed by Woodfill et al [28]. The system is implemented on a customized ASIC, an FPGA, a DSP/coprocessor, a Power PC running Linux and an Ethernet connection. The system decomposites the computational work into smaller tasks, which are then assigned to specific hardware devices. A Census transform based correlation algorithm is applied in this system. The stereo correspondence module of this system can reach 200 fps at 512×480 resolution with 52 disparity levels. And it is mentioned that the overall system runs at 30 fps for an example application in the authors' paper.

A multiple camera stereo system is developed by Kanade et al [14]. The system contains two subsystems: a Laplacian of Gaussian (LoG) subsystem and a SSAD subsystem. The input stereo image pairs are filtered by a LoG mask in order to extract the edge features before the stereo correspondence algorithm, which is based

on the Sum of SAD (SSAD) with a window-based correlation algorithm. The system is implemented on a Texas Instruments C40 DSP array, and has stereo head with up to five cameras. It performs at 30 fps for a 200×200 image size with 32 disparity levels.

Humenberger et al. [11] introduce a stereo system using the Census transform based correlation algorithm. The algorithm uses a sparse Census transform mask instead of the conventional dense mask, which improves the probability of correct matches. The system is implemented on various platforms, including implementation on PC with Intel Core Duo 2.14 GHz CPU, a TI TMS320C6461 1GHz DSP, and a NVIDIA GeForce GTX 280 GPU. The implementation on GPU provides the best performance, an estimated 573.7 fps at a 320×240 resolution.

Diaz et al. [5] introduce a novel technique for stereo correspondence based on phase measurement. The system is implemented on a Xilinx Virtex-II FPGA and can run at 52 fps for a 1280×960 resolution.

Point Grey Research Inc. [17] develops a PC-based stereo system, which is implemented on a 2.8 GHz Intel Pentium 4 CPU. It can reach 83 fps at a 320×240 image size with 32 disparity levels, and 4.4 fps at 640×480 with 96 disparity levels.

Videre Design [27] produces a stereo system called Stereo on a Chip (STOC), which uses a window-based SAD stereo algorithm. It is interfaced by IEEE 1394, and it can reach 30 fps at VGA size with 64 disparity levels.

In review of the available stereo systems, we can notice that most stereo systems are based on high-speed platforms such as FPGA, GPU and DSP, because of the massive demands for real-time stereo vision applications nowadays. Meanwhile, correlation algorithms are the most common methods for fast processing.

1.4 Target Methods Studied in This Thesis

This research focuses on two-view stereo vision. At the first stage, a couple of stereo correspondence algorithms are explored and analyzed. Specifically, global algorithms that are analyzed include the dynamic time warp (DTW) algorithm and its extension, DTW with quantization algorithm [25][18]. Meanwhile, local algorithms are several window-based correlation algorithms such as Sum of Squared Differences (SSD), Sum

of Absolute Differences (SAD) and Census transform correlations. The performances of these algorithms are analyzed and compared by varying their parameters.

The target platform to implement this stereo vision system is an Altera DE2 board featuring an Altera Cyclone® II 2C35 FPGA [1]. By this means, the stereo vision system is modeled using Verilog hardware description language (HDL) and realized by logic elements. Two Terasic TRDB-DC2 image sensors [26] are mounted to build a pair of stereo cameras and a LCD monitor is used to display the produced disparity maps.

1.5 Outline of the Thesis

Chapter 1 introduces the motivation of this research work, common strategies for stereo vision systems, current state of related technologies as well as the objective of this research work. Chapter 2 introduces the multiple view geometry on which the 3D reconstruction is based, and also some reviews on the fundamentals of stereo correspondence algorithms. In Chapter 3, several common stereo correspondence algorithms are discussed in more details. Chapter 4 analyzes the performances of selected stereo correspondence algorithms by varying their parameters. Chapter 5 demonstrates the FPGA implementation of the stereo vision system using the proposed stereo algorithm. Chapter 6 presents a functional simulation of the implemented stereo system using the simulation tool ModelSim-Altera simulator. Every single disparity map produced by the simulation is used to evaluate the capability of the system to produce good-quality disparity maps. Also, an evaluation of the run-time performance of the implemented stereo vision system is given. Chapter 7 concludes this thesis and discusses our future work.

Chapter 2

Backgrounds

2.1 Stereo Vision Geometry

Multiple view geometry is the theoretical fundamental of the reconstruction of a 3D scene structure. Our research focuses on a two camera configuration. Hence, only the relevant two-view geometry is introduced and discussed in detail here.

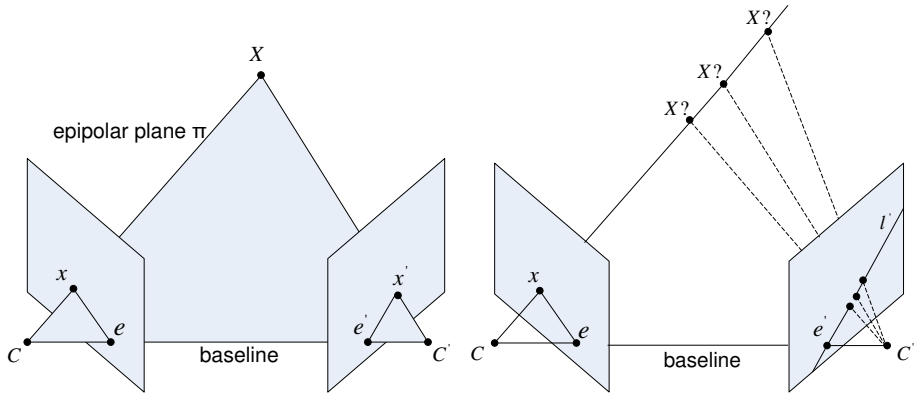


Figure 2.1: Point correspondence geometry. Left: A 3D point X forms an epipolar plane Π with the projective points x and x', the epipoles e and e', the optical centers C and C'. Right: Given a projective point x, its corresponding point x' lies on the epipolar line l'.

In the two-view geometry, the intrinsic projective between the two views is called epipolar geometry [9]. The epipolar geometry is independent of scene structure, and only depends on the cameras' internal parameters and their relative position.

Algebraically, this geometry is represented by the fundamental matrix F. The fundamental matrix is a 3×3 matrix of rank 2. With the projective points of a 3D point X on the image plane of each camera, x, x', the fundamental matrix F satisfies the relation

$$x'^{\top} F x = 0. \tag{2.1}$$

The epipolar geometry has the property that benefits the searching for corresponding points in stereo correspondence. Thus, we start from introducing the epipolar geometry. As shown in Figure 2.1, X is a 3D point in space, and x and x' are

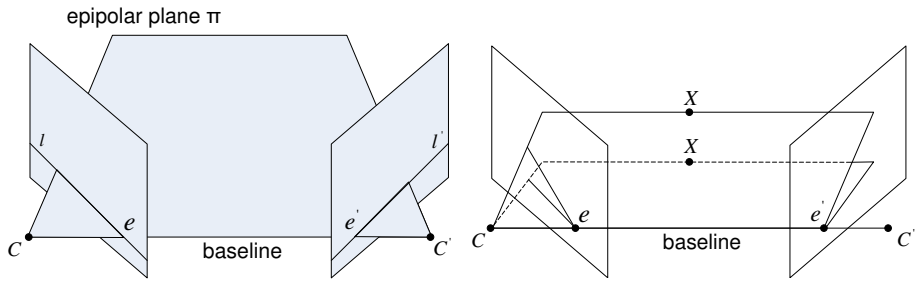


Figure 2.2: Epipolar plane. Left: An epipolar plane Π with epipoles e and e', epipolar lines l and l'. Right: The epipolar planes rotate about the baseline as the position of point X varies.

the projective points of X on each image plane respectively. c and c' are the camera centers. The line connecting the two camera centers is defined as the baseline. It is evident that the space point X, the projective points x and x', and the camera centers c and c', are coplanar. Denote this plane as π. It also shows that the rays back-projected from x and x' intersect at X, and they are coplanar in π as well.

Suppose that we only know the image point x, the stereo correspondence algorithm is to search for the image point x' in the second image. As we can see, the plane π can be determined by the baseline and the ray defined by x and C. We also know that the point x' should lie in the plane π as well. Hence, the image point x' lies on a line l', which is the intersection line of the plane π and the second image plane. From a different point of view, this line l' is also the projection of the ray back-projected from x in the second image plane. Thus, this is a map from a point in the first image to its corresponding line in the second image. In terms of stereo correspondence, to search for the point corresponding to x need not cover the entire second image but can be restricted to the line l'.

There are some terminologies in the epipolar geometry [9]. Firstly, the epipole is the point of intersection of the baseline with the image plane. We can see that the epipole is the projection of one camera center on the other image plane. Also, it is the vanishing point of the baseline direction. Point e and e' are the epipoles shown in Figure 2.2. An epipolar plane contains the baseline and the 3D point X. It is shown as the plane π in Figure 2.2. As the position of the point X varies, the epipolar

planes "rotate" about the baseline. The intersection line of an epipolar plane with the image plane is an epipolar line. All epipolar lines intersect at the epipole.

Once a set of image point correspondences $x_i \leftrightarrow x_i'$ is determined, the reconstruction of the scene structure can be started. Since our current research focuses on the stereo correspondence algorithms rather than the actual reconstruction of the 3D points, only a brief conceptual approach to reconstruction is described here.

Suppose that only the image point correspondences are known, but cameras' internal parameters and their relative position are not known. The reconstruction task is to find the camera matrices P and P', and then the 3D points X_i, which satisfy

$$x_i = PX_i \tag{2.2}$$

$$x_i' = P'X_i \tag{2.3}$$

for all i.

Given the image point correspondences $x_i \leftrightarrow x_i'$, the fundamental matrix F is computed at first, with the condition $x_i'Fx_i = 0$ for all i. Next, a pair of camera matrix P and P' can be computed with their corresponding fundamental matrix F. Based on triangulation, the two rays back-projected from the image corresponding points x and x' lie in a common epipolar plane, so the two rays will intersect in some point X, and it satisfy $x = PX$, $x' = P'X$. However, the 3D points on the baseline cannot be determined by this approach, since the two back-projected rays are collinear and intersect along their whole length.

With the epipolar constraint, the search for point correspondences over the entire image is restricted to the search on matching epipolar lines. Moreover, the calculations associated with stereo matching algorithms are considerably simplified when the pairs of stereo images are rectified. Rectified stereo images share a common image plane parallel to the baseline joining the two optical centers [8]. In this case, the rectified epipolar lines are scanlines of the images with an appropriate choice of coordinate system. To realize the rectification, a general way is to project the original images onto a new image plane. This is shown in Figure 2.3. Rectification is a crucial pre-processing to simplify the search for point correspondences. In this thesis, all discussions on stereo matching algorithms in the following chapters are

with the assumption of rectified stereo images. When the two stereo cameras are

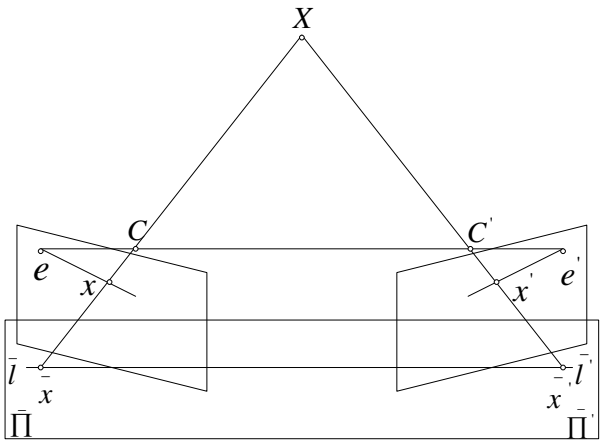


Figure 2.3: Rectification. Two image planes are projected onto a common plane. The projective points x and x' are re-projected to point $\bar{x}$ and $\bar{x'}$.

mounted with their image planes being coplanar and their focal lens axes in parallel, the stereo images are rectified. Suppose there is no rotation with either of the image plane, the computation of the distance information can be simplified in this case. Figure 2.4 illustrates the geometry for this case. Figure 2.4 only shows the geom-

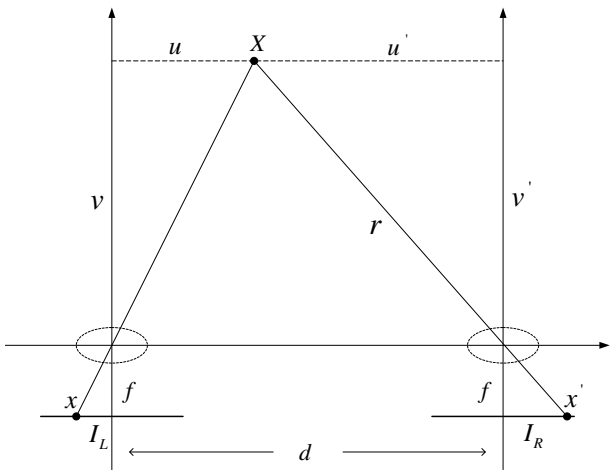


Figure 2.4: Rectified epipolar geometry. The orthogonal distance v of a point X to the focal lens is computed using the focal length f, baseline distance d and the disparity $|I_L - I_R|$.

etry in horizontal directions. Denote that the focal length of both cameras' lenses is f, and the length of the baseline is d. A 3D point X lies in the 3D space at the horizontal position (u, v) with respect to the coordinates of the left camera, and at the horizontal position (u', v') with respect to the coordinates of the right camera.

X is projected onto the left image plane at point x, and projected onto the right image plane at point x'. The disparity for the pair of corresponding points x and x' is defined as $|I_R - I_L|$, where I_L is the displacement of x to the v axis on the left image plane, and I_R is the displacement of x' to the v' axis on the right image plane.

Based on the trigonometry, it is simple to get

$$\frac{-I_L}{-f} = \frac{u}{v} \tag{2.4}$$

$$\frac{I_R}{-f} = \frac{-u'}{v'} \tag{2.5}$$

$$u + (-u') = d. \tag{2.6}$$

Thus, v is computed as

$$v = f \times \frac{d}{|I_R - I_L|}. \tag{2.7}$$

Hence, with the information about the cameras' relative position and their focal length, the distance v with respect to the left image coordinates can be computed. It is evident that the distance v is inversely relative to the disparity $|I_R - I_L|$.

2.2 Related Work in Stereo Matching Algorithms

In computer vision, the projection of the scene in the image plane is represented by pixel values. In terms of a grayscale image, each pixel is valued by a single component usually called the intensity. Areas with the continuous intensities are considered as the object surfaces, while edges or corners of an object are those areas with intensity discontinuities. This is based on the assumption of the Lambertian reflection, with which the appearance of the object surface to an observer does not vary with viewpoints. Similarly, color image is represented by pixel values consisting of multiple component rather than one. For example, a pixel value is determined by luminance and chrominance components in the YUV format, and for RGB format, there are red, green and blue components. In terms of the stereo correspondence algorithms, the intensity or the color components of each pixel are used to search for the best matches between the stereo images.

Generally, a stereo correspondence algorithm searches for the best matched pixels between the input stereo images. In a two-view case, the input of an algorithm are a

pair of stereo images captured from two digital cameras mounted on a stereo camera rig. For a stereo pair, one image is regarded as the primary image, and the other is regarded as the secondary image. A reference image is necessary. It could be one of the stereo images or a cyclopian view between the stereo pair. In our case, we will refer to the primary image as the reference image. Thus, given a pixel in the primary image, its corresponding pixel is searched in the secondary image. For the output of the stereo correspondence algorithms, most of the algorithms produce a univalued disparity function $d(x, y)$ with respect to the reference image, where x and y are the coordinates of pixels in the reference image. The function forms a disparity map which has the same image size as the reference image. There are also algorithms that produce multi-valued disparity functions, but we only focus on those with univalued disparity functions in our work. Algorithms with multiple camera inputs basically extend from the algorithms with two-camera inputs, and can be generalized to arbitrary camera configurations.

Recall the case where two image planes of the two stereo cameras are coplanar. In this case, the epipolar lines on the image plane are in parallel. With epipolar lines in parallel, the stereo images are regarded as the rectified stereo images. When the stereo images have been rectified, the calculations associated with the stereo algorithms are often considerably simplified. In the rectified images, the epipolar lines are the scanlines of the image, and they are also parallel to the baseline, with an appropriate choice of coordinate system. Therefore, in a stereo algorithm, the search for the corresponding pixel of a pixel in the primary image can be restricted to its corresponding scanline in the secondary image. Image rectification can be realized by projecting the original image onto a new image plane. But with the special case discussed previously, the stereo images are already rectified. This is illustrated in Figure 2.5.

In the case of rectified stereo images and an appropriate coordinate system, the correspondence between a pixel (x, y) in the primary image and its corresponding pixel (x', y') can be described as

$$x' = x + d(x, y) \tag{2.8}$$

$$y' = y \tag{2.9}$$

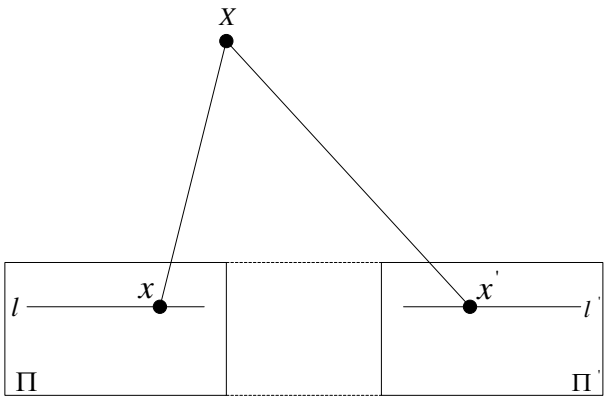


Figure 2.5: A pair of rectified stereo images. Rectified stereo images share a common plane and the epipolar lines are scanlines of the images.

where $d(x, y)$ is the disparity value with respect to the primary image. A space (x, y, d) is defined as the disparity space. The disparity space can be used for a function that represents the confidence or log likelihood of a particular match implied by $d(x, y)$.

Stereo correspondence algorithms can be categorized by different characteristics. According to the number of pixels processed, the algorithms can be divided into two types: feature-based and area-based algorithms [11]. Feature-based algorithms firstly extract the feature pixels in the images, such as the object edges or corners, and then it searches for correspondence between these feature pixels. Since it only finds the disparities of the feature pixels, it results in a sparse disparity map, and the disparities of other non-feature pixels need to be determined by some other ways. On the other hand, area-based algorithms search for correspondence for every pixel in the images. Thus, they compute disparities for all pixels and generate a dense disparity map.

A more common categorization is based on the optimization methods used in the stereo correspondence algorithms. The algorithms are then divided into global and local algorithms [21]. A global algorithm applies a global optimization strategy. Such algorithm uses all pixels in the images to construct a global cost function and minimizes it to find out the best matches. Common global algorithms include dynamic programming, graph cut, belief propagation, etc. In contrast, a local algorithm uses local pixel features to determine the correspondences. The disparity computation for a given pixel only depends on pixels within a finite window around it. Usually

the correlation between these windows of pixels is used as the cost function. The window-based correlation algorithm is a typical local algorithm.

In general, a stereo correspondence algorithm contains several steps. Firstly, a pre-processing step is usually applied to eliminate the noises in the images. Second, the matching costs for each pixel at each disparity level are computed. The matching cost function indicates the likelihood of a correct match. Some common matching cost computation methods for a pixel (x, y) are shown below. They are the Sum of Absolute Difference, the Sum of Squared Differences, the Normalized Cross Correlation and the Zero Mean Sum of Absolute Differences.

$$SAD = \sum_{i=n} \sum_{j=m} |I_1(x+i, y+j) - I_2(x+d+i, y+j)| \tag{2.10}$$

$$SSD = \sum_{i=n} \sum_{j=m} (I_1(x+i, y+j) - I_2(x+d+i, y+j))^2 \tag{2.11}$$

$$NCC = \frac{\sum\limits_{i=n} \sum\limits_{j=m} I_1(x+i, y+j) I_2(x+d+i, y+j)}{\sqrt{\sum\limits_{i=n} \sum\limits_{j=m} I_1(x+i, y+j)^2 \sum\limits_{i=n} \sum\limits_{j=m} I_2(x+d+i, y+j)^2}} \tag{2.12}$$

$$ZSAD = \sum_{i=n} \sum_{j=m} |(I_1(x+i, y+j) - \bar{I_1}) - (I_2(x+d+i, y+j) - \bar{I_2})|, \tag{2.13}$$

where

$$\bar{I} = \frac{1}{nm} \sum_{i=n} \sum_{j=m} I(x+i, y+j). \tag{2.14}$$

I_1 is the primary image and I_2 is the secondary image. The matching cost for pixel (x, y) is calculated within a $n \times m$ block surrounding it, and d is the disparity levels. Finally, the best match for each pixel is determined by an optimization method, which can be a global or local strategy.

Chapter 3

Stereo Matching Algorithms

In an image-based distance sensing system, a key part of it is to find the best matched pixels between the pairs of stereo images. This is achieved by a stereo matching algorithm. Different stereo matching algorithms result in different correctness of matches, computational complexities, etc. In this chapter, we examine a couple of different stereo matching algorithms and find out their advantages and disadvantages when used in a computer stereo vision system. Commonly, the stereo matching algorithms can be categorized into two types: global and local algorithms, depending on whether a global or local optimization technique is used in the algorithm.

3.1 Global Algorithms

A global stereo matching algorithm applies a global optimization technique, which constructs a global cost function. The cost function is then minimized so as to determine the best matches with minimum costs. Common global algorithms include dynamic programming, graph cuts and belief propagation.

Dynamic programming [7][16] is a method that finds an optimal path through all possible matches on a scanline basis. In a recursive manner, the path with the lowest matching cost is chosen and finally the recursion leads to an optimal solution to the pixel correspondences.

Another global optimization approach is called graph cuts [3]. Similar to the dynamic programming approach that finds an optimal path for the corresponding scanlines, it builds up a weighted graph and seeks for the optimal flow of the graph. The difference from dynamic programming is that the graph cuts method considers the consistencies in both horizontal and vertical directions rather than the only

horizontal direction in dynamic programming. However, the computation of the optimal flow in graph cuts is massive. Therefore, it is not appropriate for fast computing.

A third global matching algorithm is the belief propagation [22]. It is an iterative strategy that uses Markov random fields (MRFs) to determine the best matches. Fundamentally, the stereo matching is modeled by three coupled Markov random fields: a smooth disparity field, a spatial line process representing the presence or absence of depth discontinuities, and a spatial binary process indicating occlusion regions. The stereo model is then built in mathematics and the optimal solution to the model is to be found. The solution is computed using a Markov network which is an undirected graph. In the graph, a node is defined to a disparity level and holds its matching cost. Also, a belief value of a node is defined as the sum of its matching cost and the received belief values from other nodes. At every iteration, each node sends its belief value to its connecting nodes and the belief value of each node is updated. Finally, the best match is determined by the lowest belief value.

In review of other researchers' work, it is concluded that dynamic programming is a more efficient global algorithm for fast processing. In the following sections, we will examine dynamic programming in more details.

3.1.1 Dynamic Programming

Dynamic programming is a computer algorithm that is usually used in optimization problems. It solves the problems by combining the solutions to subproblems in a recursive manner [4]. In dynamic programming, the solutions to subproblems are stored in a table once computed, and will be never recomputed again when the same subproblem is encountered. When computing the solution to the problem, the solutions to subproblems just need to be looked up in the table. This technique reduces the computational complexity by avoiding the work of recomputation of the same subproblem.

A problem that is applicable to be solved by dynamic programming must have two characteristics: optimal substructure and overlapping subproblems.

Firstly, the structure of an optimal solution to the problem needs to be examined. If an optimal solution to the problem contains within it optimal solutions to subprob-

lems, the problem itself exhibits optimal substructure. In dynamic programming, an optimal solution to the problem is built from the optimal solutions to subproblems. That is, we first find the optimal solutions to subproblems and, having solved the subproblems, we find an optimal solution to the problem. When finding an optimal solution to the problem, the algorithm needs to make a choice among subproblems as to which we will use to solve the problem. The cost of the problem is usually the costs of the subproblems plus a cost of the choice itself.

Secondly, the structures of the subproblems need to be the same. In another word, the dynamic programming algorithm recursively solves the same type of subproblems over and over for the problem, rather than always generating new subproblems. In this situation, we say that the optimization problem has overlapping subproblems. Since the dynamic programming solves the same subproblem by the same way, and solve each subproblem only once, the computation takes polynomial time.

In general, the dynamic programming algorithm contains four steps. Step 1 is to characterize the structure of an optimal solution. In Step 2, the value of an optimal solution is defined recursively. Step 3 computes the value of an optimal solution in a bottom-up fashion so that the final optimal solution to the problem can be determined by recursively computed values of optimal solutions to subproblems. And finally, Step 4 constructs an optimal solution from computed information obtained in Step 3. Usually, we store which choice we made in each subproblem in a table so that we can reconstruct the solution by the table. A mathematical explanation of dynamic programming in practice is given with an example of the scanline-based correspondence problem in our research in Section 3.1.2.

Dynamic programming is suitable to solve problems such as the sequence alignment problems. One of the examples is the automatic speech recognition problem. The objective is to seek, among a number of audio sequences, an audio sequence that is the most similar to a standard audio sequence predefined in a audio library. More generally, the dynamic programming algorithm can be used to compute the similarity between two audio sequences, and finds the best match. Other problems that dynamic programming is applicable to solve include the determination of optimal play in chess endgames, the optimal order for performing chain matrix multiplication, the longest common subsequence problem, the matching of gene sequences, etc.

3.1.2 Dynamic Time Warp

Dynamic time warp (DTW) algorithm [6][23] is a technique included in the general dynamic programming algorithm. It is used to determine the correspondences between two sequences, particularly the corresponding elements of two sequences. The dynamic time warp algorithm has been widely used for temporal alignment of audio sequences in speech recognition problems. Similarly, the algorithm can also be applied to the alignment of two scanlines in a stereo correspondence problem.

Recall that the scanlines are corresponded when the pair of stereo images are rectified aforementioned. The space to search for a corresponding pixel to a given pixel is limited to only two corresponding scanlines. The dynamic time warp algorithm works appropriately for this pixel correspondences problem.

The dynamic time warp algorithm follows the general procedure of the dynamic programming approach.

Step 1. Characterizing the structure of the problem.

The objective of our problem is to find the pixel correspondences. The criteria for the correspondences is the similarity between one pixel in the left scanline and another pixel in the right scanline. Consider that there are N pixels in the left scanline and M pixels in the right scanline. Here, the similarity value between two pixels is defined as

$$s(n,m) = \sum_{k=-\frac{L-1}{2}}^{\frac{L-1}{2}} |I_l(n+k) - I_r(m+k)|, \tag{3.1}$$

where n represents the nth pixel in the left scanline, m represents the mth pixel in the right scanline, with $1 \leqslant n \leqslant N$ and $1 \leqslant m \leqslant M$. L is the length of a 1D window that contains the given pixel and its neighboring pixels. Commonly the length of L is set to odd number so that the given pixel lies at the center of the window. By Equation (3.1), the similarity between two given pixels is the sum of absolute differences (SAD) between two windows of pixels surrounding the two given pixels.

Step 2. Defining and computing the value of an optimal solution recursively.

A cost value for each pair of pixels is defined and computed recursively by using the similarity values computed previously. Denote the left scanline up to the nth

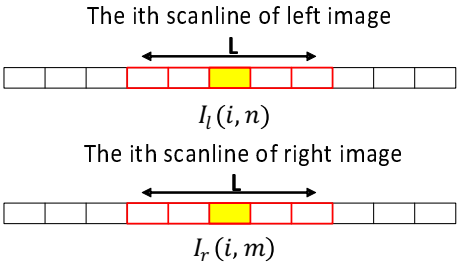


Figure 3.1: Computation of similarity value. With a window length of L, the similarity value between pixels $I_l(i,n)$ and $I_r(i,m)$ is computed using the two windows of pixels surrounding the two given pixels.

pixel and the right scanline up to the mth pixel as

$$X_{1...n} = \{I_l(1), I_l(2), ..., I_l(n)\} \tag{3.2}$$

$$Y_{1...m} = \{I_r(1), I_r(2), ..., I_r(m)\}. \tag{3.3}$$

Then the cost value can be computed recursively as

$$C(n,m) = C(X_{1...n}, Y_{1...m}) \tag{3.4}$$

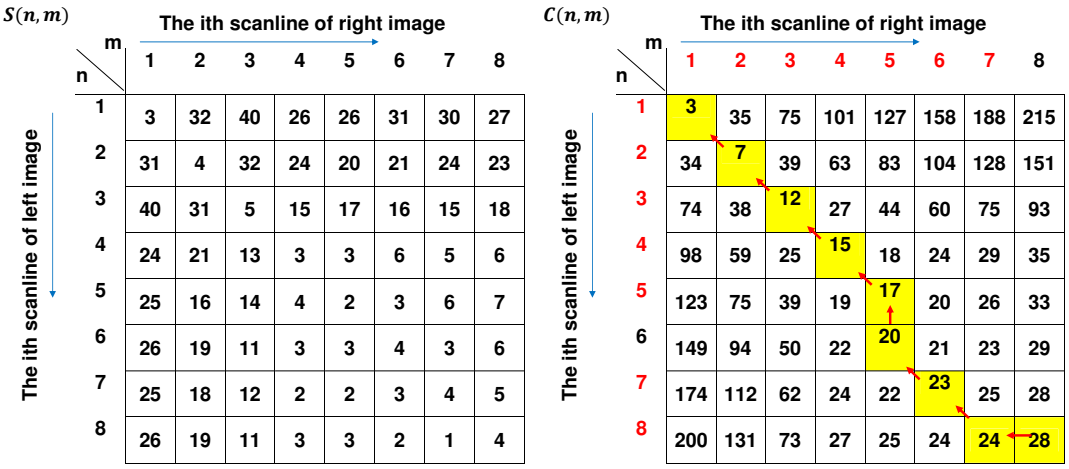
$$= s(n,m) + MIN[C(X_{1...n-1}, Y_{1...m-1}), C(X_{1...n-1}, Y_{1...m}), C(X_{1...n}, Y_{1...m-1})]. \tag{3.5}$$

The definition and computation of the cost value $C(n,m)$ can be explained as the following. For each computation in the recursion, the value $C(n,m)$ is resulted from the optimum of the three optimal solutions to its subproblems $C(n-1,m-1)$, $C(n-1,m)$ and $C(n,m-1)$, plus its own cost $s(n,m)$. Thus, from the start of the scanlines to the end, the overall cost values are computed in a recursive manner, and stored in a table (or a matrix equivalently). Finally, the optimal solution to the correspondence problem is resulted from every optimal solution to each of the correspondence subproblems.

Step 3. Reconstructing the optimal solution.

Since the values $C(n,m)$ are already stored in a table in the previous step, it is simple to reconstruct the optimal solution by looking up the values in the table. Starting from the end of the recursion, which has the optimal cost value to our correspondence problem $C(N,M)$ stored in the location (N,M) in the table, the reconstruction traces backward to reach the optimal solutions to the subproblems, which are stored at location $(n-1,m-1)$, $(n-1,m)$ and $(n,m-1)$, for a current

location (n, m). An optimal solution to the subproblems has the optimal value among $C(n-1, m-1)$, $C(n-1, m)$ and $C(n, m-1)$. When the trace reaches the location $(1, 1)$, the reconstruction finishes as every correspondence with the minimum cost is found. In the end, a disparity map is generated when the correspondences between each pair of scanlines are determined.



(a) Similarity matrix (b) Cost matrix

Figure 3.2: An example of similarity matrix and cost matrix. (a) A similarity matrix is built using the similarity values. (b) A cost matrix is built from the similarity matrix. The backward trace is shown, indicating the best matches.

The algorithm is demonstrated in the pseudo code shown in **Algorithm 1**. A table $b(n, m)$ is used to record the choice we made to the optimal solutions to subproblems in every recursion. With the choices recorded, it is simple to reconstruct the optimal solution without recomputing the cost values. Once C and b are obtained, the reconstruction can be easily achieved by the table b. That is, the pixel correspondences are determined.

When using dynamic time warp algorithm to determine the stereo correspondences, an ordering constraint of the scanlines needs to be satisfied [8]. The ordering constraint describes a situation where the order of all projective points on one scanline needs to be in the same order as those projective points on the corresponding scanline. However, the ordering constraint may not be satisfied by real scenes, in particular when small solids occlude parts of larger ones. The ordering constraint is illustrated in Figure 3.3.

For the case shown in Figure 3.3(a), the ordering constraint is satisfied. On both

Algorithm 1 Recursive computation of the optimal cost values to subproblems.

$N \leftarrow length[I_l]$

$M \leftarrow length[I_r]$

$s(1,1) \leftarrow SAD[I_l(1), I_r(1)]$

$C(1,1) \leftarrow s(1,1)$

for $m \leftarrow 2$ **to** M

  do $s(1,m) \leftarrow SAD[I_l(1), I_r(m)]$

  $C(1,m) \leftarrow s(1,m) + C(1,m-1)$

  $b(1,m) \leftarrow$ " $\leftarrow$ "

for $n \leftarrow 2$ **to** N

  do for $m \leftarrow 1$ **to** M

    do $s(n,m) \leftarrow SAD[I_l(n), I_r(m)]$

    if $m == 1$

      then $C(n,m) \leftarrow s(n,m) + C(n-1,m)$

      $b(n,m) \leftarrow$ " $\uparrow$ "

    else if $MIN[C(n-1,m-1), C(n-1,m), C(n,m-1)] == C(n-1,m-1)$

      then $C(n,m) \leftarrow s(n,m) + C(n-1,m-1)$

      $b(n,m) \leftarrow$ " $\nwarrow$ "

    else if $MIN[C(n-1,m-1), C(n-1,m), C(n,m-1)] == C(n-1,m)$

      then $C(n,m) \leftarrow s(n,m) + C(n-1,m)$

      $b(n,m) \leftarrow$ " $\uparrow$ "

    else if $MIN[C(n-1,m-1), C(n-1,m), C(n,m-1)] == C(n,m-1)$

      then $C(n,m) \leftarrow s(n,m) + C(n,m-1)$

      $b(n,m) \leftarrow$ " $\leftarrow$ "

return C *and* b

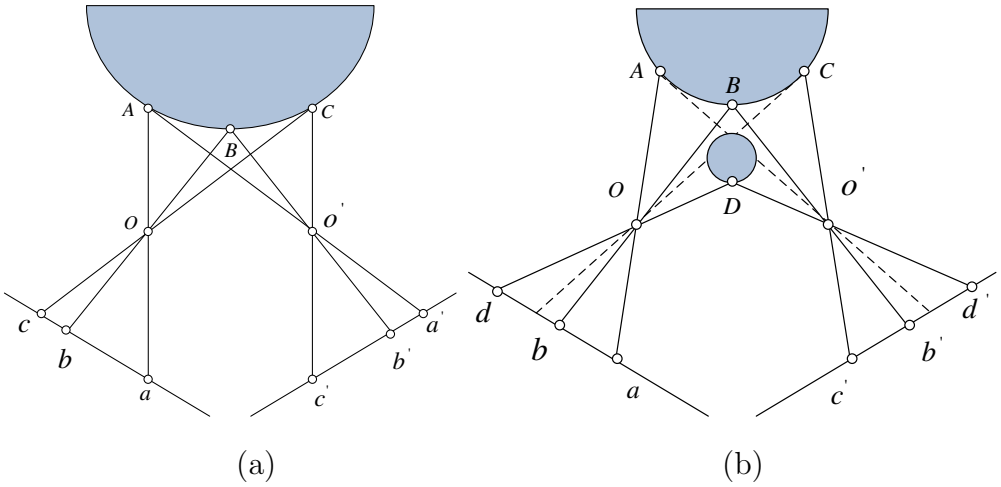


Figure 3.3: Ordering constraint. (a) Ordering constraint is satisfied. The projective points of A, B and C on both of the image plane are in the same order. (b) Ordering constraint is not satisfied. Point A and C are partly occluded and the projective points on the two image planes are in different orders.

of the image planes, the projective points of points A, B and C are in the same order from left to right. Therefore, it is appropriate to apply the dynamic time warp algorithm to seek pixel correspondences for this case. In Figure 3.3(b) where a small solid object is in front, the situation is quite different. We can see that the projective points of B and D have different orders on the image planes. Moreover, the points A and C are occluded by the small object in front. It leads to a partial occluded situation where the point can be seen in one camera but cannot be seen in the other camera. When the dynamic time warp algorithm is used for this case, it will result in errors on the search of correspondences. Since the dynamic time warp

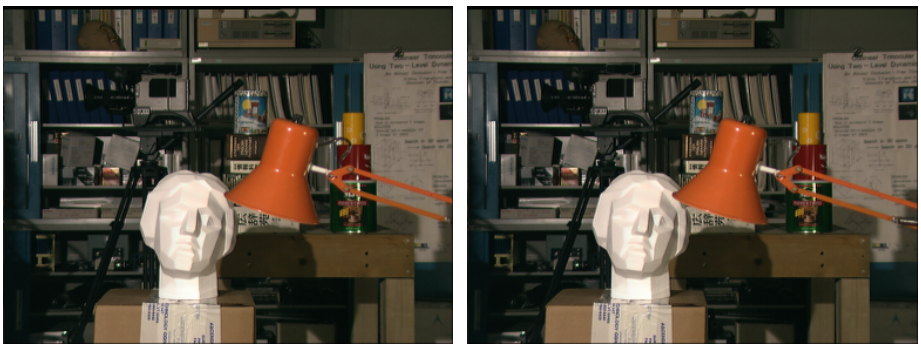


Figure 3.4: The Tsukuba stereo images.

algorithm is a scanline-based approach in searching for the pixel correspondences, it does not take into account the relation between neighboring scanlines. Therefore,

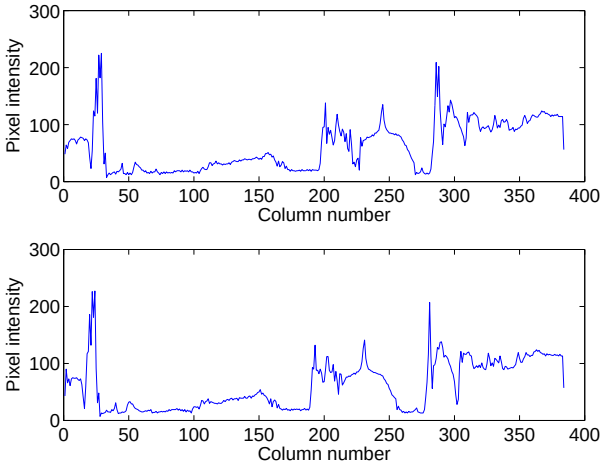
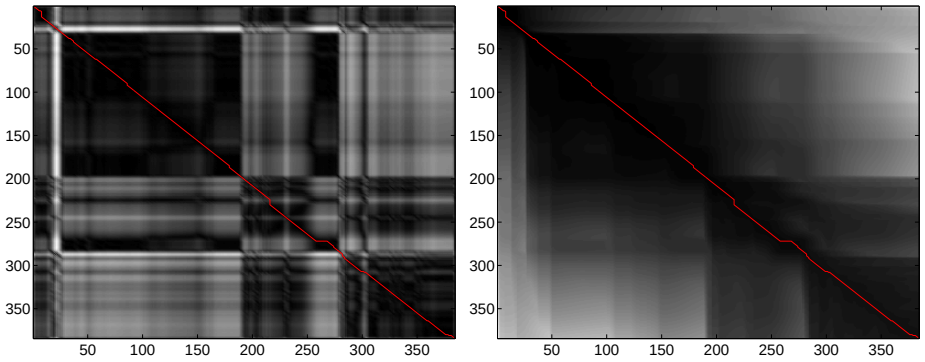


Figure 3.5: An example of scanline profiles. The upper plot shows the profile of the 107th scanline from Tsukuba left image. The lower plot shows the profile of the same scanline from Tsukuba right image.



(a) Similarity matrix (b) Cost matrix

Figure 3.6: Similarity matrix and cost matrix for the 107th scanlines. The red line shows the backward trace.

it leads to the problem of inter-scanline inconsistency and results in a horizontal streaks phenomena in the output disparity map.

Here is an example showing that the dynamic time warp algorithm is used to seek the pixel correspondences for the Tsukuba stereo pair. Figure 3.4 show the Tsukuba stereo pair. In this example, the Tsukuba stereo images in use are grayscale images with the grayscale values ranging from 0 to 255 (equivalently, the width of the grayscale value is 8 bits). The resolution of the images is 384×288.

We take the 107th lines of the stereo pair as an example to illustrate the scanline-based dynamic time warp algorithm. The scanline profiles are shown in Figure 3.5. As is seen, the scanline profile of the Line 107 contains most of the eligible grayscale values in the range of 0 and 255. And it has a couple of steep ascents and descents

Figure 3.7: Computed disparity map of Tsukuba stereo images. The disparity map is computed using the dynamic time warp algorithm with a window length of 5. Disparity values are scaled by a factor of 8.

which indicate the object boundaries. Moreover, it is evident that the left and right scanline profiles are quite similar, except for some lateral shifts. Those lateral shifts determine the disparities according to the theory mentioned previously.

The similarity value for each pixel in the scanlines is computed using the Equation (3.1). The length of the local window is set to 5 in this example. Figure 3.6(a) shows the similarity matrix with respect to the $107th$ left and right scanlines. Since there are 384 pixels in each scanline, the size of the similarity matrix is 384×384. The cost values can then be computed using the similarity values. The cost matrix for the $107th$ scanlines is shown in Figure 3.6(b). Its size is the same as the similarity matrix. The backward trace starts at the lower right corner of the cost matrix once the matrix is constructed. For cost value stored at location (n, m), the backward trace follows the rule that it traces towards the location with the minimum value among those stored in locations $(n-1, m-1)$, $(n-1, m)$ and $(n, m-1)$. The trace is highlighted by the red line in Figure 3.6.

Figure 3.7 shows the computed dense disparity map. The size of the disparity map is the same as that of the input stereo image. The disparities are directly conveyed by the value of each pixel in the disparity map. Brighter areas indicate larger disparity levels, while darker areas indicate smaller disparity levels. For the purpose of better displaying, the disparity values are scaled to a certain range so that the objects can be distinguishably displayed in the disparity map shown in Figure 3.7.

Compared to the input stereo image, the computed disparity map indicates the distance information on different objects. For example, the lamp is in front of the

statue, and the desk is further behind. It can be seen that part-occlusion exists in the Tsukuba stereo images. Part of the books behind the lamp, for example, is seen in the left image but not in the right image. For these regions with part-occlusion, correspondence errors occur and lead to multiple matches in the backward trace step. As a result, the boundary of the lamp is distorted in the computed disparity map. A common way to deal with pixels in the occluded regions is to set their disparity values to 0 by default. Also, the horizontal streaks phenomena can be noticed in the disparity map because of the nature of the scanline-based algorithm.

3.1.3 Dynamic Time Warp with Coarse Quantization

An extension to the conventional dynamic time warp algorithm is a method called the dynamic time warp with quantization method [25][18][24]. As is described in the preceding sections, the size of the similarity matrix and cost matrix depends on the number of pixels in each scanline. The number of computations to construct a similarity matrix and a cost matrix is determined by the number of pixels in a scanline when computing the correspondences for a pair of scanlines. Since there are a large amount of iterations involved, the computational expense can be enormous when the resolution of the input stereo images is large. In order to realize fast correspondence processing, a solution to reducing the number of computations needs to be found. Quantization can be an effective way to reduce the computational burden. Specifically, the input grayscale stereo images are firstly quantized into several quantization levels rather than being sent to the dynamic time warp algorithm directly. By the use of quantization, the number of computations on the quantized stereo images can be dramatically reduced in comparison to the computations on the original stereo images.

The dynamic time warp with quantization method follows the general procedure of the DTW algorithm. In addition, it applies a quantization step before the DTW algorithm.

Step 1. Quantization of the input stereo images.

Linear quantization on pixel values is applied to both of the input stereo images. The quantization enforces the input images to become more patchy and segmented images. Since an object surface is usually represented by pixels with similar grayscale

values, which are quite different from values of other object surfaces, the input images are segmented by objects and the boundaries of each object are extracted after the quantization step. Those pixels on the boundaries of objects are regarded as feature pixels.

Step 2. Dynamic Time Warp algorithm on feature pixels.

The feature pixels extracted at Step 1 are sent to the DTW algorithm instead of all pixels in the stereo images. The procedure of the DTW algorithm at this step is the same as that described in the preceding section. Firstly, the similarity value between two pixels is computed as

$$s(n,m) = \sum_{k=-\frac{L-1}{2}}^{\frac{L-1}{2}} |I_l(n+k) - I_r(m+k)| + \alpha(|n-m|) \tag{3.6}$$

where n represents the nth pixel in the left scanline, m represents the mth pixel in the right scanline, with $1 \leqslant n \leqslant N$ and $1 \leqslant m \leqslant M$. The term $\alpha(|n-m|)$ is included here in order to increase the penalty if the nth pixel and the mth pixel are too far apart. α is the weight. Since the images are quantized and only feature pixels are used in this method, two feature pixels that are further apart would probably have a smaller similarity value than the similarity value between two closer feature pixels, if computed using the Equation (3.1). Under the ordering constraint, however, the matches are probably incorrect if the two pixels are too far apart. Therefore, the term $\alpha(|n-m|)$ is necessary here.

Once the similarity values are computed, the cost matrix can be constructed using Equation (3.4). Note that the cost matrix may not be a square matrix because the corresponding left and right scanlines could have different numbers of feature pixels after the quantization step. The backward trace starts at the lower right corner of the cost matrix and it traces every optimal value based on the rule in **Algorithm 1**.

Step 3. Reconstructing the non-featured pixels.

For those non-featured pixels, the rule to assign them disparity values is simple. The run length of each segment of pixels is stored at first. After the disparity values of all feature pixels are determined, non-featured pixels at each segment would be assigned the same disparity value as that of the feature pixel in the same segment.

Here, the DTW with coarse quantization method is applied to the Tsukuba stereo

pair as an example.

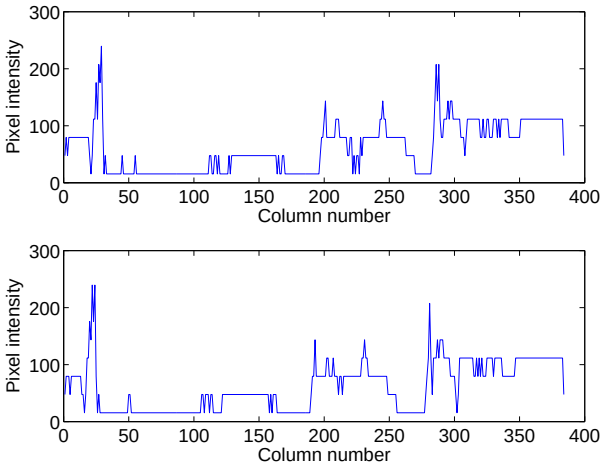
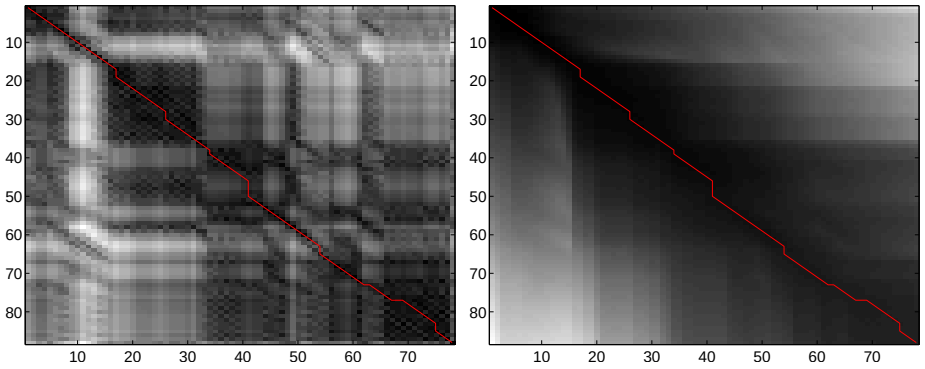


Figure 3.8: An example of quantized scanline profiles. The pixel intensity values are linearly quantized to 8 levels. The upper plot shows the 107*th* scanline from Tsukuba left image. The lower plot shows the same scanline from Tsukuba right image.



(a) Similarity matrix (b) Cost matrix

Figure 3.9: Similarity matrix and cost matrix for the quantized 107*th* scanlines. The red line shows the backward trace..

The 107*th* scanlines of the stereo pair are again used to illustrate the dynamic time warp with coarse quantization method. In this example, an 8-level linear quantization is applied to each scanline. The quantized scanline profiles of the left and right images are shown in Figure 3.8. It is evident that the scanlines are segmented by their pixel values and those feature pixels are extracted.

The similarity values for each feature pixel are computed using Equation (3.6). The length of the local window is set to 5 here. Figure 3.9(a) shows the similarity matrix with respect to the quantized 107*th* left and right scanlines. The size of this

Figure 3.10: Computed disparity map of Tsukuba stereo images. The disparity map is computed using the dynamic time warp with 3-bit quantization algorithm. The window length used is 5. Disparity values are scaled by a factor of 8.

matrix is 78×88, which indicates that there are 88 feature pixels in the left scanline and 78 feature pixels in the right scanline. Notice that, since the numbers of feature pixels in the left and right scanlines are different, it is inevitable that there would be multiple matches occurring in the backward trace. The cost matrix is shown in Figure 3.9(b) and the red line in the figure indicates the backward trace.

The computed disparity map is depicted in Figure 3.10. As is seen, the disparity map is much more segmented than that computed by the DTW algorithm. An analysis on the dynamic time warp algorithm and its quantization method will be given in Chapter 4.

3.2 Local Algorithms

A local stereo matching algorithm applies a local optimization technique to determine the correspondences. In a local algorithm, the correlations between pixels from the pair of stereo images are computed as the cost values. The cost values are directly used as the measurement to determine the best matches. For this reason, the local algorithms are often regarded as the correlation algorithms. In a local algorithm, it is common to use a winner-takes-all strategy for optimization. The winner-takes-all strategy simply chooses the pixel with the lowest cost value as the best match. There are also some drawbacks with local algorithms. Since local algorithms use finite windows of pixels to determine the correspondences, the window size directly affects the performance of the algorithms. Large windows are necessary for textureless

regions in the image, while small windows work well at boundaries. Therefore, trade-off must be made on the window size.

Local algorithms are typically window based correlation algorithms. Most commonly, the correlations are computed using the Sum of Squared Differences (SSD) or the Sum of Absolute Differences. Other approaches to compute the correlations include a local transform to the pixel values, such as the Census transform [30], etc.

3.2.1 Window Based Correlation Algorithms

The window based correlation algorithm is a method that uses the correlation among finite windows of pixels as the cost function [14]. In this algorithm, the cost value between any two windows of pixels is computed using the Sum of Squared Differences (SSD) or the Sum of Absolute Differences (SAD). Recall that the SSD and SAD over

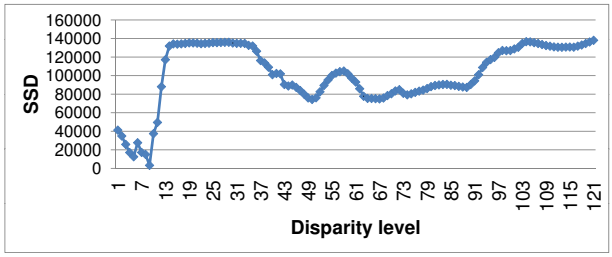


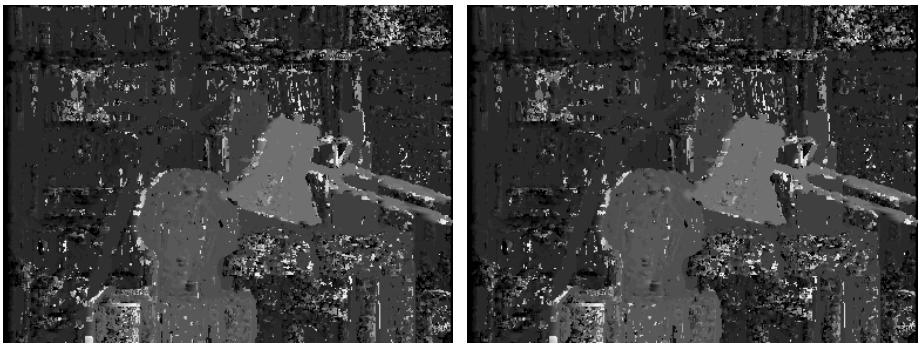
Figure 3.11: An example of SSD function. The plot shows the computed SSD values for a pixel at the $107th$ scanline and the $200th$ column, with a disparity searching range of 120. Within the range of 120, a local minimum exists at disparity level 8, which implies the best match.

two windows of pixels are computed as

$$SSD(x,y,d) = \sum_{i=-\frac{K-1}{2}}^{\frac{K-1}{2}} \sum_{j=-\frac{L-1}{2}}^{\frac{L-1}{2}} (I_l(x+i,y+j) - I_r(x+d+i,y+j))^2 \tag{3.7}$$

$$SAD(x,y,d) = \sum_{i=-\frac{K-1}{2}}^{\frac{K-1}{2}} \sum_{j=-\frac{L-1}{2}}^{\frac{L-1}{2}} |I_l(x+i,y+j) - I_r(x+d+i,y+j)| \tag{3.8}$$

where the windows are centered at pixel (x, y) in the left image and pixel (x+d,y) in the right image, respectively. With the assumption that the stereo images are rectified, there is only horizontal disparity d along the x axis. K and L are the window width and height, which are usually odd numbers so that the computed pixels lie at the centers of the windows.



(a) SSD (b) SAD

Figure 3.12: Computed disparity maps for Tsukuba images. The disparity maps are computed using window based SSD and SAD correlation algorithms. The window size is 3×3 and the searching range is 30. Disparity values are scaled by a factor of 8.

In Equation 3.7, for a given pixel (x, y) in one image of a stereo pair, its cost value computed using SSD or SAD is the function of the disparity level d. Usually, the disparity level d is set within a certain searching range. When the searching range is properly chosen, the true matched pixel with the minimum cost value would likely be found. Figure 3.11 shows an example where SSD values are computed within a searching range of 120 for a given pixel in the left image. Once the cost values are computed, a winner-takes-all strategy is used to find the pixel with the minimum cost value as the best match. The window based correlation algorithm is summarized below.

Step 1. Computing the cost values.

The Sum of Squared Differences (SSD) or the Sum of Absolute Differences (SAD) is used to computed the cost values for a given pixel.

Step 2. Finding the optimum.

A winner-takes-all strategy is used for the optimization. The pixel with the minimum cost value in the secondary image is selected as the best match. A tie-break rule is needed when there are multiple pixels with the same minimum cost value.

Step 3. Constructing the dense disparity map.

Once the best match for every pixel in the primary image is determined by **Step 1** and **Step 2**, a dense disparity map can then be constructed.

The emphasis of the window based correlation algorithm is on the matching

cost computation, which makes a direct effect on the performance of the algorithm. There are two factors: the window size and the disparity searching range. In a window based algorithm, pixels within a window are assumed to have the same characteristic. Therefore, a large window is required for a textureless region like a smooth surface in the image. On the other hand, a small window is preferred at object boundaries. For this reason, the window size needs to be chosen properly so as to solve this conflict. The disparity searching range is the other key part. As is seen in Figure 3.11, an inappropriate choice of the searching range would probably result in finding an incorrect local minimum.

Here, an example demonstrates the window based correlation algorithm applied to the Tsukuba stereo pair. In this example, the window size is set to 3×3 and the disparity searching range is 30 levels. As seen in Figure 3.12, the disparity map computed by SSD is visually very similar to the disparity map computed by SAD.

3.2.2 Census Transform Based Correlation Algorithm

The strategy of the Census transform correlation algorithm is to apply a Census transform to the images before the computation of the cost values. The Census Transform is introduced by Zabih and Woodfill [30]. The transform is based on local pixel value relations between a given pixel and its neighboring pixels within a certain window. This relation is given as

$$\xi(p_1, p_2) = \begin{cases} 0, p_1 \leqslant p_2 \\ 1, p_1 > p_2 \end{cases} \tag{3.9}$$

where p_1 is the value of the central pixel and p_2 represents the value of any other neighboring pixel within that window. If the value of a neighboring pixel is greater than or equal to the value of the central pixel, its value is replaced by 0. Otherwise, its value is replaced by 1. For the central pixel, its value is by default replaced by 0. After the transform is executed, a bit string with respect to the central pixel is produced, containing all binary values computed from the transform. An example of the Census Transform is shown below.

$$\begin{matrix} 52 & 53 & 53 \\ 50 & 51 & 53 \\ 45 & 48 & 51 \end{matrix}$$

Given a 3×3 square window of pixels. The pixel of interest is the central pixel with a value of 51. With Census transformation, the pixel values are replaced by binary values computed by Equation 3.9, which results as

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0. \end{matrix}$$

This matrix is then rearrange into a bit string. In a row-wise manner, for example, the bit string is [0 0 0 1 0 0 1 1 0].

This example shows a dense window for the Census Transform. Other than the dense window, a sparse window can also used for the transform; for example, using only every other pixel in the window.

Since the Census transform converts the actual pixel values into the relative binary values between pairs of pixels, the Census transform based correlation algorithm is insensitive to camera bias and gain differences. However, by the use of relative binary values instead of the actual pixel values, this algorithm reduces the accuracy in computing the cost values, which may lead to a higher chance of incorrect stereo correspondences. The algorithm follows the procedure as follows.

Step 1. Applying the Census Transform.

A Census transform is executed with either a dense or a sparse window. The transform is applied to a given pixel in the primary image and pixels within the disparity searching range in the secondary image.

Step 2. Computing the cost values.

Once a set of bit strings are produced by the Census transform, the cost value between two bit strings is computed using Hamming distance. The Hamming distance is the number of bits in the bit strings where bits in the same position differ. In another word, the Hamming distance can be represented by the bit-wise Exclusive-OR of the two bit strings. With less ones in the resulted bit string, the Hamming distance is smaller. The pixel windows with similar pixel values relative to the given

pixel will produce similar bit strings, which results in smaller Hamming distance.

Step 3. Finding the optimum.

A winner-takes-all strategy is simply used for the optimization. The bit string with the smallest Hamming distance is selected as the best match. If multiple matches occur, a tie-break rule must be made.

Step 4. Constructing the dense disparity map.

Repeat **Step 1** to **Step 3** to determine the best match for every pixel in the primary image. With all matches found, a dense disparity map can be constructed.

An example on the Tsukuba stereo pair shows the performance of the Census Transform based correlation algorithm. The Census transform window size is set to 11×11 in this example and the disparity searching range is 30 levels. The computed disparity map is depicted in Figure 3.13. It is evident that the disparity map is contaminated by noises.

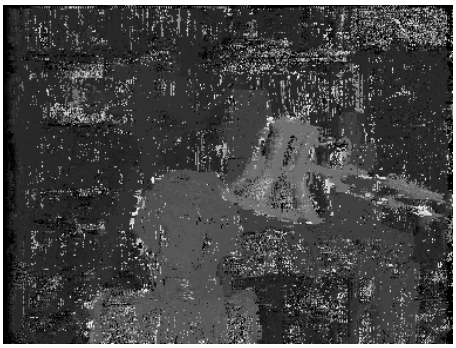


Figure 3.13: Disparity map for Tsukuba images computed using window based Census transform correlation algorithm. The window size is 11×11 and the searching range is 30. Disparity values are scaled by a factor of 8.

Chapter 4

Analysis of the Stereo Matching Algorithms

In this Chapter, the parameters for the global and local algorithms described in Chapter 3 are analyzed in detail. The objective is to draw a comparison among these algorithms by varying their parameters and pick the best choice for our implementation afterwards. The methodology used for the analysis is described and the experimental evaluations are presented in this chapter.

4.1 Analysis Methodology

To evaluate the performance of a stereo correspondence algorithm or the effects of varying some of its parameters, the strategy is to quantitatively compare the computed disparity map from a stereo image pair to its ground truth data [21].

In our experiments, there are two quality measures computed based on known ground truth data.

1. Root-Mean-Squared (RMS) error between the computed disparity map $d_C(x, y)$ and the ground truth disparity map $d_T(x, y)$,

$$R = (\frac{1}{N}\sum_{(x,y)} |d_C(x, y) - d_T(x, y)|^2)^{\frac{1}{2}}, \tag{4.1}$$

where N is the total number of pixels in one stereo image.

2. Percentage of bad matching pixels, formula

$$B = \frac{1}{N}\sum_{(x,y)} M(x, y), \tag{4.2}$$

where

$$M(x, y) = \begin{cases} = & 1, |d_C(x, y) - d_T(x, y)| > \delta_d \\ = & 0, others \end{cases}$$

N is the total number of pixels and δ_d is a disparity error tolerance.

The RMS error reflects the absolute error from the computed disparity map. A large error from one pixel would make a significant contribution to the overall error. On the other hand, the percentage of bad matching pixels provides a statistical measure on the matching error. It reflects the overall matching quality of the computed disparity map.

In addition, the elapsed time of the Matlab program to generate a disparity map for each stereo algorithm is measured. Although the actual running time depends on the platform where the algorithm is implemented in practice, the elapsed time measured here can still reflect the relative efficiency of each algorithm in terms of running time.

With reference to Daniel Scharstein and Richard Szeliski's previous work [21] on the Middlebury Stereo Vision web page [10], four sets of stereo image pairs are selected as samples to evaluate the performances of different stereo algorithms. These four image sets are the Tsukuba, Teddy, Cones and Venus, shown in Figure 4.1.

Figure 4.1: Test stereo images. Upper row are the stereo left images: Tsukuba, Teddy, Cones and Venus. Lower row are their disparity ground truth images. Disparity values for Tsukuba and Venus images are scaled by a factor of 8; disparity values for Teddy and Cones image are scaled by a factor of 4.

These stereo image pairs are available on the Middlebury Stereo Vision web page. They are captured by a high-resolution digital camera and each set of stereo images are captured at pure horizontal translation. Therefore, each pair of stereo images is noise-free and rectified images. All sets of stereo images are made up of piecewise planar objects, which conforms to our pre-condition on planar stereo correspondences.

4.2 Analysis of the Global Algorithms

In this section, an experiment is to analyze two global stereo algorithms: dynamic time warp (DTW) algorithm and dynamic time warp with quantization algorithm.

4.2.1 Objective

The objective of this experiment is to analyze two parameters: the window length and the number of quantization bits, in both DTW and DTW with quantization algorithms. As described in Chapter 3, a similarity value for a given pixel is computed using a local 1D window of pixels centered at it. Here, the window length parameter is the length of the local 1D window. Regarding the quantization bits, different numbers of quantization bits are used in the DTW with quantization algorithm, compared to the conventional dynamic time warp algorithm. During this experiment, the performance of the global algorithms is examined by varying these two parameters.

In this experiment, the 1D window length is set to 3, 5, 7, 9 and 11. Since the original input images are represented by 8 bits per pixel, the dynamic time warp algorithm uses 8 bits and the numbers of bits used for the DTW with quantization algorithm are 3, 4, 5, 6 and 7.

4.2.2 Results

Figure 4.2 plots the measurements of this experiment.

From the plots we can see that the window length does not affect the performance of the global algorithms very much. Although there are small improvements in terms of the RMS errors and bad matching percentages from length 3 to length 5, the window lengths greater than 5 do not result in a much better performance in terms of either RMS errors or bad matching percentage. The elapsed time slightly increases as the window length increases. Overall, a window length of 5 or 7 yields better results.

For the quantization bits, it is evident that the conventional DTW algorithm with 8 bits outperforms the DTW with different quantization bits in term of both

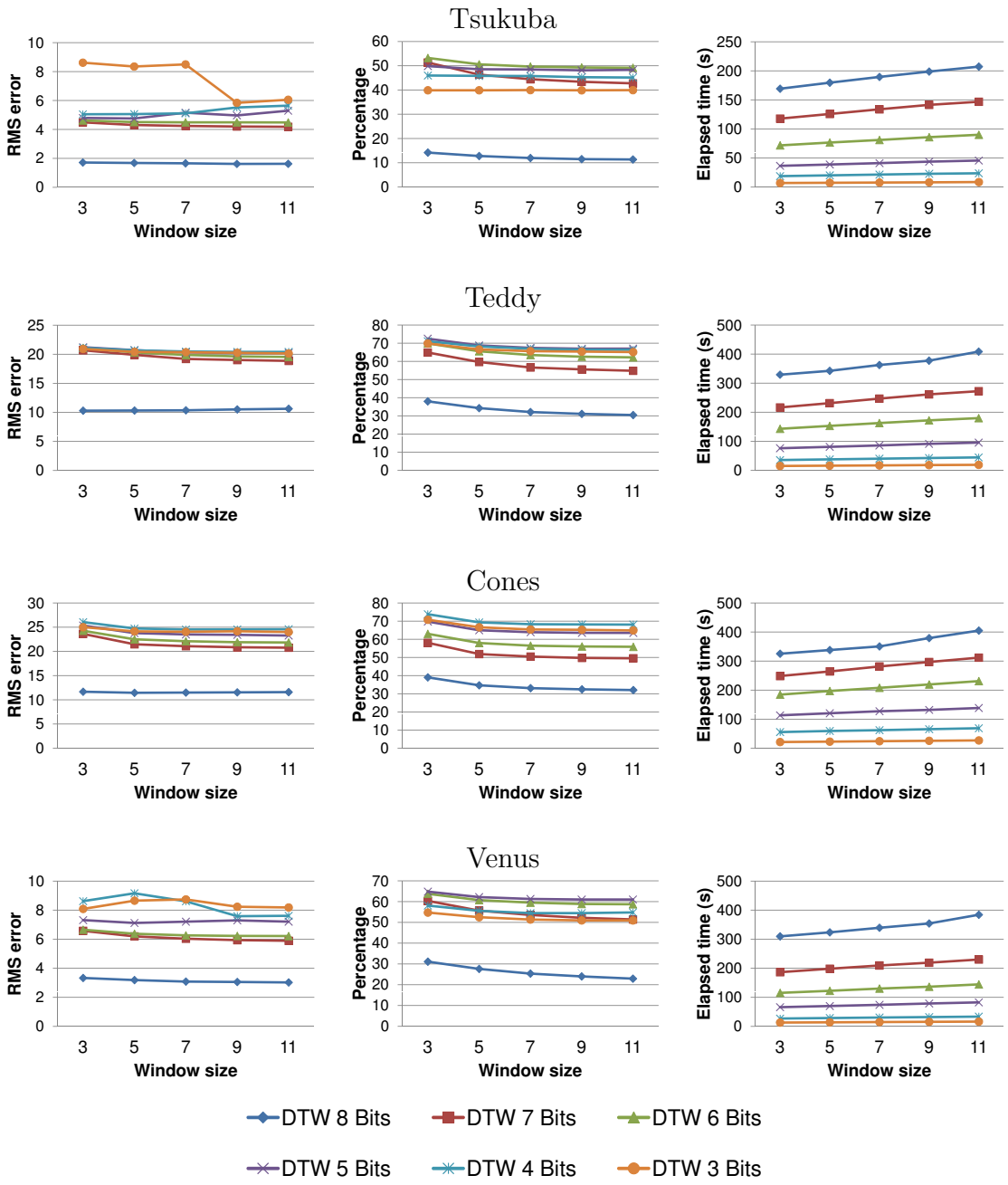


Figure 4.2: Measured values of the global algorithms. The measurements are RMS error, percentage of bad matching pixels and elapsed time.

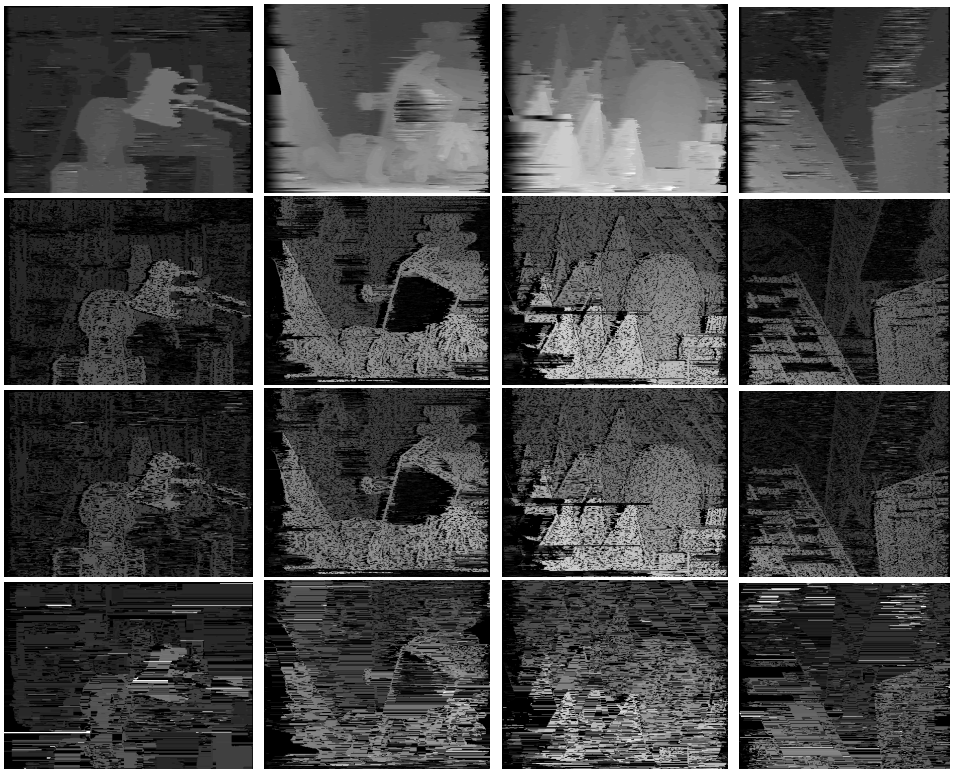


Figure 4.3: Computed disparity maps for Tsukuba, Teddy, Cones and Venus images. The disparity maps are computed using dynamic time warp (DTW) algorithm and dynamic time warp with quantization algorithm. First row: DTW. Second row: DTW with 7-bit quantization. Third row: DTW with 6-bit quantization. Fourth row: DTW with 3-bit quantization. Disparity values for Tsukuba and Venus images are scaled by a factor of 8; disparity values for Teddy and Cones image are scaled by a factor of 4.

RMS errors and bad matching percentage. For example, in all four sets of images, the RMS errors and bad matching percentage with 8 bits are approximately half of those values with 7 bits. However, the advantage of the DTW with quantization algorithm is evident. The elapsed time with quantization is much lower than the time spent with the conventional DTW algorithm. With 7 bits, the elapsed time is approximately one third less than the time with 8 bits. In this case where only one bit is reduced, the reduction of time is enormous. Therefore, a trade off between the time and accuracy is needed in order to retain an acceptable image quality in an application where time is critical.

Figure 4.3 shows some computed disparity maps for these four sets of stereo images. For the disparity maps shown in Figure 4.3, the window length is set to 5 and the results are generated by the DTW algorithm with original 8 bits, as well as the

quantization algorithm with quantization bits 7, 6 and 3. For the DTW algorithm, the inter-scanline streaks in the computed disparity maps can be easily seen. This is because the DTW is a scanline based algorithm and it is unable to retain the inter-scanline continuities. In the region of object boundaries, there is little distortion to the object boundaries. In the regions with little textures, the DTW algorithm works well for the Tsukuba and the Cones images. But for the Teddy and the Venus images which have large textureless regions, the DTW algorithm produces errors in those textureless regions.

For the DTW with quantization algorithm, it is evident that more noise is included in the disparity maps. In addition to the inter-scanline streaks, there are lots of black spots. In most of the cases, the black spots are due to the incorrect matches during searching for the correspondences. When an incorrect match is encountered for a pixel, that pixel is assigned a disparity value of zero, which is pure black. In the quantization algorithm, these regions of black spots are expanded because the pixel values are quantized into fewer levels so that the images become more patchy. One advantage of the quantization process is that it has a denoising effect to the input images so that it enables viewing global objects without being affected by the details inside the global objects. This is shown in the Teddy image as an example. With this patchy effect, the DTW with quantization algorithm still works well in the discontinuous regions such as object boundaries. However, it fails in some of the textureless regions. For example, in the Teddy image, errors occur at the surface of the big box where the algorithm is unable to yield correct disparity values. Also, it is more noisy in the textureless regions due to the black spots. Overall, the quantization algorithm with 7 bits or 6 bits can still generate a disparity map of acceptable quality. With 7 or 6 bits, the objects are distinguishable and the computed disparity maps are visually close to the ground truth image. The black spots noise can be effectively removed by some common denoising technique so that the quality of the results can be improved. On the other hand, the performance of the DTW quantization algorithm is poor when it uses too less quantization bits. When 3 bits are used, for example, the noise is enormous and the objects are hardly distinguished in the disparity map.

4.2.3 Conclusion

The DTW algorithm provides excellent results of disparity maps in terms of RMS error, bad matching percentage and visual quality. In Figure 4.2, it is seen that different window lengths do not noticeably affect the quality of the computed disparity maps. The drawback of the DTW algorithm is that it is time-consuming due to the iterations as well as the forward and backward computations. The DTW with quantization algorithm consumes much less time to produce a disparity map, compared to the DTW algorithm. When 7 or 6 bits are used for the quantization, the algorithm produces acceptable results of disparity maps in terms of visual quality. However, the quality of disparity map can be poor with too less quantization bits used.

4.3 Analysis of the Local Algorithms

In this section, the experiment examines the local stereo algorithms including the window based SSD correlation, SAD correlation and Census transform algorithms.

4.3.1 Objective

The objective of this experiment is to analyze two parameters, the window size and the disparity searching range, in all three window based local stereo algorithms. Recall the local algorithms introduced in Chapter 3, the window size parameter is the size of the local window of pixels used to compute the correlations. The disparity searching range defines the maximum searching distance with which the correspondences are searched. During this experiment, the performance of the three window based algorithms is examined by varying these two parameters.

In this experiment, a square window is used and the window size is set to the odd values from 3×3 and 37×37. The searching range is set to 30 and 60 respectively.

4.3.2 Results

Figure 4.4 plots the measurements of this experiment.

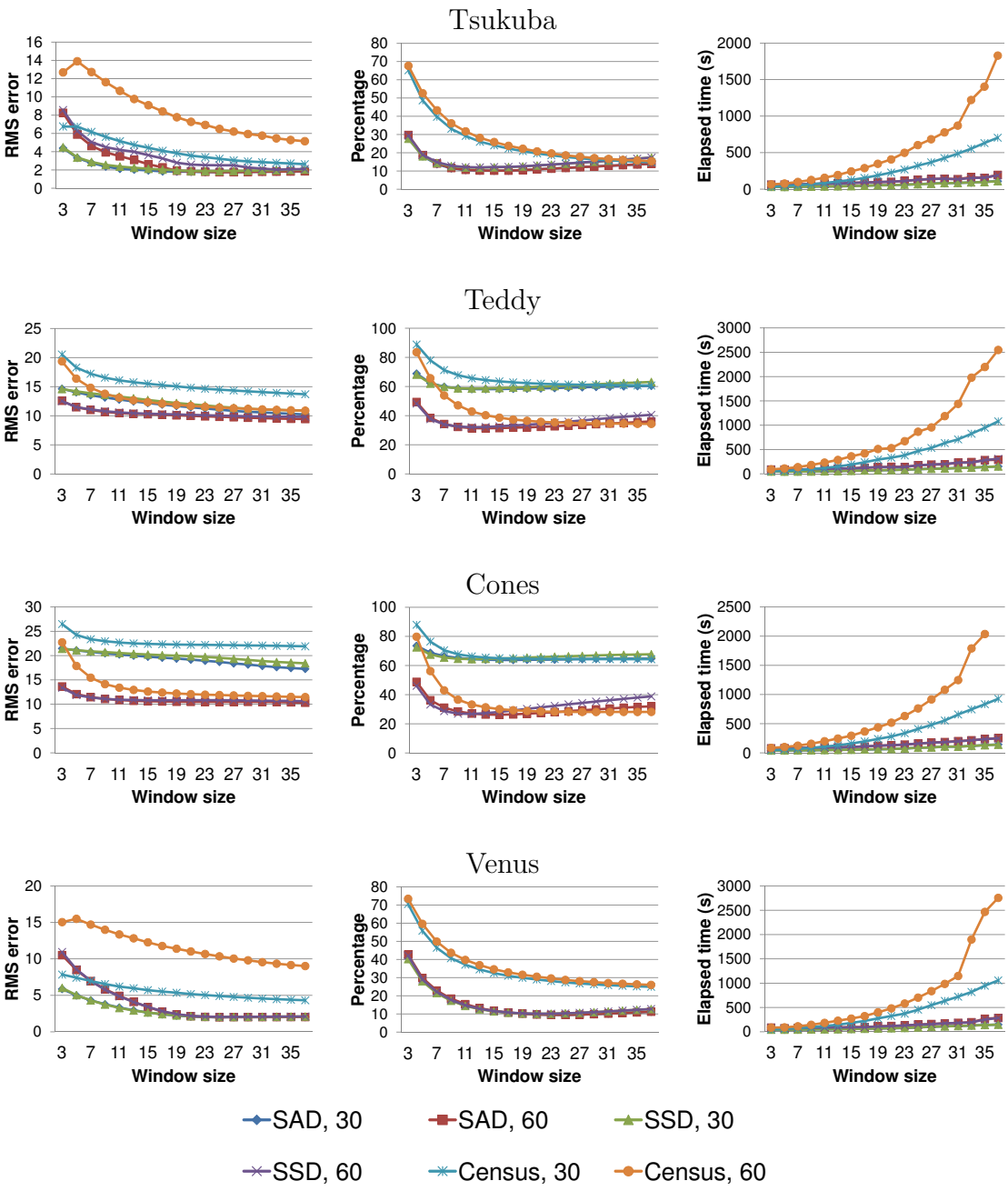


Figure 4.4: Measured values of the local algorithms. The measurements are RMS error, percentage of bad matching pixels and elapsed time.

From the plots in Figure 4.4, it can be clearly seen that the performance of a window based correlation algorithm is noticeably affected by the window size parameter. As discussed in Chapter 3, a small window size works well in the discontinuous regions such as object boundaries in the image, while a large window size works well in the textureless regions. For an image with lots of small details, e.g. the Teddy image, a excessively large window size is unable to retain the discontinuities in the object boundary regions so that the object boundaries are distorted. On the other hand, for an image with large textureless regions, e.g. the Venus image, a small window size fails to search for correct matches in the textureless regions. This is because the small window is not large enough to include the pixel information of the entire textureless region. Therefore, it is reasonable to expect that a moderate window size would probably produce better results overall. The plots in Figure 4.4 prove this expectation as the best results are produced with window sizes between 7×7 and 15×15, especially by the SAD and SSD correlation algorithms in terms of RMS errors and the bad matching percentage. In terms of elapsed time, the algorithms cost more time as the window size increases.

With respect to the different window based correlation algorithms, the SSD and SAD algorithms perform similarly. The Census transform algorithm produces more errors as expected and it needs a much larger window size to reduce the errors to the same extent as the SSD and SAD algorithms have. The Census transform algorithm is also more time-consuming than the SSD and SAD algorithms when the window size is large.

The searching range parameter defines the maximum distance which the search reaches. The window based correlation algorithms usually use this parameter so that the search does not necessarily reach the end of each scanline. As illustrated in Figure 4.4, the best searching range selected differs from image patterns. A searching range of 30 works better for the Tsukuba image, while a searching range of 60 outperforms the searching range of 30 for the Teddy and the Cones images. And the two search range values produce similar results for the Venus image. As expected, a searching range of 30 is more efficient in time.

Figure 4.5 shows some computed disparity maps for these four sets of images. As illustrated in the plots in Figure 4.5, both the SAD and the SSD algorithms produce

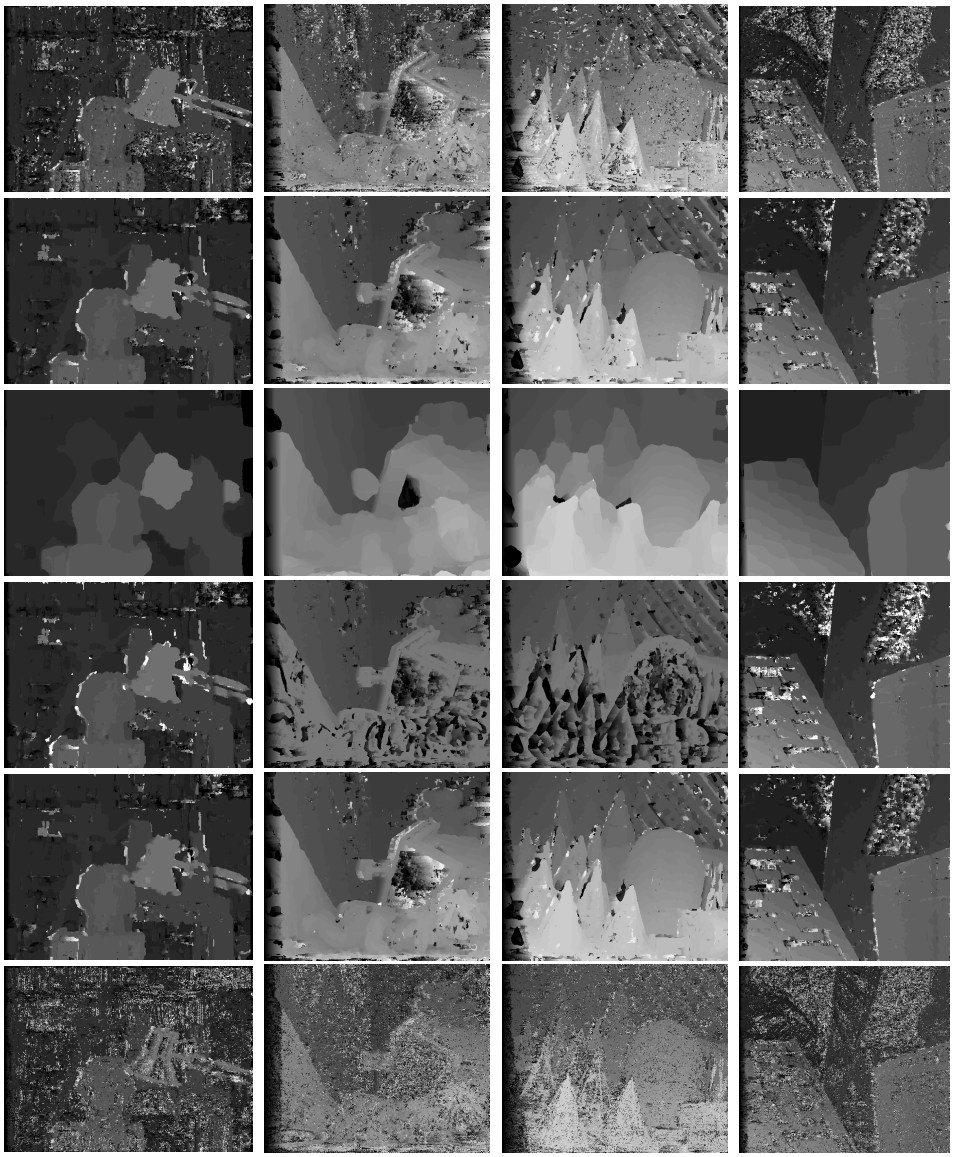


Figure 4.5: Computed disparity maps for Tsukuba, Teddy, Cones and Venus images. The disparity maps are computed using window based correlation algorithms. Tsukuba and Venus images from top to bottom: SAD with size 3×3 and searching range 30, SAD with size 7×7 and searching range 30, SAD with size 35×35 and searching range 30, SAD with size 7×7 and searching range 60, SSD with size 7×7 and searching range 30, Census with size 7×7 and searching range 30. Teddy and Cones images from top to bottom: SAD with size 3×3 and searching range 60, SAD with size 7×7 and searching range 60, SAD with size 35×35 and searching range 60, SAD with size 7×7 and searching range 30, SSD with size 7×7 and searching range 60, Census with size 7×7 and searching range 60. Disparity values for Tsukuba and Venus images are scaled by a factor of 8; disparity values for Teddy and Cones image are scaled by a factor of 4.

excellent results but the SAD algorithm is more efficient than the SSD algorithm in terms of computational complexity. Hence, some typical disparity maps computed by the SAD algorithm are selected to illustrate the performances. Also, one disparity map by the SSD algorithm and two disparity maps by Census transform algorithm are used for comparison.

For a 3×3 window, the disparity maps contain lots of noise, especially in the textureless regions. The noise is noticeably reduced as the window size is increased up to 7×7. When the window size is excessively large, the object boundaries are enormously distorted, which can be seen in the disparity maps with a 35×35 window size. For an image having many large textureless regions, e.g. the Venus image, a large window size is able to produce better results. As seen in the Venus image of Figure 4.5, the disparity map with a 7×7 window still contains lots of noise in the textureless regions. A window size larger than 7×7 is expected to performs better. In the disparity map with a 35×35 window, the noise is removed and the disparity map is close to the ground truth image, except that the object boundaries are deformed. In comparison, a window size of 7×7 performs well for the Tsukuba image and the Cones image, since there is no large textureless regions in both images.

For the disparity searching range parameter, the optimal choice of its values depends on the image patterns. For example, the SAD algorithm performs better with searching range of 30 for the Tsukuba image, while it produces a better result with searching range of 60 for the Cones image.

With respect to the three window based algorithms, the SAD and SSD algorithms performs similarly and both of them result in low-noise disparity maps with a small window size. On the other hand, the Census transform algorithm requires a larger window size to produce the disparity maps with the same visual quality. Overall, these observations conform to the numerical evaluations plotted in Figure 4.4.

4.3.3 Conclusion

The SAD algorithm provides the best performance among the three window based algorithms, in terms of both numerical evaluations and visual quality. With respect to the two parameters, a window size between 7×7 and 15×15 yields excellent results without consuming too much time in general. The disparity searching range

depends on the image patterns. Usually, it is reasonable to set the searching range as one tenth of the image width in practice.

4.4 Overall Comparison

In this section, an experiment compares the DTW algorithms with the window based SAD correlation algorithm.

4.4.1 Objective

The objective of this experiment is to draw a comparison between the global algorithms and the local algorithms. From the observations in the previous sections, three algorithms are chosen for the comparison in this experiment. The algorithms are the DTW algorithm with a window length of 5, the DTW algorithm with 6-bit quantization and a window length of 5, and the window based SAD correlation algorithm with a window size of 7×7 and searching range of 60. This experiment aims to find out the optimal algorithm among them.

4.4.2 Results

Figure 4.6 plots the measures of this experiment. As illustrated in Figure 4.6, the

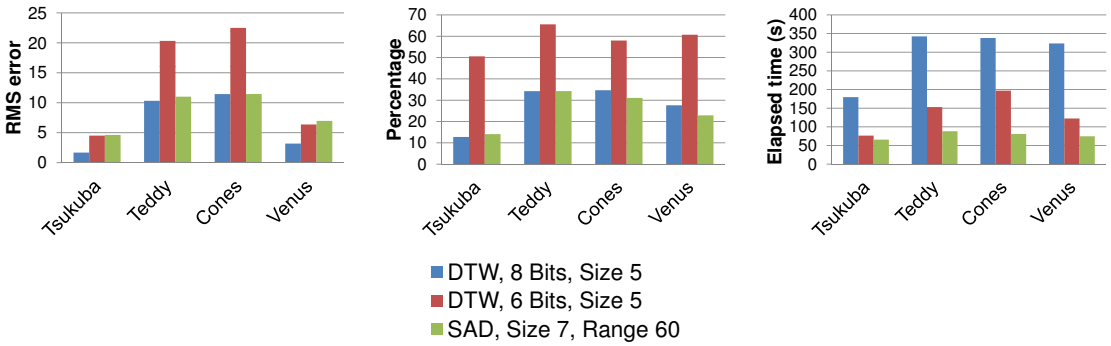


Figure 4.6: Analysis measurements of the overall comparison. The measurements are RMS error, percentage of bad matching pixels and elapsed time. The algorithms are DTW algorithm with a window length 5, DTW with 6-bit quantization algorithm with a window length 5, and window based SAD correlation algorithm with window size 7×7 and searching range 60.

DTW algorithm has lower RMS errors than the SAD algorithm for the Tsukuba and

the Venus images. But for the Teddy and the Cones images, the RMS erros of the DTW algorithm is very close to the RMS errors of the SAD algorithm. In terms of the bad matching percentage, these two algorithms have nearly the same performance for the four sets of images. The DTW with 6-bit quantization algorithm, however, performs the worst in both RMS errors and the bad matching percentage. When the elapsed time is considered, the SAD algorithm clearly outperforms the DTW algorithm, as the time consumed by the SAD algorithm is approximately one third of the time by the DTW algorithm. The DTW with 6-bit quantization algorithm also cost more time than the SAD algorithm but much less than the DTW algorithm. Figure 4.7 shows the computed disparity maps produced by these algorithms. As seen

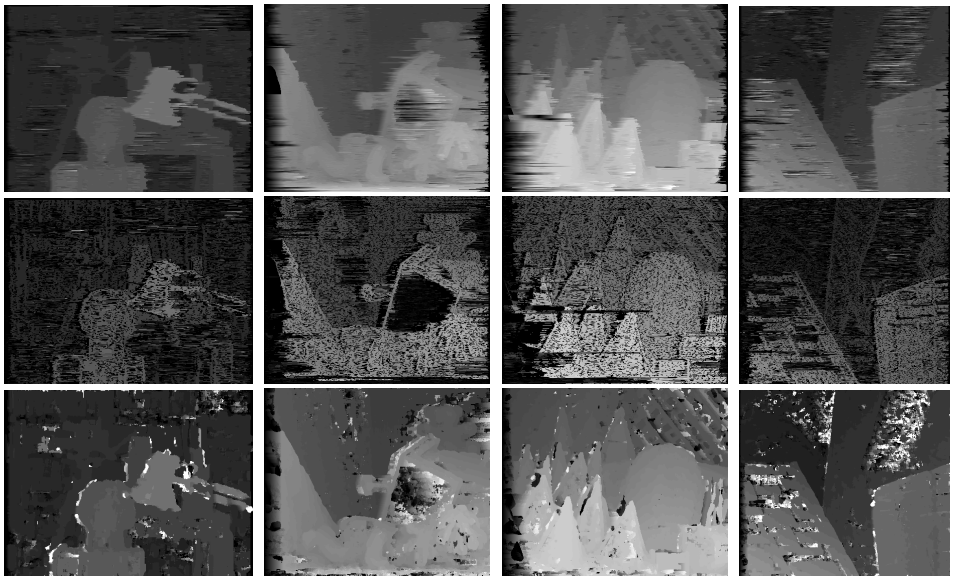


Figure 4.7: Computed disparity maps for Tsukuba, Teddy, Cones and Venus images. The algorithms used from top row to bottom row are DTW algorithm with a window length 5, DTW with 6-bit quantization algorithm with a window length 5, and window based SAD correlation algorithm with window size 7×7 and searching range 60. Disparity values for Tsukuba and Venus images are scaled by a factor of 8; disparity values for Teddy and Cones image are scaled by a factor of 4.

in Figure 4.7, the disparity maps by the SAD algorithm is slightly noisier than the disparity maps by the DTW algorithm. But the DTW with quantization algorithm introduces the most noise. The SAD algorithm performs well in the textureless regions for the Tsukuba and the Cones images, but it results in noise for the Teddy and the Venus images. On the other side, the DTW algorithm works well in the textureless regions but the streaks phenomenon is noticeable.

4.4.3 Conclusion

Both the DTW algorithm and the SAD algorithm provide better results than the DTW with quantization algorithm in terms of quality. However, the SAD algorithm is much more efficient when time is considered. Overall, the SAD algorithm is the optimal algorithm among them in practice as time processing time is critical.

Chapter 5

FPGA Implementation

5.1 Overall Strategy of the Stereo Algorithm

A digital logic architecture of the window based SAD correlation algorithm is designed so that the algorithm can be built using hardware device. The design is described using Verilog hardware description language (Verilog HDL). As mentioned in previous chapters, there are some parameters to specify in the window based SAD correlation algorithm, such as the image width, image height, window size and the disparity searching range. In our current design, the image width, window size and the disparity searching range are fixed once the architecture of the design is built.

The data flow of the design is shown in Figure 5.1. The architecture fully utilizes the parallelism and pipelining techniques in hardware design. As seen in Figure 5.1, the design is a straight-forward streaming architecture, which achieves to process one pixel data per clock cycle. Therefore, the processing speed only depends on the clock frequency, aside from the image size.

Figure 5.1 shows the data flow of the design with a 3×3 window size. The Camera 0 and Camera 1 need to be synchronized. That is, both of the cameras deliver the same pixel coordinates at the same time and their pixel clocks are synchronized. In this example, Camera 0 acts as the primary camera and Camera 1 is the secondary camera.

The pixel data from two cameras (image sensors) are firstly read into two sets of line buffers respectively. The length of each line buffer is equal to the image width and the number of line buffers for a camera is the height of a processing window. In this example, there are 3 line buffers for each camera, since the height of the processing window is 3. It is inevitable that the line buffer architecture introduces some initial delays which are proportional to the total length of all line buffers for

one camera. However, this architecture makes it possible to process the streaming data in multiple adjacent scanlines simultaneously so that a 2D window of processing pixel data is always available for a window based algorithm. The line buffers hold the 2D arrays of pixel data for subsequent processing.

The SAD module computes the Sum of Absolute Differences (SAD) values between a given window of pixels from Camera 0 and a set of windows of pixels with different disparity levels from Camera 1. The number of the SAD module instantiations is equal to the number of disparity levels. In another word, it is the disparity searching range plus one. Therefore, SAD values at all disparity levels are computed simultaneously in parallel.

A minimum selector searches for the minimum of the computed SAD values. The window with the minimum from Camera 1 corresponds to the given window from Camera 0. With this correspondence, the disparity for the given pixel from Camera 0 is determined.

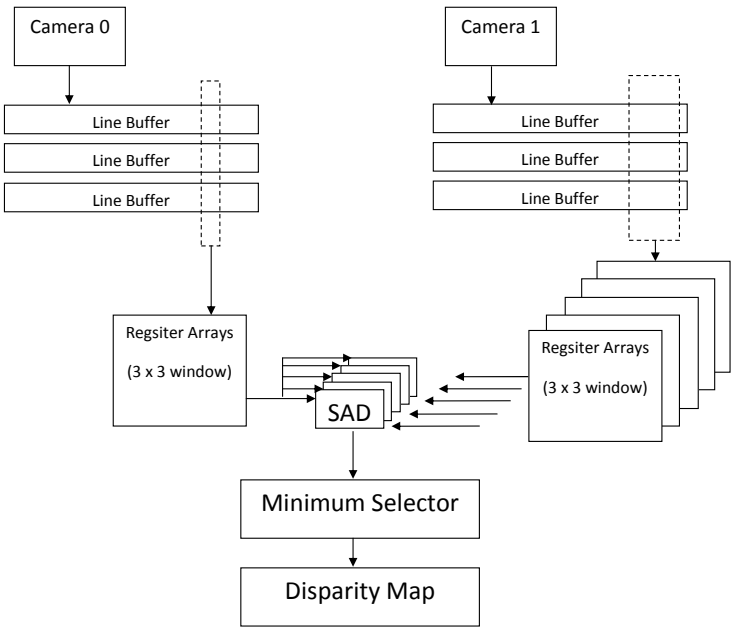


Figure 5.1: Data flow of the window based SAD correlation algorithm implementation.

5.2 Introduction to Hardware Devices

5.2.1 Altera DE2 Development and Education Board

The Altera DE2 Development and Education Board [1] features an Altera Cyclone® II 2C35 FPGA device, along with onboard clock inputs, memory chips, various ports, input buttons and switches, LEDs, etc. The hardware features on the Altera DE2 board used in our project are the following.

Altera Cyclone II 2C35 FPGA device.

As one of the FPGA devices in Altera's low-cost Cyclone® II FPGA family, the Cyclone® II 2C35 FPGA is build by high-density architecture with 33,216 logic elements (LEs) and 105 M4K RAM blocks with 483,840 total RAM bits [1]. It features 35 18- × 18-bit embedded multipliers to support high-performance arithmetics, 4 phase-lock loops (PLLs) for system clock management, and high-speed external memory interface support for SRAM and DRAM devices.

Clocks.

On-board 27MHz and 50MHz oscillators.

SDRAM.

On-board 8Mbyte Single Data Rate Synchronous Dynamic RAM memory chip, organized as $1M \times 16bits \times 4banks$.

40-pin Expansion Header.

Two 40-pin expansion headers consist of 72 Cyclone II I/O pins, as well as 8 power and ground lines. They are designed to accept a standard 40-pin ribbon cable.

VGA Output.

The VGA output port uses the ADV7123 240MHz triple 10-bit high-speed video digital-to-analog converter (DAC), with a 15-pin high-density D-sub connector. It supports up to 1600 × 1200 resolution at 100Hz refresh rate.

5.2.2 Terasic TRDB-DC2 CMOS Camera

The Terasic TRDB-DC2 CMOS image sensor [26] features a Micron 1/3-inch megapixel CMOS active-pixel digital image sensor with an active imaging pixel array of $1280H \times 1024V$ [26]. It incorporates sophisticated camera functions such as windowing, col-

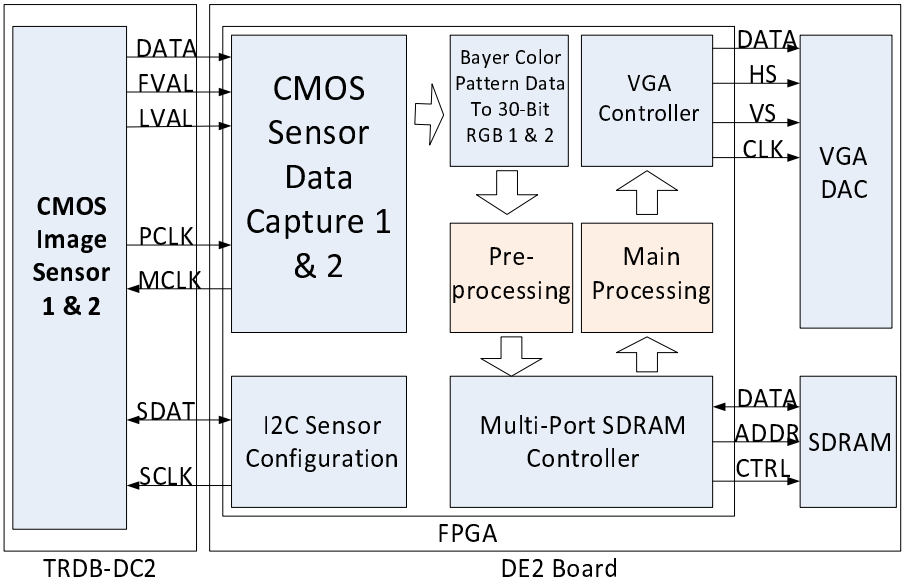


Figure 5.2: Block diagram of FPGA implementation modules.

umn and row skip mode, and snapshot mode. It is programmable through a two-wire serial interface. An on-chip analog-to-digital converter (ADC) provides 10 bits per pixel. Internal signals *FRAME_VALID* and *LINE_VALID* are connected to output ports on dedicated pins. A 25MHz pixel clock is synchronous with valid data.

5.3 Overall Implementation

In this project, the system is a synchronous design by clock signals *CCD_PIXCLK* and *VGA_CTRL_CLK*. The general configuration of the entire system is shown in Figure 5.2. The introduction to each module of the system is in the following sections.

5.3.1 Module: I^2C CMOS Sensor Configuration

This module configures the features of the TRDB-DC2 CMOS image sensor. The configuration signals are transferred according to the I^2C Protocol. Specifically, there are two serial buses, which are clock line and data line. This module is a serial master/slave interface. Registers of the image sensor are written and read through this two-wire serial interface bus. The configuration data are transferred in and out of the image sensor through the serial data line, which is synchronized to a master clock signal generated inside this module.

```
always
begin
        case(LUT_INDEX)
        0       :       LUT_DATA        <=      16'h0000;
        1       :       LUT_DATA        <=      16'h2000;
        2       :       LUT_DATA        <=      16'hF101;       //      Mirror Row and
Columns
        3       :       LUT_DATA        <=      {8'h09,iExposure[15:8]};// Exposure
        4       :       LUT_DATA        <=      {8'hF1,iExposure[7:0]};
        5       :       LUT_DATA        <=      16'h2B00;       //      Green 1 Gain
        6       :       LUT_DATA        <=      16'hF1B0;
        7       :       LUT_DATA        <=      16'h2C00;       //      Blue Gain
        8       :       LUT_DATA        <=      16'hF1CF;
        9       :       LUT_DATA        <=      16'h2D00;       //      Red Gain
        10      :       LUT_DATA        <=      16'hF1CF;
        11      :       LUT_DATA        <=      16'h2E00;       //      Green 2 Gain
        12      :       LUT_DATA        <=      16'hF1B0;
        13      :       LUT_DATA        <=      16'h0500;       //      H_Blanking
        14      :       LUT_DATA        <=      16'hF188;
        15      :       LUT_DATA        <=      16'h0600;       //      V_Blanking
        16      :       LUT_DATA        <=      16'hF119;
        default:LUT_DATA        <=      16'h0000;
        endcase
end
```

Figure 5.3: I^2C CMOS configuration Verilog code.

The two-wire serial interface defines several different transmission codes, such as a start bit, the slave device 8-bit address, an acknowledge bit (or a no acknowledge bit), an 8-bit message and a stop bit. Also, there are several different interface write and read sequences. In this project, the only one used is an 8-bit write sequence.

The code shown in Figure 5.3 is from the file *I2C_CCD_Config.v*. The registers for mirror rows and columns (set rows and columns in opposite orders respectively, compared to the input orders), exposure, green gain, blue gain, red gain, horizontal blanking and vertical blanking are written with new values. The other registers remain the default values.

5.3.2 Module: CMOS Sensor Data Capture

The image data are read out in a progressive scan. Valid image data are surrounded by horizontal blanking and vertical blanking. The amount of horizontal blanking and vertical blanking are both programmable through registers. There are other two signals, which are *FRAME_VALID* and *LINE_VALID*, to indicate whether the frame or scanline is valid. A spatial illustration of image readout is shown in Figure

5.4. Figure 5.5 demonstrates the Bayer color pattern which is the sensor output data format. As shown in Figure 5.5, the even-numbered rows contain green and red color pixels, and odd-numbered rows contain blue and green color pixels. Even-numbered columns contain green and blue color pixels, and odd-numbered columns contain red and green color pixels. Therefore, in any block of four pixel data, there are two green pixels along with one red pixel and one blue pixel.

The CMOS image sensor is controlled by a sensor master clock of 25MHz and the pixel data are output into *CMOS sensor data capture* module synchronized to the sensor pixel clock *PIXCLK* of 25MHz. When *LINE_VALID* signal is high, one 10-bit pixel datum is output every *PIXCLK* period. The *PIXCLK* signal is normally the inverted of the master clock, allowing *PIXCLK* to be used as a clock to latch the data. *PIXCLK* is continuously enabled, even during the blanking period. An example of image sensor output data timing is shown in Figure 5.6. With the *FRAME_VALID* and *LINE_VALID* signals, the total number of valid pixels in each scanline *X_Cont* and the total number of valid scanlines in each frame *Y_Cont* are counted.

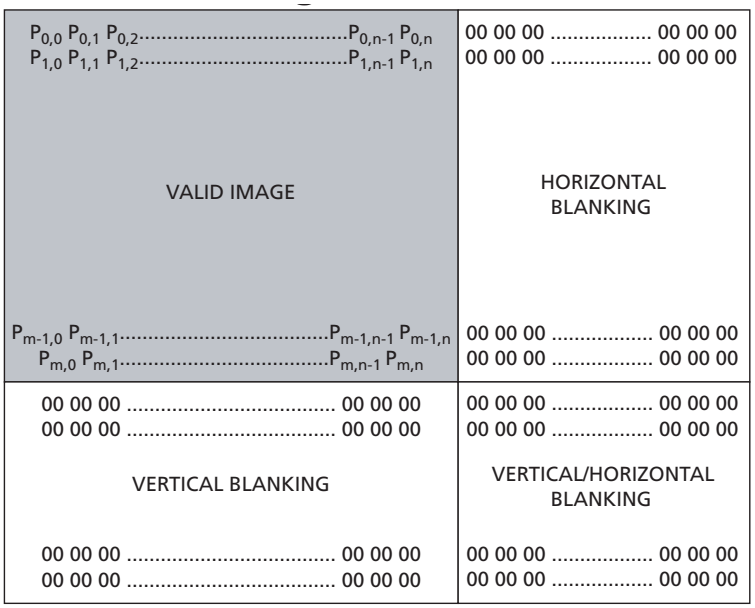


Figure 5.4: Spatial readout of a frame from TRDB-DC2 CMOS image sensor. Valid image data is surrounded by horizontal blanking and vertical blanking [26].

5.3.3 Module: Bayer Color Pattern Data to 30-bit RGB

As the captured data are in the Bayer color pattern, a conversion from Bayer pattern to conventional RGB format is necessary before the image data are further processed. In the Bayer color pattern, each pixel is presented by a single color, which is red, green or blue. The conversion is to combine each block of four pixel data so that each pixel can be represented by the combination of all red, green and blue components, as the way in which the conventional RGB format presents. By this conversion, the size of the frame with RGB format is half of the frame with Bayer color pattern. For example, a frame of size $1280H \times 1024V$ in Bayer color pattern is converted to a frame of resolution $640H \times 512V$ in RGB format.

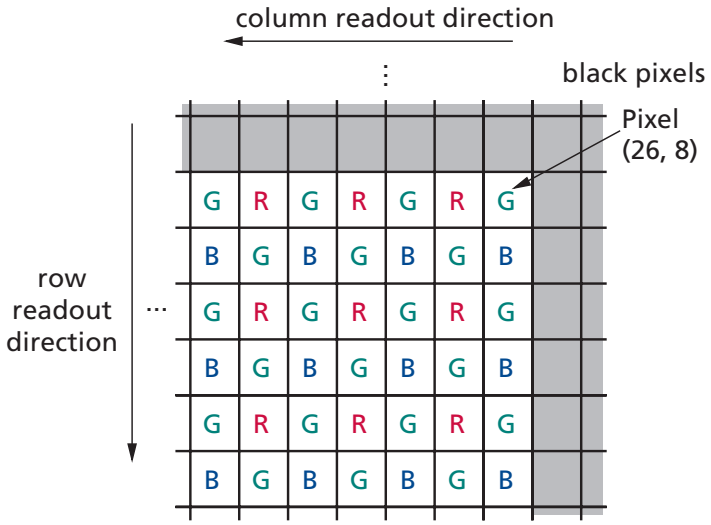


Figure 5.5: The Bayer color pattern. Valid image data starts at Pixel(26,8) as the first 26 columns and the first 8 rows of pixels are optically black [26].

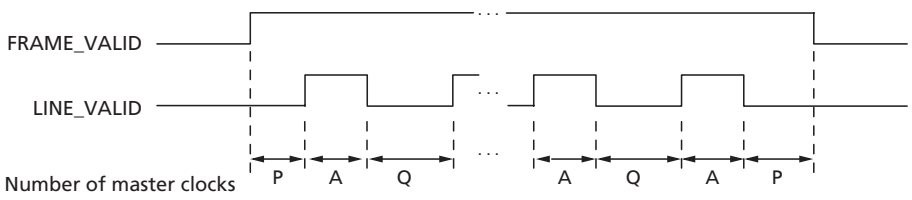


Figure 5.6: Timing example of pixel data. Parameter P is the frame start/end blanking. Parameter A is the active data time. Parameter Q is the horizontal blanking [26].

5.3.4 Module: Conversion from RGB to Grayscale

One of the accepted formula for RGB to grayscale conversion [12] is

$$Grayscale = 0.299 \times R + 0.587 \times G + 0.114 \times B. \tag{5.1}$$

However, this formula is too complicated for hardware to implement because of its floating-point computation. According to this formula, G component takes far more weight than R and B. Therefore, for the sake of simplicity, the grayscale can be approximately represented by G component. It is much easier for hardware to implement by this way. The output data only contain G component with a width of 10 bits and then stored in SDRAM at a later stage.

5.3.5 Module: Multi-port SDRAM Controller

By the multi-port SDRAM controller module, the 8Mbyte SDRAM is configured as two banks with two read ports and two write ports. Each bank is 2M $\times$ 16 bits. With this configuration, the 10-bit grayscale pixel data from each image sensor can be stored into each of the two banks respectively. The SDRAM operates with a 100MHz clock signal, while two read ports both run at 25MHz VGA control clock and two write ports both run at 25MHz image sensor pixel clock *PIXCLK*. Figure 5.7 shows the SDRAM controller configuration code. And Figure 5.8 depicts the two-bank SDRAM configuration. The two write ports and two read ports are constructed by FIFOs in order that the pixel data can be transferred between different clock signals. The codes from top module *DE2_CCD.v* shows how the 4 ports write and read the SDRAM.

5.3.6 Module: VGA Controller

The VGA controller module controls and transfers pixel to the VGA port according to the VGA timing table, with VGA parameters such as horizontal start position *X_Start*, vertical start position *Y_Start*, horizontal synchronization *H_Sync* and vertical synchronization *V_Sync*. The module operates with a 25MHz VGA control clock *VGA_CTRL_CLK*.

```
Sdram_Control_4Port      u6       (          //        HOST Side
        .REF_CLK(CLOCK_50),
        .RESET_N(1'b1),
//      FIFO Write Side 1
        .WR1_DATA(      {6'h00,
                        sCCD_G_1[9:0]}),///m
        .WR1(sCCD_DVAL_1),
        .WR1_ADDR(0),
        .WR1_MAX_ADDR(640*512),
        .WR1_LENGTH(9'h100),
        .WR1_LOAD(!DLY_RST_0),
        .WR1_CLK(CCD_PIXCLK_1),
//      FIFO Write Side 2
        .WR2_DATA(      {6'h00,
                        sCCD_G_2[9:0]}),///m
        .WR2(sCCD_DVAL_2),
        .WR2_ADDR(22'h100000),

        .WR2_MAX_ADDR(22'h100000+640*512),
        .WR2_LENGTH(9'h100),
        .WR2_LOAD(!DLY_RST_0),
        .WR2_CLK(CCD_PIXCLK_2),
//      FIFO Read Side 1
        .RD1_DATA(Read_DATA1),
        .RD1(Read),
        .RD1_ADDR(640*16),
        .RD1_MAX_ADDR(640*496),
        .RD1_LENGTH(9'h100),
        .RD1_LOAD(!DLY_RST_0),
        .RD1_CLK(VGA_CTRL_CLK),
//      FIFO Read Side 2
        .RD2_DATA(Read_DATA2),
        .RD2(Read),
        .RD2_ADDR(22'h100000+640*16),

        .RD2_MAX_ADDR(22'h100000+640*496),
        .RD2_LENGTH(9'h100),
        .RD2_LOAD(!DLY_RST_0),
        .RD2_CLK(VGA_CTRL_CLK),
//      SDRAM Side
        .SA(DRAM_ADDR),
        .BA({DRAM_BA_1,DRAM_BA_0}),
        .CS_N(DRAM_CS_N),
        .CKE(DRAM_CKE),
        .RAS_N(DRAM_RAS_N),
        .CAS_N(DRAM_CAS_N),
        .WE_N(DRAM_WE_N),
        .DQ(DRAM_DQ),
        .DQM({DRAM_UDQM,DRAM_LDQM}),
        .SDR_CLK(DRAM_CLK)      );
```

Figure 5.7: SDRAM controller configuration Verilog code.

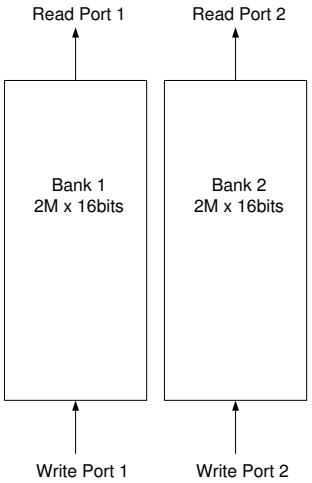


Figure 5.8: The two-bank SDRAM configuration.

5.3.7 Module: Window Based SAD Correlation Stereo Algorithm

The window based SAD correlation algorithm is implemented in the main processing module. Since this module is connected to the read side of the SDRAM, it is synchronized to the 25MHz VGA control clock *VGA_CTRL_CLK*. As shown in Figure 5.1, the stereo algorithm module contains line buffers, register arrays, SAD computations and a minimum selector.

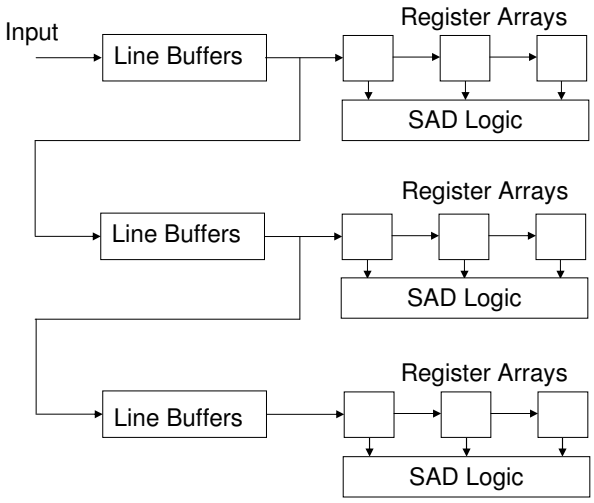


Figure 5.9: Line buffer structure for the primary camera. Line buffers and register arrays hold pixel values so that pixels from multiple scanlines can be processed at the same time.

The line buffers are built by the built-in Intellectual Property (IP) core called

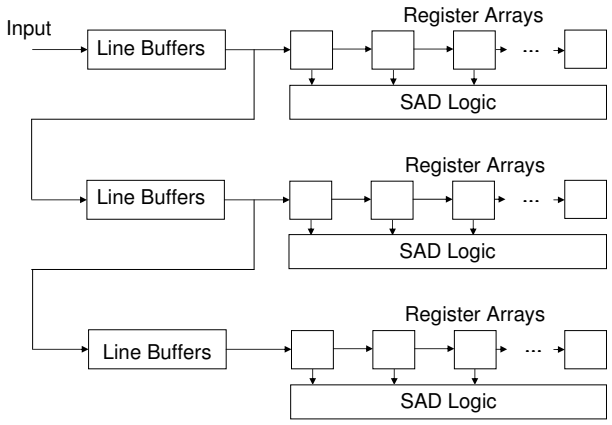


Figure 5.10: Line buffer structure for the secondary camera. Line buffers and register arrays hold pixel values so that pixels from multiple scanlines can be processed at the same time. Larger register arrays hold previous pixel values of the same scanline for the disparity search.

Altera Megafunction, using the on-chip M4K memory block. The length of each line buffer is equal to the image width. The architectures of the line buffers and the register arrays are depicted in Figure 5.9 and Figure 5.10. For a 3×3 window size, there are 3 line buffers and the register arrays are 3×3 with respect to the primary camera (Camera 0). For the secondary camera (Camera 1), the architecture is the same as the primary camera, except that the register arrays are much larger so that pixel data of different disparity levels can be retained for searching the minimum SAD value. For example, the size of the register array is 63×3 when the disparity searching range is 60. Notice that we only consider positive disparities in this implementation as the cameras are positioned to keep the lens axes in parallel. In this architecture, the left camera is the primary camera and the positive disparities are searched from the right camera as the secondary camera. If the right camera is the primary camera, our implementation is not intended to search for negative disparities from the left camera. Hence, in the case of right camera as primary camera, it is unable to search for the correct correspondences in our current implementation. Figure 5.11 illustrates the timing waveform of the line buffer architecture.

The SAD computation module is built by a set of parallel adders in a pipeline structure. There is one SAD module instantiation for each disparity searching level in order that the SAD computations for all disparity levels are able to operate at the same time. The minimum selector is constructed by multiple stages of comparators

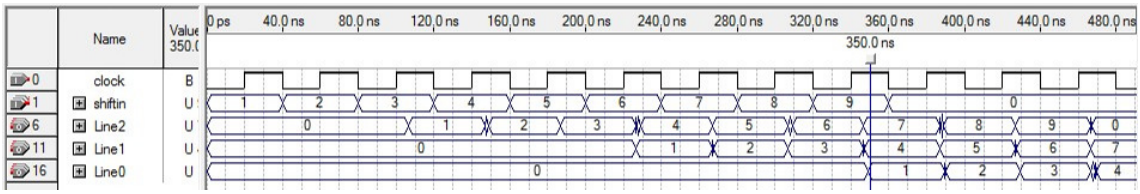


Figure 5.11: Timing waveform of the line buffer architecture. Serial data are shifted in via input port *shiftin* at every rising edge of *clock*. After 9 clocks, values 1, 4 and 7 appear at the outputs of the line buffers at the same time.

so as to locate the minimum SAD value. When the disparity searching level ranges from 0 to D, the number of stages of the comparators is the smallest integer that is not less than $log_2(D+1)$.

5.3.8 Frame Rate of the Image Sensor

The frame rate of the image sensor is programmable by setting the registers. It is also affected by the pixel clock frequency. The frame time can be calculated as follows

$$FrameTime = (RowWidth + VBLANK_{Reg}) \times (ColumnWidth + HBLANK_{Reg}) \times PIXCLK_{Period} \quad (5.2)$$

In our project, row width is set with the value of 1024, while column width is set with the value of 1280. Meanwhile, *VBLANK Reg* and *HBLANK Reg* are 25 and 136 respectively. The pixel clock frequency is 25 MHz. Therefore, the frame rate is approximately 16.83 frames per second (fps) at 25 MHz, with the number of integration rows (which is controlled by setting the exposure time) less than or equal to $(1024 + 25)$.

Chapter 6

Evaluation

This chapter gives a detailed evaluation of the proposed algorithm implementation in terms of results, quality, processing time and resource utilization. Firstly, a simulation of the implementation is described and the simulation results are discussed. Secondly, the Middlebury stereo evaluation platform [10] developed by Middlebury College is introduced and our implementation is evaluated using the standard stereo images on this platform. Lastly, the performance of our implementation as well as the processing time and resource utilization are analyzed.

6.1 Simulation of the Algorithm Implementation

In this evaluation, the simulation is a logic level simulation of the algorithm implementation. It contains three steps. First, color stereo images are converted into grayscale images and then the grayscale images are written into text files in Matlab. Second, the data in the text files are read and transferred to the Verilog module of the stereo matching algorithm as the input data in the simulation tool ModelSim-Altera. The functional simulation of the stereo algorithm is executed in ModelSim-Altera and the output data of the simulation are written into new text files. Finally, the output text files are read in Matlab and analyzed. The procedure is shown in Figure 6.1. In this simulation six sets of standard stereo images are used, which are the Tsukuba, Teddy, Cones, Venus, Bigtruck and Barn. They are typical stereo images since they represent different views of stereo vision in general. The grayscale levels of these stereo images range from 0 to 255. Hence, the grayscale level is represented by 8 bits in digital logic circuits.

Recall that there are two parameters in the window based SAD correlation algorithm: the window size and the disparity searching range. Considering the limited

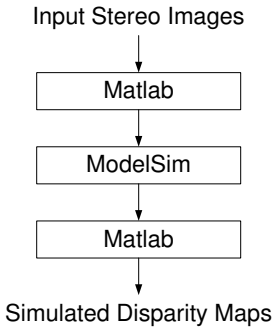


Figure 6.1: Simulation procedure.

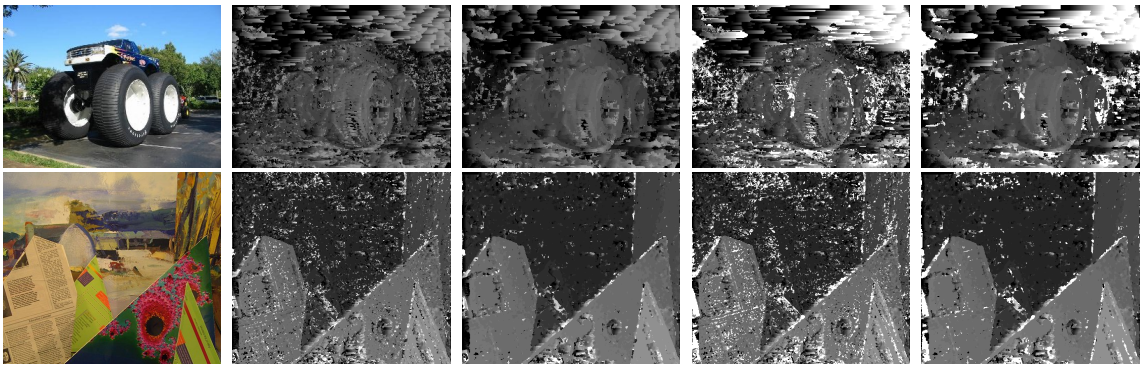


Figure 6.2: Simulated disparity maps for the Bigtruck and Barn images. Columns from left to right: Original stereo left image; disparity map by SAD correlation algorithm with window size 3×3 and searching range 30; disparity map by SAD correlation algorithm with window size 5×5 and searching range 30; disparity map by SAD correlation algorithm with window size 3×3 and searching range 60; disparity map by SAD correlation algorithm with window size 5×5 and searching range 60. Disparity values are scaled for better display.

logic resources available in the Altera Cyclone® II FPGA on the DE2 board, the parameters are set as the window sizes of 3×3 and 5×5 respectively, and the searching ranges of 30 and 60 levels respectively in the logic level simulation. Figure 6.2 illustrates the simulated disparity maps for the Bigtruck image and the Barn image. As seen in the simulated disparity maps, the window based SAD correlation algorithm using a 5×5 window performs better with less noise for either a searching range of 30 or 60. However, the relative distance information of different objects can still be indicated in the disparity maps with a 3×3 window, although the disparity maps are noisier. The disparity searching range defines the maximum disparity level. Therefore, for a computed disparity map, the range of pixel values with a larger searching range spreads to a larger extent. For this reason, the brightness of

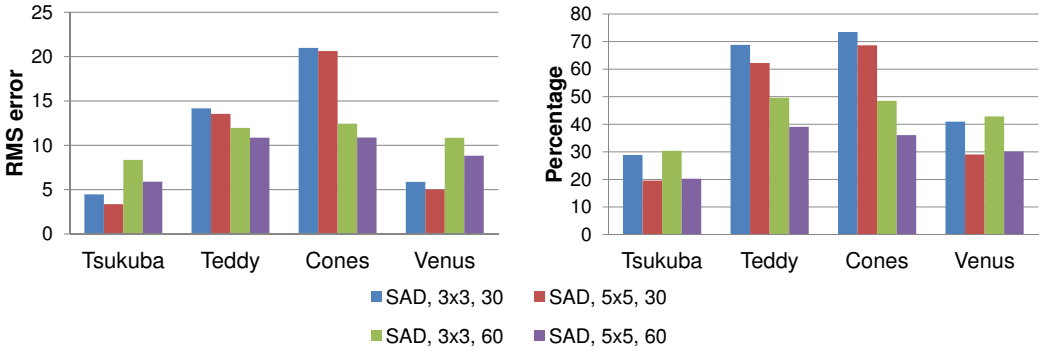


Figure 6.3: Numerical evaluation of the simulated disparity maps for Tsukuba, Teddy, Cones and Venus images. Measurements are RMS error and percentage of bad matching pixels. The simulations of the SAD correlation algorithm use window size 3×3 with searching range 30, window size 5×5 with searching range 30, window size 3×3 with searching range 60 and window size 5×5 with searching range 60, respectively.

the computed disparity maps differs with searching ranges, especially in the regions of error correspondences. In the disparity maps for the Bigtruck image, errors occur noticeably in the blue sky area of the background. This is due to the sky area with little texture taking up the entire width of the image and no feature points available for correspondences. Therefore, it is unable to search for the correct correspondences in this area.

6.2 Analysis of the Simulation Results

In this section, the simulated disparity maps are analyzed and discussed. In accordance to our previous analysis, the same standard stereo image pairs are used in this simulation. These stereo image pairs include the Tsukuba, Teddy, Cones and Venus. Also, the percentage of bad matching pixels and the RMS errors are used to evaluate the quality of the simulated disparity maps. Meanwhile, the visual quality of the disparity maps is discussed. Figure 6.3 depicts the numerical measurements of all simulated disparity maps.

In the simulation, the algorithm uses the window sizes of 3×3 and 5×5. The disparity searching ranges are 30 and 60. As illustrated in Figure 6.3, the algorithm with a 5×5 window outperforms the algorithm with a 3×3 window for all four pairs of stere images, in terms of the percentage of bad matching pixels and the RMS errors.

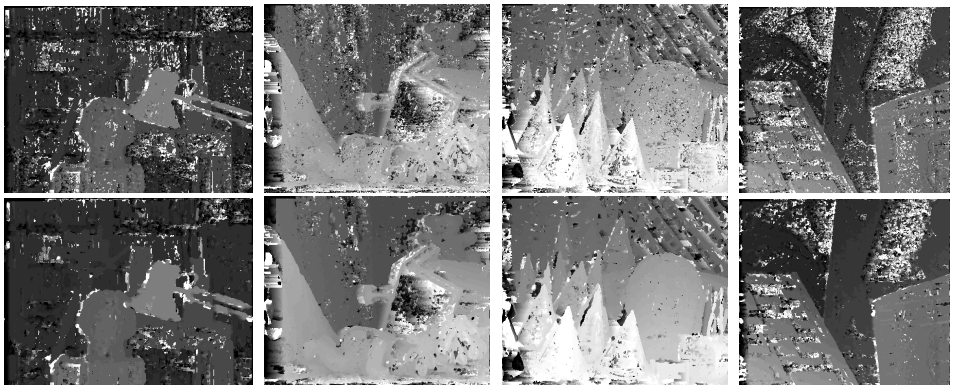


Figure 6.4: Simulated disparity maps for Tsukuba, Teddy, Cones and Venus images. Disparity maps simulated using the SAD correlation algorithm use window size 3×3 with searching range 60 and window size 5×5 with searching range 60, from left to right, respectively. Disparity values are scaled for better display.

On the other side, the searching range of 60 has a better or equal performance to the searching range of 30 for these stereo images. This can be explained by the actual maximum disparity values of these stereo pairs. In the ground truth images, the maximum disparity value of the Tsukuba images is 14; the Teddy 53, the Cones 55 and the Venus 20. For a searching range of 30, the actual maximum disparity values of the Teddy and Cones images is out of the searching range, so a good quality is not expected. Compared to the measurements in Chapter 4, the simulation results here confirm that the Verilog design of the window based SAD correlation algorithm is working in functionality as expected.

Some of the simulated disparity maps are shown in Figure 6.4. Figure 6.4 only demonstrates the disparity maps with the searching range of 60. It can be clearly seen that the disparity maps with a 3×3 window contain more noise than the disparity maps with a 5×5 window. From these observations, the simulated disparity maps are visually very close to the computed disparity maps discussed in Chapter 4. For the Venus images, because the window size in the simulation is not large enough for the textureless regions, noise occurs in these regions.

When the Verilog design is synthesized, the utilization of logic resources needs to be taken into account. Figure 6.5 shows the logic resource utilizations when the Verilog designs with different parameters are synthesized to the Altera Cyclone® II 2C35 FPGA.

It is noticed that the number of total logic elements is approximately doubled

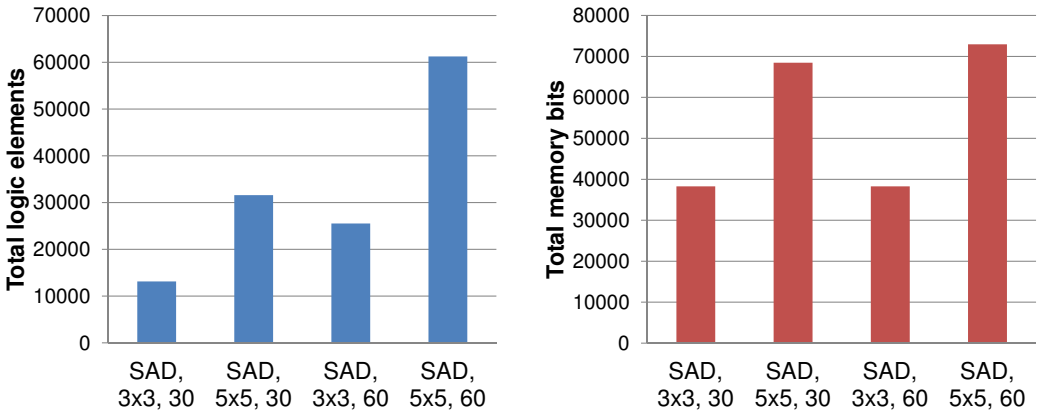


Figure 6.5: Resource utilizations. Measurements are the total logic elements used and the total memory bits used. The simulations of the SAD correlation algorithm use window size 3×3 with searching range 30, window size 5×5 with searching range 30, window size 3×3 with searching range 60 and window size 5×5 with searching range 60, respectively.

when the searching range is increased from 30 to 60. Similarly, the total logic elements by a 5×5 window is slightly more than twice as the total logic elements by a 3×3 window. This explains that the on-chip logic elements are used to construct the register arrays and the pipeline registers by the synthesizer. On the other side, the on-chip memory bits are primarily used to build the line buffers. The 5-line buffers use approximately twice memory bits as the 3-line buffers do. It is also noticed that the implementation with a 5×5 window and the searching range of 60 uses 61257 logic elements which exceed the total number of logic elements available in the Cyclone® II 2C35 FPGA.

On the Middlebury Stereo Vision webpage [10], a tool is provided to evaluate the performances of stereo matching algorithms. The methodology used by this evaluation tool is to compute the percentage of bad matching pixels for the produced disparity map based on known ground truth image. The stereo images used for the measurements in this tool are the Tsukuba, Teddy, Cones and Venus images. The produced disparity maps of these four image pairs are sent to the Middlebury evaluation tool and the tool returns a numerical evaluation as show in Table 6.1. Notice that this tool computes the percentage over three kinds of regions in the images, which are the entire image, the non-occluded regions and the regions with object boundaries. Non-occluded regions are those regions that can be seen in both left

Table 6.1: Middlebury evaluation results showing the percentage of bad matching pixels.

Algorithm		3×3, 30	3×3, 60	5×5, 30	5×5, 60
Tsukuba	nonocc	27.2	27.8	16.4	17.1
	all	28.9	29.4	18.1	18.8
	disc	33.4	29.5	25.8	26.9
Venus	nonocc	39.2	41.1	26.8	27.8
	all	40.2	42.1	28.0	29.0
	disc	34.4	35.5	29.5	30.3
Teddy	nonocc	64.6	42.9	57.1	30.9
	all	68.3	48.6	61.6	37.8
	disc	78.8	43.7	76.8	38.0
Cones	nonocc	69.5	40.3	64.2	26.1
	all	72.7	46.8	67.7	34.0
	disc	62.8	42.2	57.8	32.7
Average		52.5	41.7	43.9	29.9

Note: nonocc: only non-occluded regions;

all: the entire image;

disc: only regions with discontinuities;

average: the average of four sets of entire images.

and right images. Regions with discontinuities are regions with object boundaries.

Table 6.1 shows the measured percentages of bad matching pixels for the four pairs of stereo images by Middlebury evaluation tool, with respect to three kinds of regions. The threshold value for computing the percentages is set to 1 in this measurement. The percentages over the entire images in Table 6.1 agrees with our preceding measures in this section. Moreover, the qualities of the disparity maps over the non-occluded and discontinuous regions can be examined in detail here. It is seen that a 5×5 window always performs better in both non-occluded and discontinuous regions. For the searching range, a searching range of 60 never loses to a searching range of 30. There is no measurement for the occluded regions, since correct correspondences can hardly be found in occluded regions. Errors are highly expected to

occur when no particular strategy is used to determine the correspondences in the occluded regions.

The Middlebury evaluation tool is an effective way to evaluate the qualities of disparity maps numerically. However, it is not a comprehensive way to evaluate the performance of a stereo matching algorithm. The reason is that the tool only focuses on the produced disparity maps rather than the process of an algorithm. Thus, it is not intended to measure the efficiency of an algorithm, such as the processing time, by this tool.

6.3 Performance of the Stereo Vision System

This section demonstrates the performance of the implemented stereo system on the Altera DE2 board. Firstly, the overall setup of the system is given. Then, the run-time performance of the system is discussed.

6.3.1 Overall Setup of the Stereo Vision System

The stereo system consists of three main parts: the image sensors, the Altera DE2 board and the LCD monitor. It is shown in Figure 6.6.

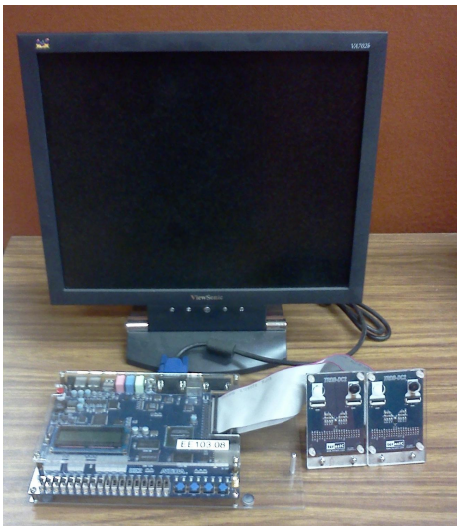


Figure 6.6: Stereo vision system.

Most parameters of the stereo system are pre-determined and fixed once the Verilog design is synthesized and downloaded into the FPGA chip. The processing window size and the disparity searching range, which are the two key parameters in

Table 6.2: Pre-determined parameters.

Stereo algorithm	SAD correlation
Window size	3×3
Searching range	60
Input image resolution	1280×1024
Output image resolution	640×480

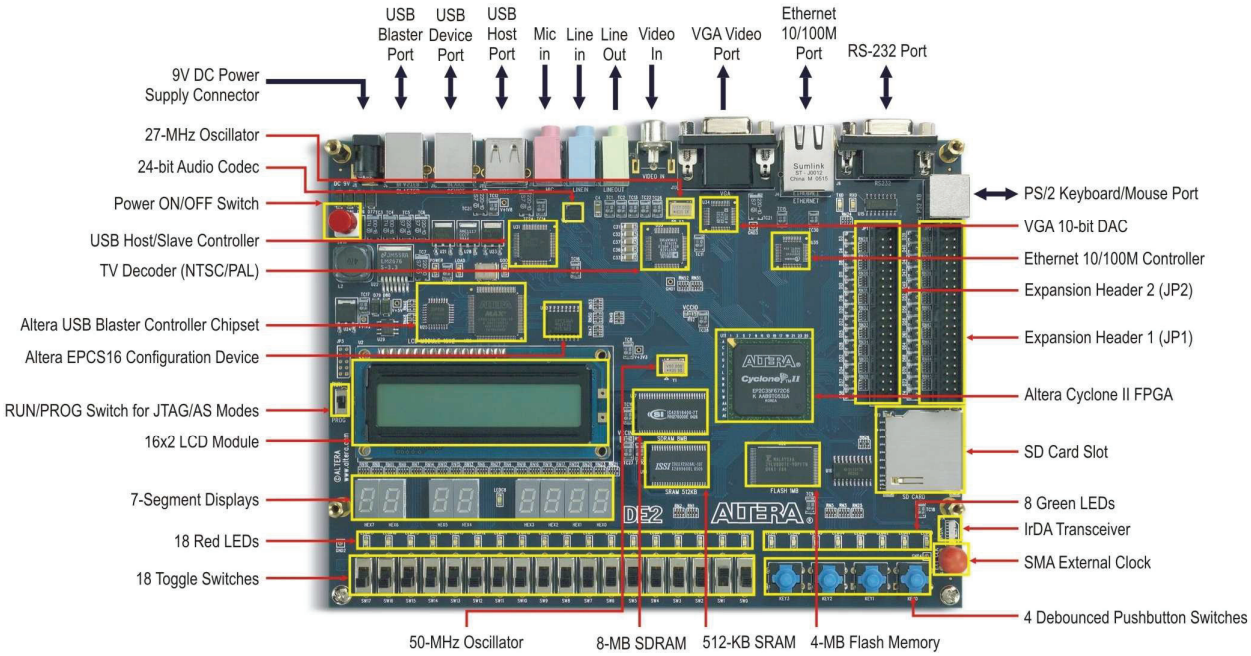


Figure 6.7: Layout of the Altera DE2 board [1].

the window based SAD correlation algorithm, are also pre-determined and unchanged by the Verilog design. Such pre-determined parameters are shown in Table 6.2.

Note that the resolution of the image sensor is 1280×1024 in Bayer color pattern. After the conversion into RGB format, the effective resolution is 640×480 in fact. Due to the limitation of logic resources in the Altera Cyclone® II 2C35 FPGA, a SAD correlation algorithm with window size of 3×3 and searching range of 60 is implemented here.

With the switches and buttons available on the Altera DE2 board, several features of the stereo system such as free run, snapshot and shutter time can be controlled by the operator. Figure 6.7 demonstrates the layout of the Altera DE2 board.

As shown in Figure 6.7, there are 18 switches SW0 to SW17 and 4 press buttons

KEY0 to KEY3. SW15 to SW0 are used to set the shutter time of the image sensors, which controls the exposure. Specifically, the shutter time is represented by a 16-bit digit. Each bit of the digit is set to 0 or 1 by a switch on the board, where SW15 is the most significant bit and SW0 is the least significant bit. SW17 selects the video stream that is displayed on the LCD monitor. When SW17 is set to 1, the monitor displays the stream of disparity maps. When SW17 is set to 0, the monitor displays the stream of left grayscale images. SW16 has no operation in this system so far. KEY0 is the reset button. KEY3 sets the system into free-run mode while KEY2 sets it to take a snapshot and freeze the frame. By pressing KEY1, the shutter time set by the switches is sent to configure the image sensors.

Since the images captured by the pair of image sensors are assumed to be rectified, the two image sensors have to be mounted with their projective planes being coplanar and their lens axes in parallel. The alignment of the image sensors is the key to the performance of this stereo system. Once the image sensors are mounted, the distance between the two sensors are $6.15cm$ in our current setting. With a searching range of 60 disparity levels, the system works best to indicate the relative positions of objects when the objects are, by our observation, at approximately 0.5 to 1.5 meters away from the image sensors.

6.3.2 Run-Time Performance of the Stereo Vision System

The stereo system runs with two 25MHz clock signals. On the image sensor side, the Altera DE2 board communicates with the image sensors by a 25MHz CCD pixel clock. On the monitor side, the Altera DE2 board communicates with the LCD monitor by a 25MHz VGA control clock. The system is of a fully pipelined streaming structure with no operation that halts the processing of each frame. Therefore, the frame rate of the system is the same as the frame rate of the image sensors. From the discussion in Chapter 5, the frame rate of the image sensors is approximately 16.83 frames per second (fps). Hence, the frame rate of the system is also 16.83 frames per second. Given that the human visual system retains each individual image for one-fifteenth of a second [19], a frame rate of 16 frames per second is sufficiently close to produce an sensation of visual continuity.

For viewing the visual quality of the produced disparity maps, some snapshots

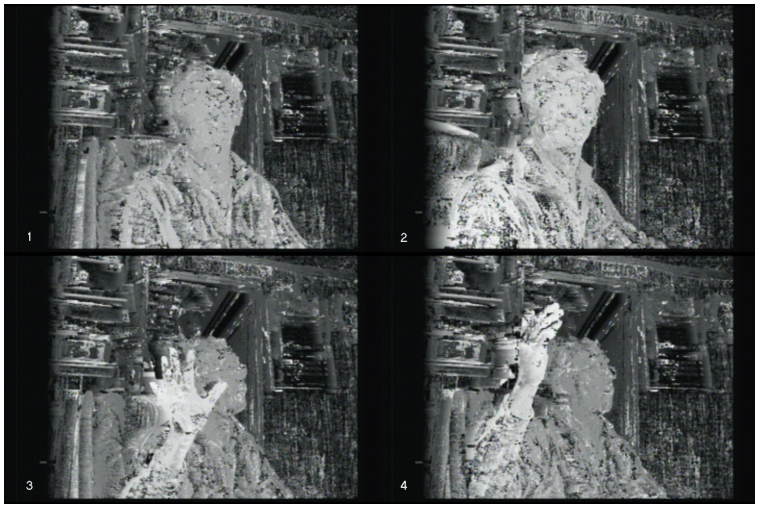


Figure 6.8: Snapshots of the disparity maps produced by the run-time stereo vision system.

from the real-world scene are taken from the LCD monitor. These are shown in Figure 6.8. As is seen in Figure 6.8, the foreground and background can be clearly distinguished by the disparity values. The stereo system provides excellent stream of disparity maps to convey the information on relative distances of persons or objects in the scene. These snapshots were taken with a different Altera DE2 board running the same FPGA design from our previous work [25]. Some AVI movie clips were captured and recorded.

In regard to the logic resources utilization of the system, it is shown in Table 6.3. With the fact that the logic resources is not rich in this FPGA chip, the system takes up most of the logic resources built in the chip.

Table 6.3: Resource utilization and frame rate of the system.

Stereo algorithm	SAD, 3×3, 60
Output image resolution	640×480
Total logic elements	27091 / 33216 (82%)
Total combinational functions	20193 / 33216 (61%)
Dedicated logic registers	25972 / 33216 (78%)
Total memory bits	128824 / 483840 (27%)
Frame rate (fps)	16.83

Chapter 7

Conclusion and Future Work

7.1 Conclusion

A stereo vision system is implemented on a FPGA platform, using the window based SAD correlation stereo correspondence algorithm proposed in this thesis. The stereo vision system uses two Terasic TRDB-DC2 CMOS image sensors to capture pairs of stereo images. The produced disparity maps are displayed via a VGA port onto an LCD monitor. The main part of the system is implemented on an Altera DE2 board featuring an Altera Cyclone® II 2C35 FPGA. In the current system, the resolution of input images from the image sensors is 640×480, while the resolution of the output disparity maps is also 640×480. The system runs at a frame rate of 16.83 frames per second (fps).

At the beginning chapters in this thesis, several stereo matching algorithms are analyzed, including global stereo algorithms and local stereo algorithms. Two global stereo algorithms are examined: dynamic time warp (DTW) algorithm and DTW with coarse quantization algorithm. On the other hand, three window based local stereo algorithms, which are the SAD correlation, the SSD correlation and the Census transform correlation algorithms, are analyzed. The performance of each algorithm is evaluated by varying its parameters. For the global algorithms, dynamic time warp algorithm produces disparity maps with high-quality in terms of RMS errors and percentage of bad matching pixels. However, the considerable complexity of the DTW algorithm requires massive computations, which make it difficult to meet the requirements of fast processing. The DTW with quantization algorithm solves the issue of slow processing time by a linear quantization step, which reduces the number of computations, but it loses the accuracy of the produced disparity maps in terms of RMS errors and percentage of bad matching pixels. For the local algorithms, the

window size parameter and the disparity searching range parameter play the key roles in the performance of the algorithms. The window size parameter defines the size of the 2D processing window. A small window size works well at object boundaries in the image while a large window size performs well at textureless regions. As the window size increases, the processing time increases accordingly. Therefore, a moderate window size is a compromise between the quality of disparity maps and the processing time. The disparity searching range parameter defines the maximum disparity levels for which the algorithm searches. The optimal choice of this parameter depends on image patterns but a larger searching range is more likely to avoid incorrect correspondences for most of the cases. Regarding the computation of correlations, the SSD and SAD correlations perform similarly and better than Census transform correlation when the same combination of parameters is used. From the analysis results, the SAD correlation algorithm is proposed for the implementation, considering that it has a streaming structure that is suitable for hardware implementation and it produces decent disparity maps in terms of RMS errors, percentage of bad matching pixels and visual quality.

The window based SAD correlation algorithm is implemented using Verilog hardware description language (HDL). The Verilog design of the algorithm is simulated using ModelSim-Altera simulator with regard to its functionality. In the simulation, the input stereo images are transferred into the algorithm design module on a scanline basis. The simulation results are computed disparity maps which are output to Matlab for display. With our targeted FPGA chip, only four combinations of parameters are implemented and simulated. The parameters are the window size of 3×3 and 5×5 as well as the searching range of 30 and 60. The simulation results conform to the preceding analysis.

In the last step, an implementation of the algorithm is realized in an Altera Cyclone® II 2C35 FPGA on the Altera DE2 board. Along with two image sensors and an LCD monitor, the Altera DE2 board composes a stereo vision system that is able to sense the depth information in a scene. The stereo vision system successfully implements a window based SAD correlation algorithm into a single FPGA chip. The system runs at a video rate of 16.83 frames per second (fps) and is able to be used in real-time stereo vision applications. When only the stereo algorithm

implementation is considered, the maximum clock frequency is analyzed to reach approximately 202MHz by Altera Quartus II Classic Timing Analyzer. This maximum clock frequency is among the highest maximum clock frequencies mentioned in other researchers' implementations on FPGA. The architecture where one pixel data is processed in one clock cycle makes our system possible to run at a much higher frame rate when high-speed cameras are used. Different from the DSP implementation using a serial approach, the FPGA implementation fully utilizes parallel processing in hardware so as to accelerate the processing. Our system establishes a good starting point to explore the stereo vision system on a FPGA platform.

Although the real-time stereo vision system is successfully designed and implemented, there are still limitations with the current system. Due to the limited logic resources available in the current Altera Cyclone® II 2C35 FPGA, the window based SAD correlation algorithm with the optimal parameters is unable to fit into the FPGA chip. Therefore, it reduces the quality of the produced disparity maps. A more advanced FPGA chip and its development board could be a better choice for a stereo vision system with greater flexibility in terms of image dimensions, disparity ranges and frame rates.

7.2 Future Work

In our current research, several common stereo correspondence algorithms are analyzed and a proposed window based SAD correlation algorithm is implemented on a FPGA platform to build a stereo vision system. Our future work would focus on two phases. One is the improvements of the algorithm and the other is the implementation strategy.

In the algorithm phase, a sub-pixel disparity estimate can be added to the produced disparity maps in order to improve the visual quality [23][29]. So far, the disparities computed by the proposed algorithm are all integers. The sub-pixel estimate provides fractional disparities by fitting a curve to the matching costs at discrete disparity levels. It is an effective way to increase the resolution of a disparity map.

In the implementation phase, an integration of hardware and software implementations will be explored. The software solution provides flexibility of the system while

the hardware solution provides acceleration of the processing speed. Altera NIOS II® processor is a programmable embedded processor that can be built in the Altera FPGA using the on-chip logic resources. With the Altera NIOS II processor, the stereo matching algorithm can be implemented in a software approach and speeded up by hardware accelerators. For a better performance, some state-of-the-art FPGA chips are embedded with a hard processor. For example, the Altera Cyclone® V SoC FPGA integrates a Single- or Dual-Core ARM Cortex-A9 Processor which provides more design flexibility.

References

[1] *DE2 Development and Education Board User Manual*, Altera Corporation, 2007.

[2] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *Computer Vision, 1998. Sixth International Conference on*, January 1998, pp. 1073 –1080.

[3] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222 –1239, November 2001.

[4] T. H. Cormen, *Introduction to algorithms.* Cambridge, Mass. : MIT Press, 2001.

[5] J. Diaz, E. Ros, R. Carrillo, and A. Prieto, "Real-time system for high-image resolution disparity estimation," *Image Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 280 –285, January 2007.

[6] D. Ellis, *Dynamic Time Warp (DTW) in Matlab*, http://labrosa.ee.columbia.edu/matlab/dtw, 2003.

[7] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt, "Real-time stereo by using dynamic programming," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, June 2004, p. 29.

[8] D. Forsyth and J. Ponce, *Computer vision: a modern approach.* Upper Saddle River, N.J. ; London : Prentice Hall, 2003.

[9] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge, UK ; New York : Cambridge University Press, 2000.

[10] *Middlebury Computer Vision*, http://vision.middlebury.edu/stereo/, 2012.

[11] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1180 – 1202, 2010, special issue on Embedded Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314210000895

[12] ITU, "Recommendation itu-r bt.470-7, convertional analog television systems," International Telecommunication Union, Tech. Rep., 1998.

[13] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. W. Jeon, "Fpga design and implementation of a real-time stereo vision system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 1, pp. 15 –26, January 2010.

[14] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, June 1996, pp. 196 –202.

[15] S. Longfield and M. Chang, "A parameterized stereo vision core for fpgas," in *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, April 2009, pp. 263 –266.

[16] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-7, no. 2, pp. 139 –154, March 1985.

[17] *Triclops, Technical Manual*, Point Grey Research Inc., 2000.

[18] Z. Qian and K. Takaya, "Dense stereo disparity maps by dynamic time warp with sparse features and fpga acceleration," in *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on*, May 2011, pp. 000 874 –000 877.

[19] P. Read, M. Meyer, and G. Group, *Restoration of Motion Picture Film*, ser. Butterworth-Heinemann Series in Conservation and Museology. Butterworth-Heinemann, 2000. [Online]. Available: http://books.google.ca/books?id=jzbUUL0xJAEC

[20] S. Sabihuddin, J. Islam, and W. MacLean, "Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array," in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, May 2008, pp. 001 461 –001 466.

[21] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, 2001, pp. 131 –140.

[22] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 7, pp. 787 – 800, July 2003.

[23] K. Takaya, "Dense stereo disparity map for video by sub-pixel dynamic time warp algorithm," in *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, May 2010, pp. 1 –4.

[24] ——, "Yuv luma/chroma quantization and sparse correspondence for real time video stereo matching," in *SICE Annual Conference 2010, Proceedings of*, August 2010, pp. 3506 –3509.

[25] K. Takaya and Z. Qian, "Fpga based stereo vision system to display disparity map in realtime," in *Information Science and Applications (ICISA), 2012 International Conference on*, May 2012, pp. 1 –4.

[26] *1.3 Mega Pixel Digital Camera Package: Frame grabber with VGA display reference design*, Terasic Technologies Inc., 2006.

[27] *Stereo-on-a-Chip Stereo Head User Manual*, Videre Design, 2007.

[28] J. Woodfill, R. Buck, D. Jurasek, G. Gordon, and T. Brown, "3d vision: Developing an embedded stereo-vision system," *Computer*, vol. 40, no. 5, pp. 106 –108, May 2007.

[29] Q. Yang, R. Yang, J. Davis, and D. Nister, "Spatial-depth super resolution for range images," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1 –8.

[30] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Computer Vision ECCV '94*, ser. Lecture Notes in Computer Science, J.-O. Eklundh, Ed. Springer Berlin / Heidelberg, 1994, vol. 801, pp. 151–158, 10.1007/BFb0028345. [Online]. Available: http://dx.doi.org/10.1007/BFb0028345