LEVEL-BASED CORRESPONDENCE APPROACH TO COMPUTATIONAL STEREO

SEYED ALI KASAEIZADEH MAHABADI

MASTER OF SCIENCE
COMPUTER INFORMATION SCIENCE

UNIVERSITI TEKNOLOGI PETRONAS

August 2010

STATUS OF THESIS

Title of thesis	LEVEL-BASED CORRESPONDENCE APPROACH TO COMPUTATIONAL STEREO

I am SAYED ALI KASAEIZADEH hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

   [ ] Confidential

   [X] Non-confidential

If this thesis is confidential, please state the reason:

_____
_____
_____

The contents of the thesis will remain confidential for _____ years.

Remarks on disclosure:

_____
_____
_____

Endorsed by

_____ _____

Signature of Author | Signature of Supervisor

Permanent address: 6, 226.2
Naji St, Shahid Dastgerdi Av.
19186, Tehran, Iran

Name of Supervisor
_____

Date : 30 August 2010 | Date : _____

UNIVERSITI TEKNOLOGI PETRONAS

LEVEL-BASED CORRESPONDENCE APPROACH TO COMPUTATIONAL STEREO

By

SEYED ALI KASAEIZADEH MAHABADI

The undersigned certify that they have read, and recommend to the Postgraduate Studies Programme for the acceptance of this thesis as a fulfillment of the requirements for the degree stated.

Signature: _____

Main Supervisor: Abas Bin Md Said

Signature: _____

Co-Supervisor: _____

Signature: _____

Head of Department: Mohd Fadzil Bin Hassan

Date: _____

LEVEL-BASED CORRESPONDENCE APPROACH TO COMPUTATIONAL STEREO

By

SEYED ALI KASAEIZADEH MAHABADI

A Thesis

Submitted to the Postgraduate Studies Programme

As a Requirement for the Degree of

MASTER OF SCIENCE

COMPUTER INFORMATION SCIENCE

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR,

PERAK

AUGUST 2010

DECLARATION OF THESIS

Title of thesis | LEVEL-BASED CORRESPONDENCE APPROACH TO COMPUTATIONAL STEREO

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by

_____ _____

Signature of Author

Permanent address:6, 226.2

Naji st, Shahid Dastgerdi av.

19186, Tehran, Iran

Date: 30 August 2010

Signature of Supervisor

Name of Supervisor

Abas Bin Md Said

Date: _____

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to Universiti Teknologi PETRONAS for supporting this work. Secondly, I would like to thank my supervisor Assoc. Prof. Dr. Abas Md Said for being as generous and understanding person. Without his helpful guidance, advice and motivations, I might have not been able to complete this project with the required quality.

During the course of this research, the author has benefited from the knowledge and advice of several individuals. I sincerely appreciate the help from Dr. Mohamed Nordin Zakaria. Next, I would like to express my gratitude to my dear family, my supporting parents who have helped me to grow as an independent person and guided me throughout my life. Last but not least, I would like to express my thanks to my friends who have truly proven their friendship and support throughout the entire process by motivating me, raising me up when I got down and being there for me when I needed a hand.

And to all those people who have shared their experience and ideas which I have forgotten to mention in this small but gratified acknowledgment,

Thank you.

ABSTRACT

One fundamental problem in computational stereo reconstruction is correspondence. Correspondence is the method of detecting the real world object reflections in two camera views. This research focuses on correspondence, proposing an algorithm to improve such detection for low quality cameras (webcams) while trying to achieve real-time image processing.

Correspondence plays an important role in computational stereo reconstruction and it has a vast spectrum of applicability. This method is useful in other areas such as structure from motion reconstruction, object detection, tracking in robot vision and virtual reality. Due to its importance, a correspondence method needs to be accurate enough to meet the requirement of such fields but it should be less costly and easy to use and configure, to be accessible by everyone.

By comparing current local correspondence method and discussing their weakness and strength, this research tries to enhance an algorithm to improve previous works to achieve fast detection, less costly and acceptable accuracy to meet the requirement of reconstruction. In this research, the correspondence is divided into four stages. Two stages of preprocessing which are noise reduction and edge detection have been compared with respect to different methods available. In the next stage, the feature detection process is introduced and discussed focusing on possible solutions to reduce errors created by system or problem occurring in the scene such as occlusion. Lastly, in the final stage it elaborates different methods of displaying reconstructed result.

Different sets of data are processed based on the steps involved in correspondence and the results are discussed and compared in detail. The finding shows how this system can achieve high speed and acceptable outcome despite of poor quality input. As a conclusion, some possible improvements are proposed based on ultimate outcome.

ABSTRAK

Satu masalah utama dalam rekonstruksi pengkomputeran stereo adalah korespondensi. Korespondensi adalah satu kaedah untuk mengesan pantulan objek di dunia nyata dalam dua pandangan kamera. Fokus kajian ini adalah mengenai korespondensi di mana kajian ini turut mencadangkan algoritma untuk memperbaiki pengesanan seperti ini untuk kamera yang berkualiti rendah (webcam) dan dalam masa yang sama cuba untuk mencapai pemprosesan imej nyata.

Korespondensi memainkan peranan penting dalam rekonstruksi pengkomputeran stereo dan ia mempunyai spektrum pelaksanaan yang luas. Kaedah ini berguna dalam bidang lain seperti struktur daripada rekonstruksi pergerakan, pengesanan objek, pengesan jejak dalam penglihatan robot dan realiti maya. Disebabkan kepentingannya, sesuatu kaedah korespondensi mestilah tepat demi memenuhi keperluan dalam bidang itu. Walau bagaimanapun, ia haruslah tidak menelan belanja yang lebih dan pada masa yang sama mudah untuk diselenggara serta digunakan supaya semua orang dapat menggunakannya.

Dengan membandingkan kaedah korespondensi yang ada sekarang serta berbincang mengenai kelemahan dan kekuatannya, kajian ini cuba untuk mengetengahkan satu algoritma untuk memperbaiki kajian yang terdahulu dalam mencapai pengesanan yang cepat, harga yang lebih rendah dan tahap ketepatan yang boleh diterima untuk memenuhi keperluan rekonstruksi. Dalam kajian ini, korespondensi ini dibahagikan kepada empat peringkat. Dua peringkat pra-pemprosesan iaitu pengurangan bunyi hingar yang tidak diperlukan dan pengesanan sisi telah dibandingkan dengan menggunakan kaedah-kaedah berbeza yang sedia ada. Dalam peringkat yang seterusnya, proses pengesanan ciri-ciri diperkenalkan dan dibincangkan di mana ia berfokus kepada beberapa penyelesaian untuk mengurangkan masalah seperti kesalahan yang timbul daripada sistem itu sendiri atau masalah seperti kesesakan dan tidak lancar.

COPYRIGHT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background of Study

1.1.1 Introduction

Computational Stereo is the study of depth perception using multiple images (Figure 1.1). Consider P as a real-world object, p_R and p_L represent object reflection on the right and left image respectively. This process uses this information on processing and understanding P attributes, which will be discussed further in the next chapter.

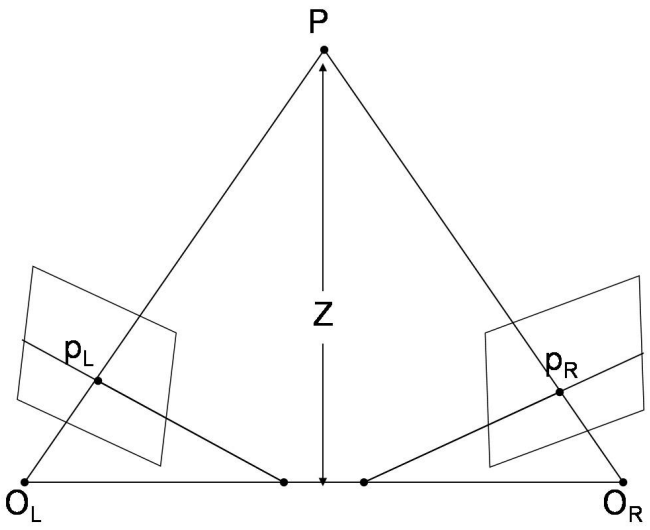


Figure 1.1: Two images captured from the same scene.

In computer vision, structure from motion refers to the process of building a 3D model from the video of a moving rigid object. Algorithmically, this is very similar to stereo vision where a 3D model is built from two simultaneous images of the same object. In both cases, multiple images are taken of the same object and corresponding features are used to compute 3D locations. In structure from motion, the images are

taken at different points in time comparing to stereo vision where images are taken at different points in space. Generally, structure from motion is sometimes used for any 3D reconstruction built from 2D images of a rigid (or static) object. Hence, because of this colloquial usage, structure from motion has significant overlap with stereo vision.

1.1.2 Application of 3D Reconstruction

3D reconstruction and animation are key elements for applications in medicine [1], games, virtual reality systems [2], [3] and robotics [4]. However, developing a 3D object generally requires a lot of time and effort to accomplish a perfect model which is as similar as possible to the real-world.

In the early stage of producing a 3D multimedia application, it often took months of hard work; since everything started from scratch. Soon, the multimedia developers started to store 3D object libraries for future use. However, the process is one of the most time-consuming because it requires huge resources (for example human labor, hardware and etc.). Conversely, in 1980s, the 3D library generation became easier and faster through the introduction of 3D scanners and system which can reconstruct 3D object form the reality in shorter time resources [5]. On the contrary, the result from these methods was not satisfactory due to high cost of equipment and low quality of result. This has urged researchers in the field to look for better algorithms.

For virtual reality, one of the most important elements is user interaction (tracking). Unfortunately, sensors used for tracking user's motion are either too heavy or low quality (Figure 1.2, Figure 1.3). These sensors have some problems such as drifting, latency and jitter. In stereo systems, since trackers are not connected to the user(s) and control the scene from distance, the tracking process becomes more accurate and more convenient.

In robotics, to give ability for a robot(s) to travel normally it requires some path lines for the robot(s) to follow and also with the help of several sensors surrounded the robot(s) it can recognize the robot(s) collision. With the help of computational stereo system, we can give robot(s) the ability to know the exact location of obstacles and to prevent collision. Moreover, instead of tracking lines to reach the target, the

implementation of computational stereo system aids the robot(s) to find the best path through camera installed on robots.

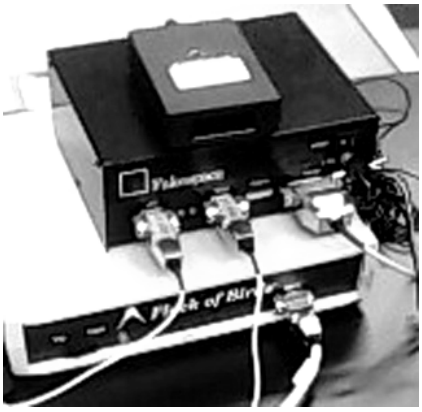


Figure 1.2: Sample of tracking device for virtual environment

Figure 1.3: Data glove

1.1.3 Evolution of Computational Stereo

Advanced Research Projects Agency (ARPA) funded a research on computational stereo in 1970s and early 1980s. Since then, it has been decades for these researches to focus on different areas on 3D scene structure. ARPA primarily focused on image understanding (IU). Barnard and Fischer reviewed research work on computational stereo [6] until 1981 by introducing well-known approaches on fundamental of stereo reconstruction and criteria for evaluating performance.

Dhond and Aggarwal [7] focused on improvement of stereo research through 1980s. Their research described different approaches on correspondence problem, grouping methods to local and global and the use of trinocular constraints to reduce ambiguity in result. Even though the research on stereo continued, but in early 1990s stereo research turned to emphasis on more specific problem. Chung and Nevatia [8] focused on occlusion problem and grouped them in three different areas. Koscha [9] in his report discussed the basic focused on stereo vision since 1989, including early research on occlusion and transparency, Area-Based and Feature-Based stereo, and implementations of real-time stereo. Substantial progress in each of these lines of research has been made in the last decade and new trends have emerged. Although some general stereo matching (Correspondence) research continued, much of the community's focus turned to more specific problems. Scharstein and Szeliski [10] evaluates most of global matching algorithm developed by 2001. They use two statistical approaches for evaluation. Root-Mean-Squared (RMS) as a comparison between ground truth disparity and the disparity generated by selected method and percentage of bad matching pixel outside of tolerance disparity error (Figure 1.4). Based on this research a website [11] designed with the mean of comparing new stereo matching approaches.

(a) Ground truth (b) Disparity generated (c) Bad Pixels (disparity error >1)

Figure 1.4: Scharstein and Szeliski example of Bad Pixels generated with tolerance disparity error of 1.

1.2 Problem Statement

3D reconstruction of real world objects can be useful in many different areas. In virtual reality providing more realistic object or structure can improve the immerse process. The same effect can be revealed in computer graphic area. Autonomous

robots in robotic area require understanding of 3D surrounding to improve performance.

Correspondence is considered as the main issue in stereo system and 3D reconstruction (which will be discussed in section 2.4 and 2.5). Due to this importance, following research focuses on solving the correspondence problem. This research tries to match pixels in pair of images which represent the same point in real world. The proposed method tries to achieve real-time process while maintaining acceptable result (achieve more than 90% correct match).

Through past decades many methods were introduced to solve correspondence issue. These methods can be divided into two groups. Local methods focused on the small group of pixels which increase the processing time but this cause the methods to be more sensitive to partial occlusion. Global methods focus more on scan line and this cause to be more effective to issues such as occlusion but the methods require higher processing power and processing time. Due to this, in our research the main focus is on local methods which try to resolve some of their issues (will discuss in more detail in chapter 2, correspondence section), proposing a combine algorithm which uses the strength of local method to achieve better performance for real-time system.

1.3 Objective and Scope of Study

3D modeling has significant impact on virtual realities, game development, simulations environment and many other graphical fields. There are many methods to develop 3D models and model databases but current methods have many issues. These issues mainly consist of sensitivity to intensity changes between images, depth discontinuity and pixels self similarity which they will be discussed in detail in section 2.5.1 and 2.5.2. By improving the correspondence method it is possible to resolve many of these problems. In this research the main objectives are as follow:

- To detect objects in a pair of images based on contours and edges for increasing immunity to changes such as translation and scene lighting.
- To match detected objects in both images with minimum 90% correct match and increasing result immunity to image quality.

- To reduce matching processing time for generation of real-time disparity map (maximum 50 ms processing time with available processing power)
- To transfer disparity map result to real-world dimension and increase precision up to one millimeter.

Scope of this research is as follow:

- The generated 3D model will be generated based on only two captured images.
- The images captured from the scene required to be non-verged
- Generate 3D model with images based on the camera(s) property is available
- For images with no available camera property only disparity map will be generated
- Matching algorithm uses local method.

The main contribution of this approach is by introducing level and grouping the contours and edges in images based on hierarchal model, called level. Edges and contours provides enormous opportunities such as immunity to intensity changes, limit the search area. Edges are more detectable in any images regardless of image quality. Thus it is expected from the proposed method to find correspondence more accurate in shorter time.

1.4 Thesis Outline

In this chapter the importance of correspondence is discussed. Correspondence plays an important role in many areas such as robotics, virtual realities, computer graphics etc. There are numerous problems such as speed, accuracy and occlusion. Align to these problems many solutions were proposed such as sensors which have their own problems. These problems also consist of speed, accuracy and the effect of external objects over the result. Some sensors can also be considered troublesome for users in

terms of their weight and effects which cause the users not to immerse in virtual environment.

Problems involved in current matching algorithm were also discussed. Local methods provide more promising result in term of speed, cost and ease of use. These algorithms provide possibility of real-time matching in images. Based on this, the main objective of this project is to study current matching processes and propose improvement in current methods. This research focuses only on two images and tries to generate third dimension sense either with disparity map or a regeneration of 3D environment in case of available camera properties.

The second chapter will focus on previous work on 3D reconstruction with further emphasis on local matching methods. Different steps involving 3D reconstruction such as camera calibration (extracting intrinsic and extrinsic camera geometry), non-verged stereoscopic images, and depth in stereo system and finally local matching algorithm will be discussed.

Chapter 3 will focus on improvement on local methods. Level-based matching and steps involved in this approach will be discussed. Level-based approach provides unique techniques to reduce the matching speed while keeping acceptance of correct match rate. In this approach the main focus is on edge property providing distinct features for matching process. It also reduced the necessity of image rectification and processing each and every pixel for the matching.

Finally in chapter 4 the result generated from the proposed method will be discussed. From this discussion, different local methods will be compared with the proposed methods. Finally few 3D reconstruction samples will be provided. Issues on 3D reconstruction from generated disparity map will be discussed.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

3D reconstruction is one of the key elements for 3D animation, games or especially in virtual reality systems. Most of current approaches in computational stereo involve in global methods. Despite great improvement in term of algorithms and improvements in processing power, these groups of matching algorithm fail to perform matching in real-time [12, 13]. This issue leads to failing in reconstruction of 3D object in real-time. To develop 3D object it requires lots of time and effort to create most realistic object. In previous time there were many algorithms developed. These algorithms can be specified into 2 groups. The first algorithms only require single image to generate 3D view. These algorithms started in the early 1970's and they were mostly relying on the intensity. They were used for geographical perspectives and long distance images such as satellite imagery. Around 1980's to 1990 some new methods were developed for multiple image systems [14, 15]. Images normally captured by more than one camera at the same time such as stereopsis computational stereo systems (two cameras) or single camera in different position or from moving object (structure from motion).

This chapter emphasis more on local matching methods which has faster performance and in some cases perform matching process in real-time [16]. This chapter starts with basic steps involve in image processing as general. Focus here is on preprocessing steps, such as noise reduction and edge detection. The following sections after preprocessing will be a review of steps involve in 3D reconstruction. Final section of this chapter will discuss on correspondence algorithms in detail.

2.2 Preprocessing

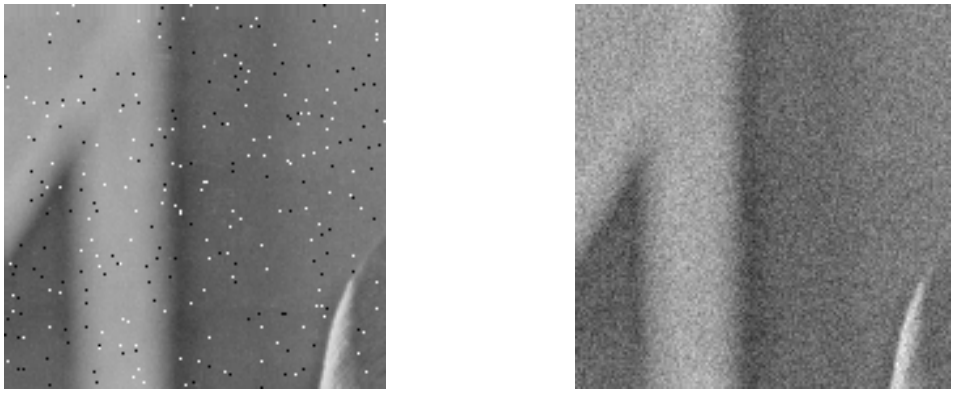
Preprocessing is a step in computer vision where the low-level operations will be applied on the image usually to reduce noise (detaching noise from signal), selecting region of interest (ROI) as a general image enhancement. There are many processes can be applied in this section where they can mostly be considered as sub-sampling the image, applying digital filters and edge detection on the image. In this section will discuss different methods on preprocessing. Discussion starts with noise reduction method followed by edge detection and feature extraction.

2.2.1 Noise Reduction Process

The purpose of this process is to remove noise from signal which could vary based on the type of signals. In computer vision, usually the source of signal is charge-coupled device (CCD) which is considered as digital signal. In case the signal is analog, it needs to be converted to digital signal.

In image processing system, there are mostly two types of noise. Salt and pepper noise are usually due to faulty CCD elements. The faulty image contains dim and white pixel, therefore they are called salt and pepper noise (Figure 2.1.a). The characteristic of this noise is that noisy pixel is not related to the surrounding pixels. This type of noise normally affects small amount of pixels but normally distributed on all image, as well as the position of noise is random even if all the images are taken from same camera.

The second noise type is Gaussian noise. Gaussian noise is random distribution of artifacts, where the main causes change the original value of pixel by small amount which may be due to digitization or faulty CCD elements (Figure 2.1.b). This type of noise may cause the image look blurry or soft; by zooming on the image one may notice tiny specks in random pattern. A plot of the amount of pixel value against the frequency, with which it arises, shows a normal distribution of noise.



(a) Salt and Pepper (b) Gaussian

Figure 2.1: Portion of Lena's image, with noise

There are many different algorithms to remove these noises from the image. Each of them has some advantages and disadvantages. Computational power, time and noise to data removal aspect ratio are a few tradeoffs that need to be considered before considering a method for noise reduction.

Linear smoothing filter (Figure 2.2.a) is one approach to reduce noise in an image. In this method [17] a low-pass filter will be selected for convolution. This method will bring value of each pixel for closer accord with the pixel neighbors by averaging values of pixel and its neighbors. This method may cause the image blur as it causes any pixel with significant difference with its neighbors, smear across the area. Due to this, linear filters are rarely used in practice for noise reduction.

Anisotropic diffusion (Figure 2.2.b) is another approach to noise reduction which is aimed to keep significant parts of the image content such as edges and other details while reducing noise in the image [18]. In this method image generates parameterized group by blurring the image more and more based on diffusion process. Once this process is done then by convolving between the image and 2D isotropic Gaussian filter where the filter width will increase based on the parameters. Even though this process tends to keep the image content but the process requires large computational power and it requires more time.

A nonlinear filter (Figure 2.2.c) is also another effective method in noise reduction[19]. Median filter can be considered as one example of this approach and it can conserve the image detail more efficient. In this method the pixel neighbor will be sorted based on the intensity in order, and then the median value will replace the value

of the selected pixel. This method mostly used for removing salt and pepper noise from image but it also causes relative blurring of the edges.

Opening operation [20] (Figure 2.2.d) which is part of Mathematical morphology has two steps. Erosion, as the first step to remove any brighter pixels that does not match with its neighbors, based on structural element defined. The second step is Dilate, which removes darker pixel that does not match with its neighbors. The combination of these 2 steps causes the noises, especially salt and pepper noise removed from the image. In the first step the salt noise will be removed and the resulting image will be darker and somehow this step blurs the image compared to original image. The second step causes the image to return to its original state while the pepper noises are removed and the image becomes brighter and at the same time sharper.

Figure 2.2: Lena's image (a) Linear filter (b) Anisotropic diffusion (c) Nonlinear filter (d) Opening operation

2.2.2 Edge Detection

Edge detection is another step in preprocessing, which is the basic step for feature extraction. The focus on the edge detection is mostly on areas that have sudden changes in the pixel intensity or in other words, areas that have discontinuity in brightness of the image. The purpose of this process is to capture important events and changes which can be considered as a property of a specific object. For example in letter recognition, “A” has a triangular shape hole whereas “B” has two circular holes, which is the way to separate these 2 letters from each other.

Generally edges can be caused by different changes such as changes in depth in one object or between two objects, changes in material or texture or as general changes in scene lighting. Based on this, the ideal situation is where the edge detection result shows the boundaries of object with connected curves. Another advantage of this process is reduction on the amount of data to be processed on the image. Unfortunately, there are many situations in real life that the edge detection process returns faulty results. These faulty results could be due to the scene lighting which causes the edges to be discontinued or fragmented. It can also cause missing edges as well as false edges not corresponding to the fact which can cause implication on the feature extraction process.

In this section, it compares different edge detection methods. Based on this discussion a method will be chosen as a default edge detection algorithm. The selected method should detect most accurate edges in the image which require minimum user input. As for discussion, the first two common methods will be discussed. These methods are known as Sobel operation and Robert Cross. The discussion will be continued with Canny’s operation (more sophisticated method). Finally, by introducing threshold method as replacement for edge detection algorithm will conclude the discussion.

The Sobel operator [21] is basically a convolution method. In this method there are two 3x3 kernels (2.1) which filter image on horizontal (using C_x) and vertical (using C_y) direction. Therefore this method is relatively computationally inexpensive. Sobel operation in other word is a discrete differentiation operator. On the other hand, the

gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image (Figure 2.3.a).

$$C_x = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A, \quad C_y = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \qquad \textbf{(2.1)}$$

Where * here denotes the 2-dimensional convolution operation and A is image.

Sobel operator is an estimation of the image's brightness gradual blend at each pixel. In this method, it gives the direction of most possible change in intensity and change in that direction. Sobel operation result shows how quick the image can be changed in specific point. This operation also shows the likelihood of an edge and its orientation in image.

In edge detection many algorithms use similar method as Sobel operation. One of the early works is Roberts Cross [21] where it calculates the sum of squares difference between diagonally adjacent pixels. In this method, same as Sobel there are two kernels (2.2), but due to small size of the kernels, this algorithm is faster (Figure 2.3.b).

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \text{ And } \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \qquad \textbf{(2.2)}$$

Canny [21] edge detection is another method which involves in multiple-stage algorithm. In this method it tries to reach to optimal edge detected in the image. As explained before, the optimal edge detection plays important role in feature extraction step. This means the detected edges should cover all the real edges in the image. Detected edges should be as close as possible to real edge location in the image. Result of edge detection process for a given edge should mark only one time in the image. Edge detection process should avoid considering noise as edge (Figure 2.3.c). Canny proposed following steps to achieve such goal.

- Noise reduction
- Finding the intensity gradient of the image
- Non-maximum suppression

One of the advantages of Canny edge detection is including noise reduction in this method. In this algorithm it uses the first derivative of a Gaussian. As discussed before the Gaussian method reduces the noise while the blur effect is minimal. The next step in Canny algorithm is to detect edges in four different direction as two diagonal edges; vertical and horizontal edges. As it was discussed before other edge detection process perform the edge detection in one or two directions. Canny improved his method proposed to use all four directions to detect all possible edges in the image. The next step is, to remove duplication and find the exact location of edge in image. Canny use the direction angle for edge detection to compare result for this purpose. In this step for example, in horizontal edge detection result, it uses the edges with highest value in north-south direction meanwhile for vertical direction it uses the west-east edges with high intensity. This process can be performed by passing [3x3] grid over intensity map.

Adaptive threshold [22] also known as dynamic threshold is a method which is similar to any other threshold systems, that converts any image to binary image based on the threshold value. In this method any pixel intensity below the threshold value will be considered as 0 and above that as 1. The difference in dynamic threshold approach with other threshold approaches is that in this method, different threshold is being used for different region in the image (Figure 2.3.d).

A simple approach to select the threshold is based on the region automatically by using the mean or median value to be selected as threshold. In this approach, if the selected region pixels are darker than the background, they should be darker than the average. In this approach also, an initial threshold (T) will be chosen. Then the image will be divided into two sections; object and background. The object pixels have the threshold above T and the rest of the pixels will be considered as background. For each section the average will be calculated and the new threshold value will be the average of the two regions average. The process of dividing the image into two sections until finding the new threshold value will continue until T become definite or being repeated. In this method starting value of T may affect the final T's value.

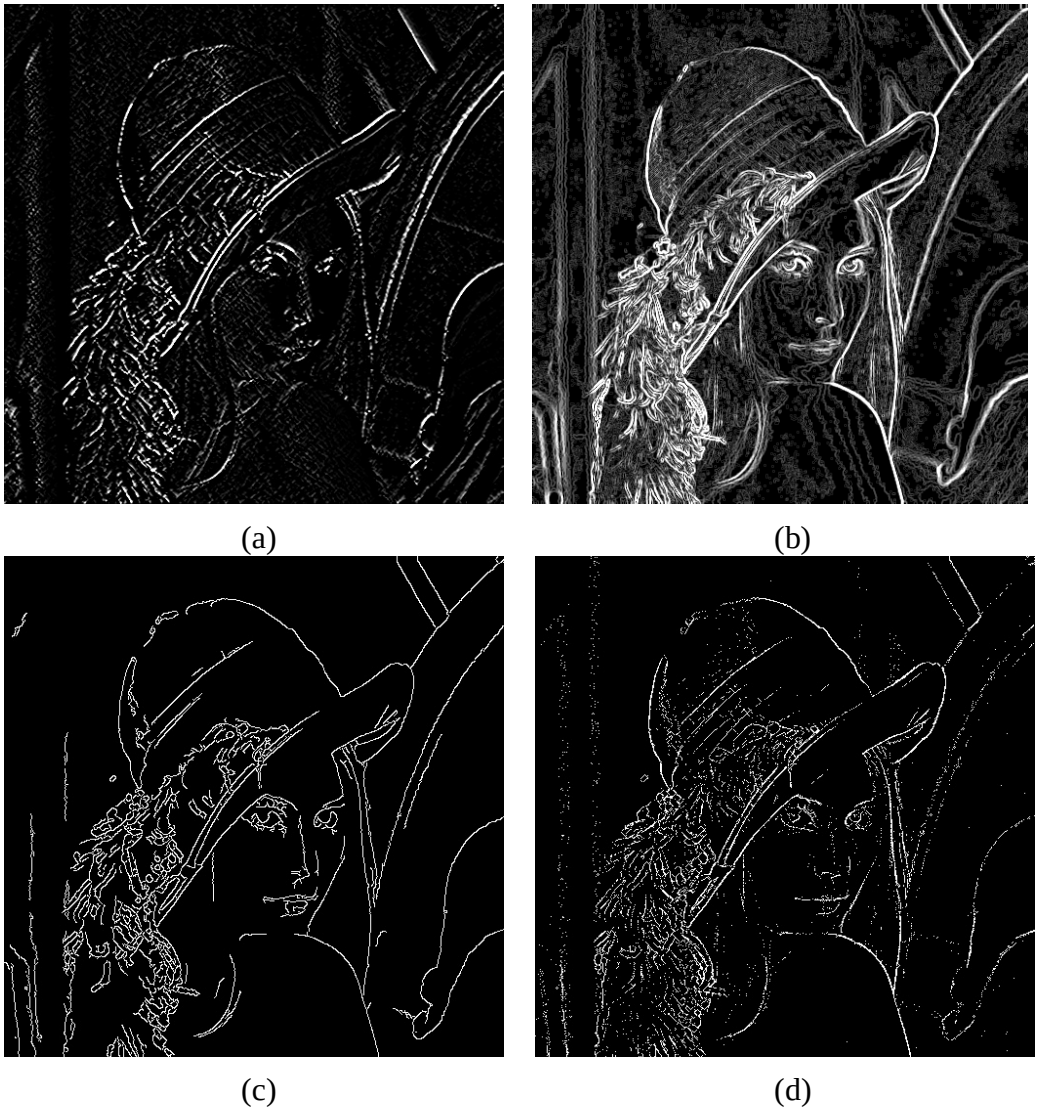


Figure 2.3: Lena's image, (a) Sobel Operation, (b) Robert Cross, (c) Canny Operation, (d) adaptive threshold

As a conclusion, in this section the two main steps of preprocessing are discussed. In each section, different algorithm are introduced and based on the criteria such as performance versus the computational power required for each method, it tries to find the best possible solution to solve the problem of noise reduction and edge detection. In this section, opening operation is selected due to advantages such as minimal affect on the image edge and also reducing the processing power required for this theory. For edge detection there were four common methods introduced where the dynamic threshold is a suggested approach as this method does not require kernel because it causes huge difference in computational power required for providing such process.

At the end after reducing the noise and detecting the correct possible edges in the image, the image is prepared for the next step, which is converting the edges in the image to the contours which will be discussed in the next section.

2.3 Feature Extraction

Matching pixels in two images are not practical. This process require great amount of time and due to similar pixels in image, it is not possible to match all pixels. For example, for one pixel in one image there may be many other possible pixels in the next image with similar neighbors in which it is possible that none of those pixels are actual correspondence of the specific pixel. By considering the situation, where in an image there is only one dark dot in the middle of white page. The possibility of finding the correspondence for each white pixel is not possible as its neighbors are all white pixels and there are many possible outcomes. Considering occlusion, pixels outside the second camera view point and change in light illumination on the time of capturing the second image are few cases that cause finding correspondence in image for each pixel which is not feasible.

Thus for the purpose of reducing the data processed and increase the accuracy of matched pixels, feature detection and feature extraction is necessary. Here the question that may rise up is that, what the feature is and what distinct a pixel from other pixels? Features are the types of data that are invariant to view point such as scale orientation of camera or changes in its position. The feature should be invariant to changes in light, shape of object or partial occlusion. Features should be fast for computation but at the same time, it should have enough time to get as much detail as possible and also be as unique as possible.

Based on this, the feature helps to detect as many objects with their correspondence as possible while the computational complexity reduced to ensure the result is more accurate due to invariant characteristic of feature to changes mentioned above. Based on the view point, light position and pixels are sensitive to changes in the object in real world or changes in camera. The most common methods for feature extraction is edge detection, corner detection, blob detection where those criteria are less sensitive to changes mentioned above and can be used in feature extraction, some of them were

discussed in the previous section. In this section the available feature extraction methods will be discussed

2.3.1 Corner Detection

One of the most common methods used in 3D modeling is corner detection. Corner in this section is defined as intersections of two edges which cause a corner to have two directional edges. A corner can also be defined as the end of a line (edge) or a point in high intensity change area. In this section, a corner will be more immune to changes such as noise, blur effect, changes in object position or camera position in terms of rotation or translation and other similar effects. In this section, a few of the most common corner detection algorithms will be discussed.

Moravec corner detection algorithm [23] was one of the early works on the corner detection. In this method the focus was on the point with low self-similarity. The similarity in this method is calculated by sum of squared difference (SSD) of the pixel itself on a 6x6 window in 4 directions. In this case, if the window moves in area with no edge, the changes in SSD will be smaller, where in the edge area along the edge direction, the changes will be smaller while against the edge, changes will be larger, and for the corners the change in any of 2 directions will be larger.

There were some issues on the method proposed by Moravec. Moravec methods can only detect small change. Another issue with his method was the detected corners were highly depended on the direction of kernel and edges on the image. This method also detects noises as corner. Harris and Stephen [24] (Figure 2.4) proposed a method to resolve problems in Moravec method. Anisotropic response due to limited direction was used in that method, where in Harris and Stephen method, the differential corner score only focused on edge direction. Moravec method uses binary and rectangular window which caused the output to be noisy. Harris and Stephen proposed Gaussian window which was smooth and circular. Another weakness in Moravec was focusing on the minimum of SSD where in proposed method by Harris and Stephen the SSD would be considered based on the direction of shift.

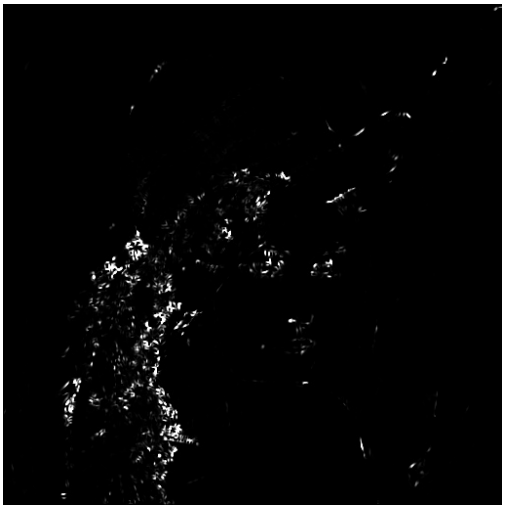


Figure 2.4: Lena's picture, Harris and Stephen Corner Detection method.

2.3.2 Blob Detection

Blob [25] detection similar to corner detection tries to find the points or areas that are less affected by change in external variable such as rotation of object or camera or intensity changes. In blob detection there are two main classes. The first class is differential methods focusing on the derivative expressions meanwhile the second class focuses on local extreme in intensity landscape. Even though blob detection can be used in interest point detection for stereo systems, this method has been used for other purposes such as object recognition, object tracking or texture analysis. There are many methods in the implementation blob detection, such as Laplacian of Gaussian (LoG) (Figure 2.5) and Difference of Gaussian (DoG) and Determinant of the Hessian (DoH) which discusses further on these methods.

One of the early works of the blob detection is Laplacian of Gaussian (LoG) [26] which is due to accuracy and comparatively less heavy computationally. This method is most common blob detection algorithm. In this approach image is convoluted by a Gaussian kernel and this operation was followed by a Laplacian operation. In this method the end result contains strong lines representing dark blob and the rest are bright blob which are similar in size.

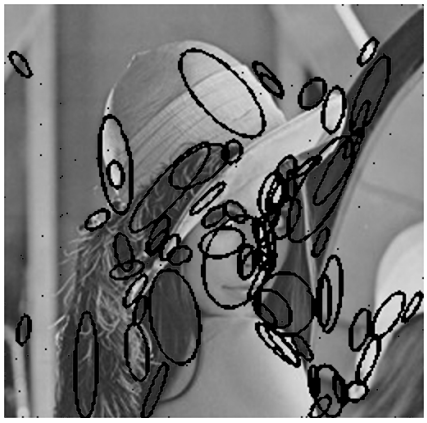


Figure 2.5: Lena's image on LoG operation

Difference of Gaussians is another algorithm similar to LoG, where the focus is in the difference of Gaussian blur. In this method original image will blurred 2 times and the blurred images will be subtracted from each other. The blurring algorithm used Gaussian function and so the method gets the name of difference Gaussian. There are many system relying on this method, such as object detection in scale-invariant feature transform (SIFT) [26] .

Determinant of the Hessian (DoH) (Figure 2.6) is another approach in blob detection. In this approach, firstly the Hessian matrix of Laplacian is calculated and then the maximum local will be selected. Furthermore, the detected blobs are minimally effective with rotation, translation and scaling. One of the famous applications of this algorithm is used as basic point selection in Speeded up Robust Features (SURF) [27] algorithm.

Figure 2.6: Lena's photo on Blob detection, using DoH Algorithm

2.4 Camera Calibration

Calibration is the process of determining camera system external geometry (Extrinsic properties such as the relative positions and orientations of each camera) and internal geometry (Intrinsic properties such as focal lengths, optical centers, and lens distortions). Accurate estimates of this geometry are necessary in order to relate image information (expressed in pixels) to an external world coordinate system. In next section, it highlights the importance of extrinsic elements such as camera distance from each other (Q) and internal geometry-focal lengths (F) as well as their effect on depth calculation. Other element such as lens distortions helps to improve images and resolve epipolar geometry issue. Camera calibration also provides camera matrices such as rotation matrix and translation matrix which help to convert depth (pixel) to depth in real-world measurement. Moreover, it also discusses about camera calibration for computational stereo and models used to calculate the intrinsic and extrinsic values for cameras.

The goal here is to calculate the camera matrices such as rotation matrix and translation matrix. Considering $[X^c, Y^c, Z^c]^{\mathrm{T}}$ as object known coordinate in camera and $[X^\omega, Y^\omega, Z^\omega]^{\mathrm{T}}$ as object known coordinate in real-world we have:

$$\begin{bmatrix} \mathrm{X}^{\mathrm{c}} \\ \mathrm{Y}^{\mathrm{c}} \\ \mathrm{Z}^{\mathrm{c}} \end{bmatrix} = \mathrm{R} \begin{bmatrix} \mathrm{X}^{\omega} \\ \mathrm{Y}^{\omega} \\ \mathrm{Z}^{\omega} \end{bmatrix} + \mathrm{T} \qquad \textbf{(2.3)}$$

Where R is rotation matrix and T is translation matrix. The main goal of camera calibration is to calculate these two matrices. In order to resolve such problem Zhang [28] extend (2.3) with a scale factor and describing a camera matrix. Zhang assumed that the position of object in image to be $m = [u, v]^{\mathrm{T}}$ and its representation in real-world as $M = [X, Y, Z]^{\mathrm{T}}$ and then augment the vectors by adding an element with value of one to them, where $\widetilde{m} = [u, v, 1]^{\mathrm{T}}$ and $\widetilde{M} = [X, Y, Z, 1]^{\mathrm{T}}$ as a result,

$$s\widetilde{m} = A[R \;\; t]\widetilde{M} \qquad \textbf{(2.4)}$$

Here s represents the arbitrary scale factor and A is the intrinsic camera matrix which is defined as:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.5)}$$

In (2.5), (u_0, v_0) represent the principal point, α and β as scale factor in image and γ senses of two image axis. Zhang solved (2.4) by using (Figure 2.9) pattern after detecting the corner of dark rectangles. He also presented a formula for camera distortion estimation using alteration and maximum likelihood estimation.

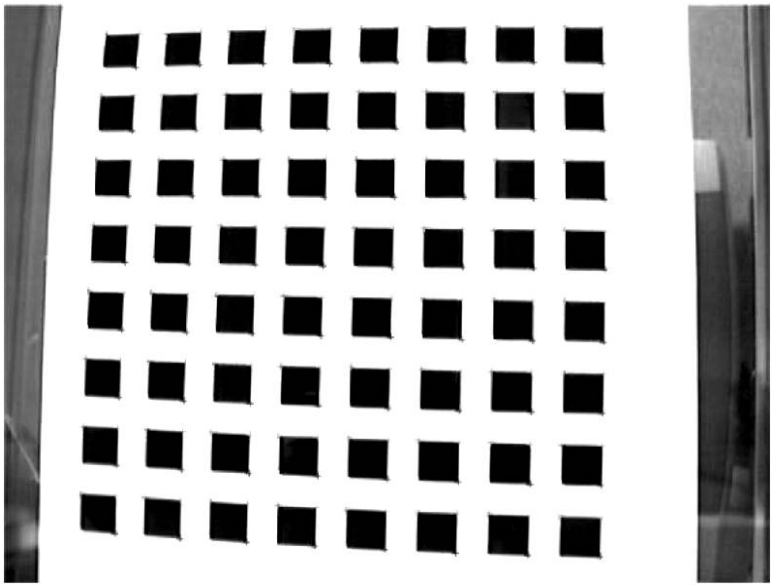


Figure 2.7: Zhang pattern on camera calibration [28]

Heikkila [29] tried different approach to this problem by presenting F as perspective transforming matrix. F is multiplication of two other matrices (PC). The first matrix P is:

$$P = \begin{bmatrix} sf & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In above equation s is considered as aspect ratio and f considered as focal length. Matrix C is:

$$C = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

The final equation is as follow.

$$\lambda \tilde{m} = PC\tilde{M} \quad \textbf{(2.6)}$$

Where λ here considered as scale factor.

The advantage of Heikkila proposed method is that in other methods the intrinsic and extrinsic matrices were combined together and the camera distortion was ignored where in his method these two were separated from each other. In his approach the reverse distortion model have been used.

2.5 Epipolar Geometry

In computational stereo most of local correspondence methods assume input images have non-verged geometry, thus relies on the scan-line to solve correspondence problem. Non-verged geometry means that images fall in same plane and correspondence point in images are parallel to y-axis. Unfortunately in practice, it is difficult to build stereo system with non-verged geometry. This problem can be resolved by relative orientation. Relative Orientation is the recovery of the position and orientation of one imaging system relative to another. Berthold K.P. Horn [30] proves that base of the relative orientation can be done by minimum 5 ray pairs of correspondence.

Figure 2.8 is example of such modification. The original images are shown with solid line, which are rectified by using relative orientation to non-verged images (dashed line) collinear scan-lines.

In order to resolve verged geometry problem, Hartley [31] proposed the use of eight-point algorithm for finding epipolar lines. The algorithm proposed minimum eight correct matches in images are sufficient for rectification (Figure 2.9). Later Mariottini and Prattichizzo [32] considered the use of contours to determine more accurate epipolar geometry estimation. This method was more effective on moving camera specially designed for autonomous robots.

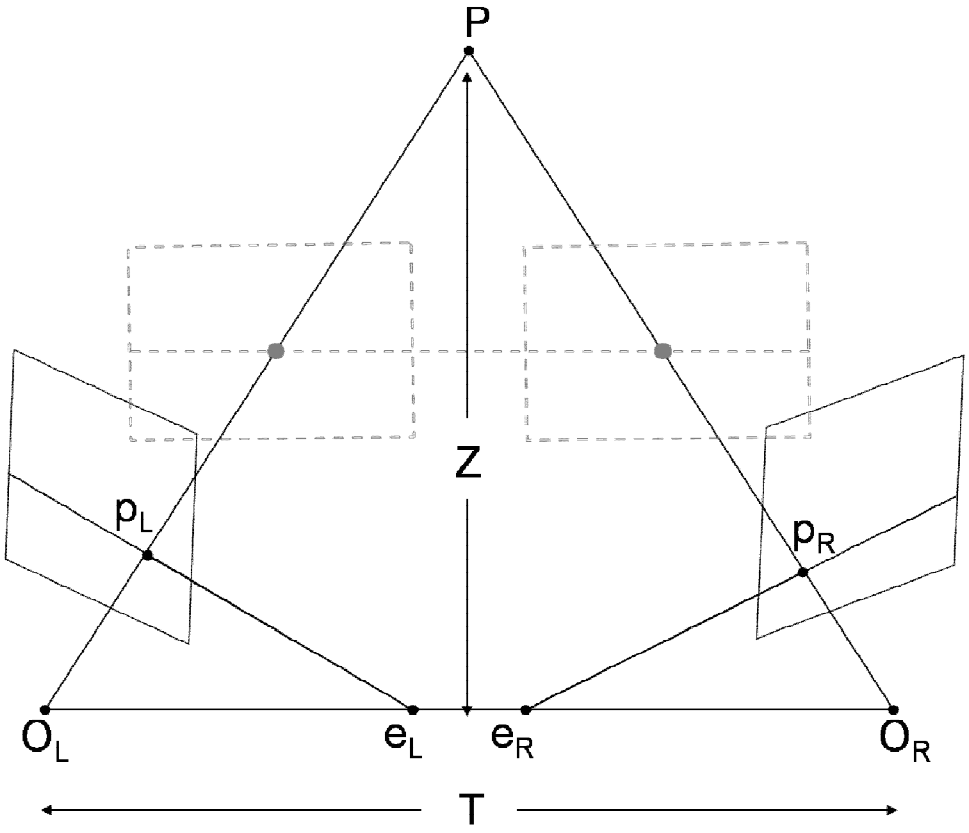


Figure 2.8: Two arbitrary images of the same scene may be rectified along epipolar lines (solid) to produce collinear scan lines (dashed).

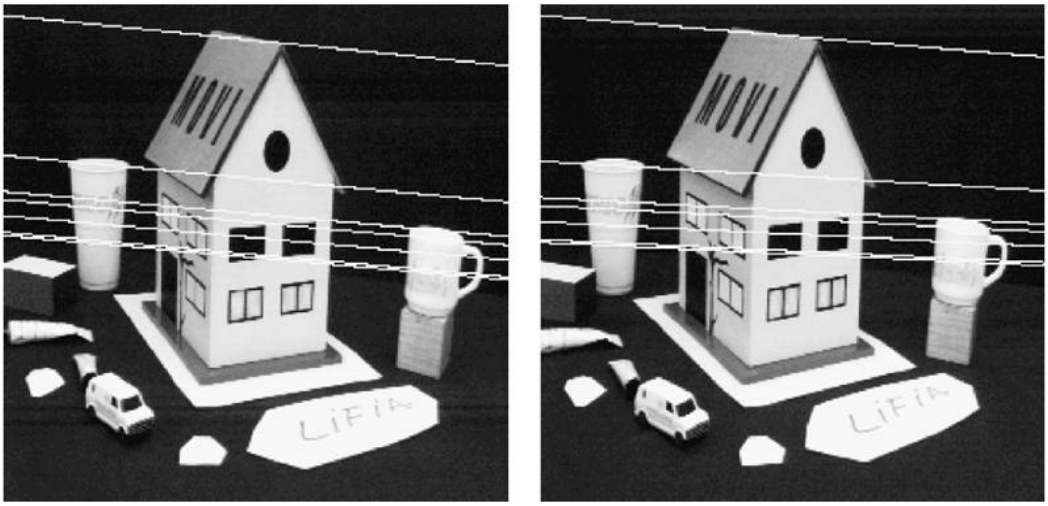


Figure 2.9: Hartley Epipolar geometry method, with epipolar lines (white lines) [31]

2.6 Depth in Stereo Systems

Stereo vision, stereopsis or Computational stereo is the process in visual perception that leads to perception of stereoscopic depth. In other word, Computational stereo refers to sensation of depth that emerges from the fusion of the two slightly different projections of the world on the two viewpoints. The fundamental basis for stereo is the fact that a single three-dimensional physical location projects to a unique pair of image locations in two observing cameras (Figure 2.10). As a result of given two camera images, if it is possible to locate the image locations that correspond to the same physical point in space, then it is possible to determine its three-dimensional location.

In this process the images may be taken by different cameras at the same time (stereo) or by the same camera at different times (motion). The reconstruction problem consists of determining three-dimensional structure from a disparity map, based on known camera geometry. The depth of a point in space P imaged by two cameras with optical centers $O_L$and $O_R$is defined by intersecting the rays from the optical centers through their respective images of P, p_L and p_R (see Figure 2.10). Given the distance between $O_L$and O_R, called the baseline T, and the focal length f of the cameras, depth at a given point may be computed by similar triangles as:

$$Z = f\frac{T}{d} \quad \textbf{(2.7)}$$

Depth is the most basic and essential part of this project. This calculation is based on similar triangles. In this part we assume that two cameras are identical and have same focal length, and two cameras view are in the same plane, and the height of object in two views are the same. With those assumptions, two point views are in one line and parallel to x-axis, as it is shown in Figure 2.10.

Based on these assumptions, every object creates a triangle $T\,[O_L, O_R, P]$. In that figure which is the top view of cameras and the object, $O_L$and O_R are the focal length of two cameras. The instance of object (P) in each camera is $p_L$and p_R. Besides that, $h_L\,[O_L, c_L]$ and $h_R\,[O_R, c_R]$ are the distance of object reflection to and the focal length where both are equal to each other. Z is distance of object and cameras optical center of camera.

Therefore, $T_L\,[O_L, P, H], T_R\,[O_R, P, H], T'_L\,[O_L, P, H], T'_R\,[O_R, P, H]$ where T_L and T'_L are similar triangles because $PH\ (Z)$ is parallel with $p_L h_L\ (F_L)$ since F_L is altitude of T'_L and Z is altitude of T_L. They have shared one angle (O_L) so the two triangles are similar to each other. Thus the ratio of $O_L h_L\ (N_L)$ over $O_L H\ (A)$ is equal to ratio of F_L over Z. The same goes for T_R and T'_R, $p_R h_R\ (F_R)$ is parallel with Z and they share O_R angle. The ratio of $O_R h_R(N_R)$ over $O_R h(B)$ is equal to ratio of F_R over Z

$$\frac{N_L}{A} = \frac{F_L}{Z} \tag{2.8}$$

$$\frac{N_L}{A} = \frac{F_L}{Z} \tag{2.9}$$

Since the two cameras are aligned to each other and the focal is same so camera view is parallel with $O_L O_R(Q)$ and F_L and F_R are perpendicular lines to O, F_L and F_R are parallel and they are equal to each other. In this situation (2.8), (2.9) can be written as:

$$\frac{N_R}{B} = \frac{F_R}{Z} = \frac{F_L}{Z} = \frac{N_L}{A} \tag{2.10}$$

In Figure 2.10, sum of A and B is equal to Q so we can replace B by $Q - A$ in (2.10). After simplifying the equation the new result will be:

$$\frac{N_R}{Q - A} = \frac{N_L}{A} \Rightarrow A(N_L + N_R) = N_L Q \Rightarrow A = \frac{N_L Q}{N_L + N_R} \tag{2.11}$$

Now if we locate A from (2.11) to (2.8) and simplify that equation we have:

$$Z = \frac{F_L A}{N_L} \xrightarrow{(2.11)} Z = \frac{\frac{F_L N_L Q}{N_L + N_R}}{N_L} \Rightarrow Z = \frac{F_L Q}{N_L + N_R} \tag{2.12}$$

Based on (2.12) depth (Z) has direct relation with focal length of two cameras and their distance. It also has reverse relation with sum of object instances from center of camera.

Disparity (d) is the resulting displacement of a projected point in one image with respect to the other. As it is shown in (2.12), depth has reversed relation with sum of N_L and N_R. Here N_L is the distance of object image from the camera center and x_L is the distance of object instance (from origin of first camera image). Assuming camera image length is $2 * O$ so the relation of x_L and N_L is:

$$N_L = x_L - O \quad \textbf{(2.13)}$$

The same result shows the relation of N_R and x_R:

$$N_R = O - x_R \quad \textbf{(2.14)}$$

Hence from (2.13), (2.14) we have:

$$N_L + N_R = x_L - O + O - x_R \Rightarrow N_L + N_R = x_L - x_R \quad \textbf{(2.15)}$$

Due to the position of cameras $x_L - x_R\ (d)$ in (2.15) is always positive because x_L is always larger than x_R. By finalizing the equation with replacing the result of $N_L + N_R$ with the object disparity we can get the depth formula based on disparity, focal length and distance of two cameras.

$$Z = f\frac{T}{d} \quad \textbf{(2.16)}$$

$x_L - x_R\ (d)$ which called disparity is the only variable that has reverse relation with depth. To estimate disparity, it is required to match object projection in each camera. To determine this projection it is required to find the correspondence of each pixel of image from left camera in image captured by right camera. This problem which is one of the most fundamental of computational stereo is discussed in next section.

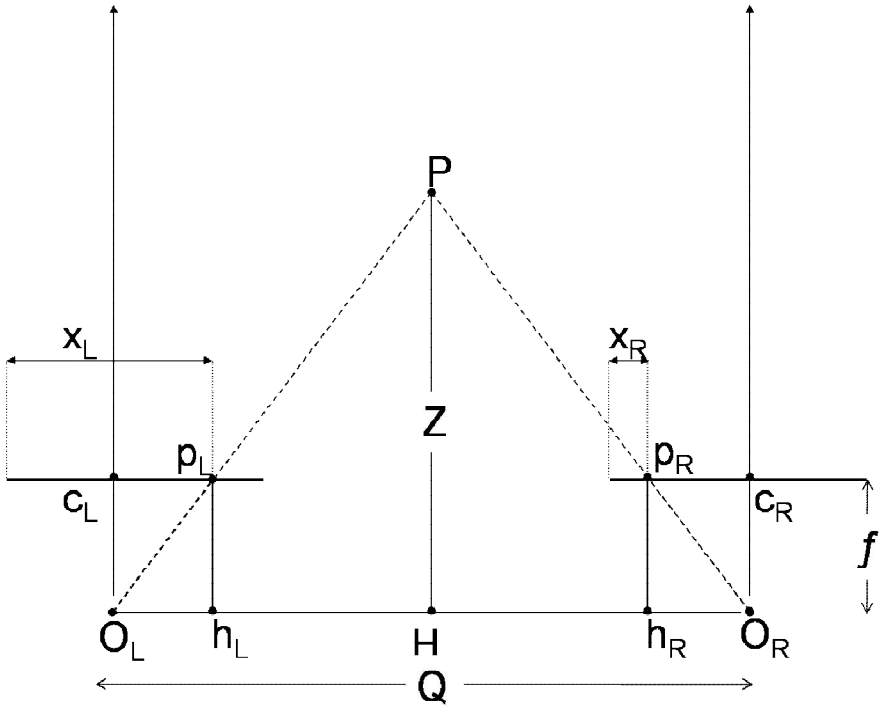


Figure 2.10: Scan line view of object reflection in each camera view

2.7 Correspondence

As it discussed in previous section, depth in computational stereo has reverse relation with disparity. To determine disparity value for selected pixel, it is crucial to determine the correspondence of that pixel. One of the challenges in stereo vision is correspondence as it discussed in the introduction. The term correspondence implies the determination of location in each camera, which in space is the projection of the same physical point. Correspondence is used to determine any two points taken from the same scene, in which they correspond to each other. In order to resolve correspondence, there are many algorithm suggested which can grouped into two different categories.

The first category is called as local method [33-37]. The local methods are methods often refer to constraints on small amount of pixels which surround a pixel of interest. Local methods which can be computed efficiently, are resistant to partial occlusion, and are relatively insensitive to changes in viewpoint. These approaches do not only detect interest points or so called features, but also proposed a method of creating an invariant descriptor. This descriptor can be used to (more or less) uniquely identify the found interest points and match them even under a variety of disturbing conditions such as scale changes, rotation, changes in illumination or viewpoints or image noise; though, due to relying on limited number of interest point these methods are not solely suitable for 3D reconstruction. They can be useful for detecting correspondence objects but for surface of those objects, there is no enough interest point for detail surface.

The second category is called as global method [38-41]. Global methods are methods which loosely refer to constraints on larger number of pixels, such as scan-lines or the entire image. With global methods, it can be less sensitive to locally ambiguous regions in image, such as occlusion regions or regions with unvarying texture, due to the fact that global constraint offers additional support to regions which are difficult to match locally. Nevertheless, these methods are more computationally intensive. In this section we will discuss these methods more in details.

Local methods will the focus of this section as up to now there are no global methods that can perform matching operation in real time. Also due to variety of local methods the two main algorithms (block matching and Optical flow) will be discussed.

2.7.1 Block Matching

Block matching [33, 42] method compares small region (pattern) about selected pixel with series of smaller regions within assumed disparity (d) range extracted from the second image (Figure 2.11). As it was discussed before, this method is one of the methods that highly rely on output result of epipolar geometry. Based on the output result the search region is one dimensional.

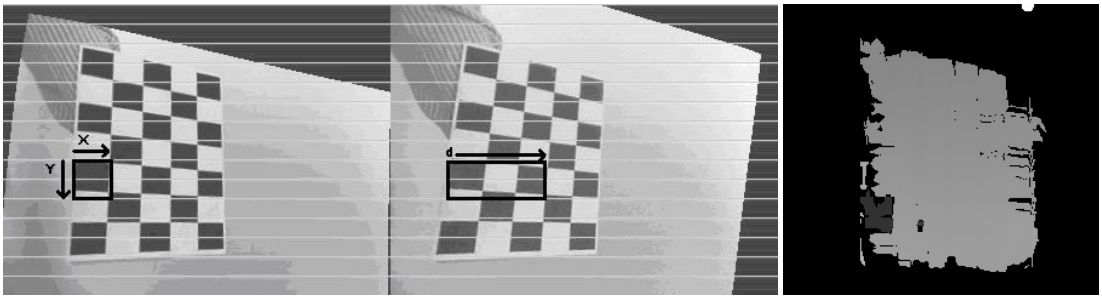


Figure 2.11: Block matching search after epipolar-geometry. The first rectangle in the left image is the template and the right image is the search region, last image is the disparity result from block matching.

Faugeras [34] proposed to use Cross-Correlation technique to determine the similarity between pattern and search window. Faugeras proposed a fixed window (kernel) on first image and shifting window on second image to calculate the score for correlation. Correspondence epipolar line is used as path of moving window in the second image. This technique is a statistical method such as Normalized Cross-Correlation (NCC) and Sum of Squared Differences (SSD) to quantify the similarity between two windows. He proposed to use the intensity value at selected pixel and compare in search window in one dimension with search area of assumed disparity, within the scan line. In his method the search window is fixed to give the method ability to search for several levels of resolution. Besides, in the proposed method the Gaussian smooth operation is applied to affect the resolution. He also mentioned the result will be more accurate if the size of the correlation window and the texture in the

image increased. In order to validate the result in this method the images will replace the position and selected correspondence point will be searched in the first image.

The accuracy in Faugeras method depends on three assumption made. Firstly, the images should be non-verged and detection of epipolar lines are nessesary. This is mainly to reduce the translation of search window to one dimention, resulting in increased matching. The next assumption is on search window. The size of the window is determined as $[2n + 1] \times [2m + 1]$. The score of each point then determined by i (between –n and n) and j (between –m and m). Faugeras mentioned that the accuracy is highly dependent on the size of the window but the increase in window size requires more processing power. The last assumption made is the object in image. As the textures increases, the possibility of similar point is reduced and thus the accuracy increases.

Overall if the images have less texture and the window's size depends on the available processing power, the final result may not be accurate and affect the matching. The proposed method by Faugeras, use actual intensity value. Actual intensity for same object in two images may vary due to changes in light direction or object reflection which can cause some issue. Since this method relies on the search window, it is highly sensitive to occlusion such as limb occlusion [8].

Zabih and Woodfill [35] proposed a local transformation system instead of using actual intensity value to reduce sensitivity to radiometric gain and bias. The problem with the current approaches during that time was the disparity discontinuity, since most of the correlation mathematical formulas designed for single distinct intensity population. Due to this, their research proposed a transformation before correlation process. This transformation caused the correlation to be based on the non-parametric features (features which are independent from the actual intensity value). They proposed two transformations.

The first transformation method called rank transform (Figure 2.12.a). The rank transform is a number of pixels in local region which their intensity is less than center pixel. This transformation cause data lost. Rank method only holds the number of pixels in kernel that their value is less than center pixel and the intensity and the sequence of the values will be lost.

Zabih resolved this problem by proposing a transform. Census transformation is a bit string representing the set of neighboring pixels in which their value is less than center pixel (Figure 2.12.b). This transform convert intensity values to 1 if the value of that pixel is less than center (selected) pixel. This string is represented as $\{0,1,\dots,d^2-1\}$ where d here is the window's size. Then these strings are compared with each other using concatenation process. Next using hamming distance the values are compared with each other. Hamming distance specifies the number of bits differs from each other. Local transformation reduces sensitivity of result to gain and bias changes.

$$\begin{bmatrix}10 & 34 & 14\\ 45 & 15 & 33\\ 53 & 13 & 48\end{bmatrix} = 3 \qquad \begin{bmatrix}10 & 34 & 14\\ 45 & 15 & 33\\ 53 & 13 & 48\end{bmatrix} = 10100010$$

(a) (b)

Figure 2.12: (a) Example of Rank, (b) Example of census

Fixed rectangular window in previous methods tend to cause many problem. Round corner and thin objects tend to disappear or expand. Kanade and Okutomi [43] proposed dynamic window as a solution to optimize matching based on intensity variation and resolving problem of fixed window. Proposed method by Kanade employs an adaptive window size based on intensity variation. Adaptive window in regions with depth of scene points varies small enough to detect correct match and for other regions large enough to overcome signal to noise ration problem. Kanade proposed the use of statistical model calculating the distribution of the intensities difference of two images within a window. This distribution model provides a base to determine the size of window used on specific location.

Even though Kanade method provides more accurate result (around 3 times less mismatch detected in results in average) but the method requires heavy great processing power and it was not feasible approach for real-time matching. Afterward, other methods try to reduce the processing power required by limiting the number of window. Geiger [44] proposed the use of three windows method. In this method, two additional windows will be placed on the right and left of original window to determine best possible correlation outcome. Fusiello [45] on the other hand proposed nine windows method. In this method as it is shown in Figure 2.13 there are nine

windows determining disparity for selected pixel. In this method SSD for each window is determined and the result for window with smallest SSD error will be selected.

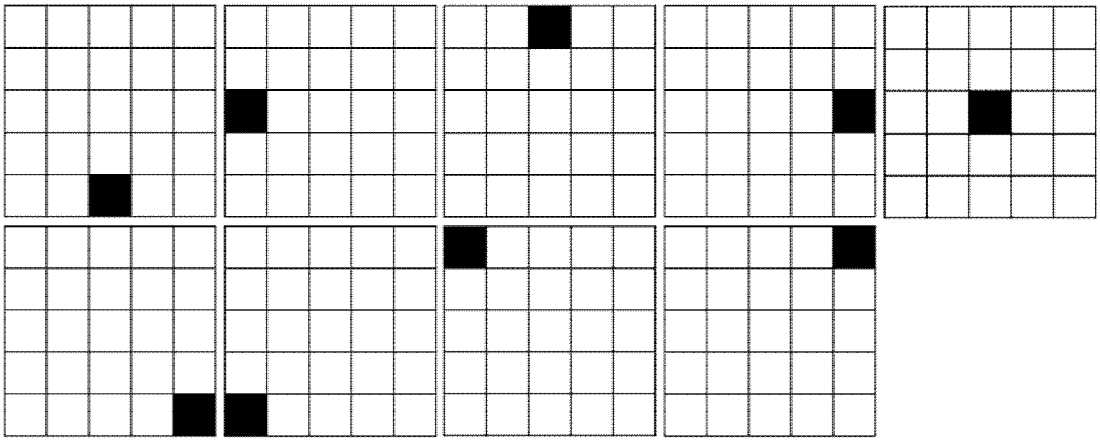


Figure 2.13: Fusiello windows to calculate disparity. Black pixel represent selected pixel to determine disparity.

Boykov [46] discussed the issues of correlation approaches with fixed window near discontinuity regions. He proposed a dynamic window method based on the intensity of selected pixel. In this method best connection around selected pixel is calculated based on the pixel intensity. These pixels should have minimum noise effect. In the proposed method the window shape is made of most plausible connected pixels with same disparity. Even though proposed method by Boykov supports real-time matching, but it includes nearby low texture area which result in increase of object in some cases.

Hirschmuller [47] proposed the use multiple overlap windows. In this method the result of window with no correlation error will be chosen. Figure 2.14 shows an example of this method. In this example, the result of the best two correlation of neighbor window will be added to correlation of centered window (C_0) as an overall correlation. Besides this configuration, the method can adapt other configuration which allows an increase in detection efficiency.

Hirschmuller also proposed a method to correct border by filtered apply on both left and right image. This filter compares neighbor disparity near to discontinuity region. Positive neighbor represents left object border and negative represents right object

border. The real position of object assumed within the distance of half the size of a correlation window.

Block matching evolved trough time, but most of the algorithm proposed is just modification on the pattern window size, shape and number of windows used. In this section it tries to introduce major modification done in this pattern window. In the other hand most of the focuses since early 2000's move to global methods due to better matching results and most of new developments where tried to combine block matching with other global methods.

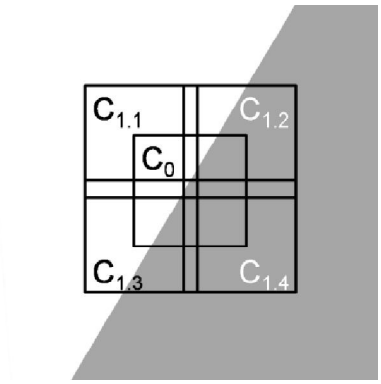


Figure 2.14: Partly Overlapped windows

2.7.2 Gradient-Based Optimization

Gradient-Based Optimization which is also known as optical flow (Figure 2.15) is another local method that tries to approach correspondence problem in free transformation environment. This method tries to resolve the problem by relating motion and image intensity. The main assumption in this approach is that the intensity of the two images remains the same all the time.

Most of the methods for optical flow rely on three basic approaches. Pre-filtering with low-pass/band-pass filters is the first approach. The goal of this method is, to extract signal structure while increasing signal-to-noise ratio. This step normally follows with extraction of basic measurements. This extraction is either done through spatiotemporal derivatives or local correlation surfaces. The last step is, the

integration of these measurements to produce a 2D flow field. This step often requires assumption about the smoothness of the underlying flow field.

Horn and Schunck [48] proposed to use the Laplacian (Figure 2.16) weighted difference to determine the rotation of an object such as cylinder and sphere. Gruen [49] proposed to use adoptive least squares correlation to overcome the problem of simultaneous radiometric correction and local geometry image shaping.

Anandan [50] proposed the use of band-pass filters to group the matching between two images. This approach tries to separate computations according to scale. The small scale intensity variation provides more accurate matching over large scale intensity changes. The matching process, perform in parallel process in each scale class and the final result will be combined. In this method the unreliable results will be propagated with neighbor reliable results.

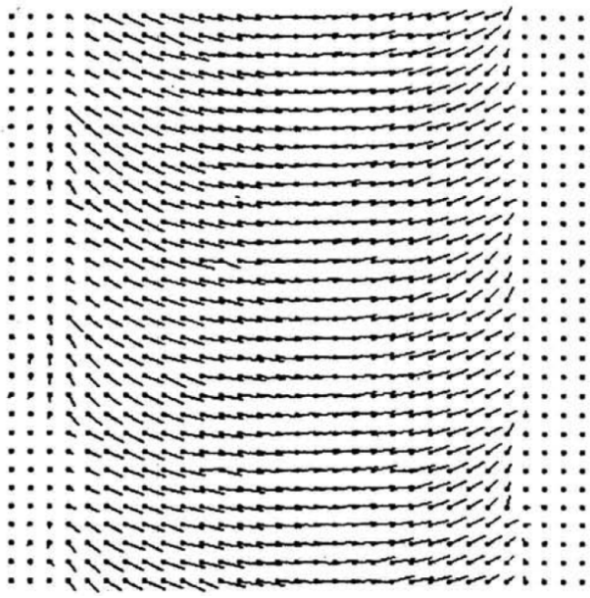


Figure 2.15: Optical flow result from rotating cylinder [48]

$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$
$\frac{1}{6}$	-1	$\frac{1}{6}$
$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$

Figure 2.16: Subtracting the weighted average from center pixel

Singh [51] proposed the use of local similarity of pixels neighbor with global smoothness. For local similarity Singh used the maximal accessible optical flow vector, similar to “d” in block matching method with the difference that “d” is along to one dimension where the vector here represents two dimensions. In this method to calculate the similarity, sum of square difference have been used. Singh also used the deviation from smoothness which attracted the vectors of neighbor pixels. Smaller directional uncertainty is the greater impact on neighbor pixels. There were two main issues with Singh’s method. This method requires great computational process. Calculating the local similarity in two dimensions requires more computational power compared to previous methods.

Gradient based optimization (optical flow) was developed mostly during 1980s and early 1990s. This method was used in other approaches such as model based tracing and rigid reconstruction. This method mostly relies on features selected by user (interest point) or features selected automatically based on the sudden intensity change on the specific point.

This method was used for tracing or reconstruction using structure from motion. Since this method highly relies on the intensity, it is not a feasible approach for stereo reconstruction. This method returns better result on small displacement where it was mentioned before as more practical for tracing or structure from motion approaches. This method can track changes in space invariant situation and it does not require verged images. As the number of selected features decreased to more specific points with high intensity change with its neighbor the performance of the method increases. In this section the focus was on the main contribution done to this method, since 1990’s most of development on this approach was on implementation of this method on other area [52, 53], and other improvement were similar to one introduce in this section.

2.7.3 Conclusion

Local matching methods prove to be a fast matching algorithm. This method performs accurate matching comparatively to other local methods. Block matching algorithm examine every pixel in reference image and tries to find matching pixel in secondary

image. Pattern window plays important rule in this matching algorithm and most of proposed improvements on this method are in this respect. These improvements were the variation on window size or number of window.

Increasing the window size will compromise processing time, thus many algorithms proposed a dynamic approach to moderate between processing time and accuracy. Number of window also has similar affect, although the focus is to minimize these numbers and search area by rectifying image. This process rectifies images to non-verge geometry and result in one dimensional search area. However faulty rectification may result in total mismatch and that is one of the drawbacks in block matching approach. Self-similar pixels are another issue that needs to be considered in block matching. Considering the self similar pixels fall under the search pattern, the block matching may fail to generate accurate match. This issue will be noticed more in situation that disparity range increased and search area needs to extend. Depth discontinuity, limb occlusion and orientation discontinuity are other block matching weakness.

Optical flow or gradient based optimization unlike block matching focuses on pixels with best feature to match. The processing time required for this method is highly depends on the number of pixels selected. The advantage of optical flow over block matching is two-dimension search. Since Optical flow search in two-dimension, thus image rectification is not necessary. Optical flow however requires the stereo cameras be identical for providing accurate result. Based on this approach the intensity of selected pixel and relation with its neighbors from reference image requires to remain the same as search image.

Issue mentioned above is one of the main reasons that optical flow fails to provide accurate result in unadjusted stereo images. This issue may occur more often in the cases that the object has many self similarity regions. Optical flow provides two-dimension search, but unlike block matching, it fails to find match for every pixels in images. Both methods suffer from self-similar texture, depth and orientation discontinuity.

2.8 Summary

Different steps involved in 3D reconstruction have been discussed. The most important step for 3D reconstruction is determination of camera matrix. Without camera matrix, intrinsic and extrinsic data, it is impossible to convert the data generated with correspondence methods (disparity values) to a real-world measurement.

Epipolar geometry and its importance in local matching are discussed. Once the fundamental information (Camera property and Images) are gathered, with help of epipolar geometry, it is possible to non-verged the stereo images. Stereo system depends on the disparity information provided by pair of stereo images. The extraction of 3D information from matching process is discussed as well. It also shows the relation of focal length (intrinsic parameter), the distance of 2 cameras (extrinsic parameters) and disparity value to determine the depth in stereo system.

Finally, for matching system two different groups of methods (Local and Global) are discussed. Among the local methods, block matching and optical flow are reviewed; and their advantages and disadvantages have been discussed. Block matching tries to eliminate the sensitivity to radiometric gain and bias. However this method is computationally intensive and epipolar geometry process is required before calculating the disparity. On the other hand, optical fellow method is sensitive to intensity change between images. Both methods are sensitive to depth discontinuity and uniform texture.

In the next chapter, a level-based method will be introduced. This method unlike the two other local methods relies on the relation of pixels with each other, based on the edge detected on the image. This method will be faster due to reducing the processed data to only edges in the image and also more accurate due to insensitivity of edge to gain and bias in the image.

CHAPTER 3
METHODOLOGY

Matching correspondence object in two pairs of images is the main goal of this research. In this section it tries to show the steps involved in 3D reconstruction using level-based correspondence. There are four main steps in this process as it is shown in Figure 3.1.

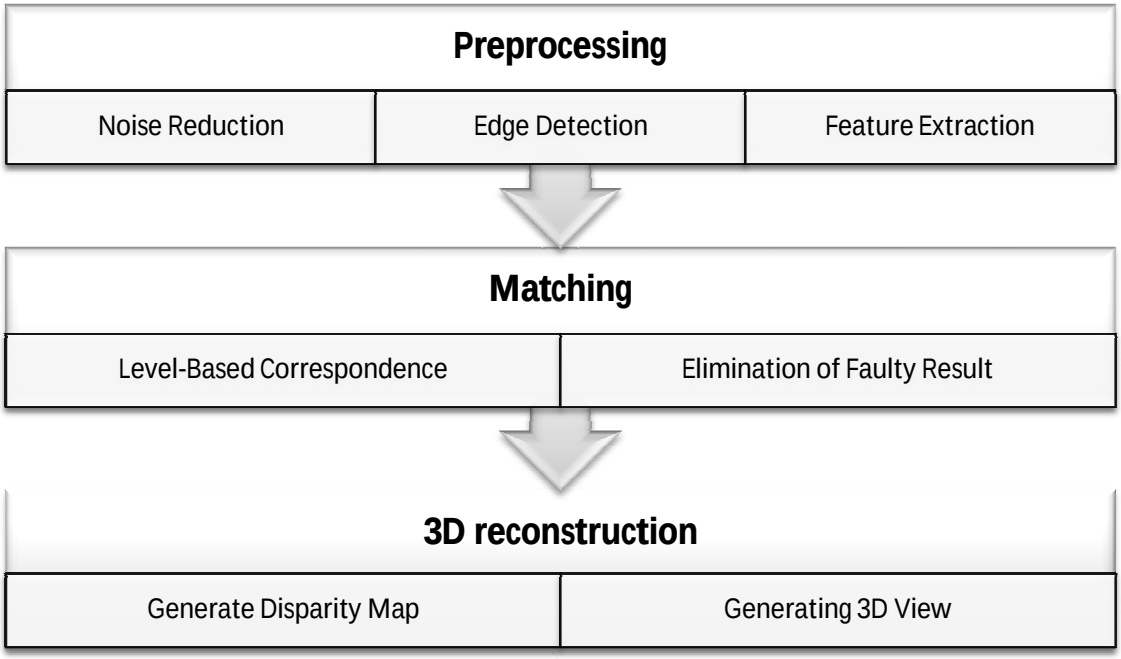


Figure 3.1 Process involve in 3D reconstruction using level-based correspondence

Even though the main focus is finding correspondence but to obtain the result, it requires certain steps such as preprocessing and feature extraction. In the following section, these steps will be further discussed.

In this chapter for illustration purpose, grayscale of Lena's picture (Figure 3.2) was used.

Figure 3.2: Lena's sample photo

3.1 Preprocessing

Preprocessing as most fundamental step in every image processing algorithm discussed in detail in chapter 2. The goal of this step is to remove unwanted data and reduce the amount of data for process. For proposed method selected preprocessing steps are noise reduction, edge detection and feature Extraction. Noise reduction process eliminates unwanted data such as noises. In the other hand the edge detection process remove every other information and remaining data are only edges and boundaries in the image. Finally feature extraction provides specific information regarding edges such as number of corners, status of edge (close edge or open edge) and levels. In the following, there will be short discussion about these steps with the algorithm used in the prototype of proposed method.

3.1.1 Noise Reduction

Noise reduction as the first step of image processing discussed in chapter 2. Among the methods morphing operation is selected as default (the reason for this decision will be discussed in next chapter). Figure 3.3.a is a basic method selection step, follow with morphing operation Figure 3.3.b and non-linear algorithm Figure 3.3.c. For linear filter and Gaussian filter the algorithm generates a filter mask and convolute the mask over the image. For the linear filter, considering n represent the kernel size every element in the mask will be $\frac{1}{n^2}$. Gaussian Filter follows the following formula to generate 2D mask.

$$m_{i,j} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right),\ \sigma = \sqrt[2]{2},\ \left(-\frac{n-1}{2} \le i \le \frac{n-1}{2}\right) \text{ and } \left(-\frac{n-1}{2} \le j \le \frac{n-1}{2}\right)$$

a: noiseFilter

```
INPUT:        Noise_Reduction_Method,Noise_Reduction_Kernel,Input_Image,Output_Image
OUTPUT:       none
ALGORITHM:
Select between Noise Reduction Methods
"Morphing":
morphinOperation (1, Noise_Reduction_Kernel, Input_Image, Output_Image)
"Linear Filter":
Filter_Mask ← create filter mask Shape: rectangle, Size: Kernel, values: 1/Kernel^2
Output_Image ←convolution of Filter_Mask over Input_Image
"Gaussian Filter":
Filter_Mask ← create Gaussian filter mask, Shape: rectangle, Size: Kernel, Sigma:
Sqrt(2)
Output_Image ←convolution of Filter_Mask over Input_Image
"Non-linear Filter":
nonLinearFilter(Input_Image, Output_Image, Noise_Reduction_Kernel);
Default:
morphinOperation (1, Noise_Reduction_Kernel, Input_Image, Output_Image)
End select
Return ;
```

b:morphinOperation

```
INPUT:        Input_Image, Output_Image , Iteration, Kernel
OUTPUT:       none
ALGORITHM:
Structure_Element ← create Structuring element Shape: rectangle, Size: Kernel
Output_Image ←Input_Image
For i←1 to Iteration step 1
Output_Image ← erode Output_Image with Structure_Element
Output_Image ← dilate Output_Image with Structure_Element
End for
Return ;
```

c:nonLinearFilter

```
INPUT:        Input_Image, Output_Image, Kernel
OUTPUT:       none
ALGORITHM:
For i ← 0 to Input_Image width-1 step 1
For j ← 0 to Input_Image height-1 step 1
Output_Image[i][j]  ← median of Kernel at Input_Image[i][j]
End for
End for
Return ;
```

Figure 3.3: noise reduction a: algorithm on selecting noise reduction method, b: Morphing operation algorithm, c: non-linear noise reduction algorithm

3.1.2 Edge Detection

Edge detection as mean to reduce processed data is next step in preprocessing discussed in chapter 2 (Refer to section 2.2.2 2.6 page12). Among the methods adaptive threshold is selected as default (the reason for this decision will be discussed in next chapter). Figure 3.4.a is a basic method selection step follow with Sobel operation Figure 3.4.b.

a:edgeDetection

```
INPUT:      Edge_Detection, Edge_Detection_Kernel, Input_Image, Output_Image
OUTPUT:     none
ALGORITHM:
Select between Edge Detection Methods
"Sobel":
sobel(Input_Image, Output_Image, Edge_Detection_Kernel)
"Robert Cross":
Output_Image[f(i,j)] ← [f(i,j)-f(i+1,j+1)]+[f(i+1,j)-f(i,j+1)]
"Canny":
Output_Image ← apply canny operation on Input_Image, kernel: Edge_Detection_Kernel
"Adaptive"
Output_Image ← apply adaptive threshold on Input_Image, kernel: Edge_Detection_Kernel
Default:
Output_Image ← apply adaptive threshold on Input_Image, kernel: Edge_Detection_Kernel
End select
Return ;
```

b:sobel

```
INPUT:      Input_Image, Output_Image, Kernel
OUTPUT:     none
ALGORITHM:
X_Sobel[f(i,j)] ← [f(i-1,j-1) + 2f(i-1,j) + f(i-1,j+1)] - [f(i+1,j-1) + 2f(i+1,j) +
f(i+1,j+1)]
Y_Sobel[f(i,j)] ← [f(i-1,j-1) + 2f(i,j-1) + f(i+1,j-1)] - [f(i-1,j+1) + 2f(i,j+1) +
f(i+1,j+1)]
Output_Image[f(i,j)] ← (X_Sobel[f(i,j)]^2 + Y_Sobel[f(i,j)]^2)^0.5
Return ;
```

Figure 3.4: Edge detection algorithm, a: edge-detection selection algorithm, b: Sobel operation

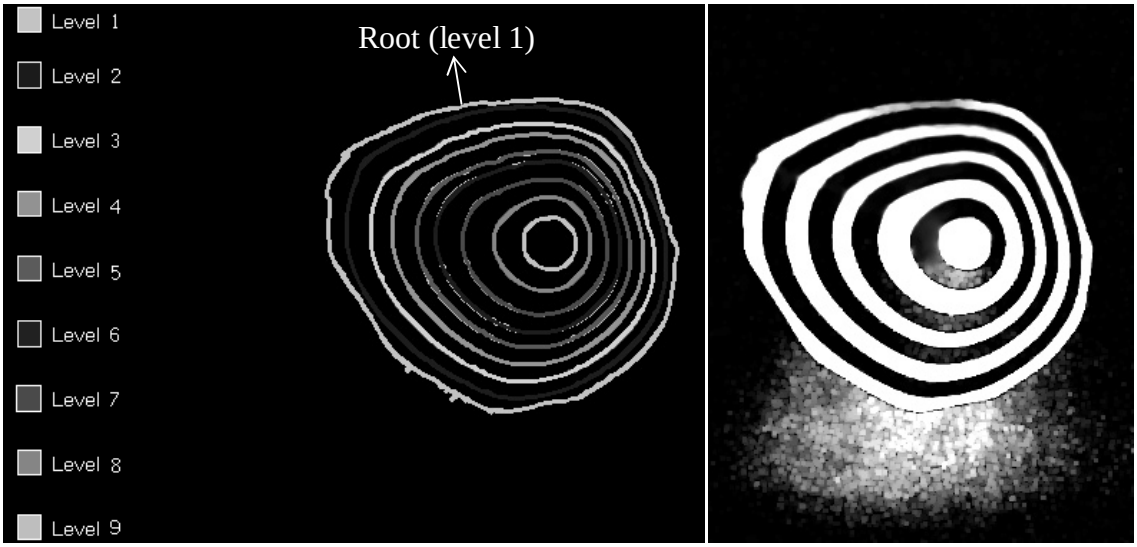
3.1.3 Contours and Level-Based Extraction

Level-based extraction is a method of feature extraction which relies on corners and their connection with each other in a hierarchal structure. In this section, the first step

is to extract the edges which determine the pixels and their relations with each other (Figure 3.5). Contours in image processing specify a close edge connecting multiple corners based on the edge detection result. Contours determine points relation with each other based on edge detected on the image. Once the contours and their boundary are determined, based on their relation with each other, they will be assigned to a level. Smaller contours or edges wrap by other contours consider as child contour of the warped contour. Warped contour in this case will be considered as parent contour. In this part, the first step of the approach is determining the parent-child relation. In other word specific level will be assigned to each contour (Figure 3.6). First level contours are the contours that the parent is the background and they are not considered as child of any detected contours.

The correspondence process in proposed method will be done on the points and their relations with each other based on the corners location in contours instead of solely focusing on each and every pixel. In this part, the contour will be more immune to changes such as intensity or translation or rotation. This is mostly due to nature of edges on the object. There is a high possibility that the relation of pixels intensity change between two images but the location of edges will remain the same in images. This is normally due to nature of edges which comes from texture on the surface and the discontinuity caused by the object. Another advantage of this method is the hierarchical access. Region of interest or in other word search area will reduced to boundary of parent contour in each image and matching process will focus only in that region. This process helps to reduce the amount of data being processed since the focus is on contours, edges and only corners on the contour or edges in the image.

Figure 3.5: Lena's photo, counter structured base on the edge detection result



(a) (b)

Figure 3.6: Level base feature, (a) in this image the circle with brighter color is the root and first level result and the inner circles are its child. (b) Original image

There are many methods proposed for feature extraction. The main goal of feature extraction approach is to identify the group of pixels that are less affected by changes such as intensity, object or camera transformation. Perfect features will be immune to noise and scaling of images. In this section, 3 groups of approaches were introduced. The first group focuses on the image corners. Corners are easier to identify in the image but due to small relation property these methods may be sensitive to transformation changes especially on discontinuity area which most of the edges are detected in that area. The next group (blob detection) focuses on the group of pixels and sudden intensity changes in that area.

Even though this group is more effective and due to reliance on higher number of pixels the features detected are more immune to transformation. Nevertheless, by detecting blob it requires more time and processing power which leads us to the proposed method. The proposed method relies on contours and edges on the image and corners located on them. Even though the relation of pixels intensities may change based on the changes on the scene lighting or transformation but the proposed method is less affected by these changes. Texture or object boundary remains the same regardless of changes in the scene lighting or camera transformation.

This method, by focusing on the result of edge detection, reduces the processing time and power since it relies on the pre-existing results from preprocessing, although at

the same time the performance of this method highly depends on the output result from preprocessing step which requires more focus on that. Due to advantages of feature extraction mentioned in this chapter, we rely on the output result to solve the correspondence problem. Figure 3.7.a is to extract the contours and their features from image, stored in Data_List to be processed followed by algorithm for corner extraction from detected contours in image (Figure 3.7.b):

a:addContours

```
INPUT:      Iterator, Contour, Data_List, Parent_Id, Self_Id
OUTPUT:     Self_Id
ALGORITHM:
While Contour has Node do
Contour_Boundary ← estimate top left and bottom right of Contour: x, y, width, height
Self_Id ← Self_Id + 1
Data_List[1] ← Contour_Boundary
Data_List[1][4] [Self_Id] ←  Parent_Id
Data_List[1][5][Self_Id] ← -1
Data_List[1][6][Self_Id] ← -1
Parent_Id ← Parent_Id + 1
Current_Node_Id ← Self_Id
exportCorners(Iterator, Contour, Data_List, Self_Id, 0)
While Node has Child do
Self_Id ← addContours(Iterator, Child, Data_List, Parent_Id, Self_Id)
Next Child
End while
Data_List[1][7][Current_Node_Id] ← Self_Id+1
Next Node
End while
Return Self_Id
```

b:exportCorners

```
INPUT:      Iterator, Contour, Data_List, Self_Id, Pair_Number
OUTPUT:     none
ALGORITHM:
Flag ← Contour type
Corners ← extract corner sequence
Count_Corners ← count corners in contour or edge
Data_List[Pair_Number][8][Self_Id] ← Count_Corners
For i ← 0 to Count_Corners step 1
Corner_Point ← extract point Corner [i]
Data_List[Pair_Number][9][Self_Id][i] ← Corner_Point
End for
Return ;
```

Figure 3.7: Feature extraction algorithms, a: Contours extraction, b: corners extraction

3.2 Correspondence

Correspondence in general is determination of relation between two sets of data. In image processing, correspondence can be referred to by identifying a set of pixel in one image identical to same pixel in another image. In other word, corresponding pixels in both images are representation of same object in real world stereo system. There are two groups of correspondence algorithms. The first group (local methods), focuses on the pixels or group of them and these methods are more sensitive to intensity changes. However they are faster in process and require less processing power compared to global method. This group of correspondence methods can perform operation on partial occlusion. This is one of the main reasons for proposing level-based correspondence to perform matching process in real time on low quality images.

As it has been discussed in the previous section before detecting correspondence the image required to go through a set of process to separate features in image from other pixels. The proposed method for feature extraction relies on the pixels and their relation with each other. These relations can be divided into three groups. The first group is the relation of pixels in one contour which relies on the result of edge detection. Pixels on the edges which are connected to each other consider building a level respect to their contours. The second group is the relation of contours with respect to their parents. As it has been discussed before, in this proposed method contours are going to be grouped based on the hierarchical level. The relation mentioned here is the parent and child relation in hierarchical aspect. Finally, the last relation is the relation of contours in same level and in the same parent level area. In this section it explains the process which detects and determines these relations and the use of these relations in solving the correspondence problem.

3.2.1 Level-based Correspondence Estimation

Correspondence problem is one of the fundamental problems in stereo systems. In this research it tries to find a way to resolve this problem relying on the pixels located inside a close edge (contour). As it is shown in Figure 3.8, the white color is edge detection result and black line (closed space) around the letter A is the outer boundary

of the edge which in this report is considers being level one contour. In the same image the inner boundary which is a triangular shape consider level 2 contours. In this section, after extracting contours and assigning level to each contour and edges which is explained in the previous section, the focus is on estimation of best possible match for each contours/edges in next image. For better understanding of concept, Figure 3.9 are designed which contain 4 level depth objects. The black color which is the background color in the image separate the first level contours from each other. The first level contours are the white color outer edge. In this image there are four objects in first level (8, A, B and &).

In this method, after detecting the first level contour, it tries to determine their correspondence in the next image. After detecting the correspondence for each of these contours the system select first contour and set the Region of Interest (RoI) of image to that level. In this section, a RoI is subset of image which only contains the boundary of the first contour (in this example contour of 8). After selecting 8 as the RoI the white color become the background and in this case the first level contours will be the outer boundary of dark gray color. Again in this sets of data there are two contours in the same level and the process continues until the background color does not contain any other contours.

In this example, the contour level goes up to 3 levels. Every time the process determine the correspondence in one group of contours up to last level the detection move to next highest level in which the example would be letter A and followed by letter B and at the end &.

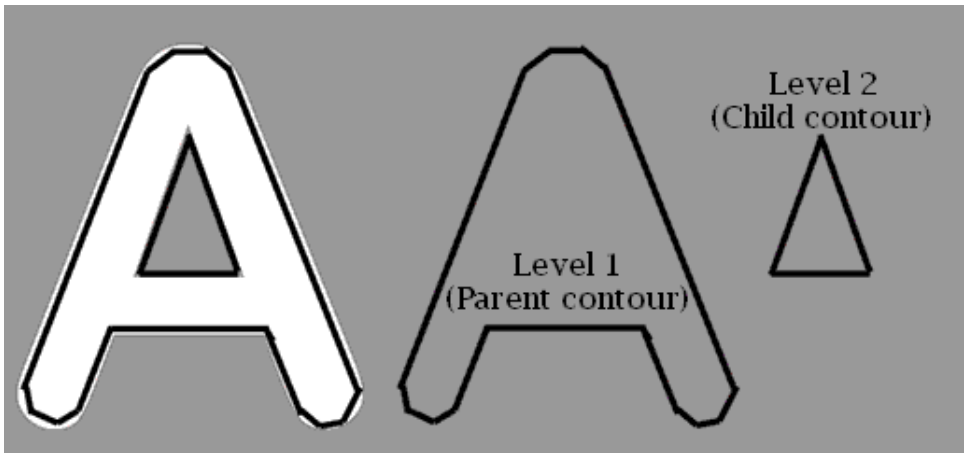


Figure 3.8: Closed edge (contour) sample for letter A

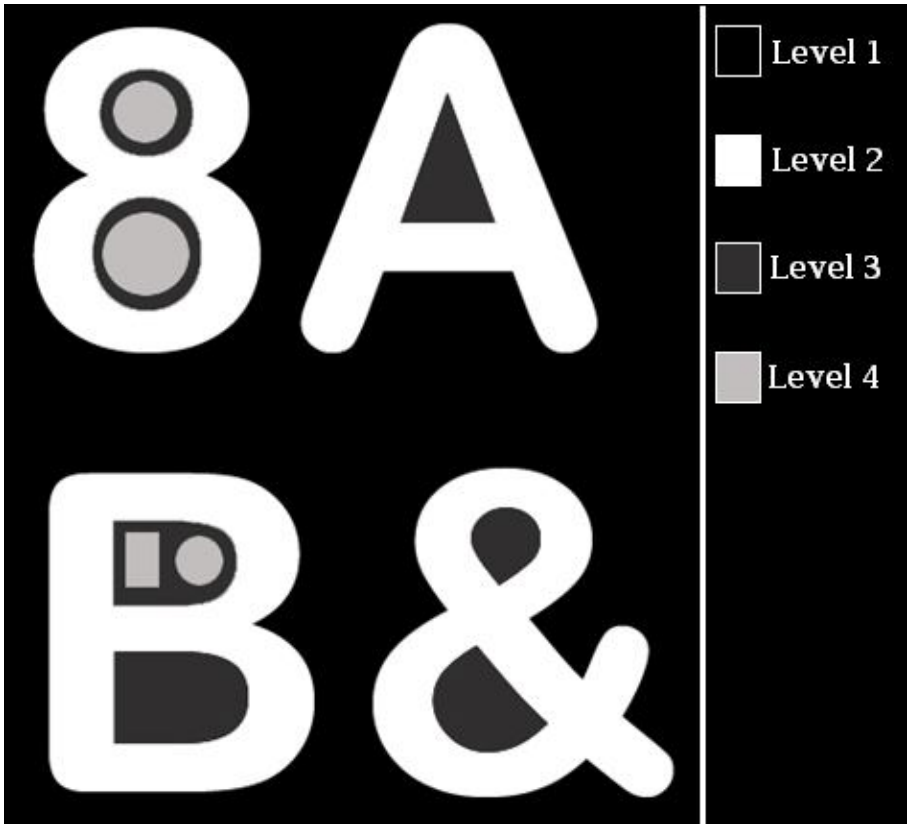


Figure 3.9: Contour sample, white color area first level contour, dark gray second level and bright gray is the third level

In this section, to estimate correspondence few assumptions are required. Firstly, both camera need to be located in same height. As it was discussed in chapter 2 this assumption is to verge camera edge to each other so the epipolar line in both image be aligned to each other. As it shown in Figure 3.10 the horizontal lines are the epipolar lines in which the images are aligned with them and transformed. The purpose of epipolar transformation is to ensure the pixels value in y-axis level remained the same in both images. In this section, the height of the contour and the height of top left point of RoI will be the same in both images.

Once the contours in the first level of first image are detected it tries to search for first level contours in second image which have the same height and x value. As it shown in Figure 3.10 it is possible to have two contours with same height in both images. To overcome this problem, it assumed that the relation of contours will remain the same in both images. In other word the sequence of contours in one row will remain the

same in both images. Moreover if the contours are in lower level the displacement in second image would be within the displacement of the parent contours.

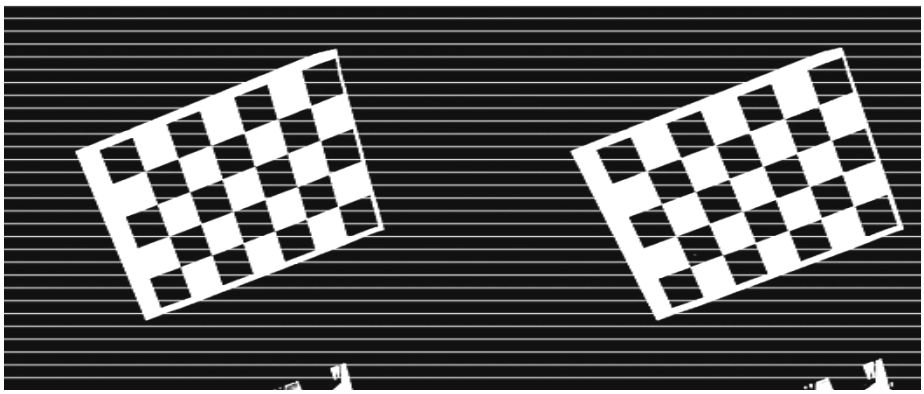


Figure 3.10: verged images align with epipolar lines

After estimating the position of correspondence object in second image the disparity value of the two objects is going to be calculated. Disparity is displacement of object (pixel or in this case top left of contour) from one image to another image. Assuming the distance of object from the left side of the first image is x_1 and this distance for object in the second image is x_2 then disparity (d) is the difference of x_1 and x_2. Once the disparity of the first level counter is calculated, the RoI will set to that contour boundary, in this case the second contour in that boundary now will consider as first level contour and same process will be done on each of the contours until there is no more second level contour in image (Figure 3.11). After all the disparity for all RoI's is calculated and adding the disparity of parent contour to the child contour is required to adjust the RoI affect.

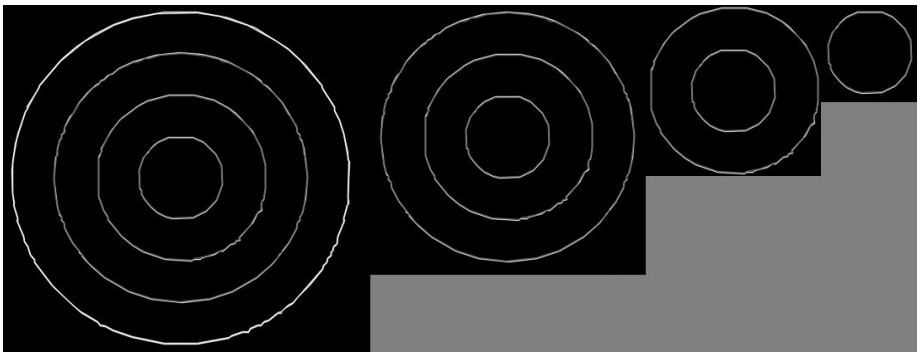


Figure 3.11: RoI in example image, from left to right the RoI is set to the next child inside the first level

In this section it defines what is contour and how level base can help to estimate overall disparity. In this method only the overall feature is detected and the detail result is not yet taken into consideration.

Figure 3.12 is matching algorithm. This algorithm searches the contours in second image and based on the level and the relations between contours attempt to find match for contours in the image.

a:matchContours

```
INPUT:      Iterator, Contour, Data_List, Y_Offset, Parent_Id, Matched_Id, Self_Id
OUTPUT:     none
ALGORITHM:
While Contour has Node do
Contour_Boundary ← estimate top left and bottom right of Contour: x, y, width, height
Correspondence_Contour ← findCorrespondence (Contour_Boundary, Y_Offset, Parent_Id, Data_List)
Self_Id ← Self_Id + 1
Data_List[1][5][Self_Id] ← -1
Data_List[1] ← Contour_Boundary
Data_List[1][4] [Self_Id] ←  Parent_Id
If Correspondence_Contour> -1 then
Data_List[0][5][Correspondence_Contour] ← Self_Id
Data_List[1][5][Self_Id] ← Correspondence_Contour
Disparity ← abosolute value of Data_List[0][0][Correspondence_Contour] - Data_List[1][0][Self_Id]
Data_List[1][6][Self_Id] ← Disparity
Y_Offset ← abosolute value of Data_List[0][1][Correspondence_Contour] - Data_List[1][1][Self_Id]
Parent_Id ← Parent_Id + 1
Matched_Id ← Correspondence_Contour
exportCorners(Iterator, Contour, Data_List, Self_Id, 1)
While Node has Child do
Self_Id ← Self_Id + 1
matchContours(Iterator, Child, Data_List, Y_Offset, Parent_Id, Matched_Id, Self_Id)
Next Child
End while
End if
Next Node
End while
Return ;
```

b:findCorrespondence

```
INPUT:      Contour_Boundary, Y_Offset, Parent_Id, Data_List
OUTPUT:     Corresponded_Contour
ALGORITHM:
```

```
Corresponded_Contour ← -1
j ← Data_List[1][5][Parent_Id] + 1
For i← Parent_Id to Data_List[1][7][Parent_Id] step 1
i ← i + 1
Y_Match ← Data_List[1][1][i] Match Data_List[0][1][j] between Y_Offset
H_Match ← Data_List[1][3][i] Match Data_List[0][3][j] between Y_Offset
if Y_Match and H_Match then
return j
Else
i ← Data_List[1][7][i]
j ← Data_List[0][7][j]
End if
End for
Return Corresponded_Contour
```

Figure 3.12: Correspondence algorithm, a: extract contours in second image and determine the level of each contour, b: matching algorithm

As it is discussed in this section even if the height of an image may not change in this stereo system but due to translation of camera, shape of object or occlusion, there is a possibility that some part of object hide from view which cause the width of counter to change. In the next section we discuss more about this problem and try to propose a solution to resolve this problem.

3.2.2 Elimination of Faulty Result

Feature extraction plays important role in correspondence, but due to changes in light intensity or camera property such as translation or rotation; it is not possible to rely on pixels all the time. Based on earlier discussion, this is the main reason that most of algorithm relies on the group of pixels to solve the correspondence problem. Even though contours are good feature to track but still it is not possible to only rely on overall edge results.

In 3D reconstruction contours provide only small amount of information required and detail will be left behind. In this section, it requires finding of the correspondence for every possible pixel in the edge result, but the same problem on missing pixel can happen here. For example the problems that are caused by occlusion, translation of camera or object rotation, intensity change or similarity of object border with

background and mistakes in edge detection process (Figure 3.13). In this section it tries to discuss about these issues and proposed a solution to resolve these issues.

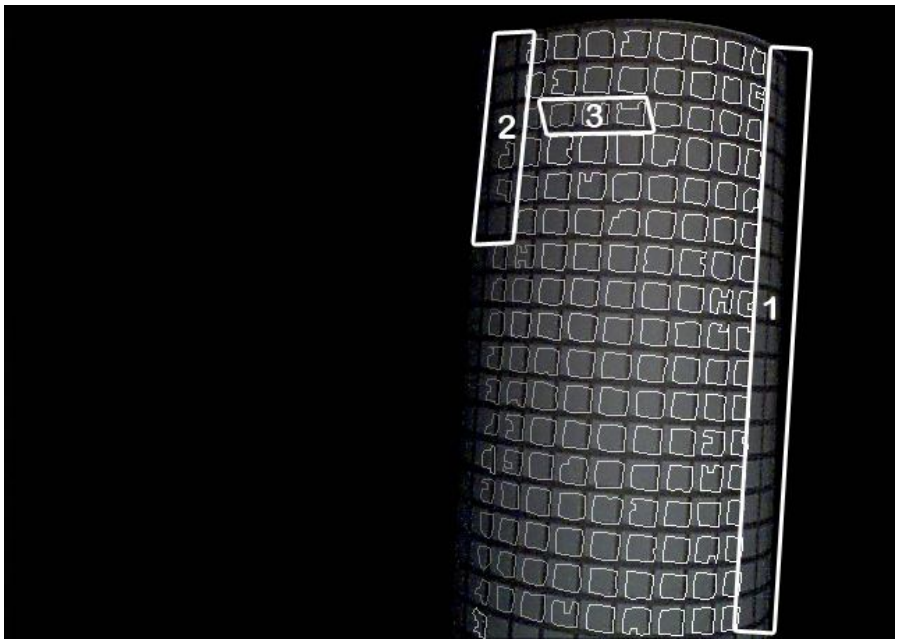


Figure 3.13: (1) misdetection due to occlusion, (2) misdetection due to affect of intensity and similarity of object and background, (3) misdetection due to faulty edge detection.

Occlusion in computer vision is defined as areas that are visible in one camera view however in other camera are invisible. Chung and Nevatia [8] define occlusion in three different categories. These categories are defined as base on nature of boundary between occluding and the occluded surface. The first category is considered as depth discontinuity, where occluding surface is occluded by another surface due to depth discontinuity. The occlusion area can be part of the same object or separate objects (Figure 3.14).

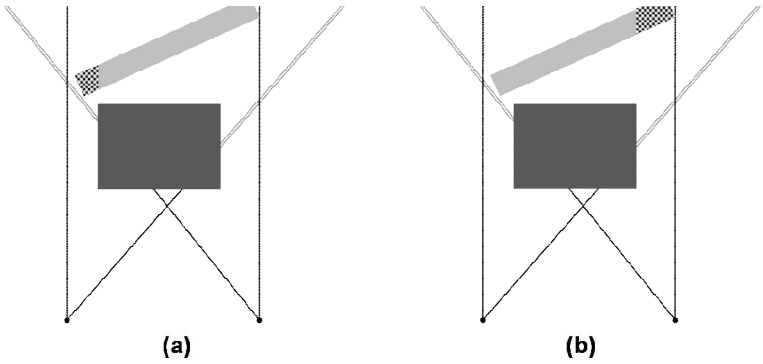


Figure 3.14: Depth discontinuity, (a) Shaded area is visible side to left camera but not to right camera (b) Shaded area is visible to right camera but not to left camera

Orientation Discontinuity occlusion is second group of occlusion where a surface of solid object occluded its adjacent surface. In Figure 3.15 it is shown that the first camera cannot see the right side of hexagon object while the left side of the object is not visible to second camera. The main characteristic of this occlusion focuses on the 2-junctions where one of them occludes the other one.

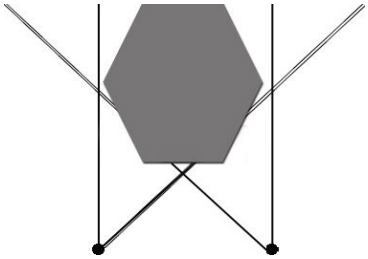


Figure 3.15: Occlusion due to orientation discontinuity.

In stereo vision, limb occlusion happen when closer side of a smooth convex such as eclipse surface occludes the more distance side of an object (Figure 3.16). It is shown in the figure that shaded area (a) is not visible to second camera while the shaded area (b) will not be visible for first camera while it is visible to second camera.

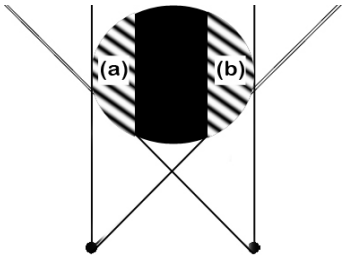


Figure 3.16: Occlusion due to Limb (a) Area that it is not visible to right camera but it is visible to left camera. (b) Area that it is not visible to left camera but it is visible to right camera.

Most of occlusion problems can be fixed by getting more sample image from different angles, but in some cases where only two images are available and the occlusion orientation is discontinued or limb occlusion, the occluded surface could retrieve from other method such as using intensity changes to estimate the overall shape of that area. In proposed method since capturing more images from different angles is possible, this method will eliminate the occluded area and it will be considered once there are more images from different angles are available. After occlusion in the next step it shows the effect of intensity

Beside occlusion when two surfaces are adjacent to each other, they have similar color which can cause misdetection. In this thesis since the focus is on one object the only problem that can occur is that the object border be similar to the background. In this case the correct edge of the first level contour may change, shrink or grow. The example (Figure 3.13) shows some of the contours are missing due to shrinking of the first level contour. In this section since this error is near to the edge of object same as problem with occlusion the process ignore the area and the problem can be fixed once more images are captured from different angle in future works.

Detected edges in images can also be different from each other (see area 3 of Figure 3.13). In this step the edges which are detected in left camera (reference camera), will be selected as base and the correction will be done based on the edges detected on the reference camera.

The first step in correcting the pixels is to count the output of each contour and find the corners on each of the contours. Next, based on the difference and by considering the closeness of points it can detect how many points are needed to remove or add to match the output result.

Figure 3.17 has shown one sample of these mistakes. In this example the number of extra corner in second edge is one (the difference of corners between matched edges). As it shown in this example the faulty corner has sudden change in y-axis direction while in x-axis direction it remains the same as previous corner on the edge and next corner in the same edge. In this case by comparison since the changes in y-axis direction should be minimal it is clear that this corner is faulty. In this situation there are two possible solutions. The first solution is to add another corner to first contour or edge to match their corners. Another approach is to eliminate the extra corner. In this approach the adding corner or removing them are depend on the reference camera (in this example left camera).

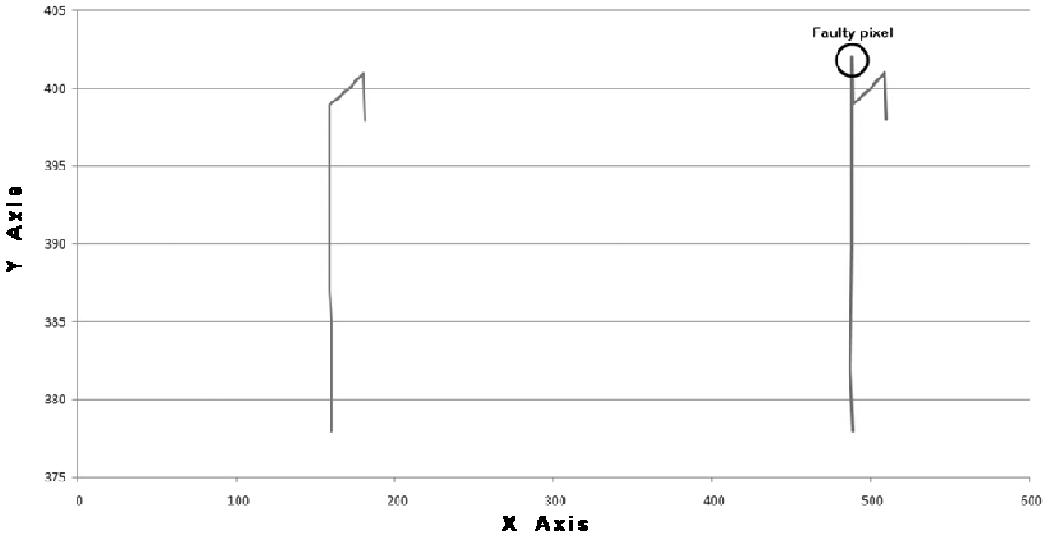


Figure 3.17: Faulty pixel result in faulty edge detection (circled pixel), the left edge belong to left camera and right edge belong to right camera

Since the changes only happened in y-axis direction and as it is discussed before, changes in y-axis direction should be minimal, due to nature of captured images. Since the images are align to horizon, the sudden changes along y-axis without changes along x-axis indicate misdetection and this result should remove from the sets of point in second pixels.

There are many cases that faulty point does not follow the above criteria and removing such point can cause losing important details of output result (Figure 3.18). In these situations to correct the faulty point, based on reference image and interpolate the corner from neighbor corners. In order to detect such mistakes again the number of point in contours will be compared and in the case that specific point has sudden change in y-axis or x-axis direction in one contour and such thing does not happen in another contour, based on the first contour as reference and comparison of previous point in that contour and next point in contour and overall disparity the result will be adjusted so the correspondence output be as detail as possible.

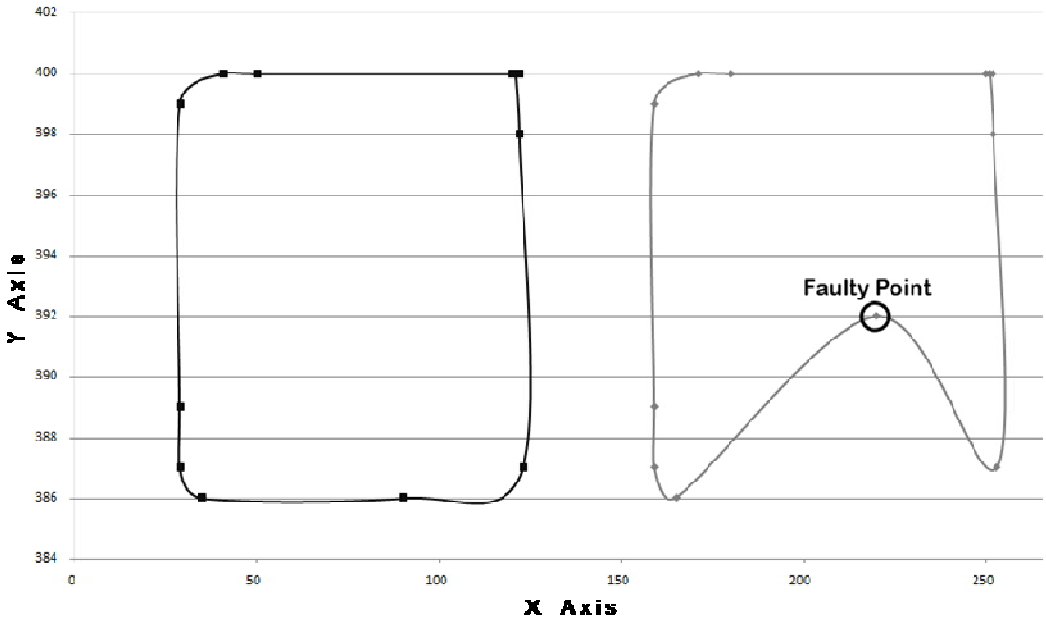


Figure 3.18: Faulty point detected on contour, the left contour belong to left camera and right contour belong to right camera

Once all the corners in contours are fixed and all corresponding contours have the same corners then based on the first image contour's result the disparity map is going to be created. In this step, the disparity of each corresponding corner is calculated (subtraction of corner position in y-axis direction, x_L for first image and x_R for second image) as: $d = x_L - x_R$. (Refer to section 2.6 page 27)

Further calculation of disparity is discussed. Correction of faulty corners in the edge detection, effect of occlusion and similarity of background and the object's border are few other issues that focused on this section. It also suggests some solution to improve such fault in system.

Figure 3.19 is algorithm for correcting data gathered from matching process. This algorithm attempt to resolve the issue of unmatched number of corners in matched contours based on the reference image (first image). If the number of corners in second image exceeds the number of corners in first image, then extra corners will be removed. In condition that the second image contour has less corners compare to matched contour in first image using interpolation new corners will be added.

correctCorners

```
INPUT:      Data_List
OUTPUT:     none
ALGORITHM:
i ← 0
While Data_List[1][5][i] do
If Data_List[1][5][i] >-1 then
Correspondence ← Data_List[1][5][i]
Self_Corner_Count ← Data_List[1][8][i]
Match_Corner_Count ← Data_List[0][8][Correspondence]
If Self_Corner_Count = Match_Corner_Count then
Continue;
End if
k ← 0
if Self_Corner_Count > Match_Corner_Count then
Difference ← Self_Corner_Count - Match_Corner_Count
For j  ← 0 to Matched_Corner_Count step 1
Matched ← compare Data_List[1][9][i][j] with Data_List[0][9][ Correspondence][k]
If not Matched then
While Difference do
Remove Data_List[1][9][i][j]
Difference ← Difference - 1
End while
End if
k← k + 1
End for
Else
Difference ← Match_Corner_Count - Self_Corner_Count
For j  ← 0 to Matched_Corner_Count step 1
Matched ← compare Data_List[1][9][i][j] with Data_List[1][9][ Correspondence][k]
if not Matched  then
New_Point  ←  estimate  the  position  of  new  point  based  on  previous  point  in
Data_List[1][9][i]
While Difference do
Add New_Point to Data_List[1][9][i]
Difference ← Difference - 1
End while
End if
k← k + 1
End for
End if
End if
i ← i + 1
End while
Return ;
```

Figure 3.19: Correcting the errors in edge detection or mismatches based on reference image

In the next section the reconstruction is discussed and the steps in reconstruction are explained. These steps are included as disparity map, dot cloud, wired reconstruction and finally solid reconstruction.

3.3 3D Reconstruction

3D reconstruction in computer vision and computer graphic in general means generating the shape and structure of object in real world using one, two or sequence of images. In this section, 3D reconstruction is the process of converting data gathered from correspondence section to a 3D model. These models consist of disparity map, dot cloud; wired structure and solid structure are discussed in detail in this section.

3.3.1 Disparity Map

Disparity map is intensity based representation of 3D model. This reconstruction is the most basic representation of 3D models. In this model the data gathered from the previous step is going to be displayed in 2D image and disparity is represented as intensity. This data includes calculated disparity in the previous section scaled from 50 to 255, position of corner, edge or contour in reference camera. In another word disparity map is a gray-scale image that depth represented as intensity or brightness. In this part as object get closer to camera the intensity increases.

Figure 3.21 shows a sample of disparity map. In this map as an example the top left corner is selected for generating disparity map. This corner has distance of 172 pixel from left side of reference image and 755 pixel from bottom of reference image, thus the position of this corner will be (172, 755) in disparity map. In the other hand the calculated disparity is 220 (in 200 to 278 disparity range) for selected corner. After scaling this disparity range (the range is 50 to 255), the intensity for representing the corner value is 64 (172, 755, 64). The grayscale image intensity range is between 0 and 255. The background of image is set to 0 (black). Due to low intensity of the background, the disparity range of corners, edges and contours are scaled within 50 to 255. Similar to this example every corner, edge and contour detected in image will be located in their position and based on calculated disparity specific intensity will be assigned to corner, edge and contour. Figure 3.21 disparity map is compared to the

right image and as it is displayed at the bottom right side of image is the brightest side of map (with higher disparity) and as it goes to top left side of the map it gets darker (lower disparity). Even though this result is fast and can provide the overall view of object but the 3D representation is not clear enough. On the other hand, disparity map is solely based on the disparity (d) (Refer to section 2.6 page 27) and does not represent the actual 3D position in real world. In other word the disparity map is a crude data which does not provide actual 3D information. This result is required to transfer and convert using camera matrix and the disparity need to scale to be relative to real world information. In this section after converting disparity to scale up the next step is to generate dot cloud which is going to be discussed in the next section.

Figure 3.20 is algorithm for generating disparity map. This algorithm based on maximum and minimum disparity value, scales the disparity value to intensity range. This algorithm then colors the contours and edges based on scaled disparity value to generate disparity map.

drowDisparityMap

```
INPUT:      Data_List
OUTPUT:     none
ALGORITHM:
Disparity_Range ← determine maximum and minimum disparity in Data_List[1]
Disparity_Max ← Disparity_Range Maximum
Disparity_Min ← Disparity_Range Minimum
Max_Min_Difference ← Disparity_Max - Disparity_Min
Scale ← 205 / Max_Min_Difference
i ← 0
While Data_List[1][5][i] do
Color = (Data_List[1][6][i] – Disparity_Min)* Scale + 50
Drow contour on Disparity_Map_Image, x: Data_List[1][0][i], y: Data_List[1][1][i]
color: Color
i ← i + 1
End while
Return ;
```

Figure 3.20: Algorithm to scale disparity value to intensity range for disparity map

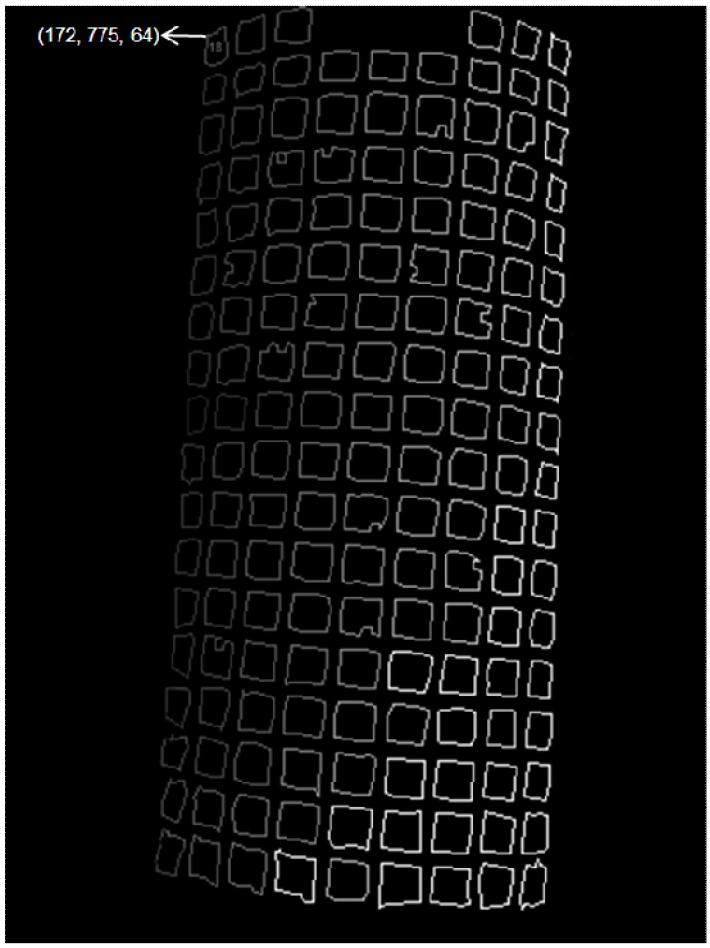


Figure 3.21: Disparity map

3.3.2 Dot Cloud and Wired Structure

Dot cloud is 3D representation of point selected in an object. This is the transformation of disparity to 3D representation. As discussed in section 2.6 , the actual depth can be estimated from focal length (f), camera distance from each other (Q) and disparity value (d).

$$Z = \frac{fQ}{d}$$

The focal length and camera distance from each other can be extracted from camera calibration (refer to section 2.2 page 9). In this section the focal length calculated is 672 pixels and camera distance from each other is 104 pixels. Consider the example in previous section. The disparity value for selected corner is 220 thus based on the

formula and available values the depth for specific point is 318 pixels, thus the position of this corner is (172, 775, 318).

Figure 3.22 shows sample of dot cloud output. This figure shows two view of dot cloud. In this result the 3D view is clearer and it is possible to translate or rotate object in 3D environment. Even though this result present better view of object in 3D world however it only relies on the point and without their connection the data is not understandable. This result is still considered as raw data and it requires further improvement to reach more realistic view of the object.

Figure 3.23 representation of corners, edges and contours in reference camera. In this representation each corner on edge and contours has their own depth. For better presentation instead of mesh or triangulation, the wired structure uses the available information such as contours or edges to represent the connection of these corners. Wired structure due to use of contours and edges, has better presentation over mesh and other triangulation methods.

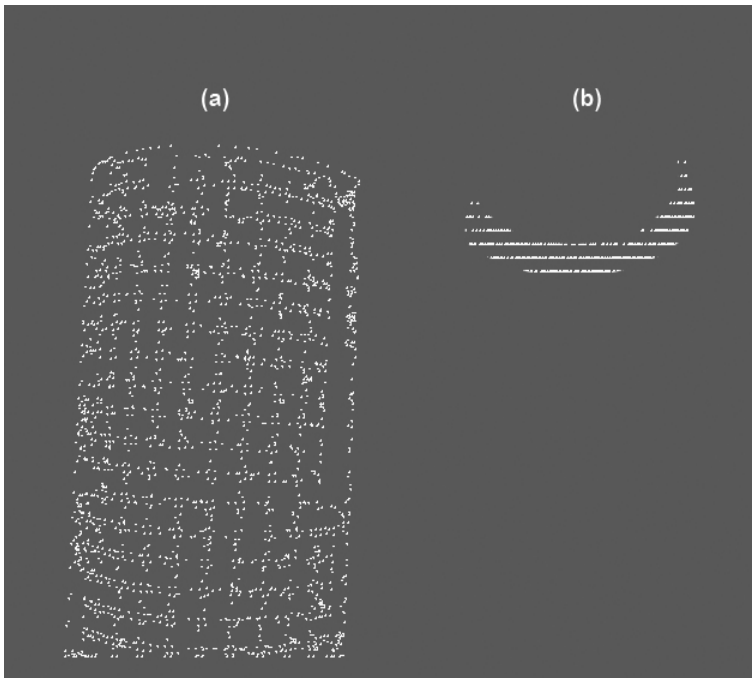


Figure 3.22: Dot cloud sample. (a) Front view of dot cloud, (b) Top view of dot cloud

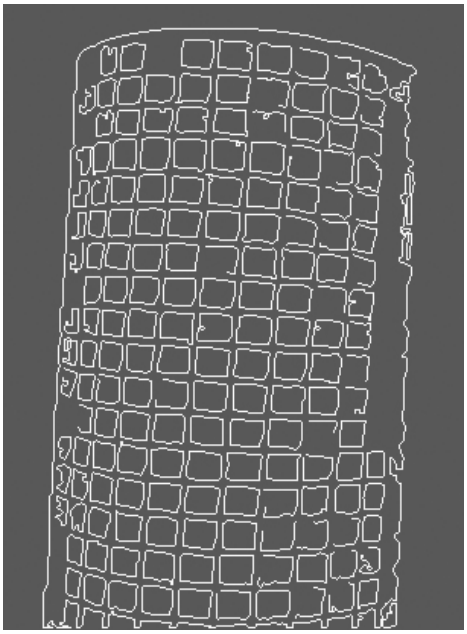


Figure 3.23: Wired structure

3.4 The Level-Based Algorithm

Level-based algorithm for local methods involve in three main sections, Preprocessing, correspondence and reconstruction. The following are overall algorithms from image acquisition to disparity map reconstruction

3.4.1 Main Algorithm

Figure 3.24 is the main function which acquire images and handle the steps involve in this process. There are no input and output from this function. This algorithm selects the default preprocessing algorithms and also allow user to modify these selections. Once the images are acquired the algorithms resizes images. After resizing images, in this algorithm the color image will be divided to three grayscale images and perform preprocessing on each individual. Finally the layers merge together and feature extraction and matching will be performed on results. Finally the matched result will be sent for correcting the corners and display the result as a disparity map.

aviSelect

INPUT: None

OUTPUT: None

ALGORITHM:

```
Edge_Detection_Method ← "Adoptive"
Edge_Detection_Kernel ← 3
Noise_Reduction_Method ← "Morphing"
Noise_Reduction_Kernel ← 3
Load Original_Image1
Color_Image1 ← imageResize(Original_Image1, 500)
For i← 0 to 3 step 1
    Create Gray_Image_List1 i with Color_Image1 property
End for
Create Binary_Image1 with Color_Image1 property
Create Disparity_Map_Image with Color_Image1 property
Load Original_Image2
Color_Image2 ← imageResize(Original_Image2, 500)
For i← 0 to 3 step 1
    Create Gray_Image_List2 I with Color_Image2 property
End for
Create Binary_Image2 with Color_Image2 property
For Infinit
    If Terminated then return;
    Gray_Image_List1 ← Split Color_Image1 to RGB layers
    Gray_Image_List2 ← Split Color_Image2 to RGB layers
    For i← 0 to 3 step 1
        noiseFilter(Noise_Reduction_Method, Noise_Reduction_Kernel, Gray_Image_List1 i,
        Gray_Image_List1 i)
        noiseFilter(Noise_Reduction_Method, Noise_Reduction_Kernel, Gray_Image_List2 i,
        Gray_Image_List2 i)
        edgeDetection(Edge_Detection_Method, Edge_Detection_Kernel, Gray_Image_List1 i,
        Gray_Image_List1 i)
        edgeDetection(Edge_Detection_Method, Edge_Detection_Kernel, Gray_Image_List2 i,
        Gray_Image_List2 i)
    End for
    Binary_Image1 ← combine Gray_Image_List1
    Binary_Image2 ← combine Gray_Image_List2
    detectContours(Binary_Image1, 0, Data_List)
    detectContours(Binary_Image2, 1, Data_list)
    correctCorners(Data_list)
    drowDisparityMap(Disparity_Map_Image, Data_List)
    display Color_Image1, Color_Image2, Disparity_Map_Image
    read user input Terminated, Noise_Reduction_Method, Noise_Reduction_Kernel,
    Edge_Detection_Method, Edge_Detection_Kernel
End for
```

Figure 3.24: Main algorithm for handling steps involve in level-based matching

Figure 3.25 gets the original image with specific image size and resize (enlarge or shrink the image to match the output size). The output size can be based on width or height of image depends on which one is bigger than the other. The output of this algorithm is the resized image.

imageResize

```
INPUT:      Original_Image, Output_Size
OUTPUT:     Resized_Image
ALGORITHM:
If Original_Image width > Output_Size and Original_Image width> height then
    Scale_Ratio ← Original_Image height / Original_Image width
    Resized_Image ← Create image with width: Output_Size, height: Scale_Ratio *
    Output_Size width
    Resize Original_Image to match Resized_Image size
Else if Original_Image height > Output_Size then
    Scale_Ratio ← Original_Image width / Original_Image height
    Resized_Image ← Create image with width: Scale_Ratio * Output_Size, height:
    Output_Size
    Resize Original_Image to match Resized_Image size
Else
    Resized_Image ← with Original_Image property
    Copy Original_Image to Resized_Image
End if
Return Resized_Image
```

Figure 3.25: Algorithm for resizing images

3.4.2 Preprocessing Stage

Preprocessing is to remove unwanted data. This includes noise reduction process and follow with edge detection. In this step a color image will be converted to binary image (edges). The followings are algorithms for this purpose. First the focus will be on the noise reduction method followed by edge detection.

Figure 3.26 is algorithm that is based on selected noise reduction method to reduce image noises. The input of this algorithm is selected noise reduction method, kernel size, input image and output image. This function does not return any value, however it will alter the value of output image. If selected noise filter be linear filter or Gaussian filter the algorithm will create corresponding filter and perform convolution on the image based on filter. For other methods respective function will be called.

noiseFilter

```
INPUT:        Noise_Reduction_Method,Noise_Reduction_Kernel,Input_Image,Output_Image
OUTPUT:       none
ALGORITHM:
Select between Noise Reduction Methods
    "Morphing":
        morphinOperation (1, Noise_Reduction_Kernel, Input_Image, Output_Image)
    "Linear Filter":
        Filter_Mask ← create filter mask Shape: rectangle, Size: Kernel, values:
        1/Kernel^2
        Output_Image ←convolution of Filter_Mask over Input_Image
    "Gaussian Filter":
        Filter_Mask ← create Gaussian filter mask, Shape: rectangle, Size: Kernel,
        Sigma: Sqrt(2)
        Output_Image ←convolution of Filter_Mask over Input_Image
    "Non-linear Filter":
        nonLinearFilter(Input_Image, Output_Image, Noise_Reduction_Kernel);
    Default:
        morphinOperation (1, Noise_Reduction_Kernel, Input_Image, Output_Image)
End select
Return ;
```

Figure 3.26: Algorithm for Selection of noise reduction filters

Figure 3.27 is morphing operation algorithm to reduce noise. The input of this algorithm is Iteration, kernel size, input image and output image. Iteration is the number of time that morphing operation need to be applied to input image. This function does not return any value, however it will alter the value of output image.

morphinOperation

```
INPUT:        Input_Image, Output_Image , Iteration, Kernel
OUTPUT:       none
ALGORITHM:
Structure_Element ← create Structuring element Shape: rectangle, Size: Kernel
Output_Image ←Input_Image
For i←1 to Iteration step 1
    Output_Image ← erode Output_Image with Structure_Element
    Output_Image ← dilate Output_Image with Structure_Element
End for
Return ;
```

Figure 3.27: Morphing Operation

Figure 3.28 is non-linear algorithm to reduce noise. The input of this algorithm is kernel size, input image and output image. this algorithm use mean of neighbor pixels and replace it with selected pixel. This function does not return any value, however it will alter the value of output image.

nonLinearFilter

```
INPUT:        Input_Image, Output_Image, Kernel
OUTPUT:       none
ALGORITHM:
For i ← 0 to Input_Image width-1 step 1
    For j ← 0 to Input_Image height-1 step 1
        Output_Image[i][j]  ← median of Kernel at Input_Image[i][j]
    End for
End for
Return ;
```

Figure 3.28: Nonlinear filter algorithm for noise reduction

Figure 3.29 is algorithm that based on selected edge detection method to reduce processing data. The input of this algorithm is selected edge detection method, kernel size, input image and output image. This function does not return any value, however it will alter the value of output image.

edgeDetection

```
INPUT:        Edge_Detection, Edge_Detection_Kernel, Input_Image, Output_Image
OUTPUT:       none
ALGORITHM:
Select between Edge Detection Methods
    "Sobel":
        sobel(Input_Image, Output_Image, Edge_Detection_Kernel)
    "Robert Cross":
        Output_Image[f(i,j)] ← [f(i,j)-f(i+1,j+1)]+[f(i+1,j)-f(i,j+1)]
    "Canny":
        Output_Image ← apply canny operation on Input_Image, kernel:
        Edge_Detection_Kernel
    "Adaptive"
        Output_Image ← apply adaptive threshold on Input_Image, kernel:
        Edge_Detection_Kernel
    Default:
        Output_Image ← apply adaptive threshold on Input_Image, kernel:
        Edge_Detection_Kernel
End select
Return ;
```

Figure 3.29: Algorithm for selection of Edge detection method

Figure 3.30 is simple edge detection algorithm to reduce processing data. The input of this algorithm is kernel size, input image and output image. This algorithm apply Sobel operation in x and y direction and calculate the length of edge in specific pixel. This algorithm does not return any value, however it will alter the value of output image.

sobel

```
INPUT:      Input_Image, Output_Image, Kernel
OUTPUT:     none
ALGORITHM:
X_Sobel[f(i,j)] ← [f(i-1,j-1) + 2f(i-1,j) + f(i-1,j+1)] - [f(i+1,j-1) + 2f(i+1,j) + f(i+1,j+1)]
Y_Sobel[f(i,j)] ← [f(i-1,j-1) + 2f(i,j-1) + f(i+1,j-1)] - [f(i-1,j+1) + 2f(i,j+1) + f(i+1,j+1)]
Output_Image[f(i,j)] ← (X_Sobel[f(i,j)]^2 + Y_Sobel[f(i,j)]^2)^0.5
Return ;
```

Figure 3.30: Sobel edge detection algorithm

3.4.3 Correspondence Stage

Correspondence is the main focus on this research. This algorithm extracts contours and corners from image to perform matching, based on the relation of contours in a hierarchical manner. Even though the feature extraction is part of preprocessing, due to importance of this process in this level-based correspondence, the algorithm for this process is included in this section.

Figure 3.31 is initialization step for extracting contours and their features. This is the main algorithm for proposed matching method. The input of this algorithm is input image, output image and a data list. Data list is a link list defined for this algorithm to hold features extracted from each contour. This feature includes contour position, contour size, level, parent, children, disparity value, matched contour, number of corners and corners in the specific contour. This algorithm does not return any value, however it will alter the content of data list.

detectContours

```
INPUT:      Input_Image, Pair_Number, Data_List
OUTPUT:     none
ALGORITHM:
```

```
Storage ←Allocate memory space for contours and corners
Mode ← Set the mode to retrieves all of the contours and reconstructs the full
hierarchy of nested contours
Method ← set to use one of the flavors of the Teh-Chin chain approximation algorithm
Contour ← extract contours from Input_Image, mode: Mode, method: Method, Storage <-
retrieved sequence
If Contour count <1 then Return ;
Y_Offset ← 12
Iterator ← initial hierarchy nodes from Contour
If Pair_Number = 1 then
    matchContours(Iterator, Contour, Data_List, Y_Offset, 0, -1, -1)
Else
    addContours(Iterator, Contour, Data_List, 0, -1)
    sortCorners(Data_List)
End if
Return ;
```

Figure 3.31: Main correspondence algorithm

Figure 3.32 is first step of feature extraction process. In this step the levels of contour is specified and all features are extracted and store in data list. This is a recursive algorithm traverse trough levels in image and return selected contour level.

addContours

```
INPUT:      Iterator, Contour, Data_List, Parent_Id, Self_Id
OUTPUT:     Self_Id
ALGORITHM:
While Contour has Node do
    Contour_Boundary ← estimate top left, bottom right of Contour: x, y, width, height
    Self_Id ← Self_Id + 1
    Data_List[1] ← Contour_Boundary
    Data_List[1][4] [Self_Id] ←  Parent_Id
    Data_List[1][5][Self_Id] ← -1
    Data_List[1][6][Self_Id] ← -1
    Parent_Id ← Parent_Id + 1
    Current_Node_Id ← Self_Id
    exportCorners(Iterator, Contour, Data_List, Self_Id, 0)
    While Node has Child do
        Self_Id ← addContours(Iterator, Child, Data_List, Parent_Id, Self_Id)
        Next Child
    End while
    Data_List[1][7][Current_Node_Id] ← Self_Id+1
    Next Node
End while
Return Self_Id
```

Figure 3.32: Feature extraction algorithm

Figure 3.33 is algorithm for extracting corners on the contour and store them in data list. This algorithm does not return any value, however it will alter the content of data list.

exportCorners

```
INPUT:      Iterator, Contour, Data_List, Self_Id, Pair_Number
OUTPUT:     none
ALGORITHM:
Flag ← Contour type
Corners ← extract corner sequence
Count_Corners ← count corners in contour or edge
Data_List[Pair_Number][8][Self_Id] ← Count_Corners
For i ← 0 to Count_Corners step 1
    Corner_Point ← extract point Corner [i]
    Data_List[Pair_Number][9][Self_Id][i] ← Corner_Point
End for
Return ;
```

Figure 3.33: Algorithm for extracting corners from contours

Figure 3.34 is first step of matching process. In this step the matched contour will be selected based on the features extracted in previous step. This is a recursive algorithm traverse trough levels in image and add matched id to data list.

matchContours

```
INPUT:      Iterator, Contour, Data_List, Y_Offset, Parent_Id, Matched_Id, Self_Id
OUTPUT:     none
ALGORITHM:
While Contour has Node do
    Contour_Boundary ← estimate top left and bottom right of Contour: x, y, width,
    height
    Correspondence_Contour ← findCorrespondence (Contour_Boundary, Y_Offset,
    Parent_Id, Data_List)
    Self_Id ← Self_Id + 1
    Data_List[1][5][Self_Id] ← -1
    Data_List[1] ← Contour_Boundary
    Data_List[1][4] [Self_Id] ← Parent_Id
    If Correspondence_Contour> -1 then
        Data_List[0][5][Correspondence_Contour] ← Self_Id
        Data_List[1][5][Self_Id] ← Correspondence_Contour
        Disparity ← abosolute value of Data_List[0][0][Correspondence_Contour] -
        Data_List[1][0][Self_Id]
        Data_List[1][6][Self_Id] ← Disparity
        Y_Offset ← abosolute value of Data_List[0][1][Correspondence_Contour] -
        Data_List[1][1][Self_Id]
        Parent_Id ← Parent_Id + 1
```

```
        Matched_Id ← Correspondence_Contour
        exportCorners(Iterator, Contour, Data_List, Self_Id, 1)
        While Node has Child do
            Self_Id ← Self_Id + 1
            matchContours(Iterator, Child, Data_List, Y_Offset, Parent_Id, Matched_Id,
            Self_Id)
            Next Child
        End while
    End if
    Next Node
End while
Return ;
```

Figure 3.34: Algorithm for matching extracted features

Figure 3.35 is algorithm for matching contours based on the features stored in data list. This algorithm will return matched contour id.

findCorrespondence

```
INPUT:      Contour_Boundary, Y_Offset, Parent_Id, Data_List
OUTPUT:     Corresponded_Contour
ALGORITHM:
Corresponded_Contour ← -1
j ← Data_List[1][5][Parent_Id] + 1
For i← Parent_Id to Data_List[1][7][Parent_Id] step 1
i ← i + 1
Y_Match ← Data_List[1][1][i] Match Data_List[0][1][j] between Y_Offset
H_Match ← Data_List[1][3][i] Match Data_List[0][3][j] between Y_Offset
if Y_Match and H_Match then
return j
Else
i ← Data_List[1][7][i]
j ← Data_List[0][7][j]
End if
End for
Return Corresponded_Contour
```

Figure 3.35: Algorithm for matching contours in levels

3.4.4 Reconstruction Stage

Reconstruction is the final step of this approach, where the data gathered will be corrected and the result will be converted to a disparity map. Figure 3.36 is algorithm for adjusting corners in contour based on matched contour. This algorithm has no output however it will adjust the corners in data list.

correctCorners(Data_list)

```
INPUT:        Data_List
OUTPUT:       none
ALGORITHM:
While Data_List[1][5][i] do
    If Data_List[1][5][i] >-1 then
        Correspondence ← Data_List[1][5][i]
        Self_Corner_Count ← Data_List[1][8][i]
        Match_Corner_Count ← Data_List[0][8][Correspondence]
        If Self_Corner_Count = Match_Corner_Count then
        Continue;
    End if
    if Self_Corner_Count > Match_Corner_Count then
        Difference ← Self_Corner_Count - Match_Corner_Count
        For j  ← 0 to Matched_Corner_Count step 1
            Matched    ←    compare    Data_List[1][9][i][j]    with    Data_List[0][9][
            Correspondence][k]
            If not Matched then
                While Difference ← Difference - 1 do
                    Remove Data_List[1][9][i][j]
                End while
            End if
        End for
    Else
        Difference ← Match_Corner_Count - Self_Corner_Count
        For j  ← 0 to Matched_Corner_Count step 1
            Matched    ←    compare    Data_List[1][9][i][j]    with    Data_List[1][9][
            Correspondence][k]
            if not Matched  then
                New_Point ← estimate the position of new point based on previous point
                in Data_List[1][9][i]
                While Difference ← Difference - 1 do
                    Add New_Point to Data_List[1][9][i]
                End while
            End if
        End for
    End if
End while
Return ;
```

Figure 3.36: Algorithm for correction of faulty corners in contours

Figure 3.37 is algorithm for adjusting disparity range between 50 to 255 to be displayed in a disparity map. This algorithm has no output however it will modify the disparity map image.

drowDisparityMap

```
INPUT:      Data_List
OUTPUT:     none
ALGORITHM:
Disparity_Range ← determine maximum and minimum disparity in Data_List[1]
Disparity_Max ← Disparity_Range Maximum
Disparity_Min ← Disparity_Range Minimum
Max_Min_Difference ← Disparity_Max - Disparity_Min
Scale ← 205 / Max_Min_Difference
i ← 0
While Data_List[1][5][i] do
Color = (Data_List[1][6][i] – Disparity_Min)* Scale + 50
Drow contour on Disparity_Map_Image, x: Data_List[1][0][i], y: Data_List[1][1][i]
color: Color
i ← i + 1
End while
Return ;
```

Figure 3.37: algorithm for scaling disparity value and generating disparity map

3.5 Summary

Correspondence was discussed in this chapter. An improvement on current local methods and the steps involved in implementing this method were discussed here. This chapter focuses further more on calculation of disparity map. Conversions of the disparity map to 3D representation also discussed. Different representations such as dot cloud, wired reconstruction and solid view are generated from the disparity result.

The basic steps in proposed method could result in some faulty matches. The Proposed method resolves some of the issues on post processing steps and generation of disparity map. The Effect of occlusion and the similarity of background are other issues on generated output. It also suggests some solutions to avoid such error.

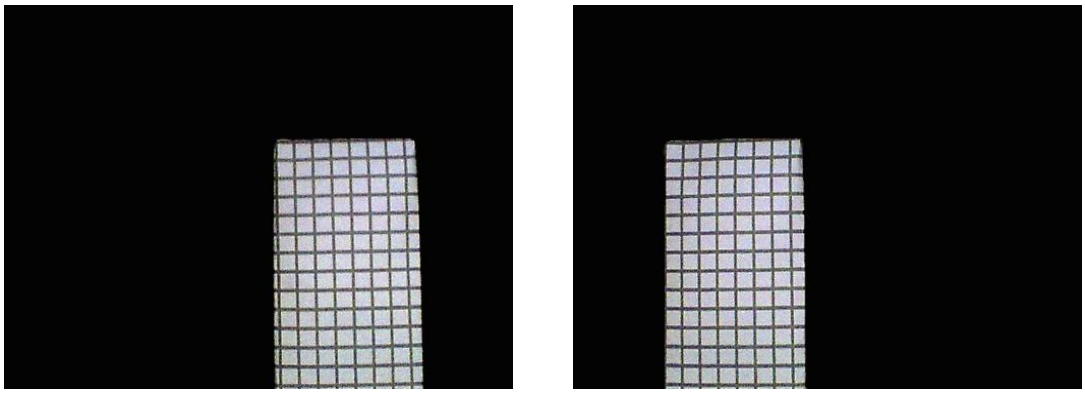
In the following chapter the result will be generated from different steps in this process will be discussed. Also there will be a comparison between different local methods. Finally based on available camera information, three different sets of stereo images will be processed and 3D result will be reconstructed.

CHAPTER 4
RESULT AND DISCUSSION

3D reconstruction from a pair of images requires a sequence of steps discussed in the previous chapter. Selecting the right method for each of these steps based on the objective is mandatory. In this chapter we try to compare different methods introduced in the previous chapter and discussed the advantages and disadvantages of them. At the end the disparity maps generated by level-based approach will be compared with different methods introduced in chapter 2.

4.1 Preprocessing Methods

In this section a flat surface (Figure 4.1) is processed with three stages introduced in chapter 3. The result of the first step is going to be compared in four different noise reduction methods with 3 different kernel windows (3x3, 5x5 and 7x7). The discussion will be on output result of noise reduction and processing time required for each method.



Left camera view Right camera view

Figure 4.1: Flat surface, original view

4.1.1 Noise Reduction

Noise reduction as primary step plays an important role in image processing. Noise reduction determines which data is unwanted and attempts to remove such data from image. The drawback of most noise reduction methods is blur effect. The goal in this step is to select a method which has minimum affect on edges in the image. Blur effect on images, as it shown in Figure 4.2, change based on selected algorithms. All algorithms successfully remove salt and pepper noise. Although linear filter is the only method that successfully removed Gaussian noise however the blur affect all the edges in the image. Among selected methods morphing operation preserver the edge boundaries however there are some traces of Gaussian noise remains in the output.

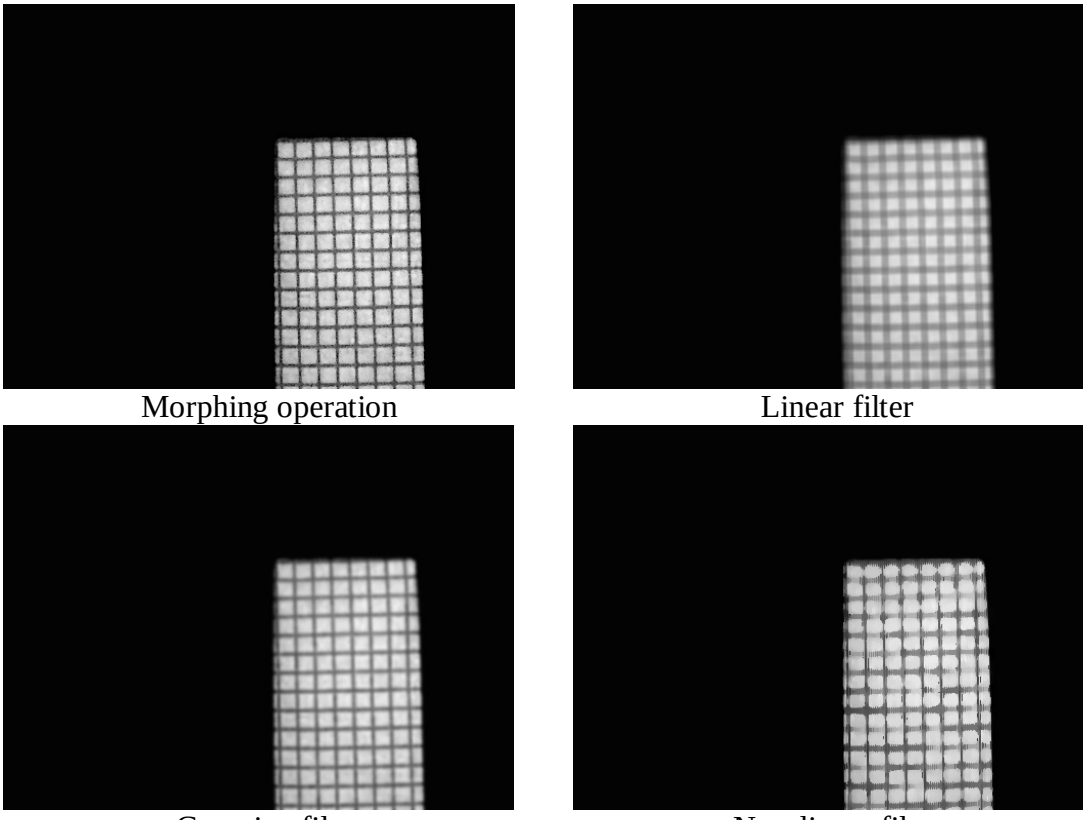


Figure 4.2: Noise reduction output with kernel size 7.

Figure 4.3 present the processing time required for different noise reduction algorithm. The processing time consumed by most of the algorithm is acceptable however non-linear method used here shows exceeding growth by increasing window size. Edge preservation, amount of noise reduced and processing time are the key

points in selection of default method for this process. Morphing operation is the only method that preserved edges regardless of kernel size. This method removes salt and pepper noise; however there are few traces of Gaussian noise remain in the output. Processing time required for this Morphing operation comparatively is acceptable. Due to this morphing is selected as default noise reduction method.

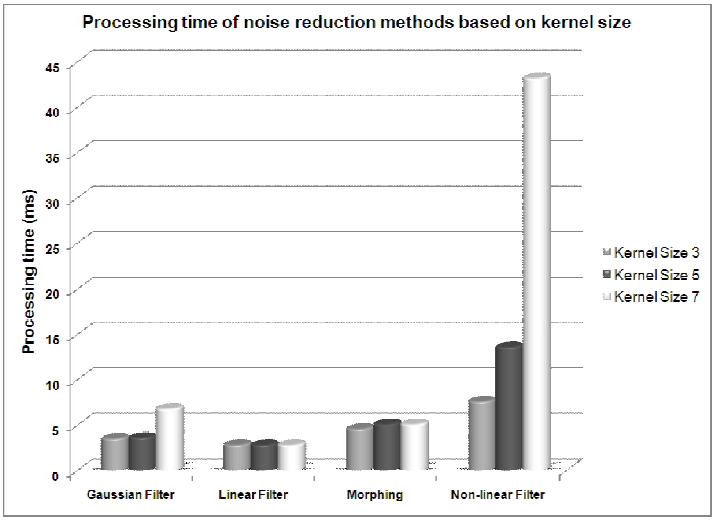


Figure 4.3: Processing time of noise reduction methods based on kernel size

Table 4.1: Processing time (ms) table for noise reduction with different kernel size

Kernel Size	Gaussian Filter	Linear Filter	Morphing	Non-linear Filter
3	3.5	2.8	4.6	7.6
5	3.6	2.8	5.1	13.6
7	6.8	2.8	5.1	43.2

4.1.2 Edge Detection

Based on the result discussed in the previous section, in this section four edge detection methods will be compared based on the kernel size used in the method. Figure 4.4 shows edge detection methods after noise reduction using morphing operation (kernel 3). The edge detection kernel used for this figure is a [3x3] matrix (kernel 3). Sobel operation detected the horizontal and vertical edges. Robert Cross detected corners in the image. Canny operation and Adaptive method detect most accurate object texture. Among the methods Adaptive method and Robert cross

required minimum processing time and Canny operation and Sobel operation requires the most processing time (Figure 4.5 and Table 4.2).

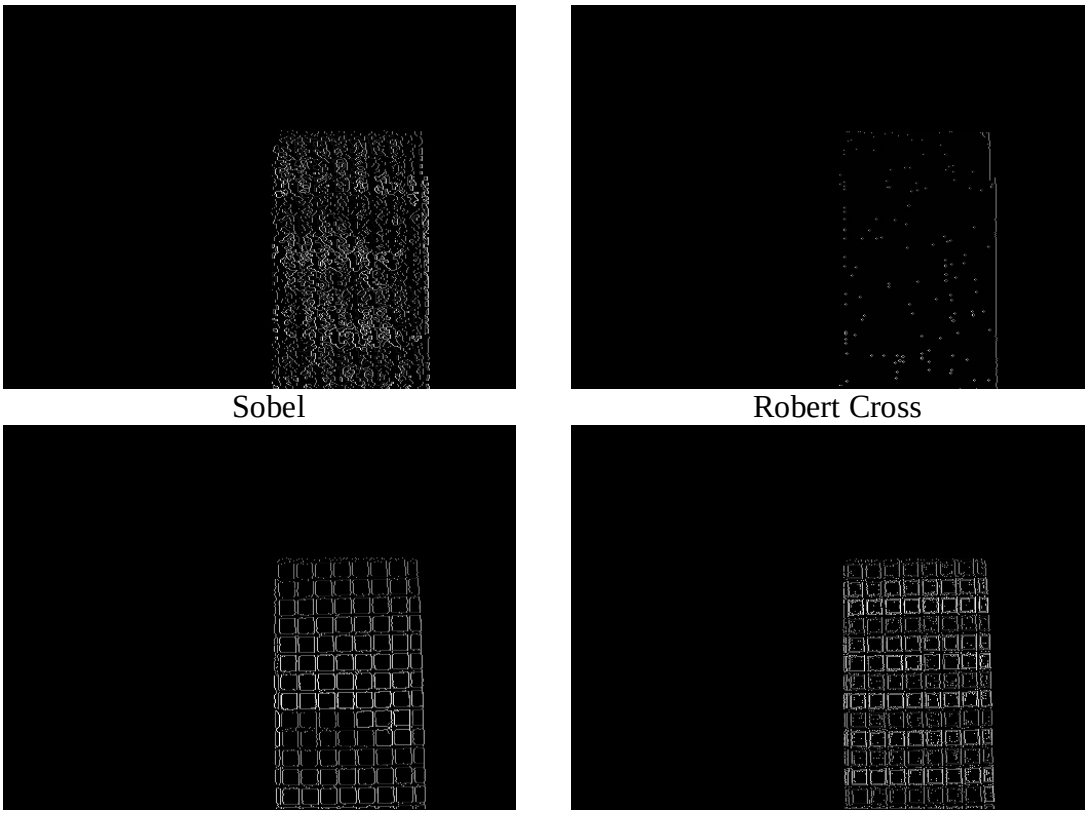


Figure 4.4: Flat surface, edge detection kernel 3 using morphing operation with (kernel 3)

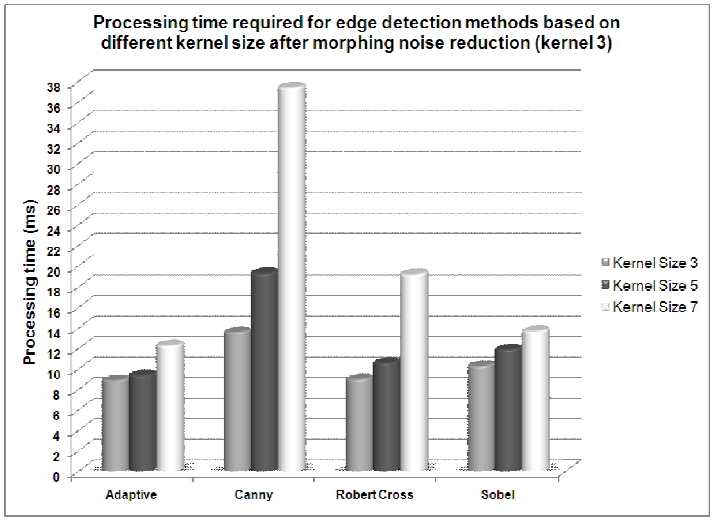


Figure 4.5 : Processing time for edge detection base on kernel size after morphing noise reduction (kernel 3)

Table 4.2: Processing time for edge detection base on kernel size after morphing noise reduction (kernel 3)

Edge detection Processing time		
Noise reduction method (Kernel 3)		
Edge detection kernel	**Edge detection method**	**Morphing operation**
3	**Adaptive**	8.9
	Canny	13.6
	Robert Cross	9.0
	Sobel Operation	10.3
5	**Adaptive**	9.4
	Canny	19.3
	Robert Cross	10.6
	Sobel Operation	11.9
7	**Adaptive**	12.3
	Canny	37.4
	Robert Cross	19.2
	Sobel Operation	13.7

After analyzing each edge detection method with regard to noise reduction algorithm in this part we discuss the overall processing time required to detect edges on the image based on the performance of the method. Figure 4.6 shows the processing time required for each edge detection algorithm based on the kernel size used. The noise reduction method in this figure is based on kernel size 3. In this figure Canny edge detection always requires more processing time however the Canny method with kernel 3 returns the best result (Figure 4.7). On the other hand, adaptive method requires minimum processing time and the output result except the kernel 3 is better than other methods due to reduction of detected noise. By focusing on the kernel 3 edge detection methods, even though Canny shows better performance but due to heavy process, the adaptive method shows better performance with regard to processing time required and edge detection output.

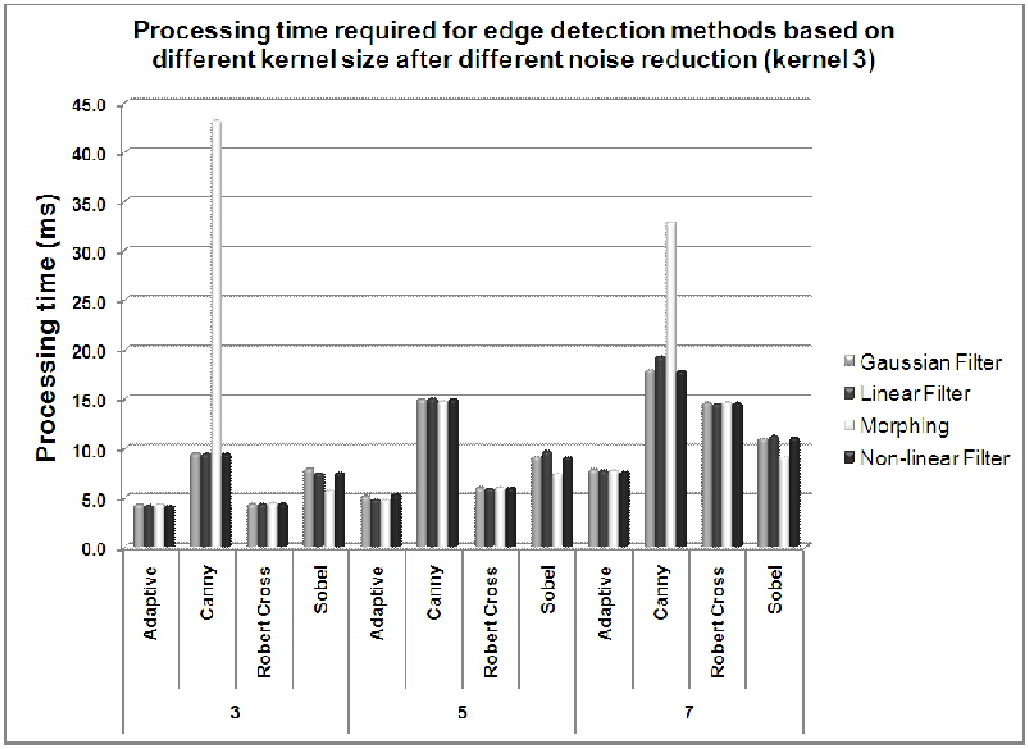


Figure 4.6: processing time comparison between different edge detection method base on kernel size used after each noise reduction method using kernel size 3

By comparing results from different edge detection methods based on their kernel size after noise reduction with their relative kernel size, it is clear that kernel size 3 of noise reduction can perform faster and better for this approach. In the following step the detected object and correspondence for different edge detection method will be discussed and compared.

Figure 4.7 shows detected contour with their correspondence for different edge detection method based on their kernel size after noise reduction process with kernel size 3. In this figure the adaptive and Canny edge detection shows the most detected object with edge detection kernel size 3 and 5. Among the noise reduction methods non-linear filter and morphing operation are the most promising methods. In edge detection kernel size 5 Canny detect slightly more objects but in kernel size 7, adaptive threshold shows the highest result compared to other edge detection methods.

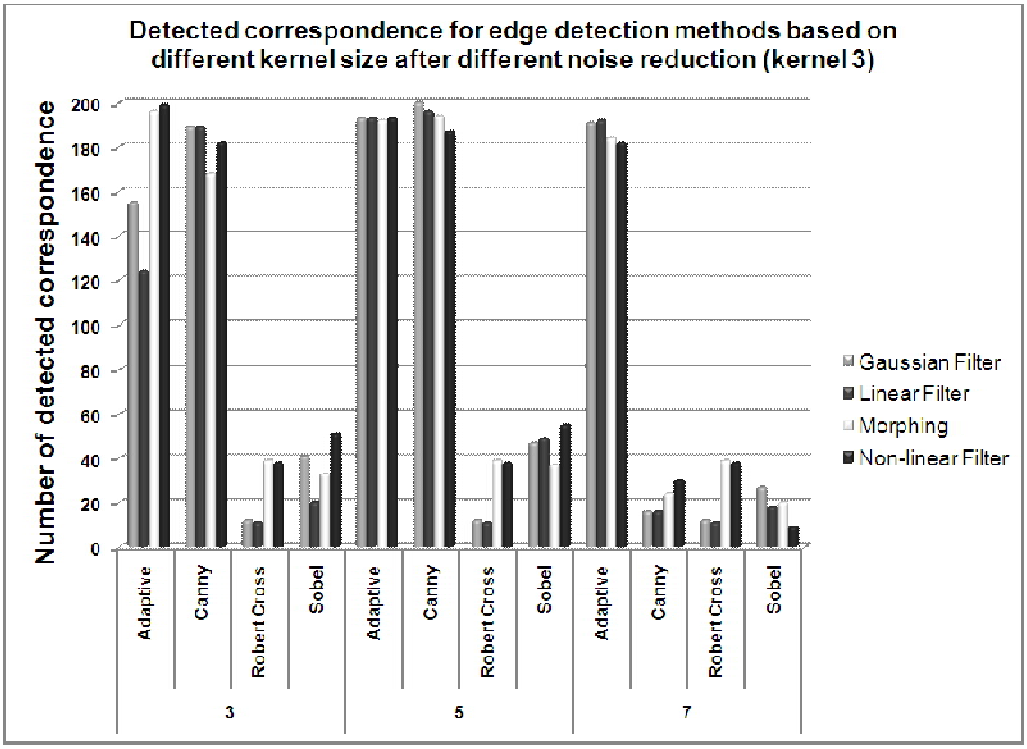


Figure 4.7: detected contours with their correspondence for different edge detection method base on their kernel size after different noise reduction method base on kernel size 3

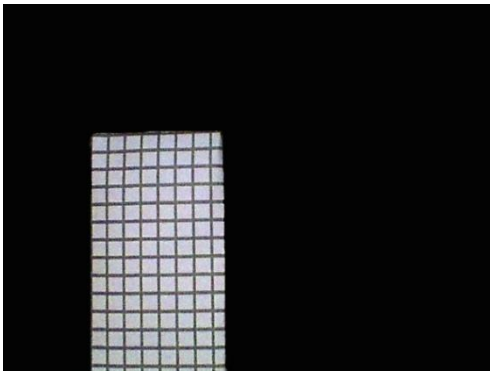
Overall the adaptive threshold with morphing operation shows best output relatively based on the detected object and processing time required for such detection. In addition the number of detection for such combination was the highest among other approaches. Finally based on the result of selected methods (adaptive threshold kernel size 3 with noise reduction method of morphing with kernel size 3) is selected as default method to be used for proposed approach to correspondence problem. This method was tested and compared with other preprocessing methods on different sets of images (for example: Figure 4.8) and the selected method shows the best performance among the other methods. Selected approach also requires minimum user influence and relies on automatic detection which improves the overall processing time and lead to faster detection.

4.2 Correspondence

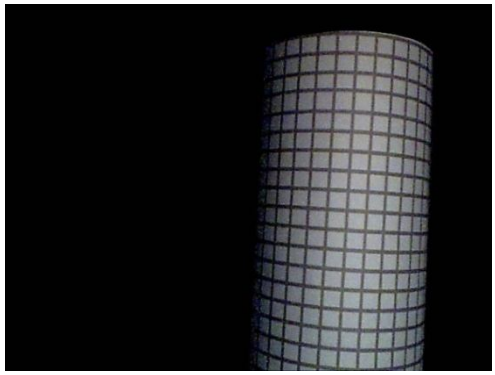
In this section it compares the result for different local methods based on the detection method output and processing time required by each of them. There are 4 samples of

data being processed in this section (Figure 4.8). Set 1 is the flat surface used in previous section for preprocessing testing. Set 2 is a curve structure with rectangular texture. Set 3 is a more complex image. Set 4 is a plant image. These data sets will be compared for 3 different algorithms (block matching [54], gradient base optimization [55] and level-based approach). This comparison is based on processing time required, and visual evaluation of output result. The reason for selecting visual verification here is that since local method (especially gradient base optimization) does not contain detail for every pixel, the suggested statistical evaluation is not possible [10].

In this section we focus on the representation of depth (disparity map or flow diagram) rather than the depth itself. This is due to lack of camera information (such as the focal length, distance of the cameras from each other) for data set 3 and 4 to transfer disparity map or flow diagram to real world measurement.

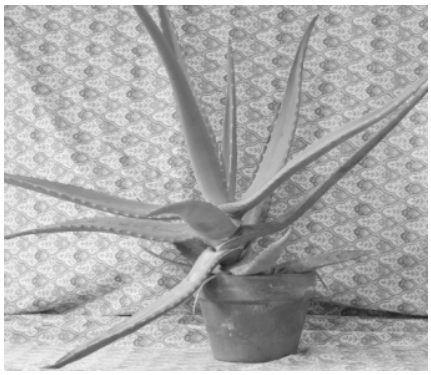


Set 1



Set2

Set3 [56]



Set4 [57]

Figure 4.8: image sets for comparison of correspondence method

Another important note is that due to implementation of gradient base optimization the result is represented in flow diagram rather than disparity map. The difference is

that instead of representing depth base on the intensity (brighter color represent contours closer to camera) in optical flow the flow of selected pixel showed by motion line, where the star point represent corners detected in second image and start of the line is the location of corners in the first image.

Figure 4.9 shows the disparity map generated by block matching for dataset 1 which is a flat surface. In this image only a slim part of the surface is detected, this is mainly due to the self similarity in texture. Another issue that could affect the output is that due to uncalibrated camera and the effect of camera distortion on the output result.

Figure 4.9: Disparity map generated by block matching algorithm (set 1)

Figure 4.10 shows the optical flow of the selected point in first image. The line in that figure shows the displacement of the corners in second image. As it was discussed in chapter 2 the problem with gradient base optimization is that, this method highly relies on the intensity and as the texture in this dataset have large self similar area this method also does not show an accurate result. The detected correspondences were limited and also there were many miss-matches. Another issue in this process is that if the disparity increase, the possibility of mismatch in this method also increases. As it shown in this figure some of the correspondences refer to outside of image boundary.

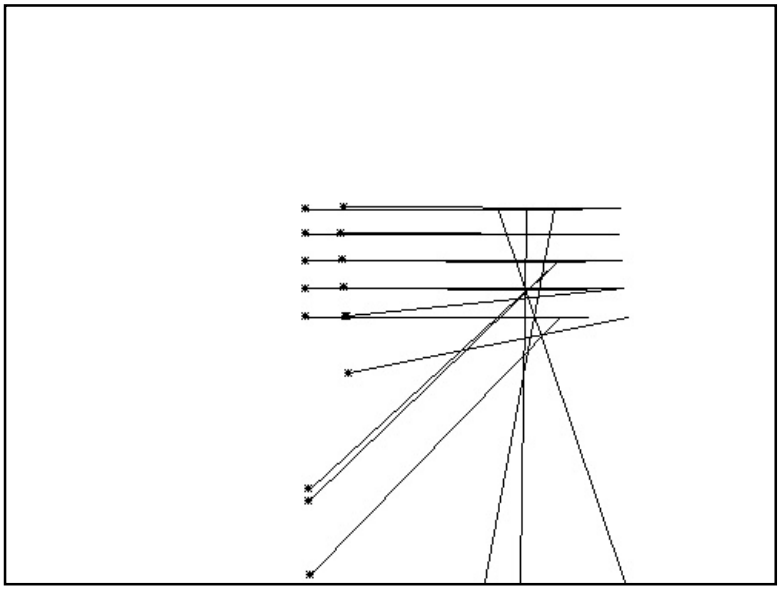


Figure 4.10: Optical flow detected base on gradient base optimization (set 1)

Figure 4.11 which is the output of proposed approach shown remarkable output. In this result only the detected contours is displayed, but the output shows many correct correspondence result.

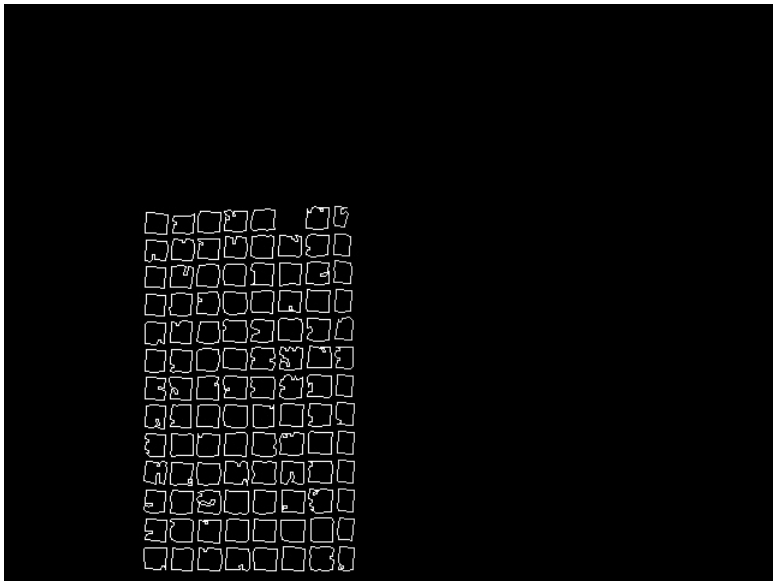


Figure 4.11: Disparity map generated by level-based approach (set 1)

Figure 4.12 shows the block matching output, the result in this output is clearer which represent the depth with darker gray. This improvement could be due to the size of the surface and as it has been discussed earlier the less self similarity due to curve surface. The light affects the texture and increase the possibility of correct match for

template in search region. Even though this result has better representation but again due to some self similar area there are many mismatch region in the output result.

Figure 4.12: Disparity map generated by block matching algorithm (set 2)

Figure 4.13 which is the representation of correspondence result for gradient based optimization shows many faulty results. This could be due to the effect of light reflection from the curve surface and changes in camera position. Since this method does not consider that the changes only occurs according to horizontal line, as block matching method does, the output have many faulty results.

Figure 4.14 is the disparity map of this curve surface based on the view point of second camera. In this figure the most left area of surface shown in darker color as it represents the more distance from the right side camera. This representation also shows some mismatch but this result can detect more than 90% correct result and around 200 contours detection which is based on the objective of this project still acceptable.

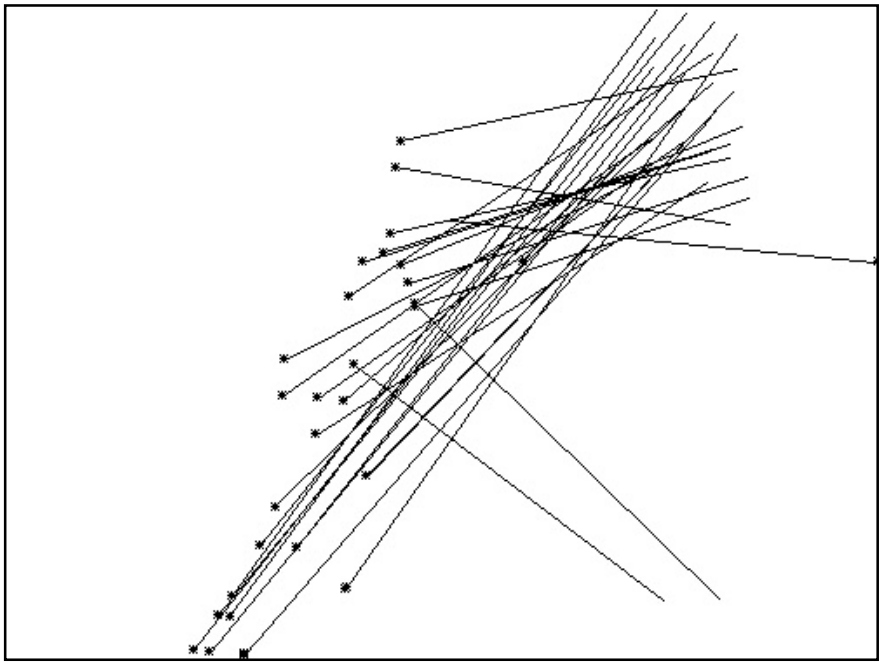


Figure 4.13: Optical flow detected base on gradient base optimization (set 2)

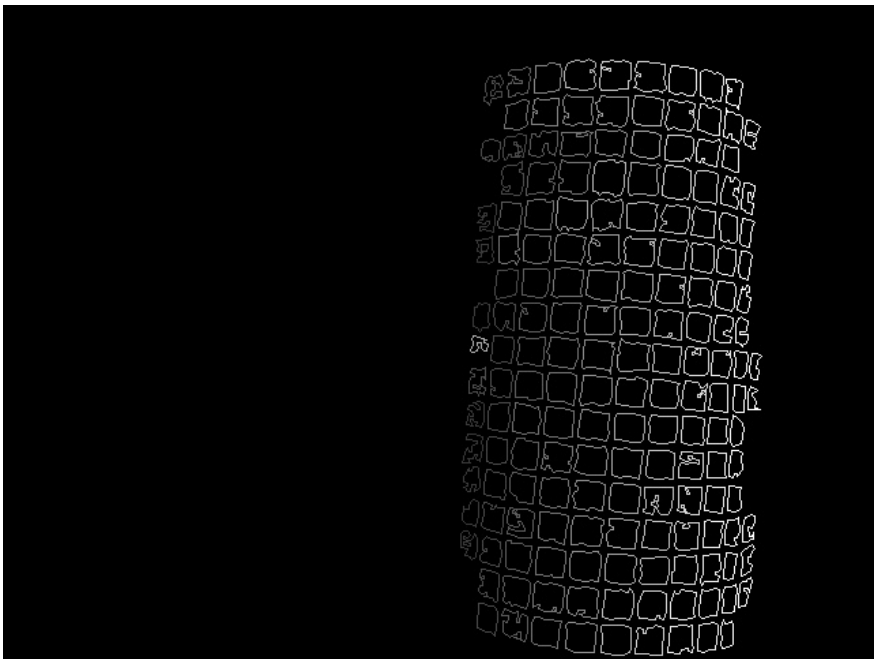


Figure 4.14: Disparity map generated by level-based approach (set 2)

Figure 4.15 shows the output result for dataset 3 which is a cave image and consider complex image due to high detail and smaller texture and the depth in the real-world is much larger compared to previous datasets. The result focused on the center of image and the surrounding considered as background. There are many mismatch

output, for example the deep area of the curve shown in brighter color which has been detected wrongly.

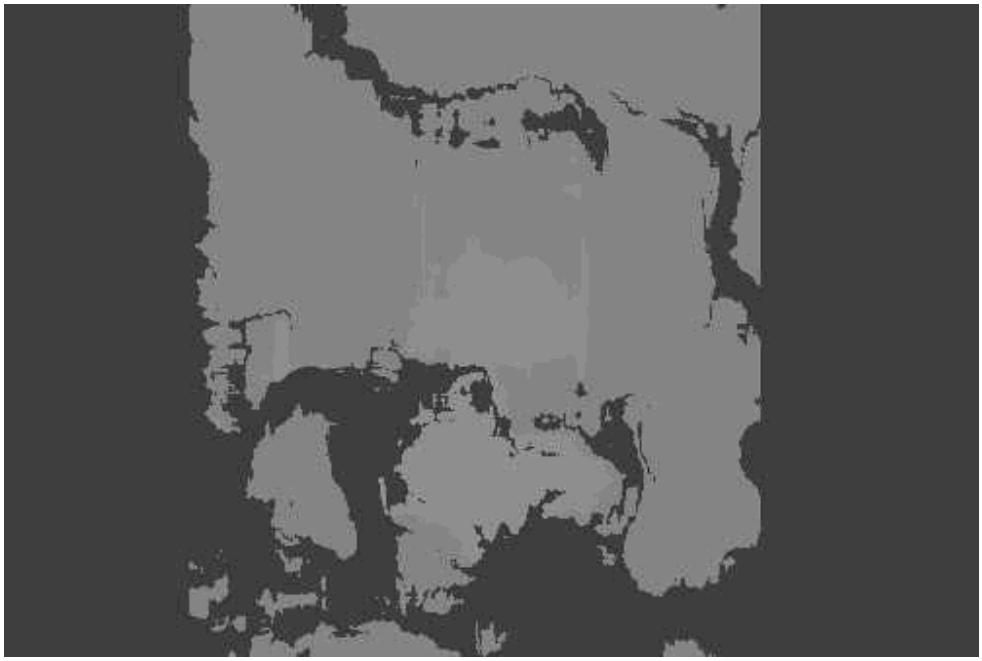


Figure 4.15: Disparity map generated by block matching algorithm (set 3)

Figure 4.16 shows the output of gradient based optimization and this result represents the disparity more accurately due to reduction in disparity length and also the images are closer in terms of intensity due to large area of the real-world view. As the result gets closer to the center of image (further in cave) the displacement increased between the two images.

Figure 4.17 shows the output of level-based approach. In this set of results the focus is on larger contours and similar to gradient based optimization this result represents depth as it was expected. As the contours are closer to center of the image the intensity decreased. The advantage of this result over the gradient based optimization is better representation of depth and due to relying of the contours instead of pixels; it has better representation of surfaces and their depth in the image.

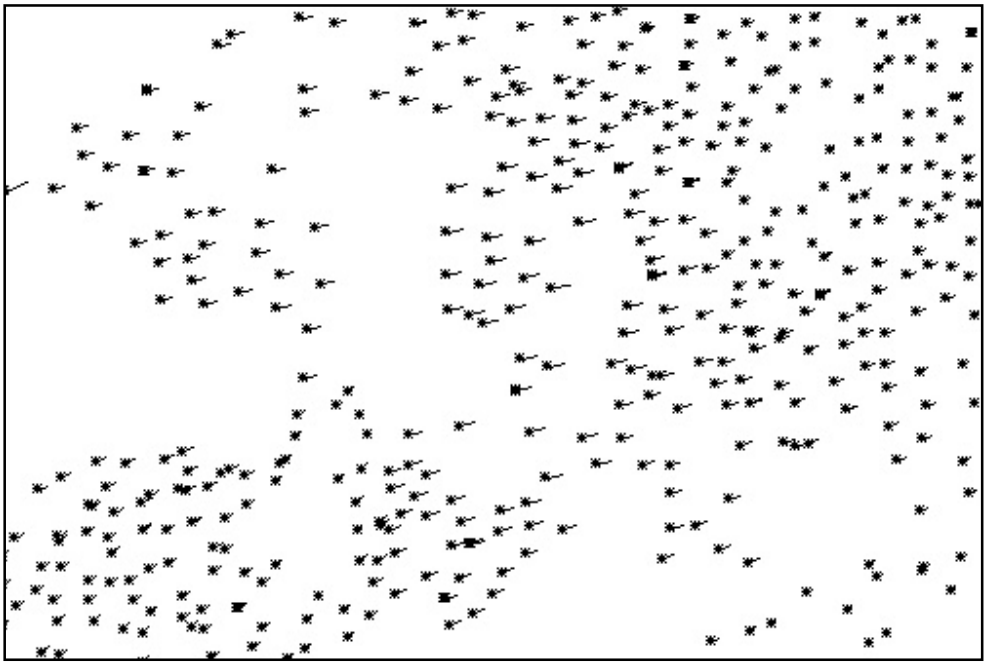


Figure 4.16: Optical flow detected base on gradient base optimization (set 3)

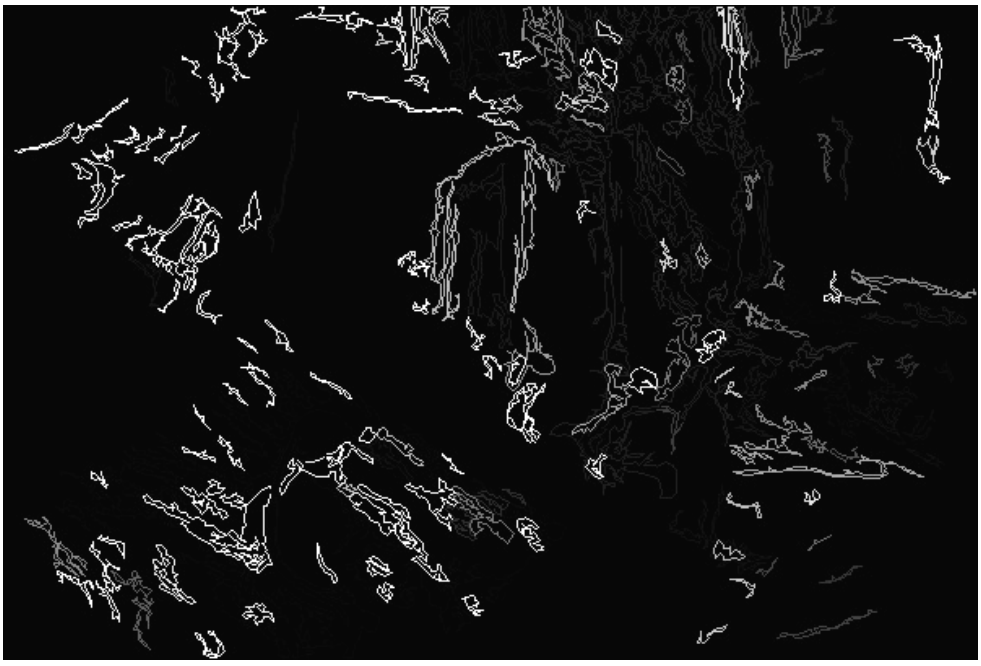


Figure 4.17: Disparity map generated by level-based approach (set 3)

Figure 4.18 shows the final sets of data processed for correspondence. In this approach the block matching method fails to detect the correspondence correctly. In this method the boundary of plant detected correctly but the overall depth has been detected wrongly as it put the background in front of the rest of surfaces .

Figure 4.18: Disparity map generated by block matching algorithm (set 4)

Figure 4.19 shows the optical flow result of the dataset 4 which is calculated based on gradient based optimization. In this result many of the background areas are detected as featured corner but the correspondence result is not accurate. This output result contains many features but due to large group of self similar area this method also fails to detect correctly. The point of this set of images is the representation of occlusion in which these methods fail to detect.

Figure 4.20 is the output result of set 4 images after processed with level-based approach. In this method some of the leaves are detected and most of their correspondence is detected correctly except one. Some of the background texture is also included in the result which is highly due to the background texture. In other hand as it has been discussed before the local methods are less sensitive to the occlusion and based on what is shown in this section these methods fail to detect the occluded area.

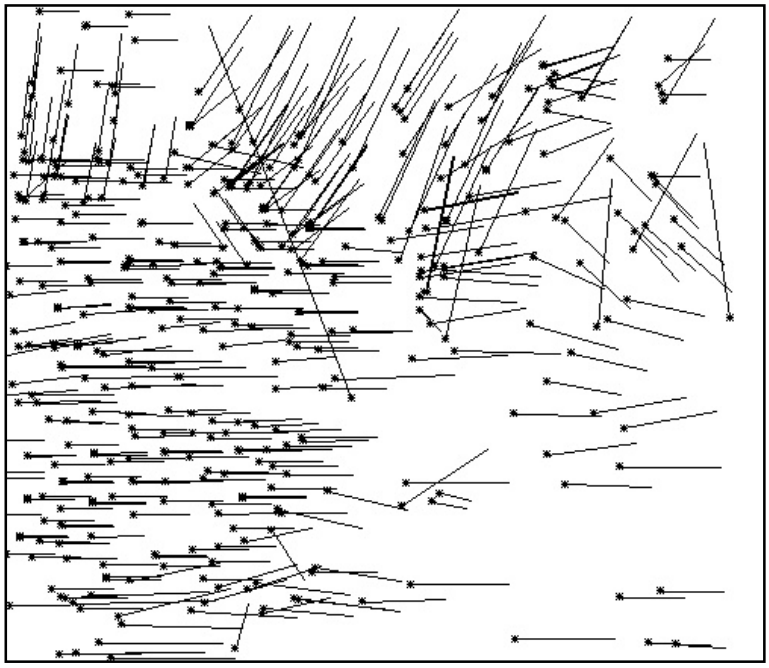


Figure 4.19: Optical flow detected base on gradient base optimization (set 4)

Figure 4.20: Disparity map generated by level-based approach (set 4)

Figure 4.21 shows the graph, representing time required to process above datasets with different algorithm discussed in this section. As it is shown level base approach requires less processing time compared to other methods regardless of the number of contours and corners detected (Table 4.3). Time for level-based approach is highly depends on the input image size. On the other hand, for gradient based optimization

the processing time is highly depend on the detected corners. As it is shown in that figure the last two datasets requires much more processing time compared to two other methods.

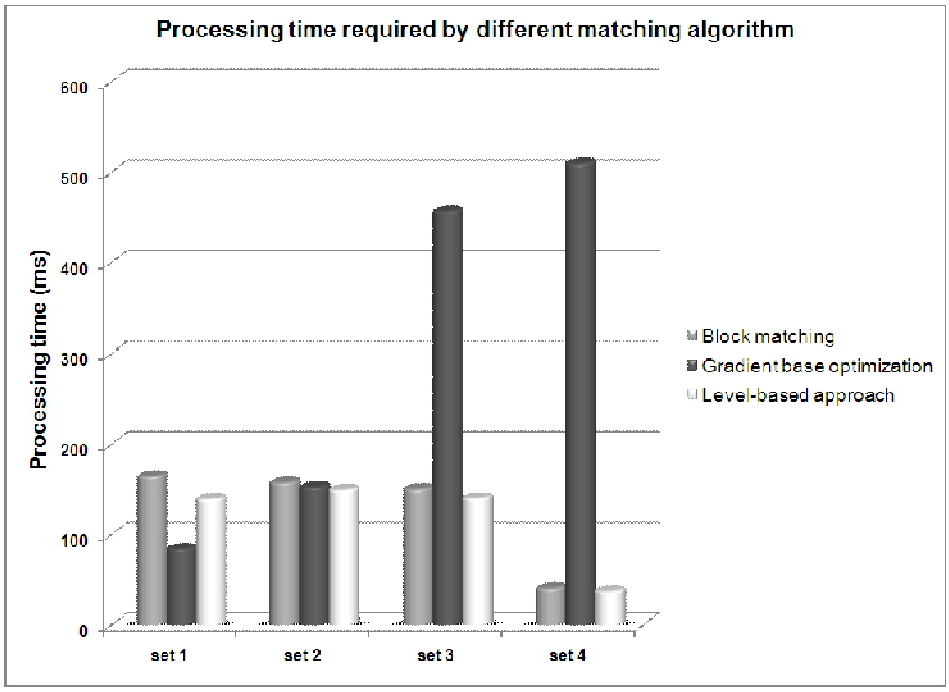


Figure 4.21: time (ms) required to process datasets with different local method correspondence

Table 4.3: time (ms) require to process datasets with different local base correspondence method

	set 1	**set 2**	**set 3**	**set 4**
Block matching	165	158	151	41
Gradient base optimization	85	153	457	510
Level-based approach	140	150	141	38

4.3 Levels Detection

Level-based approach relies on the hierarchical model to reduce search area. However in most of situations the depth of level goes up to two layers. Figure 4.22 shows two sets of line. The thin lines represent edges in image (lines with 1pixel thickness). The thicker lines represent contours in image (Lines with 2pixel thickness). The contour levels in this figure represented based on their color. Brighter colors represent inner

levels. In this figure there are two levels separated by dark thick lines and bright thick lines as child of dark lines. Please refer to section 0(page 40) for better understanding of levels and contours.

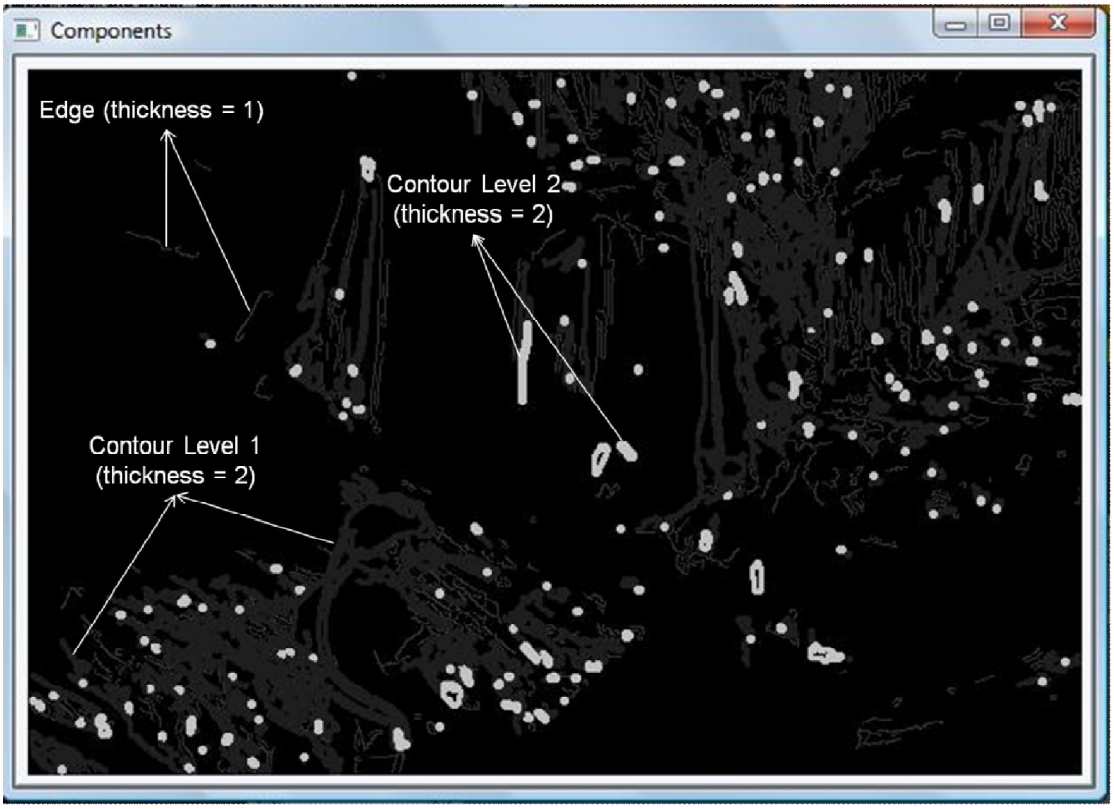


Figure 4.22: Edge and contours detected in cave image.

4.4 Reconstruction

After generating disparity map, it is possible to reconstruct a 3D model for images with available camera property. Unfortunately for examples in previous section this information was not available for the images taken from internet. Due to this in this section the reconstruction has been performed only on the images by known camera.

There are two images used in this section (Figure 4.23) the images on the top (Figure 4.23.a, Figure 4.23.b) represent left view and right view of flat surface accordingly. The bottom images (Figure 4.23.c, Figure 4.23.d) represent curve surface on the left and right view.

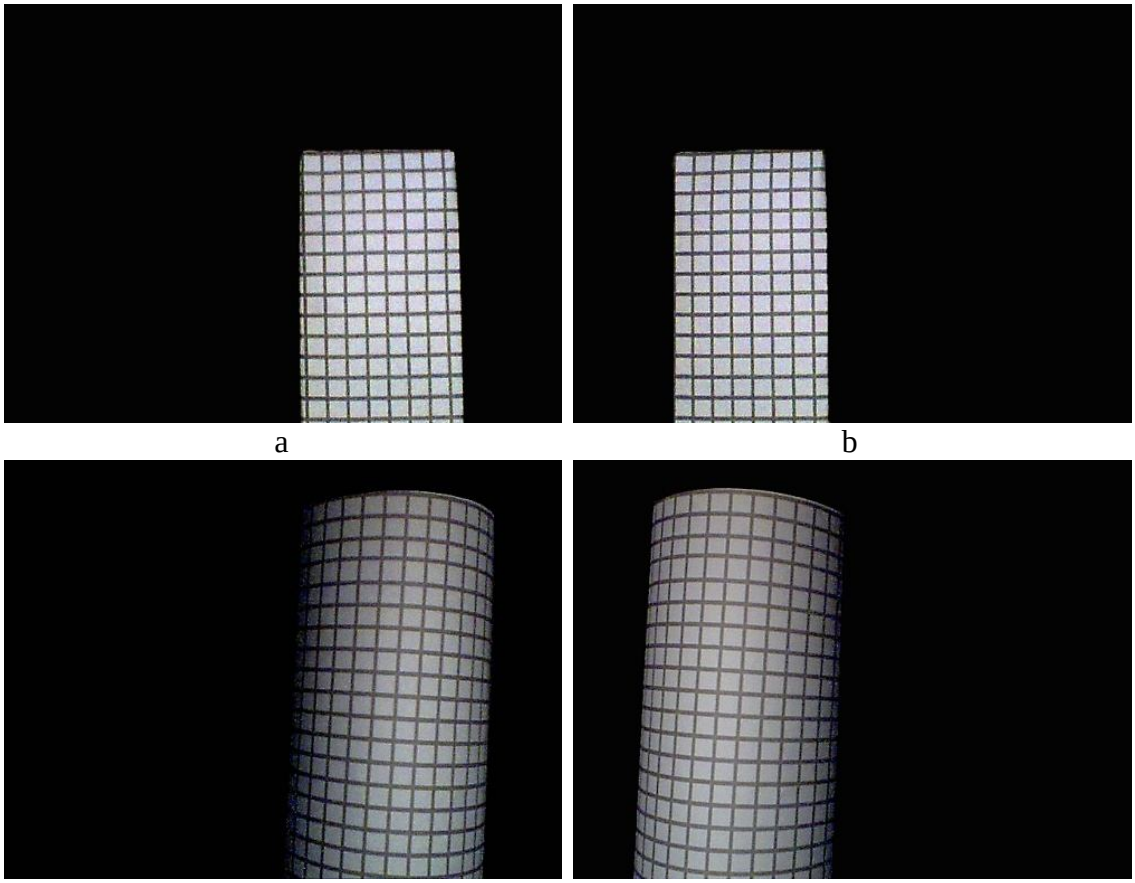


Figure 4.23: Base images, (a) flat surface left view, (b) flat surface right view, (c) curve surface left view (d) curve surface right view

In this approach the contours with their respective disparity is extracted. The disparity value using formula which has been discussed in chapter 2 (refer to 2.6 page 24) will be converted to depth (pixel) and by using Delaunay triangulation the 3D view will be reconstructed.

Figure 4.24 and Figure 4.25 shows the reconstruction result. In this result due to higher quality of images and sharper edges the result consists of all the points in the surface. Figure 4.25 shows small elevation which is due to position of surface to the right image. The base image for reconstruction in this example is the right image and since the right side of the surface is closer to the right camera then this elevation occurred.

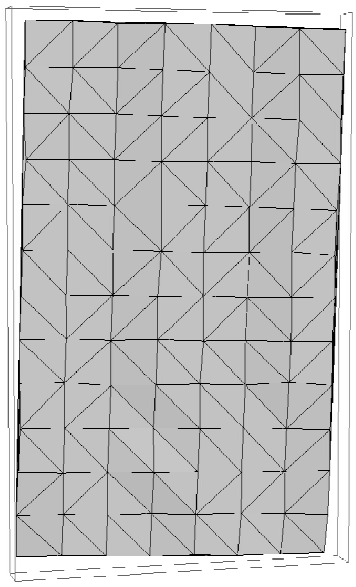


Figure 4.24: Flat surface reconstruction (front view)

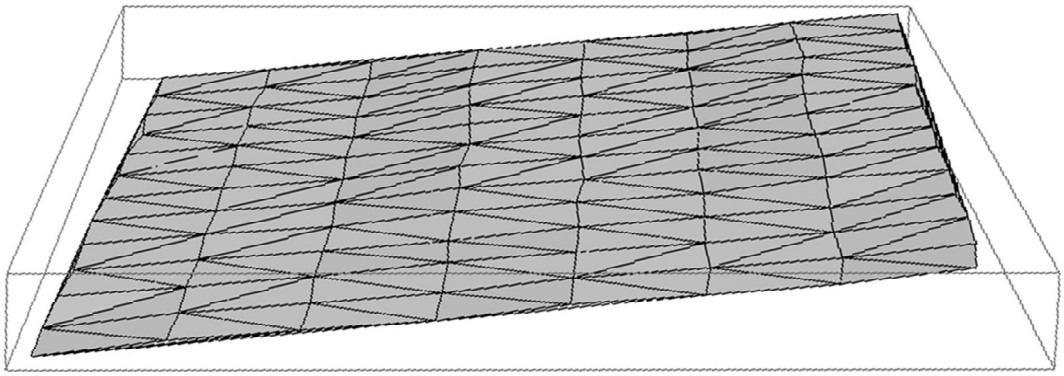


Figure 4.25: flat surface reconstruction (top view)

Figure 4.26 shows the curve surface reconstruction. In this image due to low quality and distance between the textures, some areas in the shape such as Figure 4.26 a represent misdetection for reconstruction. Another reason for faulty detection is generating data for left view camera in which the contours on the right side are misplaced. Overall this distortion on reconstruction is due to misdetection of contours for one pixel only. The area shown in Figure 4.26 b is the bottom of the view point and this irregularity is due to invisible point on the captured image (Figure 4.23.c).

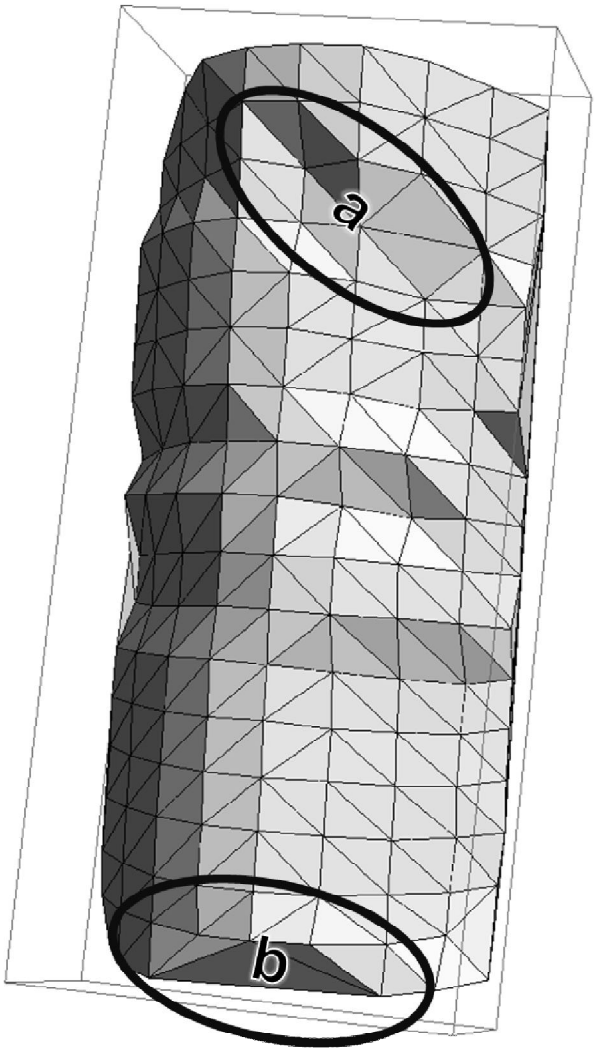


Figure 4.26: curve surface reconstruction (perspective view)

As a conclusion even though this method does not test for every object but selected images can represent the issues in each method. For gradient based optimization increase in overall disparity can cause some problem as well this approach does not provide good representation of depth. The self similarity could be another problem that both block matching method and gradient based optimization suffer from it. These results show another advantage of level-base approach which it is the reliance on the contours instead of pixel value which cause problem with the two other methods as it is shown above.

4.5 Irregular surface reconstruction

The object selected for reconstruction in the previous sections somehow is simple. The first reconstruction done in flat surface and the complexity extend to curve surface. These reconstructions either does not face any occlusion area or at most in discontinuity area of surface with limb occlusion. In this section the reconstruction surface is more complex with more occlusion in the surface. Figure 4.27 and Figure 4.28 shows the stereo view of the surface. The center of surface as it is shown in these figures is the most affected area with occlusion. This type of surface affects the stereo views with all three types of occlusion. Figure 4.29 shows the disparity map generated from these two views. In this map the effect of occlusion is clear. Despite the occlusions and non-verged stereo images, more than 60% of the surface is reconstructed in this example. This process only requires 92ms which is around 11 pair of images per second which is 22 frames per second for reconstructing disparity map.

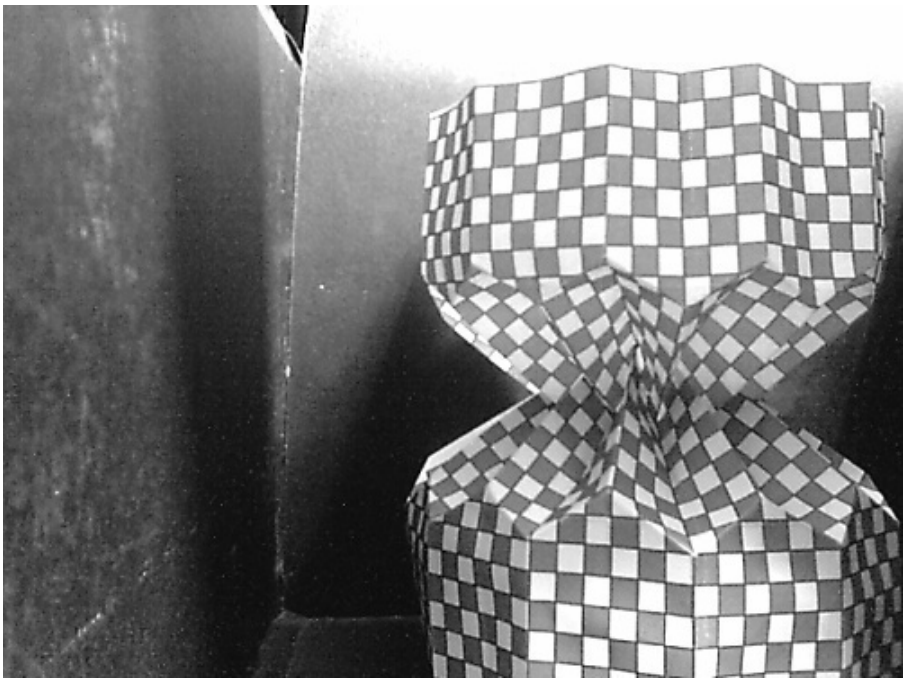


Figure 4.27: left camera view of the surface

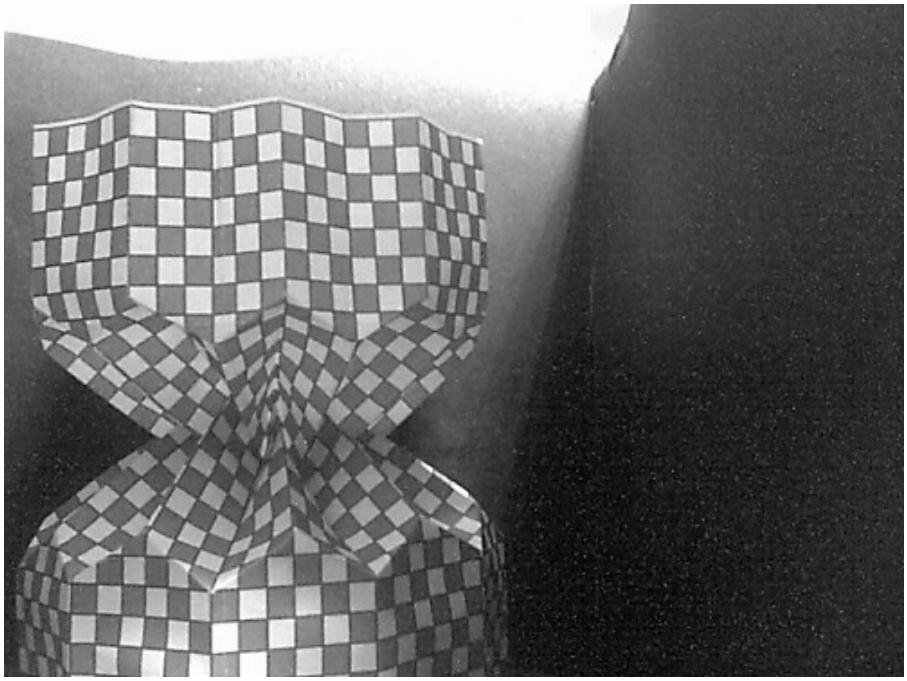


Figure 4.28: Right camera view of the surface.

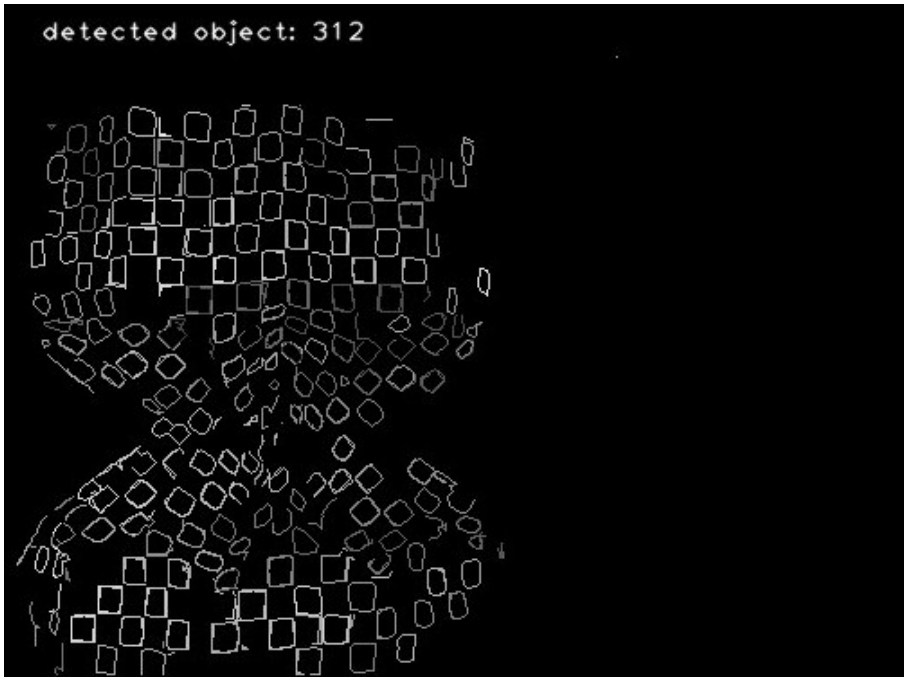


Figure 4.29: disparity map generated from the stereo images.

Figure 4.30 shows the reconstructed surface from the disparity map. In this reconstruction Delaunay triangulation is used. The disadvantage of this method is that Delaunay triangulation tries to connect every point in the image to some other point.

These causes the points near to the middle of the image connect to each other despite the fact that in the view those point are not connected to each other. Figure 4.31 also shows the side view representing the flow of the surface for visual conformation with actual surface.

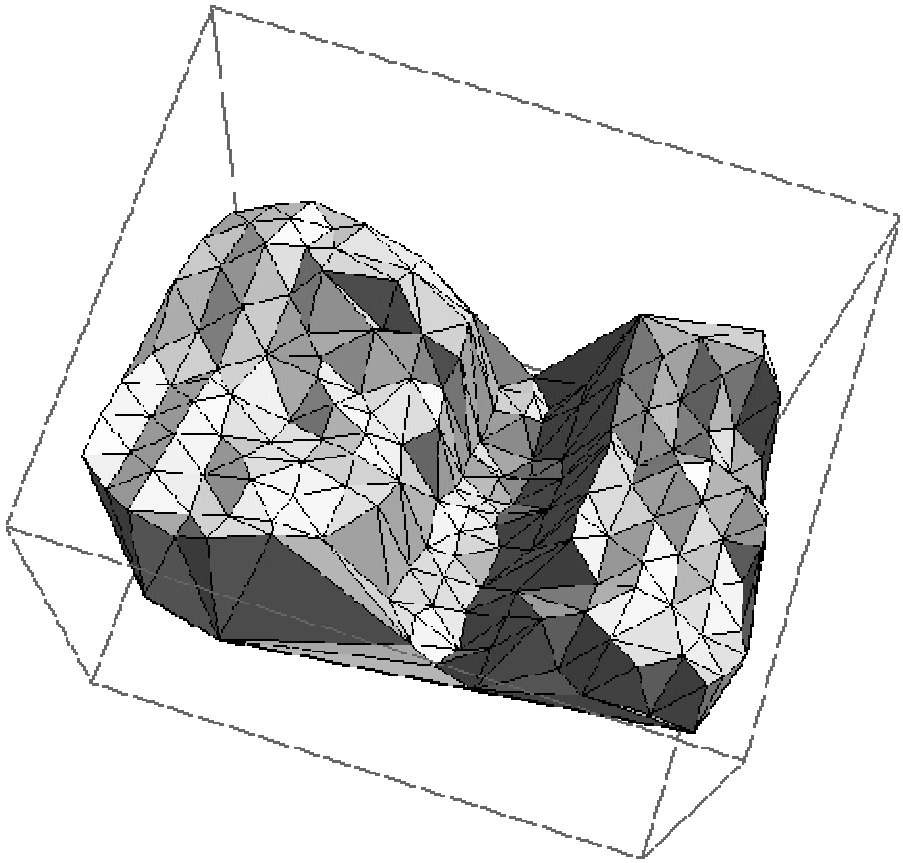


Figure 4.30: perspective front view of the reconstructed surface.

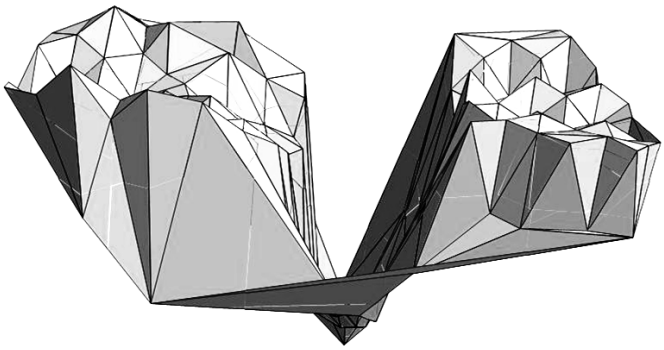


Figure 4.31: 3D reconstruction surface, side view

4.6 Summary

Each steps involved in 3D reconstruction have been discussed and results of basic image processing steps are displayed. Morphing operation in these results shows much more effect on the noise compared to other methods. Furthermore adaptive threshold was more promising compare to other approaches in the edge detection. Adaptive threshold with morphing operation shows much more correct matches in the result and it has a great effect on the processing time compare to other methods.

Disparity map generated from different local matching are compared with each others. These results clearly show the improvement of proposed approach over other local methods.
The proposed method can detect more correct match in shorter time compare to the other local methods.

Finally, 3D reconstruction from three sets of stereoscopic images are generated and discussed. The effects of different occlusion situation on captured image were shown. Occlusion as one of the main problem with the local matching method requires more time and efforts. Hopefully in future improvement of this approach this problem would be considered.

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

Level-based matching is presented as a solution to correspondence problem. Proposed correspondence method matches the points in 2D images, representing same physical points in real world. Disparity map then can be extracted from matched points. Resulting disparity map is then used to reconstruct the 3D model using camera matrix and scaling disparity map.

Level-based correspondence enables fast matching process with acceptable accuracy. This local matching method relies on feature extraction using edge detection. Detected edge can provides information about contour in image. Contours provide a relation between corners in image. This relation is immune to transformation and intensity changes in the scene. Immunity to such changes provides more accurate matching result compare to other local methods.

To increase accuracy, proposed method also based matching on contours with respect to their relation with the parent contour and relation of contours within same level and same parent. The contour then grouped based on hierarchical manner. The hierarchical grouping reduce the search area, allowing for faster matching process (about 10% faster than current local methods). This process allows real-time (50-60 fps) matching for small size images (320×240) and near real-time (15-20 fps) for larger images (640×480). This system provides more accurate result regardless of image quality. Unfortunately there is no available method for comparing local methods accuracy except visual confirmation. In this thesis some result generated with other local method approaches to be compared with proposed method.

5.2 Future work

Proposed method outperforms current local methods. Current local methods have issue with uniform texture where proposed method resolves this issue with hierarchical approach to contours. Even though uniform texture problem resolve by proposed approach, however the occlusion [8] handling as one of the main issue in local methods still remain unresolved. Among occlusion area half occlusions [58] are easier to detect. This issue is discussed in detail in chapter 3 and the effect of occluded area in the result is shown in chapter 6 where it has been shown how three different occlusions can affect the generated result. Even though occlusion handling is discussed in global methods [38, 59] [41] but real-time processing is not feasible in these approaches. The author suggests further improvement on occlusion handling for local method with possibility of real-time processing.

In other hand, since this method relies on contours and edges in the image, objects with low textures may not reconstructed accurately. There are many methods proposed to resolve such issue. This method with the help of depth from intensity approach can perform much more detailed result.

REFERENCE

[1] Q. Liu, *et al.*, "3D Construction of Endoscopic Images Based on Computational Stereo," in *Bioengineering Conference*, 2006, pp. 69 - 70.

[2] A. Woodward, *et al.*, "A 3D video scanner for face performance capture," in *Image and Vision Computing New Zealand, 23rd International Conference*, nov. 2008, pp. 1-6.

[3] C. Zach, *et al.*, "Scanline Optimization for Stereo on Graphics Hardware," in *Third International Symposium on 3D Data Processing, Visualization, and Transnission*, Jun 2006, pp. 512 - 518.

[4] J. Byrne, *et al.*, "Stereo based obstacle detection for an unmanned air vehicle," in *IEEE International Conference on Robotics and Automation*, may 2006, pp. 2830 - 2835.

[5] S. Kimura, *et al.*, "A convolver-based real-time stereo machine (SAZAN)," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, p. 463 Vol. 1.

[6] S. T. Barnard and M. A. Fischler, "Computational Stereo," *ACM Computing Surveys (CSUR)*, vol. 14, pp. 553 - 572, Dec 1982.

[7] U. R. Dhond and J. K. Aggarwal, "Structure from stereo-a review," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1489-1510, Nov/Dec 1989.

[8] R. C. Chung and R. Nevatia, "Use of monocular groupings and occlusion analysis in a hierarchical stereo system," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun 1991, pp. 50-56.

[9] A. Koschan, "What is New in Computational Stereo Since 1989: A Survey of Current Stereo Papers," Technical Report1993.

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

[11] D. Scharstein and A. Blasiak. vision.middlebury.edu/stereo/eval [Online]. Available: http://vision.middlebury.edu/stereo/code/

[12] W. Zeng-Fu and Z. Zhi-Gang, "A region based stereo matching algorithm using cooperative optimization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.

[13] M. Bleyer, *et al.*, "Surface stereo with soft segmentation," in *Computer Vision and Pattern Recognition 2010*, 2010.

[14] KITAMURA, *et al., Three-dimensional data acquisition by trinocular vision* vol. 4. Zeist, PAYS-BAS: VSP, 1990.

[15] D. Terzopoulos, "Multilevel computational processes for visual surface reconstruction," *Computer Vision, Graphics, and Image Processing*, vol. 24, pp. 52-96, 1983.

[16] G. Hu and G. Stockman, "3-D surface solution using structured light and constraint propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 390-402, 1989.

[17] R. C. González and R. E. Woods, "Smoothing Linear Filters," in *Digital image processing*, second ed: Prentice Hall, 2001, p. 119.

[18] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629-639, Jul 1990.

[19] R. C. González and R. E. Woods, "Order-Statistics Filters," in *Digital image processing*, second ed: Prentice Hall, 2001, p. 123.

[20] R. C. González and R. E. Woods, "Some Applications of Gray-Scale Morphology " in *Digital image processing*, second ed: Prentice Hall, 2001, p. 556.

[21] T. Acharya and A. K. Ray, " EDGE DETECTOR," in *Image Processing Principles and Applications*, First ed: A JOHN WILEY & SONS, MC., 2005, p. 135.

[22] R. C. González and R. E. Woods, "Basic Adaptive Thresholding " in *Digital image processing*, second ed: Prentice Hall, 2001, p. 600.

[23] Moravec and H. Peter, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Stanford University, California1980.

[24] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147-151.

[25] T. Lindeberg, "Grey-level blob," in *Scale-space theory in computer vision*, ed: kluwer, 1993, p. 166.

[26] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, pp. 91-110, 2004.

[27] H. Bay, *et al.*, "Surf: Speeded up robust features," in *Computer Vision and Image Understanding*, 2006, pp. 404-417.

[28] Z. Zhang. (2000, Nov) A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1330-1334.

[29] J. Heikkila. (2000, oct) Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1066-1077.

[30] B. K. P. Horn. (1991, Oct) Relative Orientation Revisited. *Journal of the Optical Society of America*. 1630-1638.

[31] R. I. Hartley, "In defense of the eight-point algorithm," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun-1997, pp. 580-593.

[32] G. L. Mariottini and D. Prattichizzo, "Epipolar geometry estimation for contour-based visual servoing," in *World Automation Congress*, Jul 2004, pp. 529-534.

[33] D. N. Bhat and S. K. Nayar. (1998, Apr) Ordinal measures for image correspondence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 415-423.

[34] O. Faugeras, *et al.*, "Real Time Correlation-Based Stereo: Algorithm, Implementations and Applications," Technical Report 2013Aug 1993.

[35] R. Zabih and J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence," in *Proceedings of the Third European Conference-Volume II on Computer Vision*, 1994, pp. 151 - 158.

[36] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proc. Int'l Joint Conf. Artificial Intelligence*, 1981, pp. 674-679.

[37] V. S. Kluth, *et al.*, "Global Least Squares Matching," in *International Geoscience and Remote Sensing Symposium*, 1992, pp. 1615-1618.

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

[39] I. J. Cox, *et al.*, "A maximum likelihood stereo algorithm," *Computer Vision and Image Understanding*, vol. 63, pp. 542-567, may 1996.

[40] Y. Boykov, *et al.*, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222-1239, Nov 2001.

[41] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Eighth IEEE International Conference on Computer Vision*, 2001, pp. 508-515.

[42] T. Tao, *et al.*, "A fast block matching algorthim for stereo correspondence," in *IEEE Conference on Cybernetics and Intelligent Systems*, Sep 2008, pp. 38-41.

[43] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 920-932, Sep 1994.

[44] D. Geiger, *et al.*, "Occlusions and binocular stereo," *International Journal of Computer Vision*, vol. 14, pp. 211-226, 1995.

[45] A. Fusiello, *et al.*, "Efficient stereo with multiple windowing," in *Computer Vision and Pattern Recognition*, Jun 1997, pp. 858-863.

[46] Y. Boykov, *et al.*, "A variable window approach to early vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1283-1294, Dec 1998.

[47] H. Hirschmuller, "Improvements in Real-Time Correlation-Based Stereo Vision," in *Stereo and Multi-Baseline Vision*, 2001, pp. 141-148.

[48] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *ARTIFICAL INTELLIGENCE,* vol. 17, pp. 185-203, 1981.

[49] A. W. Gruen, "Adaptive Least Squares Correlation: A Powerful Image Matching Technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography,* vol. 14, pp. 175-187, 1985.

[50] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision,* vol. 2, pp. 283-310, Jan 1989.

[51] A. Singh, "Optic Flow Computation: A Unified Perspective.," *IEEE Computer Society Press,* 1991.

[52] B. Berkels, *et al.*, "Reconstructing Optical Flow Fields by Motion Inpainting," presented at the Proceedings of the 7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, Bonn, Germany, 2009.

[53] F. Steinbrucker, *et al.*, "Large displacement optical flow computation withoutwarping," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 1609-1614.

[54] M. Agrawal, *et al.*, "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching," presented at the ECCV (4), 2008.

[55] K.-S. Kim, *et al.*, "Real time face tracking with pyramidal Lucas-Kanade feature tracker," presented at the Proceedings of the 2007 international conference on Computational science and its applications - Volume Part I, Kuala Lumpur, Malaysia, 2007.

[56] j. wattie, "Cave Frames Page," in *Stereoscopic Photography, Picture Gallery Contents.*, ed, 2006.

[57] D. Scharstein and R. Szeliski, "vision.middlebury.edu/stereo/data/scenes2006/FullSize/zip-2views/," in *vision.middlebury.edu*, ed.

[58] G. Egnal and R. P. Wildes, "Detecting Binocular Half-Occlusions: Empirical Comparisons of Five Approaches," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, pp. 1127-1133, Aug 2002.

[59] P. N. Belhumeur, "A Bayesian approach to binocular stereopsis," *Int. J. Comput. Vision*, vol. 19, pp. 237-260, 1996.