A PARALLEL STEREO RECONSTRUCTION ALGORITHM WITH APPLICATIONS IN ENTOMOLOGY (APSRA)

BY

RAJESH BHASIN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Professor John C. Hart

ABSTRACT

We propose a fast parallel algorithm for reconstruction of 3-Dimensional point clouds of insects from binocular stereo image pairs using a hierarchical approach for disparity estimation.When considering the large collections of insects Entomologists analyze,it becomes difficult to physically handle the entire collection and share the data with researchers from across the world.With the method presented in this thesis Entomologists can create an image data base for the collection and use the 3D models for studying the shape and structure of the insects.With our approach it is also easy to maintain and modify the collections. Feedback collected shows that the reconstructed 3D models are representative of the actual insects and help Entomologists identify & analyze important features of the insects like shape,size, color & texture patterns on the exoskeleton.Experimental results show the algorithm to be robust and accurate with an error of less than 0.5 cm between the dimensions of the original objects and the reconstructed models. We further optimize our results to incorporate multiview stereo which produces better overall structure of the insects.We also present some reconstruction results for faces.

To Mom, Dad & Apsra-the beautiful lady

ACKNOWLEDGMENTS

Firstly, I would like to extend my gratitude to my adviser Prof. John C. Hart for providing me an opportunity to research with the Graphics group at Illinois.His guidance and suggestions have been crucial to my work.My humble gratitude to Intel Corporation for supporting my research under grant from the Universal Parallel Computing Research Center(UPCRC) established at UIUC.

Sincere thanks to my colleague & partner on this project Won Jun Jang who supported me during the toughest times in this project.I have worked closely with Won & this work would not have been possible without him.Special thanks to Chris Dietrich for providing insect specimens & valuable feedback on the reconstructions.Thanks to my labmates -Mahsa,Victor & Kevin, David Raila for his support and Travis Ross & Joel Russ from the Imaging Technology Group at Beckman Institute for their guidance with the imaging systems.

Thanks a lot to my friends in Champaign -Urbana who made my stay wonderful.And lastly thanks to my family -Mom,Dad & my lovely sisters Ramini and Yamini for their unconditional love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

APSRA

Apsra (a Hindi word) translates to 'a beautiful celestial maiden' in English

Image Courtesy [1]

CHAPTER 1

INTRODUCTION

Stereo Vision is a branch of Computer Vision which involves extracing 3D information from images.It is similar to the biological process of Stereopsis.Wikipedia defines Stereopsis as-" The process in Visual Perception leading to the perception of depth from the two slightly different projections of the object on the retina of the eyes" [2].Evidence suggests that there are special cells in the brain(visual cortex) which are senstitive to horizontal mismatch in two images and act as detectors[2].While viewing the same object different images are formed on the retina but are fused into one due to a property known as 'Singleness of Vision'.

1.1 History

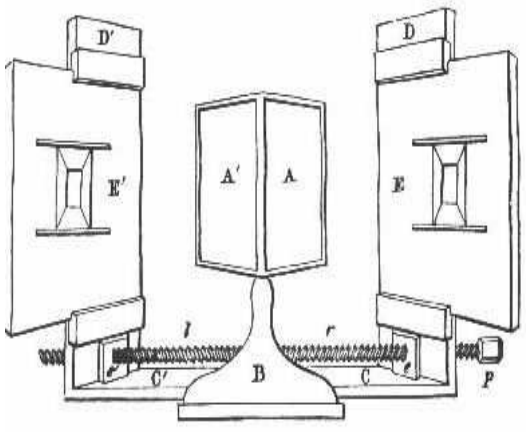


Figure 1.1: Wheatstone's Stereoscope.Courtesy Bill Gamber and Ken Withers

Stereopsis dates back to 1838 when Charles Wheatstone observed that the

slightly different views of objects from the two eyes act as an effective depth cue and create a perception of depth .He used this principle to invent the stereoscope.In 1850,William Brewster invented the Lenticular Stereoscope . Queen Victoria was impressed by the stereoscope displayed at the Crystal Palace Exposition in 1851.Commercial production followed soon & the London Stereoscopic company sold over half a million stereoscopes in less than two years.Stereoscopes also became popular as means of virtual travel.Popular sites included the pyramids of Egypt, European countrysides and scenic views of New York ,London and Chicago.

Figure 1.2: Stereograph of The Great pyramid of Gizeh.Courtesy Underwood & Underwood

The popularity of the stereoscopes reduced towards the beginning of the 20th century with the advent of movies and half-tone images but they were popularized again around the latter half of the 20th century with the invention of ViewMaster,Random-dot Stereograms,Auto-Stereograms etc. among others[2].TIME magazine ranked ViewMaster as one of the 'All Time 100 Greatest Toys'[3].

In recent times Stereo Vision has been used in various projects by NASA.One such example is the use in Mars Rover to produce the elevation maps of the terrain [4].It is also used in the entertainment industry for movie making.Other applications of Stereo Vision include Robotic navigation, modelling of human organs and teleconferencing.

1.2 Problem Statement

Entomologists study insects.They have large collections of insect samples that are used for identification, description and classification of the samples into various groups.Study involves documenting & differentiating species based on their features into various parts of the 'tree of life' .It is a hierarchical structure recording the evolutionary history and classifying all of living organisms mapping the genetic links between them.When the insect samples are prepared they contain labels documenting their features, the location where they were collected from etc. which help build a distribution map of the insects. This can then be used to find out which insects feed on which plants, animals etc.

Applied areas of this research involve differentiating insects which feed on various economically important crops like rice & wheat, insects which feed on animals & others which transmit diseases etc from the harmless ones.This classification is critical for industries manufacturing pesticides, medicines etc and often acts as a tool for them to produce superior and more effective products.

The Illinois Natural History Survey - a research institute recording Illinois' biological history located at the University of Illinois Urbana Champaign(UIUC) is a premier institute in this field. It contains close to 7 million samples and is the tenth largest collection of its kind in North America.One of the biggest problem faced by the researchers in this field is getting access to the data from geographically different locations.The current way to share information involves transporting the actual samples .This involves a lot of time and money.

Also, some of these samples are very rare.Few of the samples in the Illinois Natural History museum are 150 years old and the native vegitation that they were collected from no longer exists.As such any loss or damage to the actual specimen is irreplacable.Also physically moving the insect drawers and analyzing each sample everytime a study is performed increases the chances of wear and tear of the samples as well as maintainence costs.

Digitizing such samples would help create a virtual repository of the specimens.Also if we have high resolution images and reconstruction of such samples we can apply image processing techniques to automatically identify and search for specimens.

1.3 Proposed Solution

The motivation behind this work is to make use of the stereo vision techniques to create a virtual collection of insect samples that is representative of the actual insects but easier to maintain, modify,share and analyze.We propose making use of latest stereo reconstruction algorithms to reconstruct 3D models from stereo images of the insect samples.

Reconstruction of 3D point clouds from captured images is a well researched problem in Computer Vision and employs both active and passive techniques. Active techniques involve using lasers to yield depth information . These methods are generally more accurate but are bulky and expensive.Passive techniques involve vision algorithms and camera geometry for reconstruction.They have a comparitvely less expensive setup.Active systems are usually better at doing reconstruction of scenes at a longer distance compared to passive stereo which works better at shorter distances.

We adopt the passive approach and create a setup of two calibrated cameras in a well-lit studio.Our approach would involve taking stereo images of the whole collection only once.After that we do not need to analyze the actual drawers again-instead we now retrieve reconstructed 3D models from the database.

1.4 Organization

This thesis is divided into 7 major parts.We introduced the problem in this section .In the next chapter we will be talking about the related work and some of the research already done in this area . Chapter 3 looks at the background for our work.Here we define the theoretical concepts used in the thesis and necessary to understand our solution.Chapter 4 forms the backbone of the thesis and discusses our implementation in detail.It also provides the mathematical models and the pseudo code we use in the solution.Optimizations to our solution are discussed as part of Chapter 5.It shows the process of parallelization of the existing serial code to achieve better timing.Results are presented in Chapter 6 and we conclude with the possibilities for future work in Chapter 7.

CHAPTER 2

RELATED WORK

3D reconstruction algorithms from stereo-images are one of the most well researched topics in the vision community.One of the seminal research papers in this area is by Debevec et al. which used still images for reconstructing architecture[5].Kolmogorov and Zabih addressed this problem using graph cuts.They use multiple cameras to take different images from known points of view and then used energy minimization[6].An automatic approach to constructing 3D models from single images was presented by Hoiem et al.[7].
An algorithm to compute a panoramic depth map was presented in [8].Shum and Szeliski developed methods to produce depth maps from large collection of images[9].Several other papers by Szeliski have made notable contribution to this area.[10, 11, 12].[13] talks about reconstructing occluded surfaces.Maitre et al. presented an approach for multiview stero reconstruction from planar camera arrays[14].

Work has also been done in the area of Urban Scene reconstruction [15, 16].A comprehensive survey of the stereo vision algorithms is presented in [17]. With the recent interest in this area an online repository of stereo images and corresponding results is maintained to serve as a benchmark[18].We will compare our results using some images from this dataset.

In SIGGRAPH 2010 Beeler et al. presented a modern passive stereo reconstruction system for face capture[19].This is one of the best modern techniques for stereo reconstruction of facial geometry.It is an inexpensive and reliable system which is able to produce high detail facial reconstructions.Results are calculated with an accuracy of upto a millimeter and pore-scale geometry is reproduced.We use the algorithms from this paper as the basis for our solution.One of the shortcomings of this approach is that it takes close to 20 minutes to produce the output.This is quite a long period of time.Although we do not implement all of the techniques in the given paper (We employ the basic algorithms to generate and refine the disparity map)we

do introduce a new approach by parallelizing the algorithms for speedup.

Other work in this area includes passive facial reconstructions by Martin Klaudiny [20].This paper combines the global approach based on graph cut with a local approach.A stereo face collection is maintained at [21] for research purposes. A good reference on recovering depth and reducing errors is [22].

Application of stereo vision methods to recover 3D models of insects is fairly new and this area has substantial potential for research.Lot of interest in this approach is being shown by various history museums that want their collections to be digitized. The paper by Jianqiang [23] talks about such an application for agricultural pests using binocular stereo vision .It uses PS-15 II lamp and bionic cameras to collect the pest images.Jianqiang discusses a background difference algorithm to separate the pest objects from the images .Feature matching is applied to recover 3D reconstruction of the pests.The results report an error of less than 1.5 %.

However the paper does not show any reconstructed models.It only presents a novel view.Also there is no mention of the time taken for reconstruction in the paper.This thesis on the other hand does show the reconstructed models of the insects which are realistic and we also apply parallelization to the algorithm -a technique which is not implemented in [23].

CHAPTER 3

BACKGROUND

This chapter discusses stereo vision in detail and provides a theoretical background to understand the concepts used in the solution.

3.1 Stereo Vision

Stereo Vision deals with the problem of generating a 3D model of the object given its 2D images from different viewpoints.The underlying goal is thus to perceive depth of the object. Generally the setup consists of two or more cameras.

Before we discuss stereo vision further we need to introduce epipolar geometry .Consider two cameras taking a picture of the same scene from two different points of view as shown in the Figure 3.1 below:

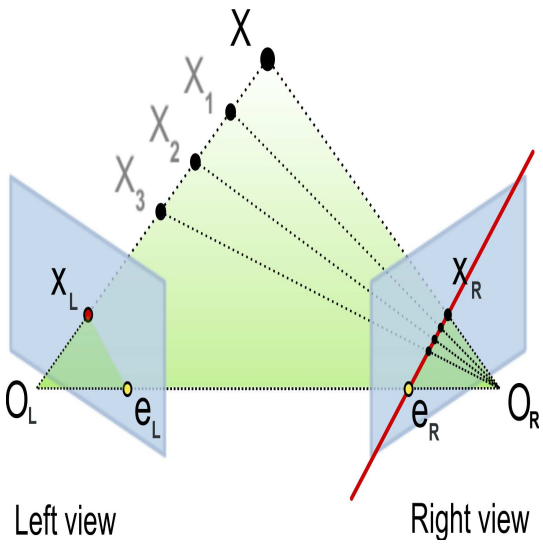


Figure 3.1: Epipolar Geometry.Courtesy Wikipedia

The cameras look at a point X in space.Each of these cameras capture

a 2D image of the 3D world.This conversion is known as the perspective projection.O_R and O_L are the focal points of right and left cameras respectively.Here are a few terminologies associated with the figure above :

- Epipole: A projection of the focal point of one camera on the image plane of the other camera.e_L is an epipole in Figure3.1.
- Epipolar Plane: A plane passing through the centers of projection and the point in the scene.The plane defined by X,O_L and O_R in Figure 3.1(colored in green).
- Epipolar Line: A line formed by the intersection of the image plane with the epipolar plane.e_R-X_R is an epipolar line in Figure 3.1.(colored in red)
- Disparity:The distance between the matching points in the two images.A map containing the disparities of all the points in the image is known as the disparity map.
- Baseline: This is the distance between the left and the right centers of projection.The distance O_L-O_R in Figure 3.1.

If we look at Figure 3.1 a point along the epipolar line e_R-X_R can project back to one of the points X_1,X_2 ...X_N .These points however project onto the same point X_L in the left image.This means that the points along the epipolar line in an image correspond to a unique point in the other image.This is known as the epipolar constraint.This constraint is of great use since we now need to search only along the epipolar line to find possible matches for a point.Hence the problem is reduced to a single dimension.

Triangulation- This is the method used to recover depth of the points from the disparity using camera parameters.Triangulation is based on basic Trigonometry and uses the angles defined to a point from the points at the end of the base of the triangle to find the depth.If we know the baseline and the two points on it we can easily calculate the third point of the triangle.The figure below shows the details:

In the Figure 3.2 L and R represent the left and right cameras respectively.P projects to M in the left image and N in the right image.Notice in the above case the image planes denoted by lM and rN are parallel.Let the

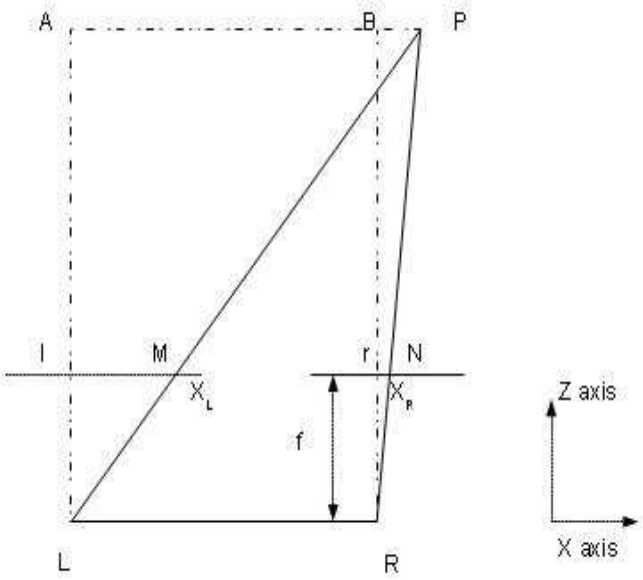


Figure 3.2: Triangulation

3D coordinates of P as seen from left camera be (L_x,L_y,L_z).Consider the triangles APL and lML .
By similarity of triangles

$$\frac{f}{L_z} = \frac{X_L}{L_x} \tag{3.1}$$

On the RHS consider the triangles BPR and rNR .By similarity of triangles

$$\frac{f}{R_z} = \frac{X_R}{R_x}. \tag{3.2}$$

Rearranging equations we get

$$X_L = \frac{fL_x}{L_z}. \tag{3.3}$$

$$X_R = \frac{fR_x}{R_z}. \tag{3.4}$$

But from the figure 3.2 we know

$$L_z = R_z = Z \tag{3.5}$$

and

$$L_x = R_x + b \tag{3.6}$$

where b denotes the baseline(distance between the two camera centers).

Subtracing equation 3.4 from equation 3.3 we get

$$X_L - X_R = \frac{f(L_x - R_x)}{Z} \tag{3.7}$$

$$X_L - X_R = \frac{fb}{Z} \tag{3.8}$$

X_L-X_R is the disparity .Hence we get the formula

$$Disparity = \frac{fb}{Z} \tag{3.9}$$

We will use equation 3.9 later in Chapter 4 to recover the 3D coordinates of the points.

We now introduce two basic problems of the stereo reconstruction process:

- Correspondance problem-It involves finding corresponding points of the first image in the other image.By corresponding here we mean that both correspond to the same physical point.The epipolar constraint that we discussed earlier helps us reduce the search space for finding corresponding matches.

Table 3.1: Correlation Methods

Correlation Methods	
Name	**Definition**
Squared Difference	$\sum_{x',y'}(A(x',y') - B(x+x',y+y'))^2$
Normalized Square Difference	$\frac{\sum_{x',y'}(A(x',y')-B(x+x',y+y'))^2}{\sqrt{(\sum_{x',y'} A(x',y')^2)(\sum_{x',y'}(B(x+x',y+y')^2))}}$
Cross Correlation	$\sum_{x',y'}(A(x',y').B(x+x',y+y'))$
Normalized Cross Correlation	$\frac{\sum_{x',y'}(A(x',y').B'(x+x',y+y'))}{\sqrt{(\sum_{x',y'} A(x',y')^2)(\sum_{x',y'}(B(x+x',y+y')^2))}}$

where

$B'(x+x',y+y') = B(x+x',y+y') - \frac{1}{width*height}.\sum_{x'',y''} B(x+x'',y+y'')$

Basic Methods to solve this include correlation and feature based ap-

proaches.Correlation based approach uses templates for matching and finds out the region having maximum correlation score.Table 3.1 shows few commonly used correlation methods.A denotes the template and B denotes the image .It works better in images which have texture.Window /Template sizes are usually varied.

Feature based methods involve extracting features consistent in both the images.Matching is based on a criteria (eg: orientation) of these detected features.Edges, corners, line segments are commonly used features.One of the problems of this approach is that it gives sparse reconstructions.
In our solution we use a correlation based solution to the problem. The method that we use is Normalized Cross Correlation(NCC).

- Reconstruction problem-This problem involves finding the 3D coordinates of A given two corresponding points a and a'.Given the corresponding points we can find the associated disparity.This disparity can then be converted to the 3D coordinates using the camera parameters as shown earlier in this Chapter.For recovering camera parameters we need to calibrate the cameras which we discuss in the next section.

3.2 Camera Calibration

In this section we are going to describe the process of camera calibration. Camera Calibration is the process of finding parameters specific to the camera.Calibration results are required for various methods including the reconstruction of 3D coordinates from the image disparity map and help us get an accurate representation of the real world.The parameters of the camera are divided into two categories

- Intrinsic Parameters-They are the parameters responsible for relating the pixel coordinates of the image to the coordinates in the camera's reference frame.These include parameters like the focal length and the principal point.

- Extrinsic Parameters-They are the parameters responsible for relating the camera coordinates to a set of world coordinates.Determining ex-

trinsic parameters involves finding out the translation vector and the rotation matrix to align both the frames.

For camera calibration in our solution we use Jean-Yves Bouguet's camera calibration toolkit for MATLAB[24]. We can also use similar methods in OpenCV for camera calibration.A checkerboard pattern is used for calibration .The cameras take the images of the checkerboard pattern in different orientations.Figure 3.3 shows two such stereo image pairs.

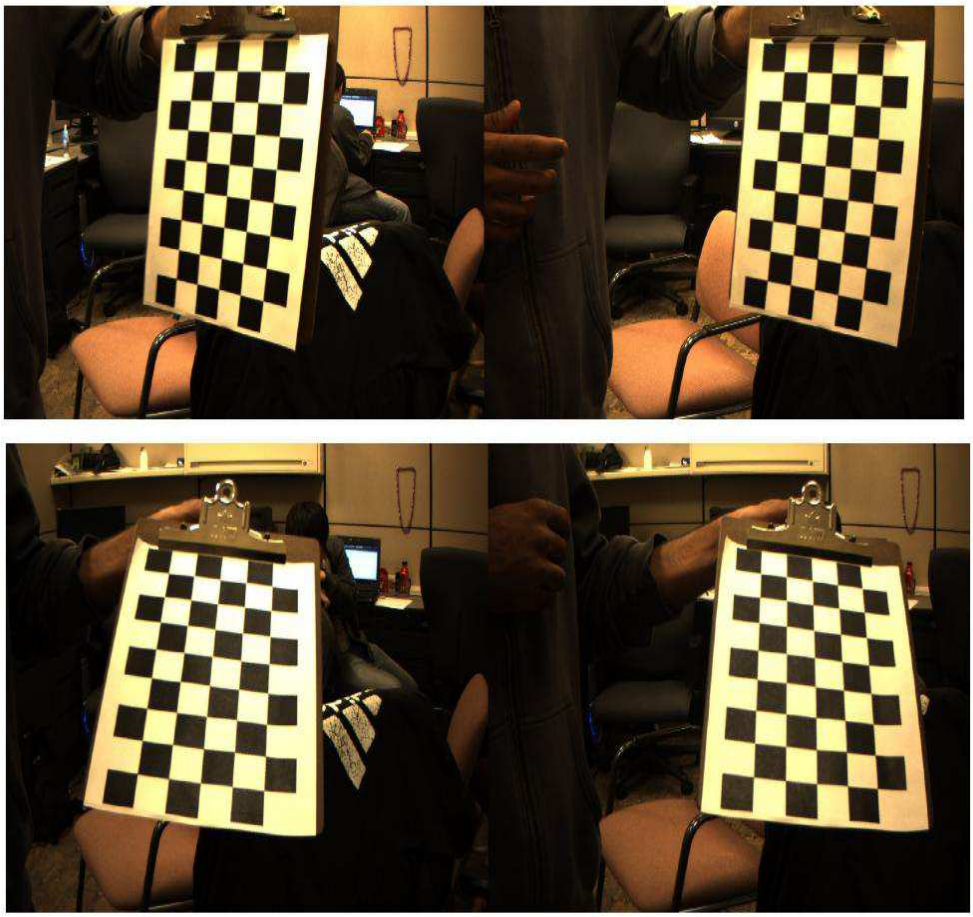


Figure 3.3: Stereo Image Pairs (L-R) used for Calibration

While calibrating the camera it is important to note that the entire pattern must appear in both the images.Initially,all the image pairs are read in.First step is the extraction of the grid corners.User is asked to enter the size of the

boxes on the checkerboard .In some cases it is possible that wrong number of squares are calculated due to distortions.In such a case the user can manually enter the values to correct the output.System is then able to extract the grid corners.
We then repeat this procedure with several images.Usually 8-10 image pairs suffice.If the guessed corners are not close to the actual corners the user can correct the results by guessing the distortion factors.Figure 3.4 shows the extracted corners.

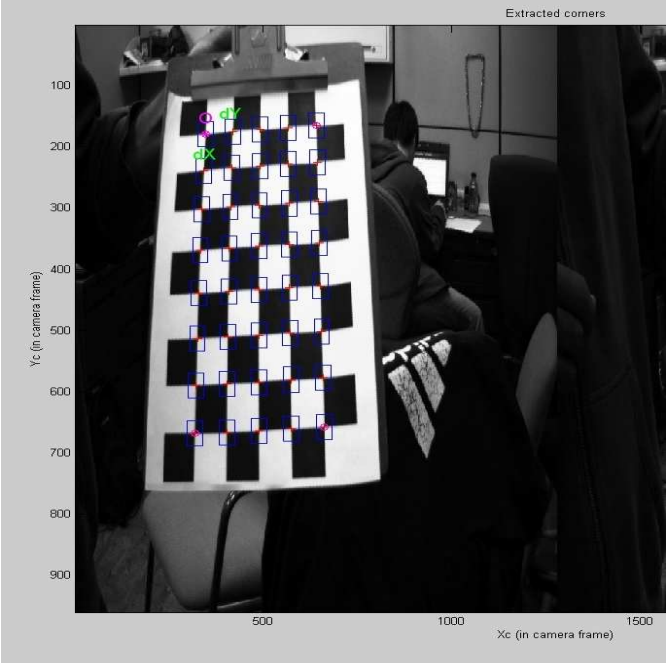


Figure 3.4: Extracted Corners

After the corner extraction the main calibration step is performed.It consists of an initialization step and then optimization of the initial result.The initialization part calculates the solution without assuming any distortion.It is based on the paper by Zhang [25].
The optimization step involves minimizing the reprojection error by repetative gradient descent. The reprojection error can be visualized and recomputations can be done to refine the results.Using this information system is able to solve for the rotation and translation vectors that relate the left and right cameras and also the intrinsic parameters.Figure 3.5 shows the extrinsics based on a set of 5 sample images.

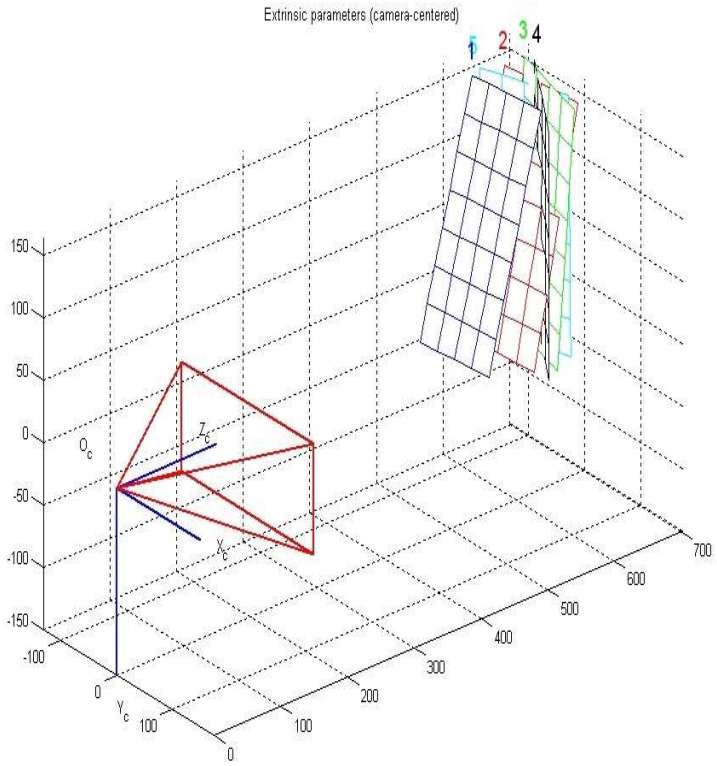


Figure 3.5: Extrinsic Parameters

3.3 Rectification

Image Rectification involves projecting the images on a common plane such that the epipolar lines get aligned horizontally.The benefit of rectification is that the search space for finding correspondances is reduced to a horizontal line in the rectified image.Figure 4.1 below shows the various steps involved-(a) is the raw image,in (b) distortion has been removed,(c) has been rectified and (d) is obtained after cropping.

First step is to apply rotation matrices such that both the cameras become coplanar.Next part involves making the epipoles align horizontally.

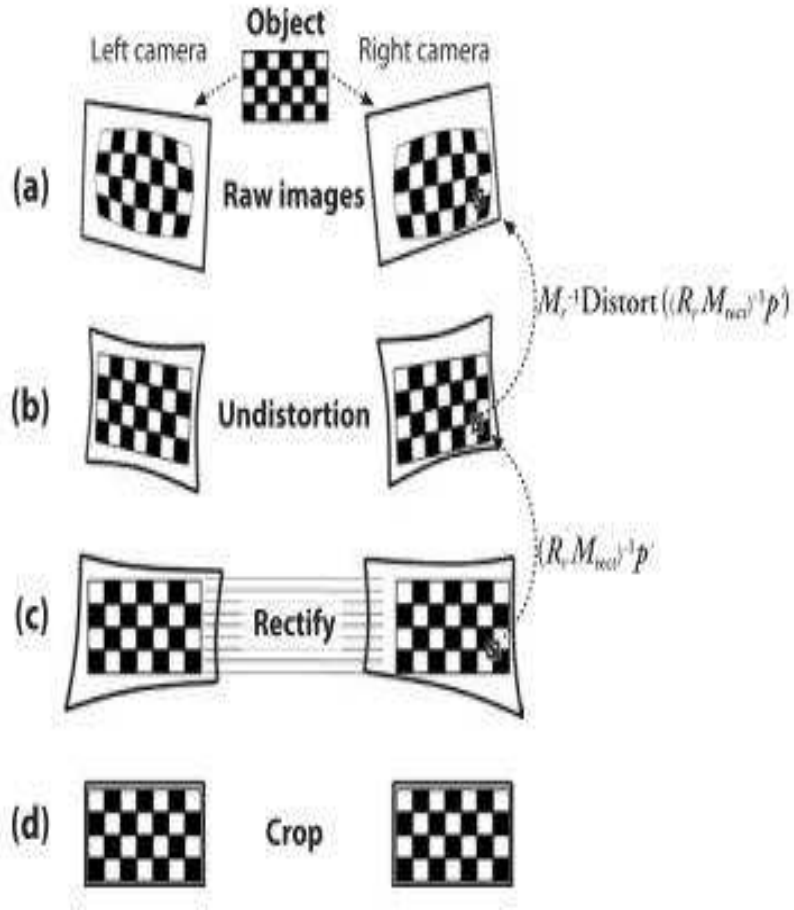


Figure 3.6: Image Rectification .Courtesy Bradski & Kaehler

In our case we have a parallel camera setup.This means that the left and right camera planes are aligned .Although , theoretically in this case we do not need to rectify the images but achieving 100% alignment is not practially possible(and there is always some lens distortion).Therefore for practical implementation purposes and reducing the error we recitfy the images.

For this we use existing methods in OpenCV which can be used to both calibrate the camera and rectify the images.First we use cvFindChessboard-Corners.This finds the corners from the images and stores them.Then we run cvFindCornerSubPix on all the corners extracted.It refines the corner locations to sub-pixel accuracy thereby improving calibration.Then we run cvStereoCalibrate.This method uses all the corners extracted from the calibration images to find the fundamental matrices of the cameras(_M1 & _M2) and it also finds the radial distortion coefficients (_D1 & _D2).

Then we run cvStereoRectify which creates the matrices for rectification using the Fundamental Matrix we found before.Next we use cvInitUndistortRectifyMap to create the undistorted rectification maps using the rectification transformation.Finally we run cvRemap to rectify and undistort the images.

CHAPTER 4

IMPLEMENTATION

This chapter of the thesis describes the entire implementation starting from capturing the images to reconstructing the 3 Dimensional model. The basic algorithms in this chapter are based on [19].We use Point Grey Research's Bumblebee XB3 cameras for image acquisition.The specifications of the camera are:

Table 4.1: Bumblebee XB3 Specification

Specifications	
Name	**Value**
Sensor	3 Sony ICX445 1/3' progressive scan CCDs
Baseline	12cm
Frame Rates	16,7.5,3.75,1.875 FPS
Gamma	0.5 to 4.00
Resolution	1280 X 960

It comes with a serial driver and software tools which are proprietary.This limits the usefulness of the system.Hence,we use OpenCV library to develop the models which makes the system open source and also makes it easier to parallelize the operations.

The flowchart in the Figure 4.1 below shows the overview of the whole process.We begin with the camera calibration step.This step has been already described in section 3.2.This gives us the set of intrinsic and extrinsic parameters of the camera which we will use in the triangulation step.Next step is image acquisition from the bumblebee cameras.The distortion is removed & images rectified before we solve for corresponding points.Details of the same have been discussed under section 3.3.

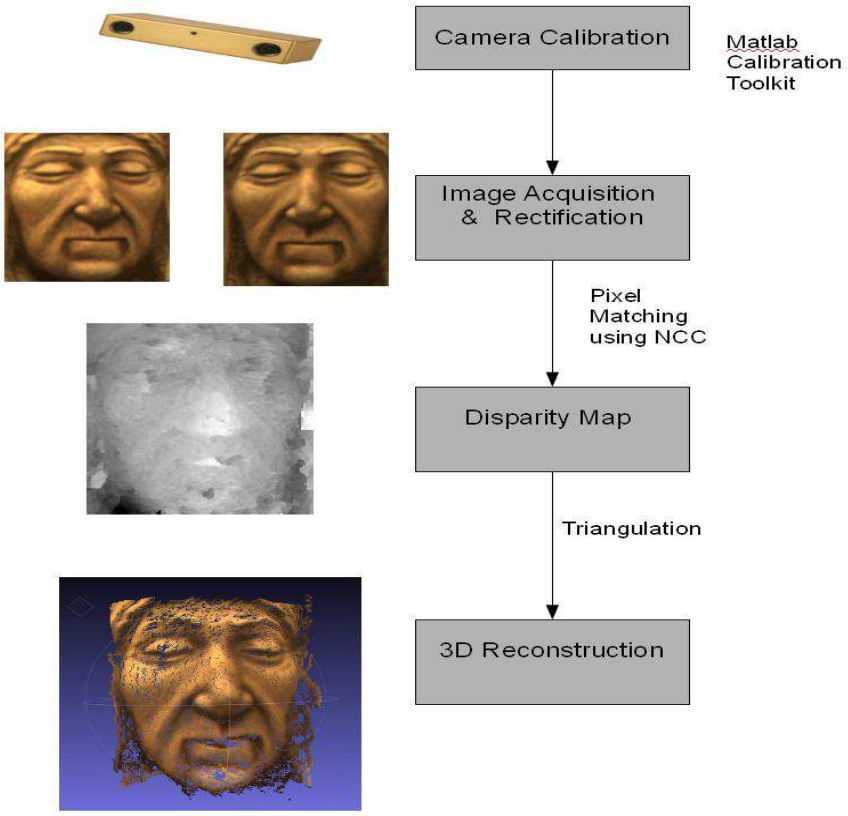


Figure 4.1: Flowchart of the Implementation

4.1 Pixel Matching

We perform matching between the two images to estabish dense matches between them.After obtaining an undistorted and rectified image pair we subsample it by a factor of two each time(using Gaussian subsampling) to generate various levels of a pyramid .The lowest resolution usually goes till 20x20 but can be varied.The lowest resolution layer of the pyramid is then processed first .For this we use Normalized Cross Correlation(NCC) as a metric to find corresponding matches.A match for a given point is searched along the epipolar line in the other image and one with the highest NCC value is retained.It is a robust method for image matching when the lighting conditions can vary.The cross correlation of a template A(x,y) with the

subimage B(x,y) is

$$(\frac{1}{n-1})\frac{\sum x,y(B(x,y)-B')(A(x,y)-A')}{S_A S_B} \tag{4.1}$$

where n is the number of pixels , A' is the average of A, S_A is the standard deviation of A and similarly for B.

In our implementation we use a 5X5 window for matching.For doing the actual matching we use OpenCV's cvMatchTemplate function with the method being CV_TM_CCORR_NORMED.This means the result is R(x,y) is defined as :

$$R(x,y) = \frac{\sum x',y'(T(x',y').I'(x+x',y+y'))}{\sqrt{(\sum x',y'T(x',y')^2).(\sum x',y'I(x+x',y+y')^2)}} \tag{4.2}$$

where I and T denote the image and the template respectively.Summation is done over the width and height.Matches are returned using the MinMaxLoc function.This returns integer disparity.Using integer values in reconstruction limits the accuracy of the 3D model.Hence we compute the disparity to a sub-pixel accuracy.We use interpolation between known values to do this.Parabola fitting is performed.The equation of a parabola is

$$y = ax^2 + bx + c \tag{4.3}$$

In our case we use y as the NCC values and x as the disparity values.We note the NCC value and the disparity value (d) returned by the cvMatchTemplate function.We now find the NCC assuming disparity to be d-1 and d+1.This gives us 3 equations and we have 3 unknowns a,b and c to solve for.This system can be easily solved and we now have the approximating parabola.We find the maximum value of NCC and return the disparity corresponding to it which now has sub pixel accuracy.

This is the first step in disparity calculation at a given level in the image pyramid.We next perform a constraint check on the disparity values .

4.2 Constraint Checking

In order to refine and improve the values that we obtained from the previous step we run a constraint check on these values.There are 3 major constraints that we check as described in [19].

- Smoothness Constraint -The aim of this constraint is to ensure that the value at a pixel is consistent with its neighbors.For this constraint to pass more than half of the neighbors in the 5X5 window must have a disparity difference not greater than one.
- Uniqueness Constraint -This constraint implies that the match for the left pixel in the right image should be the same as the match for the right pixel in the left image.We allow them to differ by atmost a pixel.
- Ordering Constraint-This constraint ensures that the match does not overstep neighboring pixel matches.In effect, disparity at a pixel does not exceed the disparity of its right neighbour by more than 1.

The disparity values are considered correct only if it passes all the three constraints.Otherwise we use the average of the pixels in the 5X5 neighborhood that passed as the disparity at that pixel.

4.3 Disparity Refinement

Next step involves iteratively improving the depth map by photometric consistency (d_p-based on NCC gradient of disparity neighbours) and surface consistency (d_s-based on a curvature weighted filter to preserve features but smooth noise).We use the techniques discussed in [19] to achieve the refinment.
To find dp we use the formula :

$$d_p = \begin{cases} m - n - 0.5 & \text{if } N_{-1} < \mathrm{N}_0, N_1 \\ m - n + 0.5 & \text{if } N_1 < N_{-1} \text{ ,}\mathrm{N}_0 \\ m - n + 0.5((N_{-1}\text{-}\mathrm{N}_1)/(\mathrm{N}_{-1}\text{+}\mathrm{N}_1\text{-}2\mathrm{N}_0)) & \text{if } N_0 < N_{-1} \text{ ,}\mathrm{N}_1 \end{cases}$$

where m and n are the corresponding pixels in the images and N_k= (1-NCC_k)/2 .The subscripts -1,1 indicate the left and right neighbors of n.

Similarly we calculate d_s as

$$d_s = \frac{w_x(d_(x-1,y) + d_(x+1,y)) + w_y(d_(x,y-1) + d_(x,y+1))}{2(w_x + w_y)} \tag{4.5}$$

where w_x= $exp(-(|d_(x-1,y) - d_(x,y)| - |d_(x+1,y) - d_(x,y)|)^2)$

Finally we get updated disparity as d' which is defined as :-

$$d' = \frac{w_p d_p + w_s d_s}{w_p + w_s} \tag{4.6}$$

w_s is user specific and w_p is as desribed in [19].Refinement can be done for a number of iterations.We usually carry out around 15-20 iterations.
At this point we get the disparity at a level and it is used as an initial guess for the next level.This helps to restrict the search at the new level.This procedure is subsequently continued at each level till it reaches the highest level i.e. the original resolution of the image.By this point we have sufficiently reduced the search space and have also taken local disparity into account.The output of this stage in the form of a disparity map is passed on to the next step which is 3D reconstruction. .

4.4 3D Points Reprojection

Reprojecting 2-D points from a set of images to 3-D points is done using disparity values of the pixel and the camera parameters.Each pixel re-projects to a point (X,Y,Z) in world coordinates. We use the triangulation formulae to recover depth from the disparity values.We have already discussed the derivation in detail in Chapter 3. In particular we have:

$$Z = \frac{FB}{d} \tag{4.7}$$

$$Y = \frac{vB}{d} \tag{4.8}$$

$$X = \frac{uB}{d} \tag{4.9}$$

where F is the focal length (in pixels), B is the baseline (in units of length)and d is the disparity(in pixels). u and v are horizontal and vertical coordinates w.r.t to the principal point of the image. This means

$$u = col - centerColumn \tag{4.10}$$

$$v = row - centerRow \tag{4.11}$$

Using these formulae we can easily reproject the disparity to a point cloud set.

4.5 Pseudo Code

This section gives the pseudo code of the algorithm we use :

Algorithm for Disparity Calculation

for all image pairs **do**
  for all pyramid levels **do**
    for $height = 2 \rightarrow imageheight - 2$ **do**
      Create a 5 by imagewidth ROI of both left and right images
      for $width = 2 \rightarrow imagewidth - 2$ **do**
        Create a 5 by 5 window of the left ROI and find match in the right ROI
        Create a 5 by 5 window of the right ROI and find match in the left ROI
        Disparity=width-match
        Perform sub-pixel interpolation
      end for
    end for
    Assign disparity to edge pixels using approximation from neighbors
    for $height = 0 \rightarrow imageheight - 1$ **do**
      for $width = 0 \rightarrow imagewidth - 1$ **do**
        Check smoothness, uniqueness & ordering constraints
      end for
    end for
    for $height = 0 \rightarrow imageheight - 1$ **do**
      for $width = 0 \rightarrow imagewidth - 1$ **do**

Assign disparity of neighbors to pixels which failed the constraints
end for
end for
for $height = 0 \rightarrow imageheight - 1$ **do**
for $width = 0 \rightarrow imagewidth - 1$ **do**
Check uniqueness constraint
end for
end for
for $i = 1 \rightarrow numberofiterations$ **do**
Refine disparity at each pixel
end for
end for
end for

CHAPTER 5

OPTIMIZATION

In this chapter we discuss a few improvements that were made to the basic algorithm implemented in the previous chapter and based on [19].

5.1 Parallelization

After implementing a serial reference version of the algorithm we parallelize it to obtain speedup. For parallelization we use Intel's Thread Building Block(TBB) library. We parallelize various tasks including subsampling the images & constraint checking.However the main candidate for parallelization is the function which performs template matching in the left and the right images to calculate the disparity.

The details in this section are based on TBB's tutorial[26]The simplest option to introduce parallelism is to parallelize loops whose iterations are independant of each other.To use the TBB library we have to first convert the loop into a body object(a STL like object) which has a *operator()* operating on a chunk.Below we show a simple example of converting a loop into a body object based on the discussion in ([26]). Given a simple loop :

```
for (int i=0; i<iterations;i++)
{
do something
}
```

This is converted to a body object as :

```
class Applyloop{
float *const my_argument;
public:
void operator()( const blocked_range<int>& range ) const {
float *arg = my_argument;
for( int i=range.begin(); i<range.end(); i++ )
do something
}
Applyloop( float arg[] ) :
```

```
my_argument(arg)
{}

}
```

The iteration space here is represented by range and is from 0 to iterations-1.Here blocked_range is use to iterate over a 1 dimensional space.We later also use blocked_range2d for iteration over 2 dimensional space. The body object has a copy constructor which creates separate copies for each thread.

After converting the loop body into a body object we can invoke the template function parallel_for:

```
parallel_for(blocked_range<int>(0,numberofiterations),
 Applyloop(arg));
```

In the above example we have an iteration space going from 0 to number of iterations-1.The constructor is specified as blocked_range <T >(begin, end,grainsize).Here we use a default grainsize of 1.This grain size can be selected dynamically.We initially use the automatic chunking but later on turn it off and use dynamic chunking instead.The grainsize and the partitioner control the chunking of loop iterations.The following code shows a grain size G introduced into the equation

```
parallel_for(blocked_range<int>(0,numberofiterations,G),
 Applyloop(arg),simple_partitioner())
```

A simple_partitioner ensures that

$$\frac{G}{2} \leq chunksize \leq G \tag{5.1}$$

We could also just specify the grainsize for the range and use an auto_partitioner and affinity_partitioner .These partitioners ensure that the chunk size is always more than G/2.The following are some pieces of sample code which show the parallelization being done:

```
for (i=0;i<NO_OF_RECTIFIED_IMAGES;i+=2)
{
for (j=tot_pyramid_levels[i/2]-1;j>-1;j--)
{
image_height=min(image_pyramid[i][j]->height,
image_pyramid[i+1][j]->height);
int left_image_width=image_pyramid[i][j]->width;
int right_image_width=image_pyramid[i+1][j]->width;
parallel_for(blocked_range<size_t>(2,image_height-2,
image_height/2),ApplyBody(left_image_width,right_image_width,
i,j,image_pyramid),simple_partitioner());
}
```

```
}
```

```
void Body( int left_image_width,int right_image_width,
int curHeight,int i ,int j,IplImage*** image_pyramid){
// create 5 by left_image_width ROI centered around curHeight
IplImage* leftImageROI=cvCreateImage(cvSize(left_image_width,5),
image_pyramid[i][j]->depth,image_pyramid[i][j]->nChannels);
for (int ii=0;ii<left_image_width;ii++)
        for (int jj=0; jj <5 ;jj++)
{
CvScalar  r = cvGet2D(image_pyramid[i][j],curHeight-2+jj,0+ii);
cvSet2D(leftImageROI,jj,ii,r);
}

// create 5 by right_image_width ROI centered around curHeight
IplImage* rightImageROI = cvCreateImage(cvSize(right_image_width, 5),
image_pyramid[i+1][j]->depth,image_pyramid[i+1][j]->nChannels);
for (int w = 0; w < right_image_width; w++)
{
   for (int h = 0; h < 5; h++)
{
 CvScalar r = cvGet2D(image_pyramid[i+1][j], curHeight-2+h, 0+w);
 cvSet2D(rightImageROI, h, w, r);
}
}

//*Main candidates for parallelization below*
MatchLeftToRight(left_image_width,right_image_width,
leftImageROI,rightImageROI,i,j,curHeight);
MatchRightToLeft(left_image_width,right_image_width,
leftImageROI,rightImageROI,i,j,curHeight);

cvReleaseImage(&leftImageROI);
cvReleaseImage(&rightImageROI);
}
```

The code snippet above shows one of the main candidate tasks that we parallelize.In the snippet below we show the code with the constraint checking task parallelized.

```
void checkConstraints(int image_height,int image_width,
float ** disparity_map, float **other_disparity_map, int** checkarray){
parallel_for(blocked_range2d<size_t>(0,image_height,0,image_width),
ApplyCC(image_height,image_width,disparity_map,other_disparity_map,
checkarray));
}
```

In order to explore more parallelism in our code we use some of Intel's tools which are a part of Parallel Studio 2011. Among these are Advisor, Amplifier, Inspector and Composer tools. Advisor helps us find out potential sites for parallelism. Amplifier helps in finding out where the application is spending its time so that we can optimize it. Composer provides support for various

parallel models and highly optimized multicore capable libraries. Inspector provides helpful information about the errors in multithreaded code.

The code below shows how to annotate tasks to find out possible gains by parallelization:

```
ANNOTATE_SITE_BEGIN(MySite3);
for (int curHeight=2;curHeight<image_height-2;curHeight++){
ANNOTATE_TASK_BEGIN(MyTask3);
Body(left_image_width,right_image_width,curHeight,i,j,image_pyramid);
ANNOTATE_TASK_END(MyTask3);
}
ANNOTATE_SITE_END(MySite3);
```

To make the algorithms work in parallel we make sure that our individual code portions are thread safe. In some cases we cannot use the default functions of OpenCV and use our own methods instead to copy data to local variables (to prevent changing the state of the source image).

5.2 Multiview Stereo

To improve the quality of the results we take multiple shots by placing camera at different view points around the object and generate point clouds.We get multiple point clouds which correspond to different parts of the same object.We need to align these point clouds to obtain a better reconstruction than a single view.For this we use the Iterative Closest Point (ICP) algorithm.

ICP is an algorithm which refines the rotation and translation between two point clouds till the difference between them reaches a minimum.The algorithm inputs the two point clouds with an initial guess for the transformation.It relates a point in the object to the point in the model by the nearest neighbor.Then it uses a mean square cost function to estimate the transformation such that it reduces distance between corresponding points in the model and object.This process is done repetatively till the cost function is minimized or a required threshold is achieved[27].

We use the implementation present in MeshLab for our project.The figure 5.1 below shows 2 point clouds (without texture mapping) before and after implmenting the ICP algorithm.Pink and blue represent the two point clouds.As visible from the figure the point clouds on the left appear separated but the ones on the right appear merged and are better aligned.

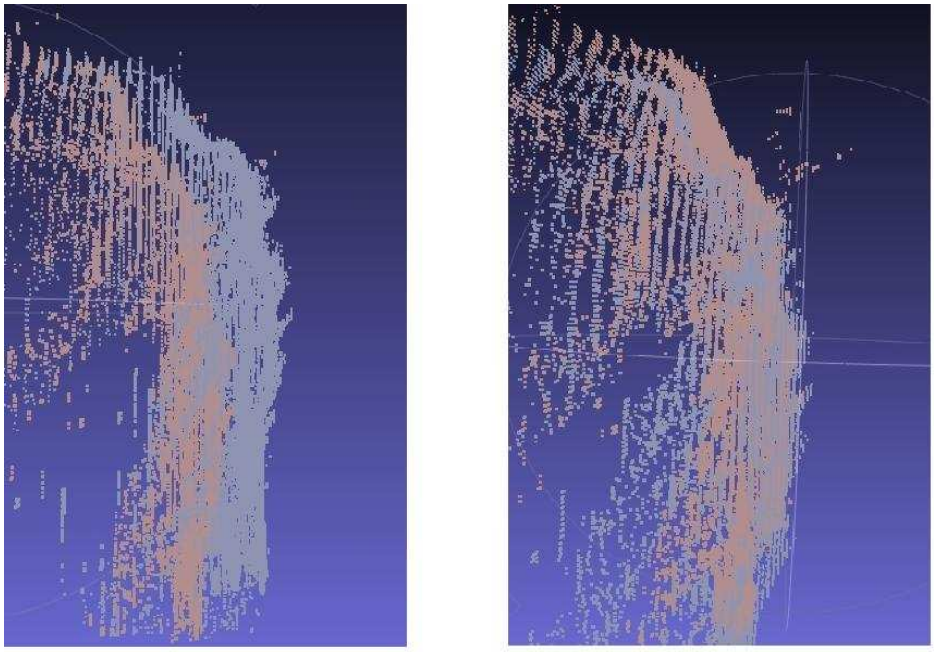


Figure 5.1: Results of ICP on Sideview of a Face -Before(L) & After (R)

CHAPTER 6

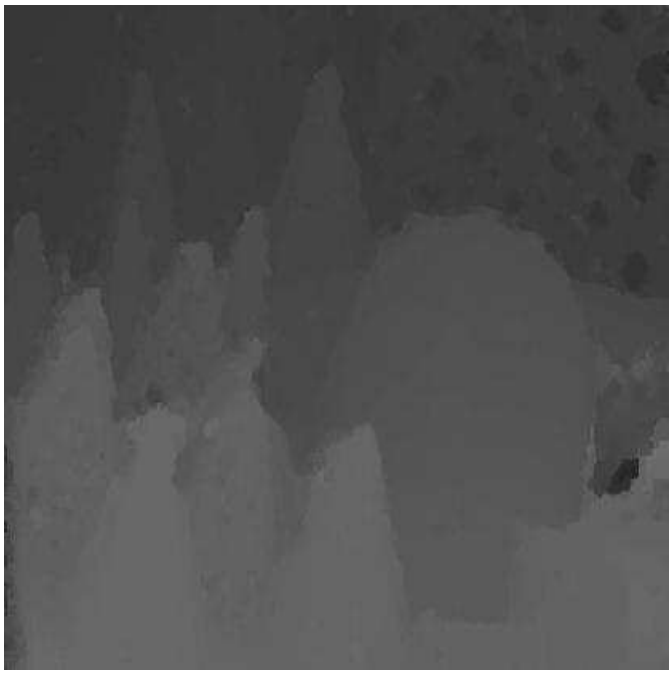
RESULTS AND DISCUSSION

In this chapter we present reconstruction results for both a face and the insects for the various stages of reconstruction. We get good reconstruction results for both- well representative of the original objects.

At the end of this chapter we present a comparitive study of the original and the reconstructed models .We reported an error of less than 0.5 cm between the dimensions of the real object and the reconstructed model.We tested our code on some of the images from the Middlebury dataset.It contains lots of samples of stereo image pairs with associated disparities.It also contains evaluation of various algorithms.This dataset is considered a benchmark for stereo matching algorithms[18].Scharstein and Szeliski also published a paper on the taxonomy of various stereo algorithms [28].

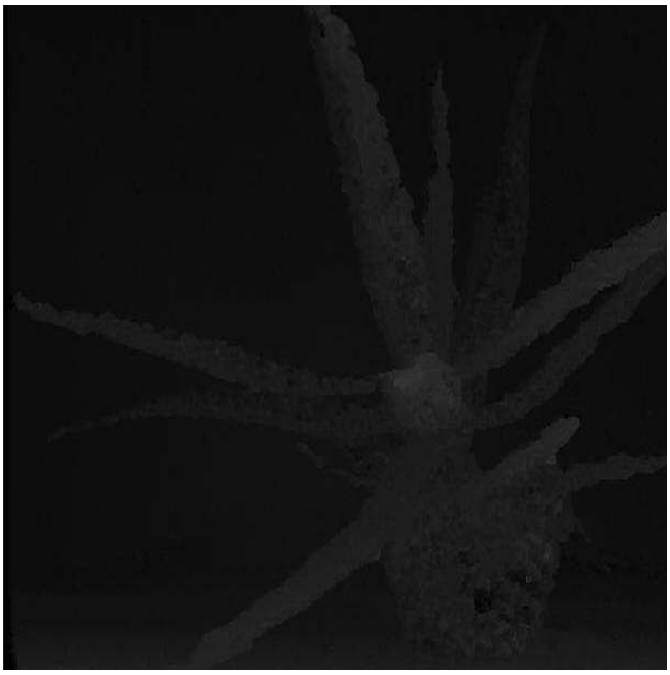
The first image pair shows the cones from the 2003 dataset [29] and the second one shows the aloe from the 2006 dataset[30].The cones dataset was made by Daniel Scharstein, Alexander Vandenberg-Rodes and Rick Szeliski.The aloe dataset was produced by Brad Hiebert-Treuer, Sarri Al Nashashibi, and Daniel Scharstein.The images below contain the original left image followed by the original right image on its side.This is followed by the disparity map of the corresponding stereo image pair.

Image Courtesy [18]



Disparity Map

Image Courtesy [18]



Disparity Map

We also present disparity maps of some general images that we used to test our code.Shown below is one such pair along with the disparity map produced.

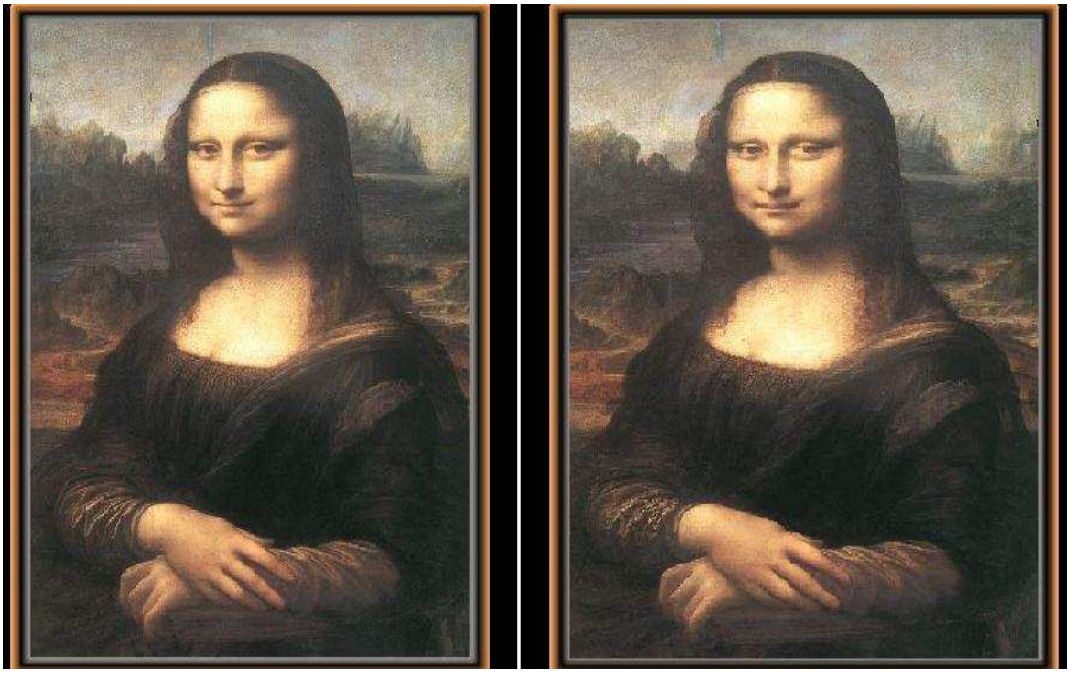
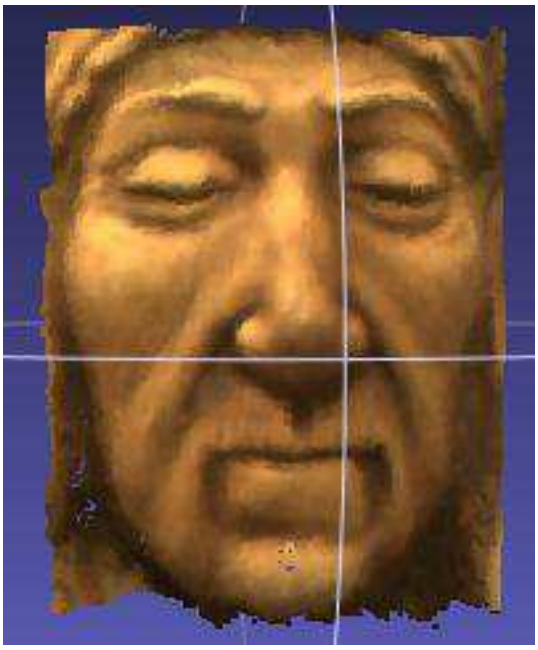
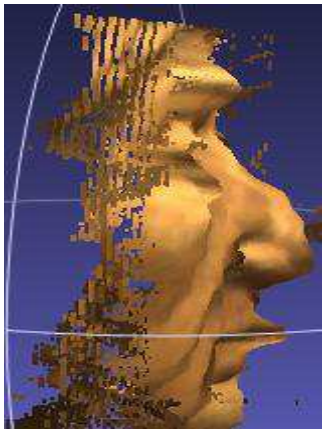


Image Courtesy [1]

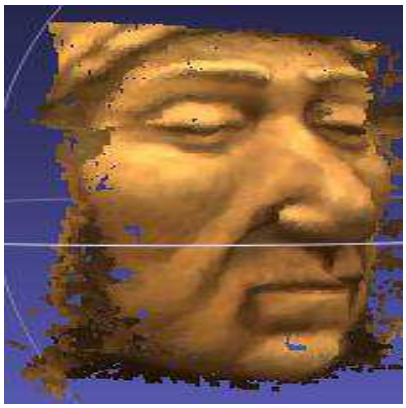
Disparity Map

Next we present some of the results from the reconstruction of the face.

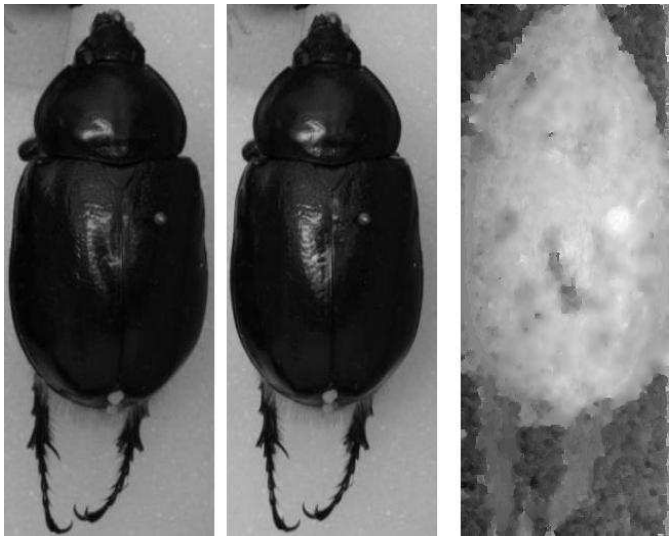




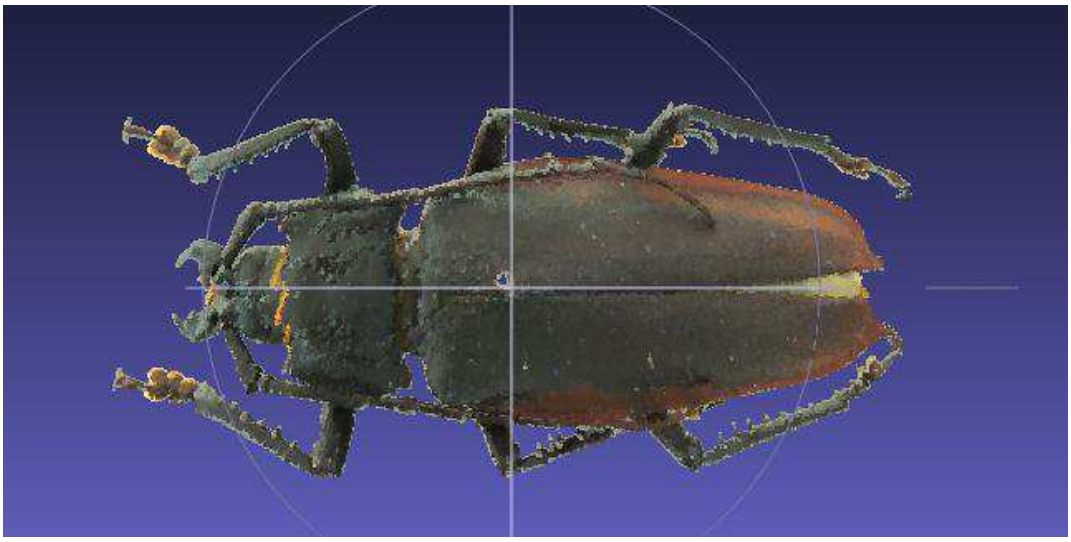
The first screenshot above shows the reconstructed face from the front and the next screenshot shows it from the side.Below is another view of the same reconstruction taken from an angle.

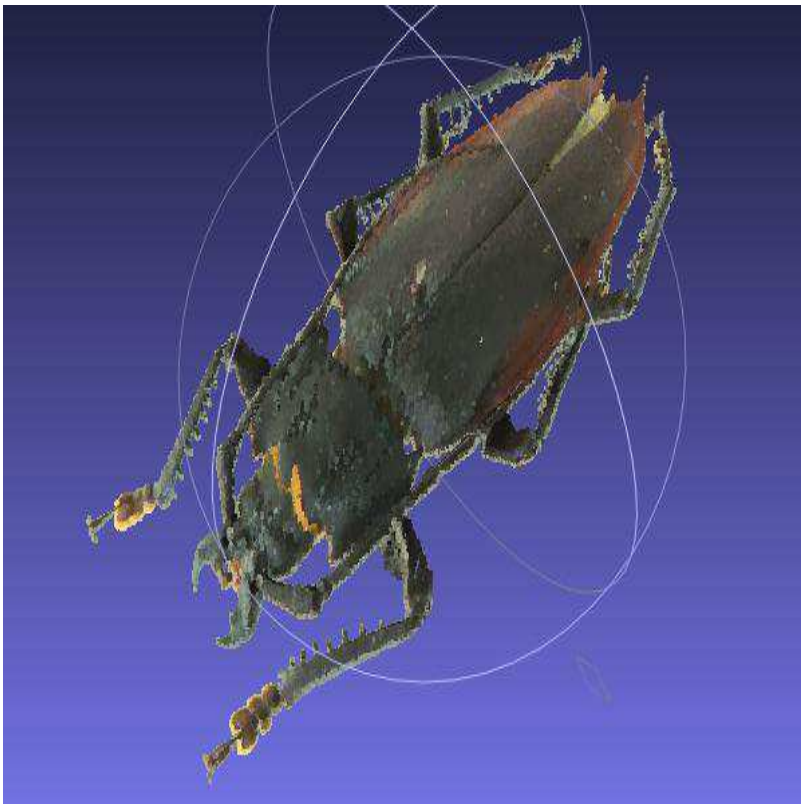


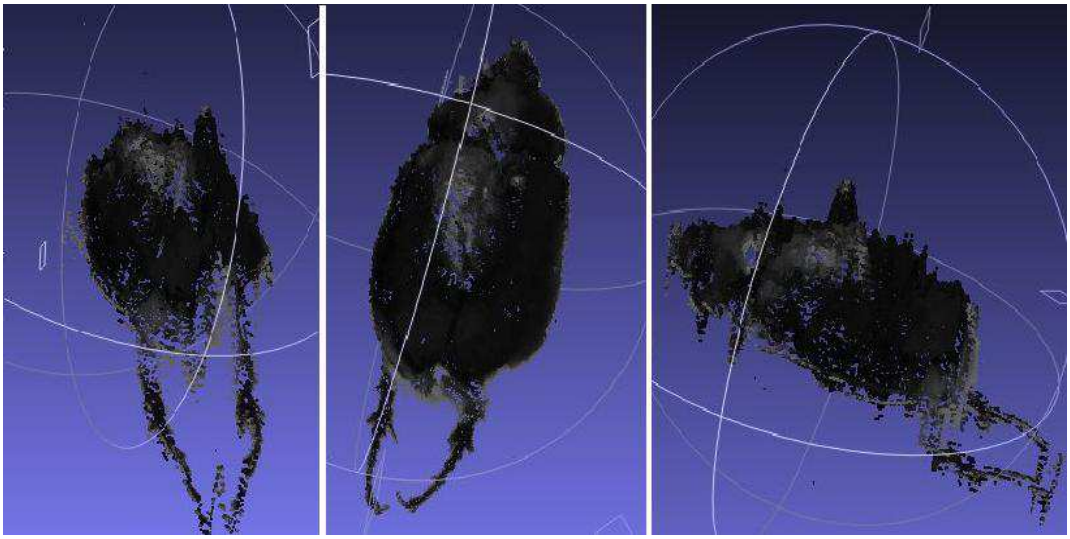
Next we present some of the reconstruction results for the insect samples.Images below show the original images and the corresponding disparity maps.It is important to notice the bright white spot in the middle of these images.It represents the pin holding the insects onto the drawers & hence the closest point to the camera.



Finally we present a few reconstructed 3D models of the insects from different angles.

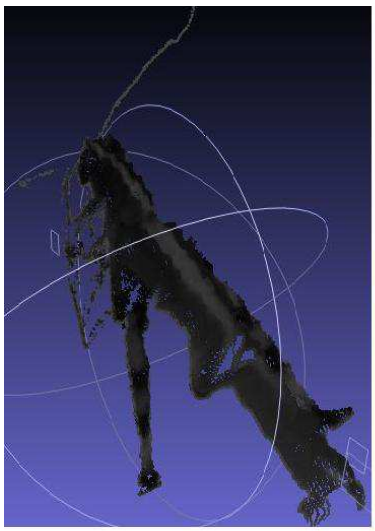
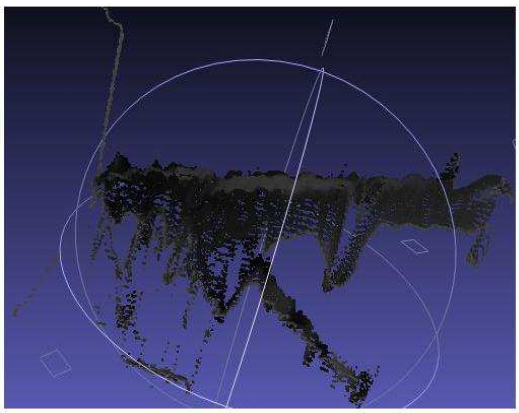
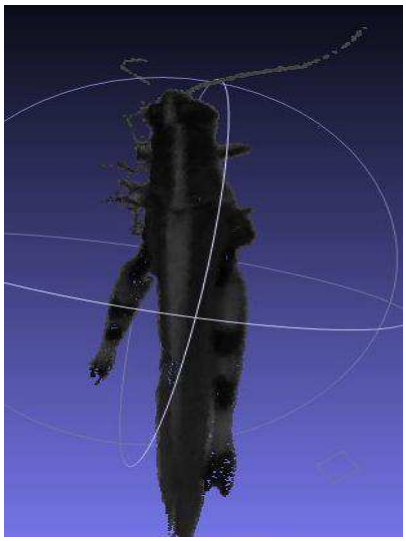






The screenshot above presents a reconstruction from different angles (back,top and side) [Specimen 1].The bump on the exoskeleton represents the pin which held the insect.Below is the reconstruction of another such specimen.[Specimen 2]



The following table shows the comparison in the dimensions of the original object and the reconstructed model.We first scale the model to the size of the length of the real object and then compare the breadth and the depth.

Table 6.1: Comparison Between Actual Object & Reconstructed Models

Face of Chief Illiniwek			
	Length	**Breadth**	**Depth**
Original	8 cm	6.3cm	3.2cm
Reconstructed	8 cm	6.0 cm	3.2cm
Specimen 1			
	Length	**Breadth**	**Depth**
Original	5.5 cm	2 cm	1.9 cm
Reconstructed	5.5 cm	2.1 cm	2.3 cm
Specimen 2			
	Length	**Breadth**	**Depth**
Original	3.6 cm	0.5 cm	1.6 cm
Reconstructed	3.6 cm	0.6 cm	1.8 cm

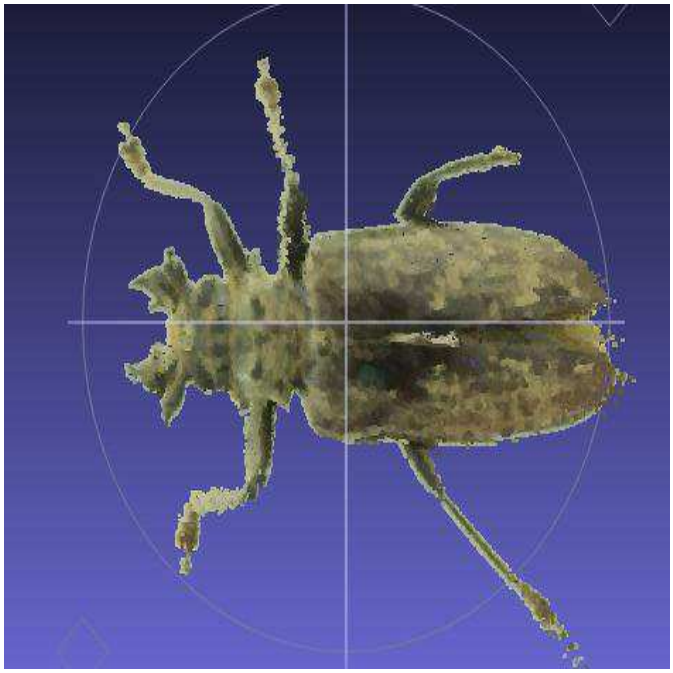
The serial version of the implementation(without refinement) on a 250 x 250 image takes about 23 seconds.The parallel version of the same implementation takes less than 10 seconds. Hence we get a speed up of approximately 2.5.

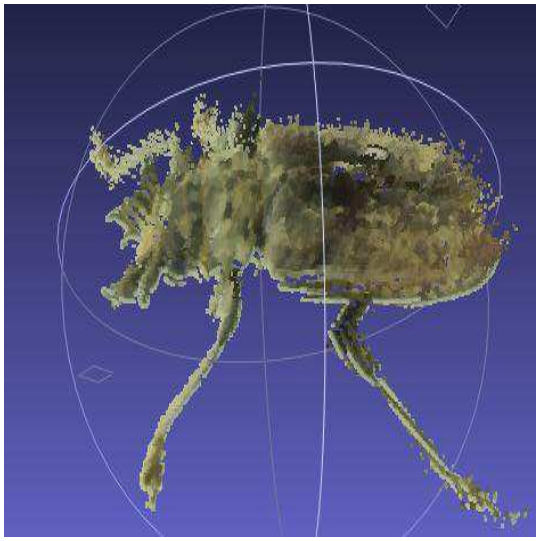
CHAPTER 7

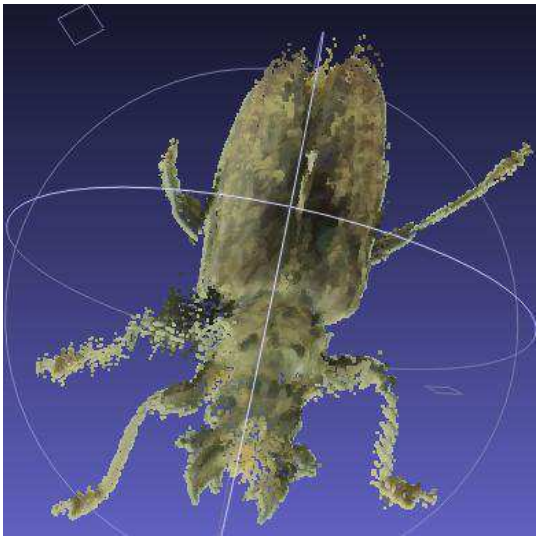
CONCLUSION

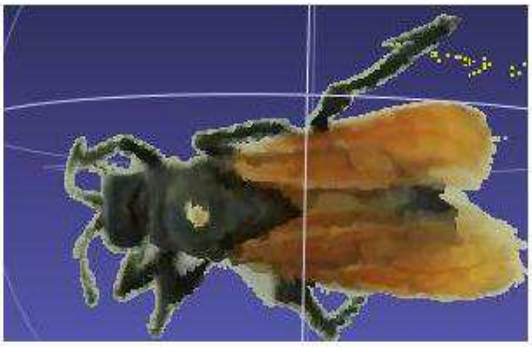
We present in this thesis a method to produce novel 3D reconstructions of insects to help Entomologists in their research.We also perform some other facial reconstructions.Our algorithm is robust in terms of producing good models and fast by virtue of parallelization.

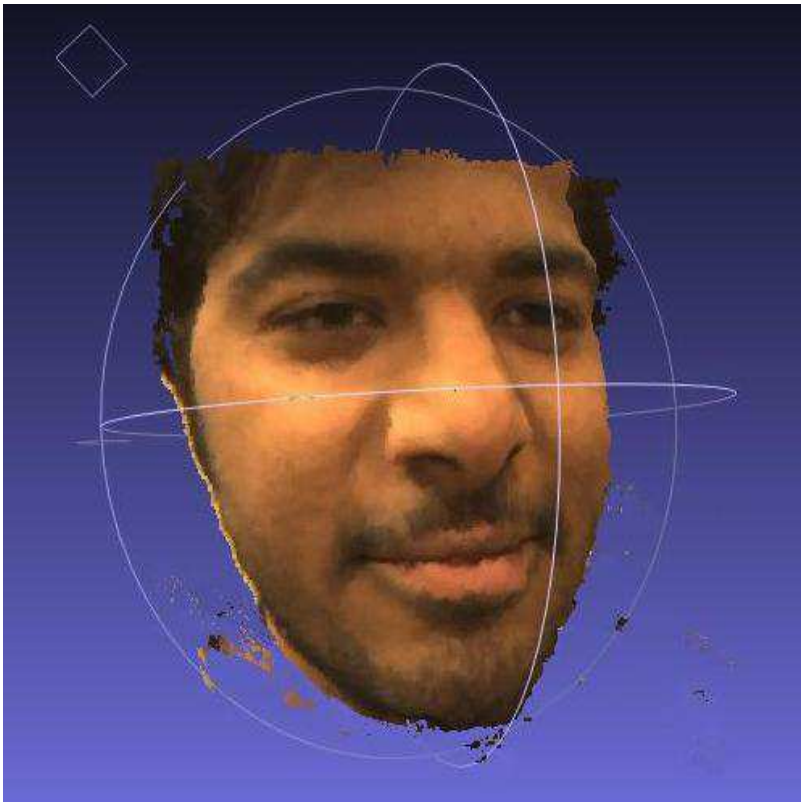
Future work involves exploring more possibilities for parallelism and obtaining speedup such that the entire process can be made real time.There is also scope for improvement by reducing noise in some of the reconstructed models.Finally we intend to perform surface reconstruction on the point cloud to generate a 3D surface.











Yeah!That's Me

REFERENCES

[1] "Stereo image pairs." [Online]. Available: http://alfa.magia.it/Images/ExamplesDirectStereo.htm

[2] Wikipedia, "Stereopsis," 2011. [Online]. Available: http://en.wikipedia.org/wiki/Stereopsis

[3] T. Magazine, "All time 100 greatest toys," 2011. [Online]. Available: http://www.time.com/time/specials/packages/completelist/0,29569,2049243,00.html

[4] L. Matthies, B. Chen, and J. Petrescu, "Stereo vision, residual image processing and mars rover localization," in *Image Processing, 1997. Proceedings., International Conference on*, vol. 3, oct 1997, pp. 248 –251 vol.3.

[5] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996. [Online]. Available: http://doi.acm.org/10.1145/237170.237191 pp. 11–20.

[6] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts." in *ECCV (3)'02*, 2002, pp. 82–96.

[7] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1186822.1073232 pp. 577–584.

[8] Y. Li, H.-Y. Shum, C.-K. Tang, and R. Szeliski, "Stereo reconstruction from multiperspective panoramas," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 1, pp. 45 –62, jan. 2004.

[9] H.-Y. Shum and R. Szeliski, "Stereo reconstruction from multiperspective panoramas," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 14 –21 vol.1.

[10] D. Scharstein and R. Szeliski, "Stereo matching with non-linear diffusion," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, jun 1996, pp. 343 –350.

[11] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," in *Computer Vision, 1998. Sixth International Conference on*, jan 1998, pp. 517 –524.

[12] R. Szeliski, "Scene reconstruction from multiple cameras," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1, 2000, pp. 13 –16 vol.1.

[13] V. Vaish, M. Levoy, R. Szeliski, C. Zitnick, and S. B. Kang, "Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2331 – 2338.

[14] M. Maitre, Y. Shinagawa, and M. Do, "Symmetric multi-view stereo reconstruction from planar camera arrays," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[15] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *Computer Vision, 2009 IEEE 12th International Conference on*, 29 2009-oct. 2 2009, pp. 80 –87.

[16] P. Mordohai, J. m. Frahm, A. Akbarzadeh, C. Engels, D. Gallup, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewnius, H. Towles, G. Welch, R. Yang, M. Pollefeys, and D. Nistr, "Real-time video-based reconstruction of urban environments," in *Proceedings of 3DARCH: 3D Virtual Reconstruction and Visualization of Complex Architectures*, 2007.

[17] H. Sunyoto, W. van der Mark, and D. Gavrila, "A comparative study of fast dense stereo vision algorithms," in *Intelligent Vehicles Symposium, 2004 IEEE*, june 2004, pp. 319 – 324.

[18] Scharstein and Szeliski, "Middlebury dataset." [Online]. Available: http://vision.middlebury.edu/stereo/

[19] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 29, no. 3, 2010.

[20] "Stereoscopic face images matching." [Online]. Available: http://www.cg.tuwien.ac.at/hostings/cescg/CESCG-2008/papers/BrnoBUT-Klaudiny-Martin/index.html

[21] "Stereo face." [Online]. Available: http://cvlab.epfl.ch/data/stereoface/

[22] W. Zhao and N. Nandhakumar, "Effects of camera alignment errors on stereoscopic depth estimates," *Pattern Recognition*, vol. 29, no. 12, pp. 2115 – 2126, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320396000519

[23] J. Ren, "Algorithm for 3d reconstruction of agriculture field pests based on binocular stereo vision," in *World Automation Congress (WAC), 2010*, sept. 2010, pp. 101 –105.

[24] J.-Y. Bouguet, "Camera calibration toolbox." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[25] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *in ICCV*, 1999, pp. 666–673.

[26] Intel, "Tbb documentation." [Online]. Available: http://threadingbuildingblocks.org/documentation.php

[27] Wikipedia, "Iterative closest point algorithm." [Online]. Available: http://en.wikipedia.org/wiki/Iterative_Closest_Point

[28] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, pp. 7–42, April 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=598429.598475

[29] D. Scharstein, "High-accuracy stereo depth maps using structured light," 2003, pp. 195–202.

[30] H. Hirschmller, "Evaluation of cost functions for stereo matching," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.