ABSTRACT

STEREO VISION ALGORITHM USING PROPAGATION OF CORRESPONDENCES ALONG AN ARRAY OF CAMERAS

by

Danyan Ganjali

One well known problem in stereo vision is the trade-off between precision and accuracy. In a conventional two camera model, as the baseline separating the two cameras becomes smaller, the images that both cameras produce become more identical. This results in a more accurate set of correspondences, however at the same time causing less precise depth measurement due to the small angle used for triangulation. Similarly, in a larger baseline model, the depth measurement is more precise while the correspondences are not as accurate as of the shorter baseline model. This thesis proposes a method that promises to solve the precision-accuracy trade-off problem. It employs an array of cameras, where each adjacent pair of cameras has a small baseline while the entire baseline of the array is relatively large. In such a system, the accuracy of a short baseline model is enjoyed, and at the same time, the produced results have the precision matched to a longer baseline model.

The proposed method in this thesis finds accurate corresponding points in the first camera pair, and then propagates this set of points along the array, searching for the same correspondences in each adjacent camera. The final result is a reliable set of correspondences between the cameras positioned at both ends of the array. The image coordinates of these points are then used to find the disparity map, followed by a triangulation algorithm for a precise depth calculation.

STEREO VISION ALGORITHM USING PROPAGATION OF CORRESPONDENCES ALONG AN ARRAY OF CAMERAS

by

Danyan Ganjali

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 2005

APPROVAL PAGE

STEREO VISION ALGORITHM USING PROPAGATION OF CORRESPONDENCES ALONG AN ARRAY OF CAMERAS

Danyan Ganjali

Dr. Stanley Reisman, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT
Professor of Biomedical Engineering, NJIT

Dr. Yun-Qing Shi, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Richard Foulds, Committee Member — Date
Associate Professor of Biomedical Engineering, NJIT

BIOGRAPHIC SKETCH

Author: Danyan Ganjali

Degree: Master of Science

Date: May 2005

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Bachelor of Engineering in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005

Major: Electrical Engineering

This Thesis is dedicated to

my father John F. Ganjali, my mother, Zahra Ghasemi

and to my beloved family members, friends and teachers

ACKNOWLEDGEMENT

I would like to appreciate the support and understanding of my advisor, Dr. Stanley Reisman. Special thanks are given Dr. Richard Foulds for his academic advisement and encouragement during this project and to Dr. Yun-Qing Shi for being a teacher and an active participant in my committee.

I would like to thank the Lord for giving me two wonderful parents, John F. Ganjali and Zahra Ghasemi. Thanks for your moral, financial and spiritual support and for sacrifices that you have made throughout my life and college career. I would also wish to thank my family members, my brother Maseeh, my uncle Morteza, my stepmother Shiina, my sister Azita, and my beloved grandparents, aunts and uncles. I am grateful and indebted to all my teachers especially Mr. Morteza Jaberi and Mr. Ben Huntington. Thank you all my friends, Roohollah Miramini, Elaine Whitworth, Kevin De Paiva, Greg Hunt, Behrang Nochirwani, Majid Bahmanpour and the rest of my friends in Tehran, Honolulu, New Jersey and everywhere else.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Specific Aims

Primates are born with both eyes positioned adjacent to one another on the anterior of the face. This anterior positioning of the eyes, while it reduces peripheral vision comparative to organisms with laterally positioned eyes, is important for perception of depth. A sense of depth plays a vital role in the survival of the primates. Each individual eye can only produce two dimensional images in the brain. However, with the help of two eyes combined, the brain is able to measure distance of objects, allowing the primate to have a sense of depth. The left and right eyes are positioned in such a way that part of each field of vision is common to both. The brain and the neural system combine the two produced images to give the primate a sense of a three-dimensional world. The process of combining images from two or more sources, either biological or man made, resulting in a 3-D image is called stereovision.

Vision in man and certain animals represents a *biological* solution to the problem of binocular stereovision. Without any apparent effort, the eyes see a three-dimensional view of objects in their environment. Given the extraordinary complexity of the human brain and its visual system, and given the limited research tools currently available, it is difficult for researchers to discern the precise mechanisms of biological vision. However, there have been many advances in computer and machine vision, and different techniques and methods have been proposed and employed to approach the problem of stereovision.

Stereovision is a process directed at understanding and analyzing three-dimensional objects based on image data [2]. During this process, the stereovision

system must solve two problems, "the correspondence problem" and "the reconstruction problem". The correspondence problem consists of determining the position of a point from the first image in the second image. The distance between the corresponding points is known as disparity. Combining the disparity information for all the points produces the disparity map. The reconstruction problem, once the disparities are known, becomes a simple geometric problem that can be solved using triangulation [14]. While solving the corresponding problem an error known as the *occlusion problem* may occur. This error is due to the fact that, at different positions of each camera, there are often points that can be only seen from one camera and vice versa.

There are various applications of stereovision in different fields including robot surgery, air craft navigation, autonomous land rovers, automated industrial machines, robotics and stereomicroscopy [18],[14],[11]. The process of recovering depth is almost the same in all these applications, as all need to solve the two problems mentioned above, namely the correspondence and reconstruction problems. Once the depth is effectively recovered, it can lead to very useful results. In robotic applications, three-dimensional vision can make systems capable of dynamically building models of their environment, detecting changes, and reacting intelligently in carrying out their tasks [18]. Depth information of objects can also be used to measure the changes in biological surfaces such as soft tissue. These changes which happen over a period of time can be identified and quantified by the computer vision, and their three-dimensional representation can ultimately help in their analysis.

There are different techniques available to solve the correspondence problem. These matching techniques can be categorized into two main areas, the area-based and

feature-base techniques [2]. The area-based stereo technique uses correlation among intensity patterns in the local neighborhood of a point or pixels in one image and looks for the same or closely similar pattern in the second image. This technique has a disadvantage since it uses intensity patterns. The intensity and contrast change that occurs from changing the position in an environment results in errors, which is not desired. The feature-based techniques extract features from one image and look for the same features in the second image.

Stereovision techniques have commonly the following sequence of processing steps [2]:

- Image acquisition - this is affected by the environment including the acquisition devices, lights etc.
- Camera Modeling - known as camera calibration, this step is explained in Chapter 2.
- Feature extraction - included in feature-base techniques.
- Correspondence analysis - an automatic computer process of determining image points correspondents.
- Triangulation - a geometric technique of depth measurement given a point in an image and its correspondent in the other image. This is discussed in Chapter 2.
- Interpolation- representation of the depth data in 3-D space.

1.2 Background

This section describes the methods used to solve the correspondence problem. It first gives an overview of camera modeling and calibration, followed by a brief definition of the correspondence problem for stereo vision. It also discusses different constraints such as epipolar geometry that simplify the process. The chapter ends with description of

conditions of image acquisition, and a discussion of triangulation, interpolation, and reconstruction.

1.2.1 Camera Modeling

The objective of camera modeling is to mathematically represent the way a camera maps the world coordinates points into pixels or pixel coordinate points. Camera modeling employs mathematical equations with variables that are called camera parameters. Camera modeling or calibration is an essential part of the stereovision process and helps to determine and estimate the intrinsic and extrinsic parameters of a camera [10]. This process is explained in Chapter 2.

The camera parameters are categorized into two types. The first are the intrinsic parameters. The intrinsic parameters do not depend on the position and the orientation of the camera in space. The knowledge of these parameters can help the user to perform measurements with the camera. The second set of parameters is the extrinsic parameters which describe the position of the camera with respect to a reference frame [10].

The intrinsic parameters are necessary to link the pixel coordinate in an image with its corresponding coordinates in the camera reference frame. These parameters help characterize the optical, geometric, and digital characteristics of a camera [5]. These parameters include f, the focal length, Ox, Oy , S_x , S_y, and finally the radial distortion coefficient. (Ox, Oy) is the principal point of the image. This is the point on the image plane at which the optical ray passes the focal length through the focal point and intersects the image plane perpendicularly. S_x , S_y, are the effective size of the pixel in millimeters in the horizontal and vertical direction respectively.

The camera reference frame can be located with respect to a known reference frame called the world reference frame. The extrinsic parameters describe the position and the orientation of the camera with respect to this known reference frame. They are represented by two matrices R and T, which are the rotation and translation matrices. These two matrices are used to uniquely identify the transformation between the unknown camera reference frame and a known world reference frame [5]. It is therefore apparent that by changing the position of the camera, the extrinsic parameters change as well.

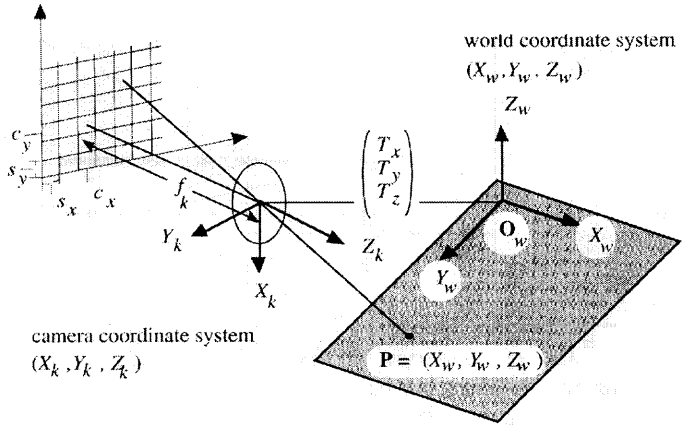

Figure 1.1 Camera and world reference frame.
(Source: Reinhard Klette et. al. 1998 [2])

As shown in Figure 1.1, the following equations are valid.

$$\begin{pmatrix} X_k \\ Y_k \\ Z_k \end{pmatrix} = R.\left(\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T \right), \quad \text{with T=} \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = \begin{pmatrix} -X_0 \\ -Y_0 \\ -Z_0 \end{pmatrix} \tag{1.1}$$

with $R = R_x.R_y.R_z$

R_x describes the rotation about X_k axis by α, the *pan angle*.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{1.2}$$

R_y describes the rotation about Y_k axis by β, the *tilt angle.*

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{1.3}$$

R_z describes the rotation about Z_k axis by γ, the *roll angle.*

$$R_y(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

With reference to E. Trucco and A. Verri [5], once the intrinsic and the extrinsic parameters are know, the relationship between the image coordinates and the world reference coordinates can be shown in a linear version of the perspective projection equation.

$$-(X_{im} - O_x)Sx = \frac{f\,\mathbf{R}_1^T(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^T(\mathbf{P}_w - \mathbf{T})} \tag{1.5}$$

$$-(Y_{im} - O_y)Sy = \frac{f\,\mathbf{R}_2^T(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^T(\mathbf{P}_w - \mathbf{T})} \tag{1.6}$$

The above equations can be written in matrix format once there is no radial distortion.

$$\begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix} = M_{int} M_{ext} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1.7}$$

where

$$\frac{X1}{X3} = X_{im} \tag{1.8}$$

$$\frac{X2}{X3} = Y_{im} \tag{1.9}$$

and

$$M_{\text{int}} = \begin{bmatrix} -\frac{f}{S_x} & 0 & O_x \\ 0 & -\frac{f}{S_y} & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & R_1^T T \\ r_{21} & r_{22} & r_{23} & R_2^T T \\ r_{31} & r_{32} & r_{33} & R_3^T T \end{bmatrix} \tag{1.11}$$

the parameters $\frac{f}{S_x}$ and $\frac{f}{S_y}$ are the focal length in pixel size in x and y directions. Matrix M_{int} is the intrinsic matrix while M_{ext} is the extrinsic matrix.

1.2.2 The Correspondence Problem

One of the main issues in stereovision is the correspondence problem. As mentioned before, the correspondence problem deals with finding the correspondent of a point from one image to another. This problem becomes more complex as ambiguities occur during the process. There is usually more than one point or pixel in the second image which is a candidate for correspondence [2],[18]. This complexity however can be made simpler with the introduction of different geometric and physical constraints. One geometric constraint that this thesis extensively depends on is the epipolar geometry. This is explained in the next section, followed by a discussion of physical constraints.

1.2.2.1 Epipolar Geometry. In order to understand the epipolar geometry, the geometry of stereovision needs to be explained first. The stereovision geometry for two cameras is shown in Figure1.2. Epipolar Geometry is the projective geometry between two views and is independent of the scene structure. Moreover, it depends only on the camera's internal parameters and the relative position and orientation of the cameras [16].

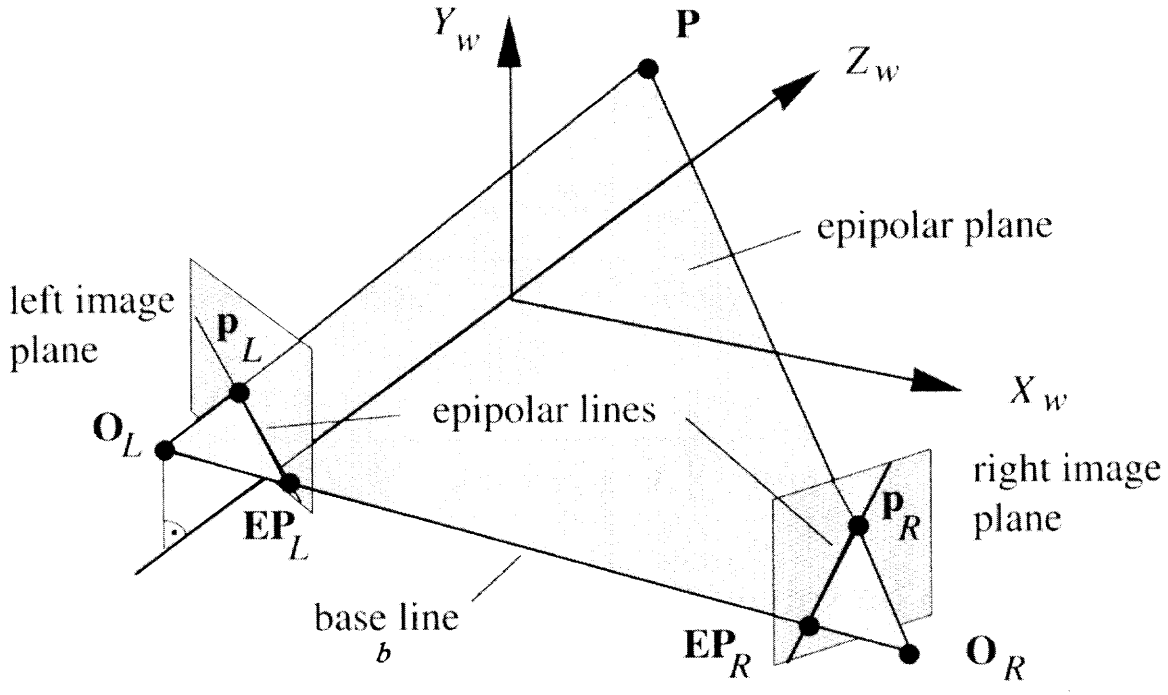

Figure 1.2 Stereovision geometry.
(Source: O. Faugeras 1996 [10])

In Figure 1.2 the line that connects the two optical centers O_L and O_R is called the *base line.* The value *b* is called the base distance. It is assumed that point P is projected onto the point PL in the left images plane. Similarly, PR is the projection of point P on the right image plane. It is desired to search for the corresponding point of PL in the second image, namely PR. This search is significantly simplified if the epipolar geometry is employed, and helps designing a time-efficient correspondence analysis process. This can constrain the search space, and instead of searching the entire second

image, only the intersection line of the second image with a so-called epipolar plane needs to be searched.

An *epipolar plane* is a plane that includes point P and the two optical centers Ol and Or. The intersection between an epipolar plane and an image plane is called an *epipolar line.* Also, once camera calibration is performed, the epipolar plane can be defined by two lines. One line passes through the projection center OL and the given image point PL of the image plane. Another line connects OL and OR and as mentioned before is called the *baseline.* Therefore once the camera calibration results are known, a selected point such as PL can specify the corresponding epipolar plane.

Epipolar geometry is a good approach to reduce ambiguities and errors in finding correspondent points. Often the general geometries are transformed to epipolar geometry, this is defined as *rectification.* Rectification is an operation to "insure a simple epipolar geometry for a stereo pair" [10]. Once rectification is performed, it can "enable the search for correspondent image features to be confined to one dimension" [17], and therefore simply the process, making the algorithm execute faster and give more accurate results.

1.2.2.2 Physical Constraints. The two physical constraints which were introduced by Marr and Poggio [14] are:

- Uniqueness: Except in rare cases, each item from either image may be assigned at most one disparity value. This condition relies on the assumption that an item corresponds to something that has a unique physical position. The exceptions can occur when two features lie along the line of sight from one eye, but are separately visible in the other eye.

- Continuity: Disparity varies smoothly almost everywhere. This condition, a consequence of the cohesiveness of matter, states that only a small fraction of the area of an image is composed of boundaries that are discontinuous in depth.

These constraints are taken into account while the algorithm for the correspondence matching is written.

1.2.3 Triangulation

Triangulation is defined as the recovery of depth information once the disparity map and the geometry of the stereo setting are known. With reference to [22] the optical setting of a *standard model* is shown in Figure 1.3.

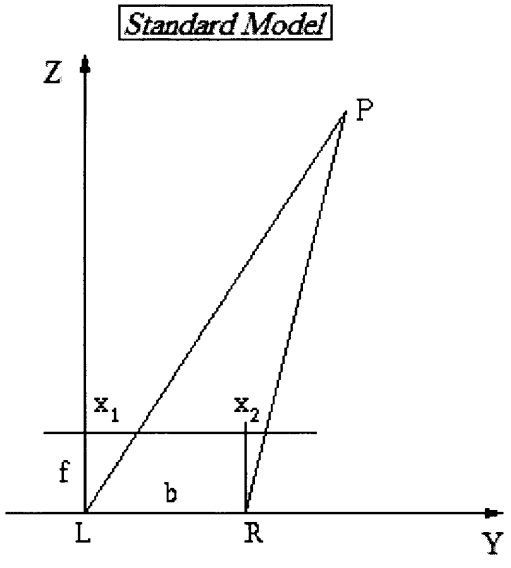

Figure 1.3 Triangulation.
(Source: L. Iocchi 2005 [13])

In this model, R and L are two pinhole cameras with parallel optical axes, with f the focal length of both cameras. The distance *b*, is the baseline and the optical axes lie on the YZ plane. This plane is parallel to the image plane. The center of the left camera L, is the origin of the world reference frame.

Once having the above settings, the equations for triangulations are the following:

$$Z = \frac{b \times f}{(x_1 - x_2)} \tag{1.12}$$

$$X = \frac{x_1 \times z}{f} \tag{1.13}$$

$$Y = \frac{y_1 \times z}{f} \tag{1.14}$$

In the case of non parallel cameras, there are a few adjustments made to the setup and the equations. In this case the optical axes are not parallel, but they both lie on the YZ plane and they intersect at a point $(0, 0, Z_0)$, this is shown in Figure 1.4. This point is called a *fixation point.*

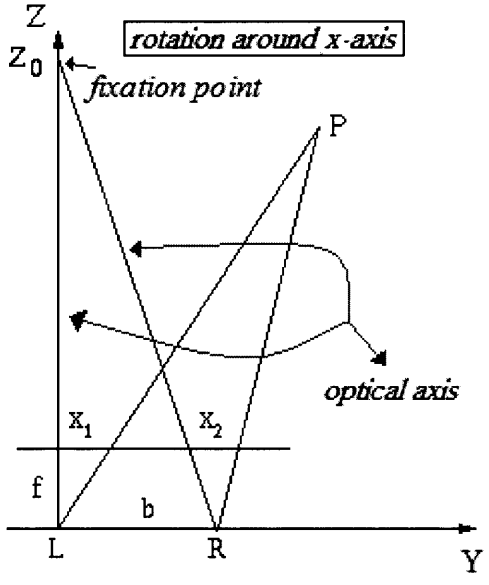

Figure 1.4 Triangulation with rotation around X axis.
(Source: L. Iocchi 2005 [13])

If θ is the rotation angle around the Y axis, then $Z_0 = \frac{b}{\tan(\theta)}$. The reconstruction formula for triangulation is then transformed into the following with reference to [22],

$$Z = \frac{b \times f}{(x_1 - x_2 + \frac{f \times b}{Z_0})} \tag{1.15}$$

$$X = \frac{x_1 \times Z}{f} \tag{1.16}$$

$$Y = \frac{y_1 \times Z}{f} \tag{1.17}$$

Since there is no axis rotation around Y and Z in this thesis, the formulas and setup corresponding to those cases are not discussed.

It is noted that once the camera position parameters such as pan angle and baseline distance is known, the triangulation and depth recovery is only the function of disparity, which is obtained during the pixel matching process.

1.2.4 Image Acquisition

In stereo vision, it is assumed that the pictures are taken at about the same time. If there is a delay or interval between the acquisitions of images on different cameras, it is assumed that no object movement occurs between the intervals. If the image acquisition technique insures that both images are taken at the same time, then the assumption is satisfied even if the object is moving or for cameras that are mounted on a mobile robot [2]. The image quality which depends on the imaging device used also affects the quality of the outcome; this however is based on the design and the specification of the project and the needs that are to be met.

1.2.5 Previous Methods

The stereo vision problem has been approached in different ways. One approach is the use of correlation algorithms in which the points in images are directly matched by looking for correlation peaks between regions created about a point. Moravec, similar to Gennery and Tsai used correlation based algorithms to solve the corresponding problems. The authors of these algorithms have used area-based correlations for pixel matching [18]. Although most algorithms have tried to use raw irradiance values, some have defined a set of interesting points [14]. These points are chosen by methods of edge detection. Such methods usually use either a cross-correlation or mean-square differences techniques to measure the similarity between nearby regions of interesting points. Prominent works in this approach include the algorithms of Moravec, Nishihara [18] Baker and Binford [14]. Some algorithms have segmented the monocular images before performing a correlation based on properties of the segments. The work of Price and Reddy adopt this stereo vision approach.

Marr and Poggio were perhaps the pioneers in using the relaxation algorithms for stereo images. Seeking to model a neural process, Marr and Poggio developed the relaxation algorithm which associated a weight to each match and described an iterative algorithm. At each iteration and for each match, an *inhibitory* process reduced the weight of conflicting matches that conflicted with the uniqueness theorem explained earlier in this chapter. There also is an *excitatory* process which increases the weight of matches with similar disparities based on the continuity constraint [18].

According to Okutomi and Kanade [7], a short baseline of a pair of cameras in a stereo vision system, results in less precise distance estimation due to narrow

triangulation. It however contains less ambiguity which results in more accurate corresponding points. In longer baseline models, the distance estimation is more precise, however as the baseline becomes longer, there is less corresponding points present from one image to another. Therefore, there is a tradeoff between precision and accuracy in matching. In stereo vision the relationship between disparity d and the depth distance z can be represented by the following equation:

$$d = b \times \frac{f}{z} \tag{1.18}$$

where b is the baseline and f is the focal length. It can be seen from the above equation that for the same depth distance, the disparity is proportional to the baseline. This equation elaborates the existence of the precision-accuracy trade off problem. It is therefore suggested that with the knowledge of this dilemma, a decent model should have both acceptable precision and accuracy and offer a solution to the trade off problem.

1.2.6 Proposed Method

The proposed method in this thesis, offers a solution to the trade off problem between accuracy and precision. The proposed solution is to employ an array of five cameras rather than the traditional pair of cameras. This allows the baseline of each adjacent camera to be fairly small, while the entire baseline of the array is an acceptably large equivalent baseline. The pixel matching process is performed on two cameras at a time, and the search for the correspondences is propagated down the array [1]. Pixels that are found in all the cameras receive a high ranking while those with correlation in only a pair receive a low rank. The triangulation method is then used with the high ranked correspondence points to compute a sparse depth map of the surface. The density of the

3-D map can be increased by examining the behavior of the correspondence gradients from one camera to another. It is assumed that corresponding points on images which represent a location on the object behave smoothly from a camera pair to another, this is assumption is valid due to the gradual change of shape in biological surfaces. With this assumption, the low rank correspondences can be promoted to high ranks once they show high correlation among some camera pairs. Similarly, the candidates which receive high rank correlations among one pair or more but have low ranks amongst the rest of the pairs, are viewed as errors and are not included in the depth map. The obtained 3-D data can be used with Bezier or B-spline surface methods to fit a smooth surface.

CHAPTER 2

MATERIALS, INSTRUMENTS AND METHODS

2.1 Apparatus

As mentioned in the previous chapter, this experiment employs an array of cameras. Five high quality Pixelink cameras (Vitana Corp, Ottawa, Canada) were securely mounted on a stable table. All five cameras were placed at the same height in an array and were connected to a computer via a firewire hub. Each of the cameras is capable of capturing 1280 x 1024 pixels images at 9.3 fps (frames per second). The computer can view streaming data of each camera and take snapshots one camera at a time. To identify each camera they were labeled A to E from left to right. An object (a Halloween mask) was chosen and positioned in front of the cameras on a table; the distance was chosen in such a way that all the cameras can view the object. The equipment setup is shown in Figure 2.1.

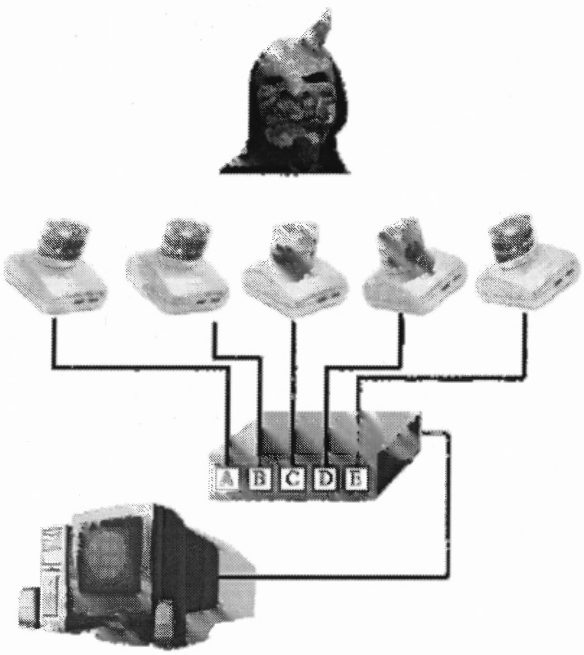

Figure 2.1 Equipment Setup.

To have a mathematical idea of where each camera is in space, all cameras were calibrated using the Camera Calibration toolbox for Matlab [8]. This procedure is described in Section 2.2. After calibrating each camera individually, each adjacent pair was stereo calibrated. Once these steps were performed, images could be taken using the Pixelink processing software.

All the algorithms and programs were written using Matlab 6.5 (Mathworks, MA, USA) and the results were analyzed using the same software. Images of each pair of cameras need to be rectified and warped (described in Section 2.3). Following the rectification and warping of pairs of images in each adjacent pair of cameras, the search for correspondences, namely the image matching, can be performed (explained in section 2.4). Section 2.5 describes the calculation of triangulation which is performed to obtain the depth information.

2.2 Camera Calibration

As cited by Oliver Faugeras, "calibration is the process of estimating the intrinsic and extrinsic parameters of a camera" [10]. Camera calibration is an important part of stereo image processing and when the purpose of the application is to "compute absolute 3-D information about a scene, then camera calibration data must be available" [17]. There are however other applications in stereovision that do not require calibration and rely on other methods. However, since the results of this section are essential for other parts of this project such as the rectification process, camera calibration was performed carefully.

After considering different methods of calibration, it was decided to use the Camera Calibration Toolbox for Matlab designed by Professor Jean-Yves Bouguet of the

California Institute of Technology. This is a simple and user-friendly toolbox that is easy to use and yet produces good results. The toolbox works with Matlab 5.x and Matlab 6.x (up to Matlab 6.5) on Windows, UNIX and Linux systems. In addition it does not require any specific Matlab toolbox [8]. During later stages of the experiment it was noted that this toolbox was also compatible with Matlab 7.0 which was therefore used for this experiment. The next section discusses the methods used to calibrate the cameras.

2.2.1 Camera Calibration Toolbox for Matlab

The first step of camera calibration is to provide a planar checkerboard. This calibration grid board is shown in Figure 2.2. The images are taken with the cameras, with the calibration grid positioned at varying orientation. This can be used to locate the position of the grids in space relative to the position of the camera once the dimensions of the grids are known. The toolbox recommends taking a total number of about 20 calibration images with each camera. However it was noted that taking more pictures resulted in more accuracy, and also with more pictures at hand, there was an option of eliminating the images with high errors.

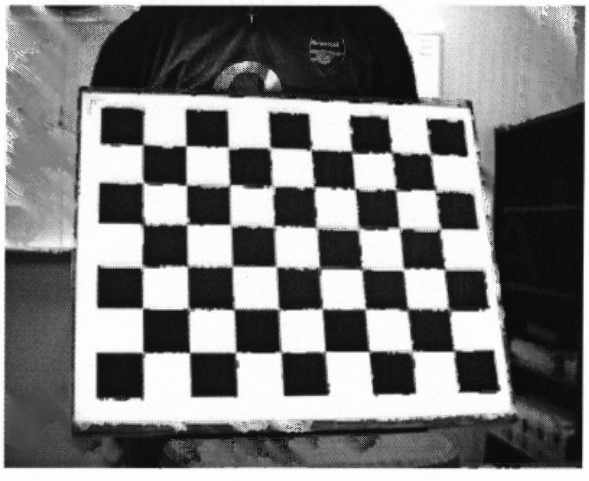

Figure 2.2 Grid used for calibration.

The Camera Calibration Toolbox can be loaded by recalling **calib.m** or by simply typing "calib" in the command line. Matlab then uploads a graphical interface shown in Figure 2.3 which can be used to start up the calibration process.

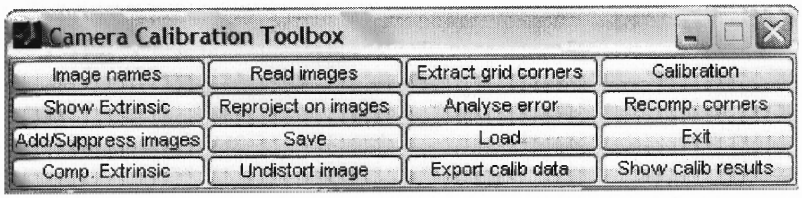

Figure 2.3 Calibration graphical interface.
(Source: Intel Corp. 2004 [8])

During the image acquisition process, the images were named similarly but with different suffix. For example, for camera A the images were named A0001, A0002 etc. The toolbox is capable of loading images that are named this way at once and the user does not need to input each picture individually. After loading all the images to the memory, they are shown in a thumbnail format illustrated in Figure 2.4

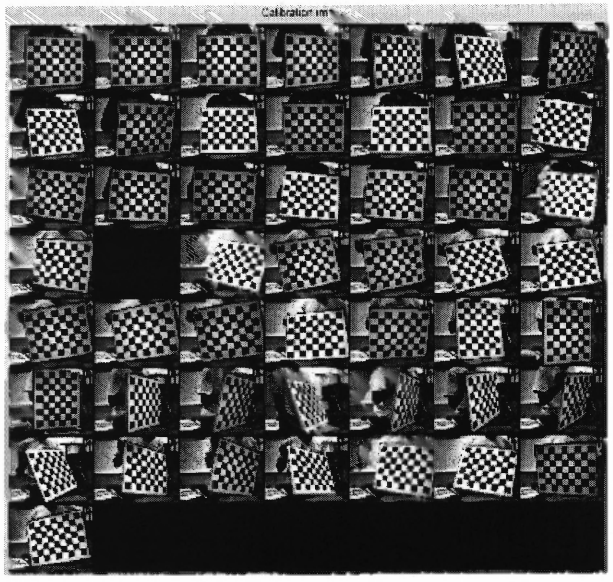

Figure 2.4 Camera calibration images thumbnail.

Once these steps are performed, the four corners of each checkerboard are manually selected. During this process, the origin is always selected first. Selections follows a clockwise direction, this is used to identify the x and y direction of the grids. The toolbox asks the user to input the number of the grids in each direction, and also the size of each grid in millimeters. With the information mentioned above, the toolbox automatically selects the corner position of the rest of the grids and the user can view the results and decide if they are acceptable. A sample of an automatic corner selection is shown in Figure 2.5

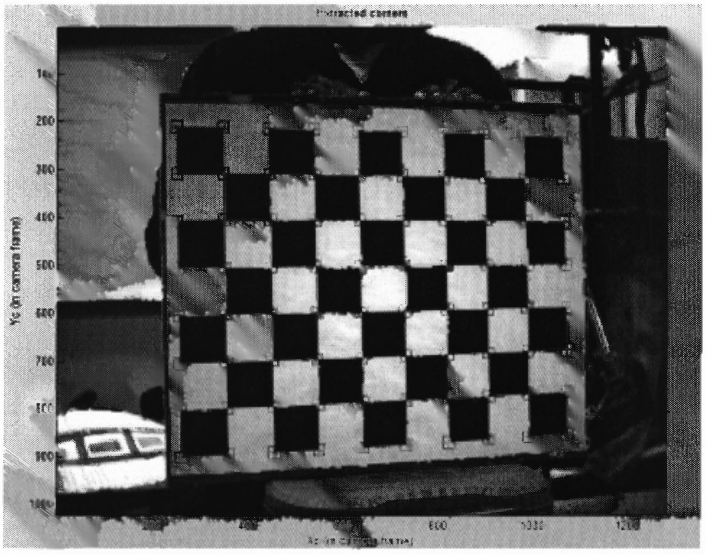

Figure 2.5 Automated corner selection.

Next, the calibration program is executed. This uses the extracted corners to compute the intrinsic and extrinsic parameters. The result of the calibration is then outputted by the program and can be saved for future use. Table 2.1 shows the calibration results for camera A.

Table 2.1 Calibration Results for Camera A

```
Calibration results (with uncertainties):
Focal Length:          fc = [ 2199.64748   2196.64726 ] ± [ 4.05987   4.01119 ]
Principal point:       cc = [ 649.13508    544.46685 ] ± [ 3.15585   2.62383 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
angle of pixel axes       = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.29005   0.62312   0.00078   -0.00022  0.00000 ]
                          ± [ 0.00804   0.09047   0.00022   0.00024   0.00000 ]
Pixel error:          err = [ 0.23285   0.22757 ]
```

The calibration results include intrinsic and extrinsic parameters such as focal length, skew factor, distortion, principal point, and rotation and translation matrices. Figure 2.6 illustrates the extrinsic parameters of a camera, with the calibration grids and the camera reference frame. This graph shows the 3-dimensional position of the checkerboard with respect to the camera.

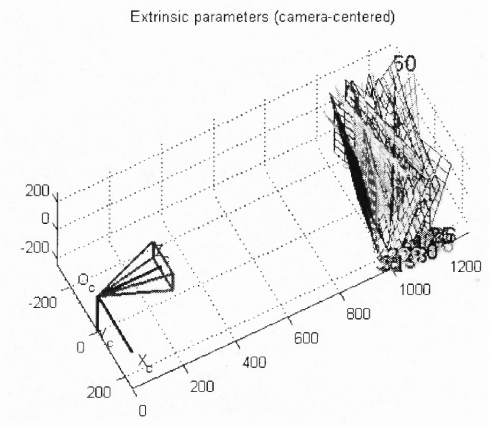

Figure 2.6 Extrinsic parameters and calibration grids.

For the calibration results, close attention should be paid to pixel error; the goal of a good calibration process is to minimize this error as much as possible. The toolbox lets the user to see which image has the highest error by choosing the "Analyze error " function. This function shows a graph similar to Figure 2.7 and clicking on each point on the graph indicates which image it is from. These images can either be analyzed and

reprocessed or be eliminated. The toolbox is equipped with other functions that can reduce the pixel error; they are not however discussed here.

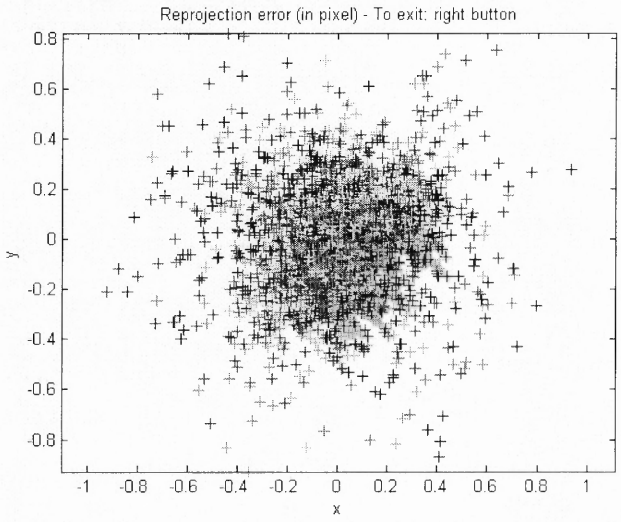

Figure 2.7 Reprojection error for calibration images.

2.2.2 Stereo Camera Calibration

To achieve stereovision, cameras need to be stereo calibrated. This is also done by using the Calibration Toolbox for Matlab. The m-file to execute stereo calibration is called **stereo_gui.m**, and once executed a user-friendly graphical interface is uploaded which is shown in Figure 2.8.

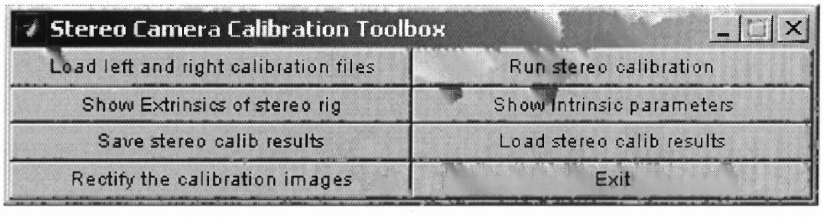

Figure 2.8 Stereo calibration graphical interface.
(Source: Intel Corp. 2004 [8])

The calibration results of the pair of cameras which were obtained from the earlier stages can be loaded and combined together. The toolbox calculates stereo calibration parameters and saves them into a new file called **Calib_results_stereo.mat**; this file is

used for the triangulation process. The stereo calibration parameters are shown in Table 2.2.

Table 2.2 Stereo camera calibration results for pair "AB"

```
Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 2198.89583   2196.03184 ] ± [ 5.04913   5.01171 ]
Principal point:       cc_left = [ 650.60136   540.90820 ] ± [ 5.23462   4.25964 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]
                               => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.29035   0.63173   0.00084   -0.00031  0.00000 ]
                               ± [ 0.01366   0.15443   0.00036   0.00038  0.00000 ]


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 2203.44695   2199.89655 ] ± [ 4.82463   4.80471 ]
Principal point:       cc_right = [ 641.63562   534.05750 ] ± [ 4.87356   3.96405 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes
                               = 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.27666   0.46214   0.00073   0.00214  0.00000 ]
                               ± [ 0.01490   0.17115   0.00034   0.00035  0.00000 ]


Extrinsic parameters (position of camera B wrt camera A):

Rotation vector:             om = [ 0.00379   0.12745  0.02668 ]
                                ± [ 0.00253   0.00312  0.00021 ]
Translation vector:           T = [ -179.91960   -0.49756  -34.45164 ]
                                ± [ 0.44492   0.36009  2.28357 ]
```

The spatial configuration and the calibration planes of the two cameras may be displayed in a form of a 3D plot by clicking on the button "Show Extrinsic of the stereo rig" in the stereo toolbox. This shows the position of the two cameras with respect to each other and the grids as shown in Figure 2.9.

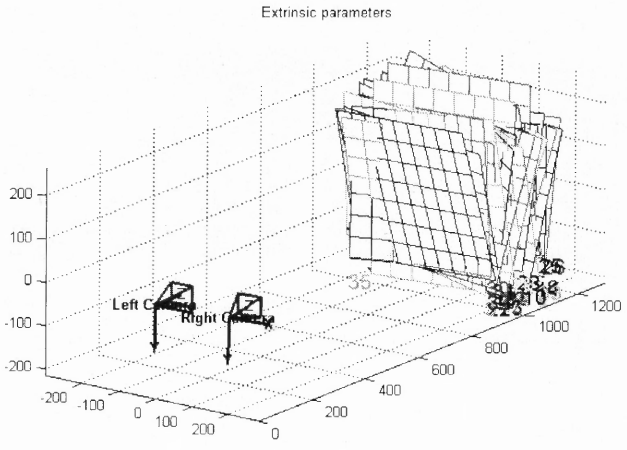

Figure 2.9 Extrinsic Parameters and spatial grid configuration of camera pair.

2.3 The Search for Correspondences

The main goal of stereovision in this experiment is the reconstruction and mapping of the 3-D object. To reach this goal, a problem arises which is known as the *correspondence problem.* This problem states that for a given pixel in image A, what is the corresponding pixel in image B. This problem can be approached in different ways. One popular method is to rectify all the images in order to produce epipolar lines. This method is chosen for this project and is discussed in the next section.

2.3.1 Rectification and Epipolar Lines

Rectification is defined as an operation to "insure a simple epipolar geometry for a stereo pair" [10]. In many stereo algorithm analyses, it is assumed that epipolar lines are parallel to the image rows. If this is valid, it can "enable the search for correspondent image features to be confined to one dimension" [17], and therefore simplify the process, making the algorithm execute faster and give it more accurate results.

The *perspective projection matrix* of a each cameras in a stereo vision system can be obtained by a calibration process [10],[15] . The following equation shows this matrix

$$P = A[R|t] \tag{2.1}$$

where A is the intrinsic matrix of a camera described in Section 1.2, and R represents the rotation matrix and t the translation vector. An algorithm available from the Camera Calibration Toolbox, along with a technique proposed by A. Fusiello [12], is used to rectify images. The rectification algorithms use the P matrices of the two cameras to calculate transformation matrices, which can transform images to rectified images. The

next equation shows the mathematical representation of the process of obtaining the coordinates of a point in the rectified image.

$$m_n = T \times m_0 \tag{2.2}$$

where m_0 represents the coordinates of the pixel in the original image, while T is the transformation matrix and m_n corresponds to the position of the point in the rectified image.

Once the transformation matrices are obtained, the rectification of each image is performed using **rectify.m**, which is listed in Appendix A. The rectified images are stored as new images to be used for the matching step. The rectification process is performed carefully since achieving epipolar lines rely on them. The accuracy of this process is directly related to the accuracy of the calibration parameters obtained in the earlier sections.

The results of rectification and epipolar lines can be manually tested by executing **test_epipolarlines.m**, which is included in Appendix A. One can simply pick a point in image A with known coordinates, and the program finds the corresponding pixel in image B and outputs the coordinates of that pixel. After execution of this file, it can be observed that the two pixels are on the same row as the two rectified images, which is what the epipolar lines state. This test is done enough times to insure epipolar lines exist throughout the experiment.

Figure 2.10 Rectified images from camera B and C.

2.3.2 Image Matching

After the image rectification process, the next step is to match pixels from one image to another in order to find the disparity. There are different techniques to do this as mentioned in Chapter 1. Since the rectification process has already been carried out, this search is only performed horizontally along the epipolar line in the second image. This greatly reduces the time taken by the process.

Different image matching techniques produce different results. However, the purpose of this experiment however is not to introduce a new image matching technique. Therefore, two common algorithms for matching were chosen, improved and adjusted according to the purpose of this project. The results of each algorithm were then analyzed and the superior algorithm was picked. The two matching techniques were SSD and Shirai algorithms, and are discussed in the next two sections. The image matching process is an automated algorithm. This process is performed for all the pixels that are to be mapped.

2.3.2.1 SSD Algorithm. One of the image matching techniques that were chosen was the Sum of Square Difference or SSD. In this algorithm, a window with size of mxm is

chosen around the pixel. The algorithm simply finds a same size window in the second image that closely resembles the one in the first image. The comparison of the blocks is done using the following equation:

$$SSD = \sum_{x,y} [I_T(x,y) - I(x,y)]^2 \tag{2.3}$$

where *I(x,y)* is from the first image, and I_T(x,y) is from the second image. The block from the second window that has the smallest value of *SSD* as compared with others, is chosen as the window that closely resembles the block from the first image. The center of this block is then chosen as the corresponding point of the pixel from the first image.

2.3.2.2 Shirai Algorithm. The second algorithm used is this project is based on block-matching algorithm first proposed by Shirai [2],[21] shown in Figure 2.11. This algorithm is written in an m-file in Matlab called **shirai.m** and is provided in Appendix A. The algorithm is modified in different ways for it to be more useful for this project; these modifications include the adjustment of the threshold included in the algorithm and a manual selection of the desired region where the object is in. For the purposes of this experiment, the object that is to be mapped is known. Therefore to simplify the search, the user is given the option of choosing the desired region of the search in the second image. This shortens the horizontal search, therefore reducing the errors and also the time of the execution. This however is optional and can be skipped.

```
for (every pixel (p, E_left(p)) of the left image E_left) do
    if ((p, E_left(p)) is an edge pixel) then begin
        initialize the window parameter k and the search interval I;
        loop
        set the window size n = 2k + 1 and the new search interval I;
        calculate the similarity measure SIMILARITY(p, q)
            for fixed pixel (p, E_left(p)) in the left image and
            for every pixel (q, E_right(q)) of the search interval;
                                                    { profile analysis }
        if (there is a unique minimum smaller than d1) then begin
            set the disparity value for point p in the disparity map;
            exit the loop
            end {then}
        else if (all similarity values are larger than d2) then begin
            a disparity assignment is not possible;
            exit the loop
            end {then}
        else if (the window already has maximum size) then begin
            a disparity assignment is not possible;
            exit the loop
            end {then}
                                    { preparation of a next search run }
        else begin
            reduce the interval size using d3;
            k:= k + 1                   { i.e. increase the window size }
            end {else}
        end {loop}
    end {if}
```

Figure 2.11 Shirai Algorithm
(Source: Reinhard Klette et. al. 1998 [2])

The Shirai algorithm is similar to the SSD algorithm, as it compares blocks from both images. It however selects matchable candidates from the first image based on edge detection techniques explained in the next section. The block comparison is performed using a function called *SIMILARITY* [2], which can be calculated using the following equations:

First, block p is selected in the first image, and function $SE(p,q)$ is calculated between p and q, where q is a block from the second image,

$$SE(p,q) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} (E_{left}(x_{left} + i, y + j) - E_{right}(x_{right} + i, y + j))^2 \qquad (2.4)$$

Then a function VARIANCE is calculated for block p,

$$VARIANCE(p) = \frac{1}{(2k+1).(2k+1)} \sum_{i=-k}^{k} \sum_{j=-k}^{k} \left[E_{left}(x_{left}+i, y+j) - AVERAGE(p) \right]^2 \quad (2.5)$$

$$VARIANCE(p) = \frac{1}{(2k+1).(2k+1)} \sum_{i=-k}^{k} \sum_{j=-k}^{k} E_{left}(x_{left}+i, y+j)^2 - AVERAGE(p)^2 \quad (2.6)$$

where $AVERAGE(p)$ denotes the arithmetic mean of the block. The function SIMILARITY can then be expressed the following way:

$$SIMILARITY(p,q) = \frac{SE(p,q)}{VARIANCE(p)+1} \quad (2.7)$$

In the Shirai algorithm there are three thresholds. The first threshold, d_1, is the threshold related to the *SIMILARITY* function, the values of this functions greater than d_1 are ignored and their corresponding pixels are not included in the results. If the value of *SIMILARITY(p,q)* is greater than d_1, then the program decides to use d_2, another pre specified threshold greater than d_1. If the values are smaller than d_2 the search interval is examined for a possible reduction where all points q are excluded for future comparisons for which they hold $SIMILARITY(p,q) \geq d_3$, *and* d_3 is an a-priori specified threshold greater than d_2. To simplify this interval reduction, the interval is only reduced at both ends [2].

2.3.3 Edge Detection

It is known that in a stereo process, there are two major problems. First is the extraction of matchable descriptions of an image which can be approached by edge detection methods. Second is the determination of the corresponding descriptors from each

processed image. The edge detection techniques consist of convolving each image with a set of filters of the form $\nabla^2 G$, where ∇^2 is the Laplacian and G is a Gaussian. For each size filter, the zero crossing of the convolved image is localized. Matching takes place between zero-crossing segments of the same sign and roughly the same orientation in the two images. In other words, matching takes place for points from a filtered image and the correspondences are used to calculate depth [14],[18].

Edge detection techniques can be used to produce better correspondence matching results. The edge detection scheme that was used on this thesis is called the Canny Edge detector, which was developed by John Canny of MIT in 1983 [10]. John Canny looked at the edge detection as a signal processing problem and used this to design an optimal edge detector. This thesis will not go in depth to explain this method or the Canny edge detection operator since it is a standard m-file of Matlab Image Processing toolbox. The function can be called in Matlab by the following command,

BW = edge (I, 'canny', thresh)

Where "I" is the image, BW is the image with detected edges and "thresh" is a two-element vector in which the first element is the low threshold, and the second element is the high threshold. The Canny method detects edges by looking for local maxima of the gradient of the image, which is calculated using the derivative of a Gaussian filter. The canny method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is less likely than the others to be fooled by noise, and is more likely to detect true weak edges [20].

Figure 2.12 illustrates an image taken by camera A, then processed using the Canny edge detection. The edge detection method is used to select points on the left image of the camera pair that are to be processed. In other words, the edges of the left image of the pair is detected, the resulting picture consists pixels having values of 1 and 0, which correspond to white and black points on Figure 2.12. The correspondence algorithm takes the pixels that have the values of 1 and searches through the rectified second image to find the corresponding pixels. This selection is done in order to minimize the time of the process and is also a good approach to choose high-quality candidates for image matching.

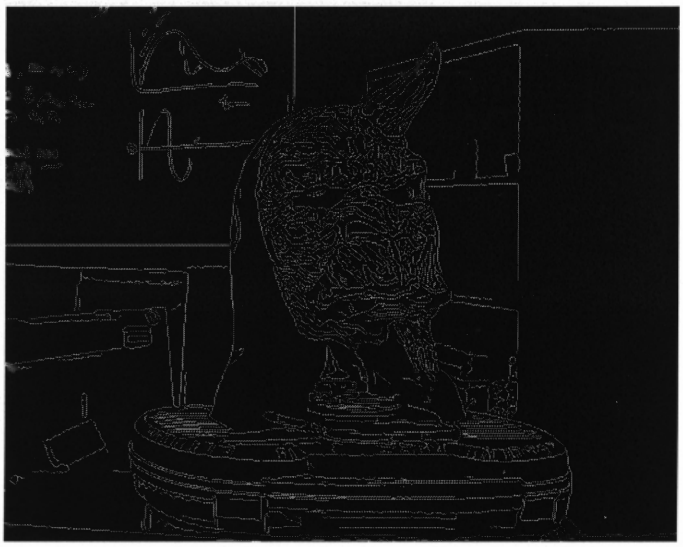

Figure 2.12 Edge detected image taken by camera A.

2.4 Triangulation

The theory of the triangulation algorithm was explained in Chapter 1. As mentioned before, triangulation uses the disparity map with the results of stereo calibration to calculate and obtain the depth map. The triangulation algorithm is included as an m-file in the Camera Calibration toolbox for Matlab. This program, along with the calibration

results and the disparity map obtained from the image matching step, are included in the main program written in Matlab to compute the 3-D depth map. The program is called **SSD.m** and is listed in Appendix A.

2.5 Experimental Procedure

As explained in Chapter one and two, the array of five cameras was mounted along the same x direction line, on a stable table. The cameras were named "A", "B", "C" , "D" and "E" from left to right. The position of the cameras can not be changed after the camera calibration step; therefore each camera was securely tightened in its position to eliminate any displacement. It was also noted that the position of the cameras were slightly changed during the course of time. This was due to various objects, often heavy ones, placed on the table by other students working at the lab. To deal with this problem and minimize pixel errors, the object images were taken immediately after the calibration images. By doing this, the object and calibration images correspond to the same camera position, and the possible movement of cameras over a long period due to the shaking would not have any effects on the overall result of the experiment.

During the rectification process, the coordinates of every epipolar line were saved in a list. This allows a simple inverse transformation through the constructed look-up table. The distance to the epipole can be computed and subtracted from the distance for the first pixel of the image row. Also the image values can easily be interpolated for higher accuracy.

As mentioned in Chapter one, the pixel correspondences are propagated along the camera array. This propagation process is started with the leftmost camera pair, namely

"AB". By using the results of the camera calibration, as shown in Figure 2.2 the pixels correspondences on rectified image pairs of cameras "A" and "B" is obtained. These set of pixels on the rectified image of camera B are identified on the original or the un-rectified image B by the method of inverse transformation mentioned earlier. Once the set of pixels are identified in the original image from "B", they are propagated on the camera pair "B-C". This is shown in Figure 2.13. The same process is also performed on the rest of the pairs down the array, and the results are saved. These results can be compared with the results of a larger baseline pair, namely "A-C" or "A-D.

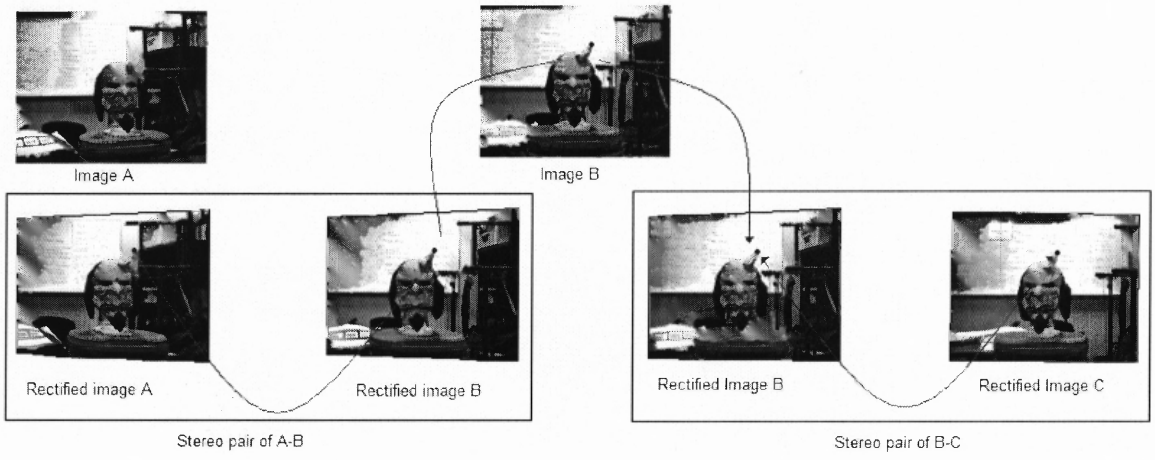

Figure 2.13 Correspondence propagation.

CHAPTER 3

RESULTS AND DISCUSSION

Once the theoretical studies and experimental procedures of the project are completed, the results need to be evaluated and processed. The following sections analyze and evaluate the results of each step of the project. During the project, due to technical difficulties, cameras "D" and "E" were eliminated from the process. The algorithm of the process however, remains the same and the proposed method can be proven to be efficient with the use of a three-camera array.

3.1 Results from calibration

The calibration results and parameters of the cameras are shown in Appendix B. The result of calibrating camera A is shown in Table 3.1; these results are obtained using the Camera Calibration Toolbox. By studying one camera, one can get a general idea about the rest since the procedure of calibration is identical for all cameras.

Table 3.1 Calibration results of Camera A

```
Calibration results of camera A (with uncertainties):
Focal Length:          fc = [ 2199.64748   2196.64726 ] ± [ 4.05987   4.01119 ]
Principal point:       cc = [ 649.13508   544.46685 ] ± [ 3.15585   2.62383 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
                            => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.29005   0.62312   0.00078   -0.00022  0.00000 ]
                          ± [ 0.00804   0.09047   0.00022    0.00024  0.00000 ]
Pixel error:          err = [ 0.23285   0.22757 ]
Note: The numerical errors are approximately three times the standard deviations (for reference).

Pixel error:          err = [ 0.23285   0.22757] (all active images)
```

Close attention should be paid to pixel error, which can be improved by reducing the sources of error. The quality of the image acquisition device, namely the camera, and

also the way it is mounted and used, can affect pixel error. The cameras should be checked to see if they are out of focus, which can also result in errors. In the corner detection step of the calibration, the four corners of the grid are picked manually. The other corners of each square on the grid are then estimated automatically. It can be observed that the initial manual picking of the corner can be a problematic task, since the corner of a grid is not one pixel. Figure 3.1 illustrates a magnified corner on an image and it can be seen that there is more than one candidate for the corner. This problem is due to the resolution of the picture, which is dependant on the quality of the camera. Therefore the results of calibration can be improved by employing a higher quality digital camera.

The grid paper that was used was a 500x375 mm printed on a special printer. During the printing process of a common printer, the paper is heated so the ink will dry on the paper and not come off easily [22]. It was noted during the project, that the shape of the paper can change slightly from this heating process. This becomes problematic when a large piece of paper is tried to be put flatly on a board. The flatness of the grid is essential in the calibration process, however the grid could not be completely flat due to the reasons stated. This affects the results and the outcome of the calibration by producing pixel errors.

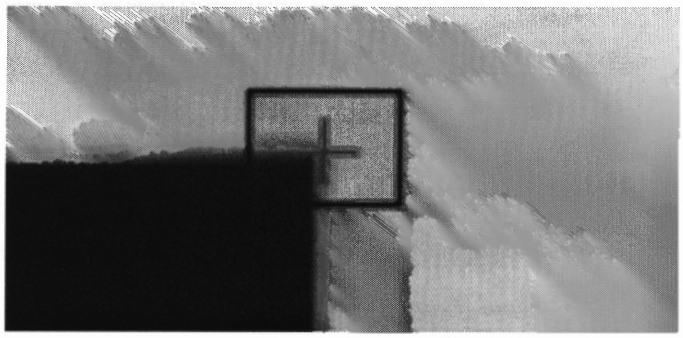

Figure 3.1 Camera A's edge detection.

The results of the calibration can be checked manually to see if they are realistic and sensible. The translation vector **Tc_1,** which is obtained from the calibration file, is the coordinate vector of the origin of the first grid pattern (**O**) in the camera reference frame. This is shown in Figure 3.2.

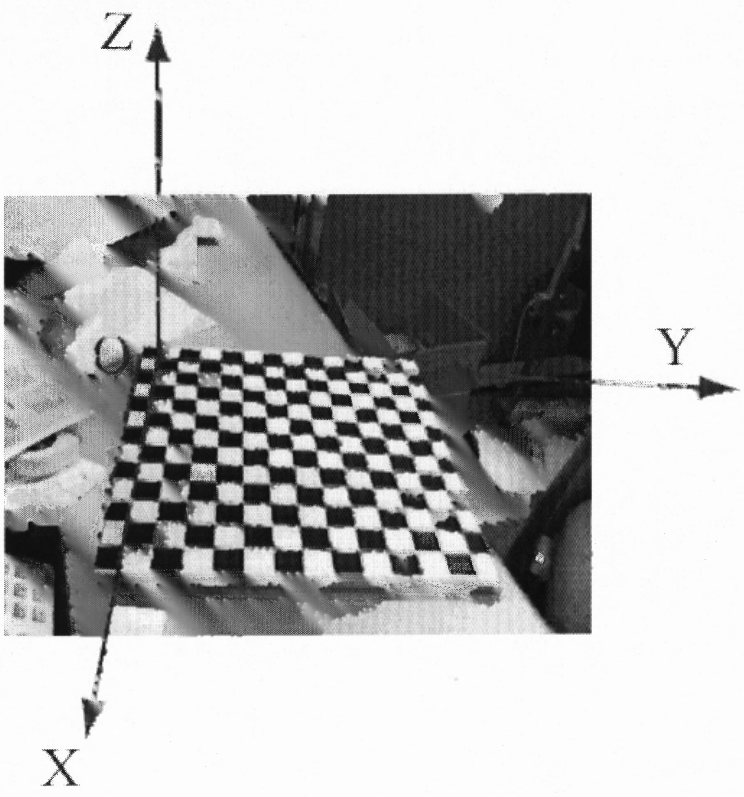

Figure 3.2 Grid Coordinates.

The translation vector of the first grid is the following:

Tc_1 = [2.368191e+002 ; 1.736355e+002 ; 1.217950e+003]

The third element is the distance between the camera reference frames and the origin of the grid, which is 1.217 meters, and it is a close match to manual measurements performed.

3.2 Rectification Results

The rectification step for each pair of cameras is performed after the stereo calibration and the rectified images of each camera pair is shown in Appendix B. Since the process of rectification is the same for all camera pairs, only the rectification results of cameras "A" and "B" will be discussed. The rectification procedure ensures the existence of epipolar lines which is essential in this experiment. The pair of rectified images is shown in Figure 3.3

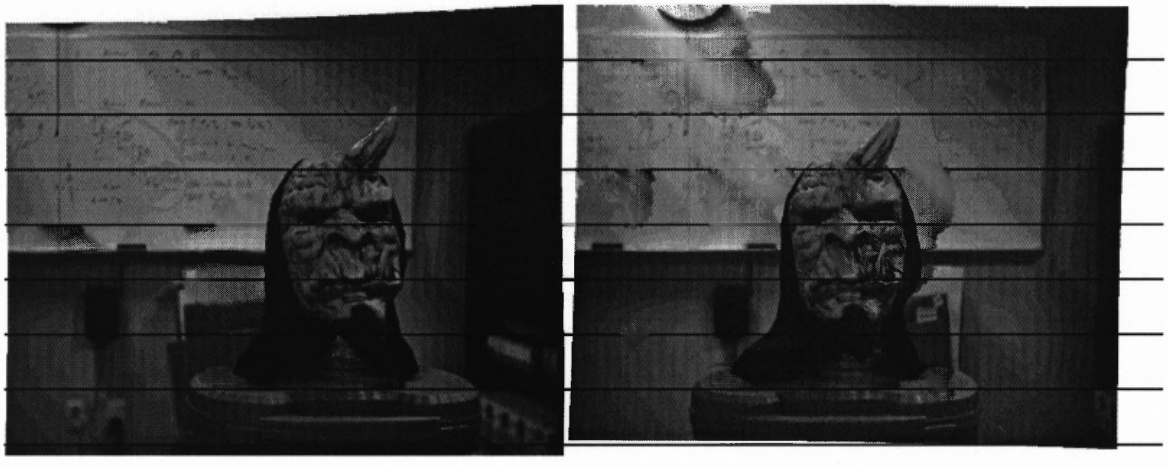

Figure 3.3 Rectified images of camera pair "AB".

The rectification process can also be checked manually. A point is picked from the first image, and manually found in the second image. It was observed that in almost all the cases, the horizontal positioning of the corresponding pixels matched in the rectified images. In some cases however, the rectified images were off by one pixel which is due to the pixel error produced while calibrating the cameras. This error is considered in the matching algorithm. The matching program was written in a way that it will search one pixel above and below the epipolar line as well as the epipolar line itself, and will pick the closest matching pixels for the corresponding point.

3.3 Pixel Matching Results

Writing a good matching algorithm was one of the more challenging parts of this project. This project uses currently available techniques and tries to propagate the resulting correspondences of two cameras along an array of cameras. There is need for a good correspondence algorithm in order to show that the proposed method works effectively. The steps of writing the algorithm were explained in Chapter 2. Due to the relatively high probability of producing an error in this part of the project, the results need to be effectively analyzed in order to reduce inaccuracies that may affect the final outcome. Two matching algorithms were considered and used during this project. The first is a SSD method which is an intensity matching algorithm, and the second is the Shirai algorithm which uses feature detection techniques to select pixels in the first image for the search for correspondences in the second image. The following sections discuss the results of these methods.

3.3.1 SSD Matching Algorithm

The SSD method is an intensity-based method and was used at the beginning of the project. Although this method produced some good data, it was noted that it also resulted in many errors. The SSD algorithm does not pick a selection of points in the first image, but searches for all corresponding points. As a result, the length of the process is prolonged. Also, the search for all the pixel correspondences is not ideal since not all the same pixels exist in both images [18]. This can lead to a large number of pixel correspondences that are not desired and are seen as errors. To minimize the high number of incorrect matchings, a threshold is chosen for the comparison of the SSD

blocks and is varied throughout the experiment to test and optimize the results. The m-files written in Matlab that perform the SSD process are included in Appendix A. The SSD algorithm was improved consistently to produce better results; however it was finally decided to employ a more accurate method for the search of correspondences, namely the feature-based method.

As mentioned in Chapter 2, the matching program is automated due to the length of the procedure. There is also a program named **testmatching.m** that lets the user manually enter a coordinate of a pixel in the first image, and find the corresponding point in the next image. The answer can be confirmed by manually locating the point in the second image and comparing the locations to the first. Although this can not be used in the automated part of the project, it helps making the algorithm more effective and reduces different sources of error in the main program. The manual input program, namely **testmatching.m,** was executed a number of times to make sure that it worked with a reasonably low percentage of error. Table 3.2 demonstrates some matching results of selected points on the first image shown in Figure 3.4. These points are selected and are manually identified in the second image. The coordinates of the identified pixels in the second image are then compared with the coordinates that the SSD program outputs. The SSD program produces a lot of errors and it can be seen on Table 3.2 that points "4" and "18" do not fall in the expected region.

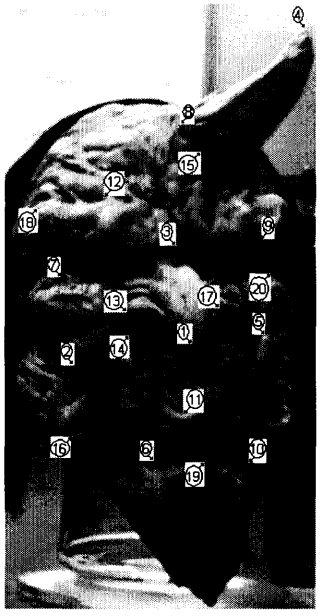

Figure 3.4 Selection of points on the mask.

Table 3.2 Search of correspondences using SSD

Points Tag	point coordinates		manually estimated position on the second image,		calculated coordinates using SSD	
	x	y	x	y	x	y
1	571	674	571	590-595	571	593
2	595	562	595	500-503	595	500
3	477	655	477	586-590	477	585
4	266	784	266	708-711	266	329
5	563	745	563	670-673	563	671
6	685	638	685	566-568	685	567
7	508	544	508	488-491	508	487
8	361	659	361	599-601	361	600
9	471	740	471	672-674	471	674
10	691	728	691	654-657	691	656
11	639	665	639	586-588	639	587
12	430	587	430	525-527	430	526
13	542	612	542	540-542	542	542
14	563	613	563	542-544	563	543
15	385	680	385	616-619	385	616
16	661	560	661	495-497	661	497
17	538	700	538	623-626	538	624
18	441	526	441	470-473	441	294
19	686	686	686	607-610	686	609
20	502	749	502	687-690	502	688

3.3.2 Shirai Method

The second method that was used was the Shirai method which was mentioned in Chapter 2. This method chooses a set of pixels based on feature based techniques such as edge detection. The Canny edge detection technique which was explained in Section 2.3 was used to select the desired pixels for process. The Shirai method is similar to the SSD method in the way that it uses the block comparison approach. However, it employs a function called the *SIMILARITY* which was mentioned in Section 2.3. This function proved to be an excellent factor in improving results and was therefore used for experimental evaluation of the proposed method.

Although not known for its time efficiency [2], the Shirai method along with the use of the constraints, reduced the time of execution drastically when compared with the SSD algorithm. As shown in Section 2.3, the Shirai method defines a threshold which is used on the numerical value of the *SIMILARITY* function. The blocks of pixels in the second image that have a value greater than the defined threshold are simply viewed as errors and are ignored. This reduces the time of the process as well as improving the outcome. In cases such as this, time efficiency is very important and can play a vital role in some applications and designs.

The Shirai algorithm was implemented using an m-file. Prior to adopting this algorithm, numerous tests on random points were performed to evaluate the accuracy of the algorithm in action. To do this, **shiraiCorrespondingpixeltest.m** was written which is very similar to **testmatching.m** and serves as a manual testing program. This m-file is included in Appendix A. Similar to Table 3.2, Table 3.3 demonstrates some matching results of selected points on the first image shown in Figure 3.4. These points are

selected and are manually identified in the second image. The coordinates of the identified pixels in the second image are then compared with the coordinates that are obtained using Shirai matching method.

Table 3.3 Search of correspondences using Shirai algorithm

Points Tag	point coordinates		manually estimated position on the second image,		calculated coordinates, Shirai algorithm Threshold=∞		Shirai algorithm With Threshold=1000	
	x	y	x	y	X	y	x	y
1	571	674	570-573	590-595	571	593	571	593
2	595	562	593-595	500-503	595	501	595	501
3	477	655	476-479	586-590	477	585	477	585
4	266	784	265-268	708-711	266	710	266	710
5	563	745	562-565	670-673	563	671	563	671
6	685	638	685-668	566-568	685	567	685	567
7	508	544	507-509	488-491	508	487	508	487
8	361	659	360-362	599-601	361	600	361	600
9	471	740	470-472	672-674	471	673	471	673
10	691	728	689-692	654-657	691	656	691	656
11	639	665	638-640	586-588	639	587	639	587
12	430	587	429-432	525-527	430	526	430	526
13	542	612	541-543	540-542	542	541	542	541
14	563	613	560-563	542-544	563	543	563	543
15	385	680	384-386	616-619	385	616	385	616
16	661	560	660-663	495-497	661	497	661	497
17	538	700	537-539	623-626	538	624	538	624
18	441	526	439-442	470-473	441	523	441	Nan*
19	686	686	685-687	607-610	686	609	686	Nan*
20	502	749	502-504	687-690	502	687	502	687

*Nan=Not a number

As it can be seen on Table 3.3 all the resulting coordinates, except point 19, fall into the region of manual estimation. Some error was to be expected as the corresponding algorithm can not be designed to give perfect results; however most of the errors can be analyzed and removed from the selection. In the case of point 19, with the selection of correct threshold, this point can be ignored and viewed as error. As an example, it is shown that the threshold of 1000 can remove and assign the value of Nan

(Not a number in Matlab) to this point. It is concluded that the lower the threshold, the higher the possibility of correct correspondences.

3.3.3 Correspondence Propagation and Disparity Map

Following satisfactory tests on the Shirai algorithm, an m-file was written to employ this algorithm for the search of all matching correspondences. Functions of this m-file include loading a pair of pictures, edge detecting the first picture for selection of points, and the search of correspondences. The coordinates of the correspondences are then used to find the disparity map and ultimately the depth of the points. This m-file is named **Shiraiwiththreshold.m** and is included in Appendix A. Table 3.4 shows the number of corresponding points found from the first pair of cameras and then the results of propagating these points along the camera array. It can be seen that the number of correspondences decrease as the search proceeds down the array. This was expected due to the fact that some pixels simply do not exist as the process is moved along the cameras.

Table 3.4 The number of corresponding points propagated

	Number of Corresponding points on camera pairs	
Threshold	AB	BC
5	1341	1092
10	2731	2430
20	4940	4342
50	7319	6973
100	8838	8245
500	11572	10074
1000	12126	11163

The number of corresponding points that are found using the proposed method need to be compared to the old methods, namely the direct matching algorithms. A pair of cameras with a relatively large baseline is chosen; this pair is "AC". The Shirai algorithm was again used for the direct matching method for large-baseline models and the results are shown in Table 3.5 the results of the proposed method are also shown on this table for comparison. It can be noted that the number of correspondences found using the proposed method is larger than the direct matching method; this is shown with different values of the Shirai algorithm threshold. It can also be noted the high percentage of points propagated from the first pair in the second pair.

Table 3.5 Comparison of number of points to previous methods

	Proposed Method				% of correspondences propagated
Threshold	AB	BC	ABC	AC	
20	4940	4342	4342	1699	87.89474
50	7319	6973	6973	3679	95.27258
100	8838	8245	8245	5245	93.29034
500	11572	10074	10074	9765	87.05496
1000	12126	11163	11163	10943	92.05839

The disparity for every corresponding point in each rectified pair is obtained and the overall disparity of these points on the entire array is calculated. The overall disparity is simply the addition of all the disparities of each adjacent pair of cameras. The selection of points used earlier and shown in Figure 3.4 are used to compare the disparities obtained from the proposed method to the manual estimation of disparities from the direct matching method. Table 3.6 shows the expected disparities of the selected points in pair "AC". Since some of the points from camera "A" are difficult to find in camera "C" using a conventional two camera model, their corresponding positions are manually identified and estimated. This gives better accuracy when the results are

compared with the result of the proposed method. Table 3.7 shows the position of the points in the rectified "B" which are obtained by transformation of coordinates from the original image. These pixels are then found in camera C, and the disparities between pairs "AB" and "BC" are added to obtain the overall disparity. The results of the proposed method closely match the estimated numbers which proves the accuracy and precision of the method.

Table 3.6 Disparities in pair "AC" obtained manually

Point's Tag	A from pair "AC"		C from pair "AC"		Disparity
	x	y	x	y	
1	576	681	576	586	95
2	600	571	597	515	56
3	483	662	483	591	71
4	276	786	276	708	78
5	568	750	568	672	78
6	689	645	688	569	76
7	514	553	514	507	46
8	369	665	368	616	49
9	477	745	477	680	65
10	695	734	694	657	77
11	644	672	644	585	87
12	437	596	437	541	55
13	548	620	549	546	74
14	569	621	569	550	71
15	392	686	392	628	58
16	666	569	666	509	60
17	544	706	544	623	83
18	-	-	-	-	-
19	690	693	690	606	87
20	508	754	508	702	52

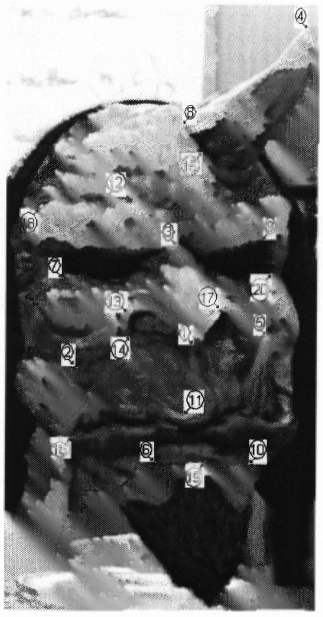

Figure 3.4 Selection of points on the mask.

Table 3.2 Search of correspondences using SSD

Points Tag	point coordinates		manually estimated position on the second image,		calculated coordinates using SSD	
	x	y	x	y	x	y
1	571	674	571	590-595	571	593
2	595	562	595	500-503	595	500
3	477	655	477	586-590	477	585
4	266	784	266	708-711	266	329
5	563	745	563	670-673	563	671
6	685	638	685	566-568	685	567
7	508	544	508	488-491	508	487
8	361	659	361	599-601	361	600
9	471	740	471	672-674	471	674
10	691	728	691	654-657	691	656
11	639	665	639	586-588	639	587
12	430	587	430	525-527	430	526
13	542	612	542	540-542	542	542
14	563	613	563	542-544	563	543
15	385	680	385	616-619	385	616
16	661	560	661	495-497	661	497
17	538	700	538	623-626	538	624
18	441	526	441	470-473	441	294
19	686	686	686	607-610	686	609
20	502	749	502	687-690	502	688

The left image on Figure 3.4 shows a depth representation of the corresponding points detected by the SSD algorithm. The scattered depth map of the face is first plotted using mesh and surface command, and then the image of the face is put over the surface. The outcome becomes a recognizable 3-D dimensional plot of the face. The plot gives a decent overall result but fails in accuracy with points on the face of the mask. It can be seen on the right image of Figure 3.4 that the depth resolution on the face is not as accurate as expected and the features of the face do not stand out.

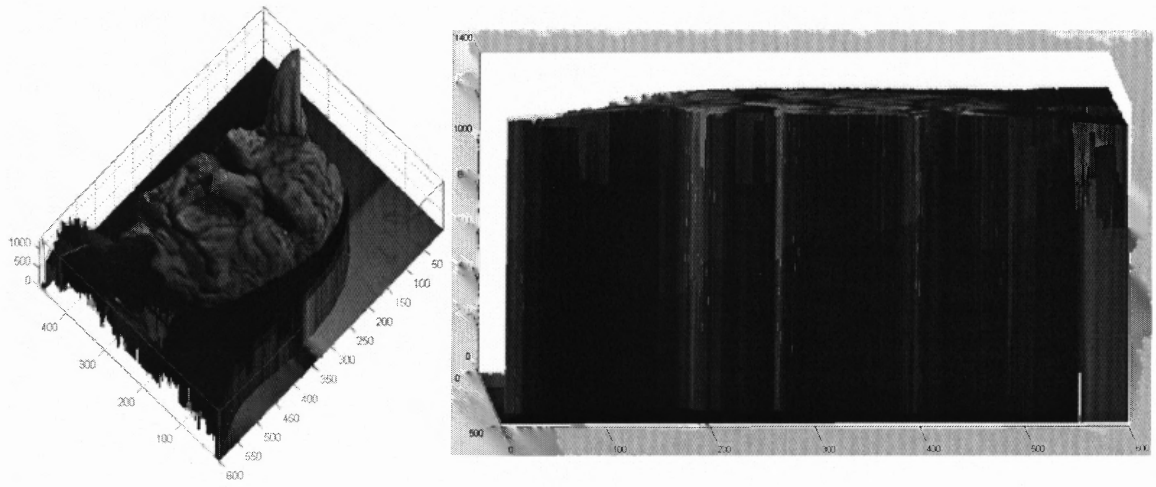

Figure 3.5 3-D representation of the mask in space using SSD algorithm.

The 3-D representation of the selected pixels using edge detection is shown in Figure 3.6. The actual grayscale image values corresponding to each point can also be used to enhance the plot, making it more recognizable. This plot is also shown in Figure 3.6.

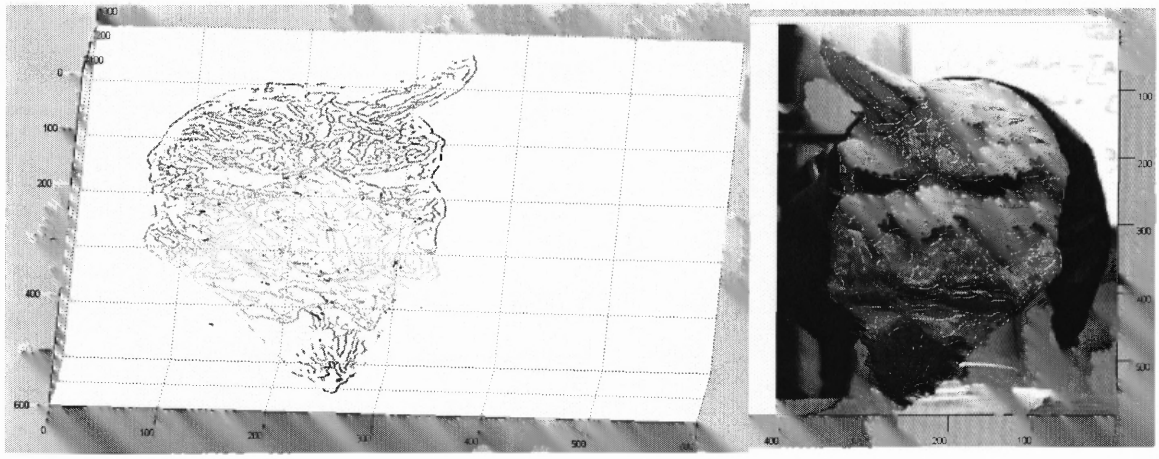

Figure 3.6 3-D representation of pair "AC" using the proposed method.

Figure 3.7 shows the depth representation of the mask from the side angle. The graph shows the good depth resolution of the proposed method and the features of the face can be clearly seen.

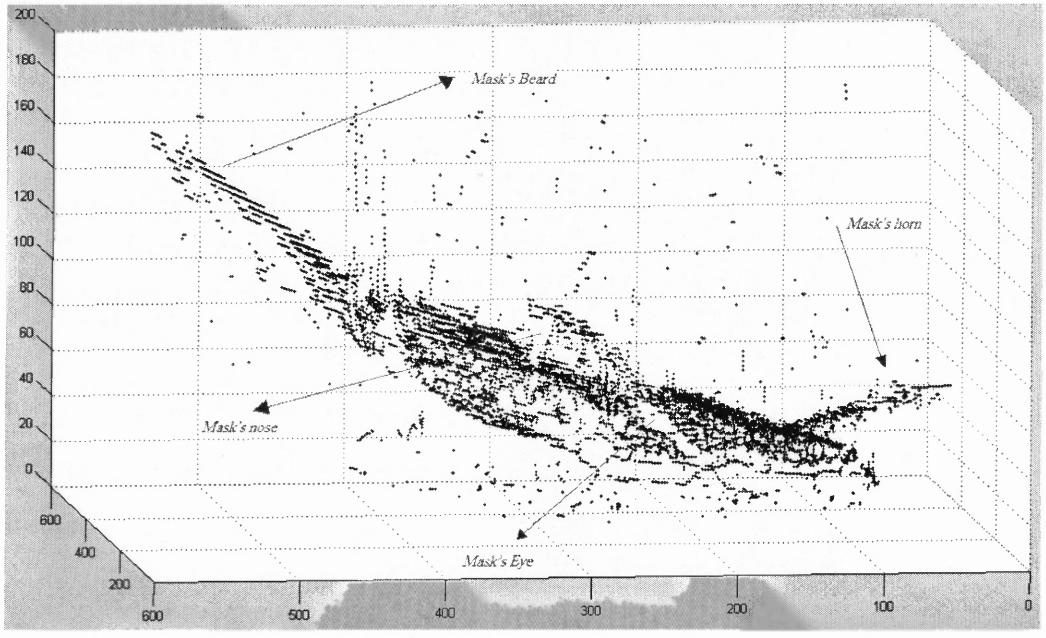

Figure 3.7 Depth representation of the face from side angle.

CHAPTER 4

CONCLUSION

In the engineering world today, there will always be need for improvement of existing methods. The method proposed in this thesis, is shown to be superior when compared to its predecessors. It is a promising technique that can be adopted for different applications ranging from robotics and navigation to biomedical engineering. The precision of the method was shown to be equal to that of large baseline. When performing corresponding pixel matching for the large baseline model, there were many errors produced due to the existence of ambiguities. This ultimately resulted in inaccurate depth measurements. The proposed method propagates correspondences along an array of cameras each having small baselines, therefore minimizing the error.

Due to the employment of a large baseline model, the proposed method has good distance resolution. At the same time, the advantages of a small baseline model such as accurate pixel and feature matching are enjoyed. In other words, the proposed method has high depth resolution while having high quality results of correspondence matching. When using this method, the accuracy-precision trade off issue is resolved.

The disadvantage of the proposed method is that, employing five cameras would significantly increase the cost of hardware, unless time constraints allow the same hardware to be used successively for the five images. Also the visual field common to five cameras is smaller than that observed by two cameras; hence the reconstructed region is smaller. This however does not cause any problem for the purpose of this thesis, which is to obtain the depth map of biological surfaces that are fixed in position while the images are taken.

As in any other engineering design, there is room for improvement. There is a need for a good algorithm that can confidently find corresponding points, while effectively reducing errors. This can help generate a more correct depth representation of biological surfaces. Other means of improvement include choosing superior image acquisition devices, better camera calibration precision, and a more accurate triangulation algorithm. The experimental results of the proposed method met the expectations and proved to be capable of combining small base and long base models, therefore enjoying both the advantages that the two models have to offer.

APPENDIX A

COMPUTER PROGRAMS

The programs that were used during this project are listed in this section. The following program takes the input of a point on the first image, and outputs its corresponding point in the second image based on a modified SSD algorithm.

Correspondingpixeltest.m

```
function Correspondingpixeltest

A1=imread('A_rectified0001.bmp');

B1=imread('B_rectified0001.bmp');

A=double(A1);

B=double(B1);

x=input('input x, x is vertical') % instead of "for x=214:1:813;" in the main program

y=input('input y, y is horizontal') %instead of " for y=514:1:983;" in the main program

MatA=[A(x-3,y-3),A(x-3,y-2),A(x-3,y-1),A(x-3,y),A(x-3,y+1),A(x-3,y+2),A(x-3,y+3);

A(x-2,y-3),A(x-2,y-2),A(x-2,y-1),A(x-2,y),A(x-2,y+1),A(x-2,y+2),A(x-2,y+3); A(x-1,y-

3),A(x-1,y-2),A(x-1,y-1),A(x-1,y),A(x-1,y+1),A(x-1,y+2),A(x-1,y+3); A(x,y-3),A(x,y-

2),A(x,y-1),A(x,y),A(x,y+1),A(x,y+2),A(x,y+3); A(x+1,y-3),A(x+1,y-2),A(x+1,y-

1),A(x+1,y),A(x+1,y+1),A(x+1,y+2),A(x+1,y+3); A(x+2,y-3),A(x+2,y-2),A(x+2,y-

1),A(x+2,y),A(x+2,y+1),A(x+2,y+2),A(x+2,y+3); A(x+3,y-3),A(x+3,y-2),A(x+3,y-

1),A(x+3,y),A(x+3,y+1),A(x+3,y+2),A(x+3,y+3)];

for Y=4:1:1277; % Y=1:1:number of columns, this is in the second image

MatB=[B(x-3,Y-3),B(x-3,Y-2),B(x-3,Y-1),B(x-3,Y),B(x-3,Y+1),B(x-3,Y+2),B(x-
3,Y+3); B(x-2,Y-3),B(x-2,Y-2),B(x-2,Y-1),B(x-2,Y),B(x-2,Y+1),B(x-2,Y+2),B(x-
```

```
2,Y+3); B(x-1,Y-3),B(x-1,Y-2),B(x-1,Y-1),B(x-1,Y),B(x-1,Y+1),B(x-1,Y+2),B(x-
1,Y+3); B(x,Y-3),B(x,Y-2),B(x,Y-1),B(x,Y),B(x,Y+1),B(x,Y+2),B(x,Y+3); B(x+1,Y-
3),B(x+1,Y-2),B(x+1,Y-1),B(x+1,Y),B(x+1,Y+1),B(x+1,Y+2),B(x+1,Y+3); B(x+2,Y-
3),B(x+2,Y-2),B(x+2,Y-1),B(x+2,Y),B(x+2,Y+1),B(x+2,Y+2),B(x+2,Y+3); B(x+3,Y-
3),B(x+3,Y-2),B(x+3,Y-1),B(x+3,Y),B(x+3,Y+1),B(x+3,Y+2),B(x+3,Y+3);];
   diff=abs((MatB)-(MatA));
   T1(1,Y)=sum(sum(diff));
end
size=size(T1);
t=[1:1277];
plot(t,T1);
k=min(T1(4:1277)); %position of the minimum value of the matrix N
[i]=find(T1==k)
```

The following program is a modified version of SSD with a defined threshold.

SSD.m

```
function SSD
load('Calib_Results_stereo');
A1=imread('A_rectified0001.bmp');
B1=imread('B_rectified0001.bmp');
A=double(A1(:,1:1280));
B=double(B1(:,1:1280));
edge=imread('edgedone.bmp');
```

```
for x=214:1:813; %this is the coordinates that the face is in picture 1 (x,y)
for y=514:1:983;
if edge(x-213,y-513)==0
MatA=[A(x-2,y-2),A(x-2,y-1),A(x-2,y),A(x-2,y+1),A(x-2,y+2);A(x-1,y-2),A(x-1,y-
1),A(x-1,y),A(x-1,y+1),A(x-1,y+2);A(x,y-2),A(x,y-
1),A(x,y),A(x,y+1),A(x,y+2);A(x+1,y-2),A(x+1,y-
1),A(x+1,y),A(x+1,y+1),A(x+1,y+2);A(x+2,y-2),A(x+2,y-
1),A(x+2,y),A(x+2,y+1),A(x+2,y+2)];
%C=A(x,y) %the value of the element at (x,y), A is image 1 ,1024 by 1280
%for X=x       %x-4:1:x+4 % X=1:1:number of rows
for Y=3:1:1277; % Y=1:1:number of columns, this is in the second image
MatB=[B(x-2,Y-2),B(x-2,Y-1),B(x-2,Y),B(x-2,Y+1),B(x-2,Y+2);B(x-1,Y-2),B(x-1,Y-
1),B(x-1,Y),B(x-1,Y+1),B(x-1,Y+2);B(x,Y-2),B(x,Y-
1),B(x,Y),B(x,Y+1),B(x,Y+2);B(x+1,Y-2),B(x+1,Y-
1),B(x+1,Y),B(x+1,Y+1),B(x+1,Y+2);B(x+2,Y-2),B(x+2,Y-
1),B(x+2,Y),B(x+2,Y+1),B(x+2,Y+2)];
   diff=abs((MatB)-(MatA));
   T1(Y,1)=sum(sum(diff));
   %diff=B(X,Y)-C;  %B is image 2
   %N(X,Y)=diff;   % matrix N is the difference between the value of every pixel
                % of B minus the value of the desired pixel in A
    end
```

```
size(T1);
t=[1:1277];
%plot(t,T);
k=min(T1(3:1277)); %position of the minimum value of the matrix N
[i]=find(T1==k);%[i]=find(T==n)
ypos=i;%y position of corresponding pixel of (x,y) in the second image
xL=[x;y];
xR=[x;ypos];
[XL,XR] =
stereo_triangulation(xL,xR,om,T,fc_left,cc_left,kc_left,alpha_c_left,fc_right,cc_right,kc_
right,alpha_c_right);
ZA(x-213,y-513)=XL(3,1); %the value of z(depth) at the pixel (x,y). the position starts
from (x-214,y-514) which is (1,1)
ZB(x-213,y-513)=XR(3,1);
else
end
end
end
```

This following m-file is the program written based on Shirai Algorithm without threshold.

Shirai.m

```
function Shirai
tic
```

```
%%%%this program is for AB
A1=imread('A_rectified0001.bmp');
B1=imread('B_rectified0001.bmp');
A=double(A1);
B=double(B1);
Aedge = edge(A,'canny'); %edge detection
load('Calib_Results_stereo');
imshow(B1);
[lefty1,leftx1] = ginput(1);
lefty=round(lefty1);
g=0; %this is the counter to position the pixels in Brectified in a stack
for x=255:255+578; %x=input('input x, x is horizontal in paint
   for y=435:435+404  %y=input('input y, y is vertical in paint
      if Aedge(x,y)==1 %if the edge is white in the edge detected picture
MatA=[A(x-3,y-3),A(x-3,y-2),A(x-3,y-1),A(x-3,y),A(x-3,y+1),A(x-3,y+2),A(x-3,y+3);
A(x-2,y-3),A(x-2,y-2),A(x-2,y-1),A(x-2,y),A(x-2,y+1),A(x-2,y+2),A(x-2,y+3);  A(x-1,y-
3),A(x-1,y-2),A(x-1,y-1),A(x-1,y),A(x-1,y+1),A(x-1,y+2),A(x-1,y+3);  A(x,y-3),A(x,y-
2),A(x,y-1),A(x,y),A(x,y+1),A(x,y+2),A(x,y+3);  A(x+1,y-3),A(x+1,y-2),A(x+1,y-
1),A(x+1,y),A(x+1,y+1),A(x+1,y+2),B(x+1,y+3);  A(x+2,y-3),A(x+2,y-2),A(x+2,y-
1),A(x+2,y),A(x+2,y+1),A(x+2,y+2),A(x+2,y+3);  A(x+3,y-3),A(x+3,y-2),A(x+3,y-
1),A(x+3,y),A(x+3,y+1),A(x+3,y+2),A(x+3,y+3)];
         Average=mean(mean(MatA));
         k=3;
```

```
Var1=0;
for i=-k:1:k
  for j=-k:1:k
    Var1=Var1+(A(x+i,y+j)-Average)^2;
  end
end
Variance=Var1/((2*k+1)^2);
 for Y=lefty:1:y% can not start from 1 because Y-3 would be outside the picture,
so starting from pixel 4
  n=0;% this is the counter to place create SEmat
  SE=0;
  for X1=x-1:1:x+1
   %calculation of SE(A,B):
    for i=-k:1:k
      for j=-k:1:k
        SE=SE+((A(x+i,y+j)-B(X1+i,Y+j))^(2));
      end
    end
    n=n+1;
    SIMILARITY(n,Y)= SE/(Variance+1);
  end
 end
end
```

```
        SEsize=size(SIMILARITY);
        v=var(SIMILARITY);
        vmin=min(min(v(:,lefty:end))); %minimum value of SEmat, starting from 4,
because the first 4 colomns are zero.
        [p,Ypos]=find(v==vmin);
        xR=[x;Ypos(1)];
        xL=[x;y];
        g=g+1;%the counter for the stack.
        BrecPos(g,1)=x; %positioning the pixels from Brectified in a stack form/
        BrecPos(g,2)=Ypos(1);
        [XL,XR] =
stereo_triangulation(xL,xR,om,T,fc_left,cc_left,kc_left,alpha_c_left,fc_right,cc_right,kc_
right,alpha_c_right);
        ZA(x-254,y-434)=XL(3,1); %the value of z(depth) at the pixel (x,y). the position
starts from (x-214,y-514) which is (1,1)
        ZB(x-254,y-434)=XR(3,1);
    end
end
save('DepthofA&B')
display('elapsed time is:')
toc
```

The following program is a modified version of Shirai algorithm written as an m-file, it defines a threshold vmin.

Shiraiwiththeshold.m

```
function Shiraiwiththreshhold
tic
clear
%%%%this program is for AB
A1=imread('A_rectified0001.bmp');
B1=imread('B_rectified0001.bmp');
A=double(A1);
B=double(B1);
Aedge = edge(A,'canny'); %edge detection
Aedgeprocessed=imread('edgeprocessed.bmp');
Bedgeprocessed=imread('Bedgeprocessed.bmp');
load('Calib_Results_stereo');
imshow(B1);
[lefty1,leftx1] = ginput(2); %gets input from the mouse
lefty=round(lefty1(1,1));
righty=round(lefty1(2,1));
g=0;
for x=255:255+578; %x=input('input x, x is horizontal in paint
   for y=435:435+578%404  %y=input('input y, y is vertical in paint
```

```
if Aedge(x,y)==1 & Aedgeprocessed(x,y)==1;%if the edge is white in the edge
detected picture
MatA=[A(x-3,y-3),A(x-3,y-2),A(x-3,y-1),A(x-3,y),A(x-3,y+1),A(x-3,y+2),A(x-
3,y+3); A(x-2,y-3),A(x-2,y-2),A(x-2,y-1),A(x-2,y),A(x-2,y+1),A(x-2,y+2),A(x-2,y+3);
A(x-1,y-3),A(x-1,y-2),A(x-1,y-1),A(x-1,y),A(x-1,y+1),A(x-1,y+2),A(x-1,y+3); A(x,y-
3),A(x,y-2),A(x,y-1),A(x,y),A(x,y+1),A(x,y+2),A(x,y+3); A(x+1,y-3),A(x+1,y-
2),A(x+1,y-1),A(x+1,y),A(x+1,y+1),A(x+1,y+2),B(x+1,y+3); A(x+2,y-3),A(x+2,y-
2),A(x+2,y-1),A(x+2,y),A(x+2,y+1),A(x+2,y+2),A(x+2,y+3); A(x+3,y-3),A(x+3,y-
2),A(x+3,y-1),A(x+3,y),A(x+3,y+1),A(x+3,y+2),A(x+3,y+3)];
Average=mean(mean(MatA));
k=3;
Var1=0;
for i=-k:1:k
  for j=-k:1:k
    Var1=Var1+(A(x+i,y+j)-Average)^2;
  end
end
Variance=Var1/((2*k+1)^2);
for Y=lefty:1:righty% can not start from 1 because Y-3 would be outside the
picture, so starting from pixel 4
  n=0;% this is the counter to place create SEmat
  SE=0;
  for X1=x-1:1:x+1
```

```
            %calculation of SE(A,B):
            for i=-k:1:k
              for j=-k:1:k
                SE=SE+((A(x+i,y+j)-B(X1+i,Y+j))^(2));
              end
            end
            n=n+1;
            SIMILARITY(n,Y)= SE/(Variance+1);
          end
        end
        SEsize=size(SIMILARITY);
        v=var(SIMILARITY);
        vmin=min(min(v(:,lefty:end))); %minimum value of SEmat, starting from 4,
because the first 4 colomns are zero.
        if vmin<=1000
          [p,Ypos]=find(v==vmin);
          xR=[x;Ypos(1)];
          xL=[x;y];
          if Bedgeprocessed(x,Ypos(1))==1
            g=g+1;%the counter for the stack.
            BrecPos(g,1)=x; %positioning the pixels from Brectified in a stack form/
            BrecPos(g,2)=Ypos(1);
```

```
[XL,XR] =
stereo_triangulation(xL,xR,om,T,fc_left,cc_left,kc_left,alpha_c_left,fc_right,cc_right,kc_
right,alpha_c_right);
                ZA(x-254,y-434)=XL(3,1); %the value of z(depth) at the pixel (x,y). the
position starts from (x-214,y-514) which is (1,1)
                ZB(x-254,y-434)=XR(3,1);
            else
                ZA(x-254,y-434)=nan;
                ZB(x-254,y-434)=nan;
            end
        else
            ZA(x-254,y-434)=nan;
            ZB(x-254,y-434)=nan;
        end
    else
        ZA(x-254,y-434)=nan;
        ZB(x-254,y-434)=nan;
    end
  end
end
%creating a scattered matrix for A
for x=1:1:578
for y=1:1:404
```

```
if ZA(x,y)>0
scatterA(x,y)=ZA(x,y);
else
scatterA(x,y)=nan;
end
end
end
%creating a scatterd matrix for B
for x=1:1:578
for y=1:1:404
if ZB(x,y)>0
scatterB(x,y)=ZB(x,y);
else
scatterB(x,y)=nan;
end
end
end
save('DepthofA&B, T=1000')
display('elapsed time is:')
toc
```

The following program takes the input of a point on the first image, and outputs its corresponding point in the second image based on Shirai algorithm.

Shiraitest.m

```
function Correspondingpixeltest
clear
A1=imread('A_rectified0001.bmp');
B1=imread('B_rectified0001.bmp');
A=double(A1);
B=double(B1);
x=input('input x, x is vertical') % instead of "for x=214:1:813;" in the main program
y=input('input y, y is horizontal') %instead of " for y=514:1:983;" in the main program
for counter=1:20;
clear vmin, clear SIMILARITY, clear Ypos, clear p, clear SE, clear Variance, clear Var1,
clear Average, clear v;
   MatA=[A(x-3,y-3),A(x-3,y-2),A(x-3,y-1),A(x-3,y),A(x-3,y+1),A(x-3,y+2),A(x-3,y+3);
A(x-2,y-3),A(x-2,y-2),A(x-2,y-1),A(x-2,y),A(x-2,y+1),A(x-2,y+2),A(x-2,y+3); A(x-1,y-
3),A(x-1,y-2),A(x-1,y-1),A(x-1,y),A(x-1,y+1),A(x-1,y+2),A(x-1,y+3); A(x,y-3),A(x,y-
2),A(x,y-1),A(x,y),A(x,y+1),A(x,y+2),A(x,y+3); A(x+1,y-3),A(x+1,y-2),A(x+1,y-
1),A(x+1,y),A(x+1,y+1),A(x+1,y+2),A(x+1,y+3); A(x+2,y-3),A(x+2,y-2),A(x+2,y-
1),A(x+2,y),A(x+2,y+1),A(x+2,y+2),A(x+2,y+3); A(x+3,y-3),A(x+3,y-2),A(x+3,y-
1),A(x+3,y),A(x+3,y+1),A(x+3,y+2),A(x+3,y+3)];
   lefty=370;
   for Y=lefty:1:y; % Y=1:1:number of columns, this is in the second image
   Average=mean(mean(MatA));
   k=3;
```

```
Var1=0;
for i=-k:1:k
   for j=-k:1:k
      Var1=Var1+(A(x+i,y+j)-Average)^2;
   end
end
Variance=Var1/((2*k+1)^2);
for Y=lefty:1:y% can not start from 1 because Y-3 would be outside the picture, so
starting from pixel 4
   n=0;% this is the counter to place create SEmat
   SE=0;
   for X1=x-1:1:x+1
%calculation of SE(A,B):
      for i=-k:1:k
         for j=-k:1:k
            SE=SE+((A(x+i,y+j)-B(X1+i,Y+j))^(2));
         end
      end
      n=n+1;
      SIMILARITY(n,Y)= SE/(Variance+1);
   end
end
end
```

```
SEsize=size(SIMILARITY);
v=var(SIMILARITY);
vmin=min(min(v(:,lefty:end))); %minimum value of SEmat, starting from 4, because the
first 4 colomns are zero.
if vmin<=1000
        [p,Ypos]=find(v==vmin);
        xR=[x;Ypos(1)]
        xL=[x;y];
else
xR=[x;nan]
xL=[x;y];
%end
end
end
```

The following programs are for rectification process, they are based on algorithm proposed by A. Fusiello [12].

rectify.m

```
function [A,R,t] = art(P)
% ART: factorize a PPM as  P=A*[R;t]
Q = inv(P(1:3, 1:3));
[U,B] = qr(Q);
R = inv(U);
t = B*P(1:3,4);
```

```
A = inv(B);

A = A ./A(3,3);

function [T1,T2,Pn1,Pn2] = rectify(Po1,Po2)

 % RECTIFY: compute rectification matrices

 % factorize old PPMs

[A1,R1,t1] = art(Po1);

[A2,R2,t2] = art(Po2);

 % optical centers (unchanged)

c1 = - inv(Po1(:,1:3))*Po1(:,4);

c2 = - inv(Po2(:,1:3))*Po2(:,4);

 % new x axis (= direction of the baseline)

v1 = (c1-c2);

% new y axes (orthogonal to new x and old z)

v2 = cross(R1(3,:)',v1);

% new z axes (orthogonal to baseline and y)

v3 = cross(v1,v2);

   % new extrinsic parameters

 R = [v1'/norm(v1)

      v2'/norm(v2)

      v3'/norm(v3)];

% translation is left unchanged

% new intrinsic parameters (arbitrary)

A = (A1 + A2)./2;
```

```
A(1,2)=0; % no skew
 % new projection matrices
Pn1 = A * [R -R*c1 ];
Pn2 = A * [R -R*c2 ];
 % rectifying image transformation
T1 = Pn1(1:3,1:3)* inv(Po1(1:3,1:3));
T2 = Pn2(1:3,1:3)* inv(Po2(1:3,1:3));
```

The following program converts from rectified coordinates to image coordinates and vice versa. It is used in propagation of correspondences. The program is based on A. Fusiello's algorithm of rectification [12].

p2t.m

```
function [un,vn] = p2t(H,u,v);
% P2T apply a Projective 2D Transform
% H is an homography of the projective plane
dime = size(u,1);
c3d = [u v  ones(dime,1)]';
h2d = H * c3d;
c2d = h2d(1:2,:)./ [h2d(3,:)' h2d(3,:)']';
un = (c2d(1,:))';
vn = (c2d(2,:))';
```

APPENDIX B

RAW DATA

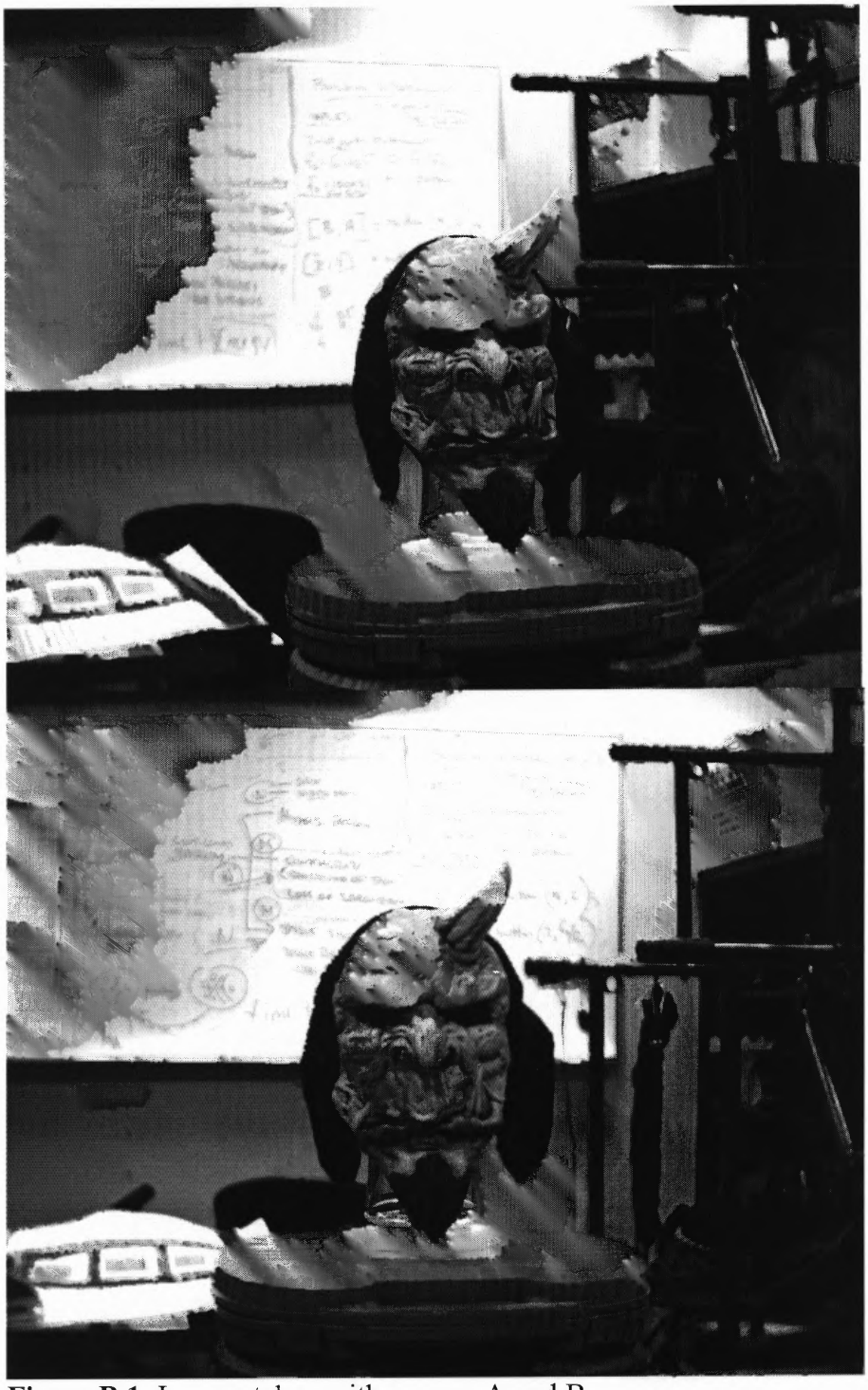

Figure B.1 Images taken with camera A and B.

Figure B.2 Images taken with cameras C and D.

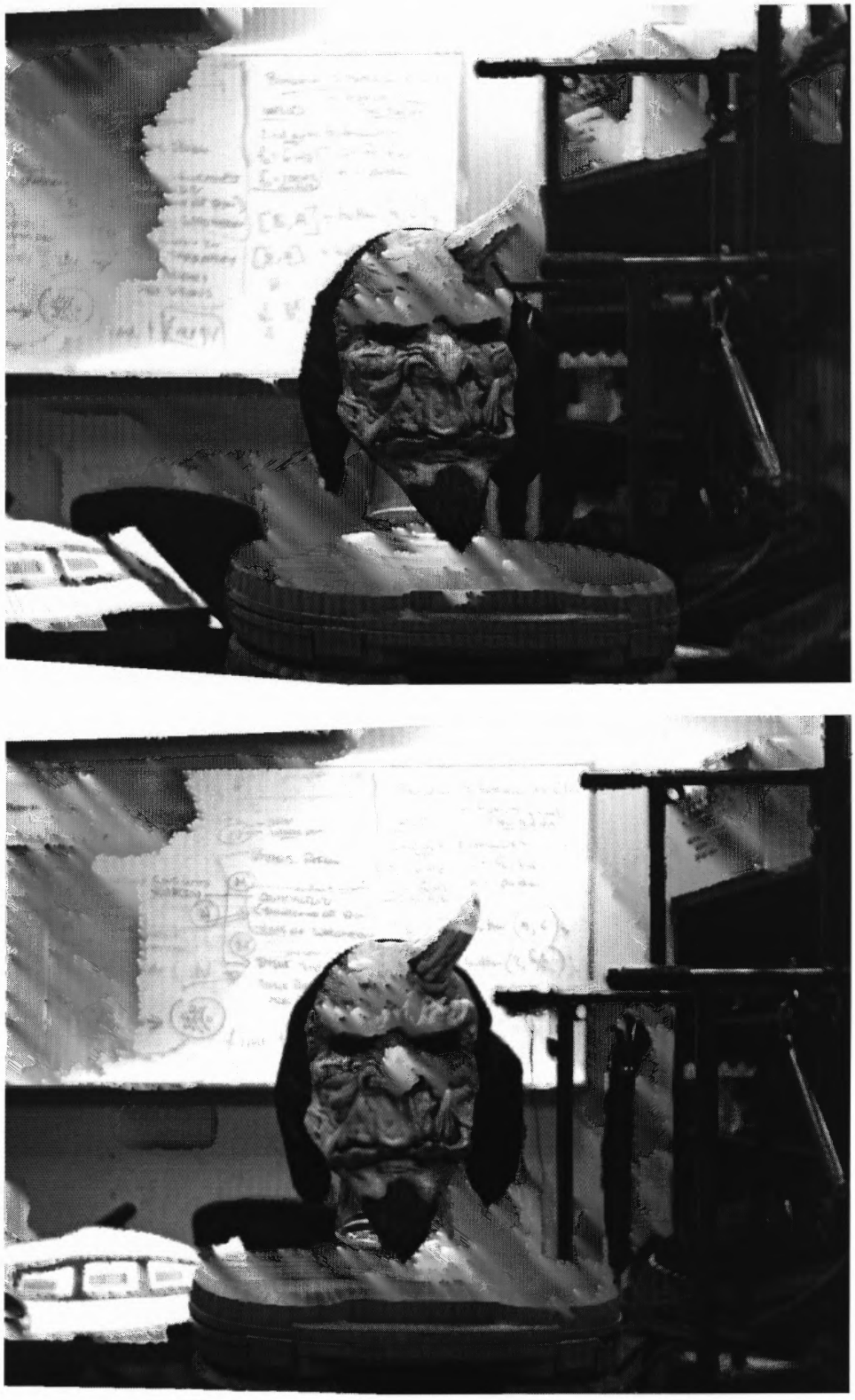

Figure B.3 Rectified images of pair AB.

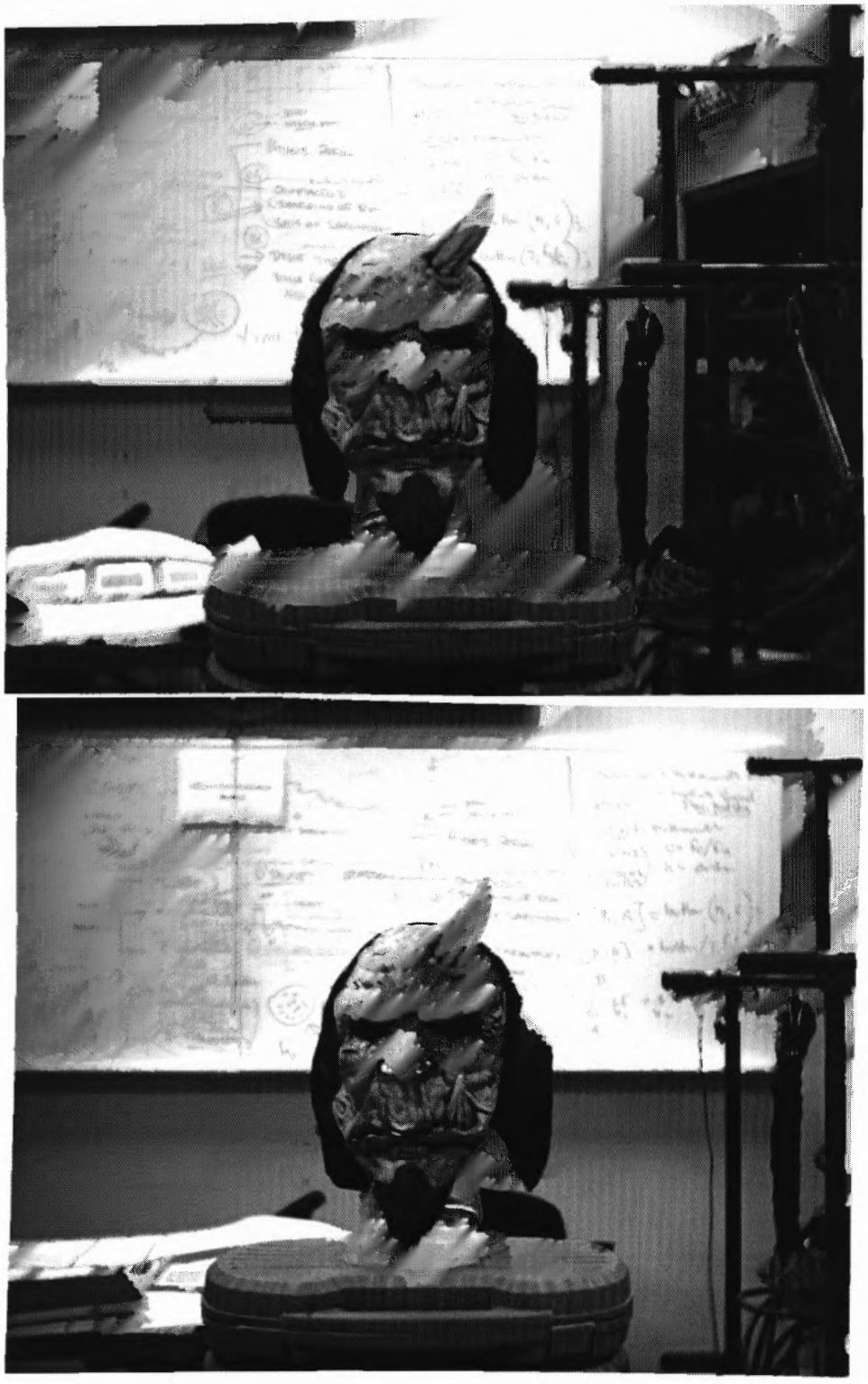

Figure B.4 Rectified images of pair BC.

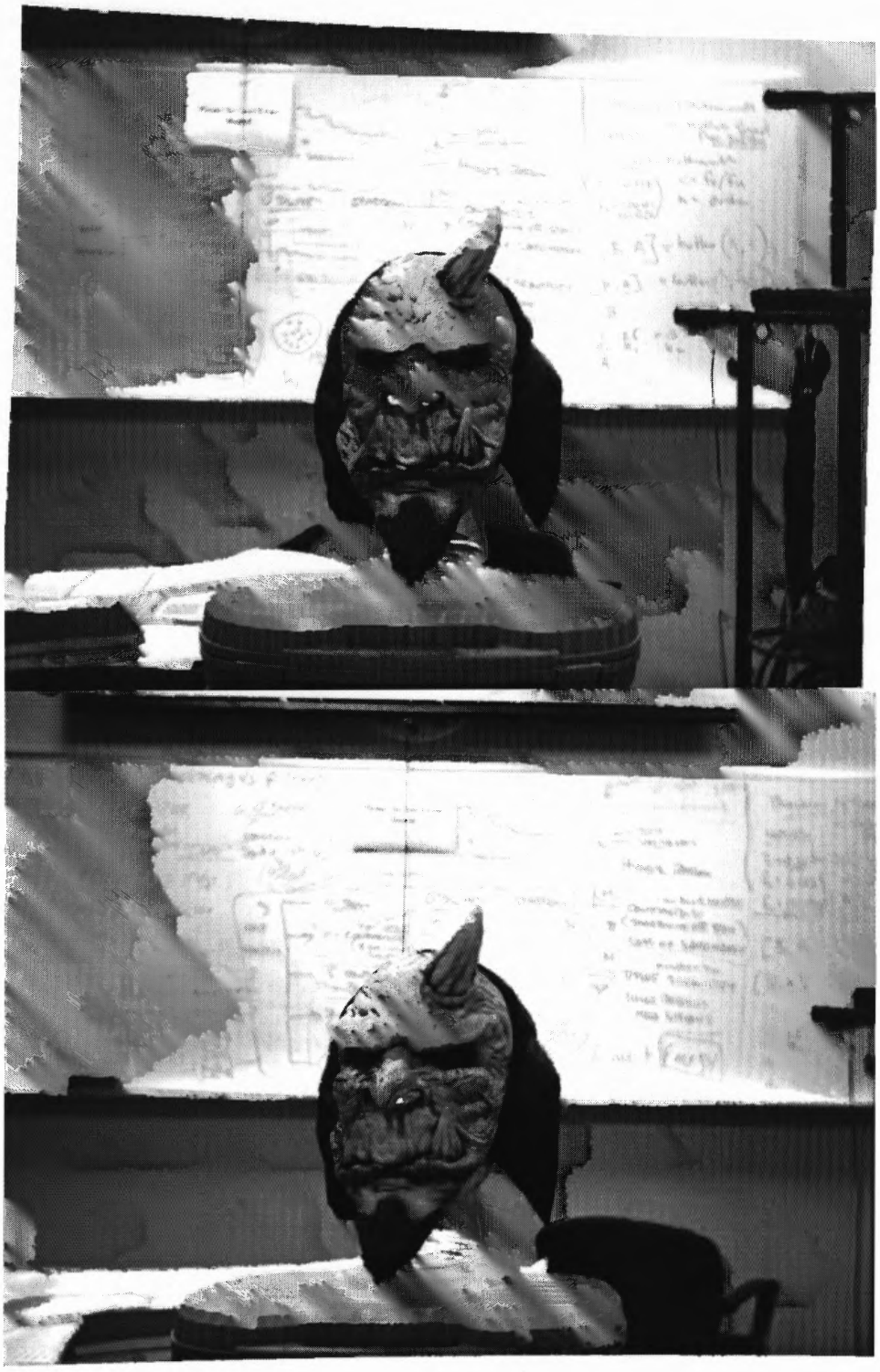

Figure B.5 Rectified images of pair CD.

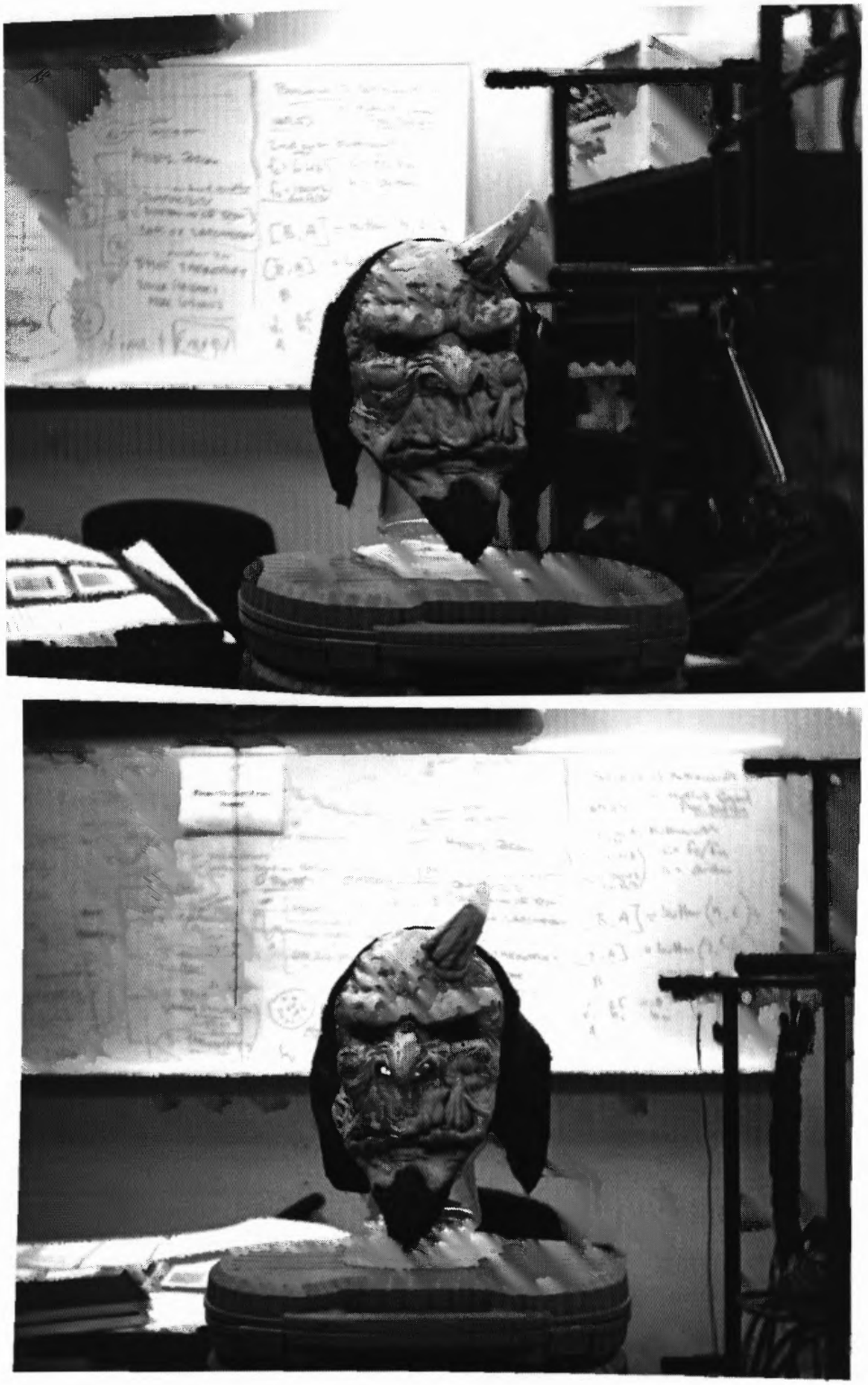

Figure B.6 Rectified images of pair AC.

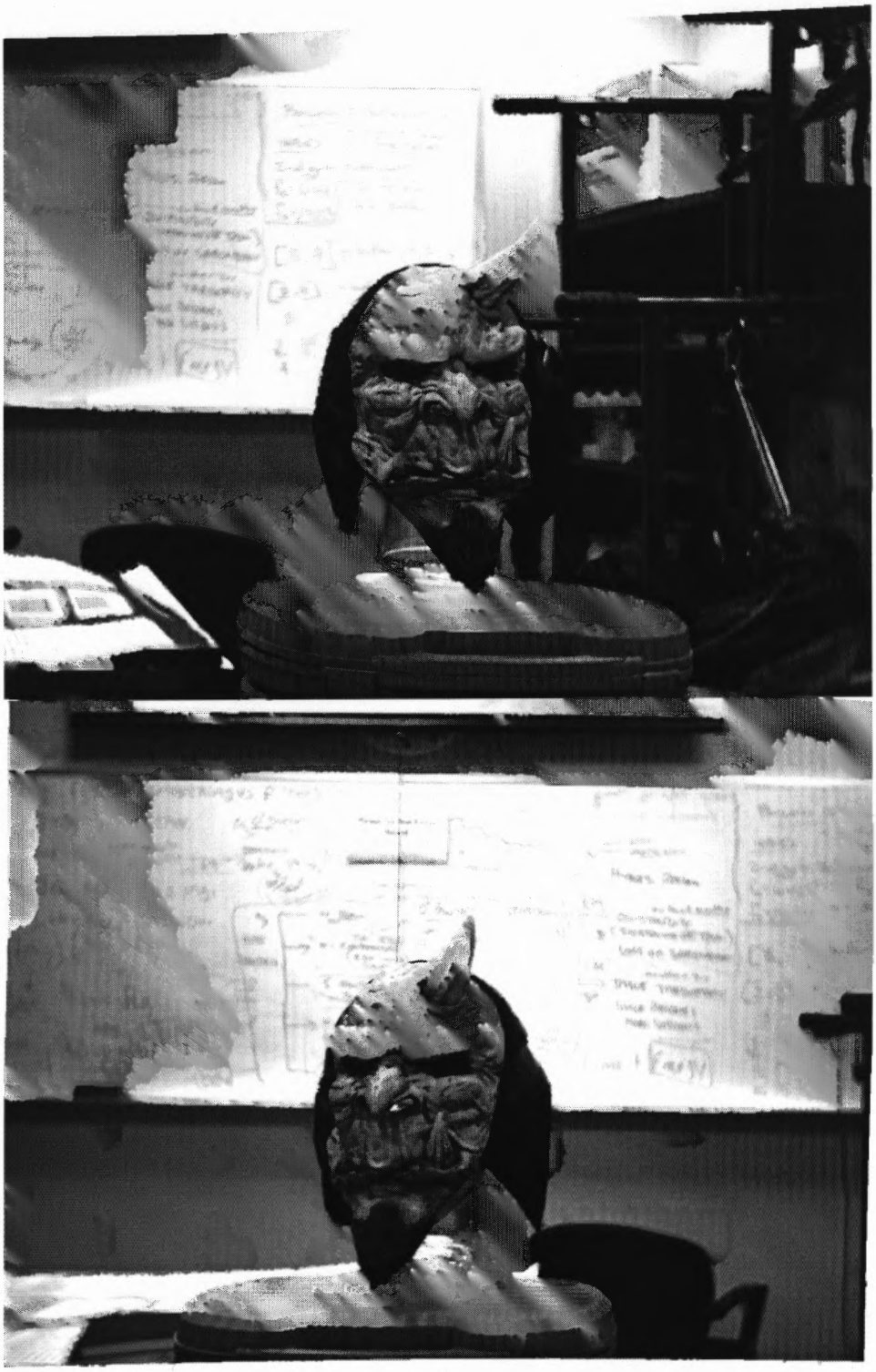

Figure B.7 Rectified images of pair AD.

Table B.1 Calibration results of Camera A

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 2199.64748   2196.64726 ] ± [ 4.05987   4.01119 ]
Principal point:       cc = [ 649.13508    544.46685 ] ± [ 3.15585   2.62383 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
angle of pixel axes       = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.29005    0.62312   0.00078   -0.00022  0.00000 ]
                          ± [ 0.00804    0.09047   0.00022   0.00024   0.00000 ]
Pixel error:          err = [ 0.23285    0.22757 ]
```

Table B.2 Calibration results of Camera B

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 2202.81143   2199.72146 ] ± [ 4.92978   4.86031 ]
Principal point:       cc = [ 639.20951    533.99291 ] ± [ 3.88209   3.15365 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
angle of pixel axes       = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.27200    0.39220   0.00085   0.00232  0.00000 ]
                          ± [ 0.01152    0.13213   0.00027   0.00028  0.00000 ]
Pixel error:          err = [ 0.27260    0.33142 ]
```

Table B.3 Calibration results of Camera C

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 2209.29608   2206.73347 ] ± [ 4.15151   4.16725 ]
Principal point:       cc = [ 682.64762    520.62560 ] ± [ 3.57649   2.61141 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.28686    0.62555   0.00146   0.00411  0.00000 ]
                          ± [ 0.00963    0.10930   0.00025   0.00029  0.00000 ]
Pixel error:          err = [ 0.28865    0.22874 ]
```

Table B.4 Calibration results of Camera D

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 2215.33405   2210.41713 ] ± [ 4.92641   5.09057 ]
Principal point:       cc = [ 684.24833    536.45763 ] ± [ 5.01625   3.30958 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]
angle of pixel axes       = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.26271    0.33375   0.00240   0.00220  0.00000 ]
                          ± [ 0.01005    0.10503   0.00033   0.00041  0.00000 ]
Pixel error:          err = [ 0.28827    0.22332 ]
```

Table B.5 Stereo calibration results of pair AB

```
Extrinsic parameters (position of camera B wrt camera A):

Rotation vector:             om = [ 0.00379    0.12745   0.02668 ]
                                ± [ 0.00253    0.00312   0.00021 ]
Translation vector:           T = [ -179.91960   -0.49756   -34.45164 ]
                                ± [ 0.44492    0.36009   2.28357 ]
```

Table B.6 Stereo calibration results of pair BC

```
Extrinsic parameters (position of camera C wrt camera B):

Rotation vector:             om = [ -0.00858   0.14994  -0.02373 ]
                                ± [ 0.00264   0.00328  0.00025 ]
Translation vector:           T = [ -187.02733   3.35819  6.28237 ]
                                ± [ 0.44808   0.35058  2.29298 ]
```

Table B.7 Stereo calibration results of pair CD

```
Extrinsic parameters (position of camera D wrt camera C):

Rotation vector:             om = [ -0.02778   0.15176  0.00121 ]
                                ± [ 0.00582   0.00751  0.00057 ]
Translation vector:           T = [ -184.25534   6.90580  35.41099 ]
                                ± [ 0.95654   0.71480  4.97058 ]
```

Table B.8 Stereo calibration results of pair AC

```
Extrinsic parameters (position of camera C wrt camera A):

Rotation vector:             om = [ 0.00379   0.12745  0.02668 ]
                                ± [ 0.00253   0.00312  0.00021 ]
Translation vector:           T = [ -179.91960   -0.49756  -34.45164 ]
                                ± [ 0.44492   0.36009  2.28357 ]
```

Table B.9 Stereo calibration results of pair AD

```
Extrinsic parameters (position of camera D wrt camera A):

Rotation vector:             om = [ -0.02724   0.42819  0.00095 ]
Translation vector:           T = [ -549.70457   18.92265  83.55727 ]
```

REFERENCES

[1] P. Witoonchart, and R. Foulds. "Three-Dimensional Surface and Volume Measurements Using a Camera Array." *IEEE 28th Annual Northeast Bioengineering Conference* (2002): 145-146.

[2] R. Klette and others, eds., *Computer Vision, Three-Dimensional Data from Images.* Singapore: Springer, 1998.

[3] W. E. L. Grimson. "A computer implementation of a theory of human stereo vision." *Phil. Trans. Royal Soc. London* (1981) Vol. B292: 217-253.

[4] D. Marr and T. Poggio. "A computational theory of human stereo vision." *Proc. Royal Soc. London* (1979) Vol. B204: 301-328.

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

[6] T. Kanade and M. Okutomi. "A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment." *Proc. IEEE Trans. Pattern Analysis and Machine Intelligence* (1994) Vol.16 No.9: 920-932.

[7] M. Okutomi and T. Kanade. "A Multiple-Baseline Stereo." *IEEE Trans. Pattern Analysis and Machine Intelligence* (1993) Vol.15 No.4: 353-363.

[8] Intel Corp, "Camera calibration Toolbox for Matlab," http://newbologna.vision.caltech.edu/bouguetj/calib_doc/ (10 April 2004).

[9] D. Marr and T. Poggio. "A computational theory of human stereo vision." *Proc. Royal Soc. London* (1979) Vol. B204: 301-328.

[10] Olivier Faugeras, *Three-Dimensional Computer Vision, A Geometric Viewpoint.* Massachusetts: The MIT Press, 1996.

[11] Michael C. Fairhurst, *Computer Vision For Robotic Systems, An Introduction.* New York: Prentice Hall, 1988.

[12] A. Fusiello, E. Trucco, and A. Verri,. "A Compact Algorithm For Rectification of Stereo Pairs." *Machine Vision and Applications* (2000): 400-408.

[13] Luca Iocchi, "Stereo Vision: Triangulation," *Dipartimento di Informatica e Sistemistica Università di Roma "La Sapienza"*, Italy, http://www.dis.uniroma1.it/~iocchi/stereo/triang.html (25 March 2005).

[14] William Eric Leifur Grimson, *From Images to Surfaces, A Computational Study of The Human Early Visual System.* Massachusetts: The MIT Press, 1986.

[15] W. E. L. Grimson. "A Computer Implementation of A Theory of Human Stereo Vision." *Phil. Trans. Royal Soc. London* (1981) Vol. B292: 217-253.

[16] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision.* New York: Cambridge University Press, 2004.

[17] D. Papadimitriou and T. Dennis, "Epipolar Line Estimation and Rectification for Stereo Image Pairs." *IEEE Trans. Pattern Analysis and Machine Intelligence,* (1996), Vol 5 No 4: 672-676.

[18] Nicholas Ayache, translated from the French by Peter T. Sanders, *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception.* Massachusetts: The MIT Press, 1991.

[19] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing.* New York: Prentice Hall, 2002.

[20] The MathWorks, Inc, "Matlab Documentation" http://www.mathworks.com/access/helpdesk/help/techdoc/ (February 2005)

[21] Yoshiaki Shirai, *Three-Dimensional Computer Vision.* New York: Springer-Verlag, 1987.

[22] David Groth, *A+ Complete Study Guide.* New York: Sybex Inc, 2001.