3D RECONSTRUCTION OF A SCENE USING STEREO IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

FARİS SERDAR TAŞEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

JUNE 2008

Title of the Thesis : **3D Reconstruction of A Scene Using Stereo Images**

Submitted by **Faris Serdar Taşel**

Approval of the Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Özhan Ç. ULUATAM
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Mehmet R. TOLUN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. Abdül Kadir GÖRÜR
Supervisor

Examination Date : June 20, 2008

Examining Committee Members

Asst. Prof. Dr. Reza HASSANPOUR	(Çankaya Univ.)	[illegible]
Asst. Prof. Dr. Abdül Kadir GÖRÜR	(Çankaya Univ.)	[illegible]
Asst. Prof. Dr. Tansel ÖZYER	(TOBB ETU)	[illegible]

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	: Faris Serdar Taşel
Signature	: [illegible]
Date	: 20.06.2008

ABSTRACT

3D RECONSTRUCTION OF A SCENE USING STEREO IMAGES

Taşel, Faris Serdar
M.Sc., Department of Computer Engineering
Supervisor: Asst. Prof. Dr. Abdül Kadir GÖRÜR
June 2008, 86 pages

Two-dimensional photographs do not have depth-information. One solution to determine the location of an object in three-dimensional environment is to use more than one photograph as exposed by the nature. Extracting the depth information using stereo images is purposed in this thesis.

The thesis analyzes the steps and encountered problems in three-dimensional reconstruction process, explains the solutions exposed with the aid of epipolar geometry using some of the feature-based matching techniques. Stereo images which are taken from two calibrated cameras viewing the same scene are used to obtain estimated three-dimensional data. Pinhole camera model, epipolar geometry and its recovery are discussed; common stereo triangulation methods are explained in the chapters of the thesis. Besides, feature extraction and matching topics which are used for the reconstruction process are examined. Some of the methods used in the thesis are presented by algorithmic solutions and mathematical notations. Significant advantages and disadvantages of the methods are briefly discussed and encountered problems are tried to be challenged by fundamental approaches.

Keywords: Stereo, Feature extraction, Feature matching, Epipolar geometry

ÖZ

BİR SAHNENİN STEREO GÖRÜNTÜLER KULLANARAK
3B YAPILANDIRILMASI

Taşel, Faris Serdar
M.Sc., Bilgisayar Mühendisliği Bölümü
Danışman: Y. Doç. Dr. Abdül Kadir GÖRÜR
Haziran 2008, 86 sayfa

Günlük yaşamda kullanılan iki-boyutlu fotoğraflar derinlik bilgisi taşımazlar. Doğanın ortaya koyduğu gibi bir cismin üç-boyutlu ortamdaki yerini belirlemenin bir yolu da birden fazla fotoğraf kullanmaktan geçer. Bu tezde derinlik bilgisinin stereo resimler kullanarak çıkarılması amaçlanmıştır.

Tezde, üç-boyutlu yapılandırma işlemindeki karşılaşılan problemler ve adımlar analiz edilmiş, epipolar geometri yardımıyla, bazı özellik-tabanlı eşleme teknikleri kullanılarak ortaya konulmuş çözümler açıklanmıştır. Aynı sahneyi gören kalibre edilmiş iki kameradan alınan stereo görüntüler üç-boyutlu tahmini verinin elde edilmesi için kullanılmıştır. Tezin bölümlerinde, pinhole kamera modeli, epipolar geometri ve eldesi tartışılmış, yaygın stereo üçgenleme yöntemleri açıklanmıştır. Bunun yanısıra, yapılandırma işleminde kullanılan özellik ayıklama ve eşleme konuları incelenmiştir. Tezde kullanılan yöntemlerin bazıları algoritmik çözümler ve matematiksel ifadelerle sunulmuştur. Yöntemlerin önemli avantaj ve dezavantajları kısaca tartışılmış ve karşılaşılan problemler temel yaklaşımlarla giderilmeye çalışılmıştır.

Anahtar Kelimeler: Stereo, Özellik ayıklama, Özellik eşleme, Epipolar Geometri

ACKNOWLEDGMENTS

I thank Asst. Dr. Reza Hassanpour and Asst. Dr. Abdül Kadir Görür for all their esteemed guidance throughout the experimental and theoretical study of the thesis and their outstanding patience.

I thank the Management of Çankaya University for providing a peaceful and useful environment for completing the thesis.

I thank my colleagues, the research assistants of Computer Engineering Department of Çankaya University, Serkan Kefel, Pelin Güvenç, Burçak Sönmez for their brilliant ideas and the brain-storming meetings we had.

I render my special thanks to my parents for their great support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Almost all living creatures with two eyes in the front of their heads have the capability of perceiving the depth in the environment although they see in two-dimensional projection with their eyes. This perception gives them the ability of better determination of an object’s location. As being inspired by the nature, a similar ability can be given to the machines to make them determine the location of an object better to interact with it in a more reliable and successful way. Although humans use the depth information they perceive easily in their daily life, this process includes many complex computations.

The three-dimensional (3D) reconstruction involves determining the object locations in a 3D environment with their shape. It is a part of computer vision and it has many application areas such as robotics, industry, medical and military applications wherever a machine needs to get an interaction with its environment. It enables the machines to perceive their environment in 3D as humans. 3D reconstruction is also important for the other parts of computer vision such as object recognition and object tracking.

1.1. Stereo Images and Background of 3D Reconstruction

Since cameras are the eyes of machines, it is possible for a machine to compute the depth information by using its cameras. A camera -like an eye- supplies a two-dimensional image containing information of light intensity and frequency for each point on it. But, it has no information about depth. It is possible to compute it using more than one camera viewing the same scene such that one of them is slightly moved according to the other. In this situation, the images taken from these cameras will be almost the same but locations of the objects are shifted. The depth information is hidden in the amount of shifts between the images. The idea is to find some matches between them and measure the amount of the shift to compute the depth information. Some specific points in an image which have some different property with respect to the other points are called features and the process that involves finding specific matches between the feature points of images is called feature matching. Corners and edges of the objects that are available in the image are very commonly used features that give an idea about the shape of the object. They are also useful for distinguishing the objects in the image that is called segmentation. If the depth information corresponding to the features is known, segmentation process may become easier and more reliable.

Stereo vision means perceiving the environment using two images: left and right image that are also known as stereo images. Left and right image have a geometric relation between each other. This relation is called epipolar geometry and it is critical in 3D reconstruction. Stereo images have great similarities. A point on an object that appears in left image should also appear in right image if it is not obscured by another object. The same point probably appears at different locations since there is a movement or orientation difference between left and right camera. The difference in locations that the point appears is called disparity. If the disparity of a point is known, it can be used to determine the depth of the point with respect to the other points whose disparities are known. The matched points in stereo images indicate the point locations in the left and right image. Therefore, determining the disparity depends on finding a match between points that appear in stereo images.

As explained above, many 3D Reconstruction methods are based on finding correspondence among stereo images. Point matches together with the calibration data of cameras are required for determining 3D locations of points. Some other methods aim at finding some correspondence without calibration data to extract relative depth information called disparity map. The disparity map contains the information of depths of objects with respect to background or other objects in the scene but no exact location. There are also some reconstruction methods working on image sequences taken from calibrated cameras with different orientations to find correspondence between silhouettes of objects in order to extract a detailed 3D model of objects. The methods presented in the thesis intend to find some feature correspondence between stereo images taken from calibrated cameras.

1.2. Two-Dimensional Images

Regular cameras supply images which are a two-dimensional (2D) array of sample points. An ideal point has no size or dimension. On the contrary of ideal points, images consist of pixels that have width and height, indicating a region. Each pixel has average light information projecting on its region.

Gray-level images contain pixel points that each one of them has a value proportional to the average magnitude of the light within a certain frequency range weighted by the energy contained in a light beam due to its frequency. Therefore, each pixel indicates the energy of the light reflected on it, which is called intensity. Feature extraction methods are generally applied to gray-level images. Color images involve intensities in three kind of frequency range known as red, green and blue channels. Each color is a mixture of red, green and blue channels with some certain intensity. Gray-level images can be obtained by converting color images to gray level such that each pixel has the weighted average intensity value according to the channel. Gray-level images contain only one channel.

Another image type, other than gray-level and color images, is the binary image. Binary images may contain pixels having values either zero or one. This kind of images are generally used for logic operations or denoting existence of a feature. For instance, if a feature point exists at some pixel location in gray-level image, the corresponding pixel of the binary image showing feature points of the image has the value one.

1.3. 3D Reconstruction Process Using Calibrated Stereo Cameras

The 3D reconstruction process consists of calibration extraction, feature matching, epipolar geometry recovery and stereo triangulation sub-processes. The input of the reconstruction process is stereo images and the output is 3D position data of the points corresponding to the image points. Stereo image pairs should be taken from a stereo camera or two different cameras that are oriented to the same scene. Left and right images in a pair should be taken at the same time or have no movement of objects in the scene. Figure 1.1 shows the data flow diagram of 3D reconstruction process based on feature matching methods.

In the calibration process, stereo images or the images taken from the multiple cameras are used to extract the calibration data indicating camera properties such as camera positions, orientations, focus length, lens distortion coefficients. In fixed camera systems, the calibration process is necessary once assuming the focus length of the camera is also fixed. Calibration data is required by stereo triangulation and useful for feature matching step. The epipolar geometry restricts the correspondence between left and right images to facilitate feature matching process. It can be constituted using the calibration data but the feature matching process is quite sensitive for errors in fundamental matrix that is computed using epipolar geometry. Matched points between stereo images can also be used for the recovery of the epipolar geometry which is a feedback for the feature matching. Hence, it may be corrected during feature matching process. After the correspondence between all possible points is found, stereo triangulation process uses the point matches together with the calibration data to compute the 3D position of each matched point. Calibration data is not required by the feature matching process actually since the epipolar geometry can also be iteratively recovered beginning from a situation without known epipolar geometry. If the error rate in the epipolar geometry become small enough, the iteration stops and stereo triangulation is done using the final matched points. There is no need for re-calibration or recovery of the epipolar geometry as long as the camera properties remain constant.

In feature matching process, features in the images are extracted and matched. In feature-based matching techniques, the most commonly used feature points are the corners in the image and they can be matched relatively easier. Matched corners are used for the further steps to find more correspondence such as contour matching. Contours points involve edge points which are another feature extracted from images. Edge points of stereo images can be matched using contour matching methods. All matched feature point pairs between stereo images are used for stereo triangulation to compute 3D position of the points. 3D positions of feature points give the geometric shape information of objects with its location in the scene that is useful for object tracking and object recognition.

Stereo triangulation and epipolar geometry recovery methods involve pseudoinverse, least squares computations and rank adjusting, thus employ SVD (Singular Value Decomposition) to solve the problem. SVD is an important factorization of rectangular matrices. It is used for computing pseudoinverse, least squares fitting of data, matrix approximation, and determining the rank, range and null space of a matrix [1].

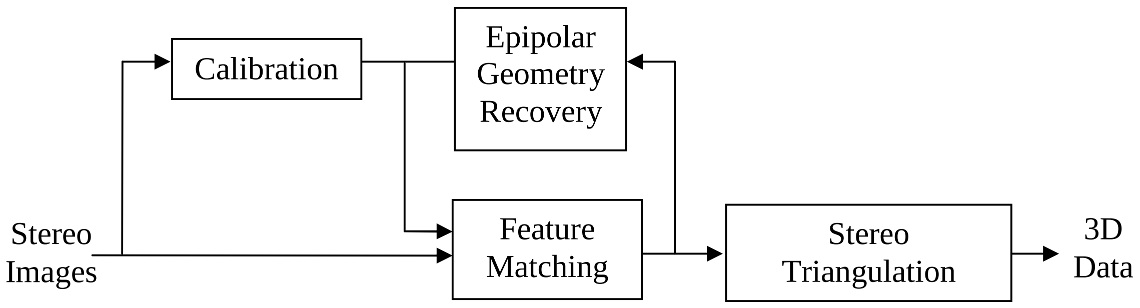


Figure 1.1 – 3D reconstruction process

1.4. Outline of Thesis

In the second chapter; camera geometry and model, the relations between two cameras, epipolar geometry and its recovery, common stereo triangulation methods are explained. Extraction methods of feature points are explained and discussed in the third chapter. In the fourth chapter, feature matching methods are explained. Application results of correlation-based matching and contour matching methods to the extracted feature points are discussed. Test images are used for the discussions about topics, results and behaviors of the methods. In fifth chapter, some test images and their reconstruction results are shown and discussed.

In the thesis, stereo triangulation methods are applied to stereo images using the matched point pairs assuming the camera calibration information is already present. Stereo images that have been taken from well-illuminated rooms are used for tests and examples.

CHAPTER 2

CAMERA GEOMETRY

Camera geometry comprises coordinate systems denoting camera positions and orientations within a spatial coordinate system, relations between cameras and the environment they are in. Camera geometry is used to represent the 3D coordinate data which the reconstruction process produces and pixels of the images taken from the cameras.

2.1. Pinhole Camera Model

Pinhole camera model can be ideally modeled as a linear projection from 3D space into 2D image [2]. The model defines four coordinate systems:

a) World coordinate system: The origin of the system is selected as a specific point in the space. The system has three dimensions.
b) Camera coordinate system: The origin of the system is on the camera center. Orientation of the camera with respect to the word coordinate system, determines the direction of Z-axis of camera coordinate system.
c) Ideal image coordinate system: This coordinate system is a 2D system which is on the image plane. The origin is the point where Z-axis of the coordinate system intersects with the image plane. It is also known as principal point.
d) Real image coordinate system: It is also on the image plane as well as the ideal image coordinate system, but on the contrary of the ideal image system, coordinates of any point in this coordinate system, are the pixel coordinates of that point on the image. X and Y axes of images are used

as they are perpendicular to each other. Ideally, this coordinate system is orthogonal. In practice, the angle between the axes may be slightly narrower or wider and the picture may be slightly distorted because of some small errors in the production phase of the camera. The angle is called skew angle.

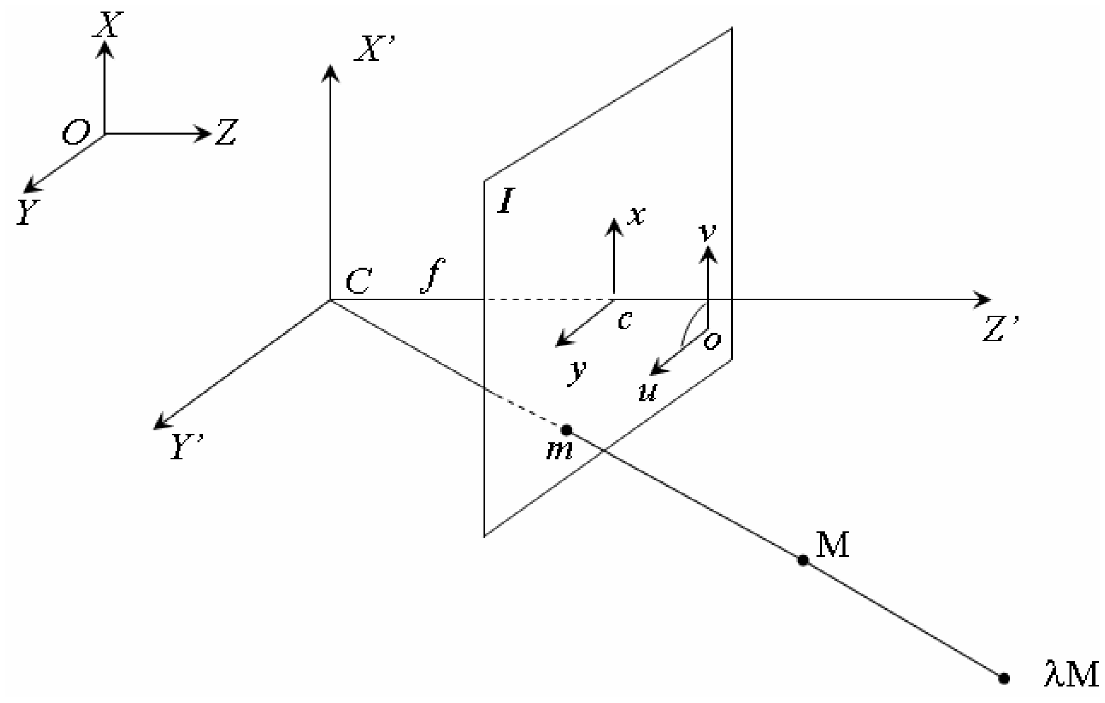


Figure 2.1 – Coordinate systems in pinhole camera model

In Figure 2.1, the origin and axes of the world coordinate system are shown as (*O*, *X*, *Y*, *Z*). Camera coordinate system is shown as (*C*, *X'*, *Y'*, *Z'*). "*I*" is the image plane. Ideal image origin and axes are (*c*, *x*, *y*). Real image origin and axes are (*o*, *u*, *v*). The image plane *I* is parallel to (*X'*, *Y'*) plane. Therefore, *Z'* axis intersects *I* at the principal point (*c*) and the angle between *Z'* axis and *I* is right angle. The distance between camera center (*C*) and principal point (*c*) –also the origin of the ideal coordinate system, is called focal length (*f*).

Let *M* be a point in space. The intersection of [*MC*] line and the image plane *I* is the projected point *m* on the image plane. Let M_C be the coordinates of the point *M* with respect to the camera coordinate system and let m_c be the coordinates of the point m with respect to the ideal image coordinate system as defined in (2.1) and (2.2).

$$M_C = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \tag{2.1}$$

$$m_c = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.2}$$

Since M and m are on the same line passing through the origin, we can set up the following relations:

$$\begin{aligned} x &= X'f / Z' \\ y &= Y'f / Z' \end{aligned} \tag{2.3}$$

Hence, a transformation can be done between M_C and $\tilde{m}_c$ that is augmented vector of m_c such that:

$$s.\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{P}\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}, \text{ where } s \text{ is a scalar and } \mathbf{P} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

P is called projection matrix. The purpose of the projection matrix is to convert the position of a 3D point in space to 2D point in the image. The projection matrix can be developed to make a transformation between the camera coordinate system to the real image coordinate system which is directly related with the pixels in an image. The point positions with respect to the real image coordinate system depend on intrinsic parameters of the camera such as principal point position, focal length and resolution of the camera and also the distortion caused by the lens of the camera and the skew angle.

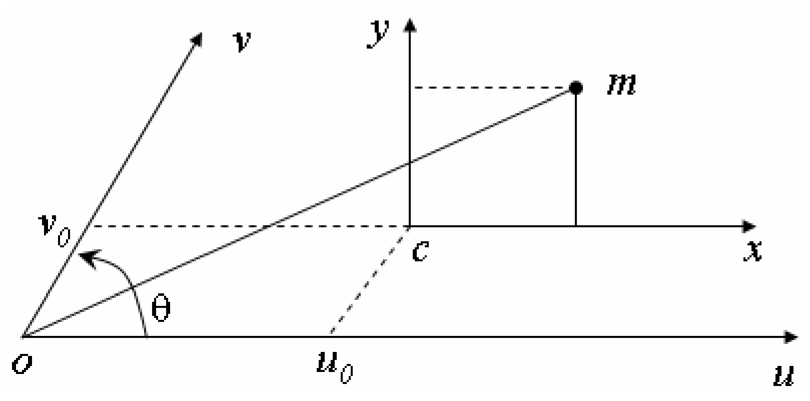


Figure 2.2 – Transformation between ideal and real image coordinate systems

Figure 2.2 shows ideal image coordinate system with the origin *c*, and real image coordinate system with the origin *o*. (u_0, v_0) is the coordinates of the principal point with respect to the real image coordinate system. The skew angle is denoted by *θ*. Let m_0 be the coordinates of the point *m* with respect to the real image coordinate system. Transformation between those coordinate systems can be written as $\tilde{m}_0 = \mathbf{K}\tilde{m}_c$ where $\tilde{m}_0 = [u \quad v \quad 1]^T$ is the augmented vector of m_0, and **K** is defined as:

$$\mathbf{K} = \begin{bmatrix} k_u & k_u \cot\theta & u_0 \\ 0 & k_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where k_u and k_v stand for the scaling along the x and y axes of the image plane respectively, and are related to the pixel cells of the camera. If each pixel in the real coordinate system is exact square, the ratio k_u / k_v is equal to 1. By using **K** with projection matrix, the transformation from camera coordinate system to real image coordinate system can be written as:

$$s.\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \qquad \text{where } \mathbf{A} = \mathbf{KP} \tag{2.6}$$

A is referred to as intrinsic matrix which is given by:

$$\mathbf{A} = \mathbf{KP} = \begin{bmatrix} fk_u & fk_u \cot\theta & u_0 \\ 0 & fk_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Real lens in cameras usually has some distortion, which major components are radial distortion and slight tangential distortion. Therefore, the above model may be extended as follows [3].

$$\begin{aligned} & x' = X'/Z' \\ & y' = Y'/Z' \\ & x'' = x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x'y' + p_2(r^2 + 2x'^2) \\ & y'' = y'(1 + k_1 r^2 + k_2 r^4) + 2p_2 x'y' + p_1(r^2 + 2y'^2) \\ & \text{where } r^2 = x'^2 + y'^2 \end{aligned} \tag{2.8}$$

In the equations (2.8), k_1 and k_2 are radial distortion coefficients, p_1 and p_2 are tangential distortion coefficients. Higher-order coefficients are not considered here. The distortion coefficients also do not depend on the scene viewed, thus they are also intrinsic camera parameters. Considering lens distortions, the transformation from the camera coordinate system to the real image coordinate system is given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \tag{2.9}$$

Let M_O be the coordinates of M with respect to the world coordinate system O. It is possible to calculate M_C using M_O or M_O using M_C if extrinsic parameters of the camera that are rotation and translation of the camera with respect to the world coordinate system, are known.

Rotation of a coordinate system can be represented as a 3x1 rotation vector with an angle or a 3x3 rotation matrix which directly transforms the system to a desired orientation. A rotation matrix is a real special orthogonal matrix whose transpose is its inverse. By using a rotation matrix, the coordinates of a point whose coordinates are known in the world coordinate system can easily be calculated with respect to the camera coordinate system if the rotation matrix of camera coordinate system with respect to the world camera system is known. Also, transpose of the matrix will rotate the system reverse. Translation of a coordinate system can be represented as a 3x1 translation vector. A complete transformation matrix including both rotation and translation is defined as $\begin{bmatrix} R & t \end{bmatrix}$ where R is the 3x3 rotation matrix and t is the 3x1 translation vector. M_C and M_O are related to each other by the equation $M_C = RM_O + t$. Let $\tilde{M}_O$ be the augmented vector of M_O. Then, the transformation between $\tilde{M}_O$ and M_C is given by:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} R_{3x3} & t_{3x1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.10}$$

$$M_C = \mathbf{D}\tilde{M}_O \tag{2.11}$$

D is called extrinsic matrix which is specified by the rotation matrix R and the translation vector t of the camera coordinate system with respect to the world coordinate system. Now, if we combine intrinsic and extrinsic parameters of camera, we can find the pixel coordinates of a point in 3D space using the equation defined by:

$$\tilde{m}_o = \mathbf{AD}\tilde{M}_O \tag{2.12}$$

$$s.\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & fk_u \cot\theta & u_0 \\ 0 & fk_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3x3} & t_{3x1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.13}$$

Purpose of the reconstruction process is to solve the equation above backwards. However, it is impossible to find M_O with the absence of Z value since there are infinitely many points on the line $|MC|$ whose projection is the point m. By using stereo camera, we will have two camera centers and two lines $|MC_1|$ and $|MC_2|$ intersecting in space on point M. Therefore, it is possible to find or estimate the position of M using pixel locations of m using more than one image.

2.2. Epipolar Geometry

The epipolar geometry defines a relationship between a point in space and its projections on two image planes. By using epipolar geometry, it is possible to find or estimate the position of a point in space using the positions of its projection on two image planes. It also helps us to find the location of projection of a point whose location is known on other image plane. The epipolar geometry is quite useful for 3D reconstruction and directly applicable for stereo cameras.

Let C and C' be two camera centers. The image planes of C and C' are **I** and **I'** respectively. The projections of a point M on **I** and **I'**, are m and m' respectively. Three points M, C and C' form an epipolar plane (**II**) together. The line $|CC'|$ is called baseline that intersects **I** and **I'** at the epipoles e and e' respectively.

Assume that R and R' are the rotation matrices of two camera coordinate systems with respect to the world coordinate system. Similarly, let t and t' be the translation vectors of cameras with respect to the world coordinate system. Then, projected points, m and m' are given by:

$$m = RM + t \tag{2.14}$$

$$m' = R'M + t' \tag{2.15}$$

By using the equation (2.14), M can be written as:

$$M = R^T (m - t) \tag{2.16}$$

Note that $R^T = R^{-1}$. Substituting (2.16) into (2.15) yields:

$$m' = R'R^T m + t' - R'R^T t \tag{2.17}$$

Equation (2.17) is the transformation from one camera coordinate system to the other camera coordinate system which can be simplified as:

$$m' = \mathbf{R}m + \mathbf{t} \quad \text{where } \mathbf{R} = R'R^T \text{ and } \mathbf{t} = t' - R'R^T t \tag{2.18}$$

R is rotation matrix and **t** is translation vector of second camera coordinate system with respect to first camera coordinate system.

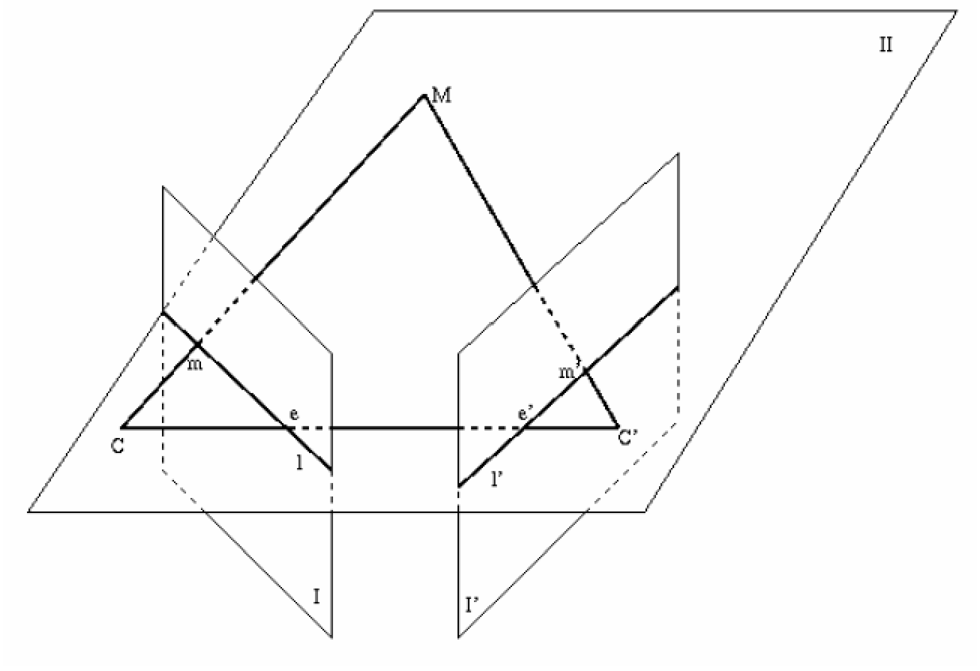


Figure 2.3 – Epipolar geometry

For all points on the line |*CM*| whose projections on **I** are the point *m*, the projected points on **I'** lie on the line |*e'm'*| which is called epipolar line *l'*. Similarly, for all points on the line |*C'M*| whose projections on **I'**, are the point *m'*, the projected point on **I** lies on the line |*em*| called epipolar line *l*. Therefore, if the projection of a point is known on an image plane, the corresponding projected point of that point on the other image plane must be available on a line where the epipolar plane **II** intersects the image plane. This is called epipolar constraint. The epipolar constraint can be formulated as:

$$\tilde{m}'^T F \tilde{m} = 0 \tag{2.19}$$

F is the fundamental matrix, $\tilde{m}$ and $\tilde{m}'$ are the augmented vectors of the points *m* and *m'* respectively. The fundamental matrix is a 3x3 determined by the intrinsic and extrinsic matrices of two cameras [2]. It is given by:

$$F = \mathbf{A}'^{-T}[\mathbf{t}]_x \mathbf{R}\mathbf{A}^{-1} \tag{2.20}$$

A, **A'** are intrinsic matrices of first and second camera respectively. $[\mathbf{t}]_x$ is the skew symmetric matrix formed by the translation **t** as follows:

$$[\mathbf{t}]_x = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \qquad \text{where } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2.21}$$

The fundamental matrix is used to obtain epipolar line parameters. $l' = F\tilde{m} = [a \quad b \quad c]^T$ is the epipolar line on the image plane of the second camera corresponding to the projected point $\tilde{m}$ on the image plane of the first camera. Then, the equation $ax + by + c = 0$ is the line equation of l'. Similarly, the line $l = F^T \tilde{m}' = [a' \quad b' \quad c']^T$ is the epipolar line on the image plane of the first camera corresponding to the projected point $\tilde{m}'$ on the image plane of the second camera. The epipolar line always goes through the projected point and the epipole which is also the projection of any point on the baseline. Therefore, if the projected point is the epipole itself, corresponding epipolar line is indefinite. Corresponding projected point of an epipole is the other epipole. Hence, $Fe = F^T e' = 0$ holds for epipoles. Matching operation of epipoles is also not possible with the aid of epipolar geometry using only two cameras, since all points on the baseline is projected on epipoles.

Fundamental matrix is quite useful for stereo matching and it can be easily computed for calibrated stereo cameras. However, calibration information may not be enough robust for stereo matching. But, it can be still successfully recovered by some known matched projected points. As being a 3x3 matrix, the fundamental matrix has 9 elements, but its scale is not significant. Determinant of the matrix is also zero. Therefore, it has 7 degrees of freedom and rank 2. It means that theoretically, at least 7 matched points are needed to recover the fundamental matrix.

Some of the fundamental matrix recovery algorithms are 7-point, 8-point and RANSAC (RANdom SAmple Consensus) algorithms [4]. 7-point algorithm can be used by using exactly 7 matched points. Consider a vector *f*, formed by the elements of the fundamental matrix $f = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})^T$ where F_{ij} is the element in i$_{\text{th}}$ row and j$_{\text{th}}$ column of fundamental matrix. Let $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{x'} = (x', y', 1)^T$ be the augmented vectors of matched points on image planes.

Since $\mathbf{x'}^T F\,\mathbf{x} = 0$, $(x'x, x'y, x', y'x, y'y, y', x, y, 1)\,f = 0$ holds. There is one equation for each of the matched points **x** and **x'**. For *n* known points, the equations can be stacked as follows:

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 \qquad (2.22)$$

For *n* = 7, the matrix *A* should have rank 7 and two dimensional null space. To solve the equation (2.22), singular value decomposition (SVD) [5] is applied to *A* such that $A = UDV^T$. The solution can be parameterized as $F = \alpha F_1 + (1-\alpha)F_2$ where F_1 and F_2 are the matrices built from the last two column of *V*. By using det(*F*) = det($\alpha F_1 + (1-\alpha)F_2$) = 0, we'll have a cubic polynomial equation in α that has three or one solution, therefore we'll have one or three solutions of *F* with rank 2 for each corresponding value of α .

In 8-point algorithm, at least 8 matched points are used (n ≥ 8). For ideal matched points, the matrix *A* shown in (2.22), will have rank 8. The solution is up to scale and *f* can be computed by using linear methods. Applying SVD to *A* and taking the last column of *V* to build *F* is a solution. In general, the recovered matrix *F* will not have rank 2, so it must be guaranteed to have rank 2. Again, we apply SVD to *F* such that $F = UDV^T$ where $D = \text{diag}(r, s, t)$ is a diagonal matrix satisfying $r \geq s \geq t$. We should now replace the last singular value *t* with zero and compute the fundamental matrix $F' = U.\ \text{diag}(r, s, 0).V^T$. This process will make *F'* to have rank 2. Ideally, there should be only one zero singular value in *D*, however since the corresponding matched points are not ideal, one singular value may be relatively small than the other such that smaller singular value is approximately zero with respect to the other singular value. Therefore, the computed fundamental matrix will be inaccurate. Used matched points should be well-distributed on the image planes in order to have better results. A solution proposed by Hartley [6], is normalized 8-point algorithm such that a transformation matrix is used for scaling and translating matched points in order to make them well-distributed. We define a transformation such that $\mathbf{y} = T\mathbf{x}$ and $\mathbf{y'} = T'\mathbf{x'}$ where transformation matrices *T* and *T'* are given by:

$$T = \begin{bmatrix} s^{-1} & 0 & -s^{-1}c_x \\ 0 & s^{-1} & -s^{-1}c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad T' = \begin{bmatrix} s'^{-1} & 0 & -s'^{-1}c'_x \\ 0 & s'^{-1} & -s'^{-1}c'_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

The scale factors *s* and *s'* are used for scaling the points on the first and second image plane respectively so that they have a RMS distance, $\sqrt{2}$ from the origin. Translations *c* and *c'* are used for shifting the points on the first and second image plane respectively to make them be distributed around the origin that is the mass center of them. c_x denotes x-coordinate and c_y denotes y-coordinate of the center point. *s*, *s'*, *c* and *c'* can be formulated as:

$$c = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad c' = \frac{1}{n}\sum_{i=1}^{n} x'_i \tag{2.24}$$

$$s = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}\left|x_i - c\right|^2} \qquad s' = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}\left|x'_i - c'\right|^2} \tag{2.25}$$

Hartley suggests using **y** and **y'** instead of **x** and **x'** to compute the fundamental matrix with 8-point algorithm and reverse the transformations to obtain the real fundamental matrix using the equation:

$$F'' = T'^{-T} F' T^{-1} \tag{2.26}$$

If more than 8 points are used (n > 8) for 8-point algorithm, the fundamental matrix may be inaccurate since the coordinates of matched points are approximately correct. Normalized 8-point algorithm [7] discussed above, will reduce the effect of error in matched points, but a robust computation of the fundamental matrix is to select proper points from the given matched points. RANSAC algorithm is a robust way to compute fundamental matrix when lots of points are used and even though some of them are outliers (false matches).

RANSAC algorithm is an iterative method which is able to compute the best possible fundamental matrix by choosing the most proper matched points. In RANSAC algorithm, randomly selected points are used to compute the fundamental matrix using 7-point or 8-point algorithm. If the computed fundamental matrix satisfies a sufficient number of all given matched points (distance to the epiline is below some threshold), a consensus set is formed by those satisfying points together with the points that have been used for computing the fundamental matrix. If 8-point algorithm is used, the fundamental matrix may be recomputed using the consensus set. Then, if it is the best consensus set, for instance, the number of elements in the set is greater than the previous ones, the fundamental matrix and the consensus set are selected as candidate and the whole process repeats until the iteration number reaches the maximum number of iterations allowed. Optionally, the elements in the consensus set may be used to measure error. If the error is unacceptable, the set can be ignored. The average distance from the points to the corresponding epilines can be used to calculate the error.

Since the false matches are not able to construct an accurate fundamental matrix and not probably closer to the epilines, they are excluded from the consensus set. Therefore, RANSAC algorithm is quite useful for robust fundamental matrix recovery using matched points such that some of them are approximately correct and some of them are completely incorrect. Major disadvantage of the algorithm is that there is no upper bound on time that is necessary for the algorithm to compute the best result. If a time limitation is applied, the algorithm may not give the best result. If the number of matched points is not a few, there will be a lot of combination to choose points so that iteration number should not be a small number. One approach to estimate the maximum number of iterations needed is use the number of inliers (number of elements in the consensus set). If the fundamental matrix is computed by using an outlier, it will be a bad result. Let k be the expected number of iterations needed to have a good result. Let p be the probability that RANSAC algorithm randomly selects only inliers in some iteration to compute the fundamental matrix. When this happens, the result will be probably useful. Let w be the probability of choosing an inlier each time a single point is randomly selected. It can be estimated after the each iteration using the formula:

$$w = \text{number of inliers} / \text{number of all points} \tag{2.27}$$

If n points are randomly chosen to compute the fundamental matrix, w^n is the probability that all n points are inliers and $1 - w^n$ is the probability that at least one of the points is an outlier. Then $(1 - w^n)^k$ is the probability of choosing at least one outlier in every k iterations. In other words, it is the probability that the algorithm never selects a set of n points which all are inliers and it is equal to $1 - p$.

$$1 - p = (1 - w^n)^k \tag{2.28}$$

Hence, number of iterations needed, k is:

$$k = \frac{\log(1 - p)}{\log(1 - w^n)} \tag{2.29}$$

We expect that in *k* iterations, the algorithm selects all inliers to compute a reliable fundamental matrix and we adjust *p* as the probability we want the result to be reliable, for instance, *p* = 99%. High probabilities will result in high number of iterations so that algorithm will take more time to produce a reliable result. Low probabilities will make the algorithm to complete its job faster but the probability of producing a bad result will increase in this case.

2.3. Stereo Triangulation

Stereo triangulation is a process of finding the 3D position of a point whose projections are known on two images. Since the known projected points are not ideal points, they do not meet in space exactly. Stereo triangulation methods intend to find the best fitted point along the projections. A basic approach is to find the mid-point of closest points of the rays passing through the camera centers and matched points on the image plane as shown in Figure 2.4. Let [R | t] be a transformation from the first camera coordinate system to second camera coordinate system where R is the rotation matrix and t is the translation vector. Assuming that we are working on the first camera coordinate system, the first camera center C_1 is the origin. The second camera center C_2 is also an origin for the second camera coordinate system. Its coordinates with respect to the first camera is given by the inverse transformation between camera systems:

$$C_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{2.30}$$

$$C_2 = -R^T t \tag{2.31}$$

Let A_1 and A_2 be the intrinsic matrices of the first and second camera respectively. Considering we have a matched pair (x_1, x_2) whose 2D coordinates are known in the real image coordinate system, we have to find some points on the rays with respect to the camera coordinate systems. Let X_1 and X_2 be the normalized pixel coordinates of the matched points x_1 and x_2 respectively on the rays which are given by (2.32), (2.33) and (2.34).

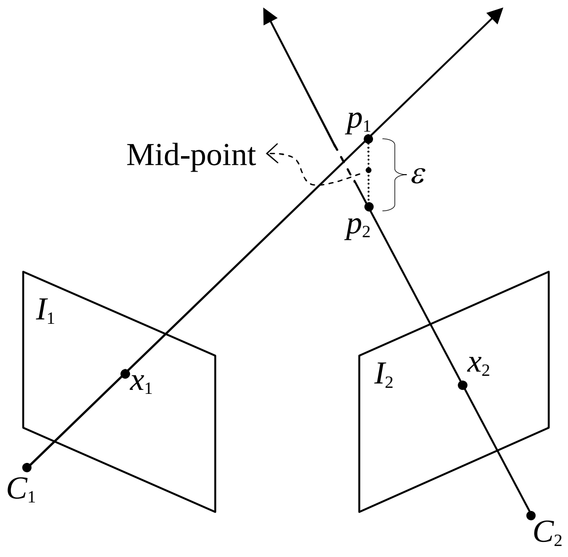


Figure 2.4 – The mid-point method for stereo triangulation

$$x_1 = [u_1 \quad v_1]^T,\ x_2 = [u_2 \quad v_2]^T \tag{2.32}$$

$$X_1 = [x_1' \quad y_1' \quad z_1']^T = A_1^{-1}[x_1 \quad 1]^T \tag{2.33}$$

$$X_2 = [x_2' \quad y_2' \quad z_2']^T = R^T\left(A_2^{-1}[x_2 \quad 1]^T - t\right) \tag{2.34}$$

The normalized pixel coordinates X_1 and X_2 indicate two points on the rays with respect to the first camera coordinate system. Let d_1 and d_2 be the vectors from C_1 to X_1 and from C_2 to X_2 respectively that are given by:

$$d_1 = X_1 - C_1,\ d_2 = X_2 - C_2 \tag{2.35}$$

The line equations corresponding to the rays can be written in as:

$$p_1 = C_1 + \lambda_1 d_1,\ p_2 = C_2 + \lambda_2 d_2 \tag{2.36}$$

Now, we minimize the squared distance between the lines p_1 and p_2:

$$\varepsilon^2 = \|p_1 - p_2\|^2 \tag{2.37}$$

A least squares solution for (2.37) is as follows:

$$p_1 - p_2 = 0 \tag{2.38}$$

$$\lambda_1 d_1 - \lambda_2 d_2 = C_2 - C_1 \tag{2.39}$$

$$\begin{bmatrix} d_1 & -d_2 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = C_2 - C_1 \tag{2.40}$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \left(M^T M\right)^{-1} M^T \left(C_2 - C_1\right) \tag{2.41}$$

where the matrix $M = \begin{bmatrix} d_1 & -d_2 \end{bmatrix}$ is 3x2 matrix. The points on the line p_1 and p_2 are the closest line points along the rays that can be calculated using (2.36). Then, the mid-point of p_1 and p_2 gives the 3D position of the point whose projection on the image planes are x_1 and x_2 with respect to the first camera coordinate system:

$$P_1 = \frac{1}{2}(p_1 + p_2) \tag{2.42}$$

Position of the point with respect to the second camera coordinate system may be found using the transformation between cameras as follows:

$$P_2 = RP_1 + t \tag{2.43}$$

The error in the estimation of the position is given by the equation (2.37) and can be used for determining how close the rays meet in space based on the closest distance between two lines. In another approach for stereo triangulation, the relation given in (2.43) is used. Let $P_1 = \begin{bmatrix} X_1' & Y_1' & Z_1' \end{bmatrix}$ and $P_2 = \begin{bmatrix} X_2' & Y_2' & Z_2' \end{bmatrix}$ denote the coordinates of the point with respect to the first and second camera coordinate system respectively. Normalized coordinates of P_1 and P_2 are given by:

$$P_1 / Z_1' = P_1' = A_1^{-1} \begin{bmatrix} x_1 & 1 \end{bmatrix}^T, \; P_2 / Z_2' = P_2' = A_2^{-1} \begin{bmatrix} x_2 & 1 \end{bmatrix}^T \tag{2.44}$$

where A_1 and A_2 are the intrinsic matrices of the first and second camera respectively. (x_1, x_2) is the matched pair in real image coordinate system. Combining two relations given in (2.43) and (2.44) yields following relation.

$$Z_2' P_2' = Z_1' R P_1' + t \tag{2.45}$$

$$\begin{bmatrix} -RP_1' & P_2' \end{bmatrix} \begin{bmatrix} Z_1' \\ Z_2' \end{bmatrix} = t \tag{2.46}$$

Let $M = \begin{bmatrix} -RP_1' & P_2' \end{bmatrix}$ be a 3x2 matrix. The least squares solution for (2.46) is:

$$\begin{bmatrix} Z_1' \\ Z_2' \end{bmatrix} = \left(M^T M\right)^{-1} M^T t \tag{2.47}$$

Then, 3D position of the points P_1 and P_2 can be found by substituting Z_1' and Z_2' into the equation (2.44). The error is given by:

$$\varepsilon = \left\| M \begin{bmatrix} Z_1' \\ Z_2' \end{bmatrix} - t \right\| \tag{2.48}$$

If the rotation matrix and translation vectors for the transformation between the world coordinate system and camera coordinate systems are available, we can easily make a transformation to find the coordinates of P_1 and P_2 with respect to the world coordinate system using one of following relations:

$$P_W = R_1^T (P_1 - t_1) \tag{2.49}$$

$$P_W = R_2^T (P_2 - t_2) \tag{2.50}$$

where R_1, t_1 and R_2, t_2 are the rotation matrices and translation vectors from the world to camera the coordinate system of the first and camera respectively. Another popular method for stereo triangulation is based on finding a common solution for the equations (2.13) of stereo cameras. Similarly, the purpose is to form a matrix as presented in previous methods and find a solution for it. Assume that we have full knowledge of intrinsic and extrinsic parameters for both cameras as well as we have in previous methods. Therefore, we have following projection equations.

$$s[u_1 \quad v_1 \quad 1]^T = A_1[R_1 \mid t_1] \; [X \quad Y \quad Z \quad 1]^T \tag{2.51}$$

$$s'[u_2 \quad v_2 \quad 1]^T = A_2[R_2 \mid t_2] \; [X \quad Y \quad Z \quad 1]^T \tag{2.52}$$

A common solution of (2.51) and (2.52) gives the point location with respect to the world coordinate system. If we want to work in the first camera coordinate system or both of the transformations [R_1 | t_1] and [R_2 | t_2] are not known, we can modify the relations by choosing [R_1 | t_1] = [**I** | **O**] and [R_2 | t_2] = [R | t] where **I** is a 3x3 identity matrix, **O** is a 3x1 zero vector and [R | t] is the transformation from the first camera to second camera coordinate system. In this case, the solution will give the location in first camera coordinate system.

Let $Q = A_1[R_1 \mid t_1]$ and $Q' = A_2[R_2 \mid t_2]$ be 3x4 matrices that can be denoted as $Q = [q_1 \quad q_2 \quad q_3]^T$ and $Q' = [q'_1 \quad q'_2 \quad q'_3]^T$ respectively where q_i and q'_i are the rows of the mentioned matrices. The equations (2.51) and (2.52) can be used to build the following 4x4 matrix with a relation it satisfies:

$$M = \begin{bmatrix} q_1 - q_3 u_1 \\ q_2 - q_3 v_1 \\ q'_1 - q'_3 u_2 \\ q'_2 - q'_3 v_2 \end{bmatrix} \tag{2.53}$$

$$M[X \quad Y \quad Z \quad 1]^T = 0 \tag{2.54}$$

The equation (2.54) indicates that the coordinate vector should be in the null space of M or the symmetric matrix $M^T M$. Applying SVD to $M^T M$, we have $M^T M = USV^T$. The solution is given by the last column of V:

$$V = [v_1 \quad v_2 \quad v_3 \quad v_4] \tag{2.55}$$

In the equation (2.55), v_i are the column vectors of V and the coordinate vector can be computed as:

$$\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T = v_4 / V_{4,4} \tag{2.56}$$

$V_{4,4}$ denotes the element in vector v_4. The error in estimation of the coordinate point is then the magnitude of the vector given by the substitution of the coordinate values into the equation (2.54):

$$\varepsilon = \left\| M \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \right\| \tag{2.57}$$

CHAPTER 3

FEATURE EXTRACTION

Detecting locations of the feature points in an image is the feature extraction process. Commonly used feature points are corners and edges points. Feature-based matching techniques use the feature points to find correspondence between images. Therefore, it is fairly important to detect features correctly and reliably as far as possible. Well known corner detectors are Moravec operator and Harris corner detector. Edge points are extracted by differential operators such as Prewitt, Sobel or Laplace. The Canny algorithm is also an advanced and popular algorithm to detect edge points.

3.1. Corner Detectors

In an image, corners exist at the points where the intensity transitions from low to high are occurred in both two dimensions. Corner detectors are based on detecting intensity transitions where corners tend to be available.

3.1.1. Moravec Operator

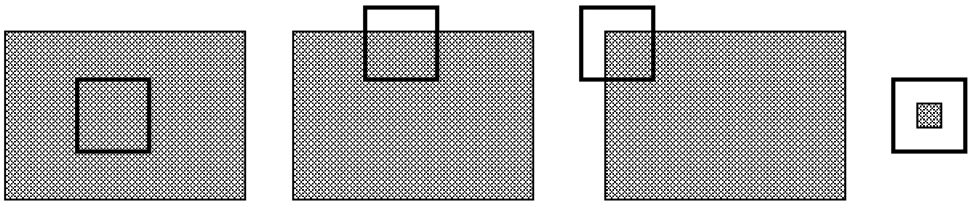
Moravec's corner detector functions compute intensity changes in the image to extract corner points. A window is placed upon a candidate point, containing the points which will be compared with the candidate point. The size of the window is typically chosen as 3x3, 5x5 or 7x7 [8]. There are three cases to be considered:

A. The points inside the window have small amount of variation in any direction, indicating that the image patch is flat. (Figure 3.1.a)

B. The points inside the window have small amount of variation in one direction but large amount of variation in other direction, indicating that the image patch contains edge points. (Figure 3.1.b)

C. The points inside the window have large amount of variation in any direction. In this case, a corner or an isolated point is inside the window. (Figure 3.1.c)

The figure below shows possible window positions for each case:



(a) Flat region (b) Edge region (c) Corner or isolated point region

Figure 3.1 – Possible window positions placed on a region

Moravec found the variance of points inside the window by shifting it over the image and computing the variances between the corresponding points inside the original window and the shifted window. Corner points are decided according to average changes of image intensities [9]. The variance of image points is given by:

$$E(x,y)=\sum_{u,v} w_{u,v}\left[(I(x+u,y+v)-I(u,v)\right]^2 \qquad (3.1)$$

where *w* is the window which is unity within a specified region, and zero elsewhere; *I* is the image intensities. The window shifts, (*x*, *y*) are considered comprise [(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)]. (*u*, *v*) denotes image coordinates.

Figure 3.2 shows the situation for the shift (1, -1) for an isolated point and a corner point. Red window is the original window and blue window is the shifted window. For this example, the variance is given in equation (3.2).

$$E(1,-1) = \sum_{i=1}^{9} (A_i - B_i)^2 \tag{3.2}$$

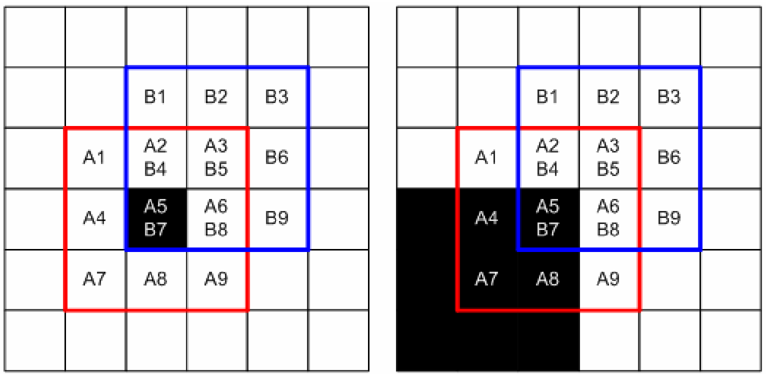


Figure 3.2 – Window positions for a given shift

The point in the center of the window is accepted as a corner according the following formula:

$$C = \begin{cases} 1 & \min(E) > T \\ 0 & otherwise \end{cases} \tag{3.3}$$

where T is a threshold allowing strong responses to be accepted as a corner. All corners in the image can be found by checking all points in the image by placing the window all possible position on the image.

The main drawback of this method is that it is not rotationally invariant. The response is noisy because of the rectangular window and effected by the edge pixels because only the minimum of E is taken into account [9]. Isolated points are also accepted as a corner; therefore the method is not noise tolerant [8].

Figure 3.3 shows the Moravec operator applied to the blocks test image [8] with a 3x3 window and the threshold was chosen in order to detect most of the corners while trying to minimize the number of false corners detected. The Moravec operator managed to do a reasonable job of finding the majority of true corners, but it can be seen that diagonal edges are also detected as a corner [8].

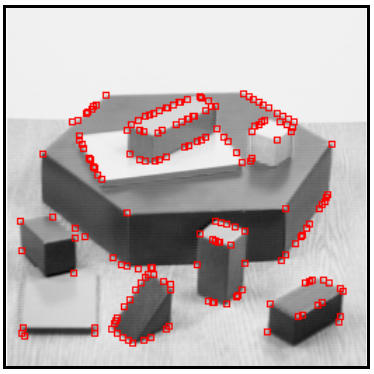


Figure 3.3 – Result of Moravec operator applied to a test image

3.1.2. Harris Corner Detector

C. Harris and M. Stephens improved Moravec's corner detection method, dissipating its problems. They used the auto-correlation function measuring the local changes in intensity inside a window shifted by a small amount in different directions [10].

Given a shift (x, y), the auto-correlation function is defined as:

$$E(x,y)=\sum_{u,v} w_{u,v}\left[I(x+u,y+v)-I(u,v)\right]^2 \tag{3.4}$$

The image points inside the shifted window is approximated by a Taylor expansion truncated to the first order terms [10],

$$I(x+u,y+v)\approx I(u,v)+\left[I_u(u,v) \quad I_v(u,v)\right]\begin{bmatrix} x \\ y \end{bmatrix} \tag{3.5}$$

Substituting approximation Eq. (3.5) into Eq. (3.4) yields,

$$E(x,y)=\sum_{u,v} w_{u,v}\left(\left[I_x(u,v) \quad I_y(u,v)\right]\begin{bmatrix} x \\ y \end{bmatrix}\right)^2 \tag{3.6}$$

$$=\left[x \quad y\right]\begin{bmatrix} \sum_{u,v} w_{u,v}\left(I_x(u,v)\right)^2 & \sum_{u,v} w_{u,v} I_x(u,v) I_y(u,v) \\ \sum_{u,v} w_{u,v} I_x(u,v) I_y(u,v) & \sum_{u,v} w_{u,v}\left(I_y(u,v)\right)^2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{3.7}$$

The first gradients in the equations above can be approximated by:

$$I_x = \partial I / \partial x \approx I \otimes (-1,0,1)$$
$$I_y = \partial I / \partial y \approx I \otimes (-1,0,1)^T \tag{3.8}$$

Here, $\otimes$ is the convolution operator. Hence, for small shifts, *E* can be written as:

$$E(x,y) = Ax^2 + 2Cxy + By^2 \tag{3.9}$$
$$A = \sum_{u,v} w_{u,v} \left(I_x(u,v)\right)^2$$
$$B = \sum_{u,v} w_{u,v} \left(I_y(u,v)\right)^2$$
$$C = \sum_{u,v} w_{u,v} I_x(u,v) I_y(u,v)$$

For the window *w* (at origin), a smooth circular window such as Gaussian function may be used:

$$w_{u,v} = e^{\frac{-(u^2+v^2)}{2\sigma^2}} \tag{3.10}$$

The change, *E*, for a small shift can be concisely written as:

$$E(x,y) = (x,y)M(x,y)^T \tag{3.11}$$

where 2x2 symmetric matrix *M* is:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{3.12}$$

Let $\lambda_{1,}$ λ_2 be the eigenvalues of matrix *M*. The eigenvalues form a rotationally invariant description. Three cases should be considered:

1. Both eigenvalues $\lambda_{1,}$ λ_2 are small, so that the local auto-correlation function is flat indicating that the intensity values in the window are close to each other.
2. If one eigenvalue is high and the other is low, then we have a ridge shaped auto-correlation function and local shifts in only one direction cause a little change in *E,* indicating that candidate point is an edge point.
3. Both eigenvalues $\lambda_{1,}$ λ_2 are high, so we have a sharply peaked auto-correlation function. Shifts in any direction will result in a significant increase in *E*, indicating a corner.

For Corner/Edge classification, C. Harris and M. J. Stephens suggested to use a corner/edge response function:

$$R = Det(M) - kTr^2(M) \tag{3.13}$$

where the determinant of *M* is,

$$Det(M) = \lambda_1\lambda_2 = AB - C^2, \tag{3.14}$$

and the trace of *M* is,

$$Tr(M) = \lambda_1 + \lambda_2 = A + B \tag{3.15}$$

Harris response *R* is positive for the corners, negative for edges and small for flat points and can be used for distinguishing corners and edges. In the formula, *k* is a free parameter, the value of it has to be determined empirically, and in the literature values in the range 0.04 - 0.15 have been reported as feasible [1].

Using Harris response function is better way to detect corners because it reduces computation time since it can be computed directly using *A*, *B* and *C* without computing eigenvalues.

All corners in the image can be extracted by repeating the process for placing the window onto all suitable positions. Corners can be found as local maxima points above some threshold in corresponding Harris response values.

Figure 3.4 shows a test image with a size of 640x360. We applied Harris response function to the image using a 3x3 binary window with a free parameter k = 0.04. Gradients were found with a noise tolerant 7x7 extended Sobel kernel given in Figure 3.5. Figure 3.6 shows normalized Harris response of the test image shown at Figure 3.4. Bright points show corners where R is positive, darker contours show edges where R is negative. Gray regions are where the image patches are flat and R is a small number. Local maxima points of Harris response are shown in Figure 3.7 and marked with a red "+" sign. Harris corner detector is very good at distinguishing corner and edge points on the contrary of Moravec operator.

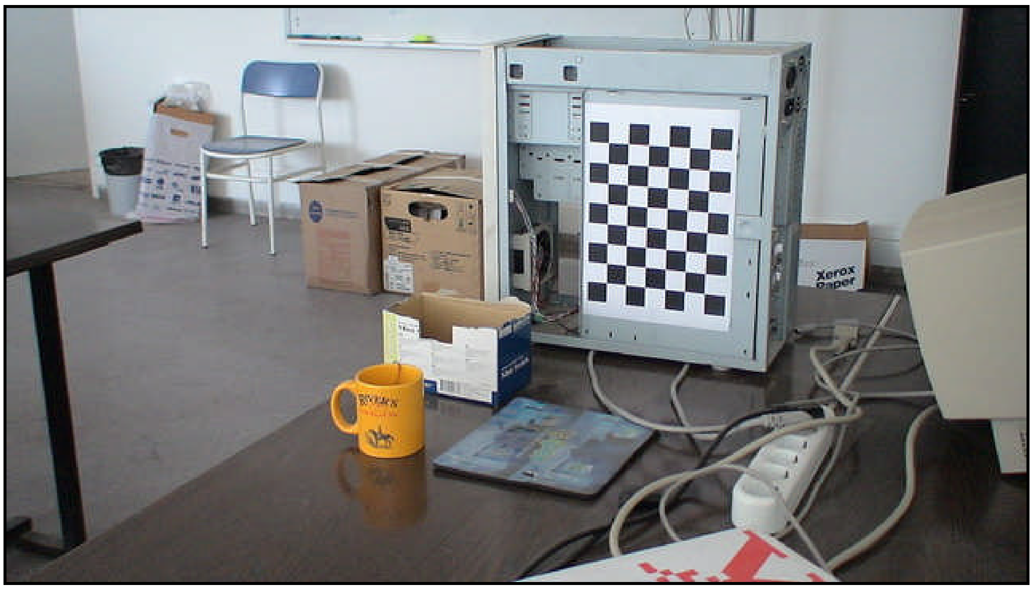


Figure 3.4 – Test image for corner detection

1	4	5	0	-5	-4	-1
6	24	30	0	-30	-24	-6
15	60	75	0	-75	-60	-15
20	80	100	0	-100	-80	-20
15	60	75	0	-75	-60	-15
6	24	30	0	-30	-24	-6
1	4	5	0	-5	-4	-1

S_x

1	6	15	20	15	6	1
4	24	60	80	60	24	4
5	30	75	100	75	30	5
0	0	0	0	0	0	0
-5	-30	-75	-100	-75	-30	-5
-4	-24	-60	-80	-60	-24	-4
-1	-6	-15	-20	-15	-6	-1

S_y

$$I_x = \partial I / \partial x \approx I \otimes S_x$$

$$I_y = \partial I / \partial y \approx I \otimes S_y$$

Figure 3.5 – 7x7 extended Sobel kernels

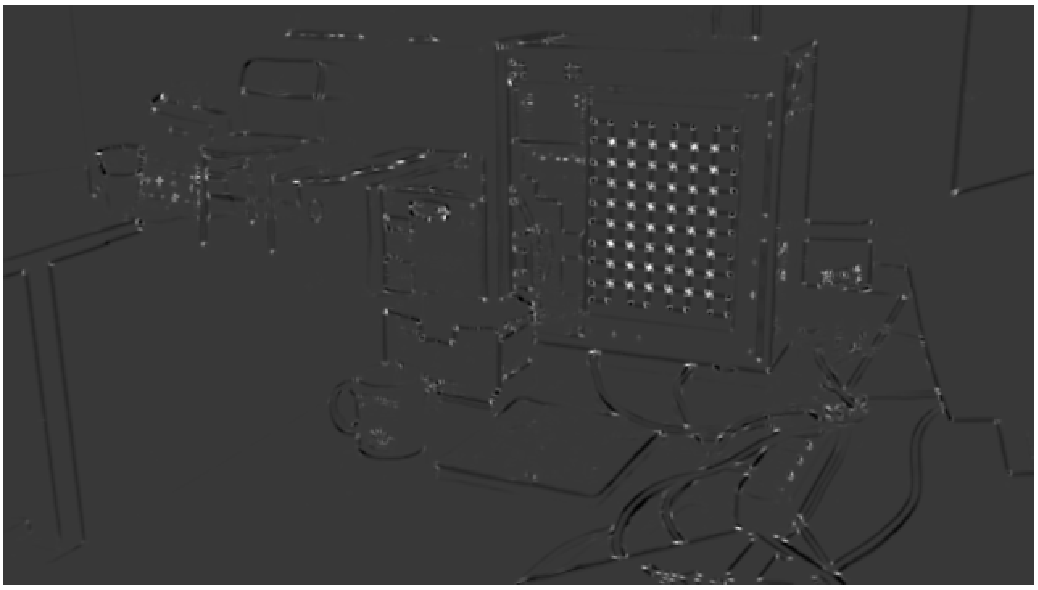


Figure 3.6 – Harris response for the test image

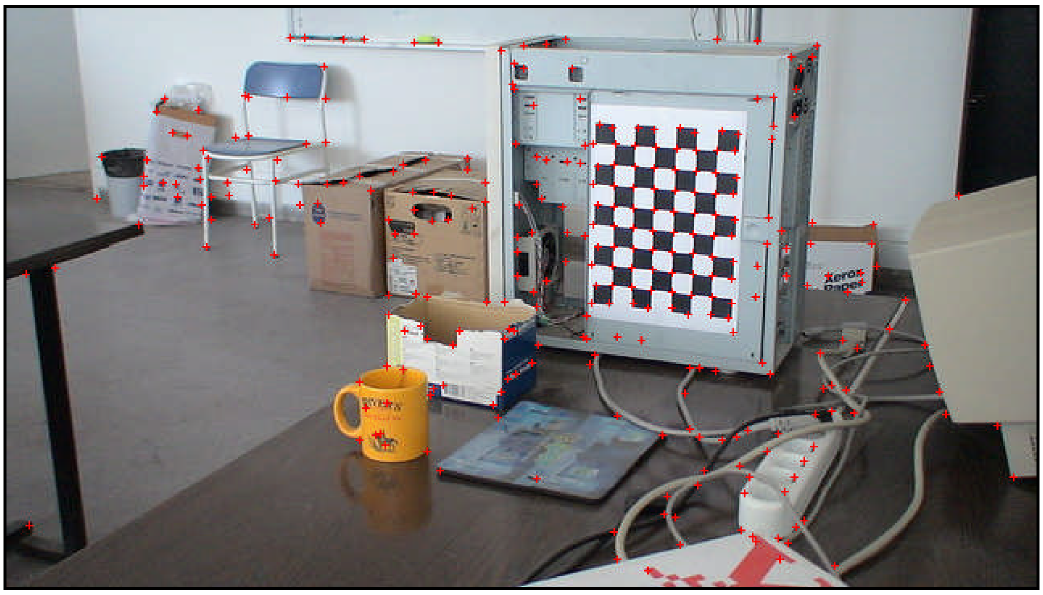


Figure 3.7 – Local maxima points corresponding to the corners

3.2. Edge Detection Methods

Edge detection has a great importance in image segmentation to extract object boundaries in an image. It is very common approach for finding meaningful discontinuities in intensity values. Edge detection methods can be classified in two categories:

a. First order derivatives (Gradient based),
b. Second order derivatives (Laplacian based).

In an ideal edge, there is a sudden change in intensity and ideal edges can be easily detected by computing intensity differences in consecutive image points, but in photographs we see a smooth transaction in intensity known as ramp edge.

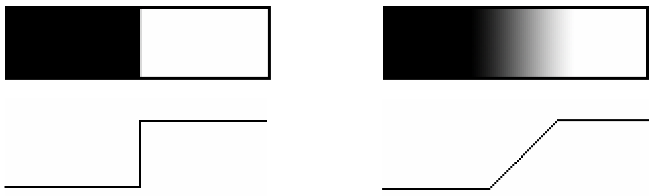


Figure 3.8 - Ideal and ramp edge models and gray-level profiles

According to the ramp edge model shown in Figure 3.8, the edge ramp can be detected using derivatives of the image. The first derivative of the gray-level profile is positive and constant at the points on the ramp and zero at out of the ramp. Since there is more than one point on the ramp, the transition has a thickness. This is known as thick edge problem. Thick edges are useless to detect an object's contour in the image. To cope with the problem, we can use second derivative of the gray-level profile. In second order derivative, we see a positive impulse at beginning of the ramp and a negative impulse at the end of the ramp. If we assume that high level intensity values belong to an object, when we move into the background from the object, we will have again a negative impulse and then a positive impulse. Since we have two impulses on a transition, we will detect two edges on the contour of the objects. This is also known as double edge problem.

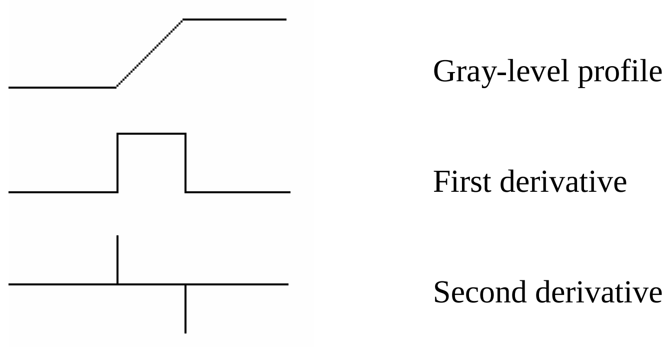


Figure 3.9 – Gray level profile, first and second derivatives

As the order of derivatives increases, the sensitivity to noise will increase. First order derivative is sensitive to noise and second order derivative is even more sensitive to noise. Therefore, edge detector operators which use derivatives are generally used together with smoothing filters to reduce the noise effect.

3.2.1. Gradient Operators

First order derivatives of a digital image are based on approximations of the 2D gradient. The gradient vector of the image *f* at the location (*x*, *y*) is defined as the following equation in (3.16).

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.16}$$

Magnitude of the gradient vector is very important to find whether the edge is strong or not. It is defined as:

$$\left|\nabla f\right| = \sqrt{G_x^2 + G_y^2} \tag{3.17}$$

This quantity gives the maximum rate of increase of f per unit distance in the direction of ∇f . Since the magnitude of gradient involves squares and square roots, it requires more computations than the approximation of the vector magnitude given in equation (3.18).

$$\left|\nabla f\right| \approx \left|G_x\right| + \left|G_y\right| \tag{3.18}$$

The approximation above is more attractive computationally but not rotationally invariant in general, however gradient operators such as Sobel and Prewitt which give isotropic results for only horizontal and vertical edges, are quite suitable to be used with this approximation [11].

Direction of the gradient vector is important for linking the neighbor edges. Direction of the edge is perpendicular to the direction of the gradient vector. The angle α between the gradient vector located at (x, y) and the x-axis is given by:

$$\alpha(x, y) = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.19}$$

Gradient image is formed by gradient vectors obtaining G_x and G_y values at each pixel. These values are approximated by computing the convolution of the image with an operator such as Prewitt and Sobel operators.

$$\nabla f \approx \begin{bmatrix} f \otimes K_x \\ f \otimes K_y \end{bmatrix} \tag{3.20}$$

where K_x and K_y are the horizontal and vertical kernels respectively.

3.2.1.a. 3x3 Prewitt Operator

Prewitt operators are computationally fast but do not have good noise suppression characteristics since they have same coefficient values in both diagonal and middle points.

1	0	-1
1	0	-1
1	0	-1

P_x

1	1	1
0	0	0
-1	-1	-1

P_y

Figure 3.10 – Horizontal (P_x) and vertical (P_y) 3x3 Prewitt kernels

3.2.1.b. 3x3 Sobel Operator

In Sobel operators, non-diagonal neighbors of the center have twice importance with respect to the diagonal neighbors causing a smoothing effect which reduces noise effect. Sobel operators are used as a gradient operator for many purposes.

1	0	-1
2	0	-2
1	0	-1

S_x

1	2	1
0	0	0
-1	-2	-1

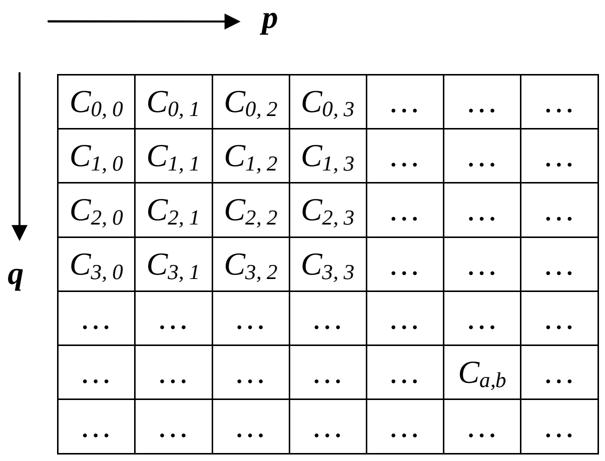
S_y

Figure 3.11 – Horizontal (S_x) and vertical (S_y) 3x3 Sobel kernels

3.2.1.c. Extended Sobel Operators

A Sobel kernel can be expressed as a polynomial expansion and each element in the kernel can be expressed as a term in the expansion [3]. Let p and q be the column and row numbers of the kernel respectively and kernel elements, *C* be the coefficient of a polynomial term defined as:

$$T_{a,b} = C_{a,b}\, p^{a} q^{b}$$



$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{0,3}$	…	…	…
$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	…	…	…
$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	…	…	…
$C_{3,0}$	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	$C_{a,b}$	…
…	…	…	…	…	…	…

Figure 3.12 – Coefficients in an extended Sobel kernel

Coefficients of each term in the polynomial expansion denote kernel elements. An *N*x*N* Sobel kernel can be written as:

$$\begin{aligned} S_x &= (1+p)^{N-2}(1+q)^{N-1}(1-p) \\ S_y &= (1+p)^{N-1}(1+q)^{N-2}(1-q) \end{aligned} \qquad \text{where } N > 0 \text{ and } N \text{ is odd.}$$

Primitive kernels $p = 1.p^1q^0 = [0 \quad 1]$, $q = 1.p^0q^1 = [0 \quad 1]^T$ can be used to construct the kernel. Using *p* and *q*, factors of S_x and S_y can be written as:

$$(1+p) = 1.p^0q^0 + 1.p^1q^0 = [1 \quad 1]$$

$$(1+q) = 1.p^0q^0 + 1.p^0q^1 = [1 \quad 1]^T$$

$$(1-p) = 1.p^0q^0 - 1.p^1q^0 = [1 \quad -1]$$

$$(1-q) = 1.p^0q^0 - 1.p^0q^1 = [1 \quad -1]^T$$

Hence, Sobel kernels can be constructed by convolving factors. For *N*=5, 5x5 horizontal and vertical Sobel kernels are given by:

$$S_x = (1+p)^3(1+q)^4(1-p) = \begin{bmatrix}1 & -1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix}$$

$$= \begin{bmatrix}1 & 2 & 0 & -2 & -1\\ 4 & 8 & 0 & -8 & -4\\ 6 & 12 & 0 & -12 & -6\\ 4 & 8 & 0 & -8 & -4\\ 1 & 2 & 0 & -2 & -1\end{bmatrix}$$

$$S_y = (1+p)^4(1+q)^3(1-q) = \begin{bmatrix}1\\-1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 1\end{bmatrix}$$

$$= \begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 2 & 8 & 12 & 8 & 2\\ 0 & 0 & 0 & 0 & 0\\ -2 & -8 & -12 & -8 & -2\\ -1 & -4 & -6 & -2 & -1\end{bmatrix}$$

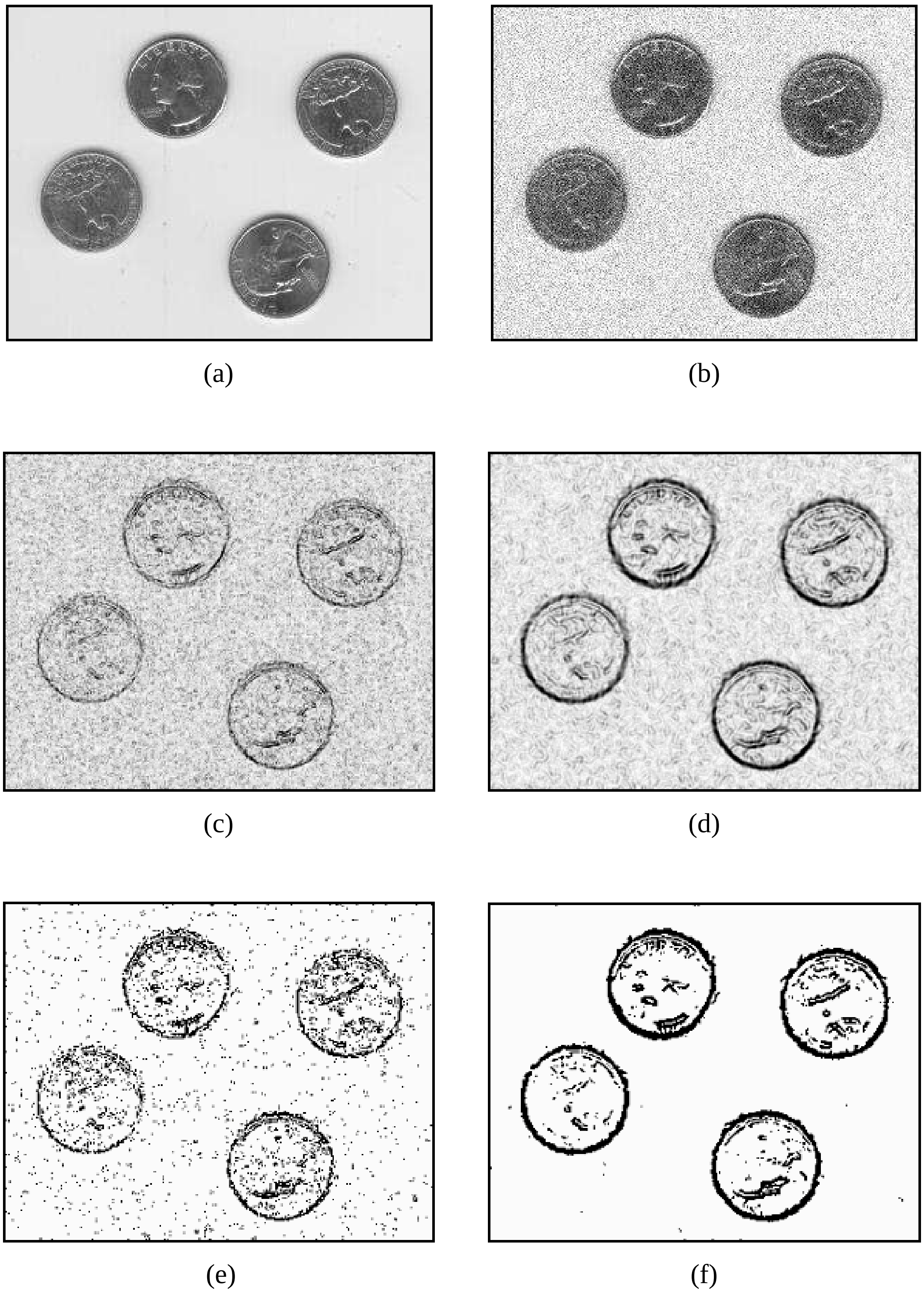


Figure 3.13 – Test image, Gaussian noise added test image, Sobel operator results and thresholded Sobel results

Figure 3.13(a) shows a 308x242 test image with a white plain background which has 256 gray levels from 0 to 255. Figure 3.13(b) is the result image after white Gaussian noise [1] is added to the image in Figure 3.13(a) with a variance $\sigma^2 = 0.01$ and a mean value $\mu = 0$. Then, 3x3 Sobel operator was applied to obtain the image in Figure 3.13(c) and 7x7 extended Sobel operator is applied to obtain the image in Figure 3.13(d). Intensity values of the edge images shown in Figure 3.13(c) and 3.13(d) were obtained by scaling the absolute values of the results of Sobel operators to the maximum gray value 255. The edge images in Figure 3.13(c) and 3.13(d) were used to obtain the images shown in Figure 3.13(e) and 3.13(f) respectively by using the following formula:

$$T(x,y) = \begin{cases} 255 & I(x,y) > 100 \\ 0 & otherwise \end{cases}$$

where *I*(*x*, *y*) is the edge image and *T*(*x, y*) is the thresholded edge image.

Clearly it is seen that the larger sized Sobel operator is more noise tolerant. In Figure 3.13(e), more false edge pixels were detected than the image in Figure 3.13(f) in the background region where we do not expect an edge point. However, thicker edges were obtained due to the size of Sobel operator. Thick edges are generally not useful for segmentation of objects from background.

3.2.2. Laplacian Operators

Laplacian based edge detection operators are based on the second derivative of the image that is the derivative of the gradient image. Laplacian of an image *f*(*x, y*) is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (3.21)$$

Basic approximations of definitions of the first-order partial derivative of *f*(*x, y*) with respect to *x* and *y* are:

$$G_x = \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \tag{3.22}$$

$$G_y = \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y) \tag{3.23}$$

Second-order partial derivatives of *f* with respect to *x* and *y* are the derivatives of G_x and G_y given by:

$$\frac{\partial G_x}{\partial x} = \frac{\partial^2 f}{\partial x^2} = G_x(x, y) - G_x(x-1, y) \tag{3.24}$$

$$\frac{\partial G_y}{\partial y} = \frac{\partial^2 f}{\partial y^2} = G_y(x, y) - G_y(x, y-1) \tag{3.25}$$

Using the equation (3.22) and (3.24) with substituting for G_x yields:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - f(x, y) - f(x, y) + f(x-1, y) \tag{3.26}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \tag{3.27}$$

Using the equation (3.23) and (3.25) with substituting for G_y yields:

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - f(x, y) - f(x, y) + f(x, y-1) \tag{3.28}$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \tag{3.29}$$

Adding (3.27) and (3.29) together, we get Laplacian of *f*(*x, y*):

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \tag{3.30}$$

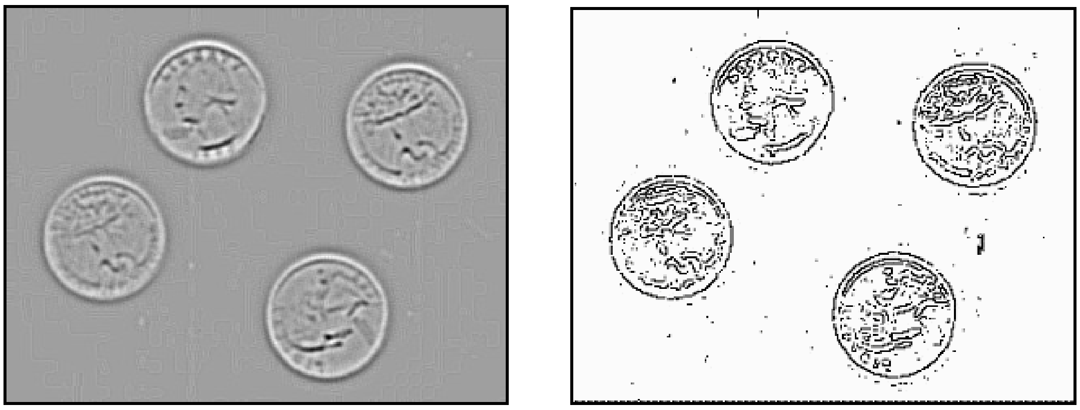
The equation (3.30) is the convolution of *f*(*x*, *y*) with the following Laplacian kernel given in Figure 3.14(a):

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(a) (b)

Figure 3.14 – Laplacian kernels

Another approximation of Laplacian kernel including diagonal neighbors is given in Figure 3.14(b). The Laplacian is very sensitive to noise. Therefore it is not used in its original form for edge detection. In theory, it responses with a strong signal at the beginning of the edge ramp and it responses with a signal with opposites sign at the end of the edge ramp leading a double edge problem. Location of the edge point is obtained on the change in the sign of the Laplacian. This location is called zero-crossing where the Laplacian goes through zero while it changes its sign. Zero-crossing point is a single point showing where the edge point is exactly located and it is more useful for image segmentation than thick Sobel edges.



(a) (b)

Figure 3.15 – Laplacian response and zero-crossings

Figure 3.15(a) shows the image of Laplacian response applied to the image in Figure 3.13(a) using the kernel shown in Figure 3.14(a). Since Laplacian is very sensitive to noise, Gaussian smooth was applied to the original image with standard deviation $\sigma = 1.5$. Laplacian was applied to the smoothed image. The response image values were shifted and scaled for visual purposes. Small negative values are seen as dark gray contours, big positive values are seen as light gray or white contours. Gray areas are where the Laplacian is zero or close to zero. Edge points are hidden in the strong dark-to-light and light-to-dark transitions. Zero-crossings of Laplacian are shown in Figure 3.15(b). To filter out weak zero-crossings, a threshold (T = 10) was used. Thin edges were obtained but false edges were also detected because of the noise. Zero-crossings are useful for edge localization but the response is too noisy although the original image is used without Gaussian noise. Therefore, we use smoothing and threshold to reduce the noise.

In some cases, a combined kernel of Laplacian and Gaussian smooth is used. This is called Laplacian of Gaussian (LoG). LoG is computationally faster and has the same effect as applying Laplacian to a Gaussian smoothed image.

We also miss some true edge points while reducing the effect of noise. Hence, Laplacian is not used alone but it is used together with the first derivative which gives us gradient vector that is useful for edge linking to recover some lost edge points.

Edges on zero-crossing points can also be extracted by finding local maxima points of first derivative. In other words, local maxima points of first derivative are zero-crossings of second derivative. Again, to reduce the noise, generally first derivatives are also used with a smooth operator. A threshold is also used to suppress weak signals.

3.2.3. Canny Edge Detector

Canny edge detection algorithm is an optimal edge detection method developed by J. Canny in 1986. It gives optimal responses to different edge models. According to his studies, a good edge detector must satisfy the following three criteria [12]:

i. Good detection: The algorithm should detect correct edges with a high probability and there should be a low probability of missing a correct edge or detecting a false edge.

ii. Good localization: The algorithm should detect edge pixels of the image as close as true edge points.

iii. Single response: The algorithm should give a single response for a single edge point.

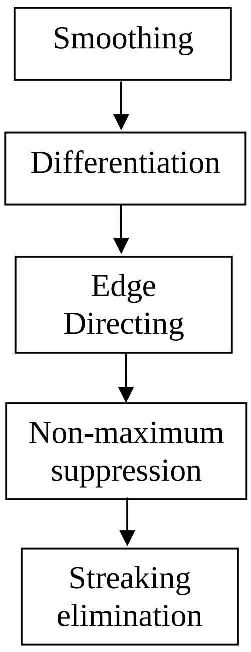
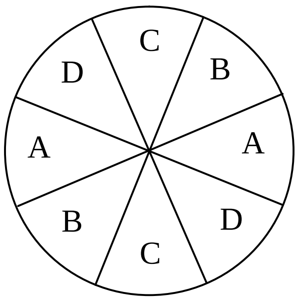


Figure 3.16 – Canny edge detection process

Canny edge detection can be summarized in five steps. In order to satisfy the first criteria mentioned above, a smoothing process must be done to maximize SNR (signal-to-noise ratio) at first step. An isotropic Gaussian smoothing may be applied to the image. Larger Gaussian masks will reduce the detector's sensitivity to noise but it will also slightly increase the error in edge localization. Next step is to find gradients of the smoothed image. A gradient operator such as Sobel may be applied to approximate the edge strengths and edge directions. An extended Sobel operator can also be used for first two steps to reduce the computation process. The purpose of third step is to establish a relation between neighbor edge pixels according to their directions approximated by the gradient operator. Since an edge pixel in an image can be a neighbor of 8 different pixels at most, there exist 4 different orientations for the edge: Up-down, left-right, positive diagonal and negative diagonal. Edges are connected to each other, according to the gradient orientation which the direction of gradient vector is closer.

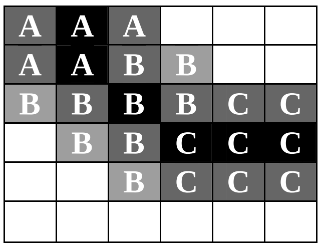


Gradient angle	Orientation
337.5° - 22.5°	A
22.5° - 67.5°	B
67.5° - 112.5°	C
112.5° - 157.5°	D
157.5° - 202.5°	A
202.5° - 247.5°	B
247.5° - 292.5°	C
292.5° - 337.5°	D

(a) (b)

Figure 3.17 – Edge directing diagram and orientation table

Gradient angles are calculated using the equation (3.19). Orientation of the gradient vector is chosen according to the table shown in Figure 3.17. If the orientation is ‘C’, the edge pixel is connected to the pixels to the left and right. If the orientation is ‘A’, the edge pixel is connected to the upper and lower pixels. Similarly, if the orientation is ‘D’, it is connected to up-right and down-left pixels and finally, if the orientation is ‘B’, the edge is connected to up-left and down-right pixels. Note that the direction of gradient vector is perpendicular to the edge direction.



A	A	A			
A	A	B	B		
B	B	B	B	C	C
	B	B	C	C	C
		B	C	C	C

Figure 3.18 – Sample response patch

A sample patch of the gradient operator response is shown in Figure 3.18. Darker areas show the pixels where the gradient magnitude is greater. Letters show the gradient orientations. At fourth step, Canny edge detector applies non-maximum suppression to the response of the gradient operator using the information of gradient vector. The response contains thick edges and the purpose of the step is to obtain thin edges. A 3x3 sized window is passed across the gradient response. Center pixel of the window is compared with its two neighbor pixels which are along the orientation of the gradient vector shown in the diagram in Figure 3.17(a). If the center pixel is non-maximum, that is not greater than the neighbors, it is suppressed.

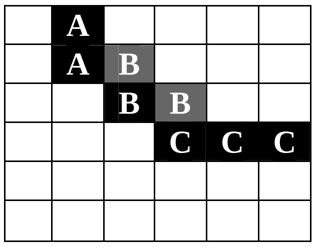


Figure 3.19 – Sample response patch after non-maximum suppression

Remembering that the maxima points are the zero-crossings of the Laplacian, the algorithm tries to find local maxima points that are the strongest pixels for thin edges without using a threshold at fourth step. Furthermore, weak responses such as noise should still be eliminated. By using a threshold, weak responses can be eliminated. However, correct edge pixels may be lost as well as false edges. Using such a threshold may cause "*streaking*" edges that has some discontinuity such as gaps between edge pixels which should be connected to each other. The image shown in Figure 3.15(b) is an example for this phenomenon. J. Canny suggested a "*hysteresis*" thresholding method to overcome the problem. Instead of using a single threshold, two thresholds are used. One of them is called as low-threshold and the other is high-threshold. At final step, responses after non-maximum suppression are used. The responses below the low threshold are eliminated immediately and the responses above the high threshold are accepted as an edge point immediately. If the response is between these two thresholds, the point is a candidate edge point and it is accepted as an edge point if there is a connection between that candidate point and an edge point where the response is above the high threshold. In other words, a candidate edge point is accepted as an edge point if it is 8-connected to an edge point. The final step has a great importance for eliminating the isolated weak signals and recovering correct edge points and generally, the algorithm gives continuous edge points that are suitable to be used as contours of the objects. Object contours are used for segmentation and 3d reconstruction.

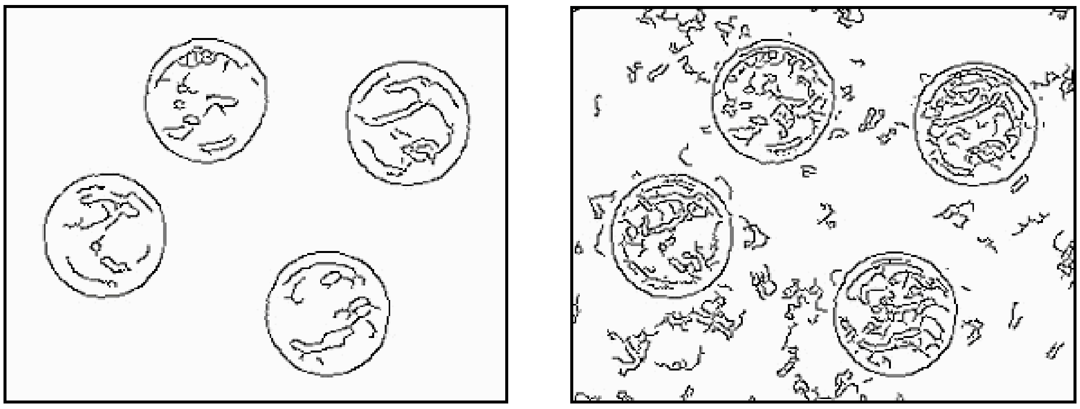


Figure 3.20 – The results of Canny edge detector applied to the image in Figure 3.13(b)

Figure 3.20(a) shows the result edge image of the noisy image in Figure 3.13(b) after Canny edge detector has been applied. For the first step, Gaussian smooth was used with a standard deviation $\sigma = 2$. In the second step, 3x3 Sobel operator was used to approximate gradients. High and low thresholds were chosen as 60 and 20 respectively. The result is clear and edges are denoting the object boundaries successfully despite of existence of noise.

In Figure 3.20(b), the edge image was found using the same parameters as Figure 3.20(a), but the standard deviation of Gaussian smooth was chosen as $\sigma = 1.5$. We have higher detail of edges in the image, but also false edges were obtained because of the noise. In Canny edge detector method, smoothing is very important and it should be configured according to the noise level and the need of detail. More smoothing will decrease the number of false edge points together with the amount of detail.

Another sample result of Canny edge detector, is shown in Figure 3.21. The source image is shown in Figure 3.4. Gaussian smooth with standard deviation $\sigma = 1$ and 3x3 Sobel operator were used. High and low thresholds were selected as 60 and 20 respectively. The method has a problem on corner points as it can be seen at the edge pixels on the corner points of checkerboard pattern in the image. Corner points may be suppressed in non-maximum suppression step since they have a lesser magnitude than the magnitude of edge points. The algorithm simply excludes the corner point and makes the edge point be connected to another edge point which is a neighbor of the corner. Connected edge points are useful for contour matching in stereo images since they are traceable. Edges can be grouped according to the information of that if edges are connected to each other or not. It is also useful for image segmentation process.

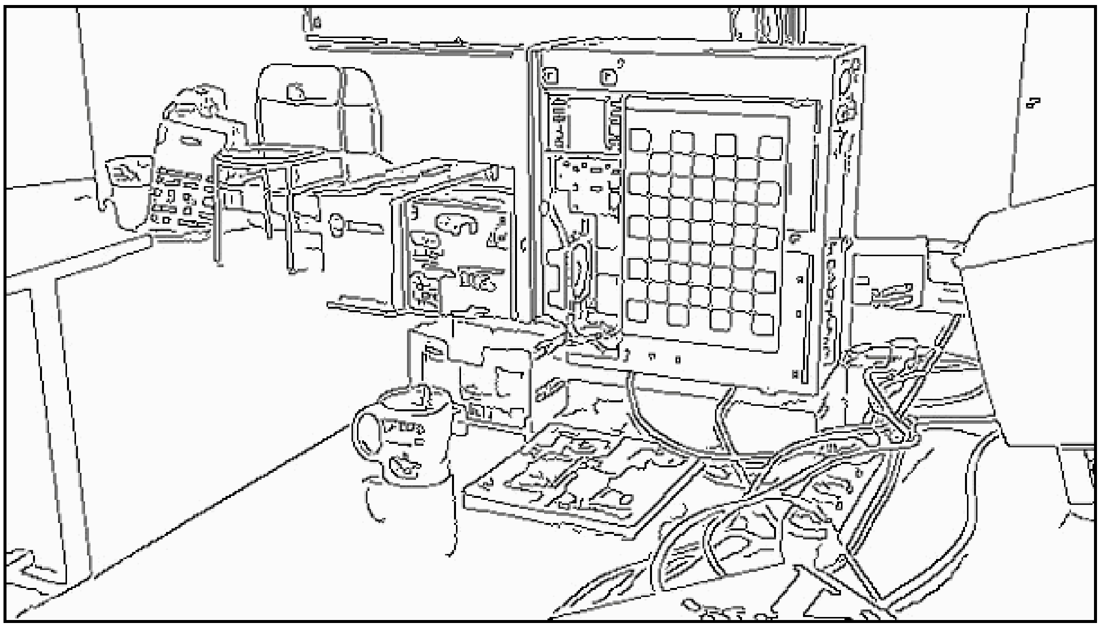


Figure 3.21 – The result of Canny edge detector applied to the image in Figure 3.4

CHAPTER 4

FEATURE-BASED MATCHING

Feature-based matching techniques establish correspondence between similar feature points in each view. Feature points may constitute points, lines or curves corresponding to the corners or edges. Matching process in multiple views simply matches a point in an image to another point where the projection of the point is available in another image. Once two feature points in two or more images are matched to each other, 3D location of the point whose projections are those matched feature points, can be calculated by using epipolar geometry. The process consists of two steps. First step is to extract the feature points in all views. Second step is to find a correspondence between extracted feature points. In first step, corner or edges points may be extracted using related algorithms. In second step, we try to match feature points and find a solution for the correspondence problem.

In stereo images, three major constraints can be used to solve the correspondence problem: Epipolar constraint, disparity constraint and similarity constraint. Disparity is the amount of spatial shift of the same point between two images taken from different views. In stereo images, a point in left image appears in the right image as shifted to the left for the horizontally mounted parallel stereo camera systems. Of course, direction and magnitude of the disparity depend on the location of the point, rotation and translation of stereo cameras with respect to each other. If the disparity is exactly known for a point, the corresponding point in the other view can easily be computed. Epipolar constraint reduces our search region to a line and disparity constraint narrows the search region to very small area. The idea behind using disparity factor for the correspondence problem is that the disparity of neighbor points of a point, whose disparity is known, is close to the disparity of that point and difference of disparities should be below some limit if the neighbor points are connected to that point physically. This is also known as continuity constraint.

Similarity constraint is based on the distribution of the pixels around a point. Correlation of the regions around some certain points can be used for corners as well as the zero-crossings and edge gradient similarities can be used for edges.

If no disparity information is available, the first step should be to extract feature points that can be roughly matched without using the disparity information but an indefinite disparity assumption. Then, we may have a general clue about an average disparity. Corners are suitable for the case.

4.1. Corner Matching

Corners can be extracted using a corner detection algorithm such as Harris corner detector. Corners consist of just single points and ideally, there is one-to-one correspondence between corners (uniqueness constraint). However, extracted corners may have more than one similar corresponding corner as well as they may have no corresponding corner at all. Therefore, corners cannot be forced to be matched and they cannot be matched to more than one corner. A decision should be made to find their matching corners if they have more than one similar corner using the other constraints. As a similarity constraint for corners, correlation can be used to quantify the confidence of a possible match. Since we have different images from different views, the step is called cross-correlation.

Assuming that we have two images of a scene taken at the same time from two cameras such that one of them is slightly rotated and translated with respect to the other camera; we expect roughly similar image patches around corners. Here we assume that a corner of an object may be slightly shifted according to the background so that the background around the corner remains the same in the image patch or the edges going through the corner may be slightly distorted because of the difference in the angle of view. The important matter is that there should not be a significant change around the corresponding corners to match them using correlation. Figure 4.1

shows image patches around the same corner which is extracted from the images taken from left and right camera. The brighter region is a part of the object which is not too close to camera. The points that are closer to camera will be more distorted because the difference in the angle of the view will increase.

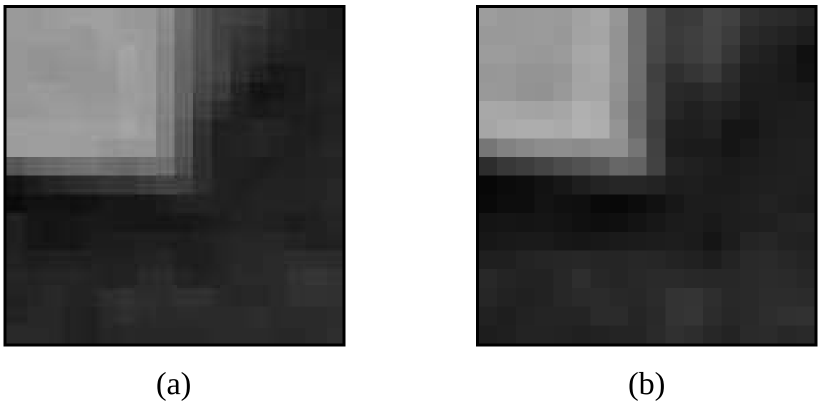

(a) (b)

Figure 4.1 – Sample image patches around the same corner taken from the left (a) and right (b) camera

Cross-correlation can be applied to the small image patches using a window. Suppose that we have a corner c_1 at (x, y) in the first image I and another corner c_2 at (x', y') in the second image I'. Let w be the set of points in a window centered at the origin. Then, the correlation between c_1 and c_2 in the window is given by:

$$c(c_1, c_2) = \sum_{(s,t)\in w} I(x+s, y+t) I'(x'+s, y'+t) \tag{4.1}$$

Normalized images should be used for the formula in (4.1), so the intensity values in I and I' should be between 0 and 1. The correlation is sensitive to changes in the amplitude of I and I'. An approach to overcome this problem is to perform matching using correlation coefficient which is suitable for non-normalized intensity values and invariant to scaling the intensity values. Correlation coefficient between c_1 and c_2 in the window is given by:

$$\gamma(c_1,c_2)=\frac{\sum_{(s,t)\in w}\left[I(x+s,y+t)-\bar{I}_w\right]\left[I'(x'+s,y'+t)-\bar{I}'_w\right]}{\sqrt{\sum_{(s,t)\in w}\left[I(x+s,y+t)-\bar{I}_w\right]^2\sum_{(s,t)\in w}\left[I'(x'+s,y'+t)-\bar{I}'_w\right]^2}} \tag{4.2}$$

$\bar{I}_w$ and $\bar{I}'_w$ are the average intensity values of image patches within the window that are defined in (4.3) and (4.4).

$$\bar{I}_w=\frac{1}{s(w)}\sum_{(s,t)\in w}I(x+s,y+t) \tag{4.3}$$

$$\bar{I}'_w=\frac{1}{s(w)}\sum_{(s,t)\in w}I'(x'+s,y'+t) \tag{4.4}$$

Correlation coefficient γ is scaled in the range -1 to 1 and denotes the similarity between the corners c_1 and c_2. If γ increases, the similarity increases. In order to establish a one-to-one (but not onto) relationship between the corners extracted from two images, we need to satisfy that each corner should be matched to at most one corner. Let A be the set of points of corners extracted from the first image and let B be the set of points of corners extracted from the second image. There is a connection between $c_1\in A$ and $c_2\in B$ if all of the following conditions hold:

$$\gamma(c_1,c_2)>T \tag{4.5}$$

$$\gamma(c_1,c_2)>\gamma(a,c_2)\ \text{for}\ \forall a\in A, a\neq c_1 \tag{4.6}$$

$$\gamma(c_1,c_2)>\gamma(c_1,b)\ \text{for}\ \forall b\in B, b\neq c_2 \tag{4.7}$$

If the correlation coefficient between two corners is below some threshold T, we immediately eliminate the possibility of being a connection between them (condition in 4.5). This condition also allows a corner point to be left without being matched to any other corner if it has no strong similarity with any corner. Using a threshold to filter out weak connections prevents some false matches. However, it may also prevent to connect a corner to its real match if it is too much distorted. Nevertheless, a corner may be more similar to another corner even if the corner is not its real match. Therefore, using a threshold generally helps to eliminate some unwanted matches. In condition (4.6), we check that a matching between c_1 and c_2 has a greater similarity value for all other possible matches to c_2. Similarly, in condition (4.7), we check that a matching between c_1 and c_2 has a greater similarity value for all other possible matches to c_1. In other words, the correlation coefficient between the pair c_1 and c_2 is the maximum value of correlation coefficient values of all combinations of pairs containing c_1 or c_2 if there is a connection between c_1 and c_2. It is slightly different than finding the most similar match of a corner. If c_1 has its maximum correlation value with c_2, the pair c_1 and c_2 is a candidate but the result is not certain. In this case, if c_2 has a greater correlation value with a corner other than c_1, then there is no connection between c_1 and c_2. Now, the corner c_1 should be tried to be matched to another corner that has the greatest similarity value just after c_2 satisfying the condition given in (4.5). This process may be implemented using the following algorithm with a simple approach:

```
Create a table $t_1$ containing entries of all combinations of the corner pairs from the
sets $A$ and $B$ with their correlation coefficient values ($\gamma$) satisfying $\gamma > T$.
Create a new empty table $t_2$.
For all entries $k$ in $t_1$, // k = (matching corner, matched corner, correlation value)
    Set flag to 1.
    For all entries $j$ in $t_1$,
        If the matched corner in the entry $k$ is the same corner as the matched
        one in the entry $j$ and the correlation value in $k$ < the correlation
        value in $j$,
            Set flag to 0.
    If flag is 1, add the entry $k$ into the table $t_2$.
```

The algorithm above simply checks all corner pairs whether they satisfies the conditions given in (4.5), (4.6) and (4.7). The table t_2 will have all one-to-one matches between the corners from the sets A and B.

Since the angle of view is slightly changed in stereo cameras, the correlation coefficient values between correct matches will probably be above 0.7, so that the threshold may be chosen as 0.7. However, the major problem here is that the best match between two corners does not correspond to the real match always. In order to reduce the effect of the problem is to use another constraint narrowing the search region. We expect that objects in an image taken from the second camera will be slightly shifted to some distance according to the image taken from the first camera. Therefore, the matched corner points should not be too far but at the same time should not be too close, so they should have some certain distance between them. That distance is up to orientations and positions of the cameras and z-values of the corners according to the cameras. Since the depth information is not known yet, the distance should be chosen according to the size of images and approximate average disparity. For instance, if we assume that an approximate shift rate of the most of the objects is about 20% of the image size, we can estimate the expected distance as 20% of the image diagonal length. This process is quite conjectural and can be used to prevent absurd matches. If the epipolar constraint is absent or not trustable, it can be the first step to have some correct matches. If the distance between the corners of a candidate pair is different from the expected distance, the correlation value between them should be reduced with respect to the difference of distances. Let (p,q) be a pair from the set AxB where $p \in A$ and $q \in B$. The Euclidian distance between *p* and *q*, is defined as:

$$d = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{4.8}$$

Now, we define a penalty factor as a coefficient of the correlation coefficient such that zero-penalty should be given to the correlation value if the difference of distances is zero:

$$m = \frac{1}{1 + w|d - e|} \tag{4.9}$$

where *d* is the measured distance, *e* is the expected distance and *w* is the weighting coefficient for the difference of distances. The weighting coefficient is chosen as a small positive real number such that $w|d-e| \leq 1$. Hence, $w = 1/(\max(d) - e)$ holds. The purpose of using such a coefficient is to make the penalty factor independent from the image size and it can be chosen as the inverse of the maximum possible difference of distances. The maximum distance between *p* and *q* cannot be greater than the image diagonal length. The weighting coefficient is approximately equal to 1 / (image diagonal length - expected distance). The correlation coefficient value between *p* and *q* now becomes:

$$\gamma' = m\gamma(p,q) \tag{4.10}$$

m is the penalty factor in (4.10). The new correlation coefficient (γ') value can be at least the half of the old value when the difference between *d* and the expected distance has its maximum value. If the difference is zero, the correlation value is kept as the same. The weighting coefficient should be increased in order to increase the effect of penalty factor. Previously given algorithm can easily be modified by using this approach. The algorithm will give more chance to the pair (p,q) if the distance between *p* and *q* is closer to the expected distance. The correlation coefficient value of too distant corners will be reduced to prevent to be selected as a match.

To find more reliable matches, the epipolar constraint should also be taken into consideration. If epipolar constraint is available, some improvements can be made for the algorithm. If it is not known or not sufficiently reliable, the fundamental matrix of the epipolar constraint can be recovered using known corner matches. However, false matches may be present although similarity and disparity constraints are both used. We can gather the most reliable matches to compute the fundamental matrix or RANSAC algorithm may directly be applied.

We expect that some group of corners that belong to the same objects, are close to each other in z-axis so that they have approximately same disparities. Directions of the disparity vectors will also be close to each other. But, a false match will probably have an irrelevant disparity vector since we do not force the disparities of corners to have some specific direction in correlation process. Some reliable matches can be found by finding groups of matches having approximately same disparity vector. The group which is containing the maximum number of corners may be used to compute the fundamental matrix. If we iteratively compute the average disparity of all matches and eliminate some of them to find the new average, we can find such a group.

Let *M* be the set of known matched pairs (p_i, q_i) where $0 \le i < s(M)$, p_i denotes the corners extracted from the first image and q_i denotes the corners extracted from the second image. Disparity vector of the pair (p, q) is given by:

$$v(p,q) = q - p = (q_x - p_x, q_y - p_y) \qquad (4.11)$$

The average disparity of matched pairs in set *M* is given by:

$$\mu_M = \frac{1}{s(M)} \sum_i v(p_i, q_i) \qquad (4.12)$$

The average disparity is also a vector determining the magnitude and direction of the average shift. The distance from a disparity vector to the average disparity for the pairs in the set *M* is given by:

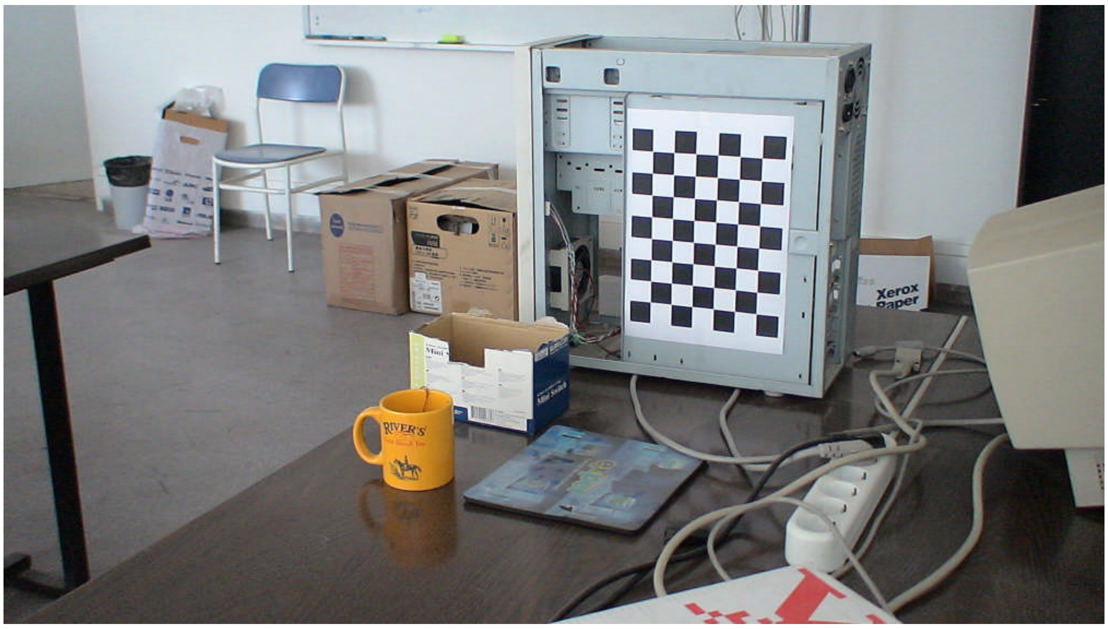
$$f_M(p,q) = \left| v(p,q) - \mu_M \right| \qquad (4.13)$$

Now, we construct sets of pairs M_j containing the pairs taken from set M_{j-1} such that M_j has round[k.s(M_{j-1})] elements whose corresponding $f_{M\,j-1}$ values are the smallest among the elements of the set M_{j-1} where $M_0 = M$ and *k* is a threshold value satisfying $0 < k < 1$. Thus, we eliminate some of the pairs whose disparity vector is distant from the average at each of the iterations. Iterations may continue a few times or until the standard deviation of the values of *f* given in (4.13) for the set M_j is below an acceptable value. The process may be implemented using the algorithm given below.

```
Let $M$ = Array of given pairs, $N$ = Empty array, $k$ = Threshold value,
$s$ = accepted standard deviation, $m$ = maximum number of iterations.
$i = 0$;
do
    $\mu = 0$;
    for all pairs $(p, q)$ in $M$
        compute $v$ for $(p, q)$;
        $\mu = \mu + v$;
    $\mu = \mu / \mathrm{s}(M)$;
    for all pairs $(p, q)$ in $M$
        compute $f$ for $(p, q)$;
        add $f$ with the pair $(p, q)$ into $N$;
    sort $N$ according to $f$;
    empty $M$;
    add the first round[$k$*s($N$)] elements of $N$ into $M$;
    empty $N$;
    compute the standard deviation $\sigma$ for $f$ values in $M$;
    $i = i + 1$;
while $i < m$ and $\sigma > s$
```

The algorithm iteratively finds a dominant group which consists of corner matches having approximately same disparity. Standard deviation of the distance values to the average disparity is below a threshold. Choosing the standard deviation threshold too low, or to allow the iteration repeating too many times may cause to eliminate most of the matches so that the result may not meet the minimum number of matched points. The algorithm may be improved to take the number of matched points available into the consideration.

Figure 4.2 shows stereo test images with a resolution 2016x1134 pixels that have been applied cross-correlation using similarity constraints given in (4.5), (4.6), (4.7) with weighted correlation coefficients using the formula given in (4.10). Corners were extracted using Harris corner detector [9] with 3x3 binary window and Harris free parameter $k = 0.04$. Gradients were supplied by the 7x7 extended Sobel kernels shown in Figure 3.5 without any extra smoothing process. 1174 corners were extracted from the left image and 1217 corners were extracted from the right image by detecting the local maxima points in a 5x5 window with a threshold 0.1. Then, the extracted corners were cross-correlated using correlation coefficients within a 31x31 window. The correlation values were weighted using an approximate average disparity assumption. The parameters of the assumption were chosen as e = 150 pixels, w = 0.0005. The connections between corners were established with respect to a threshold T = 0.7 that is used for the relation given in (4.5). 498 one-to-one correspondences were found between the corners of left and right images. The extracted corners are shown as (blue) circles and the connections between the corners are denoted as (yellow and red) lines in the Figure 4.3. Connections denoted by the parallel lines are the probable correct matches. In the regions such as the calibration pattern which contains similar corners, available connections are generally false matches. To find some reliable matches, we iteratively eliminated the matches by selecting k = 70% of the matches closest to the average disparity at each iteration. In Table 4.1, the table shows the results of matching elimination process. The first row indicates the state before iterations. Iterating the process 5 times yielded 85 matches with a slightly different average disparity from our first assumption. The standard deviation reduces at each of the iterations, indicating that the disparity vectors of matched corners are getting closer to the average disparity. The red lines in the images shown in Figure 4.3(a) and 4.3(b) indicate the corner matches that are the output of the matching elimination process. Almost all of them are correct matches. In order to have a greater number of reliable corner matches, the next step is to use the epipolar constraint, recovering the fundamental matrix using RANSAC or 8-point algorithm with the reliable matches we have. If the epipolar constraint is already known or sufficiently accurate, there is no need to compute the fundamental matrix and matched corners can be found by using all known constraints at once. As a disparity constraint, an assumption may be used to have more reliable matches since the exact average disparity is not known at the beginning.


(a) Left image

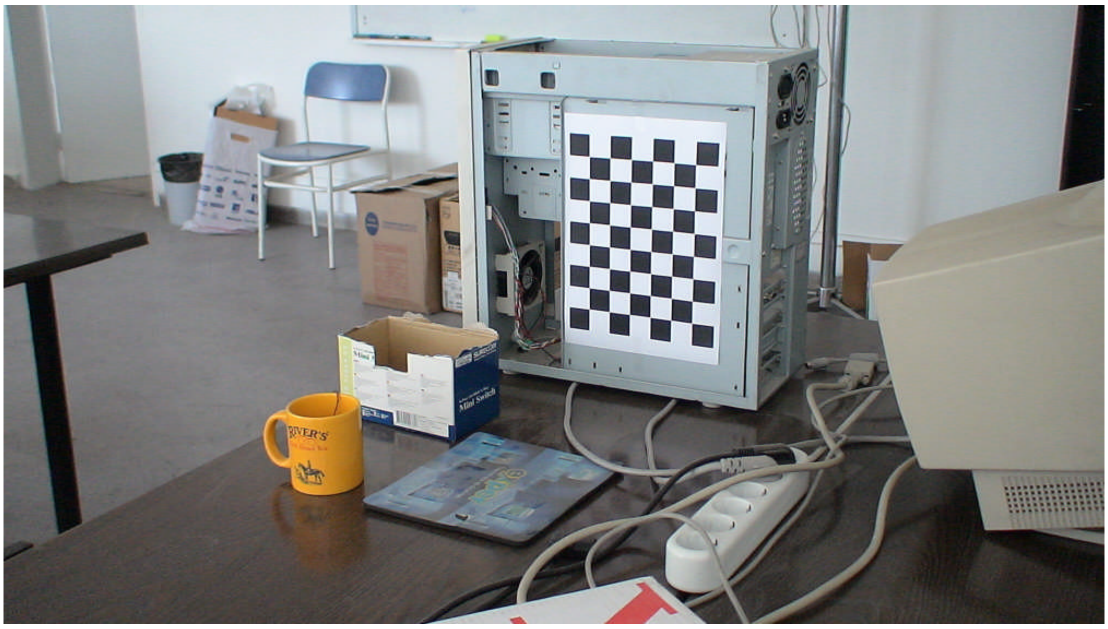
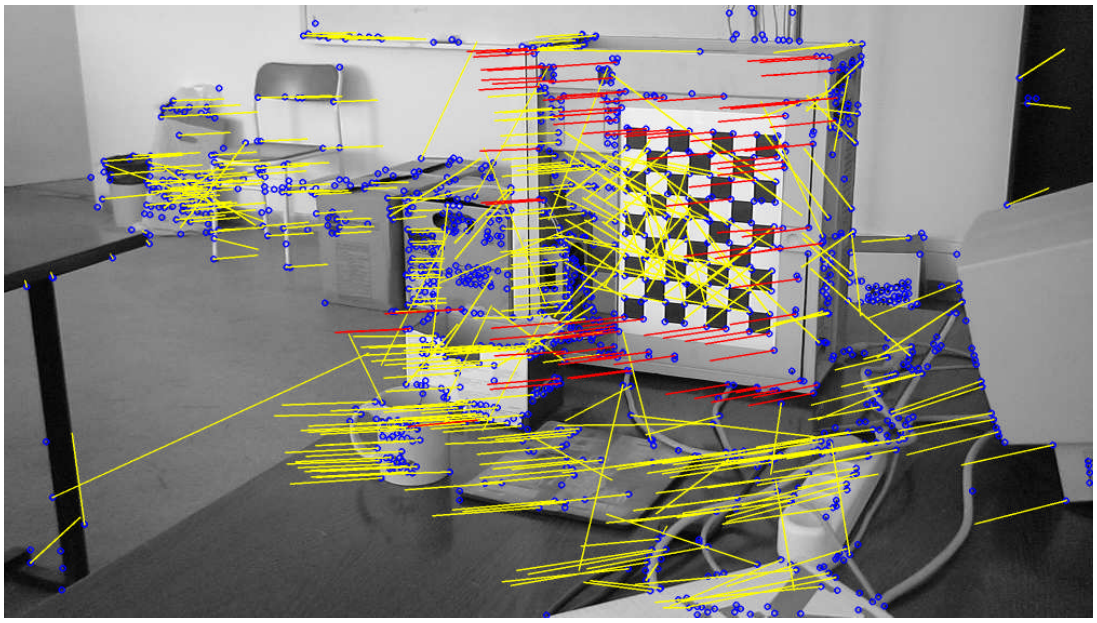
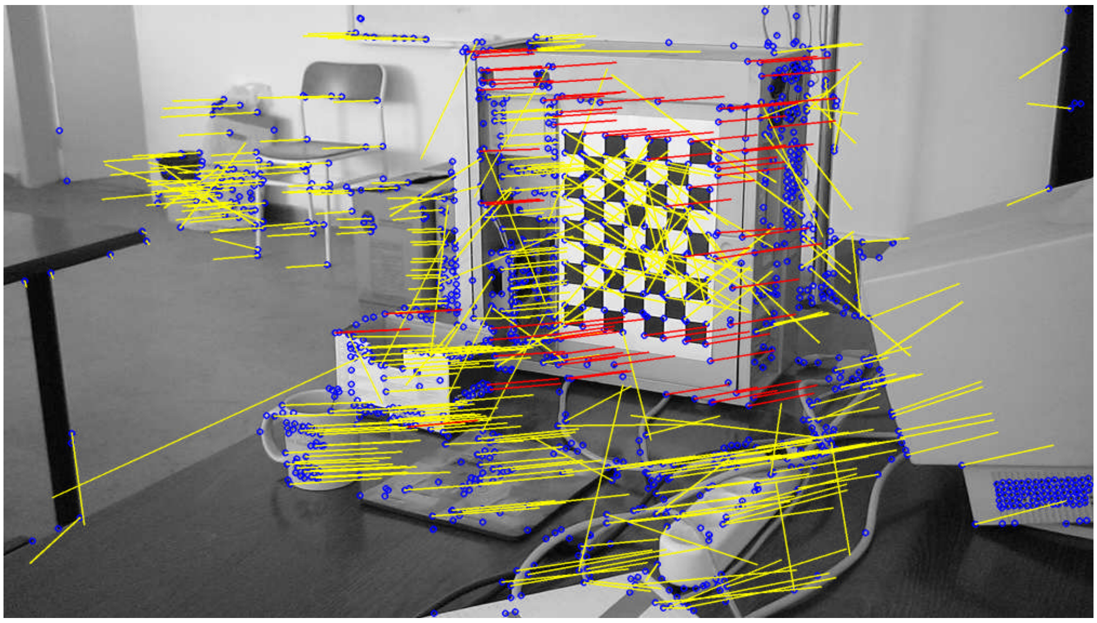

(b) Right image

Figure 4.2 – Stereo test images
The image (a) has been taken from the left camera
and the image (b) has been from the right camera.



(a) Left image



(b) Right image

Figure 4.3 – Matching results of test images using cross-correlation and approximate disparity information (yellow lines) and reliable matches (red lines)

Table 4.1 – Results of the matching elimination process

Iteration	**# of matches**	**Magnitude of the average disparity**	**Standard deviation**
Initial	498	75.714735	126.100822
1	349	117.442554	65.283710
2	245	128.123615	28.200456
3	172	122.020385	16.253874
4	121	120.319887	11.388010
5	85	118.815721	7.926933

Using the epipolar constraint additionally yields better results for corner matching process. We use the fundamental matrix to find the possible matches for a given point on the image planes. Any possible corner pair (*m*, *m'*) satisfies the equation given in (2.19). By using this constant, together with the other constraints, we can have more reliable matches. However, because of the error in the fundamental matrix and the location of corner points, no pair would exactly satisfy the equation. Instead of using the equation directly, the epipolar line corresponding to a corner point may be used. Assume that we have a candidate corner pair (*m*, *m'*) where *m* and *m'* are points on the first and second image respectively. The epipolar line corresponding to the point *m* should pass close to their possible satisfiers. If the minimum distance from the point *m'* to the line is below a threshold, the candidate pair is then checked for the other constraints. If it is too distant to the epipolar line, it is immediately eliminated. The epipolar line corresponding to the point *m* is given by:

$$l' = F\tilde{m} = \begin{bmatrix} a & b & c \end{bmatrix}^T \tag{4.14}$$

where $\tilde{m}$ is the augmented vector of the point *m*. The parameters of the epipolar line are *a*, *b* and *c* such that $ax + by + c = 0$. For the point $m'(x', y')$, the minimum distance to the line is given by:

$$\varepsilon(l', m') = \frac{\left|ax'^2 + by'^2 + c\right|}{\sqrt{a^2 + b^2}} \tag{4.15}$$

The candidate corner pair (*m*, *m'*) is eliminated if the following inequality holds:

$$\varepsilon(l', m') > \varphi \tag{4.16}$$

where φ is a threshold. Using this approach, reduces the probability of selecting a false match and computation time for matching process since the number of the corner pairs that are to be cross-correlated reduces. The pairs satisfying the epipolar constraint are then checked for the similarity constraint. Magnitude of the average disparity computed in the matching elimination process may be used for disparity constraint instead of an assumption. If the epipolar constraint is not sufficiently accurate, some of the correct matches may also be eliminated together with the false matches. Increasing the threshold φ may not be a good solution since the number of false matches will also increase. Accuracy of matching process depends on the accuracy of the fundamental matrix. Since we compute the fundamental matrix using the matched corners, the accuracy of the fundamental matrix depends on the accuracy of matching process. If we iteratively compute the fundamental matrix and match the corners, we can find more reliable corner pairs. At each iteration, the average of the magnitude of disparity vectors should be recomputed. It is different from the average disparity as computed in (4.12). This time, we compute the average of vector magnitudes since there may also be some far objects with reverse shift. Algorithm of the iterative process is like as follows:

```
$\alpha$ = magnitude of the last known average disparity or an assumption for disparity constraint
$T$ = a threshold for similarity constraint
$\varphi$ = a threshold for epipolar constraint
$k$ = number of iterations
for $k$ times
    Match corners using all known constraints ($\varphi$, $\alpha$, $T$).
    Re-compute the fundamental matrix.
    $\alpha$ = the average of the magnitude of disparity vectors.
Finally, match corners one last time using all known constraints ($\varphi$, $\alpha$, $T$).
```

We applied the algorithm to all corner pairs available in the images shown in Figure 4.3. The initial fundamental matrix was computed using RANSAC algorithm for 85 reliable matches extracted by the matching elimination process. The probability parameter *p* of RANSAC algorithm was chosen as 99% and the outlier threshold parameter was chosen as 1 pixel. Therefore, our expectation is that with 99% probability, the epilines will pass through the inliers with a 1 pixel error. For the disparity constraint, the initial average magnitude of disparity vectors was chosen as the same as the output of the matching elimination process shown in Table 4.1. After each iteration, it was recomputed as mentioned in the algorithm. The other parameters remained the same as used in previous matching operation (w = 0.0005 and T = 0.7). The epipolar constraint's threshold was chosen as φ = 2 pixels. As it seems from the test results, the accuracy of the epipolar constraint increased at each step and the final matching operation recovered (393 matches) about 80% of 498 matches that had been found without using the epipolar constraint.

Table 4.2 – Iterative matching results

Iteration	**# of matches**	**Magnitude of the average disparity**
Initial	85	118.815721
1	237	119.018957
2	279	118.777131
3	309	121.469930
4	373	120.780041
5	392	118.651585
Final	393	116.933613

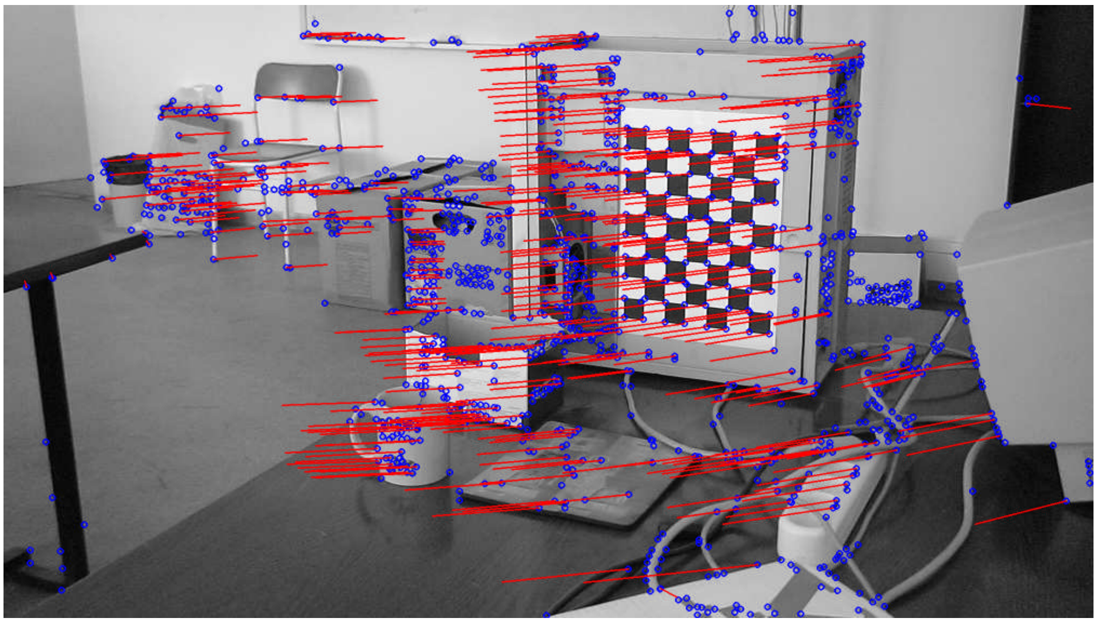
If the fundamental matrix is computed from the calibration information, the final matching can be done without iterating. This will remove a lot of computation. We also applied the final matching operation directly using the fundamental matrix computed using the calibration information which is extracted using the calibration pattern in the image. The results show that it did not cover as more points as the iterative matching. However, the result is much better than the first iteration of iterative matching as shown in the Table 4.3.

Table 4.3 – Matching results using calibration information

State	**# of matches**	**Magnitude of the average disparity**
Initial	85	118.815721
Final	319	127.268183

Figure 4.4 shows matching results after the iterative matching. The accuracy is better compared to the matching results without using the epipolar constraint (Figure 4.3). A few false matches are still available. But, most of them are reliable match. The corners extracted from the far objects that have reverse shift have been matched correctly.

Correct corner matching and number of matched corners are extremely important for the next step, contour matching. False corner matches may mislead the contour matching. In general, the number of matched contour points of objects increases as the number of corners matched increases if the extracted corners are well-distributed along the contour paths.


(a) Left image

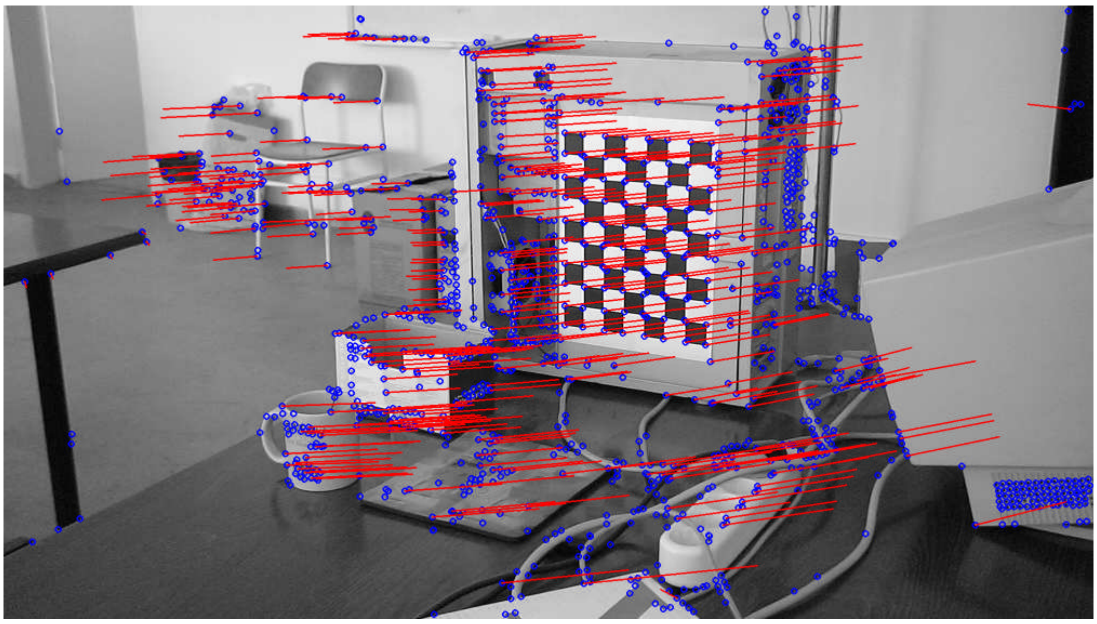

(b) Right image

Figure 4.4 – The final matching results after iterative matching

4.2. Contour Matching

More detailed 3D information can be extracted by finding matches of the contour points of an object in a scene. Thin edge points extracted by an edge detector such as Canny edge detector can be used as the contour points in the image. Edges should be as thin as possible to increase the reliability of the contour matching process. The uniqueness constraint cannot be used for matching the edge points since a long edge segment in the right image may be detected as a short edge segment containing less edge points in the left image. Epipolar and disparity constraints have the vital importance for the contour matching. The idea behind the contour matching is that neighbor contour points should have approximately the same disparity. If we know some seed points on contours whose disparity vectors are known, the estimation of disparity vectors of some contour points is possible by tracking contours along their neighborhoods. There is no one-to-one correspondence between contour points. Contour points are just sample points on the edges of objects whose approximate locations are known. The edge pixels can give the location of a contour point roughly. Therefore, establishing a direct relation between edge pixels may not yield accurate results. Contour matching process can be summarized in three steps:

a) Contour extraction: Extraction of contours is based on detecting gradients and can be obtained by the zero-crossings of the first derivative or thin edge detectors. Canny edge detector is suitable.
b) Finding seed points: Seed points are some starting points that are on contours and have a known disparity vector. Their disparity information is used for their neighbors. Seed points can be chosen from the known matched pairs.
c) Estimation of disparity vectors of contour points: Disparity vectors of contour points are estimated starting from the seed points. Neighbors of seed points are matched by using all known constraints and then neighbors of neighbors are tried to be matched until all possible contour points are matched. Disparity vector of a contour point is the difference of the location vectors of the matched contour points on the left and right image.

To start to match the contour points from the first image to the contours to second image, seed points are needed. Seed points can be obtained from the pre-matched corner points which are on an edge point or close to an edge point. Let *M* be the set of known matched pairs (p_i, q_i) where $0 \le i < s(M)$, p_i denotes the points on the first image and q_i denotes the points on the second image. The set *M* may be constructed via the corner matching process. Let C_1 be the set of contour points extracted from the first image. Similarly, let C_2 be the set of contour points extracted from the second image. Then, a point $s \in C_1$ is a seed point if all of the following conditions hold:

$$d(s, p_i) < d(c, p_i)\text{, for } \forall c \in C_1,\ c \ne s \qquad (4.17)$$

$$d(s, p_i) < T \qquad (4.18)$$

where *d* is the Euclidean distance between given two points as formulated in the equation (4.8) and *T* is a threshold to guarantee that two points are close enough. The approximate disparity of the seed point *s* is $D(s) = v(p_i, q_i)$ where *v* is the vector difference given in the equation (4.11) and q_i is the corresponding matched point of p_i in the pair. For each pair in *M*, we accept contour points as seed points if the contour point is the closest point to the given point *p* in the pair with a distance below some threshold *T*. Therefore, if the conditions in (4.17) and (4.18) hold, we assume that the contour point and the given point whose disparity information is known have approximately the same disparity.

A basic approach to implement the method is to test the conditions (4.17) and (4.18) for each contour points with all elements of the set *M*. If the contours are available in an image and accessible by their coordinates, the seed points can easily be found by placing a small circular window with a radius *T* on each point p_i that is available in the pairs contained in *M*. If the window contains some contour points, the closest one will be selected as a seed point. For small windows, using a square window is an option to reduce the computation time. For each point p_i, 8-neighbors of the point are searched when a 3x3 window is used.

After all seed points are collected, their approximate disparity D is used to compute the estimated disparity of themselves. Then, the estimated disparity of seed points are used as the approximate disparity of their neighbors. Hence, any point on the contour whose approximate disparity is known becomes a new seed point and the process repeats until all possible points on the contours are processed. Let s be a seed point in the set C_1. The approximate location of the matched contour point s' corresponding to s is given by:

$$L(s') = s + D(s) \tag{4.19}$$

Since $D(s)$ is accepted as the disparity of the point of which s is a neighbor, the disparity of s should be slightly different than $D(s)$. We expect that s' should exist at a location that is close to $L(s')$. The epipolar geometry helps to find probable contour points around $L(s')$. The epipolar line corresponding to s is given by:

$$l' = F\tilde{s} \tag{4.20}$$

where F is the fundamental matrix and $\tilde{s}$ is the augmented vector of the point s. The line l' is passing through the points closer to the some probable contour points such that some of them are closer to s', but some of them are completely not related. The epipolar constraint is awkward when it is used alone. However, the seed point indicates a region of interest which is located at $L(s')$. The set of the probable contour points corresponding to s is given by:

$$P_s = \{x \mid \varepsilon(l', x) < \omega, x \in W_{L(s')} \cap C_2\} \tag{4.21}$$

ε is shortest distance to the epipolar line (4.15), ω is a threshold restricting x to be sufficiently close to l'. $W_{L(s')}$ is the set of all points in the region around $L(s')$. The intersection of W and C_2 is the set of contour points in the second image that are inside the region. The set P_s contains the probable contour points extracted from the second image corresponding to the contour point s from the first image. Since there is no one-to-one relationship between contour points, the set may contain more than one point. The both sets C_1 and C_2 contain the points that have been found via estimation, so the match of an estimated contour point would be estimation again. A basic approach to find the estimated location of the matching point *s'* corresponding to *s* is to compute average position of the points contained in P_s which is given by:

$$r = \frac{1}{n} \sum_{x \in P_s} x \tag{4.22}$$

where *n* is the number of elements in P_s. The approach may be improved such that the points closer to the epipolar line have more weight to reduce the effect of the points which are less related to the point *s'*. The linearly weighted average of the points in P_s is given by:

$$r = \frac{1}{w} \sum_{x \in P_s} \left[\omega - \varepsilon(l', x)\right] x \tag{4.23}$$

$$w = \sum_{x \in P_s} \left[\omega - \varepsilon(l', x)\right] \tag{4.24}$$

The total weight is denoted by *w*, and ω is the threshold used in (4.21). The estimated matching point location *r* given by (4.22) or (4.23) may be used to find the closest line point on *l'* to *r* for another improvement. Let $\tilde{r}$ be the augmented vector of the point *r*. The closest line point *c* on $l' = \begin{bmatrix} a & b & c \end{bmatrix}^T$ to *r* is the intersection of the epipolar line *l'* with the orthogonal line to *r*, that is:

$$c = \frac{1}{a^2 + b^2} \begin{bmatrix} b^2 & -ab & -ac \\ -ab & a^2 & -bc \end{bmatrix} \tilde{r} \tag{4.25}$$

Since the matching point of *s* is a point on the line *l'*, the closest line point *c* may also be used as the estimation to *s'* considering the reliability of the fundamental matrix.

According to the method proposed by Han and Park [13], the point *c* is used as an initial estimate of *s*′. Then, within a small window placed on *c*, the point is selected as the final estimation that gives the maximum correlation value compared to the point *s*. Thus, the matching contour point *s*′ in the second image corresponding to the contour point *s* in the first image is given by:

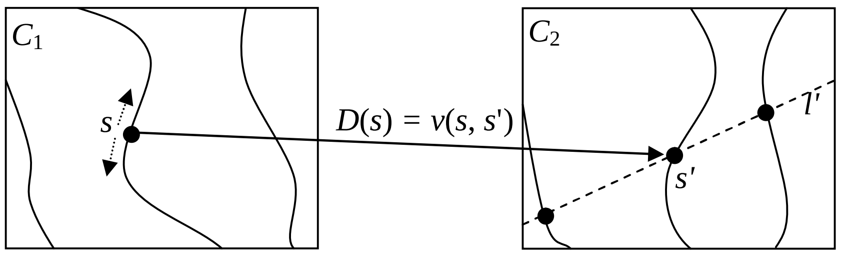
- The average location of the points contained in P_s that is *r* given by (4.22) or (4.23), or;
- The closest line point *c* on the epipolar line (4.25), or;
- The point that has the maximum correlation value with *s* around *c* inside a window.

Let s' denote the estimated location of the matching point of *s* and let *N* be the set of neighbor points of *s*. Then, the approximate disparity of the neighbors of *s* is given by:

$$D(m) = v(s, s') \text{ where } m \in N \tag{4.26}$$

Since the set *N* has some points whose approximate disparities are known, the points contained in *N* is used as new seed points. The process repeats for a new seed point in *N*. As the estimated locations of the matching points of the points in *N* are computed, their disparities will be used as the approximate disparity of their neighbors to create newer seed points. When a seed point is used for creating a new point, it is not used anymore. If the matching point of a contour point is already known, it is not used as a new seed point. Therefore, the process will stop eventually when no other seed point is available.

Figure 4.5 shows some example contour points in left and right image. Although the epipolar line *l*′ intersects with the many contour points, the disparity vector *v* shows the closest contour points for the neighbors of *s*. We can summarize the process that the epipolar line determines the estimated position as the disparity vector chooses probable contour points. The major problem of the approach is encountered when the epipolar line is parallel to the direction of the contour. The intersection region of the line and the contour is large in this case, which leads a bad estimation of the matching point location. Therefore, using a bad estimated disparity as the approximate disparity of a new seed point may propagate the error that results in missing the contour points when the epipolar line passes by too far from any probable matching contour point or no contour point is available in the window *W*.



Left image Right image

Figure 4.5 – Contour matching using epipolar geometry starting from a seed point

Choosing a larger window is not a good solution. A larger window will contain the contour points from the same contour in a larger area. Although the equation (4.23) gives more weight to the contour points which are closer to the epipolar line reducing the amount of error, a large window may also contain irrelevant contour points from other contours that will effect the location of the matching corner.

The error rate may be reduced if the disparity information of more than one seed point is used along the same contour. Let s_1 and s_2 be two seed points on the same contour. It is known that while moving along the contour from one seed point to another, the disparity will change beginning with the disparity of the seed point. While getting closer to the other seed point, the disparity will get closer to the disparity of the other seed point. According to the model, if the rate of the change in disparity along the contour is fixed, the disparity of any contour point between s_1 and s_2 can be successfully pre-estimated. However, the disparity usually changes with a variable rate, so the estimation becomes unreliable when getting far from seed points. Using the pre-estimation together with the approximate disparity of the contour point as mentioned in the equation (4.26) may reduce the propagation of the error.

The pre-estimated disparity of a contour point $c \in C_1$ is given by:

$$P(c) = \frac{n(c)}{n(s_2)}\left[D(s_2) - D(s_1)\right] + D(s_1) \tag{4.27}$$

$n(c)$ denotes the contour length between *c* and s_1, and $n(s_2)$ is the total contour length between s_1 and s_2. Then, $n(c)/n(s_2)$ is the completion ratio of the path along the contour. Hence, *P*(*c*) is a linear interpolation of the disparity of the point *c*. Note that if we substitute for *c* = s_1, we have the ratio $0/n(s_2)$ since $n(s_1) = 0$, thus we have $P(s_1) = D(s_1)$, the same disparity approximation of s_1. Similarly, $P(s_2) = D(s_2)$ holds for *c* = s_2. Assuming the contour points are homogenously distributed along the contour, the number of contour points along the contour can be used instead of contour length for the function *n*. Let z_c be the minimum distance (or the minimum number of contour points as mentioned in the assumption) from the contour point *c* to the contour segment tips s_1 and s_2. The formulation of z_c is given below:

$$z_c = \begin{cases} n(c) & n(c) < \dfrac{n(s_2)}{2} \\ n(s_2) - n(c) & otherwise \end{cases} \tag{4.28}$$

The distance z_c indicates unreliability of the pre-estimation having a value in the range $\left[0, n(s_2)/2\right]$. The unreliability increases while z_c increases. Considering z_c, we construct a new approximate disparity as follows:

$$D'(c) = \frac{z_c D(c) + \left[\dfrac{n(s_2)}{2} - z_c\right] P(c)}{\dfrac{n(s_2)}{2}} \tag{4.29}$$

As shown in the Figure 4.6, the function given in (4.29) is a linear weighting function that relies on *P*(*c*) more than *D*(*c*) when *c* is closer to the tips (closer than the quarter path). From the first quarter to the third quarter, $D(c)$ is more reliable. At the mid-point of the path, it is equal to $D(c)$ indicating that $P(c)$ has no effect.

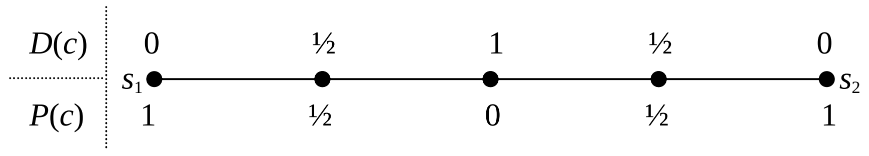


Figure 4.6 – The pre-estimated and approximate disparity weights along a contour segment

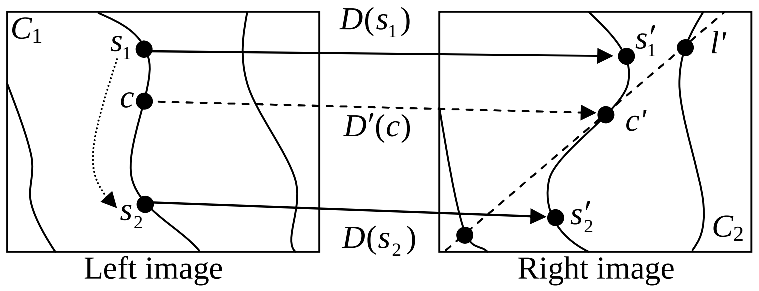


Figure 4.7 – Contour matching using epipolar geometry between two seed points

Figure 4.7 shows two matched seed points, s_1 and s_2 whose approximate disparity vectors are $D(s_1)$ and $D(s_2)$ respectively. A point c between s_1 and s_2 on the contour has an approximate disparity vector $D'(c)$ as defined in (4.29). The epipolar line l' intersects some contours and passes slightly parallel with respect to the related contour. The approximate disparity of the point c depends on both its neighbor's disparity that has been estimated before, and the pre-estimated disparity between s_1 and s_2. Many contour points are too close to the epipolar line which makes the location of the match uncertain. The uncertainty in the pre-estimation also increases while going far from the seed points. The approximate disparity vector $D'(c)$ has dynamically weighted estimations to reduce the effects of the problem. If the contour is completely parallel to the epipolar line, epipolar geometry will have no effect in this case and the interpolation using disparity vectors of the seed points will have the major effect in the estimation of disparity of the contour point.

Another serious problem of contour matching process is to use pre-matched corners if false matches exist. An incorrectly matched corner point mislead the contour matching process on a contour which has the seed point corresponds to that corner. The best case in this situation is that the approximate disparity used for the seed point points an empty region. Therefore the process will end immediately. If the mismatched corner point is on a contour, the matching process may continue until the epipolar line passes too far. In this case, there may be several false matches.

Edges of two different objects may also coincide with each other. At the intersection of the edges, a corner may be extracted. Such a corner is not a real corner. From a different view, the intersection point slightly moves according to the depth of edge points as shown in Figure 4.8. Therefore, the extracted corner point does not belong to the same point of the objects. Using the matching of such a corner for the seed point may cause some false matches.

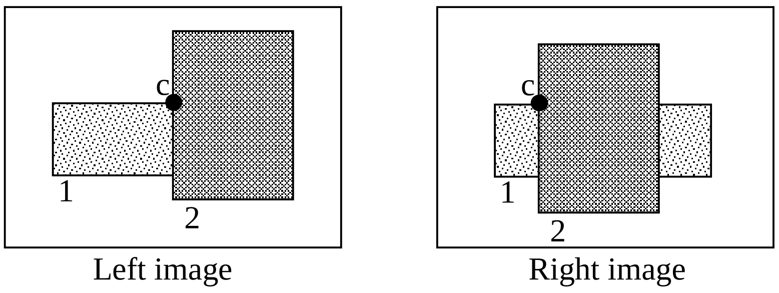


Figure 4.8 – The intersection of the edges of two objects

Using the edge points without any segmentation process as the contour points may also raise several problems causing false matches, if:

- Some edges extracted from the first image are not available in the second edge image.
- Edges of an object coincide with the edges of another object and there is 8-connection between these edge points.

To reduce the number of false matches, we need to stop matching process according to the some certain conditions. Since the process of contour matching using two seed points is due to a linear interpolation, all matched points between the seed points should be cancelled if the process is stopped without completing the whole contour segment or a point on the contour cannot be matched. Assume that we start the contour matching process from the top-left corner of the first object in Figure 4.8 along its horizontal edge. Let the first image be the left image and the second image be the right image. At some point on the edge, we hit the edge of the second object in the right image although the edge continues horizontally in the left image. In this case, the closest edge point to match will be the “c” point. As we continue on the edge in the left image, the matched point in the right image will always be the same while the epipolar line is close to the “c” point. Hence, the same edge point should not be matched several times ($parameter_1$). If such a situation is encountered, the process should stop immediately.

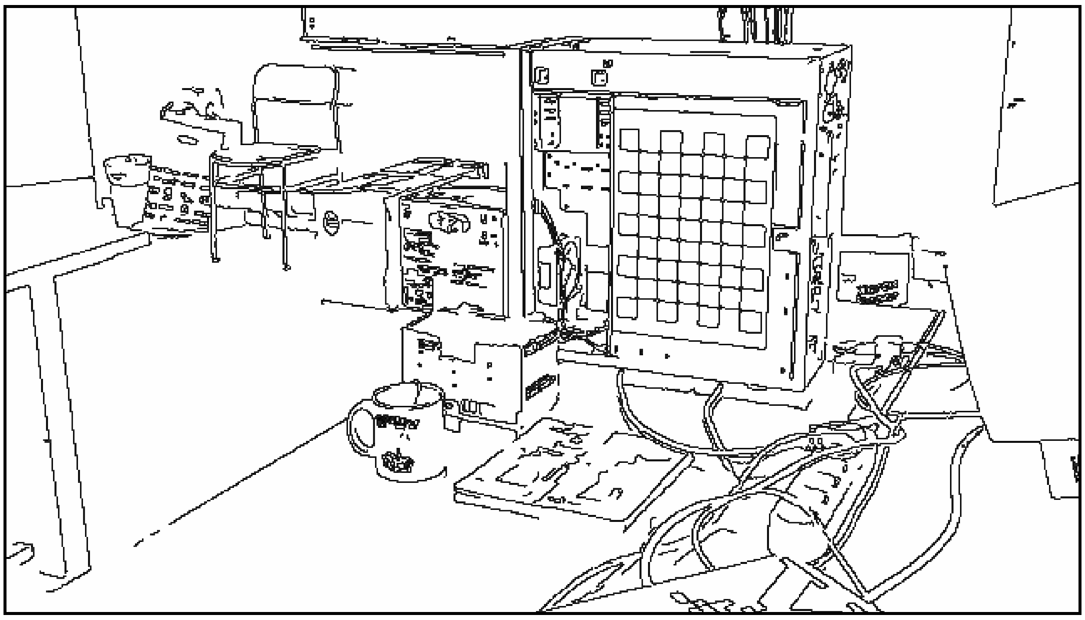
When the contour matching method using one seed point is being applied, we have another condition to continue or stop if we hit another seed point during matching process. The seed point should have the approximately same disparity as it is approximated before. If the estimated disparity of the seed point by the contour matching process is too different than its approximate disparity, it means that the process is producing false matches and it should stop. The maximum allowed magnitude of the difference of the two mentioned vectors is the $parameter_2$.

The contour matching method using two seed points usually matches less points since contour segments between two seed points may not be available for the whole contour. In order to have more matched points, one seed point version of the method may be applied after applying the two seed point version.

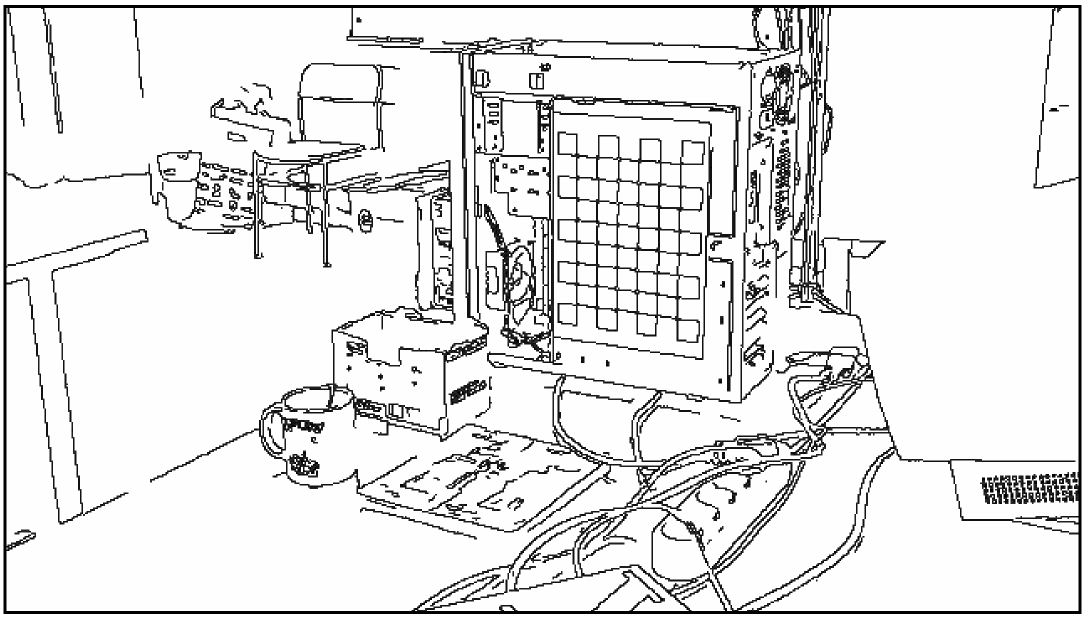
Figure 4.9 shows the left and right edge images extracted from the images given in Figure 4.2. The Gaussian smooth was applied to the source images in Figure 4.2 with the standard deviation $\sigma = 2$. The edge images were found by the Canny edge detector that was applied to the smoothed images. A 3x3 Sobel operator was used for the Canny algorithm. The high and low thresholds were chosen as 50 and 30 respectively. The Canny edge detector extracted 64899 edge points in left image. The edge points were used as the contour points for the contour matching process. A 3x3 window was used for the seed point determination. 337 matched corner points out of 393 were chosen as seed points. Edge points extracted from the left image were used for contour matching. The last found fundamental matrix was used for the epipolar geometry. In the first experiment, contour matching was applied starting from each seed point. An edge point in the right image was allowed to be matched at most 5 times ($parameter_1$ = 5). The maximum allowed magnitude of the difference of the approximate and estimated disparity vectors was chosen as 2 pixel length ($parameter_2$ = 2). The maximum epipolar line distance was chosen as $\omega = 3$. The size of the contour searching window W was decided to be 7x7. As a result, 25564 edge points out of 64899 were matched after all seed points had been used.

In the second experiment, contour matching process was applied in two phases. In the first phase, contours were matched between two seed points for each possible seed pair. The parameters are; $\omega = 2$, the size of W is 5x5, $parameter_1$ = 7 and there is no need for $parameter_2$. 12236 points were matched at the end of the first phase. In the second phase, the rest of unmatched contour pairs were tried to be matched by one seed point version of the method using the matched contour points as seed points. The parameters of the second phase were chosen as the same as the parameters used in the first experiment. As a result, a total of 26264 matched edge points were found at the end of the second phase. All matched points in both experiments were then used for stereo triangulation explained in Chapter 2. 3D locations of the points were computed using the equation (2.47) and the results of both experiments were compared. As a result, more incorrectly matched contour points are available in the result of first experiment with respect to the second experiment.

Figure 4.10 shows a specific region taken from the left and right images that the edge points have been superimposed. In Figure 4.10(a), a circle is centered at a point which is on an edge. The corresponding epipolar line for this point is shown in Figure 4.10(b) which is slightly parallel to the edge. The region where the line intersects the edge and the line passes closely through the edge is too wide to determine the matching point accurately. The 3D reconstruction results of the experiments are shown in Figure 4.11. The results of the first experiments show that matching process has been started individually from left and right tips of the same edge segment which the point is located on. As the matching process going from the neighbor to neighbor, the error propagates and then finally the method completely misses the edge so that the edge segment seems two separate edge segments in Figure 4.11(a). But in the second experiment, the edge segment perfectly fits into the model and the same segment has been matched successfully as shown in the Figure 4.11(b). In the second experiment, the number of false matches also has reduced and some more edge points have been able to be matched by the methods applied in two phases.

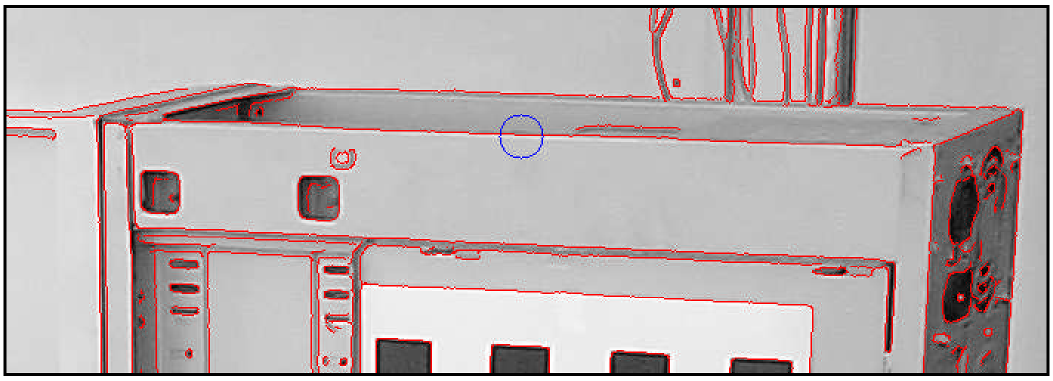


(a) Left image

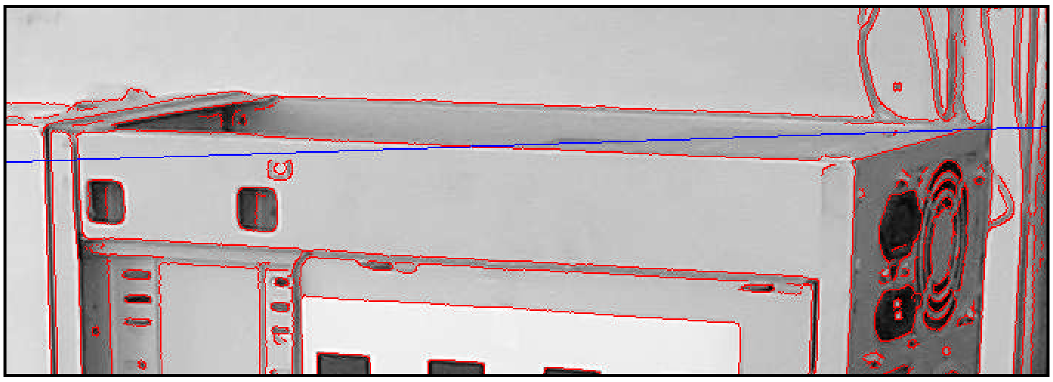


(a) Right image

Figure 4.9 – Edge images

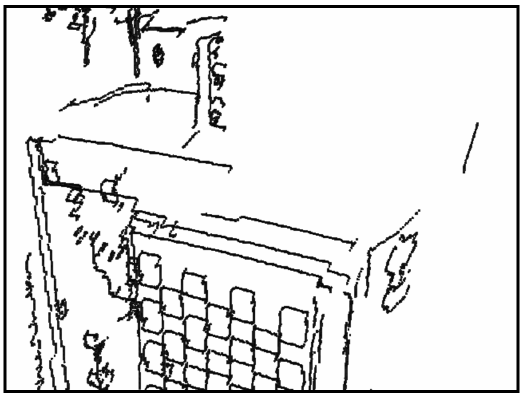


(a) A point on an edge in left image

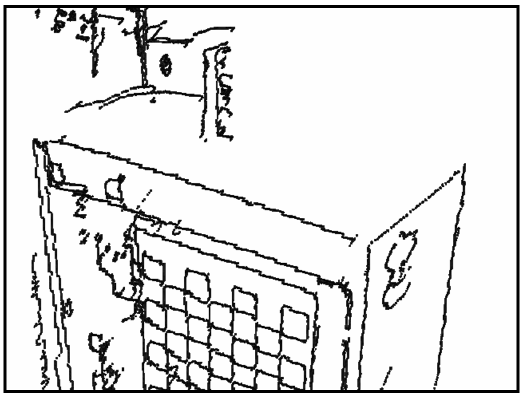


(b) Corresponding epipolar line in right image

Figure 4.10 – A point on an edge and its corresponding epipolar line



(a) First experiment



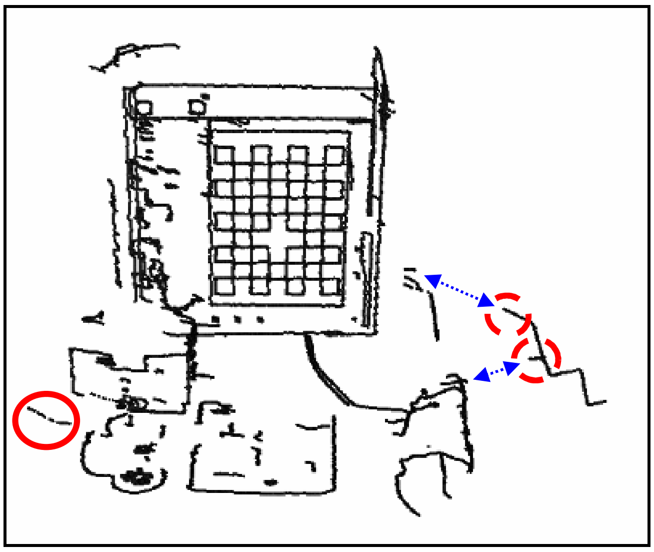
(b) Second experiment

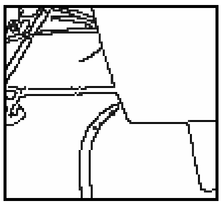
Figure 4.11 – 3D reconstruction results of the experiments

CHAPTER 5

CONCLUSION

Reconstruction process has been applied to sample calibrated stereo images. Test results show that reliability of contour matching process directly depends on corner matches since they are used as seed points. Incorrectly matched corners lead incorrectly matched contours having quite different depth value from the depth that they should have. Some corners may be matched to a similar false corner incorrectly in spite of used constraints. Using false seed points produce some false contour matches due to shape similarity of contours. If non-similar contours are matched incorrectly, epipolar geometry constraint generally stops the process in a few iteration. But, if incorrectly matched contours have a similar shape, number of false matches increases and even whole contours of the object may be matched incorrectly and therefore, depth of contours of the object may be found incorrectly. Since contour points are located on pixel positions in a binary image, their locations in 3D have the error such that the contour follows a distorted path that is not smooth, but the error is insignificant and shape of the object remains consistent for the objects that are close to the cameras. Shapes of the far objects are corrupted since the angle between the rays from the cameras to the point is too narrow. Using edges as contours also leads some problems. An important flaw occurs where the edges of two different objects are connected. Since the connected edges are expected to have the similar disparity, the reconstructed edge continues connected as if it belongs to the object that is the wrong object actually (see Figure 5.1). The method finds false matches until the epipolar constraint prevents it. In some cases, reconstructed edge also tends to be connected between contours of these different two objects (see Figure 5.2). Generally, disparity changes rapidly in this situation and a number of false matches are found depending on the adjusted thresholds.

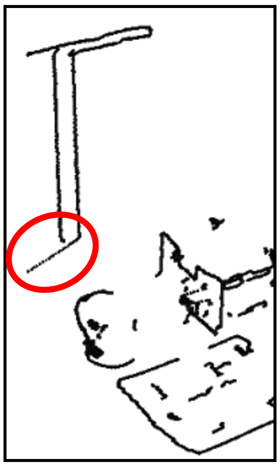


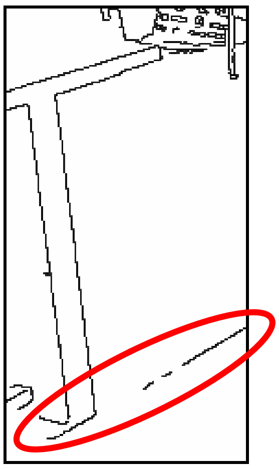


(a) (b)

Figure 5.1 – Reconstruction results and false matches (1)

Figure 5.1(a) shows the reconstruction result of the test images shown in Figure 4.2 that has been reconstructed using the method mentioned in previous chapter, applied in second experiment. Figure 5.1(b) shows a region from the edge image containing connected edges that belong to different objects and lead false matches. In Figure 5.1(a), dashed circles and arrows indicate the effect of the connection between edges. The circled points on the left are also false matches caused by a false seed point.





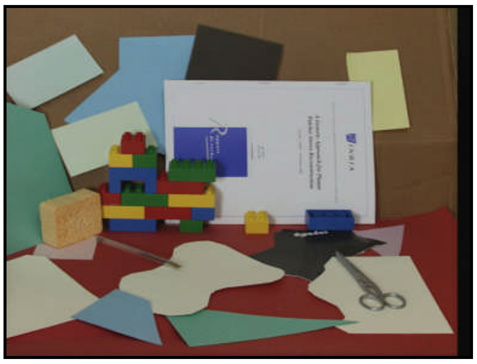
(a) (b)

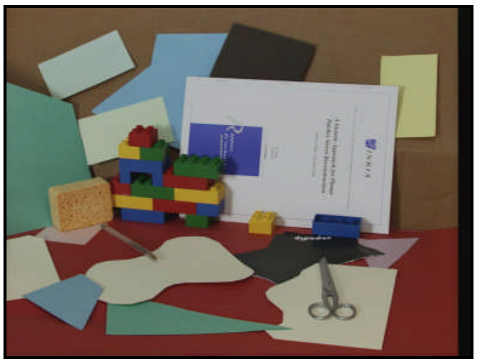
Figure 5.2 – Reconstruction results and false matches (2)

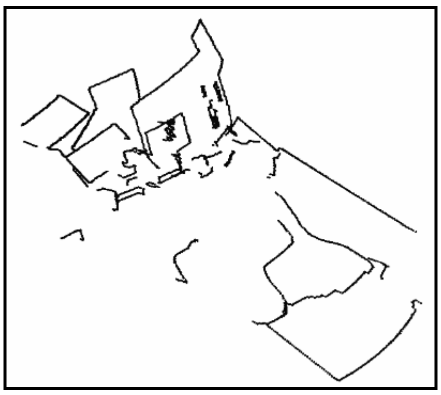
In Figure 5.2, a similar edge connection problem is shown. A tip of reconstructed edge is connected to the far object whereas the other tip is connected to the near object.

5.1. Test Results

Test images and reconstruction results from different views are given below for the images taken from [14] and our test images.


(a) Left image


(b) Right image


(c)

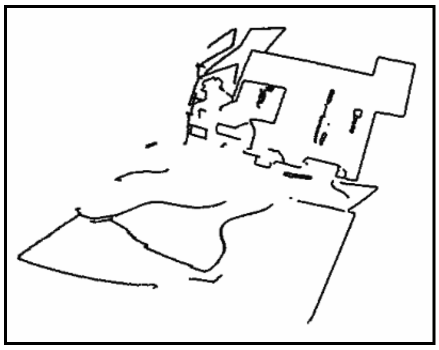
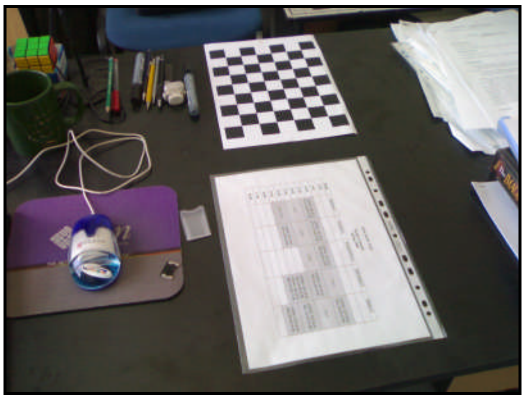
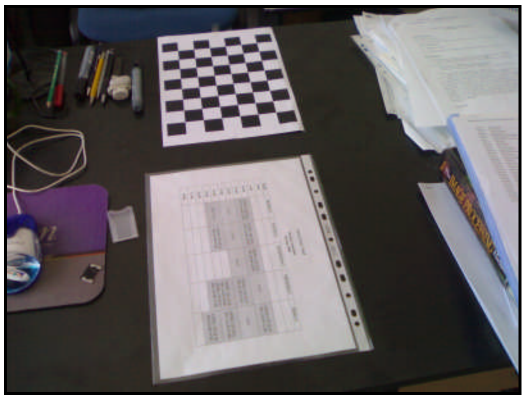
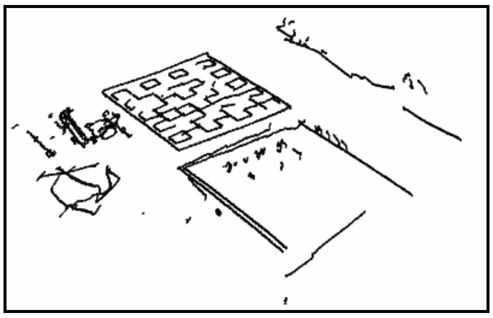
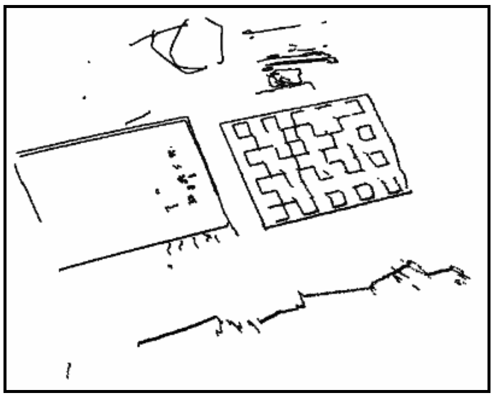

(d)

Figure 5.3 – Reconstruction results and test images


(a) Left image


(b) Right image


(c)


(d)

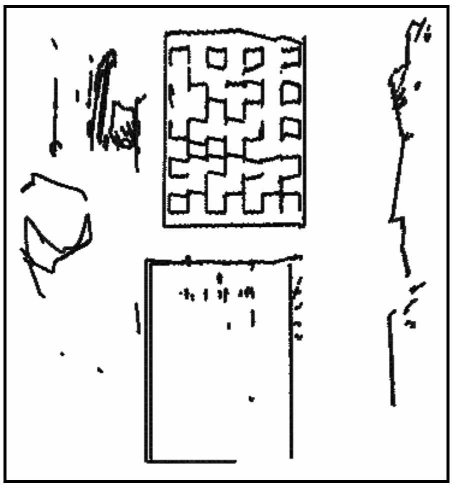
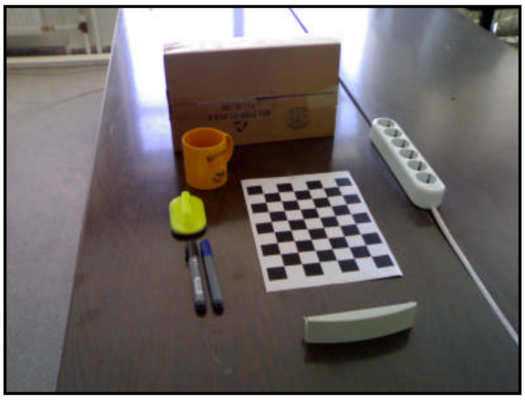
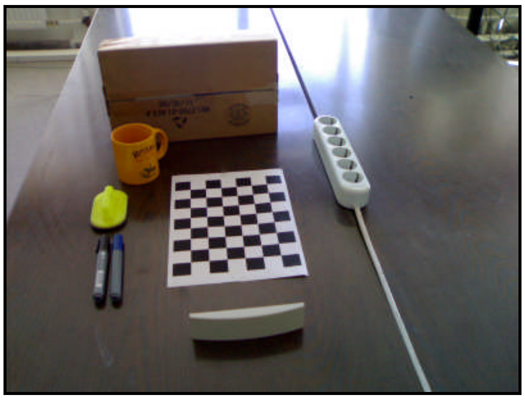
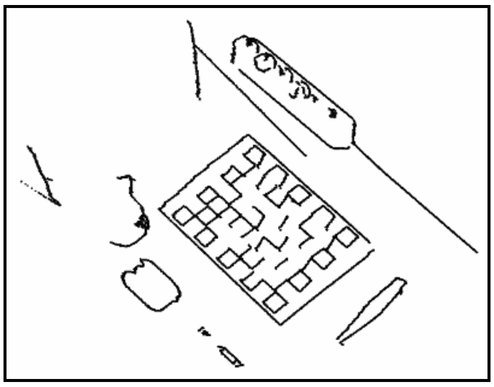
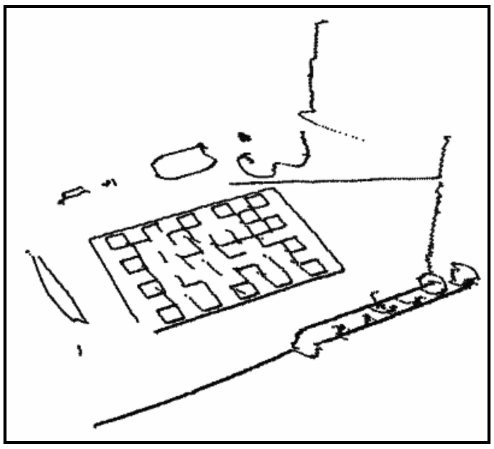

(e)

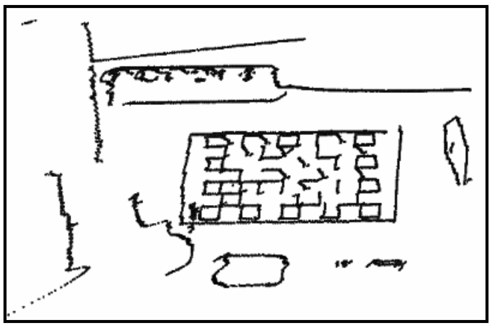
Figure 5.4 – Reconstruction results and test images (2)


(a) Left image


(b) Right image


(c)


(d)


(e)

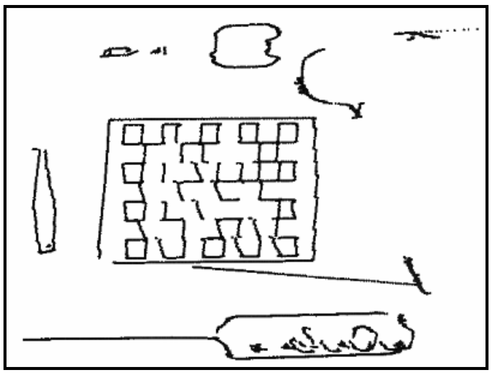

(f)

Figure 5.5 – Reconstruction results and test images (3)

5.2. Future Work

Reconstruction of near objects is generally more reliable and their shapes are preserved. Average depth of far points may be useful to compare the depth of objects but their shapes are not useful. Figure 5.6 shows the distribution of reconstructed points with respect to the depth and reconstruction error for the test images shown in Figure 4.2.

Using depth vs. error graph, we can classify the point groups and measure the average error for each group. According to the average errors, groups can be neglected when the shape of objects is in question. Besides, points contained in each group that have relatively high error can also be cancelled in order to have more reliable reconstructed point groups. Each group can also be analyzed individually to be divided into sub-classes. Classifying process may be useful for object tracking and segmentation.

Another development can be made to reconstruct surfaces that belong to the objects in the scene. Region growing methods may be applied in the segmentation process using available 3D data. Then, reconstructed points that belong to each region can be used to approximate the location of surface points contained in the region.

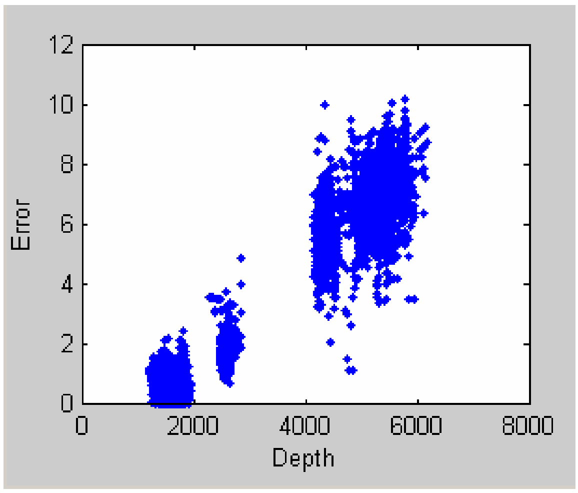


Figure 5.6 – Depth vs. Error graph

REFERENCES

[1] http://www.wikipedia.org/

[2] **ZHAO, B., MANDAL, C., VEMURI, B.C., AGGARWAL, J.K.** (2000), 3D Shape Reconstruction from Multiple Views, *Handbook of Video and Image Processing*, A. Bovik, Academic Press, The University of Texas, Austin, Texas 243-257.

[3] Intel Corporation (2001), *Open Source Computer Vision Library Reference Manual*.

[4] **CARLSSON, S.** (2005), *Point Cloud Reconstruction from Image Sequences – Calibrated Versus Uncalibrated*, Master's Thesis in Computing Science, Umea University, Umea.

[5] **GOLUB, G., LOAN, C.V.** (1983), *Matrix Computations*, John Hopkins University Press, Baltimore.

[6] **HARTLEY, R.I.** (1997) In Defense of the Eight-Point Algorithm, *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 580-593. Vol. 19(6).

[7] **CHOJNACKI, W., BROOKS, M.J.** (2003) Revisiting Hartley's Normalized Eight-point Algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1172-1177. Vol. 25(9).

[8] http://www.cim.mcgill.ca/~dparks/CornerDetector/mainMoravec.htm

[9] **HARRIS, C., STEPHENS, M.J.** (1988) A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*, 147-152.

[10] **DERPANIS, K.G.** (2004) The Harris Corner Detector, Technical Report, York University, New York.

[11] **GONZALEZ, R.C., WOODS, R.E.** (2002), *Digital Image Processing*, Prentice Hall, New Jersey.

[12] **CANNY, J.** (1986) A Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 679-698. Vol. 8(6).

[13] **HAN, J.H., PARK, J.S.** (2000) Contour Matching Using Epipolar Geometry, *IEEE Trans. on Pattern Anal, Mach. Intell.*, 358-370. Vol. 22(4).

[14] http://perso.lcpc.fr/tarel.jean-philippe/syntim/paires.html

[15] **ALVAREZ, L., DERICHE, R., SANCHEZ, J., WEICKERT, J.** (2000), *Dense Disparity Map Estimation Respecting Image Discontinuities: A PDE and Scale-Space Based Approach*, Technical Report RR-3874, INRIA, France.

[16] **BIRCHFIELD, S., TOMASI, C.** (1999) Depth Discontinuities by Pixel-to-Pixel Stereo, *International Journal of Computer Vision*, 269-293, Vol. 35(3).

[17] **FISCHLER, M.A., BOLLES, R.C.** (1981) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Comm. of the ACM*, 381-395. Vol. 24.

[18] **FORSYTH, D.A., PONCE J.** (2003), *Computer Vision – A Modern Approach*, Prentice Hall, New Jersey.

[19] **HARTLEY, R., ZISSERMAN, A.** (2003), *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge.

[20] **HARTLEY, R.I., STURM, P.** (1997) Triangulation, *Computer Vision and Image Understanding*, 146-157. Vol. 68(2).

[21] **KARLSTROEM, A., TAKASE, F.K., FURUKAWA, C.M., MARUYAMA, N.** (2006) Position Estimation and Error Quantification – An Epipolar Geometry Based Approach.

[22] **KIM, H., SOHN, K.** (2005) 3D Reconstruction from Stereo Images for Interaction Between Real and Virtual Objects, *SP:IC*, 61-75. Vol. 20(1).

[23] **MITSUHASHI, S., HAMADA, N.** (2002) Improved Estimation Method of Disparity by Its Recursive Use, *International Symposium on Information Theory and Its Applications, Xi'an PRC*, 391-394.

[24] **MORENCY, L.P., RAHIMI, A., DARELL, T.** (2002) Fast 3D Model Acquisition from Stereo Images, *3DPVT 2002*, 172-176.

[25] **SILVA, L.C., PETRAGLIA, A., PETRAGLIA, M.R.** (2003) Stereo Vision System for Real Time Inspection and 3D Reconstruction, *Industrial Electronics ISIE '03, IEEE International Symposium*, 607-611. Vol. 1.

[26] **TORR, P.H.S., MURRAY, D.W.** (1997) The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix, *International Journal of Computer Vision*, 271-300. Vol. 24.

[27] **YUAN, M., XIE, M., YIN, X.** (2002) Robust Cooperative Strategy for Contour Matching Using Epipolar Geometry, *Asian Conference on Computer Vision*.

[28] **ZHANG, Z., DERICHE, R., FAUGERAS, O., LUONG Q.T.** (1995) A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry, *Artificial Intelligence. J.*, 87-119.Vol. 78.

[29] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm

[30] http://www.pages.drexel.edu/~weg22/can_tut.html