2013-12-05

Indoor Navigation based on Fiducial Markers of Opportunity

Lakhani, Mazhar Ali

UNIVERSITY OF CALGARY

Indoor Navigation based on Fiducial Markers of Opportunity

by

Mazhar Ali Lakhani

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

NOVEMBER, 2013

Abstract

Indoor navigation has gained significant importance in the last decade due to its applications in the location based services industry. The outdoor navigation technology of GNSS by itself does not offer good solutions due to presence of multipath and low SNR in indoor environments. Complimentary sensors like IMU, magnetometers, cameras, etc., have their own set of problems. A camera however, provides a wealth of information for position estimation.

The main aim of this thesis is to analyse camera based navigation systems using stationary figures of opportunity. Unlike traditional navigation systems that utilise coded markers, this thesis performs navigation based on planar stationary figures that are inherently present in the environment, hence the more structure the more information. The novelty of this thesis lies in its usage of Hough transforms to do the same. Initial experimental results demonstrate that an average webcam quality camera provides enough information to extract reasonably good navigation coordinates under specific conditions.

Acknowledgements

To my supervisor, Dr. John Nielsen and my co-supervisor, Dr. Gérard Lachapelle, this thesis could not exist without your guidance and encouragement. Thank you for your gracious support and insight and for providing me with this opportunity to explore my interests in the field of navigation.

I would also like to thank all the wonderful and diverse members of Position Location And Navigation (PLAN) Group especially my team members Yuqi Lee and Neha Dawar. I would also like to thank Piyush Rawat for assisting me in writing my thesis. To the administrative staff at the Electrical Engineering Department, thank you for all your hard work behind the scenes and making my research possible.

To my parents, thank you for instilling in me a love for engineering and natural sciences. Thank your also for your continued support and encouragement through the many years that led up to the completion of my thesis.

Table of contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations

AGPS	Assisted Global Positioning System
CML	Concurrent Mapping and Localization
CV	Computer Vision
FM	Fiducial Markers
FMO	Fiducial Marker of Opportunity
FOV	Field of View
FP	Feature points
GFTT	Good Features to Track
GHT	Generalised Hough transforms
GIS	Geographical Information System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPRS	General Packet Radio Service
HND	Handheld Navigation Device
HT	Hough transforms
I.I.D	Independent and Identically Distributed
IMU	Inertial Measurement Unit
LBS	Location Based Services

MEMS	Micro Electro-Mechanical Systems
PT	Perspective Transformation
RANSAC	Random Sample Consensus
ROI	Region of Interest
RSSI	Received Signal Strength Indication
SLAM	Simultaneous Location and Mapping
SNR	Signal to Noise Ratio
TTFF	Time to First Fix
UWB	Ultra wide band
VLC	Visible Light Communication
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 THE PROBLEM

Navigation and positioning occupy a very important role in our lives today. A number of commercial, industrial and security applications require the precise location of the user, the map of the surroundings and the navigation trajectory. The applications of location based services (LBS) range from aiding navigation in malls and hospitals all the way to audio tours and entertainment, to name a few. There has been a lot of ongoing research in the area of determining the location coordinates outdoors using Global Navigation Satellite Systems (GNSS) like GPS, GLONASS and Galileo. While determining the location coordinates is fairly possible outdoors, the job becomes considerably complicated in the indoors due to multipath scattering of the signal in the indoor environment. Also in order to make practical use of indoor navigation systems, the position accuracy needed is of the order of 1 to 3 m. However, the accuracy provided by most state-of-the-art satellite based navigation systems in indoor environments range from 20 to 30 m when available at all [1]. In typical multipath environments of indoor and urban areas, meeting this requirement based on generic wireless and GNSS signals is generally not feasible. Furthermore, satellite sourced GNSS signals are usually attenuated such that the SNR is too low for extracting reliable observables [2-3].

If a GPS receiver is new or unused for long periods of time or transported to new locations while it was in off state, it will usually take a long time to acquire data sets. If the GPS receiver cannot get a clear view of the sky or if the data transmission is constantly interrupted, the transfer time for this data set may take significantly longer, or worse, be unable to perform signal acquisition, if the GPS receiver is in a very weak signal environment such as indoors and urban canyons. This time frame in navigation world is usually termed as Time to First Fix (TTFF). It can be described as the time and processing required by a GPS device to acquire enough usable satellite signals and data to provide accurate navigation.

According to a research, people spend 90% of their time indoors [4]. In such a scenario, the most common aiding device used to assist in navigation is a cellular phone. One recommended solution to enable indoor GPS navigation using a cellular phone is by using Assisted GPS (AGPS). AGPS gets approximate location information from server/host such that the search space can be reduced. It not only uses mobile technologies like GPRS (General packet radio service), but also often uses the service provider's network information. AGPS caters to the glitches rendered by traditional GPS by enabling the GPS module to perform faster acquisition, perform faster TTFF (time to first fix), and obtain faster positioning information under weak signal environments. Moreover the processing to be performed by the GPS phone is reduced by diverting most of the work to the assistance server. While this technology offers quite accurate location coordinates, the frequent use of AGPS to perform location fixes drains the cellular phone of battery power. Another problem hindering the use of this technology is that AGPS is not universally available indoors [5]. Hence there exists a gap in research with respect to indoor positioning and indoor navigation.

Figure 1.1: Difference in accuracy requirements between indoor and outdoor environments [6]

Figure 1.1 illustrates the difference in the accuracy requirement between a global positioning system and an indoor positioning system (IPS). While the accuracy requirement for a GPS can

range in a few tens of meters as shown in the figure, an IPS on the other hand needs to attain a few meters of accuracy in order to be functionally useful.

1.2 DIFFERENT APPROACHES TO LOCATION ESTIMATION

To fill this gap in indoor environments, research has been done to determine the indoor location using different techniques like location fingerprinting (scene analysis), triangulation, trilateration, hyperbolic lateration, proximity and dead reckoning. [7-11]. This paved the way for usage of various wireless technologies like Received Signal Strength Indication (RSSI), Ultra wide band (UWB), Radio Frequency Identification (RFID), Bluetooth, WLAN Based systems, Visible Light Communication (VLC), etc., in location estimation.

Typically the motivation behind using these technologies is to integrate them with the predominant outdoor GNSS technologies to give a seamless Indoor-Outdoor positioning system. [12], for instance talks about a method describing the fusion of Wi-Fi positioning technology and GPS satellite system to provide seamless positioning. Similarly, [13] discusses different methods for seamless indoor-outdoor positioning with GPS as the predominant outdoor technology and sensor technologies like Inertial Measurement Units (IMU) and micro electromechanical systems (MEMS) to aid during the loss of GPS signals.

While the above mentioned technologies offer real time solutions to positioning problems, sometimes these techniques introduce additional hardware costs. This leads to an increase in the overall infrastructure cost. Hence, there is a need for more flexible solutions that could potentially reduce this cost.

1.3 VISION SENSORS

In the scenario described above, an ideal positioning technology would be one that can cater to the following requirements

1. accuracy and reliability
2. security
3. low sensor cost
4. low integration cost

Computer Vision (CV) techniques have proven to be a good solution to the aforementioned specifications and offer a number of advantages as listed below.

Firstly, as it will be demonstrated in this thesis, CV observables provide high accuracy, especially over short-range trajectories. Also, sensors like IMU exhibit a dynamic drift in their measurements, because of which they need to be continuously calibrated. CV sensors on the other hand, are immune to such a drift, i.e. their performance remains stable with time. Secondly, CV sensors are highly secure, as the CV observables are immune to spoofing or signal jamming. Finally the overhead of additional hardware and infrastructure cost is low as CV observables can be sourced from an inexpensive webcam quality camera which is a standard component in most smartphones today. Hence considering all the above mentioned reasons, visual sensors have been chosen as the predominant sensor technology in this research along with coded fiducial markers and fiducial markers of opportunity.

1.4 CODED FIDUCIAL MARKER (FM)

A coded fiducial marker or fiducial template is defined as any object or template that is visible in the field of view (FOV) of the camera such that it can be used as a point of reference or measure. In general, fiducial markers (FM) have been extensively used for localising in both indoor and outdoor environments [14]. Such markers are typically comprised of known patterns and mounted in locations known to the handheld navigation device (HND). For example [15] presents a method for assigning markers to locations to enable navigation by observing short sequences of these markers, while [16] illustrates a method of encrypted patterns where the 3D position information is extracted by decoding the patterns.

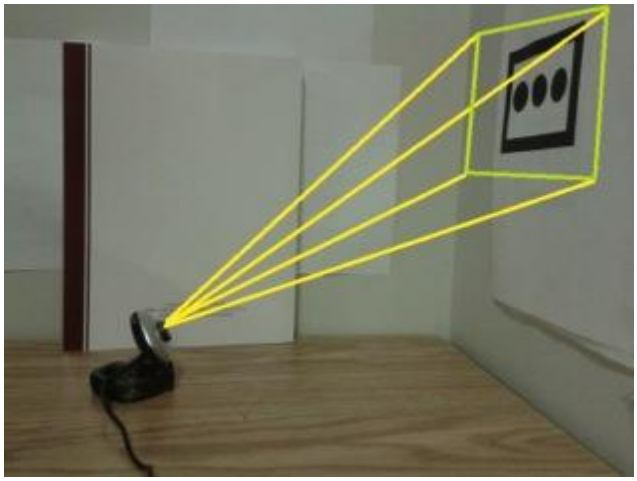


Figure 1.2: Agama V-1325R camera observing a stationary fiducial marker on the wall

Figure 1.2 depicts a setup where an Agama V-1325R webcam quality camera is observing a stationary marker on the wall. The yellow lines originating from the camera center and stretching across the marker form the FOV of the camera, which is shown in green.

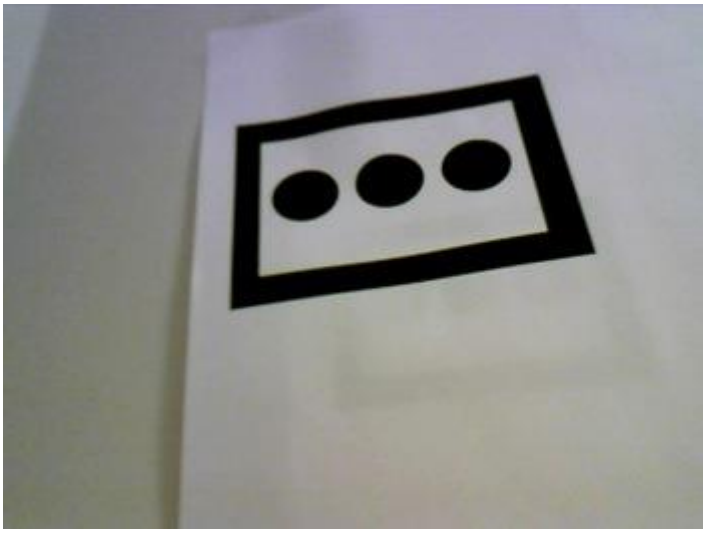


Figure 1.3: Coded fiducial marker as observed by the camera

Figure 1.3 shows of the camera observable showing the fiducial marker as observed in the camera's FOV. Figure 1.4 shows the true geometrical shape of the fiducial marker.

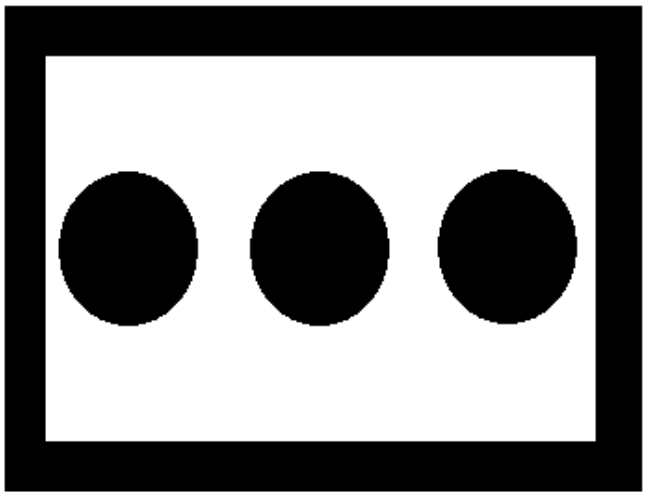


Figure 1.4: True Geometry of the fiducial marker

A handheld navigation device (HND) in the current context is any device capable of performing navigation in indoor or outdoor environments. In this thesis, this is equivalent to a smart phone that has a modest webcam quality camera able to provide camera observables. The integration of this camera based indoor navigation system with the predominant outdoor GNSS is possible since most modern day smart phones are embedded with a standard GPS.

By observing figure 1.3 and figure 1.4, it can be deduced that the camera observables undergo a spatial transformation based on the camera's orientation and its movement. This transformation in CV literature is called a Perspective Transformation (PT). Now if an inverse perspective transformation is applied to the camera observable of figure 1.3, the corresponding orientation and movement of the camera can be extracted. This is the central theme used in this thesis. More information on Perspective Transformation is given in the subsequent chapters.

1.5 FIDUCIAL MARKER OF OPPORTUNITY (FMO)

FM systems rely on the use of FM in the camera's FOV to perform navigation. This imposes a constraint on the system that a FM should always be in the FOV of the camera. This also enforces the indoor environment to be filled up with a large number of FM, which is visually not very appealing. This thesis aims to reduce this dependency. The application of the FM is extended in this thesis from known FM patterns and generalized such that arbitrary fiducial markers of opportunity (FMO) in unknown locations can be used for navigation. These FMO can

consist of random objects that occur frequently in an indoor environment such as doors and windows.

A known FM can be set as the origin of the world coordinate axes. If the camera initially observes this known FM as the starting point of the trajectory, then the remaining trajectory can be calibrated with any arbitrary FMOs that appear in the FOV of the camera. Since a typical indoor environment already has a sufficient number of such random FMOs, they can be employed to perform navigation. The only necessary assumption made is that the FMO is stationary relative to the world coordinate axes and that the HND is moving relative to this world coordinate axes. The FMO may consist of simple patterns known only to be rectangular and stationary. It is important to note that rectangles are used as a predominant example of FMO since they occur more frequently than other geometrical figures in a typical indoor environment. However this is not a generalised rule and FMO can consist of any arbitrary shape of generally unknown dimensions. Figure 1.5 illustrates an example in which an FMO (a window) is detected by Hough transforms. More details about Hough transforms is given in subsequent chapters.

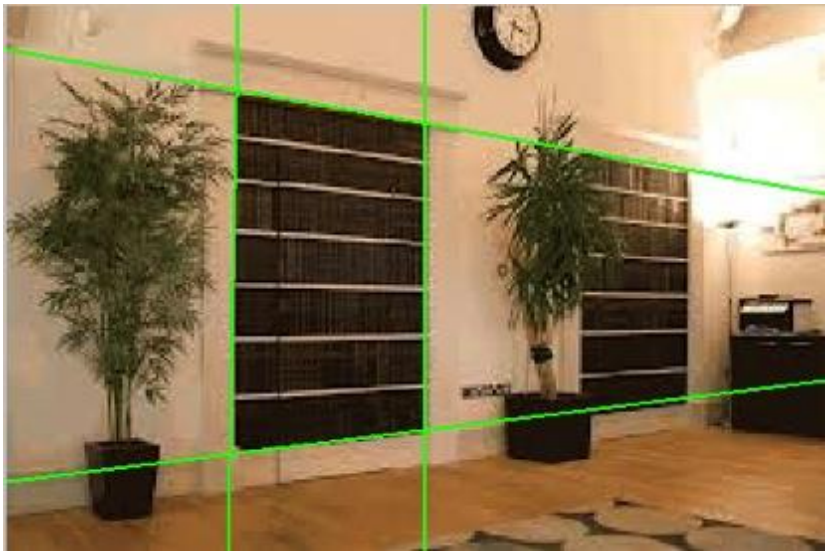


Figure 1.5: A fiducial marker of opportunity detected by Hough transform

In this research, a range of FMs and FMOs are considered from which the perspective transformation can be estimated. The rotation and translation vectors which together define the six (Degrees of Freedom) DOF motion of the HND can then be extracted from the perspective transform. These range from simple arbitrary rectangle shapes to mounted coded template

markers. The novelty of this research is using a marker as an opportunistic fiducial that is only assumed to be stationary as opposed to being in a known location as assumed in conventional FM algorithms. Another novelty lies in the fact that Hough transforms are used for the same.

The specific goals of this research are as follows:

1. Extraction of perspective transformation by observing FMs and thereby estimating the trajectory of the HND. Even though trajectory estimation from FMOs alone will yield the required trajectory, it does not permit to reference the trajectory against a map of the environment. It is for this reason that FM systems are introduced and used in this thesis.
2. Evaluating the robustness and accuracy of the FM system for trajectory estimation.
3. Trajectory estimation using a combination of FMO and FM in indoor environments.
4. Analysis of the camera trajectory estimated with multiple FMOs. This is required since the trajectory needs to be calibrated from one FMO to another.
5. Extraction and analysis of FMO using Hough transforms.

The contributions of this thesis are as follows

The FMO can consist of feature points to lines to arbitrary shapes seen in the camera's FOV. As shown in chapter 4, Hough transforms proves to be a very important tool for extraction, analysis and processing of FMO. All of these combined together led to the publication of two conference papers which are listed as follows:

1. A conference paper titled "*Computer Vision Navigation based on Fiducial Markers of Opportunity*" was published in the ION (Institute of Navigation) PNT conference held in July 2013 from the research conducted in this thesis. This paper emphasised that unlike traditional FM navigation schemes, an FM is not always required to be in the FOV of the camera and navigation can be performed using FMOs that are already available in the environment. [17]
2. Another conference paper titled "*Indoor Navigation Based on Fiducial Markers of Opportunity*" was presented at the 2013 International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV) held in July 2013 emphasizing the use of SLAM techniques to extract navigation estimates for the FM-FMO system. [18]

1.6 THESIS OUTLINE

- Chapter 2 gives the necessary background needed to understand computer vision based indoor navigation system. It also gives the necessary theoretical explanation needed for 6 DOF egomotion (3D motion estimation of a camera in an environment). Camera calibration is also performed by means of 2D planar calibration pattern. RANSAC outlier rejection algorithm which is employed during experimental verification in chapter 4 is also explained using an example. The chapter concludes by a section describing FMO and line detection using Hough Transforms. This is later employed in chapter 4 for FMO processing.
- Chapter 3 gives the flowchart of the algorithm along with the explanation of each of the steps involved. The theory behind obtaining rotation angles from homography is also explained. The chapter also explains the similarities between the proposed algorithm and Simultaneous Localisation and Mapping (SLAM) techniques.
- The proposed algorithm outlined in Chapter 3 is validated with sets of independent experiments in Chapter 4. These experiments include known trajectory estimation through the use of a precision turntable to unknown trajectory estimation using back projection method for error estimation. Experimental results also demonstrate the process of a transition from FM to FMO. As already mentioned, rectangles are selected as the predominant geometrical figures corresponding to FMO in this thesis. Chapter 4 also discusses the processing required for FMO and extraction of rotation and translation for these FMO in the light of Hough transforms.
- In Chapter 5, the thesis contributions and suggestions for future work are presented.

  Theoretical and experimental results are presented in this thesis that demonstrate that any known or assumed attribute of an observed FM can provide information of practical significance in the context of the navigation objective. That is, the CV processing of the observed FMs will result in constraints that can be directly applied to the navigation estimate. Furthermore, as will be shown, the CV processing required can be accomplished in real time with a modest processor.

Chapter 2

Background

The last decade has seen a tremendous increase in the use of visual sensors across different industries. These industries include medicine, robotics, navigation, life sciences, entertainment and leisure, LBS, geographical information system (GIS), astrophotography and traffic surveillance, to name a few. Since this thesis aims to contribute to the CV based navigation space, one of the most fundamental problems in this space is to compute the three dimensional motion of the camera measured in the image plane. This in CV literature is termed as Egomotion. Egomotion can be defined as the process of determining the 3D motion of a camera within an environment using a sequence of images. The de facto standard of egomotion is done using feature detection to construct an optical flow from two image frames in a sequence generated from either single cameras or stereo cameras. An introduction of optical flow is given in Appendix A. Using stereo image pairs offers an advantage over a single camera system by providing additional depth and scale information. Even though optical flow is the most widely accepted method for egomotion estimation, there are many other algorithms available in the CV literature that have gained wide acceptance.

This chapter is dedicated to developing a thorough understanding of the basic building blocks of the proposed algorithm. These include feature points, thresholding, smoothing, edge detection, contour detection, perspective transforms, camera calibration, RANSAC and Hough transforms. All of these building blocks are employed in the proposed algorithm for FM and FMO based trajectory estimation.

2.1 Feature Point

In computer vision and image processing, the concept of feature is used to denote a piece of information which is relevant to solve the computational task related to a certain application. In other words, a feature can be defined as any local, meaningful and detectable part of an image

that can be used for processing. An important characteristic of a feature is that it has independent variation in two dimensions. The most common type of features in an image consists of corners, blobs and edges. Feature detection algorithms like Good Features to Track (GFTT) [19], Canny edge detector[20], Hough transform[21] etc. are all examples of different feature detection algorithms. Feature detection has been discussed in this chapter because even though is not directly employed in the proposed algorithm, it forms a fundamental building block in more complex algorithms like contour detection that is later employed in this thesis. Hence GFTT is described in the subsequent section. Canny edge detector and Sobel operator are also discussed in this chapter as they are quite frequently used in this thesis.

2.2 GOOD FEATURES TO TRACK

Images in general are expressed as 2D grids of intensity values at each pixel coordinate. The example given in figure 2.1c shows the same using histogram plot.

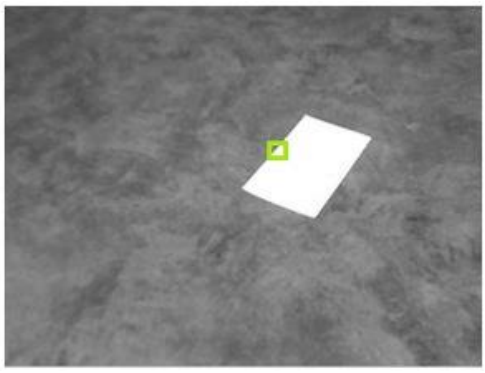


Figure 2.1a

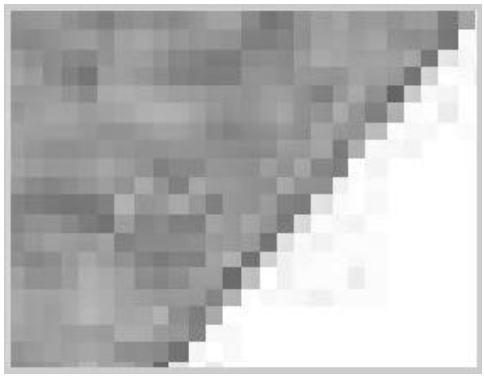


Figure 2.1b

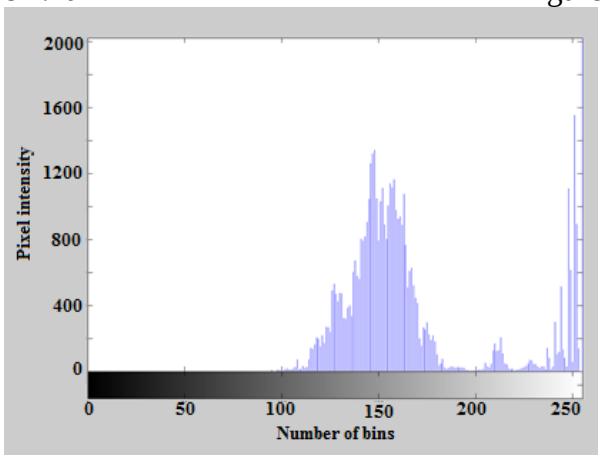


Figure 2.1c

Figure 2.1: Pixel intensity variation of the ROI in a scene.

Figure 2.1a illustrates a scenes with a region of interest (ROI) shown in green. In figure 2.1b, the ROI is magnified to observe the intensity values at each pixel. Figure 2.1c is the histogram representation of figure 2.1b which shows the variation in the intensity values for the aforementioned ROI along the edge. From figure 2.1b, it can be deduced that pixel intensities vary quite significantly especially along the edges.

No feature based system can work effectively if the features they work on cannot be appropriately identified. Features can more often be generalised as physical points in the real world appropriate for tracking from one frame to the next. This section is dedicated to explaining the concepts of feature points and its characteristics.

In [19], Shi and Tomasi used dissimilarity as a measure to find features. When tracking features in consecutive image frames, dissimilarity can be described as the feature's RMS residue between the first frame and the current frame. If there is a large growth in the dissimilarity of a feature, then the feature is abandoned. Since complete details of GFTT are not a necessary part of this thesis, it is out of scope. However a brief explanation of the same relating to a single image is given below.

Consider an image with intensity function $\mathrm{I}(\mathrm{x},\mathrm{y})$. Also consider the partial derivative of this intensity function in the X and Y directions as follows:

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x} \text{ and } I_y(x,y) = \frac{\partial I(x,y)}{\partial y}$$

Consider a sub region (window) W of the image about which a scan is undertaken. The values of $I_x(x,y)$ and $I_y(x,y)$ is found about this window W. The values of these partial derivatives can take one of the following forms:

1. $I_x(x,y)$ and $I_y(x,y)$ are both small in the window W: This indicates that $\mathrm{I}(\mathrm{x},\mathrm{y})$ is featureless in the window.

2. $|I_x(x,y)|$ and $|I_y(x,y)|$ are moderately large: This indicates the presence of a feature. However $I_x(x,y)$ and $I_y(x,y)$ being highly correlated indicates the presence of single dimensionality. This is similar to edge which is not very useful if corners are being searched.
3. $|I_x(x,y)|$ and $|I_y(x,y)|$ are sufficiently large and not correlated: This indicates the presence of a two dimensional feature which can be used for tracking.

The size of the window W has to be selected very carefully. A very large window would include more than one feature leading to ambiguity in feature detection. A very small window will lead to too few samples leading to incorrect partial derivatives. Hence the problem of detecting a good feature in the search window W is equivalent to determining the covariance of the random functions $I_x(x,y)$ and $I_y(x,y)$ in this window. This covariance can be numerically determined using the following equations:

$$I_{xx} = \mathrm{E}_W[I_x(i,j)^2] = \frac{1}{N_x N_y}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} I_x(i,j)^2 \tag{2.1}$$

$$I_{xy} = I_{yx} = \mathrm{E}_W\left[I_x(i,j)I_y(i,j)\right] = \frac{1}{N_x N_y}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} I_x(i,j)I_y(i,j) \tag{2.2}$$

$$I_{yy} = \mathrm{E}_W[I_y(i,j)^2] = \frac{1}{N_x N_y}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} I_y(i,j)^2 \tag{2.3}$$

The covariance of the intensity gradients of $I_x(x,y)$ and $I_y(x,y)$ is equivalent to the following Q matrix:

$$Q = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \tag{2.4}$$

The eigenvalues of this Q matrix determine the characteristics of the feature. If λ_1 and λ_2 denote the eigenvalues of Q such that $\lambda_1 \geq \lambda_2$, then the case of $\lambda_1 >> \lambda_2$ indicates a single dimension feature such as a line. Q being symmetric forces the eigenvalues to be real and positive. Two strong eigenvalues imply the presence of a two dimensional feature in the search window. Hence it is important to consider the second eigenvalue followed by comparison with a threshold.

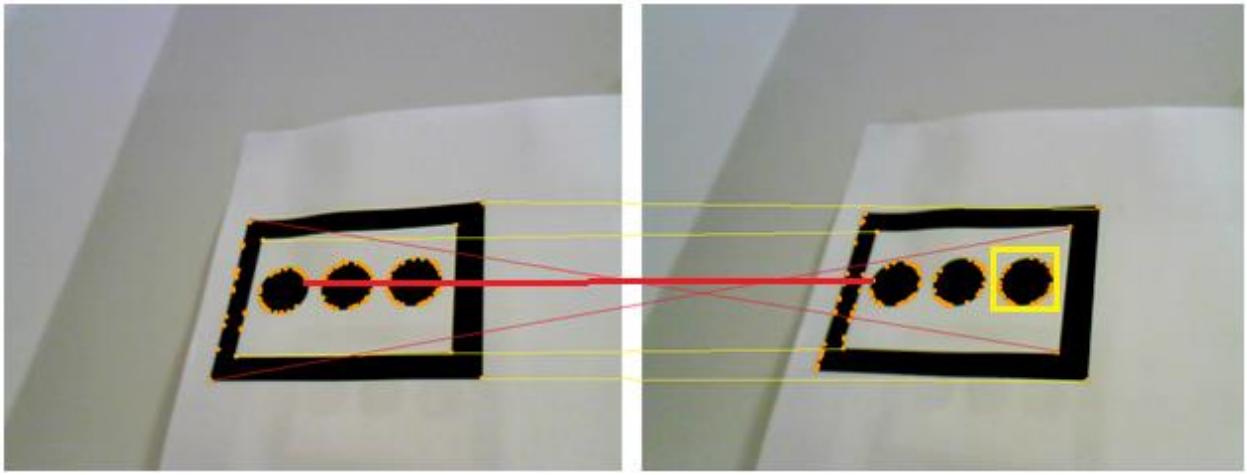


Figure 2.2a Figure 2.2b

Figure 2.2: GFTT algorithm results for a test marker in two consecutive frames

Figure 2.2 shows an example in which GFTT algorithm is used to identify features of the same test marker in two consecutive image frames of a video. The maximum number of features to be detected in each frame is set to 50. Ideally, only 8 features are required in each frame to track the marker in consecutive image frames. These features correspond to 4 corners in the first image frame and 4 corners in the next image frame. If too few features are selected, then the probability of one or more corners being missed is high. If too many corners are selected, then noise features are added to the image. Hence, the correspondence between these 8 required features cannot be correctly established since there is no way of determining which 8 points should be considered. This makes the tracking problem all the more difficult to solve. Hence a total of 50 features are selected in each image and a quality level of 0.01 was chosen, where quality level is defined such that the threshold $\lambda = \text{quality level} * \lambda_{max}$. λ_{max} is the largest eigen value through the image. The minimum distance between two features is fixed to 0.5 pixel units. The two images are taken

from a consecutive video frame and feature point extraction is performed. The detected points are shown in orange.

While performing perspective transform in images using feature point extraction techniques like GFTT, extraction of feature points alone is not sufficient. Another very important quantity to determine is the correspondence between these points. The lines drawn in thin yellow in figure 2.2 illustrate the correct correspondence. The line drawn in thin red color illustrates incorrect correspondence. If yellow lines alone are used, then correct estimates of perspective transform can be obtained. However estimating this correspondence correctly is itself a difficult problem. It is also important to note that even though GFTT is not directly employed in this thesis, feature detection algorithms such as these are the basic building blocks of determining contours in an image. These detected contours are ultimately employed in the proposed algorithm. A fundamental type of such contour can be seen around the circle in the yellow box in figure 2.2b which consists of sequence of such features. When the number of features to be detected is increased, these contours become all the more clear to observe. However this comes with the addition of more noise features as already mentioned previously in this section.

Features that are detected on a circle have an inherent problem of rotational ambiguity associated with them. There is no direct way of estimating if a circle has undergone rotation or not. Hence determination of the precise correspondence between features on a circle is difficult. This is depicted by the solid red line in the two images. However, if all of the detected points cumulatively account to a circular contour, then the displacement of the center still yields useful information in the form of translation between the two images.

2.3 PRE-PROCESSING

In most real time computer vision schemes, the object under consideration tends to have an almost equal apparent brightness across different frames. This is because the brightness of an object in a certain scene varies very slowly such that the difference in the brightness in consecutive frames is negligible. This is termed as brightness constancy.

Images are inherently noisy in nature. Image noise can correspond to anything that is unwanted in an image. Examples of this noise include sensor noise, finite precision, quantization issues etc. The lack of brightness constancy in consecutive image frames can also lead to the addition of

noise. Hence it is always a good idea to preprocess an image for noise reduction before applying any advanced image processing or computer vision algorithms to it. Some of these algorithms include smoothing, thresholding, edge detection and image differentiation. In this section, only those algorithms that are directly applicable to the proposed computer vision algorithms of this thesis will be discussed.

All the aforementioned forms of noise can be analysed by means of using appropriate noise models like impulsive noise (sometimes also called salt and pepper noise), shot noise, additive gaussian noise, quantization noise, multiplicative noise, anisotropic noise etc. The most commonly used model used to denote image noise is the additive gaussian noise model which is as follows:

$$I(i,j) = S(i,j) + n(i,j) \tag{2.5}$$

where 'n' is an Independent and Identically Distributed (i.i.d) random variable for all pixels. S(x,y) denotes a deterministic signal. 'n' is zero-mean Gaussian (normal distribution) and 'i' and 'j' denote pixel indices. In addition,

$$E(n) = 0 \tag{2.6}$$

$$Var(n) = \sigma^2 \tag{2.7}$$

$$E(n_p, n_q) = 0 \text{ (condition for independence)} \tag{2.8}$$

where n_p and n_q denote two discrete noise samples observed in two different images. Image Smoothing can be used in order to reduce the above mentioned types of noise. This is equivalent to suppressing the high frequency components present in the frequency domain.

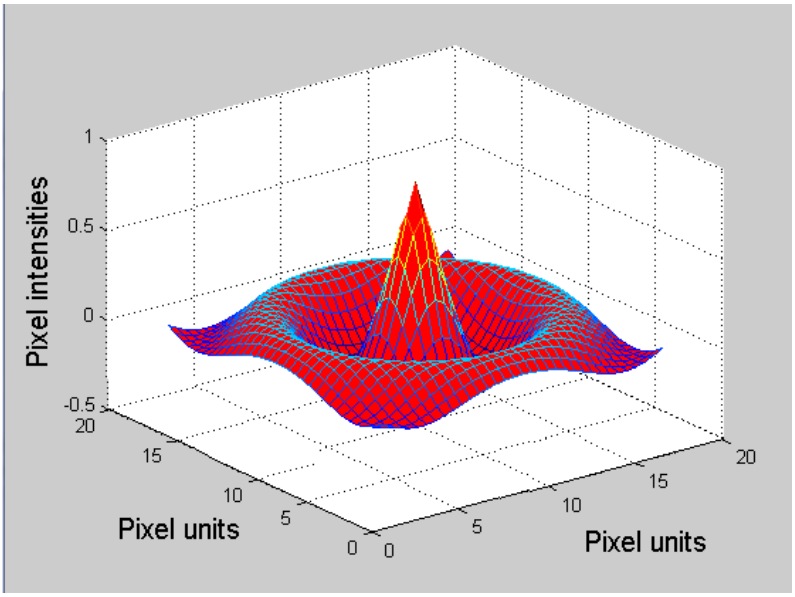


Figure 2.3a: Figure illustrating the surface plot for an intensity function of an image

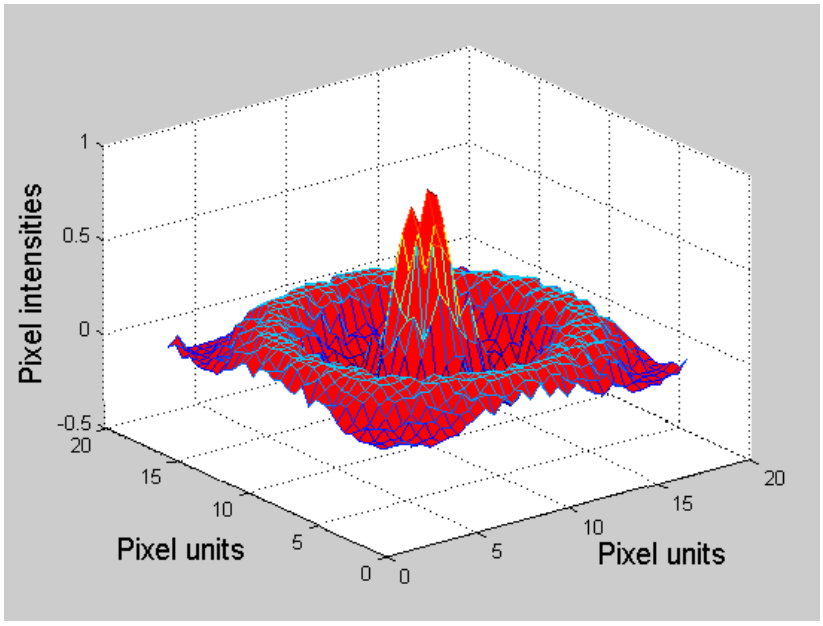


Figure 2.3b: Figure illustrating the surface plot for the intensity function of figure 2.3a with noise added

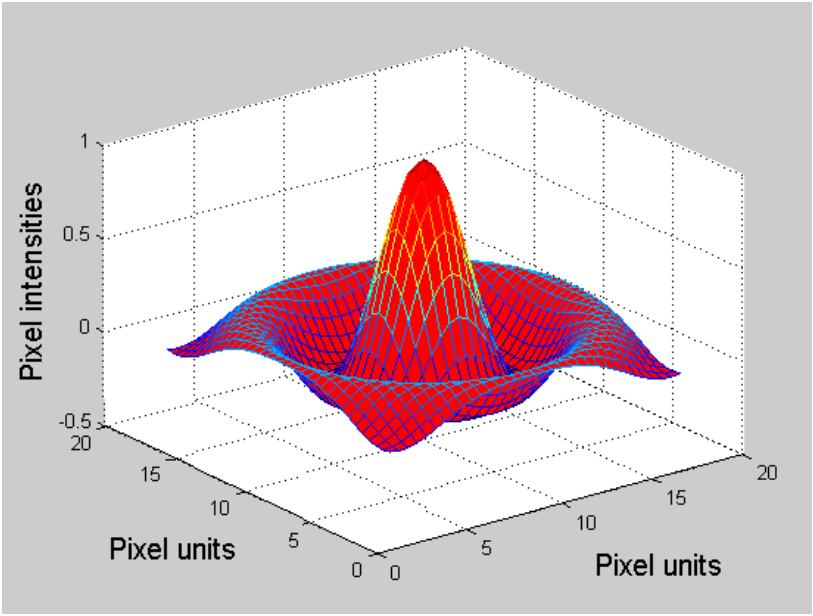


Figure 2.3c: Figure illustrating the surface plot for the intensity function of figure 2.3a after noise reduction

Figure 2.3 Difference between the intensity values for a noisy image vs. a smooth image

Figure 2.3a illustrates an example of a surface plot of the intensity function of an image. If noise is added to this intensity function, the resulting intensity plot will be similar to figure 2.3b. If a smoothing filter is now applied to this noise intensity function then the high frequency components are suppressed leading to figure 2.3c. Even though image smoothing offers a reasonable noise reduction, it can be seen that smoothing an image leads to blurring of the sharp edges which leads to loss of information about the geometry of the object under consideration. This can be deduced by comparing figure 2.3a and figure 2.3c. This is particularly important if the object under consideration is composed of only a few pixels. Some of the pixel intensities in figure 2.3 are set to negative values for convenience. Negative intensities are however not practically possible in images.

2.3.1 BOX FILTER:

A box filter or a mean filter as it is sometimes called, distributes every pixel with the same weight which is an average of its neighbours. Since all the weights allotted are the same, it is termed a box filter.

Figure 2.4 denotes a box filter with 16 neighbours with an average pixel value of 1. The mask value after normalization of this filter is 1/16. The smoothing operation using this filter is the convolution of every pixel with this 4x4 filter.

1	**1**	**1**	**1**
1	**1**	**1**	**1**
1	**1**	**1**	**1**
1	**1**	**1**	**1**

Figure 2.4: A box filter kernel of size 4x4

Figure 2.5a: Original Image

Figure 2.5b: Output after box filtering

Figure2.5: Image undergoing box filtering

Figure 2.5a shows the original image upon which box filtering algorithm of kernel size 4x4 is applied. 2.5b shows the output after box filtering.

2.3.2 GAUSSIAN SMOOTHING:

A significant disadvantage offered by a box filter is that the mask value of the filter is dependent on all pixel values of the image under consideration. This means that a single pixel that does not represent its true value can significantly impact the mean of the total image. Hence a gaussian filter is used for this purpose. A gaussian filter, unlike the box filter, does not assign equal weights to all the pixels but instead assigns weights similar to gaussian pdf.

The Gaussian function in one dimension is characterised by the given relation

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad (2.9)$$

The corresponding distribution in two dimensions is given as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \qquad (2.10)$$

It is important to note that a 2D Gaussian distribution is the product of two 1D Gaussian distributions applied to two orthogonal axes. Applying a gaussian smoothing on an image is identical to convolving every pixel in the image with a 2D Gaussian distribution.

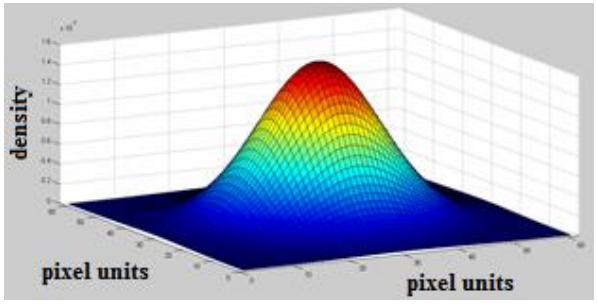


Figure 2.6a: Gaussian kernel surface view

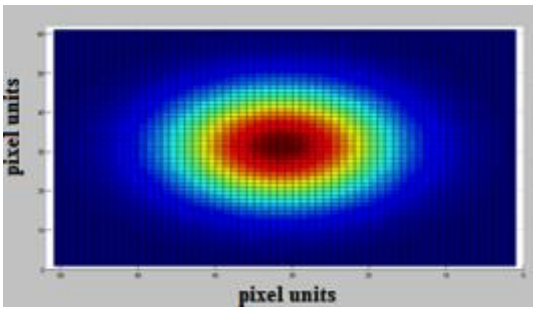


Figure 2.6b: Gaussian kernel top view

Figure 2.6: Surface view and top view of a Gaussian Kernel with standard deviation of x = 1 pixel unit and standard deviation of y = 1 pixel unit.

Figure 2.6a shows the Gaussian kernel in a 3d surface view. Figure 2.6b shows the top view of the Gaussian kernel. As shown below, a Gaussian filter is a linear filter that performs weighted average with a higher weight given to the central pixel and less weight to the neighbouring pixels.

Figure 2.7 shows the image after Gaussian smoothing of size 4x4 pixel units and a standard deviation of 2.75 pixel units. The input image used for smoothing is the same image shown in 2.5a. As with any smoothing scheme, the small features present in the original image are almost lost in the smooth image. However based on the application under consideration, smoothing may or may not be applied. This is particularly important in the current application where any excessive smoothing can lead to shape distortion which can eventually lead to incorrect results.

Figure 2.7: Image undergoing gaussian smoothing with kernel size 4x4 and standard deviation of 2.75 pixel units.

It can be observed that both the gaussian smoothing and box filtering are special cases of averaging filters. Assuming the noise present in the images is zero mean gaussian noise and using the image model described in equation 2.5,

$$I(i,j) = S(i,j) + n(i,j)$$

where I(x,y) denotes the image, S(x,y) denotes the signal and $n(i, j)$ denotes the i.i.d noise with distribution

$$G(0, \sigma^2)$$

$$A = \frac{1}{n^2}(\sum_{k=1}^{n^2} I)$$

$$E(A) = \frac{1}{n^2}\sum s^2 \tag{2.11}$$

$$\text{var}(A) = E[(A - E(A))^2] = \frac{\sigma^2}{m} < \sigma^2 \tag{2.12}$$

Hence, the variance of the average is lower than the variance of pixel noise which eventually leads to reduction in the overall noise.

2.3.3 THRESHOLDING

Thresholding can be defined as a segmentation technique in which a color image or a grayscale image is converted into a binary image such that the object under consideration in the foreground is separated from the background clutter. As the name suggests, it is a process of comparing the intensity value of each pixel of an image with a threshold value to determine if it is a useful pixel or not. Since it converts a higher resolution image into binary format, it can be used as a technique for data compression in a real time data intensive application. Based on specific applications, images can have a single threshold or multiple thresholds. There are already a number of methods discussed in the literature that perform automatic estimation of thresholds from images. These most often determine the threshold using image gray level histograms [22-25].

Thresholding comes in many different forms like Binary, Inverse Binary, Threshold Truncation, Threshold to Zero and Threshold to Zero Inverse. In the proposed algorithms discussed in this thesis, any type of thresholding can be applied. Since it works with any thresholding scheme, a detail about each scheme is not required. However, an example of each scheme is given in figure 2.8. All of these techniques can be easily formulated mathematically. For instance a binary threshold scheme can be formulated as

$$dst = \begin{cases} \max value & if\ \ src(x,y) > threshold \\ 0 & otherwise \end{cases} \qquad (2.13)$$

The above equation formulates such that each pixel in src(x,y) whose intensity value is greater than threshold will be assigned to max value and the rest of the pixels assigned to zero.

As shown in figure 2.8, different types of thresholding schemes can be used to extract different objects under the same scene. In spite of all the advantages offered by thresholding, there are certain conditions under which image thresholding should not be used. For instance, when objects to segment are not very easily distinguishable or when there is an improper or rapidly varying illumination.

ORIGINAL IMAGE

Figure 2.8a

THRESHOLD TRUNCATION

Figure 2.8b

THRESHOLD BINARY

Figure 2.8c

THRESHOLD BINARY INVERSE

Figure 2.8d

THRESHOLD TO ZERO

Figure 2.8e

THRESHOLD TO ZERO INVERSE

Figure 2.8f

Figure 2.8: Examples of different types of thresholding schemes

It is important to note that all of the methods discussed above refer to static or fixed thresholding. There are however other methods for dynamic thresholding already available in the literature that are outside the scope of this thesis. As already mentioned, thresholding is a good technique for

object detection from background noise. This is one of the primary reasons behind employing thresholding in the proposed algorithm.

Also, image processing algorithms are very demanding in terms of processing and memory requirements. The motivation behind the current research is to ultimately employ it in mobile handset environments which have very stringent memory and processing power requirements. Hence any reduction in the processing of the algorithm can lead to significant reductions in its processing time, rendering the algorithm to be usable in real-time. Since thresholding is a very effective data compression technique that offers all the above mentioned advantages, it is used in the proposed algorithm.

2.3.4 EDGE DETECTION

In image processing and computer vision systems, it is often necessary to determine the edges of an object. This is particularly true in the current research where edges offer significant information about the geometry of the object ultimately leading to trajectory estimation. There are various forms of the edge detection routines that are specific to the application at hand. The difficulty of edge detection is the noise associated with the necessary derivative operations. All the routines depend on different derivative values of $\left|I^{(x)}\right|, \left|I^{(x,x)}\right|, \left|I^{(y)}\right|, \left|I^{(yy)}\right|$. One of the most common routines for edge detection is the Canny edge detection. In this algorithm initially a quantity I^e is computed as follows:

$$I^e = \left|I^{(x)}\right|^2 + \left|I^{(y)}\right|^2 \tag{2.14}$$

where $I^{(x)}$ and $I^{(y)}$ are the image derivatives in X and Y directions respectively. I^e is then compared against two threshold parameters denoted as λ_{high} and λ_{low}. A binary output denoted by $I^b_{i,j}$ is then assigned depending on how I^e compares with these two thresholds based on the following rule:

$$\begin{aligned} I^b_{i,j} &= 1 \text{ if } I^e_{i,j} > \lambda_{high} \\ I^b_{i,j} &= 0 \text{ if } I^e_{i,j} < \lambda_{low} \end{aligned} \tag{2.15}$$

where i and j denote pixel indices. If $\lambda_{low} < I^{e}_{i,j} < \lambda_{high}$, the pixel is accepted as 1 only if the pixel is connected to a pixel with $I^{e}_{i,j} > \lambda_{high}$ as a neighbour. Examples of the Canny edge detector are given below.

Figure 2.9a: Original image

Figure 2.9b: Output after Canny edge detection

Figure 2.9: Image undergoing Canny edge detection.

Figure 2.9a shows the original image under consideration. Figure 2.9b shows the output of a Canny edge detector.

Similar to the Canny edge detector, the Sobel operator performs a 2D spatial gradient measurement on an image. These 2D spatial gradient measurements correspond to the high spatial frequencies that correspond to edges. In order to better understand the Sobel operator, an example is given in this section. Let a signal f(x) represents the edge that shows the jump in the pixel intensity, as shown in figure 2.10a.

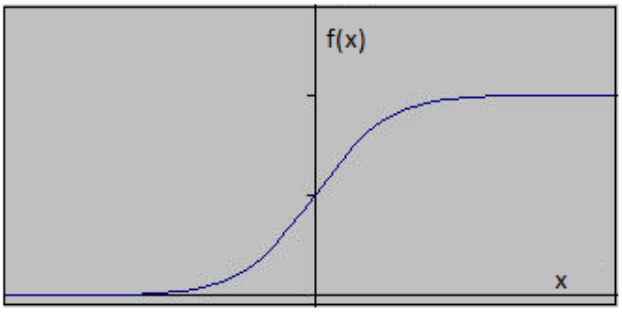


Figure 2.10a: 1D intensity image plot

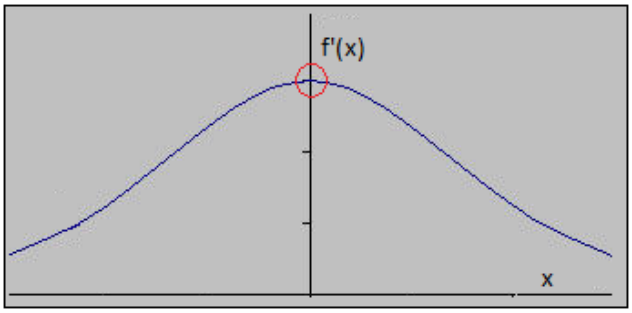


Figure 2.10b: First derivative of the signal f(x) with respect to x

The first derivative of the signal f(x) with respect to x represents the gradient of the signal denoted by $f'(x)$ as shown in figure 2.10b.

It can be seen that the first derivative of the signal f(x) attains a maximum at x=0. The second derivative of the same signal with respect to x gives the waveform shown in figure 2.10c.

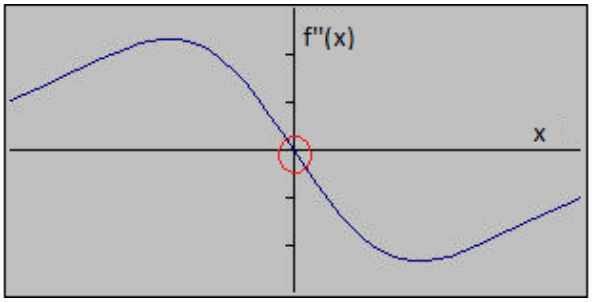


Figure 2.10c: Second derivative of the signal f(x) with respect to x

The edges determined by the Sobel operator correspond to those points that have $f''(x) = 0$. As shown in figure 2.10d, the first derivative in the X direction can be computed using the first principle as follows:

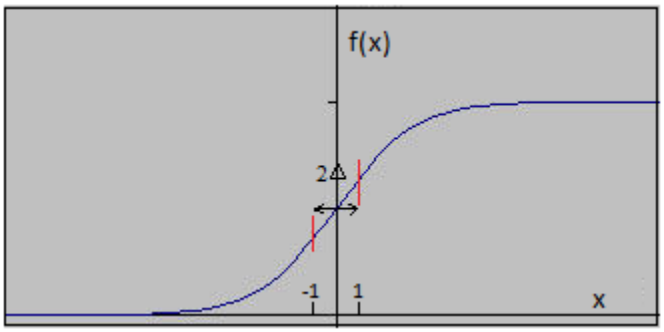


Figure 2.10d: Kernel derivation using first principle

$$f^{(x)}(x_0) \cong \frac{f(x_1) - f(x_{-1})}{\Delta} \tag{2.16}$$

From equation 2.16, the first derivative can be written as

$$f^{(x)}(x_0) \cong \frac{-f(x_{-1}) + 0Xf(x_0) + f(x_1)}{\Delta}$$

where X denotes multiplication. The scaling factor of $\frac{1}{\Delta}$ can be ignored as it is constant for the entire kernel. Hence the coefficients of the kernel for first derivative are given as follows.

$$G_{X-direction} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Differentiating it again gives the coefficients for second derivative as

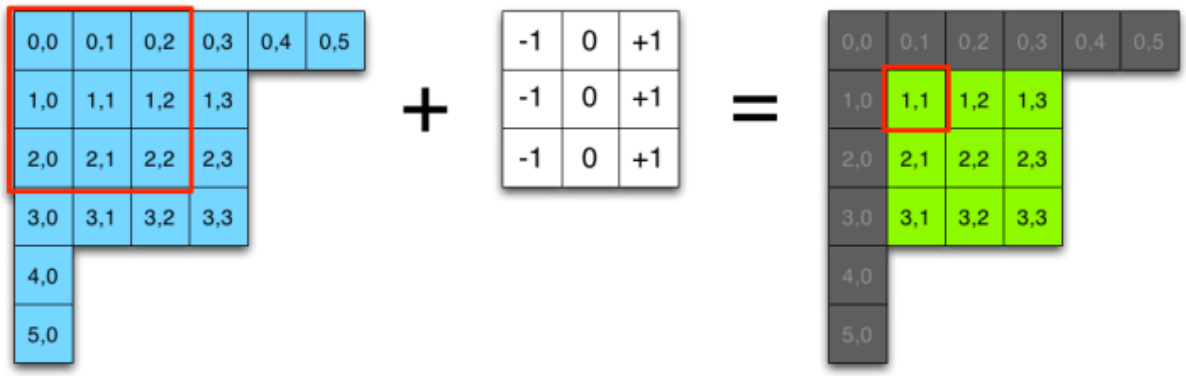
$$f^{(xx)}(x_0) \cong \frac{f^{(x)}\left(\frac{x_0 + x_1}{2}\right) - f^{(x)}\left(\frac{x_{-1} + x_0}{2}\right)}{\Delta} \tag{2.17}$$

$$f^{(xx)}(x_0) \cong \frac{\left(\frac{f(x_1) - f(x_0)}{\Delta}\right) - \left(\frac{f(x_0) - f(x_{-1})}{\Delta}\right)}{\Delta} \tag{2.18}$$

$$f^{(xx)}(x_0) \cong \frac{f(x_1) - 2f(x_0) + f(x_{-1})}{\Delta^2} \tag{2.19}$$

Hence the kernel coefficients for the second derivate are given as [1 -2 1].

The output of every pixel in a kernel operation is performed such that each pixel is made to convolve with the center of the kernel called ‘anchor’ in a raster scan fashion as shown in figure 2.10e.



Original image Kernel Final image F_{output}

Figure 2.10e: Kernel convolution in a raster scan fashion [26]

Figure 2.10: Kernel derivation for Sobel operator

A kernel applied in the horizontal direction, as shown in figure 2.10e will need to have the aforementioned coefficients for its rows. Since the derivative to be estimated is along the X-direction only, the y-coordinate should remain unchanged. Hence the output for one such anchor point (1,1) is given as follows:

$$F_{output}[1,1] = \left(f[0,0]X-1\right)+\left(f[0,1]X0\right)+\left(f[0,2]X1\right)+$$
$$\left(f[1,0]X-1\right)+\left(f[1,1]X0\right)+\left(f[1,2]X1\right)+\left(f[2,0]X-1\right)+\left(f[2,1]X0\right)+\left(f[2,2]X1\right) \quad (2.20)$$

F_{output} represents the final output and f represents the original image before convolution operation was performed and X denotes multiplication. Filter kernels are designed such that they minimize shift and filtering side lobes similar to the considerations for typical DSP filtering [27]. Hence the center row containing the anchor is usually given more weight than the other rows by multiplying with an additional scaling factor of 2. The final kernel to be used to find the derivative along the X-directions is given as follows.

$$G_{X-direction} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Similarly the kernel along Y-direction can be found as

$$G_{Y-direction} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I(x,y) \quad \text{and} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I(x,y)$$

where $I(x,y)$ is the input test image and $*$ denotes 2D convolution. The output after convolution at every pixel is a scalar value which can be computed using equation 2.20. If bi-directional operator needs to be employed then the approximate gradient at each pixel in the image can be calculated using the following formula:

$$G = |G_x| + |G_y| \tag{2.21}$$

Note that the output of the convolution operation at a point is a scalar such that in equation 2.21, Gx and Gy are scalars. The example given in figure 2.11 shows the output after using a horizontal, vertical and both horizontal and vertical Sobel operations simultaneously on the original test image described in figure 2.9a. The matlab function 'graythresh' was used to first find the grey level threshold of the image, which was found to be 0.04 and then the same value was used to find the Sobel operator in all three directions.

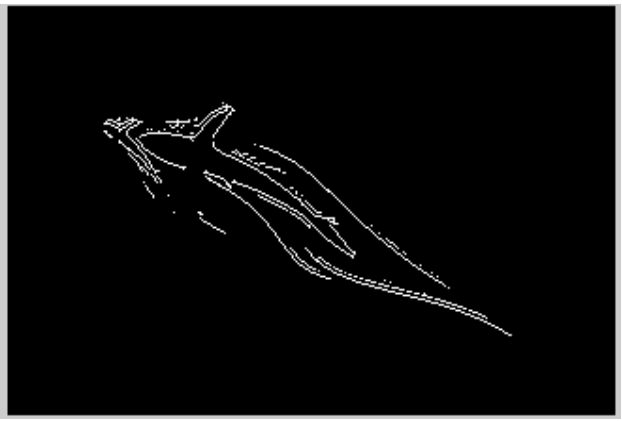


Figure 2.11a

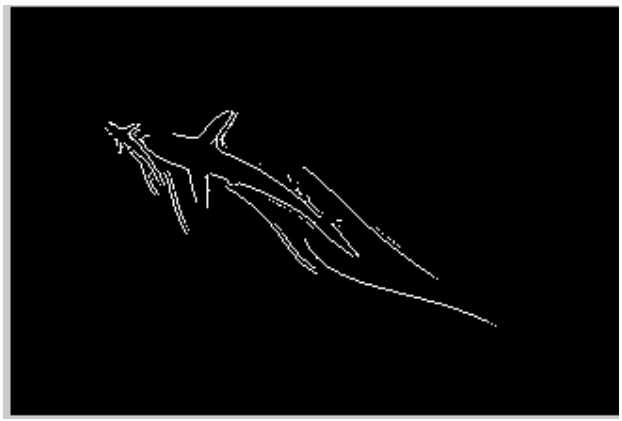


Figure 2.11b

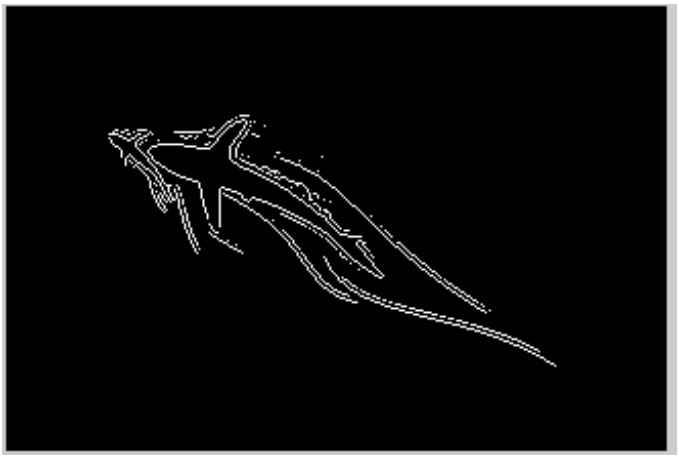


Figure 2.11c

Figure 2.11: Sobel operator output to an image in various directions. 2.11a: Sobel operator output in horizontal direction. 2.11b: Sobel operator output in vertical direction. 2.11c: Sobel operator output in both horizontal and vertical directions simultaneously

2.4 CONTOUR DETECTION

A contour can be defined as any list of points that are connected such that they form a structure. Often the objects under consideration are not straight lines and may consist of curved structures like circles or ellipses. In order to detect figures such as these, contour detection techniques are employed. In the present context, contours are made up of points such that each entry in the sequence leads to the very next entry in the contour ultimately encompassing the entire contour. This type of an arrangement is termed as a ‘Linkedlist’.

As discussed previously, thresholding an image produces a binary image. In image processing literature, such binary images can also be produced by other techniques like the Canny edge detector [19]. The binary images hence produced can be made to retrieve contours in the image using the algorithm [28]. The algorithm implements a border-following raster scan like technique for topological analysis. Additionally it is a good idea to first perform Canny edge detection prior to contour detection as this can aid in removal of background noise.

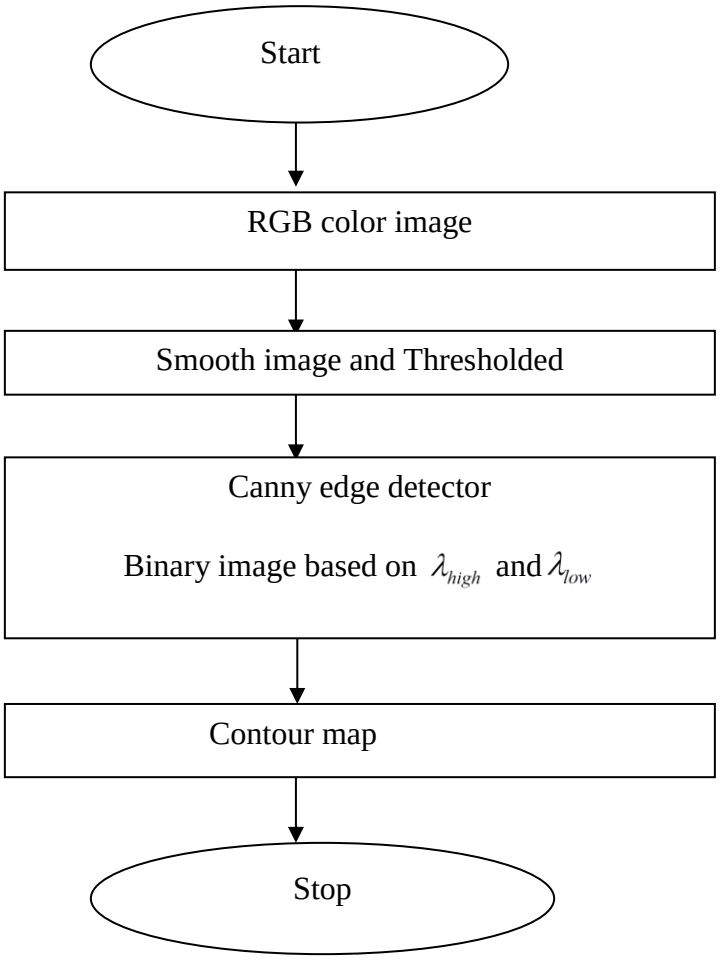


Figure 2.12: Flowchart for contour map of an image

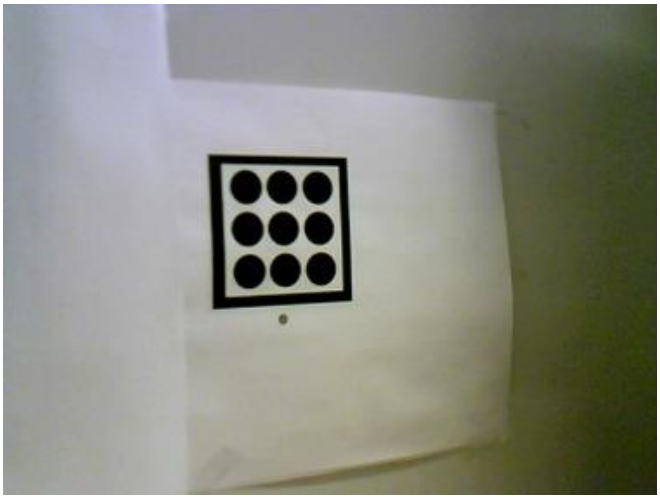


Figure 2.13a

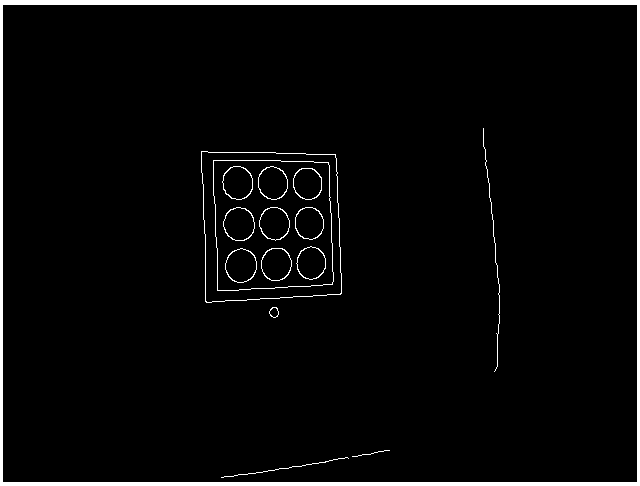


Figure 2.13b

Figure 2.13: Contour map of the test image. Figure 2.13a: Input RGB test image. Figure 2.13b: Output of the contour detection after processing various stages described in figure 2.12.

Figure 2.12 illustrates the flowchart for estimating the contour map. Figure 2.13a shows the test image undergoing contour detection. Figure 2.13b shows the output after processing with contour detection algorithm.

The algorithm shown in figure 2.12 starts with an RGB image which is converted into a binary image using the thresholding scheme described in section 2.3.3. A 'Threshold Binary' scheme was employed for the contour detection test. A gaussian smoothing was also performed to reduce the noise associated with the background clutter. As described in section 2.3.2, smoothing has to be employed very carefully as insufficient smoothing leads to the presence of a lot of noise

making it difficult to extract meaningful information from the marker while excessive smoothing leads to loss of information in the test image as mentioned in section 2.3. It can be observed that even after smoothing and edge detection, contour detection still leaves some traces of noise or background clutter. This is because neither thresholding nor edge detection are perfect processes.

2.5 PERSPECTIVE TRANSFORMATION

An affine transformation can be defined as a transformation that preserves the collinearity of the structure i.e., all points lying in a line still lie on a straight line after the transformation [29]. The ratio of the distances is also preserved, for example the midpoint of a line segment remains the midpoint after transformation. Functionally, this is equivalent to parallelograms mapping to other parallelograms. Affine transformation being a 2D transformation works well for weak perspective assumption [30] but cannot always hold well in 3D real world applications. For this purpose another transformation called Perspective Transformation (PT) is employed. PT can be defined as a spatial transformation in which the 3D world coordinates (X, Y, Z) map on 2D (x, y) image plane. Since this transformation is non-linear, the geometry is not preserved unlike affine transforms forcing parallelograms to be mapped to trapeziums. It is noteworthy that all affine transforms are perspective transforms but the converse is not necessarily true. This section is dedicated to understanding PT. PT is inherently composed of rotation and translation. Hence these concepts are discussed first following a detailed explanation about PT.

An image or a ROI inside an image undergoes translation along a certain axis if all the pixels within that region translate by the same amount. Figure 2.14 shows an example of the same in which an image is translated by 100 pixels to the right along the x-axis. Let the destination and source images corresponding to pixel index (i, j) in this example be denoted by ‘d’ and‘s’ respectively. Let x and y denote corresponding axes. The transformation in case of pure translation along x-axis is given by

$$x_i^d = x_{i-100}^s \tag{2.22}$$

$$y_j^d = y_j^s \tag{2.23}$$

Figure 2.14a

Figure 2.14b

Figure 2.14: Image undergoing translation along X-axis by 100 pixels. Figure 2.14a: Input image. Figure 2.14b: Output after translation by 100 pixels. The red region in figure 2.14b is the area captured by 100 pixels on the x-axis. Since the width of the image was found to be 720 pixels, 100 pixels were arbitrarily selected to show the effect of translation. If too few pixels are selected then the effect will be difficult to observe. Note that any value greater than 100 pixels but less than 720 pixels can also be selected.

Similarly, an image or a ROI inside an image is said to have undergone a rotation with respect to a point such that the entire region of interest rotates by an equal amount about the axis of rotation. Hence the mapping is given

$$f_x(x,y) = x\cos(\alpha) + y\sin(\alpha) \qquad (2.24)$$

$$f_y(x,y) = -x\sin(\alpha) + y\cos(\alpha) \qquad (2.25)$$

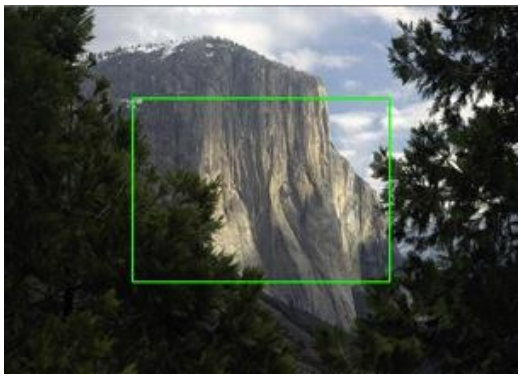


Figure 2.15a

Figure 2.15b

Figure 2.15: Image rotation of the selected region with respect to the image center by 30 degrees counter clockwise. Figure 2.15a: Input image with ROI shown in green. Figure 2.14b: Output after rotation by 30 degrees around the image center in counter clockwise direction.

In the example shown in figure 2.15, a region of interest (ROI) of an image, shown by the green box is rotated by an angle of 30 degrees in the counter clockwise direction and superimposed back on the image.

It can be seen in figure 2.14 and figure 2.15 that pure translation or pure rotation are examples in which the collinearity is preserved making them special cases of affine transformation. However a pure rotation or a pure translation with respect to one axis is rarely seen in 3D camera motion. Instead a combination of these is seen which can be catered by PT. Since the affine transformation is not directly employed in this thesis, detailed discussion is out of its scope. However a brief overview of affine transformations is given in Appendix B.

If an ideal pinhole camera observes a planar geometrical figure (like as a rectangle on the wall as shown) such that the plane of the wall is coplanar with the image plane, then the camera observes the true shape of the object (rectangle).

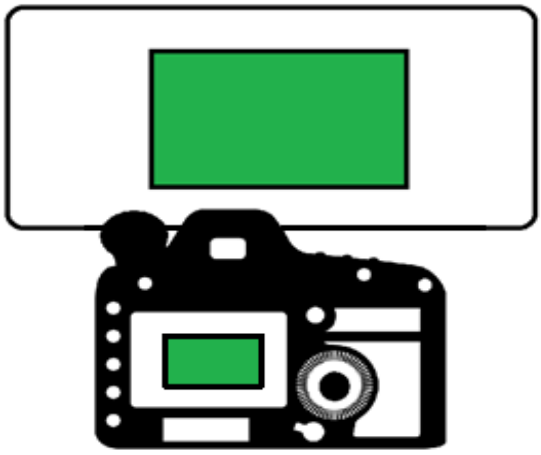


Figure 2.16a

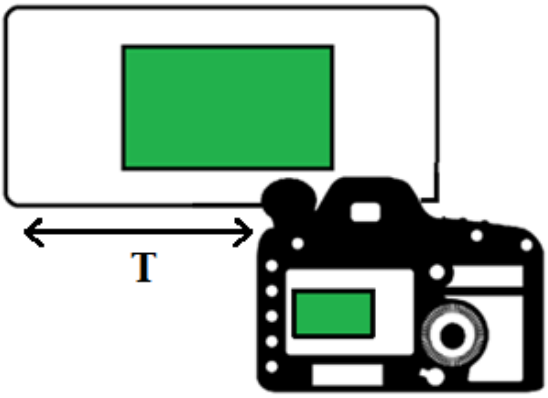


Figure 2.16b

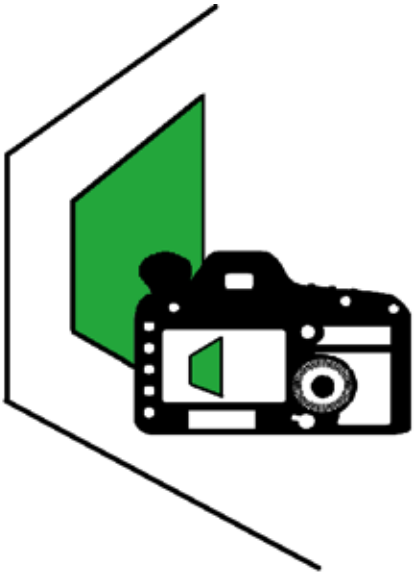


Figure 2.16c

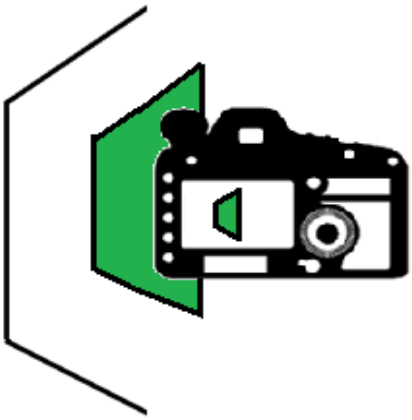


Figure 2.16d

Figure 2.16: A camera observing a stationary marker from different viewing angles [31]

As shown in figure 2.16a and 2.16b, if the camera moves to the right by T units then a pure translation is seen. Figure 2.16c shows a case in which an affine transformation does not hold well. As the trajectory progresses, it can be seen that the rectangle in figure 2.16b is transformed into a trapezium in figure 2.16c. This is the case of a Perspective transformation. If now the camera continues to move further to the right, a new trapezium with different dimensions is formed in the FOV of the camera. Similar to the transformation between figure 2.16a and figure 2.16b, this transformation between figure 2.16c and figure 2.16d is a small spatial transformation. However this transformation is different from the previous transformation in the sense that an affine transformation does not hold and PT is observed between these image frames.

The transformation between any two frames gives the relative PT between them. It is not only possible to estimate the PT between two consecutive image frames which is relevant in the current application, but it can be generalised to estimate PT between any two frames in the trajectory. Since PT can be decomposed into corresponding rotation and translation vectors, it is

possible to correctly estimate the exact trajectory between any two points in space along the trajectory.

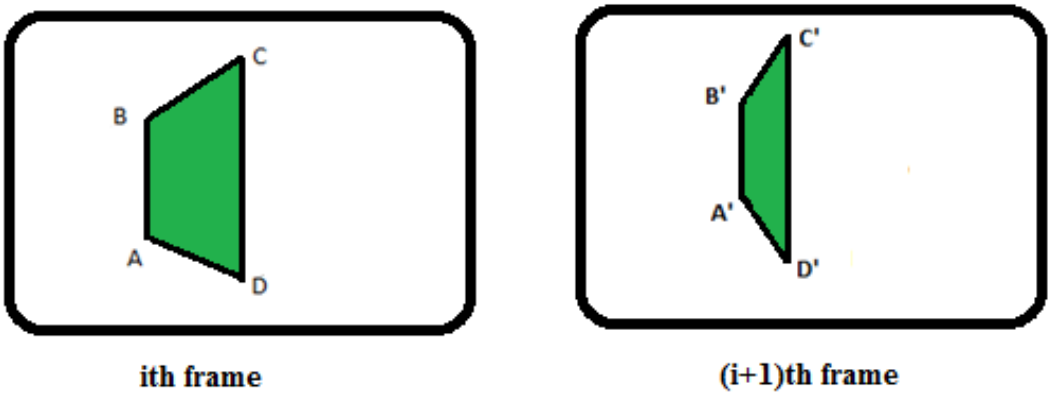


Figure 2.17: A quadrilateral ABCD in camera's i^{th} image frame undergoing PT to form quadrilateral A'B'C'D' in $(i+1)^{th}$ frame

As described in figure 1.4, the true geometry of the marker is rectangular in shape; however the presence of PT makes the FM appear as a quadrilateral. Figure 2.17 shows the transformation observed in the FOV of the camera in two consecutive frames as observed in figure 2.16c and figure 2.16d. This is still a case of PT and the corresponding rotation and translation between them can be extracted. In this thesis, left hand coordinate system has been used for both the world frame of reference and camera frame of reference.

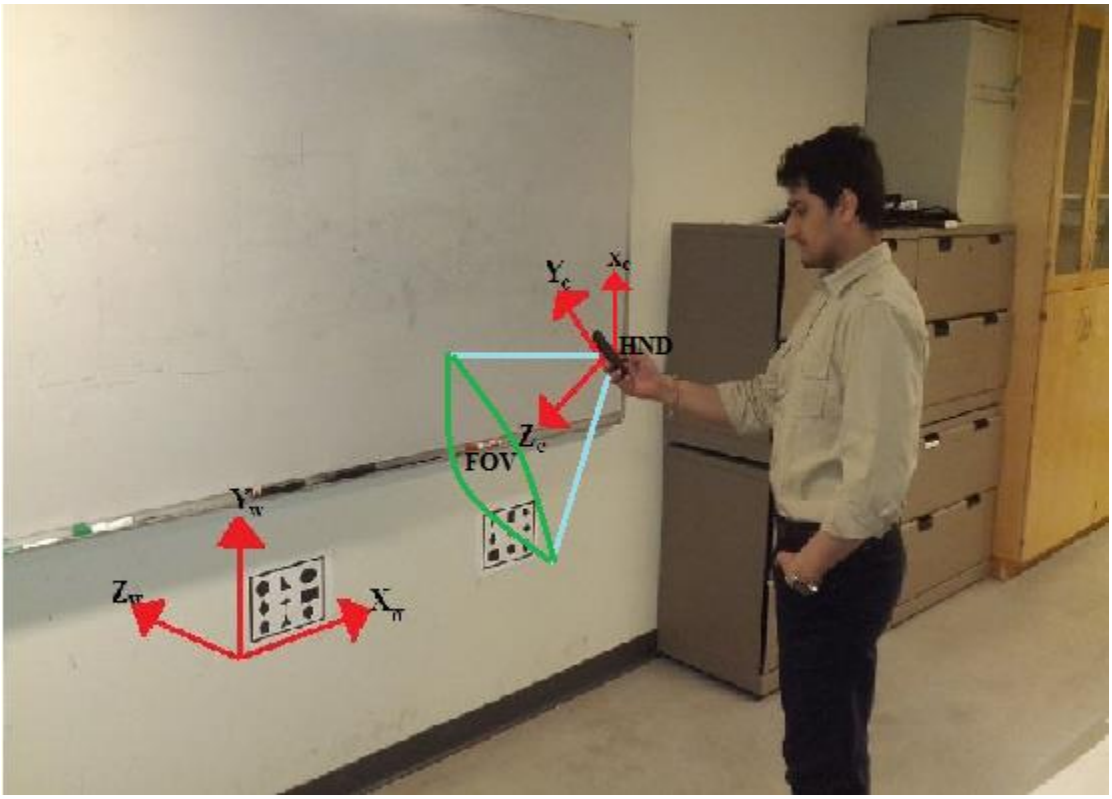


Figure 2.18: Figure depicting the camera and world coordinate system

Figure 2.18 illustrates the typical setup of the problem in which the user is holding the HND which forms the camera coordinate axes. This is represented by $\{\mathbf{X_c}, \mathbf{Y_c}, \mathbf{Z_c}\}$ which denote the camera coordinate system directional unit vectors. The corresponding world coordinate axes, as shown in figure 2.18 are given by $\{\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w}\}$ which represent the world coordinate system directional unit vectors. A pinhole camera model defining the relationship between a 3D point and its 2D corresponding projection has been employed. The setup for the same is given in figure 2.19 and figure 2.20.

Figure 2.19 shows the pinhole camera model with C as the camera center. The line segment formed by C and c denotes the optical axis. F denotes the focal length.

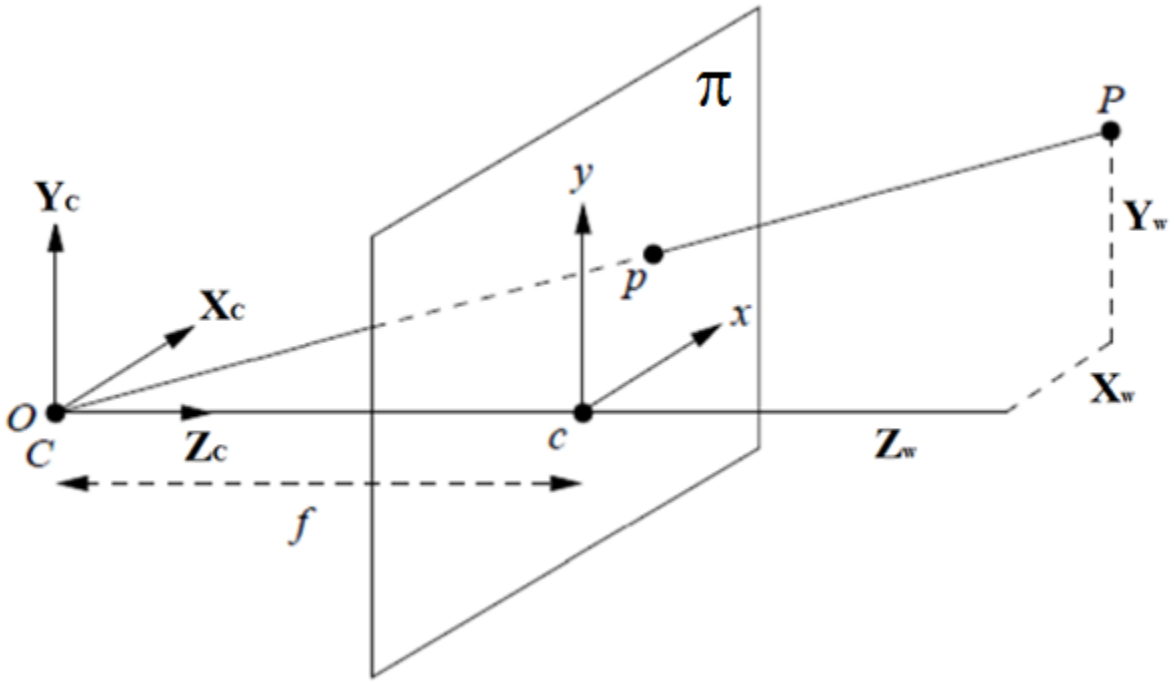


Figure 2.19: Pinhole camera model setup

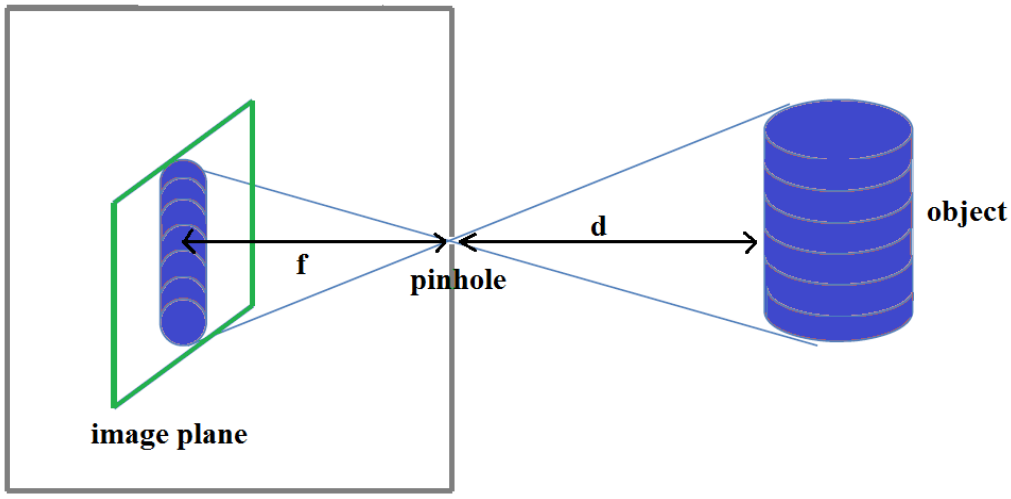


Figure 2.20: Pinhole camera for a 3D object

Consider a feature point $\mathbf{P}$. Let $\mathbf{P}_w$ and $\mathbf{P}_c$ represent the position vector from $\mathbf{O_w}$ (origin in world coordinates) and $\mathbf{O_c}$ (origin in world coordinates) to $\mathbf{P}$ respectively.

Let $\{\mathbf{x}, \mathbf{y}\}$ represent unit vectors of 2D camera image plane $\boldsymbol{\pi}$

Let the position vector $\mathbf{P_w}$ be composed of the following components:

$$\mathbf{P_w} = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \tag{2.26}$$

such that $$\mathbf{P_w} = x_w\mathbf{X_w} + y_w\mathbf{Y_w} + z_w\mathbf{Z_w} \tag{2.27}$$

Similarly the position vector $\mathbf{P}_c$ can be represented as

$$\mathbf{P_c} = x_c\mathbf{X_c} + y_c\mathbf{Y_c} + z_c\mathbf{Z_c} \tag{2.28}$$

This can be written in vector form as

$$\mathbf{P_c} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{2.29}$$

Subsequently, that projection of the point $\mathbf{P}$ onto the pinhole camera image plane is denoted as $\mathbf{p} = x\mathbf{X_c} + y\mathbf{Y_c} + f\mathbf{Z_c}$ where f is the focal length of the image plane. Based on the ideal pinhole model one has the following parameters

$$\begin{aligned} x &= f\frac{X_c}{Z_c} \\ y &= f\frac{Y_c}{Z_c} \end{aligned} \tag{2.30}$$

The mapping between the world and pinhole camera frames consists of translation and rotation operators denoted as **T** and **R** respectively. The direction of **T** depends whether it is referenced with respect to the world or the camera frames. Similarly, the rotation matrix **R** also has to be

referenced as a transformation from the world to the camera or vice versa. Also the rotation operators are not commutative.

To understand this concept more clearly there is another method that can be used. Instead of defining the coordinate system transformations, the desired relation between the position vectors $\{\mathbf{P}_w, \mathbf{P}_c\}$ is directly considered. Next the relationship of the image coordinates to the world frame coordinates of the scattering point of **P is** directly considered as

$$\mathbf{P}_c = \mathbf{R}(\mathbf{P}_w - \mathbf{T}) \qquad (2.31)\ [30]$$

The required rotation matrix and the translation vector can be defined using this transformation setting where the translation vector $\mathbf{T}$ is given as

$$\mathbf{T} = \mathbf{O}_w - \mathbf{O}_c \qquad (2.32)$$

which is referenced to the world coordinate frame. The rotation matrix **R** maps the world coordinate frame into the camera coordinate frame after translation. It is noted that **R** consists of the projection coefficients of the unit vectors of

$\{\mathbf{X}_w, \mathbf{Y}_w, \mathbf{Z}_w\}$ onto $\{\mathbf{X}_c, \mathbf{Y}_c, \mathbf{Z}_c\}$.

Let the rotation matrix be written as a matrix of row vectors as shown below

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_3 \end{bmatrix} \qquad (2.33)$$

such that

$$\mathbf{P}_c = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_3 \end{bmatrix} [\mathbf{P}_w - \mathbf{T}] = \begin{bmatrix} \mathbf{R}_1 (\mathbf{P}_w - \mathbf{T}) \\ \mathbf{R}_2 (\mathbf{P}_w - \mathbf{T}) \\ \mathbf{R}_3 (\mathbf{P}_w - \mathbf{T}) \end{bmatrix} \qquad (2.34)$$

From the similar triangles of the pinhole camera it follows that

$$x = f\,\frac{\mathbf{R_1}\left(\mathbf{P_w}-\mathbf{T}\right)}{\mathbf{R_3}\left(\mathbf{P_w}-\mathbf{T}\right)}$$
$$y = f\,\frac{\mathbf{R_2}\left(\mathbf{P_w}-\mathbf{T}\right)}{\mathbf{R_3}\left(\mathbf{P_w}-\mathbf{T}\right)} \qquad (2.35)$$
$$z = f$$

It is safe to assume that $Z_w = 0$ since the current application deals with stationary and planar markers in the form of FM and FMO:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R_1} & -\mathbf{R_1T} \\ \mathbf{R_2} & -\mathbf{R_2T} \\ \mathbf{R_3} & -\mathbf{R_3T} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \qquad (2.36)$$

Hence with $Z_w = 0$, equation 2.36 can be rewritten as

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{H} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

where $\mathbf{H}$ is a 3x3 map matrix with elements defined as

$$\mathbf{H} = \begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} \qquad (2.37)$$

Now the objective is to determine the nine components of $\mathbf{H}$ from the pinhole image components. Hence,

$$x = X_c/Z_c = \frac{H_{11}X_w + H_{12}Y_w + H_{13}}{H_{31}X_w + H_{32}Y_w + H_{33}}$$
$$y = Y_c/Z_c = \frac{H_{21}X_w + H_{22}Y_w + H_{23}}{H_{31}X_w + H_{32}Y_w + H_{33}} \qquad (2.38)$$

which can be rearranged as

$$H_{31}xX_w + H_{32}xY_w + H_{33}x = H_{11}X_w + H_{12}Y_w + H_{13}$$
$$H_{31}yX_w + H_{32}yY_w + H_{33}y = H_{21}X_w + H_{22}Y_w + H_{23}$$

These results in a pair of constraints expressed as

$$\begin{aligned} \mathbf{u_x b} &= 0 \\ \mathbf{u_y b} &= 0 \end{aligned} \tag{2.39}$$

where

$$\mathbf{b} = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{21} & H_{22} & H_{23} & H_{31} & H_{32} & H_{33} \end{bmatrix}^T$$
$$\mathbf{u_x} = \begin{bmatrix} -X_w & -Y_w & -1 & 0 & 0 & 0 & xX_w & xY_w & x \end{bmatrix}$$
$$\mathbf{u_y} = \begin{bmatrix} 0 & 0 & 0 & -X_w & -Y_w & -1 & yX_w & yY_w & y \end{bmatrix}$$

Note that assuming a set of 4 points in 2D gives 8 constraints but 9 coefficients of $\mathbf{H}$. This is consistent with the solution of the homogeneous equation given to within a scaling constant as

$$\begin{bmatrix} \mathbf{u}_{\mathbf{x},1} \\ \mathbf{u}_{\mathbf{y},1} \\ \vdots \\ \mathbf{u}_{\mathbf{x},4} \\ \mathbf{u}_{\mathbf{y},4} \end{bmatrix} \mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.40}$$

The homogeneous solution involves finding the right singular vector of the SVD of the 8x9 matrix that corresponds to the singular value of zero. Note that this singular vector is arbitrarily scalable. As such $\mathbf{H}$ is known to within a scaling factor. Once $\mathbf{H}$ is determined then the homogeneous coordinates of the pinhole camera can be determined and then the ratios of $x = X_c/Z_c$ and $y = Y_c/Z_c$ are determined. In order to determine the scaling associated with PT the dimensions of the FM or the FMO should be known.

2.6 CAMERA CALIBRATION

All the pre-processing methods mentioned earlier are usually applicable for estimating the external parameters of the camera system. However while mapping 3D objects on 2D planes,

there are certain internal camera quantities like camera distortion coefficients and principal point, etc. that play a vital role. Since most of these internal quantities remain constant with time, estimating them prior to using the camera is usually a good option.

There is vast literature that is already present on this topic. Broadly speaking, camera calibration can be performed in four main categories namely

- 3D apparatus based calibration
- 2D or planar objects based calibration
- 1D objects or collinear points based calibration
- Self-calibration

If the geometry of a 3D object is known to a good precision, then a 3D apparatus based calibration is a good option to use. This scheme of calibration employs two or three orthogonal plane setup [32]. Sometimes a plane is employed whose motion is purely translational [33]. 3D calibration can incur costly equipment charges. 2D calibration on the other hand is an efficient and inexpensive calibration scheme in which the camera is made to observe a planar object in various poses. Even though this scheme is not as accurate as 3D calibration, it is by far one of the most popular calibration methods in CV literature for its ease of setup. 1D calibration, which is a fairly new type of calibration [32], is composed of collinear point based objects and the calibration process is achieved by observing a moving object around a fixed point.

It would be appropriate to count self-calibration as 0D as it does not use any objects. The internal parameters are estimated on the fly observing a static scene. Since this thesis is predominantly based on FM and FMOs which are planar objects, using a 2D calibration scheme aligns perfectly with the central theme of using planar structures for navigation. It gives reasonably good results which suffice with the application at hand.

Almost all imaging devices introduce a certain amount of nonlinear distortions. Amongst all non-linear distortions introduced in an imaging system, radial distortion is the most severe [32]. Radial distortion tends to introduce a displacement of an image point from its ideal location. A positive radial displacement introduces an outward effect leading to barrel distortion, while a negative radial displacement introduces the inward looking pincushion distortion. The radial

distortion model described in [34-36] has been employed in this research. Figure 2.21 shows the different types of radial distortions.

If (x_d, y_d) and (x_u, y_u) represent the normalized distorted image points and the corrected image points respectively and $r_d = \sqrt{x_d^2 + y_d^2}$ represent the radial distance, then the radial distance r of the corrected image can be represented by the formula

$$r = r_d(1 - k_1 r_d^2 - k_2 r_d^4) \tag{2.41}$$

The corrected image points (x_u, y_u) can be calculated by using the formula

$$x_d = x_u(1 + k_1 r + k_2 r^2) \tag{2.42}$$

$$y_d = y_u(1 + k_1 r + k_2 r^2) \tag{2.43}$$

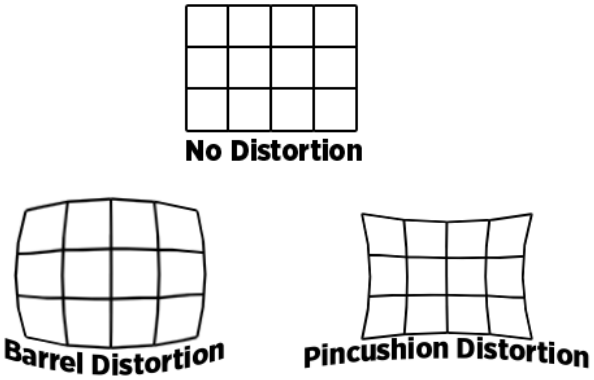


Figure 2.21: Types of Radial distortions [37]

Apart from radial distortion, the other most common type of distortion is the tangential distortion which occurs due to imperfect alignment of the camera lens not making it perfectly parallel to the image plane. It can be corrected using the following:

$$x_{corrected} = x_u + [2p_1 x_u y_u + p_2(r^2 + 2x_u^2)] \tag{2.44}$$

$$y_{corrected} = y_u + [p_1(r^2 + 2y_u^2) + 2p_2 x_u y_u]$$

The correction of camera distortion requires estimation of $[k_1, k_2, p_1, p_2]$. Along with distortion corrections, another important set of parameters needed to perform real-time CV processing are

the camera's intrinsic parameters. For a pinhole camera model with U and V representing the homogeneous pixel coordinates, any rigid body motion can be represented as

$$\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} = \mathbf{A}_{\mathbf{int}} \mathbf{A}_{\mathbf{ext}} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{2.45}$$

where the matrix $\mathbf{A}_{\mathbf{ext}}$ represents the extrinsic parameters, which is essentially the perspective transform using an ideal pinhole camera. $\mathbf{A}_{\mathbf{int}}$ is termed the "Camera matrix" with Intrinsic parameters. The intrinsic parameters are represented by the following matrix:

$$A_{\text{int}} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.46}$$

where f_x and f_y represent the focal length in pixel units along x and y axes respectively. c_x and c_y represent the coordinates of the principle point that usually occurs at the image center. The experiment given in figure 2.22 and 2.23 are employed for estimation of both distortion and intrinsic parameters.

Figure 2.22a and figure 2.22b show the two input images captured by an AGAMA V-1325R webcam quality camera. More details about the camera are given in chapter 3. Figure 2.22c and figure 2.22d show 35 feature points that are detected in each image. Figure 2.22e and figure 2.22f show the rectified images after camera calibration is performed.

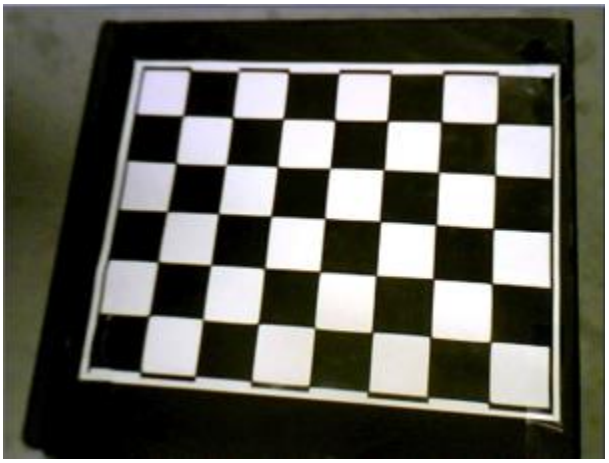


Figure 2.22a

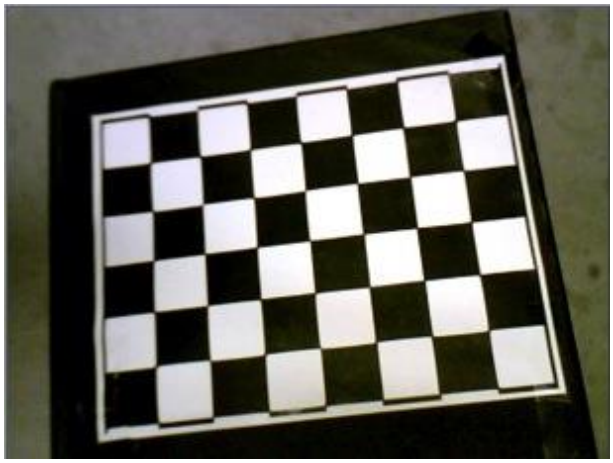


Figure 2.22b

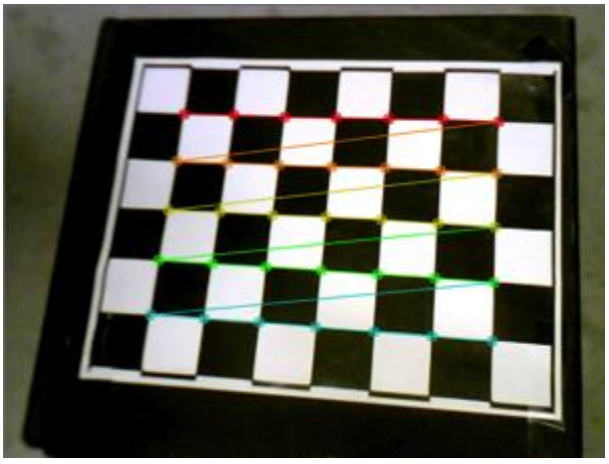


Figure 2.22c

Figure 2.22d

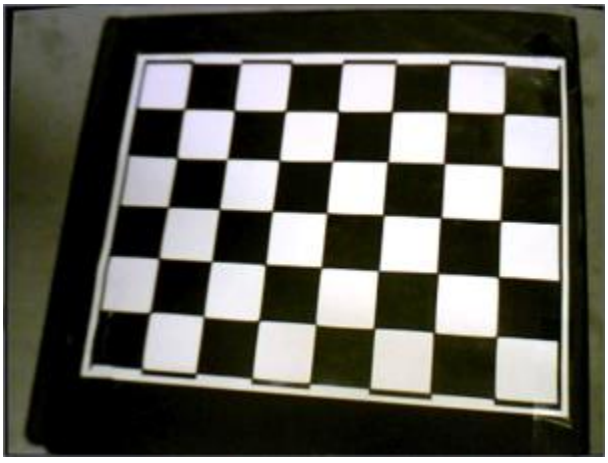


Figure 2.22e

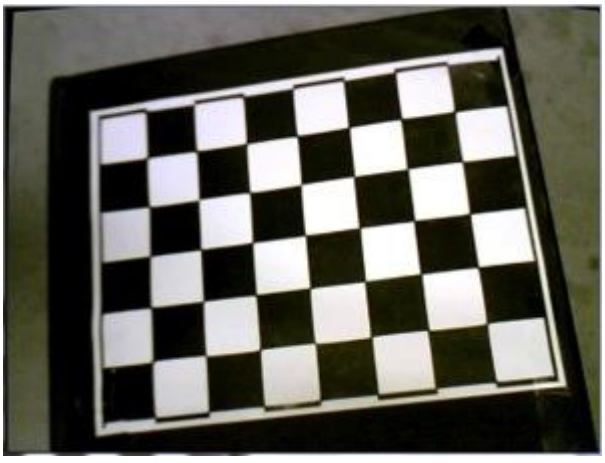


Figure 2.22f

Figure 2.22: Camera Calibration using a planar chessboard pattern

It can be seen that the camera undergoes barrel distortion as evident from the fact that the corrected image is similar to pincushion distortion. The feature points from these rectified images are taken and made to re-project back on the input images. Once this is done, the re-projection error in in unit of length (in cm) is calculated. As already mentioned in equation 2.45, the total transformation is composed of a product of intrinsic parameters (A_{int}) and extrinsic parameters (A_{ext}). Because the intrinsic parameters are constant over time, it is always a good idea to estimate them in advance and use them while computing the extrinsic parameters which define the trajectory of the camera.

The algorithm has been tested for two independent sets of 2 images and 3 images respectively and the results are documented below.

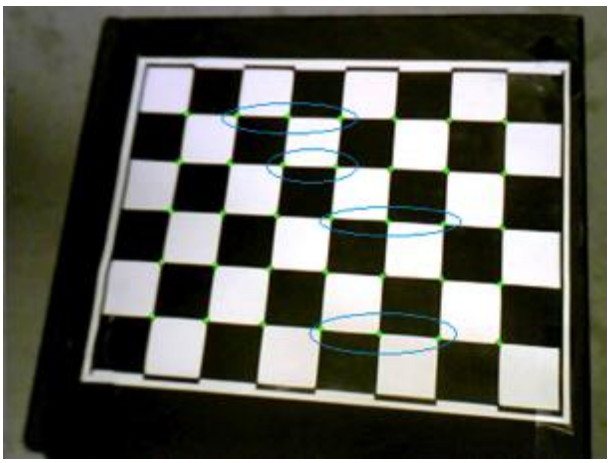


Figure 2.23a

Figure 2.23b

Figure 2.23: Calibration Re-projection Error

Figure 2.23a and 2.23b illustrate the re-projection error associated with the camera after the process of camera calibration. The green dots denote the feature points using the input images. The red dots denote the feature points from the rectified images. The blue ellipses in figure 2.23a are drawn to show some of the re-projected points that coincide. It can be observed that while some points do coincide, there are also some points that show variations up to a certain pixels proving that radial distortion has quite non-linear characteristics.

Another set of experiments with three images was performed. The results for both these set of experiments are given in table 1. The experiment shown in figure 2.22 and 2.23 correspond to

experiment 1. By comparing the two experiments, it can be deduced that there is a decrease in the average re-projection error implying that increase in the number of images can have an impact on obtaining better calibration. However the average re-projection error for set of 2 images is still quite small making it appropriate for real-time processing. The code and algorithm of camera calibration is modified using the standard code provided in [38].

Table.1. Camera distortion parameters and re-projection error for two independent experiments

Re-projection error	**Experiment 1**	**Experiment 2**
k1	0.09	0.12
k2	-0.04	-0.07
p1	-0.02	-0.03
p2	0.01	0.01
error sum for image 1	0.01 cm	0.01 cm
error sum for image 2	0.01 cm	0.01 cm
error sum for image 3	NA	0.01 cm
Average re-projection error	0.01 cm	0.01 cm

2.7 RANSAC

As already described in section 2.5, any 3D rigid motion can be described using perspective transform which can be reduced into corresponding rotation and translation vectors. Hence the problem of indoor navigation in the present context reduces down to the estimation of perspective transforms from consecutive images and decomposing them into rotation and translation vectors. The cumulative integration of these vectors gives the 3D trajectory of the camera. While this method seems quite robust theoretically, the rotation and translation vectors do contain a lot of outliers leading to a large error if used directly for navigation estimation.

Hence some form of outlier rejection mechanism has to be used in order to reject the error generated due to the presence of these outliers. This is the motivation behind using RANSAC.

RANSAC [39] which stands for "RANdom SAmple Consensus" is a method to estimate the parameters of the mathematical model generated by the observed data of an experiment that is robust to outliers as a process to remove them. RANSAC is not specific to vision systems alone and has been employed for various applications since its inception in 1981.

RANSAC can be defined as an iterative optimization method that uses a data-driven random sampling of the parameter space to find the maximum or minimum of the cost function [40]. The cost function in the current context is the model that is to be determined.

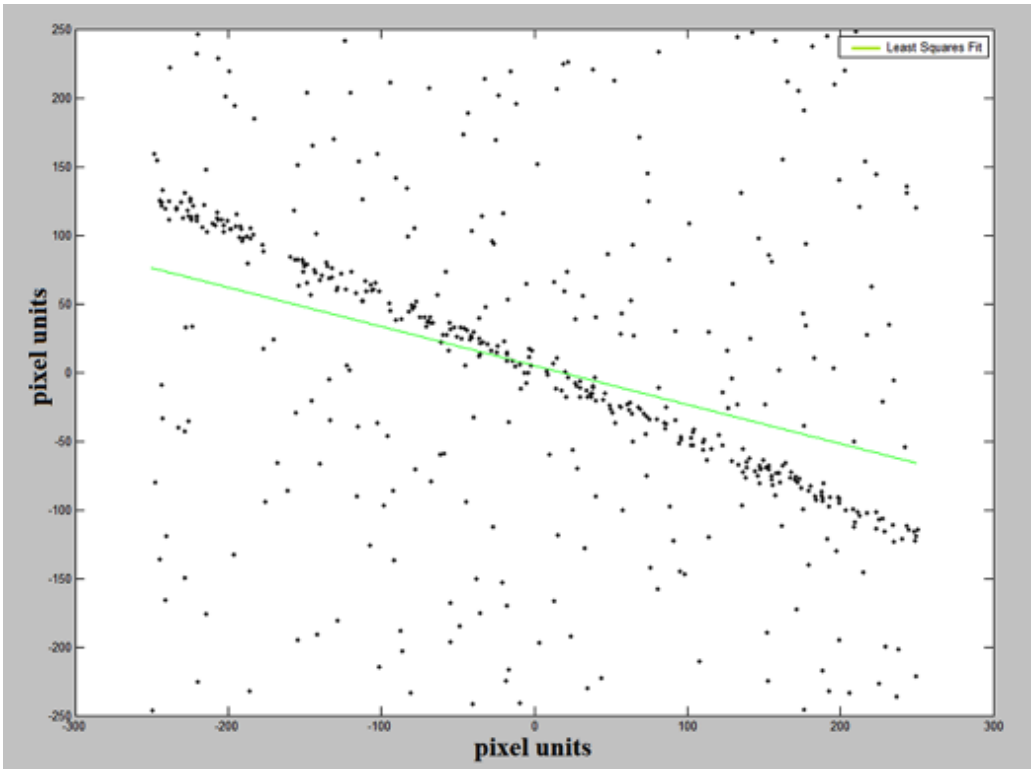


Figure 2.24a: A set of random points in pixel units with a linear mathematical model whose parameters are to be estimated

Figure 2.24a illustrates an example of RANSAC in which certain points with a linear model are observed in a 2D space. In order to relate better to the current application, pixel units are used. The example comprises of 500 points of which 50% are concentrated within a certain region forming a linear model. This region is the inlier or correct model region. The remaining 50% as

it can be seen are scattered throughout the two dimensional space forming the outliers. It can be seen from figure 2.24a that linear least squares do not give satisfactory results either. Hence in order to generate the correct model a threshold has to be set and the model estimation has to be performed within this threshold. This is shown in figure 2.24b.

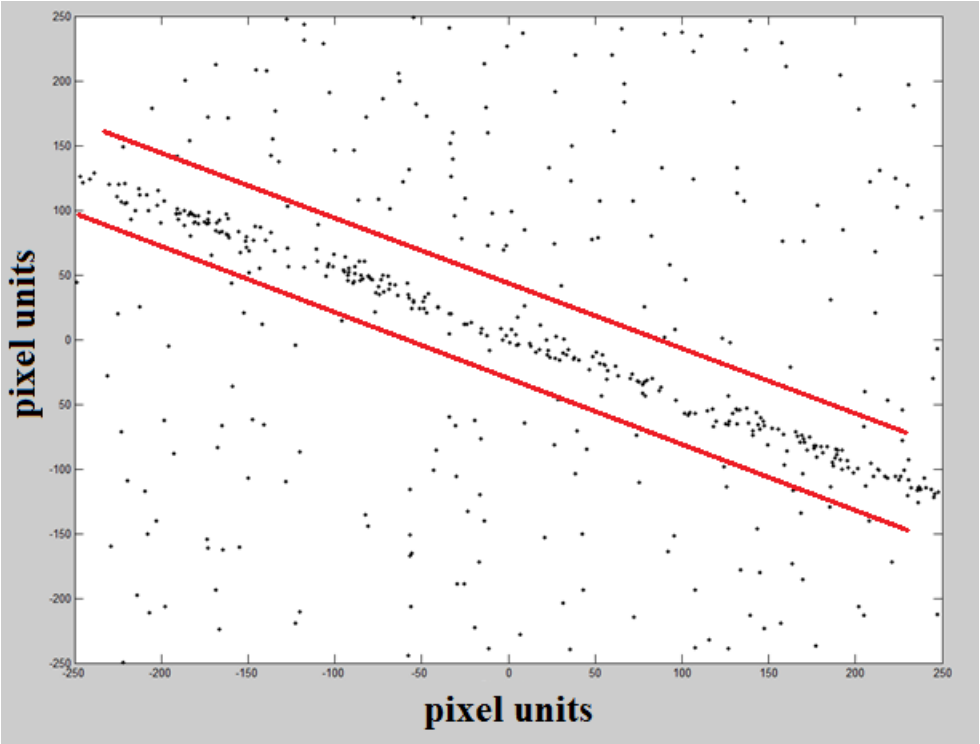


Figure 2.24b: Figure illustrating the threshold required for estimation model using RANSAC for one model iteration.

Threshold selection is an important step in RANSAC and has to be done very carefully. A very low threshold can lead to rejection of some points that actually correlate to the underlying model while a very high threshold leads to error introduced by the outliers.

In the current example, a minimum of 3 points are required in order to generate a line. Since more than 2 points are used to generate the model, the number of models generated is reduced. This reduces the computational complexity of the algorithm. The current example employs the slope-intercept form of a line $y = mx + c$ to generate the original model with a negative slope and adding noise to it to generate the outliers. The inlier standard deviation is set to 5 pixel units. The main idea behind RANSAC is to randomly select a subset of the available data and estimate the model parameters from this. To each data observation, a merit is then applied based on the

error residuals between the model and the data. Many such random subsets are used where the outlier data is identified as having consistently high error residuals. Once the outliers have been identified then they can be justifiably be excluded from the data set used to do the final computation of the model parameters. A few of the models generated are shown in figure 2.24c.

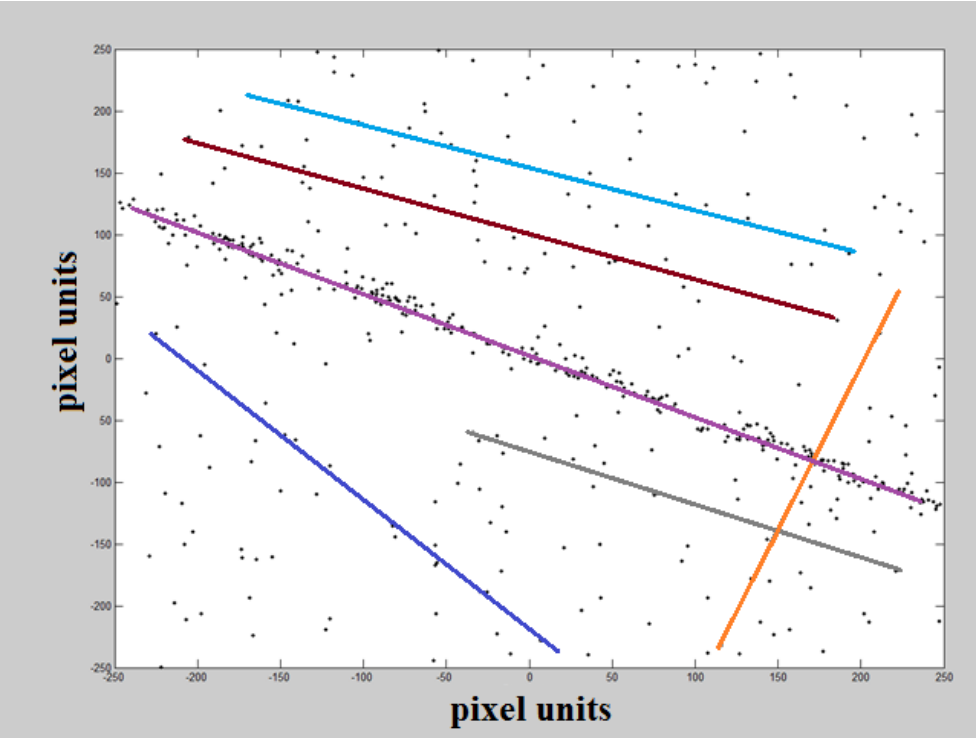


Figure 2.24c: Figure illustrating some sample models generated by RANSAC

Each model can be represented in polar form using the transformation

$$x\cos(\theta) + y\sin(\theta) = \rho$$

Where ρ represents length of the normal between the line and origin and θ represents the angle that the normal makes along the x-axis in counter clockwise direction. Hence each model can be represented with a corresponding value of (ρ, θ). Slope intercept form is not directly used as a vertical line corresponds to infinite slope which cannot be modelled. This also aligns well with Hough Transforms which is developed in the following section. Once this is done, the coefficients with the most inliers are selected as a model. Figure 2.24d shows the original model, the least squares output and the RANSAC output. It can be seen that RANSAC offers relatively accurate results.

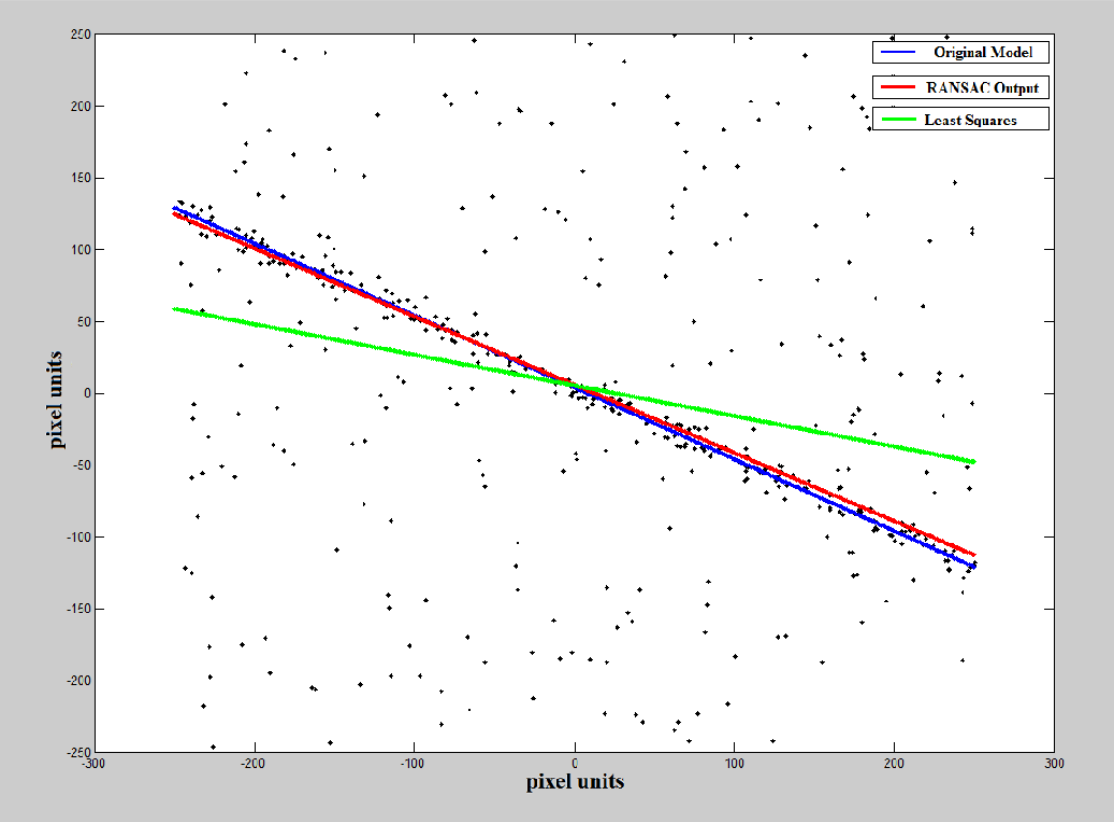


Figure 2.24d: Figure illustrating original model, least squares and RANSAC output

Figure 2.24: Mathematical model estimation using RANSAC and Least Squares

2.8 FIDUCIAL MARKER OF OPPORTUNITY

As shown in figure 1.4, the FMs considered in this thesis consist of rectangular structures with coded pattern inside it which are used for data association. The FMOs considered in this thesis were initially composed of the same rectangular structure of the FM but with no coded patterns inside. Intuitively this is correct as they do not add any additional information to the existing system. Since FMs and FMOs are the focus of this thesis, it is not sufficient to use non-coded planar FMs as FMO, and the theory of FMO has to be extended further by the use of Hough transforms. This section is dedicated to this objective. More effective techniques need to be employed in order to understand FMOs and use them for navigation or positioning purposes.

FMOs have been used for various purposes as described in the literature, however the approach to estimation of the extrinsic parameters (rotation matrix and translation for the perspective transform) has not been explicitly done. The use of FMOs has been rather indirect in the sense

that they have been used as preprocessing or at intermediate stages. For instance [41] uses lines that are inherently FMO in a scene for roll measurement using vanishing points. It can be argued that vanishing points itself act as FMO. However since every scene has a unique vanishing point, they can be generalised more as anchor point that are specific to a scene rather than FMOs that are constant and fixed either in time, space or both. The FMOs described in this thesis are not limited to certain shapes or points and can be extended to any planar structures. Since rectangles are the predominant figures in any scene, this chapter discusses FMOs emphasizing that their geometry is rectangular.

HOUGH TRANSFORMS

Hough Transforms (HT) have been used in this thesis to estimate rectangles in the scene. Hough Transform can be defined as a method that reduces a global detection problem in image space to a more easily solvable local peak detection problem in parameter space [42]. It can also be defined as a method of detecting parametric curves by exploiting the duality of the points of these curves in the image space and the parameters of that curve [43].

Hough transforms have been extensively used in this thesis because of the following reasons. Firstly HT can easily be implemented. This may be attributed to the fact that HTs are conceptually simple. Secondly HT gives reasonably good results even in noisy conditions. Thirdly, as it will be seen in chapter 4, missing data such as occlusions are handled very effectively with HT algorithms. Finally, even though HT is predominantly used for geometrical shapes like lines, circles or ellipses, they can also be generalised to any arbitrary figure using the Generalised Hough Transform (GHT).

This thesis will process FMOs as rectangles. GHT can very easily be implemented for the same. However since traditional HT offers good results for line segment detection, the same is used for rectangle detection in this thesis. The examples shown in figures 2.25, 2.26, 2.27 and 2.28 illustrate the logic behind the HT algorithm. Consider a straight line L1 with a slope ‘m’ and y-intercept of ‘c’ in an image as given in the figure 2.25. Hence the line can be represented in the slope intercept form of $y = mx + c.$

If another space called ‘parametric space’ is defined in terms of slope ‘m’ and y-intercept ‘c’, then the line L1 can be represented in the parametric space by the parameters (m,c). Also, since the parametric space is the space in ‘m’ and ‘c’, fixing ‘x’ and ‘y’ in Cartesian coordinate space can give different values of ‘m’ and ‘c’ by the transformation c=y-mx. Hence each point in Cartesian coordinate space corresponds to a line in parametric space. Since vertical lines have an infinite slope which cannot be represented in the parametric space of ‘m’ and ‘c’, a much better solution is to use polar coordinates.

The polar representation of the line L1 can be given as

$$x\cos(\theta) + y\sin(\theta) = \rho \qquad (2.47)$$

Where ρ represents length of the normal between the line L1 and origin and θ represents the angle that the normal makes along the x-axis in counter clockwise direction as shown in figure 2.25.

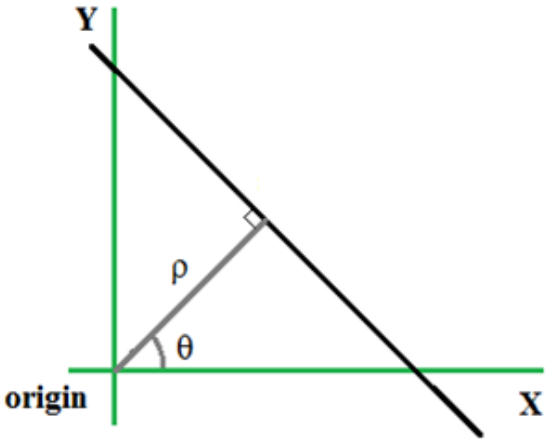


Figure 2.25: Hough transform of a line

Equation 2.47 can be rewritten as

$$y = -x\frac{\cos(\theta)}{\sin(\theta)} + \frac{\rho}{\sin(\theta)} \qquad (2.48)$$

If I(x,y) denotes the intensity image then keeping the point (x,y) constant in equation 2.48, all possible values are assigned to (ρ, θ) such that all possible lines passing through this point (x,y) are determined. Each point (x,y) in the image determines the family of possible lines this point generates in (ρ, θ) space that is weighted by the intensity function I(x,y). Once this is completed,

the cumulative (ρ,θ) gives the likelihood map of all possible lines in the image [44]. Suppose that an I(x,y) consists of a single line of parameters (ρ_o,θ_o), then in the (ρ,θ) Hough transform space there will be a peak centered at (ρ_o,θ_o) consistent with a high likelihood of a line existing in the image with these parameters. Using this parameterization, points in the Cartesian space are represented as sinusoids in the parameter space and lines in Cartesian coordinate space correspond to the intersection of these sinusoids thereby forming points in the parameter space. This is shown in figure 2.26 and 2.27. Note: Figure 2.25 shows the origin in the bottom left corner of the image. This is only done to explain the concept. However in most computer vision and image processing algorithms, the origin is represented at the top left corner as seen in figure 2.26 and figure 2.28.

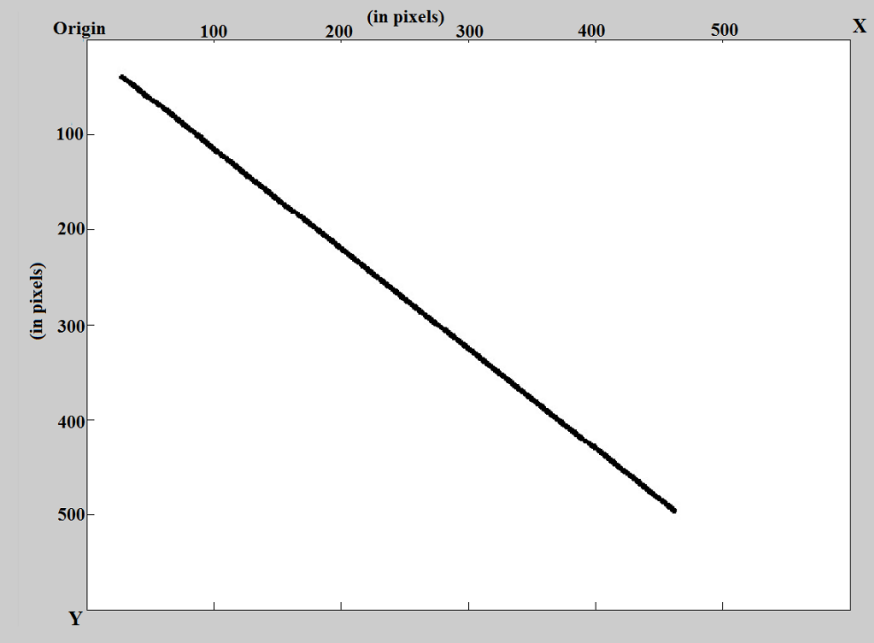


Figure 2.26: Line under consideration L1

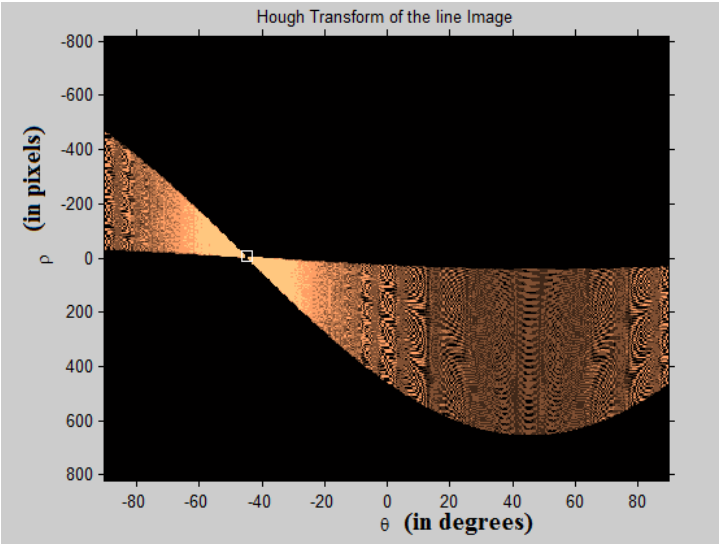


Figure 2.27: Hough Transform of the line L1

The line L1 in Cartesian space corresponds to a point in the parameter space as represented by a white square box in figure 2.27. Hence the inverse parameter mapping from the Hough transform parameter space to the Cartesian coordinates of the image space determines the line offset and orientation as shown in figure 2.28.

The essential idea behind the working of HT is to find imperfect instances of objects within a certain class of shapes like lines or circles using a voting procedure. This voting procedure is carried out in the parameter space which in current case is (ρ, θ).The voting procedure takes place by extracting the local maxima in the accumulator space.

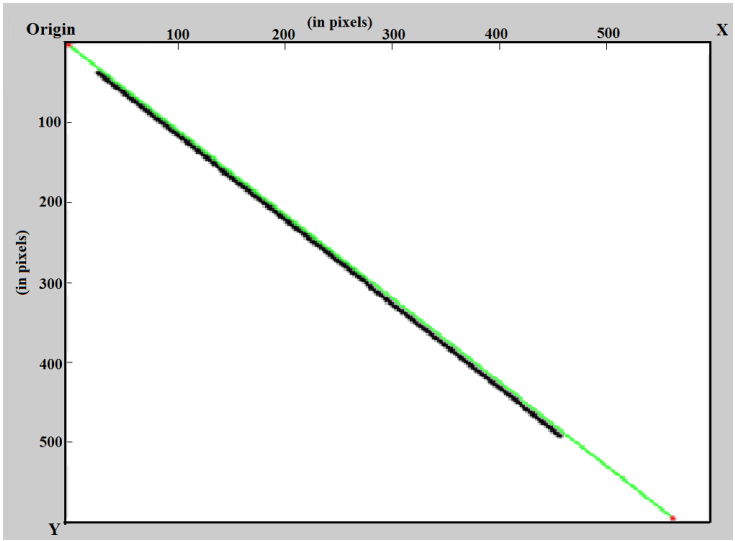


Figure 2.28: Line detection using HT

Chapter 3

Proposed System and Concepts

As already mentioned in section 2.5, PT is composed of rotation and translation vectors which together define the 6 DOF motion of the camera. These rotation and translation vectors form the extrinsic parameters. The overall transformation however consists of both intrinsic and extrinsic parameters of the camera as shown in equation 2.45. Estimation of intrinsic parameters was done in advance using the camera calibration so that the only parameters left for estimation are the extrinsic parameters. This chapter aims at explaining the proposed algorithm to perform extrinsic parameter estimation which eventually yields the overall trajectory of the HND. Since trajectory estimation performed with FMO alone is a relative trajectory with no start or end coordinates, some mechanism to anchor the trajectory is required. Since 3D location data can be associated with these coded markers, the same is used for anchoring the trajectory.

3.1 Proposed FM Algorithm:

The flowchart of the overall CV algorithm is given in figure 3.1. An introduction to camera calibration has been provided in section 3.1. Step 3 in the flowchart given as S & T stands for Smoothing and Thresholding which has been described in sections 2.3.1 and 2.3.3. Contour detection step has also been described in brief in section 2.4, followed by a brief introduction to Perspective transformation given in section 2.5. In the following section, process of marker coding is described. This is followed by a section on estimation of rotation angles from PT which is validated by an experiment in chapter 4. Finally, the similarities between the proposed algorithm and SLAM techniques are also given.

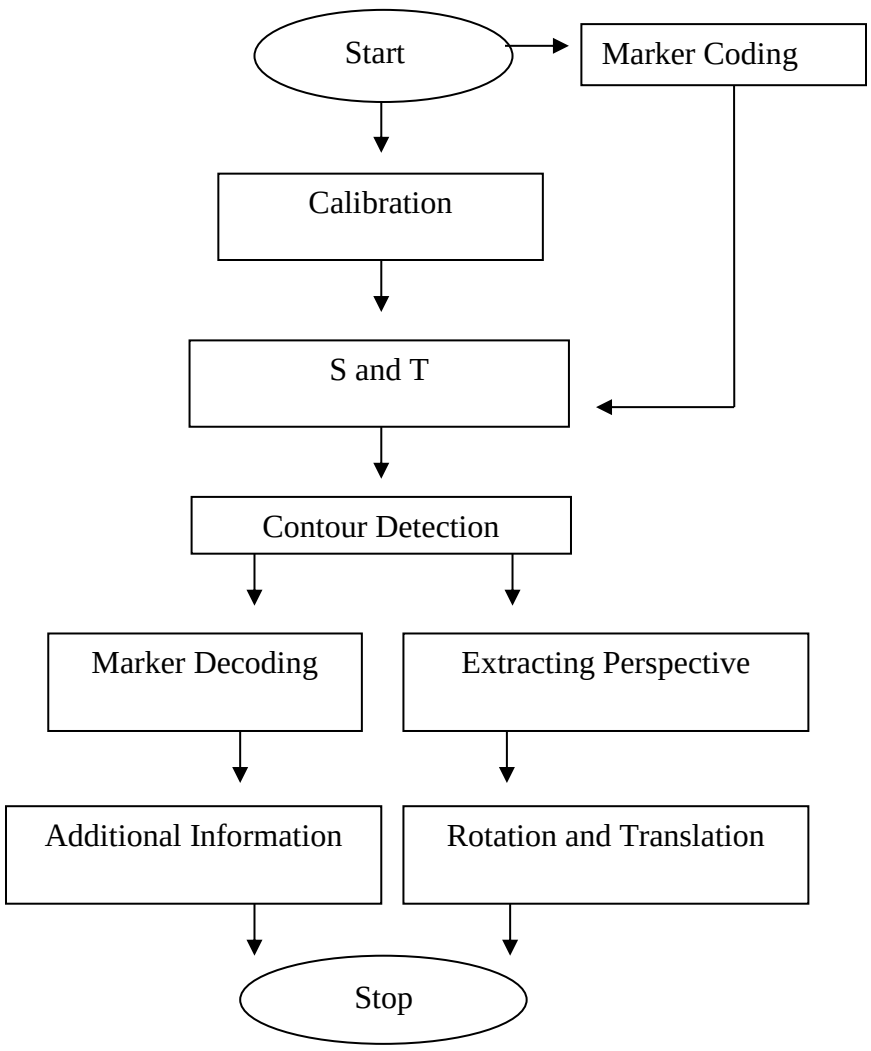


Figure 3.1: Flowchart of the proposed computer vision algorithm

3.2 MARKER CODING SCHEME:

Marker coding and decoding is an important step in the process of Homography estimation using coded FM. While pre-processing and calibration are important stages for the algorithm to function, it is the marker coding and decoding that truly adds value to the FM algorithm, as a decoded pattern can directly give an estimate of the initial location before the camera starts the

trajectory. This step is also important as it helps identify the points that are used for homography estimation using 4 point algorithm [45].

The marker decoding takes place by identifying contours in the image frame. Since an OpenCV programming function library version 2.1 [46] has been used in the algorithm, the already existing and optimised CVFINDCONTOURS function has been used for this purpose. The thresholded output of the pre-processing stage gives a good binary image output that can directly be used by the cvFindContours routine. Contours as already mentioned in section 2.4 are a sequence of points in which every entry has a corresponding value that encodes information about the next entry in the contour (i.e. linked list data structure). The underlying principle behind the contours in OpenCV is the use of contour trees that were first described in [28]. Since binary images are used for the process of contour extraction which occupies a very small memory size, the process becomes quite fast and hence can be used for real-time processing.

It is important to note that the presence of cluttered objects in the environment sometimes does pose a problem as false positives occur while marker decoding. In order to avoid this problem, the markers are designed such that two stages of contours are detected, the first comprising of the outer contour detection followed by an inner contour detection. The FM marker is considered a genuine marker for decoding only if both contour stages are present.

Once the marker has been identified as a genuine marker, the algorithm counts the number of figures present inside this marker. Since binary images have been used, this process consists of counting all the non-zero elements inside the marker. Initially when the first set of experiments were performed, figures were manually decoded such that each geometrical figure identifies with a certain code, namely 1 for circle, 2 for ellipse, etc., which were found by using Hough circle and Hough ellipse like algorithms. The relative position of these codes in the marker determines their position in the final code. For example, if the marker is composed of three circles as shown in figure 1.3, then the code for such a pattern is 111. This value assigned to each row is referred as row_count in this thesis. If more than 1 row exists in the FM, then the marker code consists of a total of all the individual row_count values.

Since the above mentioned scheme would need an underlying identification algorithm for all possible figures which is practically impossible, a different marker coding scheme was later

employed in which the total number of contours were identified irrespective of shape and, based on the number of such contours present in the marker, a code was given to each of these markers. Even though this algorithm does not produce unique markers as in the former process, it does save a lot of computational power which is more important. Also marker coding and decoding are areas that have a lot of literature associated with them. Hence any coding scheme can be employed for anchoring the trajectory. The essential goal of this thesis is not to design a robust marker coding/decoding scheme but to perform navigation based on FMO.

It will also be seen in chapter 4 that every FMO introduced in the FOV of the camera leads to the addition of a static drift in the trajectory. This can be minimised if coded FMs are used either at the end of the trajectory or along the trajectory with FMO.

The four vertices of the rectangular FMO are referred as marker_points in this thesis. Identifying the inner level contour helps in identifying these marker_points. The marker_points from consecutive image frames can be used as corresponding points from which a perspective transform can be extracted. The pseudo code of the process is given below where H(i) denotes the PT computed for each frame. Once this PT is determined, the corresponding values of rotation and translation can be extracted from H(i).

```
Estimate marker_points set
Estimate marker_points_index
For i=1:N
{
If (marker_points_index ~= 1)
{
src= marker_points (i-1)
dst= marker_points (i)
H(i)=GetPerspectiveTransform (src, dst)
}
}
```

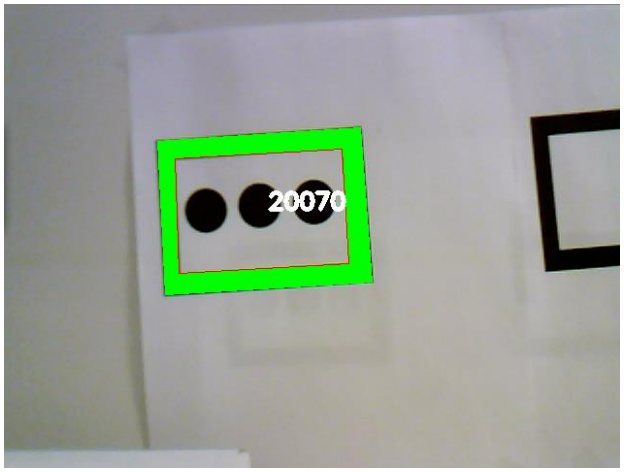


Figure 3.2: Example of a FM with unique ID

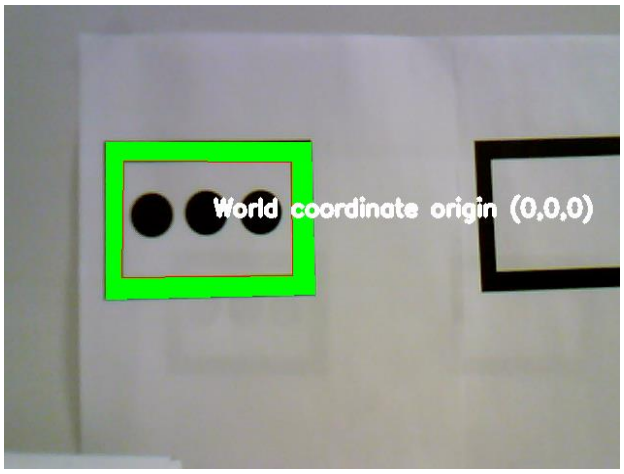


Figure 3.3: Example of a coded fiducial marker with three dimensional data association

Figure 3.2 is an example of a coded FM with its unique ID. The same ID was then used to encode location in the form of world coordinate origin as shown in figure 3.3.

3.3 OBTAINING ROTATION ANGLES FROM PERSPECTIVE TRANSFORMS

Egomotion is based on identifying feature points (FP) in the camera FOV which are tracked from one frame to the next. The Lucas Kanade Optical Flow method is generally used to determine the correspondence of the feature points (FPs) between these frames [47]. There are various methods discussed in the literature that help to determine if the tracked FPs are stationary or not. Furthermore another important consideration is that the FPs should not be very close to the edge of the image frame as it might exit the image frame making an FP impossible to track. Also FPs

may be identified as corresponding to a known landmark or anchor point and are therefore suitable for absolute trajectory estimation.

For egomotion that is limited to planes or surfaces, a single camera is generally sufficient. However for full 6 DOF egomotion with three orientation angles and three translations, a stationary single camera solution becomes ill-posed. There are a number of different approaches possible to cater to such a scenario. One possible solution is to use a set of two 3 DOF egomotions and then integrate both solutions. Another approach is to use two cameras in a stereoscopic configuration such that the depth information is extracted and used for tracking. The third and presumably the most widely accepted method is to consider two consecutive frames from the same non-stationary camera at two different times as two frames being obtained from two identical cameras in a stereoscopic setup. By using this method we can apply all the rules of epipolar geometry between two consecutive frames [48]. The same idea is employed in the current application for 6 DOF egomotion estimation.

As already mentioned in equation 2.45, the homogeneous pixel coordinates for any rigid body motion can be estimated as follows:

$$\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} = \mathbf{A}_{\mathbf{int}} \mathbf{A}_{\mathbf{ext}} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{3.1}$$

where

$$A_{ext} = \begin{bmatrix} \mathbf{R}_1 & -\mathbf{R}_1\mathbf{T} \\ \mathbf{R}_2 & -\mathbf{R}_2\mathbf{T} \\ \mathbf{R}_3 & -\mathbf{R}_3\mathbf{T} \end{bmatrix} \tag{3.2}$$

$$\mathbf{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$ where T_i represents the translation in the i^{th} axis

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \\ \mathbf{R_3} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$ represents the rotation along all three axis combined together.

Let us consider a frame, whose x-axis is rotated by an angle α, the y-axis by an angle β and z-axis by an angle γ, to obtain the corresponding axes for the next frame. Consider

$$\cos(\alpha) = c(\alpha)$$
$$\cos(\beta) = c(\beta)$$
$$\cos(\gamma) = c(\gamma)$$
$$\sin(\alpha) = s(\alpha)$$
$$\sin(\beta) = s(\beta)$$
$$\sin(\gamma) = s(\gamma)$$

The differential rotation matrices for each of the three axes are given by

$$\mathbf{R}^{\Delta}_{x}(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c(\alpha) & s(\alpha) \\ 0 & -s(\alpha) & c(\alpha) \end{pmatrix} \tag{3.3}$$

$$\mathbf{R}^{\Delta}_{y}(\beta) = \begin{pmatrix} c(\beta) & 0 & s(\beta) \\ 0 & 1 & 0 \\ -s(\beta) & 0 & c(\beta) \end{pmatrix} \tag{3.4}$$

$$\mathbf{R}^{\Delta}_{z}(\gamma) = \begin{pmatrix} c(\gamma) & s(\gamma) & 0 \\ -s(\gamma) & c(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

where $\mathbf{R}^{\Delta}_{i}(\theta)$ represents the differential rotation about the i[th] axis by an angle θ; θ being positive in the counter clockwise direction as viewed along the axis towards the origin.

Then the total differential rotational matrix between the two frames is given by $\mathbf{R}^{\Delta}_{z}(\gamma)\mathbf{R}^{\Delta}_{y}(\beta)\mathbf{R}^{\Delta}_{x}(\alpha)$

We already know from section 2.5 that $\mathbf{P_c} = \mathbf{R}_t\left(\mathbf{P_w} - \mathbf{T}_t\right)$ where

$\mathbf{P}$ is the scattering source point in a generic coordinate system (world or camera)

$\mathbf{P}_w$ is the position vector from $\mathbf{O}_w$ (origin in world coordinate system) to $\mathbf{P}$

$\mathbf{P}_c$ is the position vector from $\mathbf{O}_c$ (origin in camera coordinate system) to $\mathbf{P}$

$$\mathbf{R}_t^{\Delta} = \mathbf{R}_z^{\Delta}(\gamma)\mathbf{R}_y^{\Delta}(\beta)\mathbf{R}_x^{\Delta}(\alpha) \tag{3.6}$$

where t refers to the time instance.

Also since $\mathbf{R}_i^{\Delta}(\theta)$ is a specialised direction cosine matrix, it is orthogonal $\left(\mathbf{R}_i^{\Delta}(\theta)\right)^{-1} = \left(\mathbf{R}_i^{\Delta}(\theta)\right)^{T}$ and its inverse is also the "reverse rotation": $(\mathbf{R}_i^{\Delta}(\theta))^{-1} = \mathbf{R}_i^{\Delta}(-\theta)$

The total transformation is again orthogonal since each rotation matrix is orthogonal. This means

$$(\mathbf{R}_z^{\Delta}(\gamma)\mathbf{R}_y^{\Delta}(\beta)\mathbf{R}_x^{\Delta}(\alpha))^{-1} = (\mathbf{R}_z^{\Delta}(\gamma)\mathbf{R}_y^{\Delta}(\beta)\mathbf{R}_x^{\Delta}(\alpha))^{T} = \mathbf{R}_x^{\Delta}(-\alpha)\mathbf{R}_y^{\Delta}(-\beta)\mathbf{R}_z^{\Delta}(-\gamma)$$

Using (3.3), (3.4), (3.5) & (3.6) we have

$$R_z^{\Delta}(\gamma)R_y^{\Delta}(\beta)R_x^{\Delta}(\alpha) = \begin{pmatrix} \mathrm{c}(\gamma)\,\mathrm{c}(\beta) & c(\beta)\,\mathrm{s}(\gamma) & \mathrm{s}(\beta) \\ -s(\alpha)s(\beta)c(\gamma) - \mathrm{s}(\gamma)\,\mathrm{c}(\beta) & -\mathrm{s}(\gamma)\,\mathrm{s}(\beta)\,\mathrm{s}(\alpha) + \mathrm{c}(\gamma)\,\mathrm{c}(\alpha) & \mathrm{s}(\alpha)\,\mathrm{c}(\beta) \\ -c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) & -c(\alpha)s(\beta)s(\gamma) - \mathrm{c}(\gamma)\,\mathrm{s}(\alpha) & \mathrm{c}(\beta)\,\mathrm{c}(\alpha) \end{pmatrix} \tag{3.7}$$

where

$$\alpha = \tan^{-1}(c_{3,2} / c_{3,3}) \tag{3.8}$$

$$\beta = \sin^{-1}(c_{3,1}) \tag{3.9}$$

$$\gamma = \tan^{-1}(c_{1,2} / c_{1,1}) \tag{3.10}$$

with c being the total matrix shown in (3.7) and $c_{i,j}$ represent the i^{th} row and j^{th} column of the c matrix.

For smaller angles, α, β and γ, $\cos(\alpha) \approx 1$ and $\sin(\alpha) \approx \alpha$. Hence

$$R_z^{\Delta}(\gamma)R_y^{\Delta}(\beta)R_x^{\Delta}(\alpha) = \begin{pmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & -\alpha & 1 \end{pmatrix} \tag{3.11}$$

Estimation of roll pitch and yaw can directly be done using (3.7) or (3.11) if the approximation is to be followed. The underlying assumption for both of these cases is that the order of the motion that the camera undergoes is known.

An example of determination of rotation angle of an FMO from perspective transform is given in chapter 4. In order to facilitate the understanding, the example uses a single image that was rotated about its center along a single axis. The angle by which the image was rotated is then estimated from the algorithm.

Note: angle estimation from an FMO requires the determination of the edges of the FMO which is discussed in chapter 4.

3.4 SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) TECHNIQUES

For visual navigation schemes, the presence of a map can provide significant information. This is true because the presence of an observed anchor point can be put to use by comparing it with the anchor points of the map and defining an initial estimate with a certain variance. This initial estimate can be employed with Recursive Bayesian filtering schemes to perform navigation. Even though this research does not directly aim at mapping, the use of such maps can still prove to be very useful. Estimating the map coordinates while simultaneously performing localization is termed as SLAM. This sections aims at illustrating the similarities between the proposed algorithm and SLAM techniques.

Maps can be broadly classified into two categories namely “location-based”, corresponding to the anchor points in the current context and “feature-based”, corresponding to FMO. While maps generally represent the state at the time the map was drawn, this is not necessarily consistent with the state of the environment at the time the map is used. Hence, online determination of the map coordinates is often advisable.

In a probabilistic sense, SLAM can be of one of two types namely “Online SLAM” and “Full SLAM” [49]. Online SLAM involves estimation of variables, namely location coordinates in the current context at instantaneous time ‘t’. Online SLAM is incremental in the sense that past measurements are discarded once the processing is done. Contrary to this, full SLAM estimates the trajectory over the entire path along the map once all the measurements have been obtained. The proposed algorithm does not exploit full SLAM methodology while it does not any discard

previous measurement values either. I.e., hypothetically the measurement at t=k+1 does allow for the location estimation at t=k, t=k-1, t=k-2 and so on. Hence the proposed algorithm is a hybrid of online and full SLAM.

Coded FMs have been used in this thesis with an assumption that their position is known before the trajectory starts. Hence while calibrating the trajectory using such coded FMs, the proposed algorithm bears resemblance to SLAM whose correspondence is known which is called anchor points in [49]. In this type of SLAM scheme, the combined state vector to be estimated, not only consists of the current rotation and translation but also the current landmark (feature point or anchor point) which is in the FOV of the camera. This thesis also assumes that the location of the FMO is unknown to the user. Hence while using FMO alone, the proposed algorithm behaves like a "SLAM algorithm with unknown correspondence".

If an already observed landmark is being re-observed after making a large traverse in an environment then it is termed as "loop closure". Knowing if the landmark has been previously observed gives a way to re-calibrating the entire trajectory such that the drift in the trajectory can be corrected [48]. If only FMOs are used to calibrate the trajectory, then it is difficult to determine if the observed landmark was previously observed or not. However, if a coded FM is used as a landmark, then this process becomes considerably easy as the location of the coded FM is already known before the trajectory starts. Hence re-observing these landmarks helps in re-calibrating the entire trajectory thereby the total drift in the trajectory can be corrected. Hence, if coded FM is used, then loop-closure can be used to provide a very strong constraint for trajectory estimation.

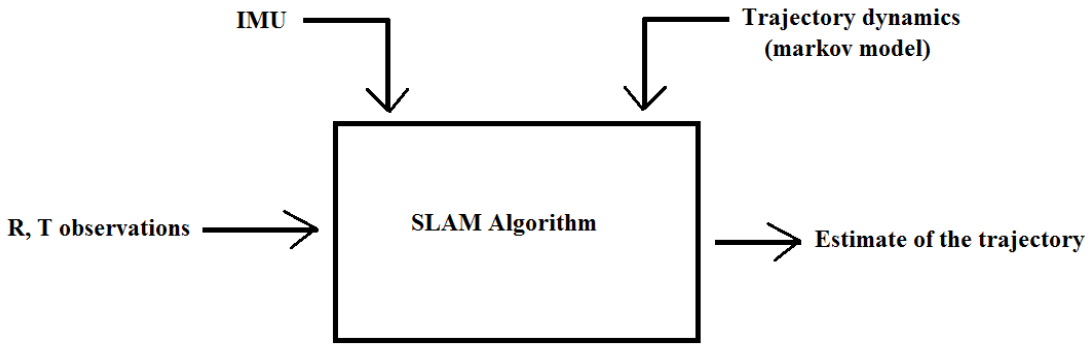


Figure 3.4: Block diagram of the overall SLAM algorithm with camera observables integrated with IMU to estimate the final trajectory

Figure 3.4 illustrates the block diagram of the overall SLAM algorithm that can be used to integrate the data trajectory estimates obtained by the camera observables and IMU. Both cameras and IMU provide good trajectory estimates individually but have their own set of problems. Cameras find it very difficult to deal with occlusion while IMUs suffer from dynamic drift. However both of these sensors combined together can complement each other to provide the best trajectory estimates. This was one of the central ideas discussed in [18]. This is another future work as a consequence of this research.

Chapter 4

Experimental Validation

This chapter is dedicated to validate and provide results for the CV algorithm, proposed in this thesis. The algorithm is tested in its intended environment .i.e. an indoor facility. As it will be shown in this chapter, the CV algorithm has been tested in real-time scenarios and the corresponding rotation and translation is calculated. The primary application setting behind this research is a mobile handset (HND). Hence, a modest 1.3 mega pixel camera has been used for all validations. This chapter will also provide experimental validation which proves that HT is indeed a potential scheme for real-time FMO processing. Since the camera used for all the experiments remains the same, the camera specifications are described in this introduction. Any setup that is exclusive to an experiment will be discussed before the corresponding experimental results.

The AGAMA V-1325R camera shown in figure 4.1 contains a 2M pixel CMOS image sensor. The camera runs a software interpolation algorithm to store up to 8M pixel still images in JPEG format. The camera also captures a 1.3M pixel resolution video with a frame aspect ratio of 5:4 at a frequency of 9 frames per second. The camera can capture video in the WMV format and can be interfaced to a 32 or 64 bit computer using the USB 2.0 interface provided. Since the current application deals with pedestrian navigation, a high frame rate is desirable. Hence the camera is employed in VGA camera mode, which can capture video frames at a frequency of 30 FPS. It is important to note that higher resolution will give finer details in the image leading to better FM recognition. However, this paves a way for more room for noise, demanding rigorous pre-processing than what is currently being used. Hence there is always a trade-off associated with using better resolution cameras. A 1.3M pixel camera was employed in this research considering the fact that a 1.3M pixel resolution is common with most modern day mobile handsets. In the experiments given below, the trajectory that is estimated from the experiments is

termed as ‘true trajectory’. The trajectory predicted using apriori information about the motion is termed as ‘predicted trajectory’.

Figure 4.1: AGAMA V-1325R with a 2MP CMOS image sensor

4.1 BACK PROJECTION – EXPERIMENT TRIALS

In order to evaluate the performance of the algorithm in various types of motion, the algorithm is tested with test cases of both random motion and deterministic motion. This experiment is intended to estimate the error for a random motion of the camera. Since a single camera is employed to perform indoor navigation whose true trajectory is a random motion with 6 DOF, there is no real means to obtain the estimated trajectory in order to compare it with the true trajectory and obtain an error estimate. Hence, the method of back projection is employed to verify the accuracy of the proposed algorithm. Figure 4.2 shows the setup used.

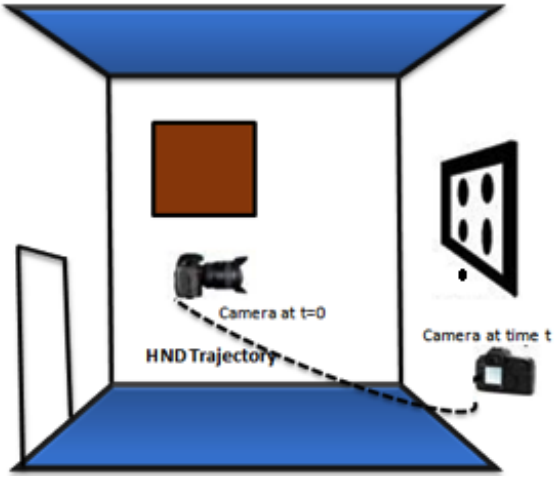


Figure 4.2: Diagram showing 6 DOF Experiment setup

The camera is moved from the first position at t=0 to the current position at time ‘t’ in a random fashion keeping the FM in its FOV. Based on the camera observable of the FM at each point, the perspective transformation is extracted, which gives the corresponding differential rotation and translation vectors for every image frame.

Using these differential rotation and translation vectors, an inverse mapping was performed at every time instance. Figure 4.3 shows the camera observables using a test marker followed by the Figure 4.4 that shows the inverse perspective transformation of the observables of Figure 4.3. This is used to calculate the back projection error. The intensity of the rest of the image in Figure 4.4 is equated to zero to have a better view of the inverse perspective mapping.

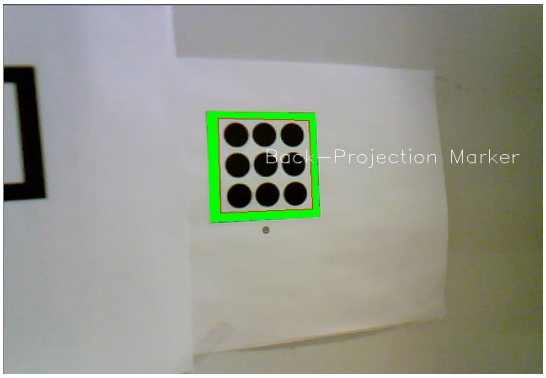


Figure 4.3: Camera observable of a test FM

Figure 4.4: Inverse perspective transformation to calculate the back projection error

In order to evaluate the performance in a random motion, the method of back-projection is applied. In this method, a certain interest point is considered outside the FM. At every point along the trajectory, the inverse perspective transform of the marker is performed making the interest point to be back-projected near the marker. Since the true position of this interest point is already known, the deviation in the position of this interest point after inverse perspective

transform determines the error introduced by the algorithm. The only assumption made is that the FM is stationary. The above mentioned procedure is illustrated in figure 4.5. The dotted lines represent the FOV of the camera.

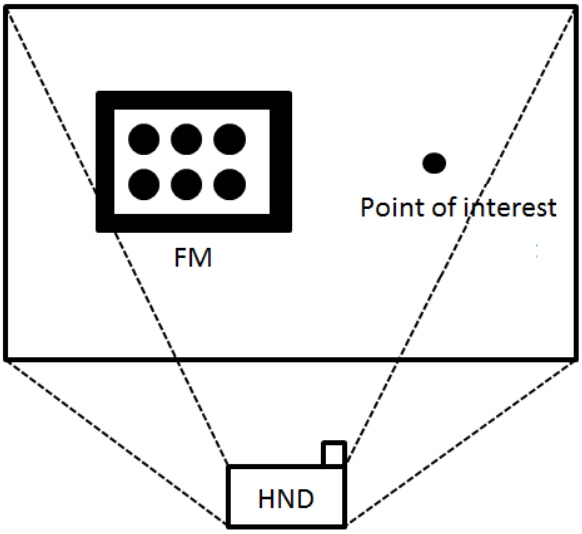


Figure 4.5: Block diagram of experiment setup to estimate back projection error

One such point of interest is utilized to generate the 3D histogram to estimate the back-projection error which is shown in figure 4.6. The X and the Y axis represent the marker dimensions in centimeters. The Z-axis represents the density of the back projection of the points of interest. As already mentioned, the point of interest should be in the neighbourhood of the FM so that it is captured in every frame captured by the camera. Hence a point is chosen at a distance of 2 cm from the edge of the marker or 6 cm from the center of the marker. Once the point was chosen in the FOV of the camera, at every frame an inverse perspective transformation was applied and the interest point was made to back-project on the original calibration position. The standard deviation in the X coordinates of the back projection point was found to be 0.15 cm. The standard deviation in the Y coordinates was found to be 0.159 cm. This result implies that the trajectory estimated in a least squares sense varies from the actual trajectory by a little more than a millimeter. The high accuracy of the proposed algorithm is reflected by the standard deviation of the back projection point being as low as 1.5 mm.

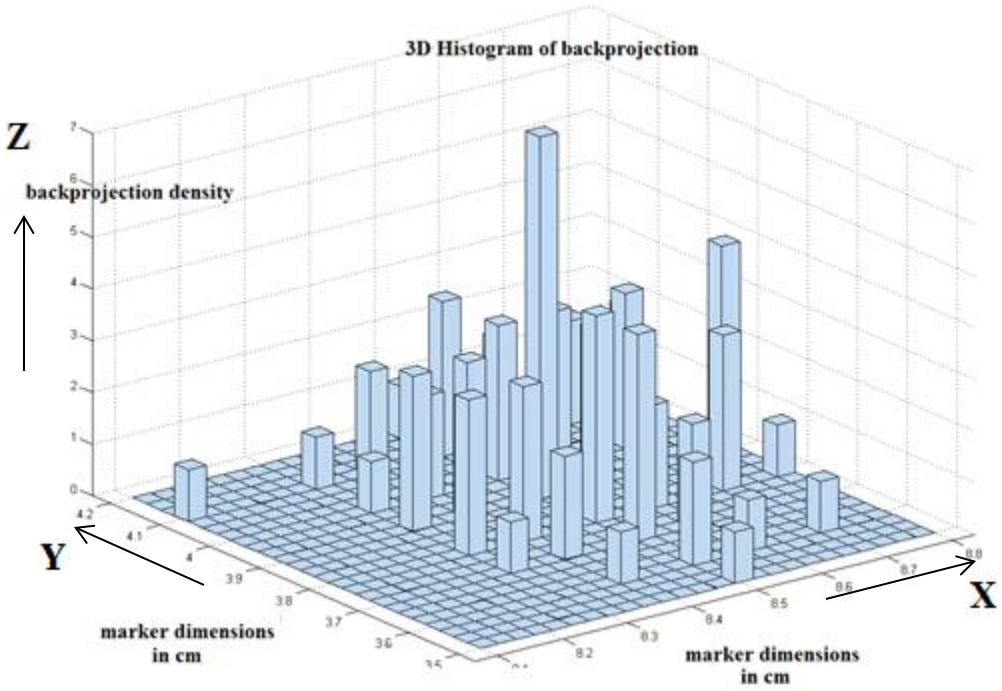


Figure 4.6: 3D histogram of back projection of points of interest

4.2 KNOWN TRAJECTORY - TURN TABLE EXPERIMENT TRIALS

Apart from generating the histogram to verify the accuracy of the proposed method, a second set of experiments was performed to calculate the total absolute RMS error in case of a known trajectory. The experimental setup of the same is given below.

Figure 4.7: Newmark systems NSC-1 series motion controller

Newmark systems NSC-1 series motion controller is a single axis machine control system. It utilizes a 32-bit microprocessor to control the trajectory profile, acceleration, velocity and deceleration that can be controlled using the software provided with the device. It can also be programmed using C or Matlab to pass ASCII command strings to the controller through the RS-232 port.

Newmark systems RT-5 is a 2 inch thick motorized rotary stage which is made to interface with a NEMA 17 stepper motor. The rotary stage can bear a maximum weight of 25kg and can attain a maximum speed of $25^{o}/\sec$.

Figure 4.8: Newmark systems RT-5 Rotary stage

The AGAMA V-1325R camera was mounted on the Newmark systems RT-5 motorized rotary stage, at a distance of 46.3 cm from the axis of the motor. An arbitrary distance of 46.3 cm was chosen and can be extended or reduced to any distance provided the camera is able to extract meaningful information from the FM. The motor was rotated at an angular velocity of 0.3 degrees/sec. The experimental setup is shown in figure 4.9.

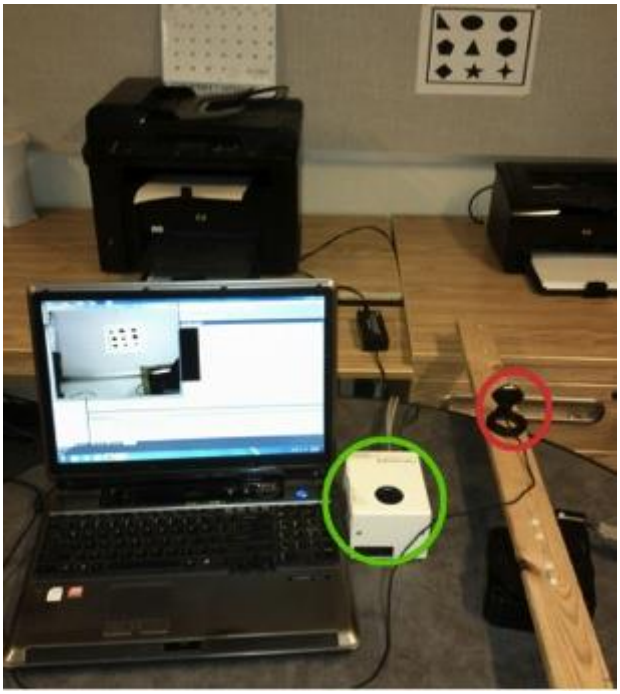


Figure 4.9: Turntable Experimental setup

The AGAMA V-1325R camera is shown in red which is mounted on Newmark systems RT-5, observing a stationary FM on the wall. The Newmark systems NSC-1 series motion controller, shown in green can be controlled by a 64 bit laptop computer.

In order to calculate the RMS error, the predicted trajectory vs. the true trajectory was plotted for a small spatial interval shown in fig. 4.10. The experimental data is represented in the form of red dots, each of which represents the centroid of the tracked FM to estimate the trajectory in a least square sense. The trajectory of this experimental data is calculated in a least square sense and is represented by the blue curve. The thick green curve represents the predicted trajectory. Table 1 gives some sample values of the RMS error.

Table.2. Sample RMS Error values for turntable experiment

Frame number	**Error in cm**
50	1.01
200	0.20
255	0.01
300	0.06
400	0.73

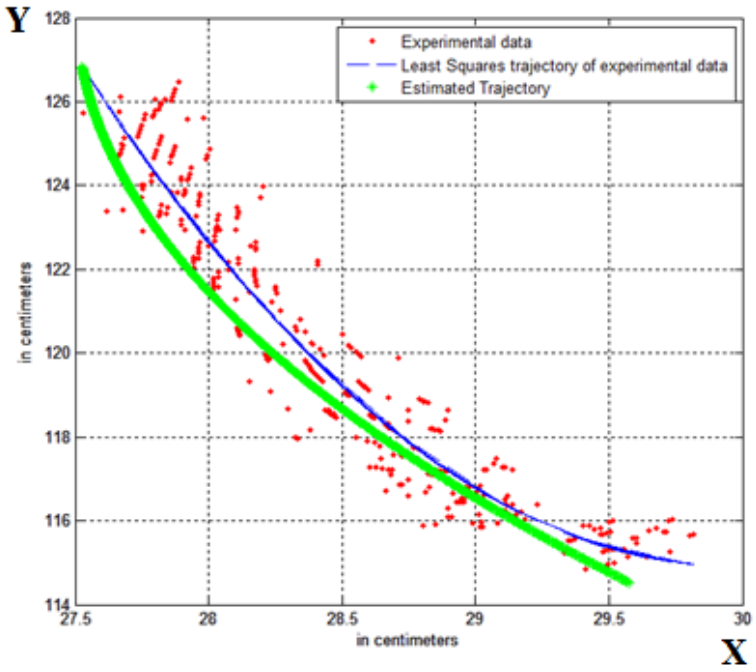


Figure 4.10: Plot of the predicted data vs. experimental data and least square trajectory

A total root mean square error of 0.74 cm was achieved. Also an average processing time of 42.2ms per frame was achieved. The individual rotation matrices for each processed frame were also estimated. The roll, pitch and yaw angles, represented by α, β and γ in chapter 3 have also been calculated. The average value of yaw, which was the primary angle of rotation for the current experiments, was found to be 0.73 degrees. The current sets of experiment are performed pertaining to deterministic motion to calculate the RMS error. However it has already been demonstrated in section 4.1 that this algorithm is not a limiting case and can be applied to a full 6 DOF motion.

4.3 Transition from FM to FMO – Experiment Trials

In a real-world scenario, the proposed approach will not always observe a single marker. It might sometimes encounter an FM or an FMO as it progresses in its trajectory. In order to verify the transition process between two markers (FM or FMO) an experiment was performed. In this experiment the HND was made to move in a trajectory such that an FM appears initially in the FOV of the camera. As the trajectory progresses, there is a transition in the FOV of the HND.

The coded FM gradually leaves the FOV and an FMO progressively starts to appear. The FMO as seen in the previous experiments is assumed to be a rectangle. As it will be shown in subsequent sections, estimation of rectangular objects can be performed using HT. For this experiment, an FM with no code is taken as an FMO. Once the FM is completely outside the FOV of the camera, the trajectory estimation is entirely based on the observables from the FMO. Hence the new trajectory differential updates are based on the FMO, relative to the FM. The transition from an FM to an FMO is shown in figure 4.11

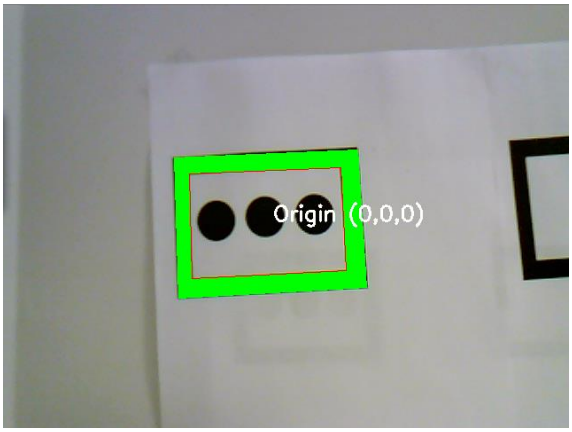


Figure 4.11a

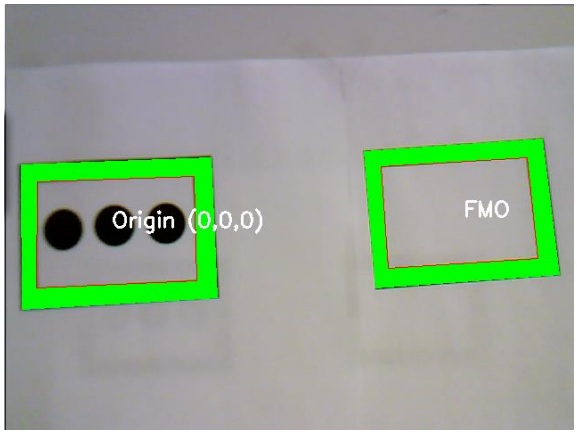


Figure 4.11b

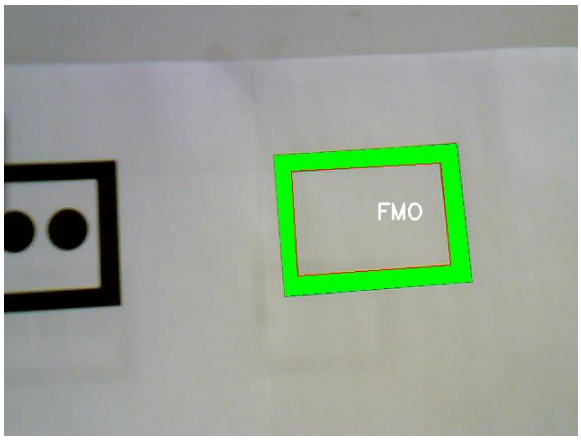


Figure 4.11c

Figure 4.11 Transition in the FOV of the camera from FM to FMO. Figure 4.11a: Camera observing the FM in the FOV. Figure 4.11b: Camera observing both the FM and the FMO in the FOV. Figure 4.11c: Camera observing only the FMO in the FOV.

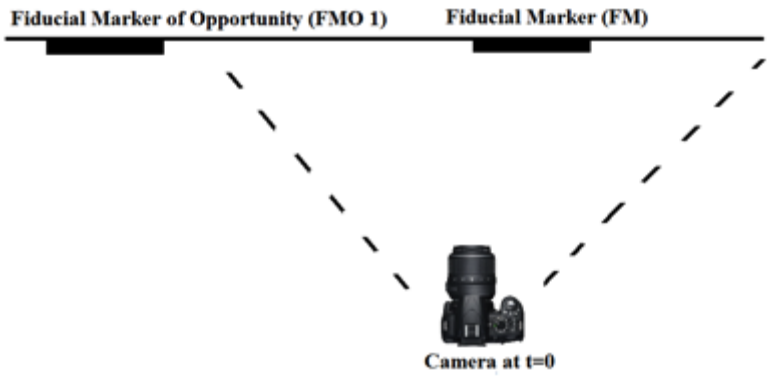


Figure 4.12a

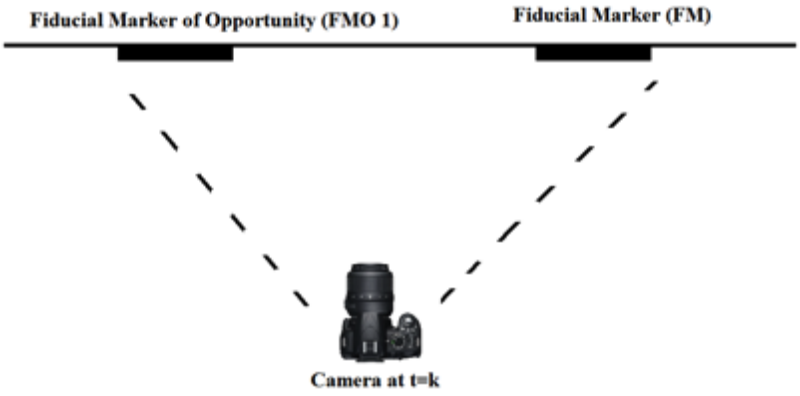


Figure 4.12b

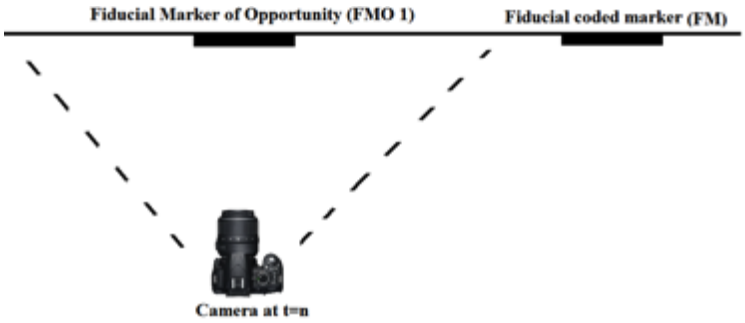


Figure 4.12c

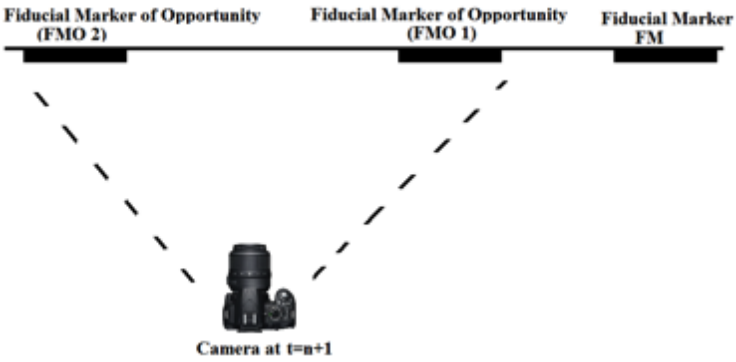


Figure 4.12d

Figure 4.12: Figure showing the transition between markers as observed by the camera

An illustration for the test case for the proposed algorithm in a transition between markers is shown in Figure 4.12. As shown in Figure 4.12a, the camera first observes a known FM in the camera's FOV. The perspective transformation and the coded information from the FM are extracted. This provides the first set of R and T vectors which can be employed in the very next frame to estimate differential R and T values.

In Figure 4.12b, the camera is moved along an unknown trajectory but the FM is still contained in the FOV. Hence the initial segment of the HND trajectory is accurately calculated. As the HND's location at this point is accurate, the perspective matrix of the new FMO can be estimated. The FMO is checked to verify if it is still in a rectangular shape as described before. Note that the HND is assumed to move slowly relative to the frame rate of the camera such that multiple views of the simultaneous observation of the FM and FMO are obtained. In Figure 4.12c, the camera has now moved such that only the FMO is in the FOV of the camera. However, as the transformation matrix and location of the FMO are estimated, the HND trajectory can still be estimated. In Figure 4.12d, the HND trajectory moves further to include the first FMO and a new additional FMO (FMO 2) in the FOV of the camera. The algorithm then proceeds to determine the perspective transform and the location of the second FMO based on the first FMO. This process continues indefinitely until an FM of known location is recognized. At this point the overall location trajectory of the HND is recalculated.

Figure 4.13 shows the trajectory of the above mentioned method for the setup described in figure [4.12a- 4.12c]. The points on the left correspond to the trajectory estimates calculated by the FM

observables. The points starting almost in the middle of the plot correspond to the trajectory estimates calculated from the FMO.

If the trajectory progresses such that a second FM was observed after observing the first FM then the location estimation can be immediately corrected such that the true trajectory estimate can be found. This is the case of trajectory estimation using two independent map points, both of which are precisely known in advance. This provides a better trajectory estimate than the case employing an FM and an FMO alone because more corrections of the trajectory are performed in this case.

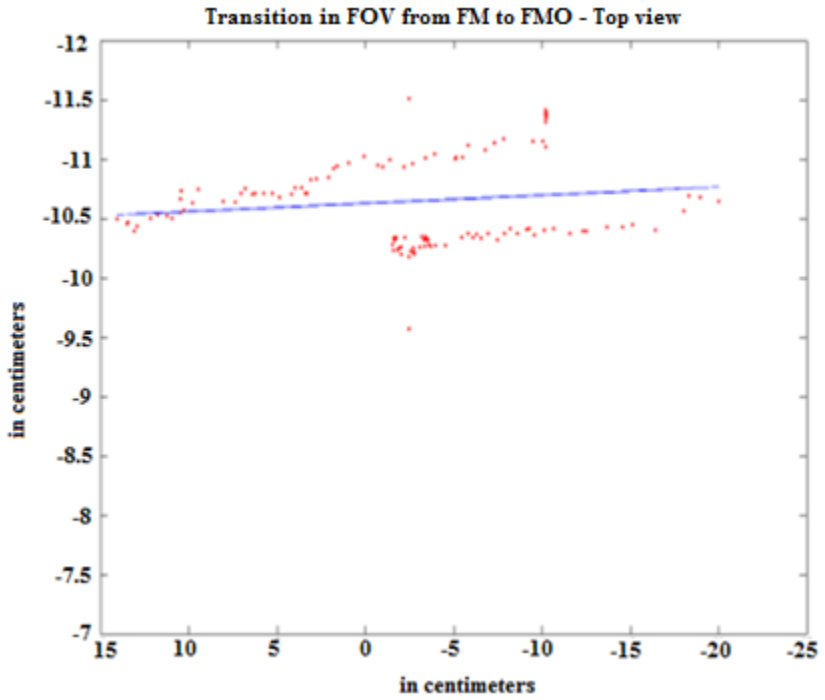


Figure 4.13: plot of the trajectory in case of a transition in the field of view from FM to FMO

It can be observed that for a small interval, the trajectory estimation is applied with respect to both the FM and the FMO. RANSAC was employed to reduce the error occurring due to the presence of outliers. The blue line between both set of points represents the least squares fit. The least squares fit establishes a relationship between both set of points and hence can act as a very important constraint in estimating the cumulative trajectory.

It is evident from figure 4.13 that the trajectory is not smooth. This can be due to a number of reasons. To begin with, R and T vector values are an approximation performed by the digital

camera to a certain finite precision. Secondly the camera was being manually pushed with the hand which is not very smooth especially for such a small duration. Finally, the lens distortions which have been accounted for, in section 2.6 are also an approximation. All of these issues together lead to a cumulative drift in the trajectory after the introduction of each new FMO.

A more accurate trajectory can be obtained if a coded FM is present at the end of the trajectory or if multiple instances of such coded FMs are present along the trajectory. The setup for the same and the camera trajectory direction is shown in figure 4.14.

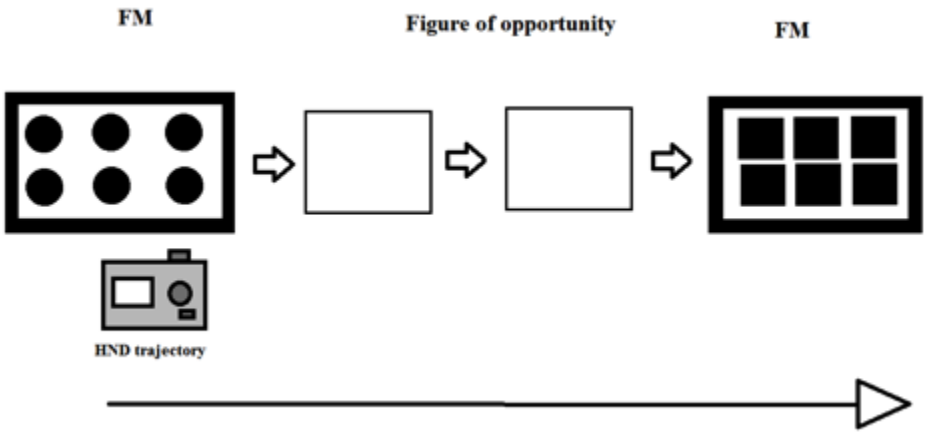


Figure 4.14: Block diagram of experimental setup to minimize the drift

4.4 TRAJECTORY ESTIMATION FOR LONG RANGE – EXPERIMENT TRIALS

In this experiment, the algorithm is tested for robustness for long range trajectories. The experimental setup is shown in figure 4.15. It consists of a webcam quality camera connected to a 64 bit laptop computer, both mounted on a trolley cart. The setup is similar to figure 4.14 except that there is no FM to minimize the drift. The experiment starts with the camera observing the marker as an anchor to observe the initial R(t) and T(t). The trolley is made to undergo a translation in a direction perpendicular to the plane of the marker. The camera undergoes translation in a direction which is shown below in figure 4.15 as 'Trajectory Direction'. The camera then begins to observe the first fiducial marker of opportunity (FMO-1) in order to retrieve its R(t) and T(t) with respect to its previous frames. This is done such that the trajectory is now calibrated with respect to only FMO-1 while the coordinate axis is calibrated with respect to the marker (that acts as an anchor). The trolley cart continues its trajectory in the

aforementioned direction and the camera observes the second fiducial marker of opportunity (FMO-2) and the same process is repeated.

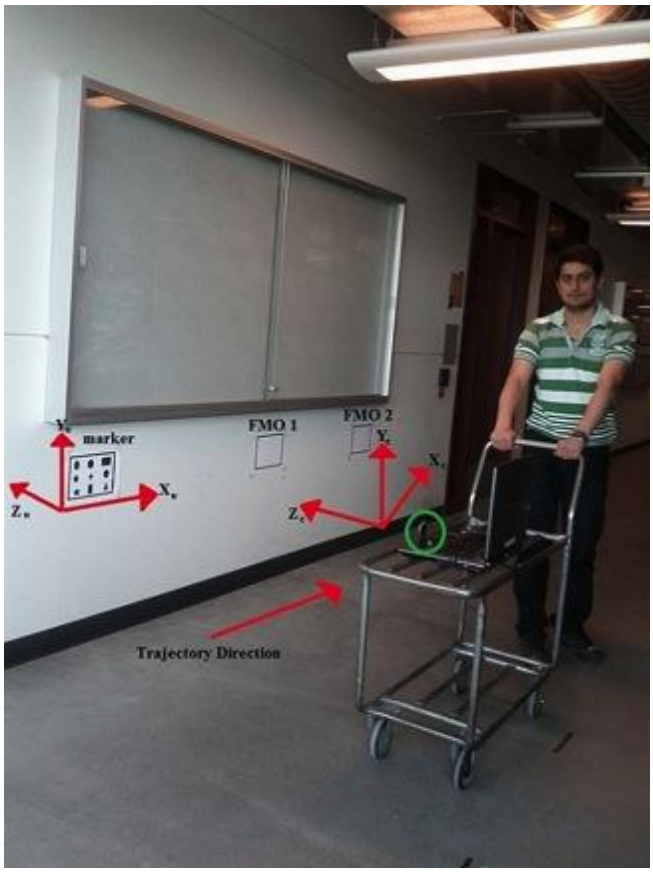


Figure 4.15: Experimental setup long range trajectory estimation experiment

Figure 4.16a shows the trajectory of the above-mentioned method. The points in black correspond to the trajectory estimates calculated by the FM observables. The points in green denote the trajectory estimates calculated corresponding to FMO-1. The trajectory estimates with respect to FMO-2 is denoted by the red points. The blue line in the center which connects all the three trajectories denotes the least squares fit of the overall trajectory based on the assumption that the camera is moved in a straight line trajectory. The addition of each FMO will introduce a minor drift in the trajectory due to reasons mentioned in section 4.3. If now a fiducial marker is introduced in the field of view then it will act as an anchor node to offset the drift.

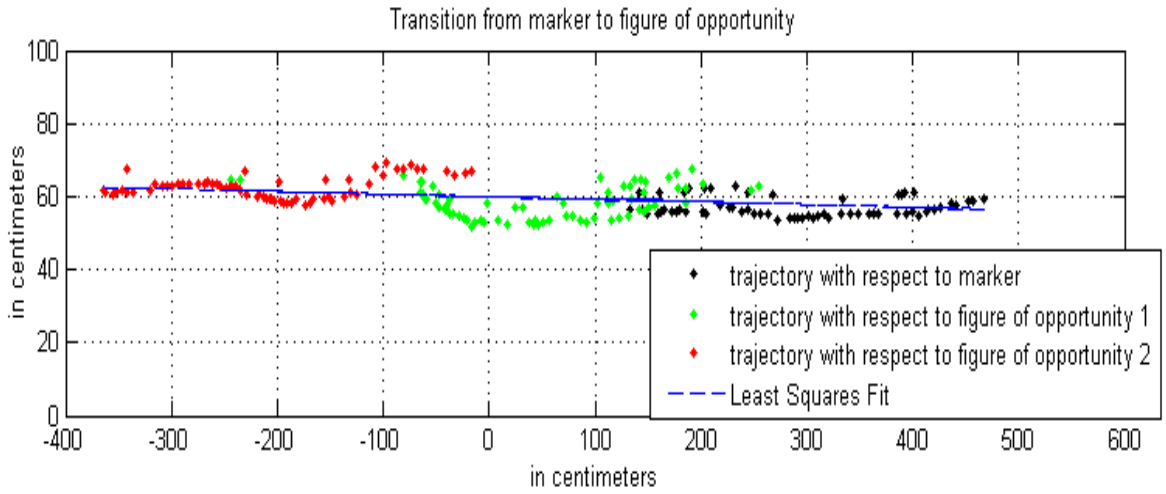


Figure 4.16a: Plot of the trajectory generated from the long range trajectory estimation experiment

The trajectory in figure 4.16a is not very smooth as there is no spatial filtering applied to the trajectory. That is, the trajectory plotted is a set of independent estimations. Another observation from the figure is the presence of more variations in the long range trajectory as compared to the short range trajectory shown in figure 4.12. This might be attributed to the fact that the long range trajectory was moving on a floor which is not perfectly flat. Since the short range trajectory was calibrated on the Newmark systems RT-5 rotary stage moving at a constant angular velocity, it had a much smoother operation. The long range trajectory was calibrated by a camera sitting on a cart that was pushed manually. The presence of imperfect wheels of the cart only exacerbates the problem all the more.

Since a trajectory estimate has been found for both short range and long range trajectory, a comparison between the two for better understanding can now be made. A comparison between the long range trajectory in figure 4.16a and its short range counterpart in figure 4.13 reveals that as the range of operation increases, the error introduced by the introduction of each new FMO increases. However the error still remains tolerable and does not drift away. A few points however were found to be outliers and were rejected by RANSAC outlier rejection technique discussed in section 2.7. Figure 4.16b-4.16d shows some more examples of the same experiment in different environment conditions.

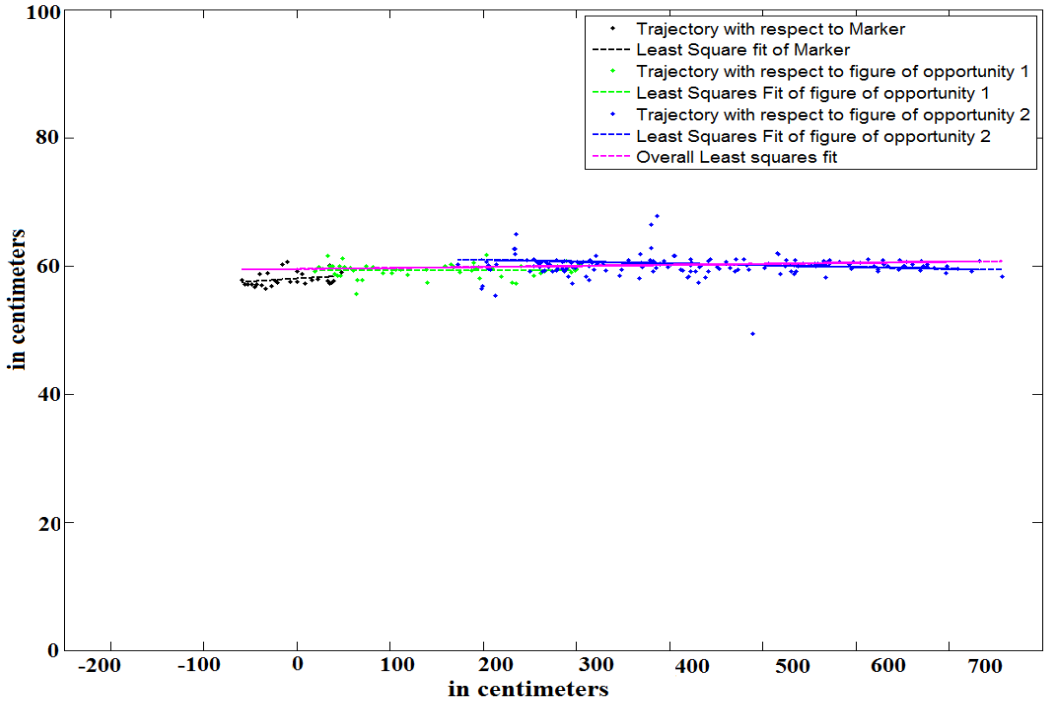


Figure 4.16b: Plot of the trajectory generated from the long range trajectory estimation experiment with the trolley at a distance of 1.5m from the marker wall

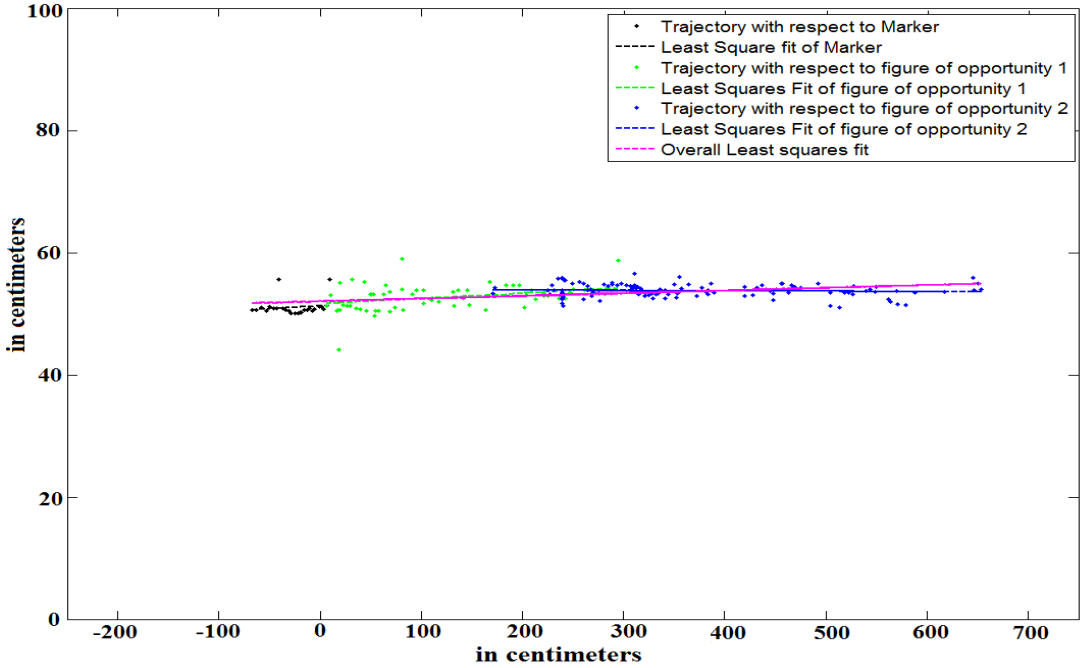


Figure 4.16c: Plot of the trajectory generated from the long range trajectory estimation experiment with the trolley at a distance of 1.5m from the marker wall with different marker sizes

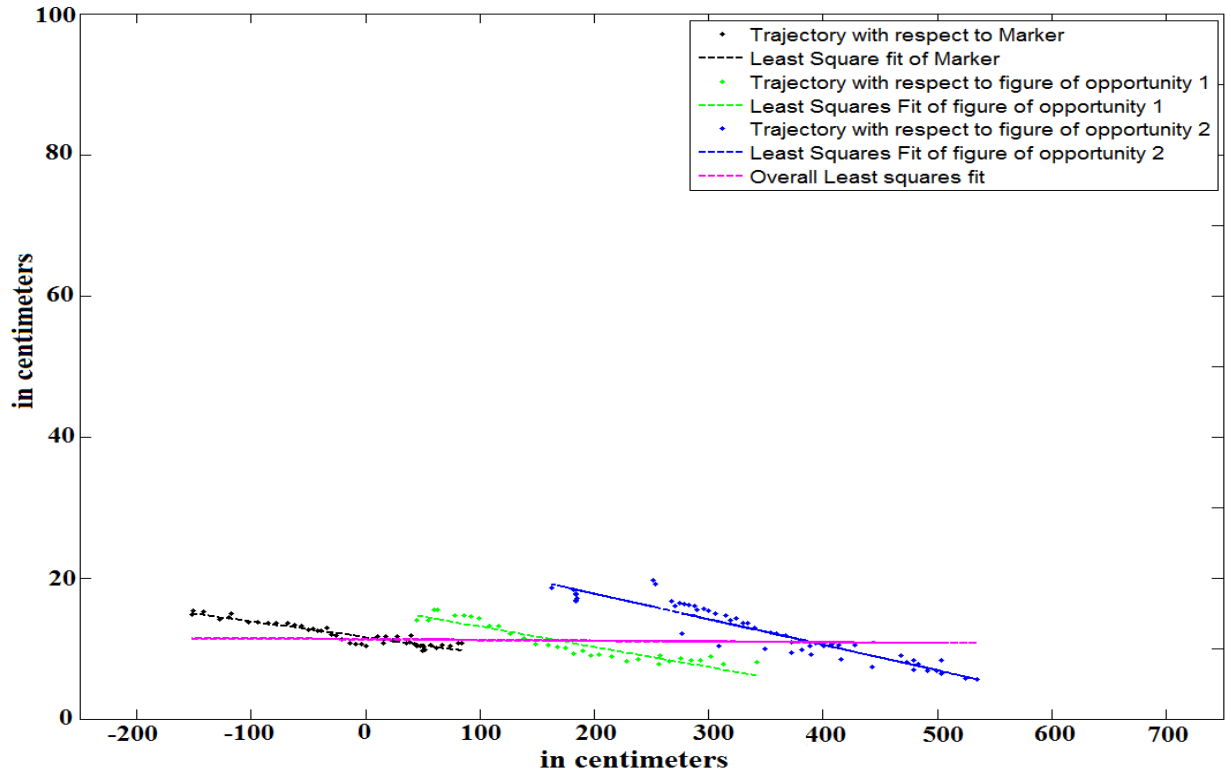


Figure 4.16d: Plot of the trajectory generated from the long range trajectory estimation experiment with the trolley at a distance of 2.5m from the marker wall with different marker sizes

Figure 4.16: Plot of the trajectory generated from the long range trajectory estimation experiment

4.16b shows the plot of the long range trajectory of the camera plotted with markers of same dimensions. Similar to figure 4.16a, the camera was mounted at a distance of 1.5m from the markers along the z-axis. Any distance less than this will scale the marker such that a single marker occupies the entire FOV of the camera making it impossible to accommodate two markers in the FOV at any given time. The individual trajectories generated with the marker, FMO1 and FMO2 along with their least square trajectories are shown. The overall least square fit is also shown. An assumption made in all these experiments is that the markers are mounted on a flat surface.

4.16c shows the plot of the long range trajectory of the camera plotted with markers of different dimensions. It can be seen from figure 4.16c that the trajectories generated by this experiment is

almost similar to the trajectory shown in figure 4.16b. This is because the data measured is dependent on the center of the markers which remains almost the same even if different sized markers are used.

4.16d shows the plot of the long range trajectory of the camera plotted with markers of different dimensions. The camera was mounted at a distance of 2.5m from the markers along the z-axis. As mentioned in section 3.2, for a marker to be considered a genuine marker, it should be composed of an outer contour and an inner contour. It was observed that if the camera was placed at any greater distance than 2.5m, then the camera was unable to differentiate between the outer contour and inner contour of the marker. Hence the marker was completely rejected by the algorithm. Since the camera was placed at a greater distance from the marker, the FOV of the camera was able to accommodate more objects in its FOV. Hence the trajectory is scaled along the y-axis. Also unlike figure 4.16a-4.16c, the individual trajectories in figure 4.16d have a negative slope. This is attributed to the fact that the trolley was not perfectly parallel to the wall but was slowly approaching the wall on which the markers were mounted. The distance between two consecutive trajectories is due to the cumulative drift that was mentioned in section 4.3.

4.5 FIDUCIAL MARKER OF OPPORTUNITY

Figure 1.5 and Figure 4.17 are examples of FMO detection using the proposed HT and FM detection algorithm respectively. It is important to note that both of these contours are stationary rectangular figures lying on a plane. Even though the proposed algorithm detects FMO in figure 4.17, employing useful camera observables in the form of R and T directly from the algorithm output is usually difficult. This is because the presence of a large number of rectangular FMO in the surroundings makes it difficult for robust estimation of the same FMO in every frame. Apart from the problem of estimating correspondence between FMOs in different frames, another problem encountered by the FM detection algorithm is that it does not detect the FMOs in all consecutive frames. It is for the latter reason that HT is employed. As already mentioned finding correspondence is itself a big problem and has to be dealt with as a separate problem.

As it can be seen from figure 4.17, the detected FMO is in fact a rectangle. Apart from this, no other information is conveyed by the FMO directly. The robust detection of this FMO is performed in all consecutive frames leading to the estimation of the differential values of R and

T. The cumulative sum of these R and T acquires the trajectory relative to the starting point at which the FMO was initially detected. Since there is no real way to estimate this initial point, it is advisable that the algorithm be calibrated initially using coded FM.

Figure 4.17: Detecting a cabinet window as a figure of opportunity

4.5.1 HOMOGRAPHY ESTIMATION FROM HOUGH LINE TRANSFORMS

Hough lines in a scene can be detected similar to figure 2.28. Once these lines are detected, they could be infinitely extended over the entire image such that if the FMO under consideration comprises of a rectangle, then intersection of these infinite Hough lines can lead to the detection of vertices of this rectangular FMO. An example of this was shown in figure 1.5. Once this is done, pairs of 4 points each can be taken from consecutive image frames and a 4 point algorithm can be employed to extract PT. This is the concept employed for homography estimation from rectangular FMO.

The flowchart of the algorithm is given in figure 4.18. The example given in figure 4.19 illustrates this process. Similar to the FM detection algorithm described in chapter 3, the algorithm begins with pre-processing comprising of smoothing and thresholding. This is followed by Hough lines estimation vertices of the FMO which are detected using the method described in the above paragraph. Then, as described, vertices of the FMO from consecutive

image frames are employed to determine perspective transform. In the example given below, the estimation of vertices of the FMO is followed by perspective comparison.

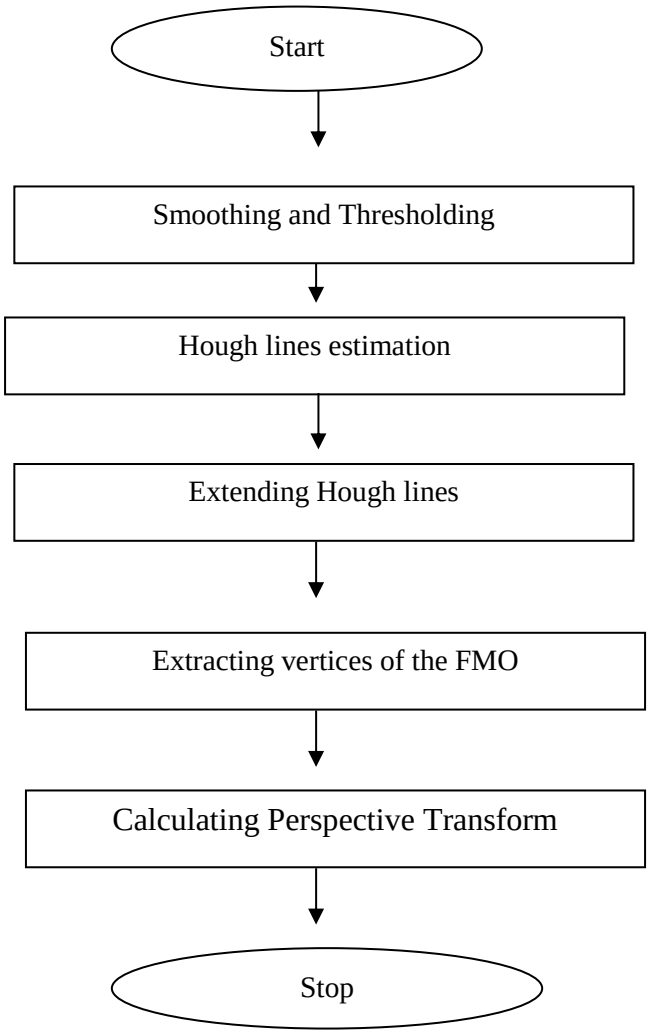
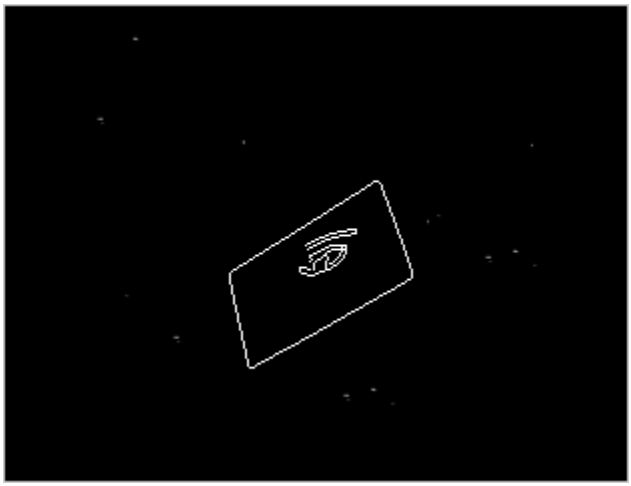


Figure 4.18: Flowchart of Homography estimation using Hough line transform

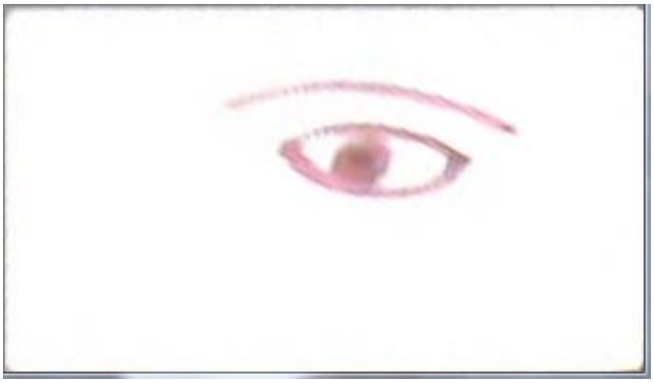
4.19a. Original image under consideration

4.19b. CV output after applying box filter of size 3x3



4.19c. CV output after applying binary threshold of 128 and a maxvalue of 256

4.19d. CV output after extracting vertices of the FMO



4.19e. Homography comparison after inverse perspective transformation

Figure 4.19: Perspective transformation using HT

Figure 4.19a shows the original image under consideration. Figure 4.19b shows the output of the smoothing after applying a box filter of size 3x3. More details about box filtering are given in section 2.3.1. Figure 4.19c shows the output of the FMO after applying a Canny edge detector of λ_{low} = 95 and λ_{high} = 110. Binary thresholding is then applied as per equation 2.13 given in section 2.3.3 with a threshold of 128 and a max value of 256. Figure 4.19d is the processed image after FMO vertices extraction. Figure 4.19e shows the output after applying inverse perspective mapping H^{-1}. The algorithm shown in figure 4.18 was carried out for an image sequence. Testing the same algorithm for a video sequence is performed in section 4.5.3.

4.5.2 HOMOGRAPHY ESTIMATION USING HT FOR IMAGE SEQUENCE

As already mentioned in section 4.5.1, the algorithm was tested for a single image shown in figure 4.19a. Image sequences are good sources to test the functionality of the algorithm. It is however important to note that they do not validate all test cases of the algorithm. It is for this reason that any CV algorithm needs to be tested on a real-time video or a real-time captured video. The essential difference between the two is that the former is captured by a camera in real-time, while the latter is a captured and saved version of the same which is stored on the test computer that is processing the algorithm. The proposed CV algorithm is first tested with 14 image frames and later validated in a larger real-time captured video to test for occlusions.

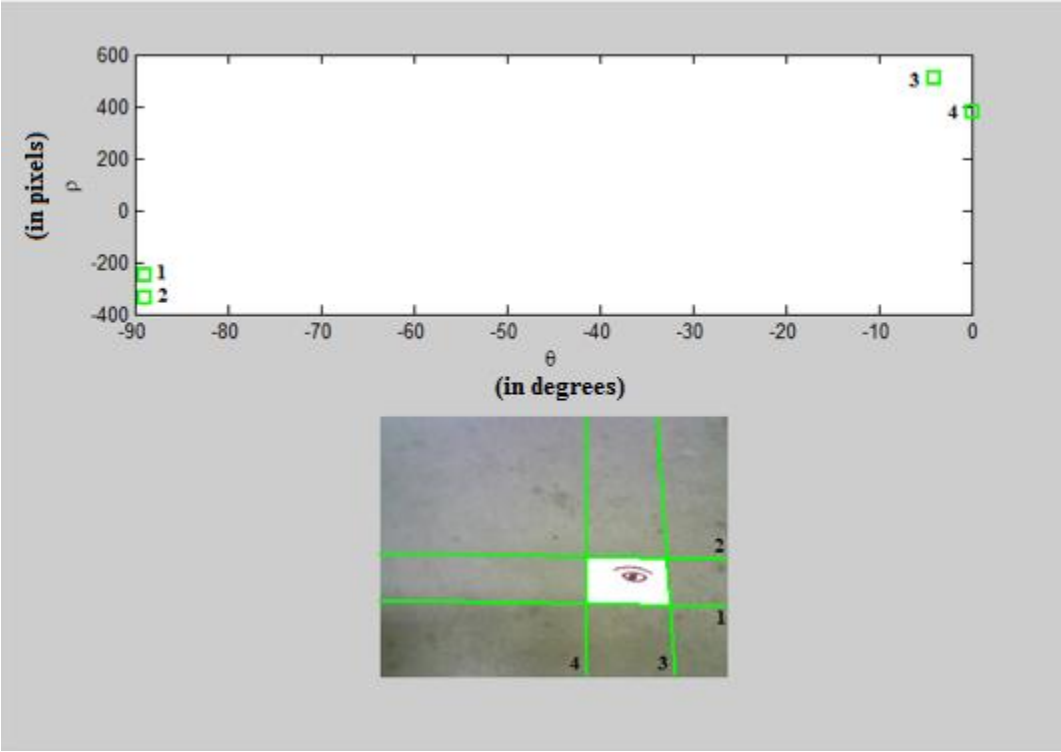


Figure 4.20a: Image frame and corresponding parameter space value

Figure 4.20a represents the first image amongst the 14 images that are used to test the functionality of the algorithm. The 4 edges of the marker are determined using the HT algorithm and are extended across the entire image similar to figure 1.5. As already mentioned, these 4 lines are represented by 4 points in the parameter space which are shown in figure 4.20a. In order to verify if these estimated lines are correct, the set of all four points was taken and re-projected

back on the original image. This is shown in figure 4.20b. It can be seen that the estimated vertices are almost pixel level accurate.

Figure 4.20b: estimated vertices and their re-projection of the original image.

Figure 4.20 Image frame and corresponding parameter space value

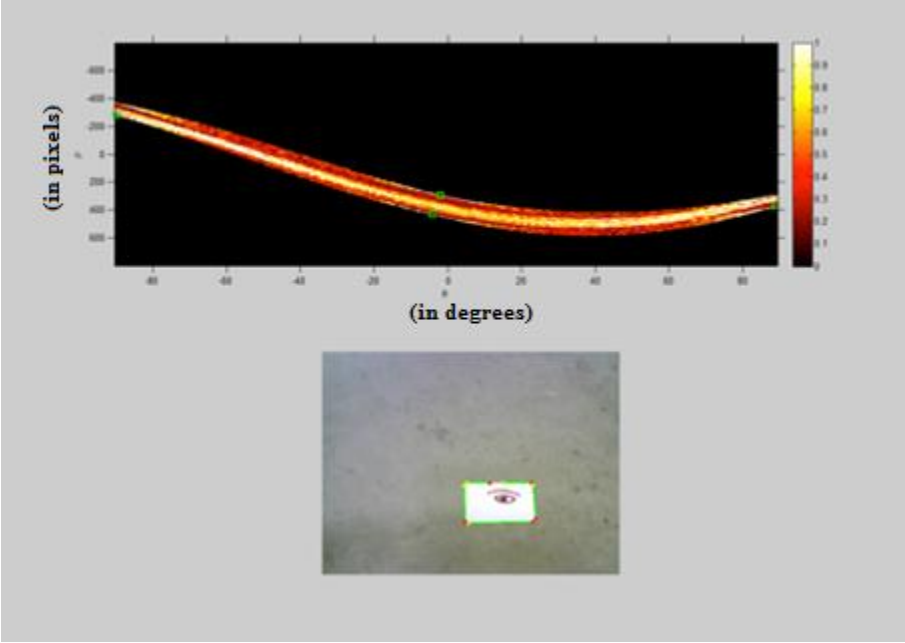


Figure 4.21: Image frame and its corresponding parameter space value- test 2

Figure 4.21 represents frame 4 with a similar true positive case. This example is shown only to depict the HT peak extraction from the HT algorithm.

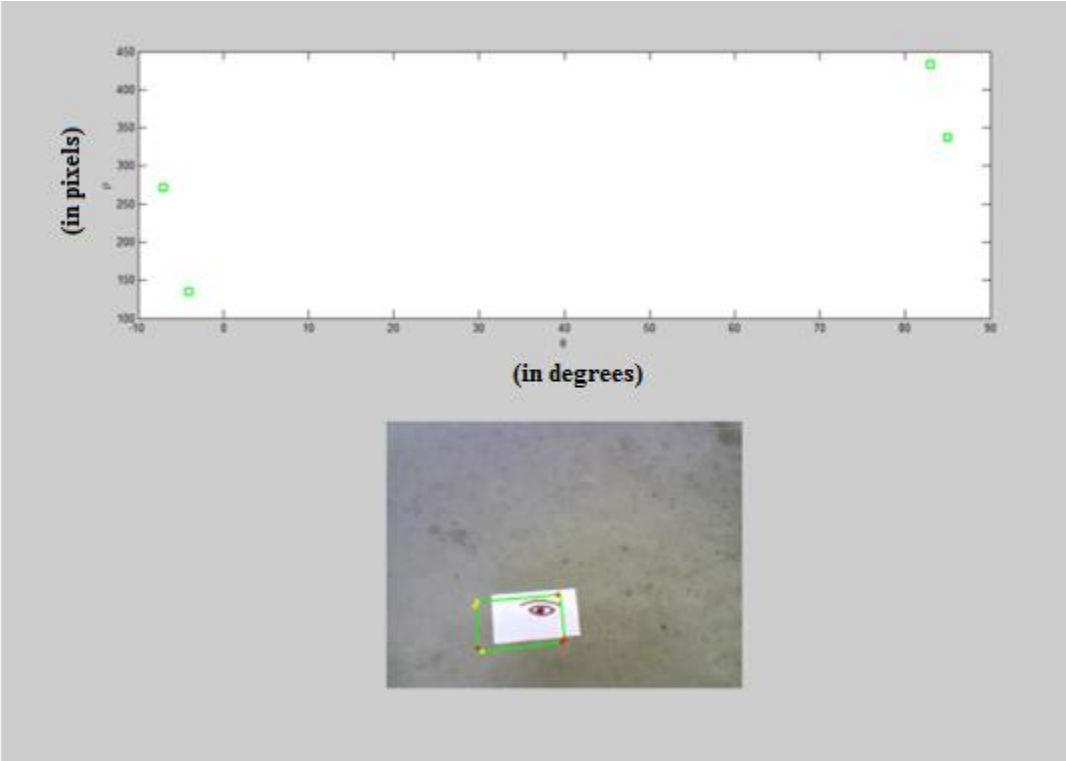


Figure 4.22: HT between 2 consecutive image frames

Figure 4.22 represents the transformation between frame 8 and frame 9. The Hough lines shown in green are estimated from frame 9 and then re-projected on frame 8 to depict the spatial transformation. The parameter space representation of (ρ, θ) is for frame 9.

While calculating the intensity values of the image in order to further process it using the HT algorithm, a Canny edge detector preprocessing was initially employed. It was observed that the Canny edge detector preprocessing is more sensitive to background clutter as compared to the Sobel operator preprocessing. This can be attributed to the fact that the proposed algorithm is itself very sensitive to thresholding. A Canny edge detector, unlike the Sobel operator, has two thresholds and hence operates on a region of threshold instead of a single threshold. Hence the sensitivity of the algorithm in the Canny edge detector is observed to be even greater. Both Canny edge and Sobel detector are described in section 2.3.4. The preprocessing for the experiments in figure 4.20, 4.21 and 4.22 was performed using the Sobel operator.

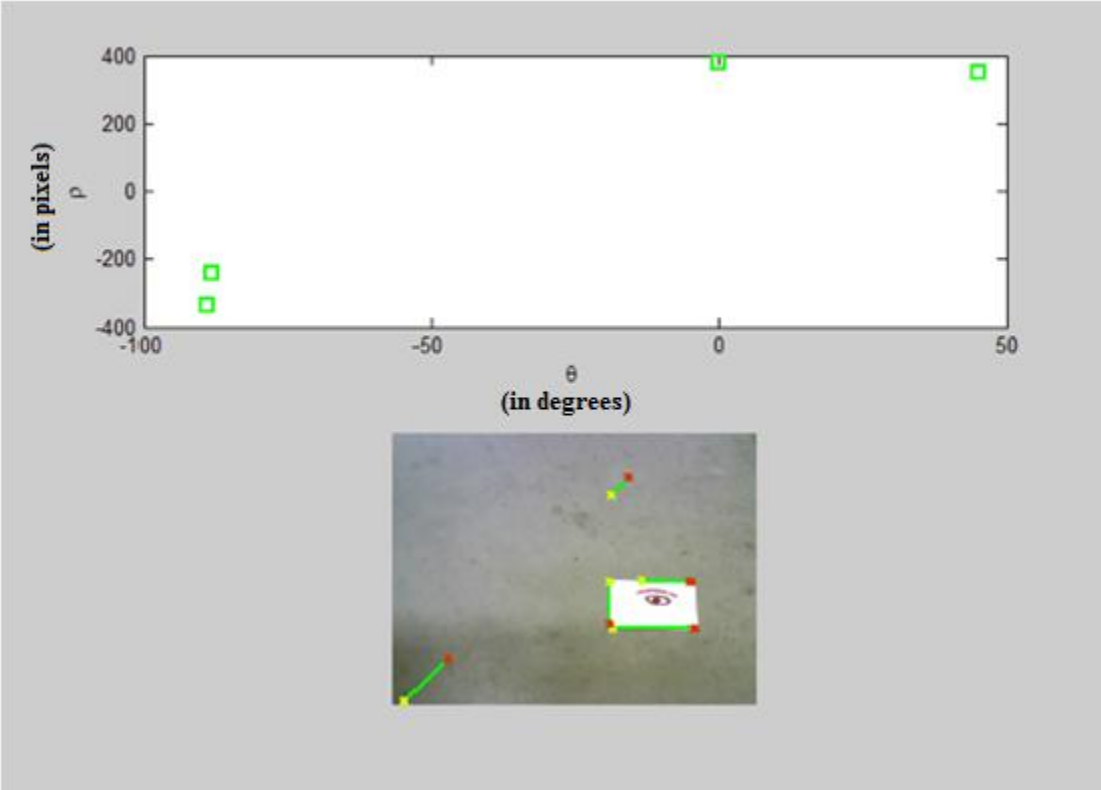


Figure 4.23: Image frame and HT false positives

Figure 4.23 represents the HT algorithm with preprocessing using Canny edge detector. As mentioned earlier, the proposed algorithm when preprocessed with Canny edge detector is found to be more sensitive to background noise than Sobel operator. Also it can be seen that even though five lines are detected in the image frame, only four are represented in parameter space. This is because two lines in background clutter represent the same line at different spatial locations.

Out of the 14 frames used, 11 frames were successfully detected using Sobel operator preprocessing with all 4 edges of the template. Two frames detected only 3 edges while 1 frame detected only 2 edges. This is attributed to the fact that the threshold used cannot have a constant value as the environment conditions like brightness play a significant role in the working of the algorithm. Hence some method of adaptive thresholding scheme can be used in order to make the algorithm work in all possible environmental variations.

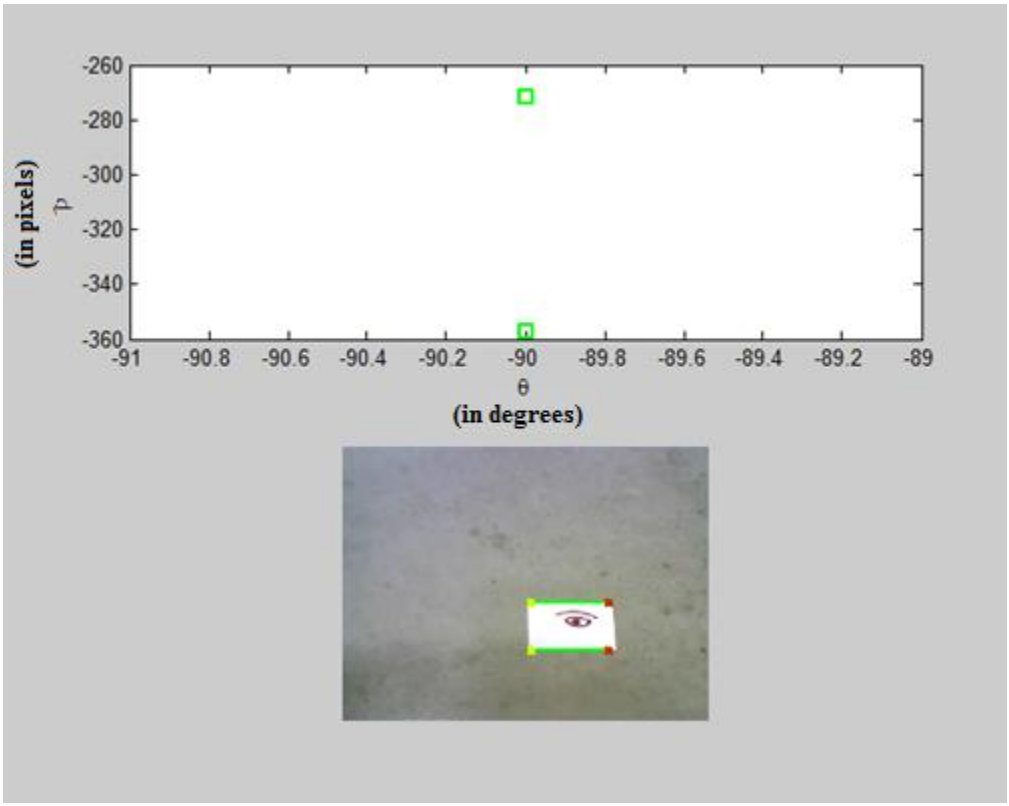


Figure 4.24: Image frame and HT false negative

Two matlab functions namely Hough and Hough peaks were employed to achieve the above results. The surface plot of the HT in parameter space of figure 4.24 is shown in figure 4.25. The x y and z coordinates represents ρ, θ and the density of the HT respectively.

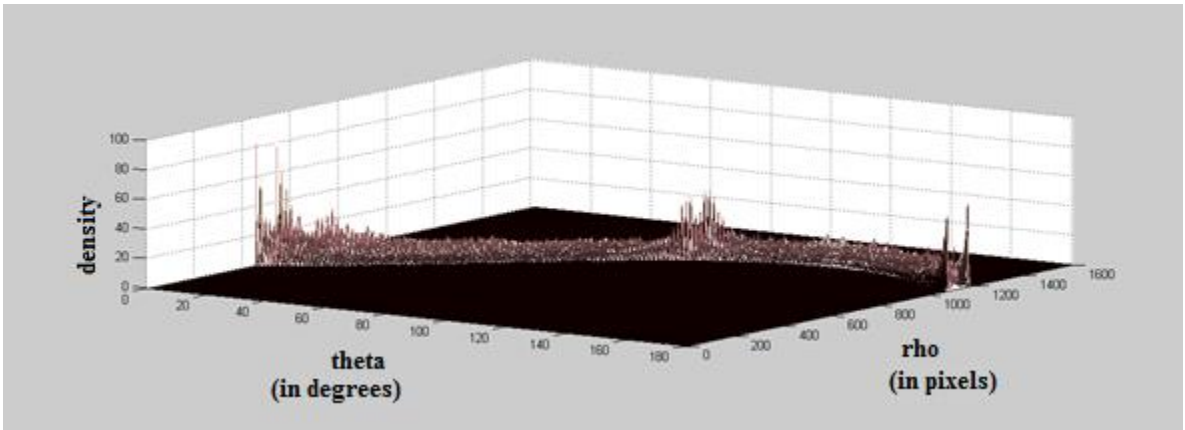


Figure 4.25: Surface plot of HT false negative

It can be seen that at least 6 peaks can be detected in the parameter space with the right thresholds. However, only two of them are detected in the frame even though 4 peaks are being searched for. This is evident from the function call P = houghpeaks (H, 4);

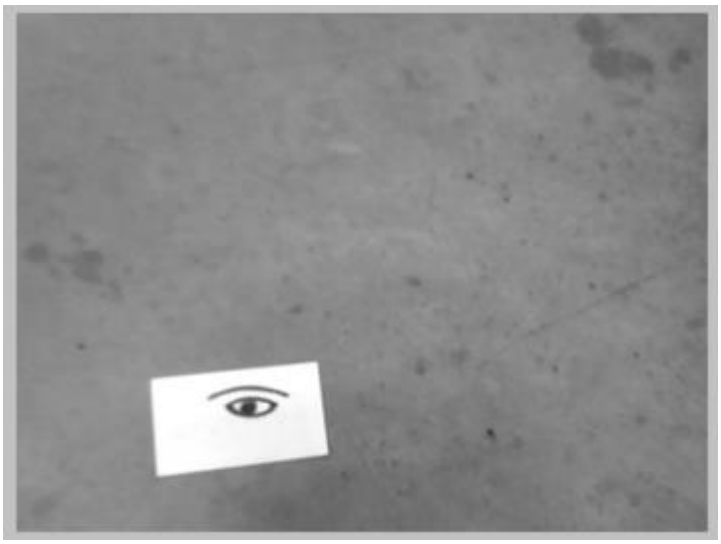


Figure 4.26a: Image frame 10 of the image sequence - FMO under consideration

Figure 4.26b: DCT of image frame 10 considered over spatial frequency of 0 and 0.5 in X and Y directions

As already mentioned, the motivation behind using an image sequence is to verify the behaviour of the algorithm in all possible environmental variations that the FMO undergoes. In the following set of experiments, one of the images (image frame 10) was selected and gaussian noise was applied to it. Figure 4.26a-4.26f shows the test results.

Figure 4.26a shows the image frame 10 of the image sequence. The DCT of this image frame is represented in figure 4.26b. This is scaled such that the displacement along the X and Y directions represent spatial frequencies from 0 to 0.5 with (0,0) (origin) representing the top left corner of the image. Hence it can be seen that most of the image content is concentrated in the top left corner occupying lower frequencies. An important point to consider is that DCT shown in figure 4.26b includes the entire FMO as signal content, while in the current application, the signal represents only the edges of the FMO and not the entire FMO. However it has been shown previously [50] that DCT can be used to extract lines from an image. Hence a future continuation of this research would be to compare the lines detected by HT and DCT. It is expected that this will bring forward an analysis to compare if both HT and DCT can estimate corresponding lines. If so the problem of correspondence finding discussed in section 4.5 can be solved.

All the test results explained so far have been employed directly without adding any additional noise. In this set of experiments, a gaussian noise of zero mean and variance 0.02 pixel units is added to the input image. The corresponding output is shown in figure 4.26c.

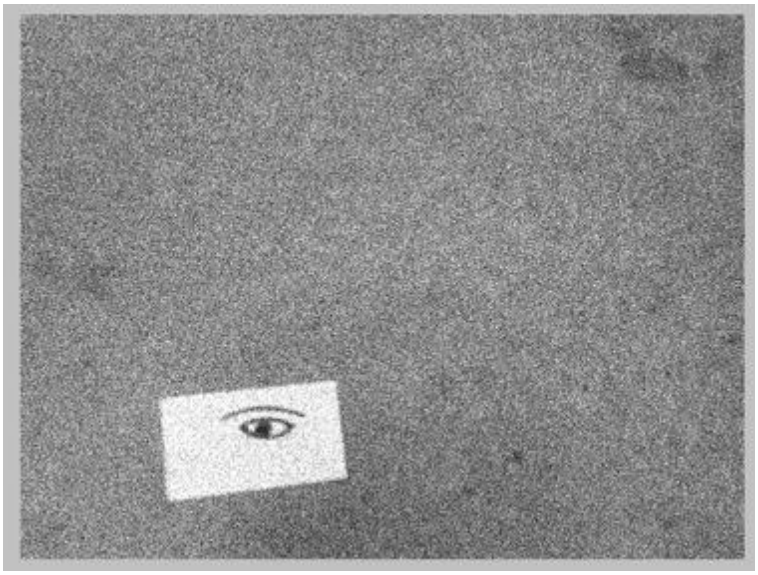


Figure 4.26c: Image frame 10 of the image sequence after adding gaussian noise of zero mean and variance 0.02 pixel units

The presence of additional noise makes the smoothing stage all the more important. A gaussian smoothing of kernel size 17 x 17 pixel units is applied and a standard deviation of 5 pixel units is

selected. The output of this gaussian smoothing is shown in figure 4.26d. It can be seen that the edges are starting to blur leading to a difficulty in edge detection.

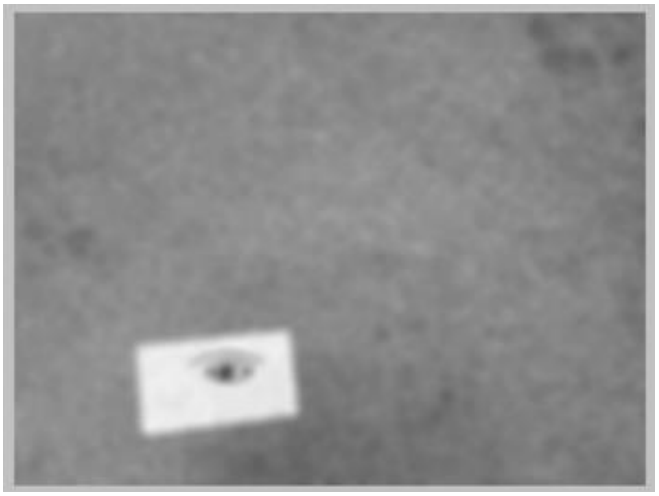


Figure 4.26d: Output after smoothing with kernel size 17x17 and standard deviation of 5 pixel units.

Under such a scenario, this image is preprocessed with Sobel operator and the proposed algorithm is then applied. The result of this method is shown in figure 4.26e.

Canny edge detector preprocessing was also performed on figure 4.26d.The output of this was used as the input of the proposed algorithm. The results for this method are given in figure 4.26f

Similar to the previous experiments, it can be observed that the algorithm is sensitive to background noise and hence Sobel operator offers better results than the Canny edge detector.

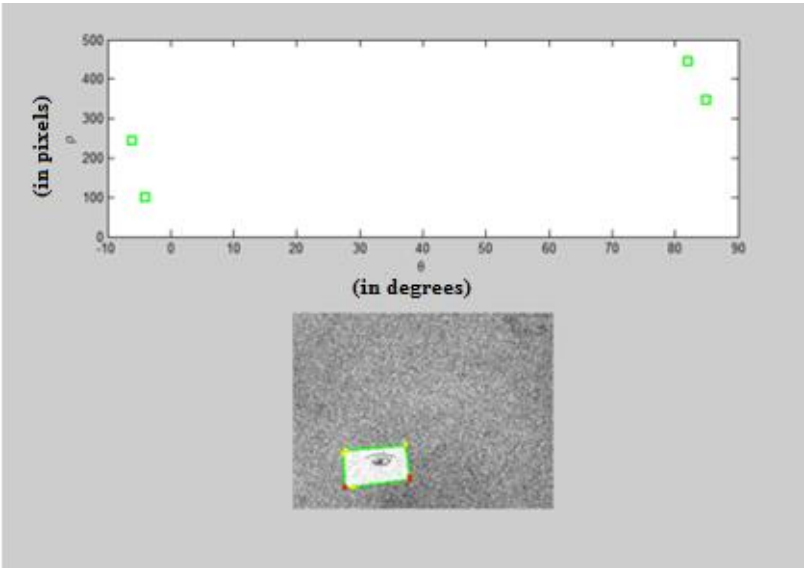


Figure 4.26e: Output of the algorithm with Sobel pre processing

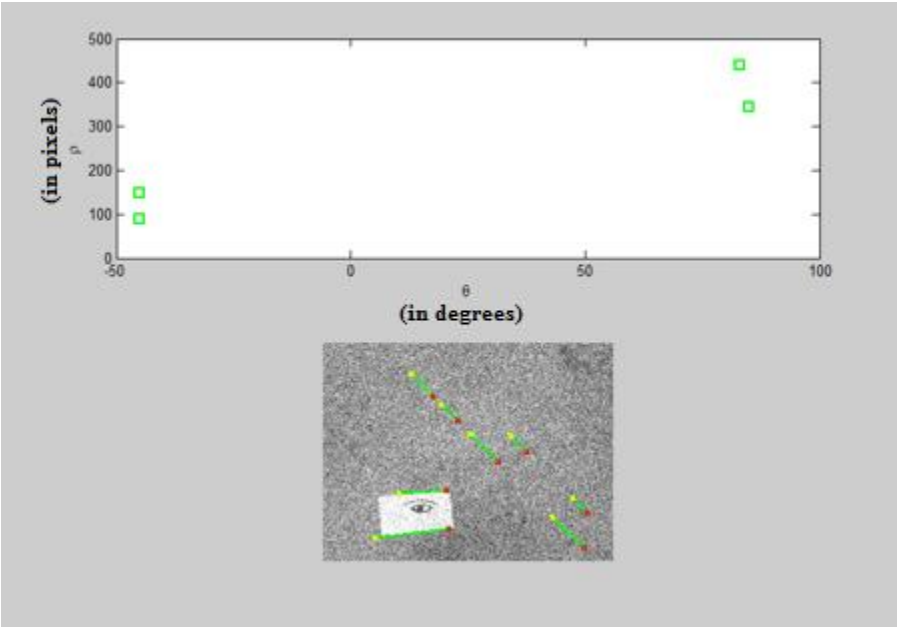


Figure 4.26f: Output of the algorithm with Canny pre processing

Figure 4.26 Output of the proposed algorithm on test image with gaussian noise under different edge detection schemes

It can be observed that similar to figure 4.23, 4.26f also shows multiple instances of the same line. This is evident from the fact that only 4 points are detected in the parameter space.

Figure 4.27a is the combined parameter space representation for all the 11 detected image frames. Each of the 4 different coloured squares in figure 4.27a represents 4 different lines detected in each of the input image frames. The ellipses surrounding each of these points are the error ellipses drawn with a probability level of 0.95.

It can be observed that the variation between consecutive values for all the four lines is not significant, especially for line 3 and line 4. Both of these lines represent the two parallels that are approximately horizontal. It is important to note that the 14 test images used for this set of experiments are not 14 consecutive frames captured from a video but 14 independent images taken from a camera at different times.

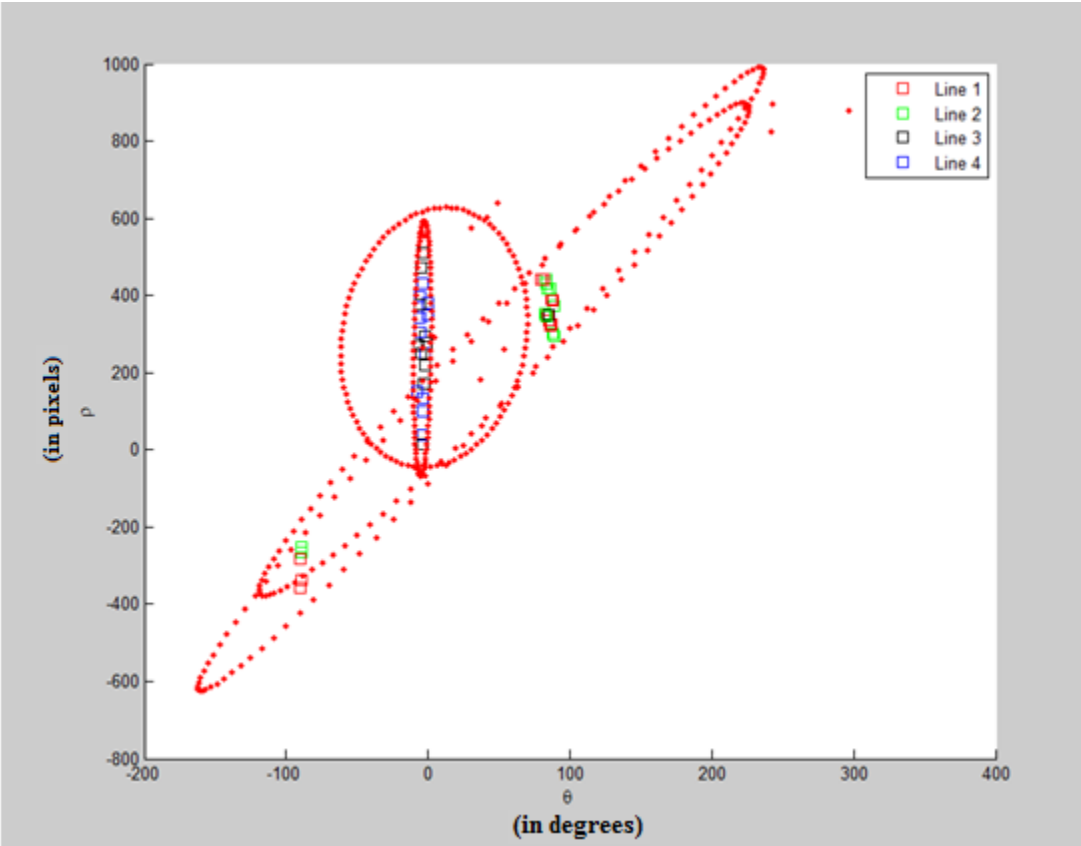


Figure 4.27a: Parameter space representation for all image frames

With a modern camera acquiring images at a modest 30FPS, further reduction in the variation between these image frames could be observed. Hence the variation between these parameter space representations being small signifies that the variation between the underlying lines in the image frames is also small. This implies that Hough transforms which is a powerful tool for FMO estimation but is fundamentally costly in terms on computational complexity can now be optimised to first search in a specific error ellipse region for lines.

If the required lines are not found in this parameter space error ellipse region, then the search can be extended to other regions in the parameter space thereby indirectly extending the search to test the entire image frame. This approach is illustrated in figure 4.27b.

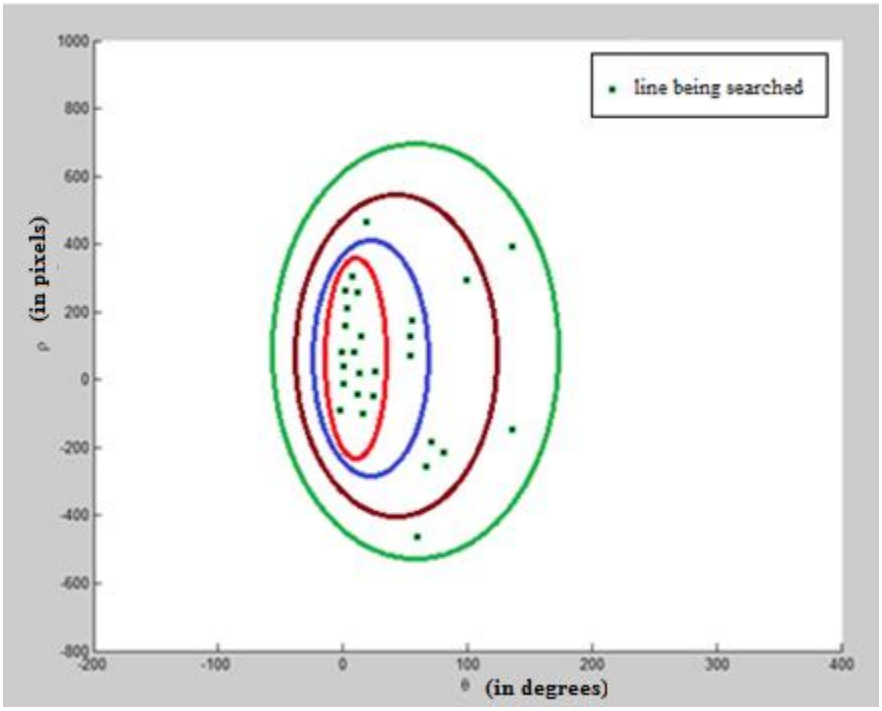


Figure 4.27b: Figure illustrating the error ellipse search mechanism

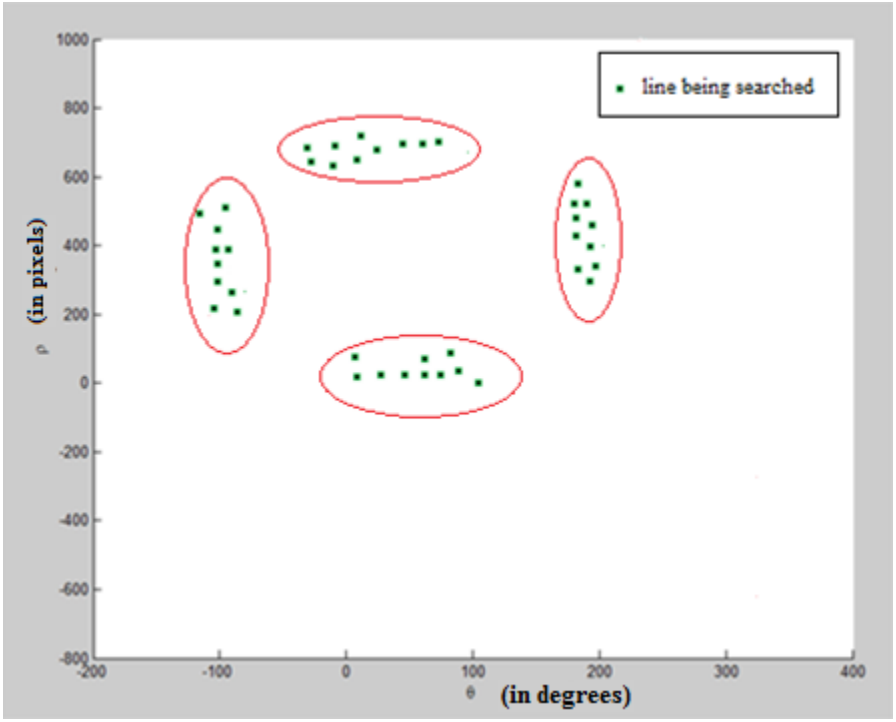


Figure 4.27c: Figure illustrating method of ROI estimation for HT

Figure 4.27 Parameter space representation and proposed search mechanism

The green squares in figure 4.27b show the points in parameter space which are equivalent to lines to be searched in the image frame. Figure 4.27b is only an illustration to explain the concept more clearly and does not reflect real data like in figure 4.27a. In figure 4.27b, if the line to be searched varies very slowly then the search within the inner most red error ellipse will give reasonable results. If now the line starts to vary more abruptly then the area under the error ellipse can be increased such that a wider range of (ρ, θ) values can be covered thereby more lines can be searched. This is illustrated by blue, brown and green error ellipses in the order of increasing area. Ellipses are considered since they form a standard for measurement errors. Interchangeably, circles can also be used for search.

As already mentioned, a future task could be to design a SLAM based system that can integrate IMU and cameras. This SLAM algorithm can be composed of any recursive bayesian filtering scheme, EKF for instance for online SLAM [44] which inherently contains a prediction step in the algorithm. These error ellipses as shown in figure 4.27c can be used to define the region of interest (ROI) for the HT. Hence convergence can be achieved faster.

This approach can also decrease the computational cost of the HT algorithm by appropriately searching in only certain regions in the image frame initially. This is further discussed in the future work section of chapter 5.

4.5.3 HOMOGRAPHY ESTIMATION USING HT FOR REAL-TIME CAPTURED VIDEO

All the experiments discussed so far are carried out on rectangular shaped objects. Even though this thesis characterizes FMO as rectangles, it is not a necessary condition that all FMO should be rectangular. If the FMO under consideration is composed of a completely irregular object, then HT can be replaced by GHT and the same processing can be repeated. FMOs should not be always generalised as rectangles as a partially occluded rectangular FMO might also exhibit non-rectangular geometry

Occlusion is one of the most important test cases for functional verification. They can be represented such that the geometry is maintained for certain regions of the FMO and an irregular shape of a certain sort for the rest of the FMO.

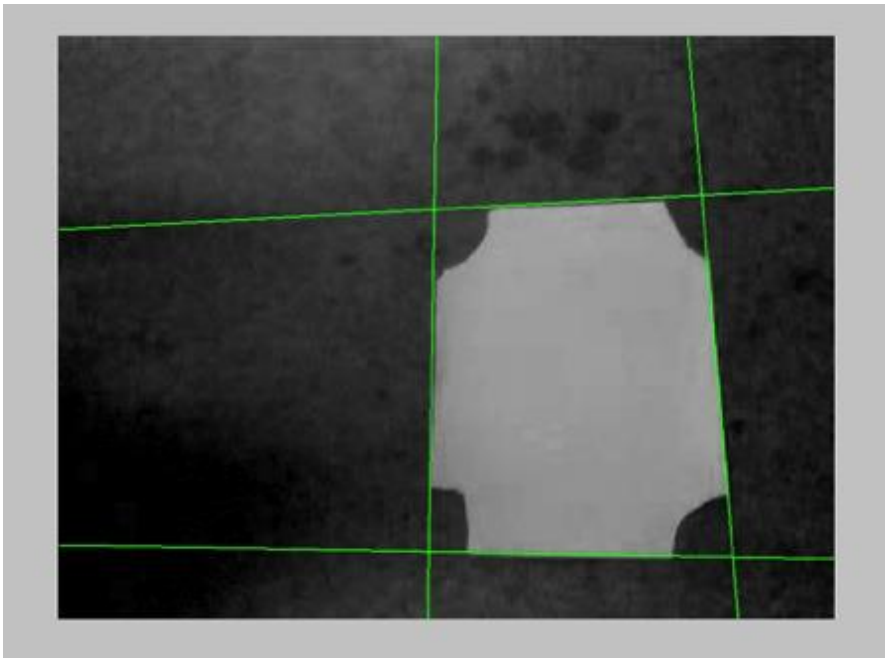


Figure 4.28: Vertex estimation for Perspective Transform

The experiments performed in this section are on image frames that are extracted from a real-time captured video which is stored on the processing computer. The test case shown in figure 4.28 is one in which the vertices of the FMO have been removed. Hence the vertices now need to be estimated dynamically which are given as an input to the 4 point algorithm. Figure 4.28 shows one of the video frames on which vertex estimation has been performed. These vertices can later be employed in consecutive image frames to determine the perspective transformation to extract R and T vectors. The goal of this section is to explain the process of trajectory estimation. As it can be seen, the vertices that are estimated are fairly accurate. It is important to note that this method needs four lines for rectangular FMO vertex estimation. To verify if this method accurately determines the vertices, a second set of experiments was performed in which one of the vertices was still in place. If this vertex can be estimated correctly in the test frame, then the algorithm can be attributed as a robust algorithm.

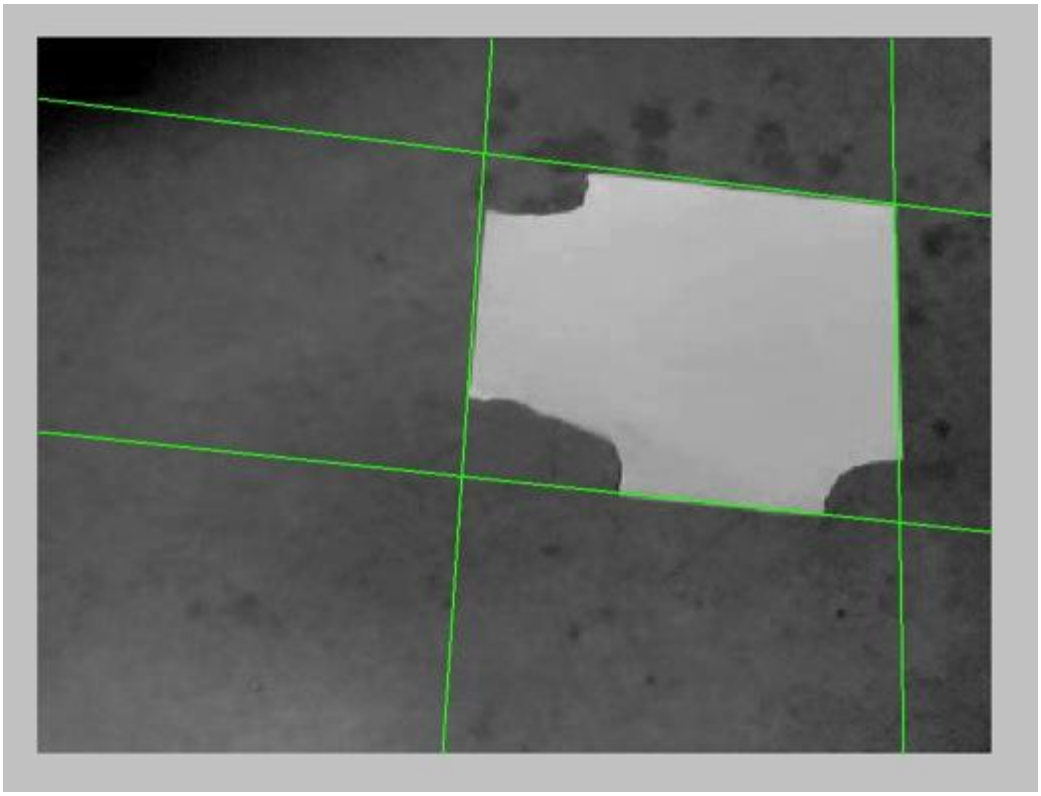


Figure 4.29: Robustness testing of vertex estimation for homography estimation

Figure 4.29 shows the above mentioned test case in which one vertex of the FMO is still intact. It can be seen in the test frame that the vertex estimation is quite accurate proving that the algorithm is indeed quite robust. It can also be seen that the Hough lines are not perfectly coinciding with the edges of the FMO. This is attributed to the fact that the FMO is not a perfectly smooth plane. In the next set of experiments shown in figure 4.30-4.32, analysis is done on various false-negative test cases encountered by the algorithm.

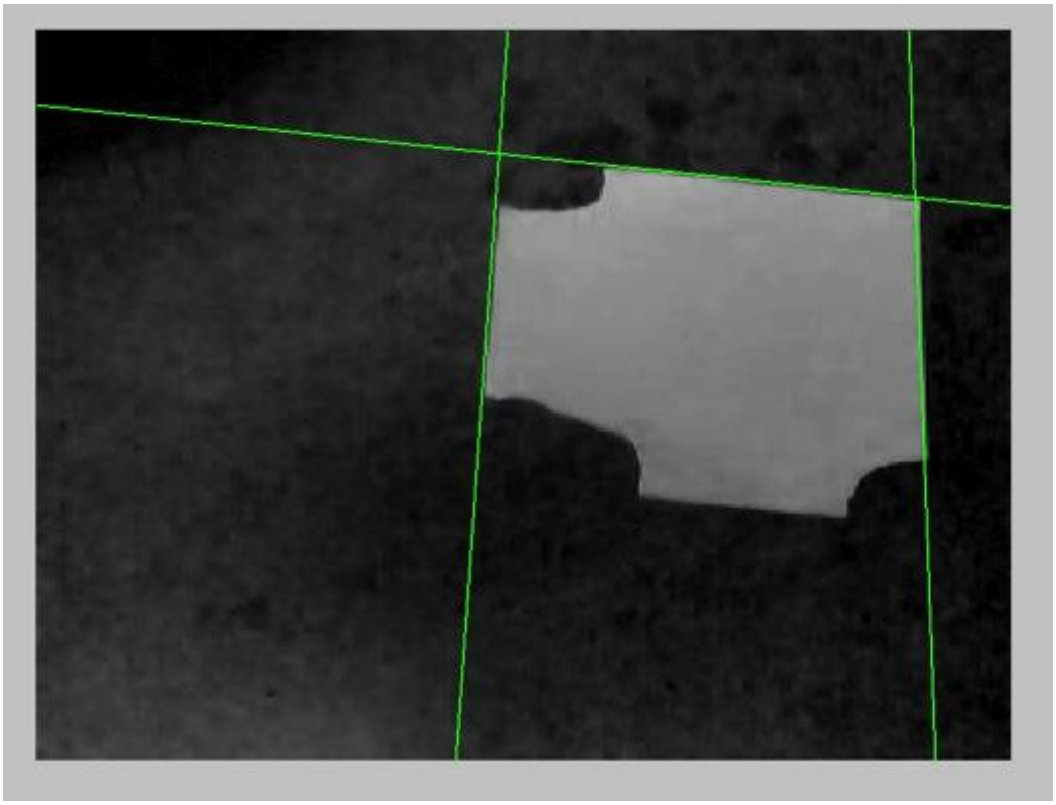


Figure 4.30: Vertex estimation false negative test case

It was observed that the algorithm sometimes does not detect one of the edges. This is similar to the test case shown in figure 4.24-4.25 where edge detection is not acceptably performed due to improper threshold during edge detection. In such a scenario, threshold can be adaptively increased. If less than 4 vertices are detected as shown in figure 4.30, then the test frame can be completely disregarded using RANSAC as discussed in section 2.7. Even if all 4 vertices are detected, they don't always form a quadrilateral. This is illustrated in the experiment given in figure 4.31-4.32.

Hence some additional processing might still be required. Sometimes the vertices might form a triangle with three collinear vertices. In such a case, another test to check for collinearity of the detected points might prove useful.

Following the former approach of applying adaptive thresholds is more advisable as the change in the threshold can be attributed to change in the brightness conditions which might not change for quite a few frames. If the second method is employed and frames which do not form quadrilaterals are discarded then a loss in the R and T values might be experienced which might

lead to improper trajectory or worst if the lighting condition does not change, the trajectory from that instance might be permanently lost.

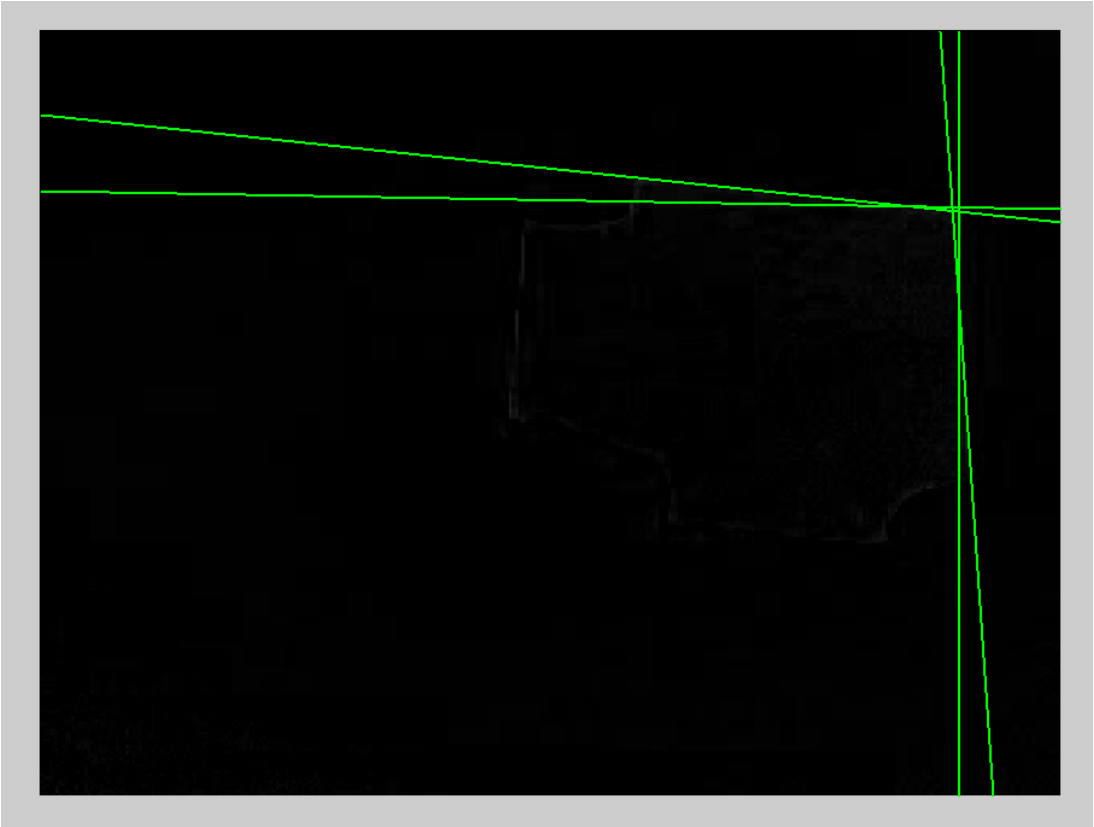


Figure 4.31: vertex estimation false positive test case

Checking for collinearity will not always lead to correct vertices. Sometimes checking for all of these conditions will still lead to incorrect results. Figure 4.31 illustrates this case in which even though 4 lines are detected, they do not necessarily fail the test. A closer look into the region reveals that this is because all vertices are indeed non-collinear with a definite positive area forming a quadrilateral. Figure 4.32 is a magnified version of figure 4.31 near the intersection region formed by the 4 lines.

Since the assumption made on the CV system is that the frames move very slowly relative to the camera's frame rate. Hence the estimated vertices for the current frame must lie very close to the estimated vertices of the previous frame. It can be seen that the four lines sometimes form a quadrilateral, however with wrong result as seen in the present test case. In such a scenario, a test can be conducted to see if the estimated vertices are within a certain range from the estimated

vertices from the previous frames by a small threshold. If not, the test frame can be disregarded. Fortunately this test case was not observed to happen for the entire video as in that case all frames would give wrong data that will lead to dealing with this problem in a completely different perspective.

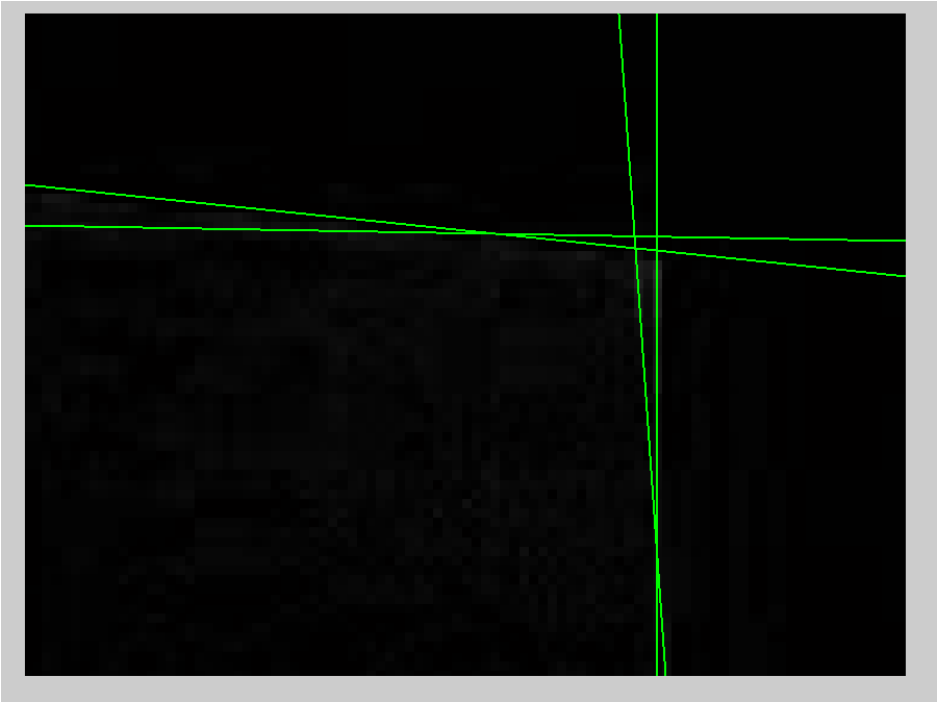


Figure 4.32: vertex estimation false positive test case- closer look

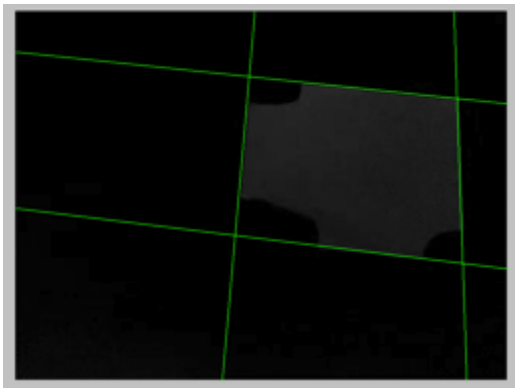


Figure 4.33a

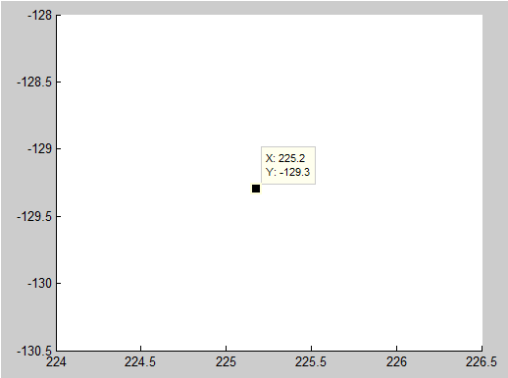


Figure 4.33b

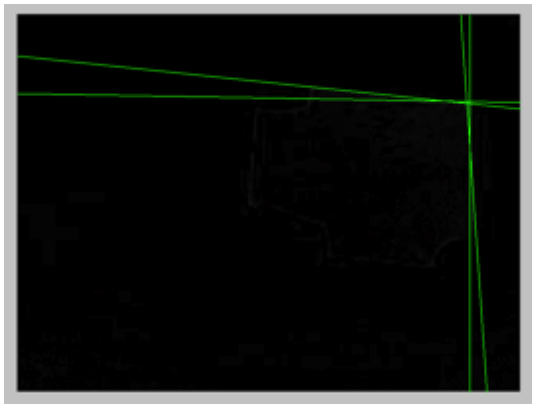


Figure 4.33c

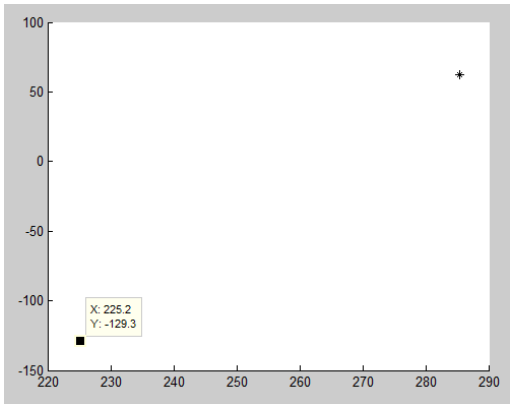


Figure 4.33d

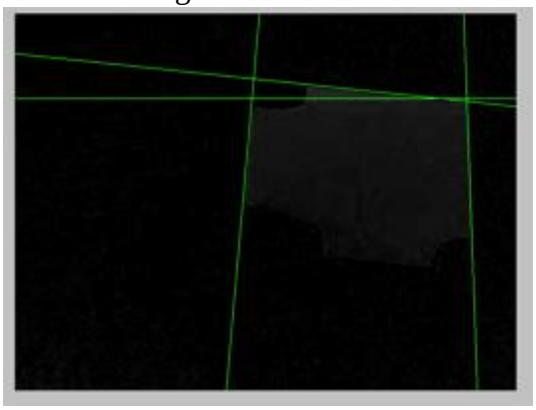


Figure 4.33e

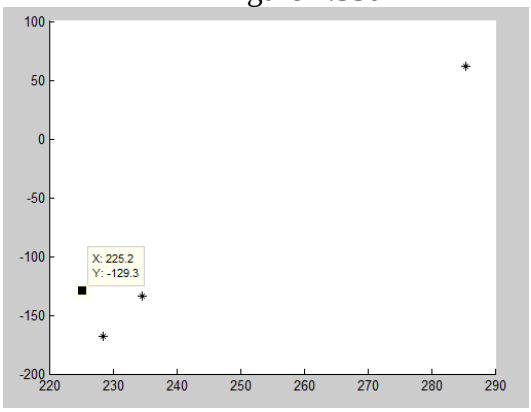


Figure 4.33f

Figure 4.33: Various test cases of vertex estimation

Figure 4.33 shows the various test cases for vertex estimation. For each test case, the figure on the left shows the test case while the figure on the right shows the centroid of the estimated geometrical figure in pixel units. The point at (225,-129.3) is the centroid for the first test case with correct results. This is shown in figure 4.33a and 4.33b. This point is shown in all test-cases to show the drift in the centroid. The image frame in 4.33a is fairly dark. This is the first frame in the saved video and any video saved on a computer usually tends to get this effect, gradually achieving its true intensity within the first few frames. This process is usually very fast and cannot be captured by the human eye in most cases. The test case shown in figure 4.31 and figure 4.32 are also tested. It can be seen that the centroid of this false positive case is farther away from the true centroid of the previous frame. The last set shown in figure 4.33e and 4.33f is another false positive test case in which two triangles are formed by the 4 detected lines. Hence it can be seen that the addition of this frame introduces two centroids instead of one from a single

quadrilateral. The fact that two centroids are detected instead of one itself can be employed to test if a quadrilateral has been detected or not. Also the two centroids are quite farther away from the true centroid of the first frame. The difference might not seem significant in figure 4.33f but this variation is quite considerable because these frames are consecutive frames in a video.

A very useful test to verify if the lines form a quadrilateral is to check if a contour is detected or not. Since contours alone cannot provide much help if partially occluded figures like the examples of section 4.5.2 are used, they have to be coupled with HT to estimate PT. Hence a combination of these two complementary algorithms might yield better results.

This method might prove useful to extract PT from occluded shapes which are not necessarily rectangular. I.e. the edges of a partially occluded circle or an ellipse or any other geometrical figure can also be extracted which can be processed to extract PT. This will eventually lead to extraction of R and T vectors from consecutive image frames. This is a prediction and needs more experimental verification. More details about the same are given in the future work proposed in chapter 5.

4.6 ANGLE ESTIMATION FROM PERSPECTIVE TRANSFORM

Since the estimation of the vertices of the FMO is complete, extracting the rotation angles can now be done as shown in figure 4.34. The example comprises of an FMO which is shown in figure 4.34a. The image is rotated by 45 degrees using the matlab function imrotate. The remainder of the image is left in black due to the spatial transformation. This was done since a black background gives the highest gradient making the algorithm to always select this border as an edge. Since there is no additional black background encountered in a real-time scenario, is filled up with the same background. This is visible in the outer edges of the rotated FMO figure shown in figure 4.34b.

Figure 4.34a

figure 4.34b

The image is rotated about its center with respect to only one angle so that the effect can be observed visually. The vertices detection algorithm which is shown in Figure 4.18 is first used to determine the edges of the FMO. This is shown in figure 4.34c and 4.34d. The Hough lines that are detected are shown in both of the figure in green. The black stars depict the vertices detected by the proposed algorithm.

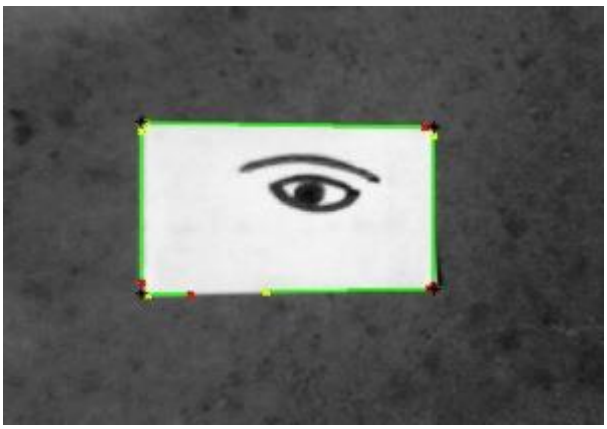


Figure 4.34c

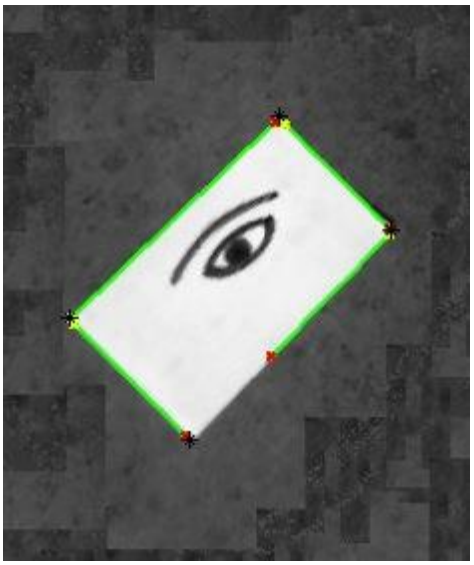


figure 4.34d

Figure 4.34 Example illustrating the extraction of rotation angle from perspective transform

Since the image was rotated along the z-axis, the angle γ, which represents the rotation angle about the Z-axis, should be estimated. The four points occurring in each of the images in pixel units are found to be as follows.

Table.3. vertices estimated by the proposed method in pixel units

vertices shown in figure 4.28a		vertices shown in figure 4.28b	
152	151.6	82.6	341.4
152	331.3	209.5	468.2
457	156.9	302.1	129.5
457	326.1	421.4	248.8

These vertices are given as inputs to the function GetPerspectiveTransform, the output of which gives the perspective transform matrix using the 4 point algorithm. The rotation and translation matrices are extracted from this perspective transform matrix. The extraction of rotation and translation from perspective transform requires coordinates along all three axes. Since the FMOs used in this thesis are assumed to be planar hence the Z coordinates of all the vertices can be safely assumed to be zero. Since the current example aims at estimating the rotation angle, only the rotation matrix extracted from this transformation is considered in this section. Equation 3.7, which gives the overall rotation matrix for a generalised case after any rotation with respect to any axes, is utilised to estimate the angle γ. The rotation angles along X, Y and Z axis can be computed using equations 3.8, 3.9 and 3.10. Using these equations the values of the rotation angles were found. The translation vector values in pixel units were also found. The results are shown in table 4

Table.4. Extraction of rotation angles and translation vector from PT

Rotation angles in degrees			Translation in pixel units		
α	β	γ	T_x	T_y	T_z
-7.07E-10	-2.03E-05	45.05	-132	341.5	0

The value of γ is found to be 45.05 degrees, which is very close to its true value of 45 degrees proves that the angle extraction method discussed in section 3.3 is correct.

Chapter 5

Conclusions and Future Work

This thesis presents an analysis and formulation of 6 DOF egomotion using FM and FMO. The use of HT for FMO extraction and processing was demonstrated in chapter 4. The use of visual sensors in location and trajectory estimation is the primary motivation behind this thesis. The present chapter summarizes the main contribution of this research work. Finally a list of future activities for further enhancement of the proposed algorithms is made.

5.1 CONTRIBUTIONS

The objectives as outlined in section 1.6 were achieved. The following list describes a more detailed summary of this research.

- Literature survey and formulation of the pre-processing for CV based algorithms including smoothing, thresholding, contour detection and RANSAC.
- Performed camera calibration and estimated the camera's intrinsic parameters. The radial and tangential distortions were also estimated using two independent test cases.
- Analysed the proposed algorithm for extraction of rotation and translation vectors from fiducial markers and a single camera on a 64 bit computer with modest configuration using perspective transformation and capable of storing codes that enable 3D location data to be decrypted from these codes. An example of the 3D location data association was given in figure 3.2 and 3.3. The use of 8 bit grayscale images for this algorithm enables an almost universal use of the algorithm in sufficient brightness. It also reduces the load on the processing computer as 8 bit images are faster to process and occupy less memory space of the processing computer. This is a required condition since most modern day navigation or LBS algorithms are employed in mobile handset environments which have stringent memory and processing requirements.

- Implemented the back-projection test method to evaluate the accuracy of the proposed algorithm for a random camera motion and accomplished millimetre level accuracy for the same as shown in figure 4.5 and figure 4.6. Back-projection is a method to estimate the error introduced by an algorithm by comparing the inverse transformation of a state space with the original state space on which the inverse transform is projected back. More details are given in Appendix C.
- Evaluated the performance of the algorithm by estimating the RMS error using a rotary stage and a series controller to provide a known trajectory to the camera and accomplished a centimeter level accuracy as shown in figure 4.9-4.10.
- Performed experimental verification to test for functionality of the algorithm in case of a transition from FM to FMO as shown 4.13.
- Performed experimental verification to test for functionality for long range trajectory as shown in figure 4.15-4.16
- Proposed and analysed an algorithm to perform homography estimation using Hough line transform for FMO processing as shown in figure 4.18-4.19. The use of 8 bit grayscale images offers all the advantages of the FM detection algorithm that was previously shown in figure 3.1.
- Evaluated the performance of the algorithm using 14 image frames to verify various test cases. The experiments show promising results when the algorithm was tested using Sobel operator preprocessing. Adaptive thresholding is also recommended for better performance of the algorithm. This is shown in figures 4.20 - 4.27.
- Performed experimental verification to prove that the parameters space representation of a FMO changes very slowly in an image sequence. This is shown in figure 4.27a. This makes it possible for the HT algorithm of the FMO to be optimised. The optimisation problem is yet to be solved and is discussed in future works.
- Experimental verification of the algorithm for vertex extraction for various occluded test cases is performed. The experiment was performed on a video frame as shown in figure 4.28 to 4.33. Various other recommendations to make the algorithm more robust are also given in section 4.5.3.

Experimental verifications given in this thesis demonstrate that FM systems work quite robustly when detecting figures that are either FM, FMO type or a combination of both. As demonstrated in sections 4.5 and 4.6, if a set of points (corresponding to vertices in the current case) with an assumed geometry (rectangular) are observed then meaningful information in the form of rotation and translation can be extracted from these points. This was demonstrated in section 4.6.

5.2 FUTURE WORK

The 6 DOF egomotion estimation and the Hough techniques for FMO discussed in this thesis are demonstrated in qualitative and quantitative experiments; however, there is a potential to make them more robust, portable and accurate. Following is a list of future tasks that are proposed in the domain of these algorithms and techniques.

- One of the most important future tasks is to integrate trajectory data from other sensors like IMU with camera observables. This was discussed in the SLAM algorithm shown in section 3.4.
- Another important future work is to compare the lines detected by using DCT and HT algorithm as discussed in section 4.5.2. It is yet to be observed if both these algorithms estimate the same corresponding lines taken from the same video. If so, the problem of finding correspondence can be solved.
- If the 6 DOF egomotion algorithms described in this thesis is to be employed in real-time basis on a modern day mobile handset, then using a self-calibrated camera is advisable. This is because the environment will not always be in the same brightness. The regular usage can also incur wear and tear in the camera thus changing the camera's parameters which might need recalibration.
- It has been demonstrated in figure 4.27 that the parameter space representation for HT for image frames does not vary significantly. This is all the more true in the case of consecutive image frames occurring from a real-time camera. A low data rate is generally sufficient for processing of the algorithms as evident from the fact that only 8 bit binary grayscale images are needed to process the algorithm. Hence the total frame rate can be exploited for

processing such algorithms leading to an even lesser variation of the lines in the image frames. This can indirectly lead to lesser variation in the parameter space representation of the FMO. Hence the recommendation of searching for lines within a certain error ellipse region to estimate HT can prove significantly useful. There are however a few algorithms that process faster HT [51]. The current recommendation might prove to be a useful addition in the class of similar algorithms like probabilistic HT [52]. Once proven that it does offer correct results, verification can then be performed to check which method offers a faster solution for a thorough comparison.

- The vertex estimation algorithm discussed in section 4.5.2 and 4.5.3 can be made to work more efficiently by checking better conditions for quadrilateral estimation. The most important addition will be checking for a contour before detecting the quadrilaterals. The RANSAC algorithm, which is currently a post processing technique, can be made a part of the algorithm such that erroneous data can be rejected as soon as they are encountered.
- From experiments shown in figure 4.28-4.33 which are test cases for partially occluded rectangles, it is evident that such occluded rectangles can still be used for homography estimation as vertices of these figures can be successfully extracted. These can later assist in rotation and translation estimation using 4 point algorithm. As recommended in the same section, this can be extended to any partially occluded geometrical shapes like ellipses or circles and hence homography can be estimated. This is similar to probabilistic Hough transforms which detect occluded regions in the image. As already mentioned, further experimental verifications are yet to be performed to test as to which method gives a better and faster estimate of the true geometry of the FMO. Circles however will have the problem of rotational ambiguity associated with them making it difficult to estimate if rotation occurred on the circle or not. However translation can still be extracted.
- The experiments performed in chapter 4 are cases of a single FMO in the FOV of the camera. If multiple FMO are present in the FOV then estimating the relative pose variation between the FMO is a completely different problem. In such a scenario, HT can again prove to be a useful tool. As shown below, the difference in the angle that the edges make with one of the coordinate axis can provide the relative pose angle between the FMO. This can prove to be a

useful addition in the proposed FMO extraction algorithm. This angle can directly be estimated by extracting the θ value for both of the edges shown and then subtracting one angle from the other. The major problem encountered by this simple technique is the correspondence between an edge of the FMO and the HT detected line. The algorithm does not automatically know which two edges to consider in order to extract the relative angle between them As mentioned above, if the combination of DCT and HT prove useful in correspondence estimation then this is a simple algorithm to design and analyse.

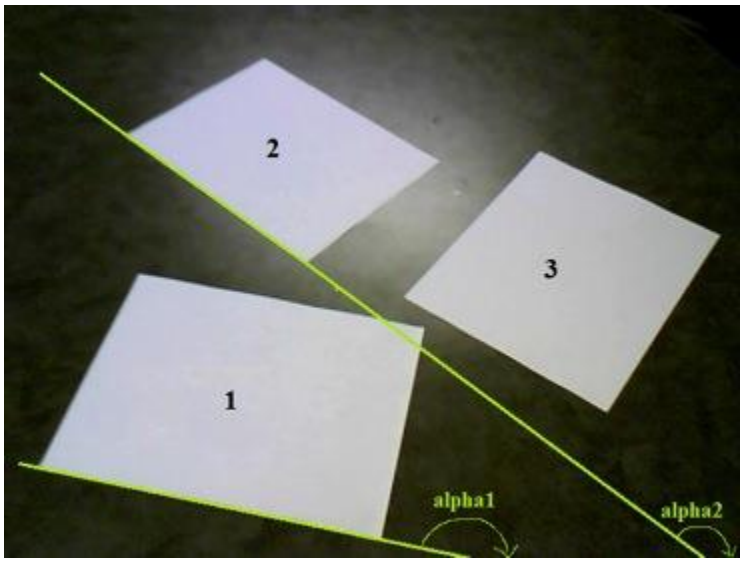


Figure 5.1: Pose estimation for multiple FMO

It has been demonstrated in this thesis along with many other documents in the literature that a single non-stationary modest resolution camera can offer information to such an order that 3D navigation and positioning can be entirely performed. The current technology used in visual sensors is such that the data acquired by these sensors is difficult to process and to aid the situation, dimensionality reduction techniques are already being widely accepted [53]. Technological advancement in CV based systems are only limited today in terms of the processing capabilities of the processing computer. This gap is being filled with state of the art GPU systems which are also findings their ways in mobile phones [54].

BIBLIOGRAPHY

[1] A. Virtanen and S. Koskinen, "*Towards Seamless Navigation*," in Proceedings of the MobileVenue'04, 2004.

[2] F. v. Diggelen, "*Indoor GPS theory and implementation*," in IEEE Position Locotion and Navigation Symp, April 2002.

[3] M. lrsigler, B. Eisrfeller, "*Comparison of multipath mitigation techniques with consideration of future signal structures*," Proc. ION GPS/GNSS, pp. 2584-2592, Sept. 2003

[4] C. C. A. Shepherd, "*Chapter 5: Indoor Air Pollutants and Toxic Materials*," in *Healthy Housing Reference Manual*, The Center for Disease Control (now known as the Centers for Disease Control and Prevention), Original Printing 1976, Reprinted 1988,Updated and Revised 2006, p. 2.

[5] X. Wang; A. K.-S. Wong; Y. Kong, "*Mobility tracking using GPS, Wi-Fi and Cell ID,*" 2012 International Conference on Information Networking (ICOIN), pp.171,176, 1-3 Feb. 2012

[6] S. Anthony, "*Think GPS is cool? IPS will blow your mind*," 24 April 2012. [Online]. Available:http://www.extremetech.com/extreme/126843-think-gps-is-cool-ips-will-blow-your-mind. [Accessed 24 January 2013].

[7] K.W. Kolodziej; J. Hjelm; "*Local positioning systems: LBS applications and services*". CRC Press, Boca Raton (2006)

[8] J. Krumm;"*Ubiquitous Computing Fundamentals*". CRC Press, Boca Raton (2010)

[9] A. Bensky; "*Wireless positioning technologies and applications*",Artech House, Boston (2007)

[10] H. Liu; H. Darabi; P. Banerjee; J. Liu: *"Survey of Wireless Indoor Positioning Techniques and Systems*". IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 37(6), 1067–1080 (2007)

[11] G. Deak; K. Curran; J. Condell, "*Evaluation of Smoothing Algorithms for a RSSI-Based Device-Free Passive Localisation*," in Image Processing & Communications Challenges 2, Springer.com, 2010, p. 470.

[12] M. Yan, X. Yubin and C. Xiuwan, "*Wireless Local Area Network Assisted GPS in Seamless*

Positioning," in 2012 International Conference on "Computer Science and Electronics Engineering (ICCSEE)", 23-25 March 2012

[13] K. Bakhru, "*A Seamless Tracking Solution for Indoor and Outdoor Position Location*," in IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin, Germany, 2005.

[14] T. Manabe; S. Yamashita; T. Hasegawa, "*On the M-CubITS pedestrian navigation system*," in Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems (ITSC), Toronto, Canada, Sept. 2006, pp. 793–798.

[15] S.S.Chawathe; "*Marker-Based Localizing for Indoor Navigation*", Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE, vol., no., pp.885-890, Sept. 30 2007-Oct. 3 2007.

[16] S. Saito; A. Hiyama; T. Tanikawa; M. Hirose; "*Indoor Marker-based Localization Using Coded Seamless Pattern for Interior Decoration*," Virtual Reality Conference, 2007. VR '07. IEEE, vol., no., pp.67-74, 10-14 March 2007.

[17] M.A. Lakhani; J. Nielsen; G. Lachapelle; "*Computer Vision Navigation Based on Fiducial Markers of Opportunity",* Proceedings of the ION 2013 Pacific PNT Meeting, Honolulu, Hawaii, April 2013, pp. 352-361.

[18] M.A. Lakhani; J. Nielsen; G. Lachapelle, *"Indoor Navigation Based On Fiducial Markers Of Opportunity",* Proceedings of the IPCV 2013 conference, Las Vegas, Nevada, July 2013.

[19] J. Shi.; C. Tomasi, "*Good features to track*," 1994. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '94., 1994. pp.593,600, 21-23 Jun 1994.

[20] J. Canny, J., "*A Computational Approach To Edge Detection*," IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[21] Hough, P. V. (1960, Mar 25). *United States Patent No. US 3069654 A.*

[22] T.G.Stockham, "*Image Processing in the Context of a Visual Model*,'' Proc. IEEE, vol. 60, pp. 828-842, 1972.

[23] J.W.Weszka, R.N.Nage1, and A.Rosenfeld, "*A Threshold Selection Technique*," IEEE Trans. Computer, vol. C-23, no 12, pp. 1322-1326, 1974.

[24] N .Otsu, "*A Threshold Selection Method from Gray-Level Histogram*,'' IEEE trans. Sys., Man and Cyb., vol. SMC-0, 12, pp. 62-66, 1979.

[25] J .Kittler, J Jllingworth, J .Foglein and K.Paler, "*An Automatic Thresholding Algorithm and Its Performance*," Proc. Seventh Int. Conf. Pattern Recognition, vol.1, pp.287-289, 1984, Montreal, P.Q., Canada.

[26]R. Mukherjee, "pythongeek.blogspot.ca," 7 June 2012. [Online]. Available: http://pythongeek.blogspot.ca/2012/06/canny-edge-detection.html

[27]J. Nielsen, "Discrete Time Signal Processing", retrieved from lecture notes of , ENEL 655 University of Calgary

[28] S. Suzuki, and K. Abe, "*Topological Structural Analysis of Digitized Binary Images by Border Following*." CVGIP 30 1, pp 32-46 (1985)

[29] H. Rhody, "Geometric Image Transformations" [PDF document]. Retrieved from Lecture Notes available at: http://www.cis.rit.edu/class/simg782/lectures/lecture_02/lec782_05_02.pdf

[31] Camera image adopted from, N. Raymond, "*Digital Camera Back -EPS Vector*", available online at "http://freestock.ca/vectors_g83-digital_camera_back__eps_vector_p2123.html

[30] J. Nielsen, "Transformations and Image mappings", retrieved from lecture notes of ENEL-503, Computer Vision, University of Calgary

[32] Z. Zhang, "*Camera calibration with one-dimensional objects*," IEEE Transactions on Pattern Analysis and Machine Intelligence, , vol.26, no.7, pp.892,899, July 2004

[33] R.Y. Tsai, "*A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses*," IEEE J. Robotics and Automation, vol. 3, no. 4, pp. 323-344, Aug. 1987

[34]F. Devernay and O. Faugeras, "*Straight lines have to be straight*," Machine Vision and Applications, vol. 13, no. 1, pp. 14–24, 2001.

[35] L. Ma, Y.Q. Chen, K.L. Moore, "*A new analytical radial distortion model for camera calibration*", http://arxiv.org/pdf/cs/0307046.pdf, 2003.

[36] L. Ma, Y.Q. Chen, and K.L. Moore, "*Analytical piecewise radial distortion model for precision camera calibration*", Proceedings in Vision, Image and Signal Processing, vol 153 (4), IEEE, pp. 468-474, 2006.

[37] Radial distortion image adopted from, *Z. Chagany, "Lens Distortion" available online at* http://dslrfilmschool.com/wp-content/uploads/2013/01/Lens-Distortion.png

[38] Calibration code adopted from "*Camera calibration With OpenCV*" available online at http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[39] M. A. Fischler and R. C. Bolles ,*"Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*". Comm. of the ACM 24 (6): 381–395, June 1981.

[40] O. Chum, "Two-view geometry estimation by random sample and consensus" PhD thesis, CMP, CTU in Prague, 2005.

[41] L. Ruotsalainen, J.B. Bancroft, G. Lachapelle, H. Kuusniemi and R. Chen, "*Effect of Camera Characteristics on the Accuracy of a Visual Gyroscope for Indoor Pedestrian Navigation"*. Second International Conference on Ubiquitous Positioning, Indoor Navigation and Location-Based Service, Helsinki, 3-4October, 8 pages.

[42] J. Illingworth, J. Kittler, "A survey of the Hough transform Computer Vision", Graphics, and Image Processing Volume 44 Issue 1, Oct. 1988, Pages 87 – 116\

[43] D. H Ballard, "Generalizing the Hough transform to detect arbitrary shapes", Pattern Recognition, Vol. 13, No. 2. (January 1981), pp. 111-122, doi:10.1016/0031-3203(81)90009-1

[44] J. Nielsen, "Finding, isolating and tracking objects ", retrieved from lecture notes of ENEL-503, Computer Vision, University of Calgary

[45] L. Zhi, J. Tang, "*A Complete Linear 4-Point Algorithm for Camera Pose Determination 1)"*available online at http://www.mmrc.iss.ac.cn/pub/mm21.pdf/zhi3.pdf

[46] Bradski, G. (2008-01-15 19:21:5). Dr. Dobb's Journal of Software Tools

[47] B. D. Lucas, T. Kanade, "*An Iterative Image Registration Technique with an Application to Stereo Vision*", Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2 Pages 674-679

[48] C. Golban, C. Mitran, S. Nedevschi, "*A practical method for ego vehicle motion estimation from video,*" Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on , vol., no., pp.87-94, 27-29 Aug. 2009.

[49] S. Thrun, D. Fox, W. Burgard, "*Probabilistic Robotics*", MIT press, 2005, pp. 309-322.

[50] Y. Chiang; C.A Knoblock, "*Classification of Line and Character Pixels on Raster Maps Using Discrete Cosine Transformation Coefficients and Support Vector Machine*," 18th International Conference on Pattern Recognition, 2006. ICPR 2006., vol.2, no., pp.1034,1037.

[51] N. Guil; J. Villalba; E. L Zapata*; "A fast Hough transform for segment detection,"* IEEE Transactions on Image Processing, , vol.4, no.11, pp.1541,1548, Nov 1995.

[52] S. Guo; Y. Kong; Q. Tang; F. Zhang, "*Probabilistic Hough transform for line detection utilizing surround suppression*," International Conference on Machine Learning and Cybernetics, 2008 , vol.5

[53] Jolliffe I.T. "*Principal Component Analysis*", Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

[54] Lindholm; J. Erik , S.F. Oberman; "*Nonlinear resonant circuit devices,"* U.S. Patent 3 624 125, July 16, 1990.

[55] K. Yamaguchi; "MexOpencv"- Collection and a development kit of matlab mex functions for OpenCV library, available at http://www.cs.stonybrook.edu/~kyamagu/mexopencv/

[56] Input image in figure 1.5 adopted from, padstyle.com available online at http://padstyle.com/wp-content/uploads/padstyle/Hidden-Doors.jpg

[57] Input image in figure 2.5, figure 2.7 and figure 2.8 adopted from sample pictures titled "El Capitan.jpg" available in Windows 7 version of Microsoft windows

[58] Input image used in figure 2.9 and figure 2.11, adopted from shark abstract – 3D abstract wallpaper, available online at http://img.wallpapergang.com/110processed/shark%20abstract.jpg

Appendix A

OPTICAL FLOW

In the process of trajectory detection, tracking certain objects in consecutive frames of images is often required without any other prior knowledge about the contents of these frames. For problems like these it is best to introduce the concept of Optical Flow. Optical Flow is the de facto standard for egomotion estimation. Even though the algorithm presented in this thesis does not directly employ Optical Flow, it is however important to understand this concept as it might aid in future works of this project.

Optical flow is the pattern of apparent motion of objects, surfaces, edges or features in a visual scene caused by the relative motion between the object and the viewer. In simple words it could be defined as the process of tracking target features over time and converting their movements into velocity vectors. Optical flow can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects.

Optical Flow can be broadly categorised into two categories namely the dense optical flow and the Sparse optical flow. We can associate an arbitrary velocity to every pixel in the frame. Such a construction is referred as dense optical flow. One good example to this is the Horn-Schunck method. It makes use of brightness consistency assumption and derives the basic brightness consistency equations. The brightness consistency assumption is described in the equation given below.

$$I(x+\Delta u, y+\Delta v, t+\Delta t) = I(x, y, t)$$

A voxel (a volume element, representing a value on a regular grid in three dimensional space) at location (x,y,t) with intensity $I(x, y, t)$ is moved by Δx, Δy and Δt between the two image frames.

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get the following condition

$$I(x+\Delta u, y+\Delta v, t+\Delta t) = I(x, y, t) + I_x.\Delta_x + I_y.\Delta_y + I_t.\Delta_t + Higher.Order.terms$$

Thus, the optical flow constraint equation (OFCE) is obtained by using Taylor expansion and dropping its nonlinear terms. Therefore the OCFE can be expressed in the form as

$$I_x.\Delta_x + I_y.\Delta_y + I_t.\Delta_t = 0$$

$$I_x.\frac{\Delta_x}{\Delta_t} + I_y.\frac{\Delta_y}{\Delta_t} + I_t = 0$$

$$I_x.U + I_y.V + I_t. = 0$$

The solution of the brightness consistency equations mentioned above is solved by hypothesizing a smoothness constraint on the velocities V_x and V_y. This constraint was derived by minimizing the regularized Laplacian of the optical flow velocity components as shown below.

$$\frac{\partial}{\partial x}\frac{\partial\, Vx}{\partial x} - \frac{1}{\alpha} Ix(IxVx + IxVy + It) = 0$$

$$\frac{\partial}{\partial y}\frac{\partial Vy}{\partial y} - \frac{1}{\alpha} Ix(IxVx + IxVy + It) = 0$$

Here α is a constant weighting coefficient known as the regularization constant. The Larger the values of α, the smoother are the vectors of flow.

This brings us to the second type of optical flow namely Sparse Flow. Calculating sparse flow is relatively easier than dense flow because in sparse flow algorithms, we specify the set of points that are to be tracked beforehand. If these points have certain desirable properties like corners, the tracking is more robust and reliable. One of the most popular forms of sparse tracking technique is the Lucas-Kanade (LK) optical flow.

There is a chance of recovering the velocities if every point of brightness pattern can move independently. In a finite size opaque object undergoing rigid motion, neighbouring points have similar velocities and fields of the brightness in the image vary smoothly almost everywhere.

This result in discontinuities in the flow where one objects occludes another. An algorithm based on a smoothness constraint is likely to have difficulties with occluding edges as a result. One way to express the additional constraint is to minimize the square of the magnitude of the gradient of the optical flow velocity:

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2} \text{ and } \frac{\partial^2 V}{\partial x^2}+\frac{\partial^2 V}{\partial y^2}$$

Lucas and Kanade assumed as an additional constraint that the optical flow is varying smoothly long the neighbouring object points that possessed the same velocity. The least squares estimator has been adapted in (iii) to minimize the squared error, expressed as

$E_{lk}(u,v)=\sum_{\bar{x}}\left[I_x U+I_y V+I_t\right]^2$ where $g(\bar{x})$ is the Gaussian weighing function that determines the support of the centred estimator. Thus, two partial derivative equations can be expressed as:

$$\frac{\partial E_{lk}(U,V)}{\partial U}=\sum_{\bar{x}} g(\bar{x})[UI_x^2+VI_xI_y+I_xI_t]=0$$

$$\frac{\partial E_{lk}(U,V)}{\partial V}=\sum_{\bar{x}} g(\bar{x})[VI_y^2+UI_xI_y+I_yI_t]=0$$

Lucas Kanade solves these equations using Least Mean Square estimation:

$$\begin{bmatrix}\sum gI_x^2 & \sum gI_xI_y\\ \sum gI_xI_y & \sum gI_y^2\end{bmatrix}\begin{bmatrix}U\\V\end{bmatrix}=-\begin{bmatrix}\sum gI_xI_t\\ \sum gI_yI_t\end{bmatrix}$$

$$\mathrm{A} * \vec{u} = \mathrm{b}$$

$$\vec{U}=\left(A^T A\right)^{-1} Ab$$

The images of optical flow are computed using partial derivatives between pixels in the *x, y* and *t* directions as follows:

$$I_x=m_x*(I_1+I_2) \text{ where } m_x=\frac{1}{4}\begin{pmatrix}-1 & 1\\ -1 & 1\end{pmatrix}$$

$$I_y=m_y*(I_1+I_2) \text{ where } m_y=\frac{1}{4}\begin{pmatrix}-1 & -1\\ 1 & 1\end{pmatrix}$$

$$I_t=m_t*(I_2-I_1) \text{ where } m_t=\frac{1}{4}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$$

Figure I.

Figure II

Figure III.

Examples of optical flow

Appendix B

Affine Transformation: Let the vertices of the ABCD parallelogram be represented as $\{(x_A, y_A),(x_B, y_B),(x_C, y_C),(x_D, y_D)\}$ and the corresponding vertices of the transformed A'B'C'D' parallelogram be represented as $\{(x_a, y_a),(x_b, y_b),(x_c, y_c),(x_d, y_d)\}$

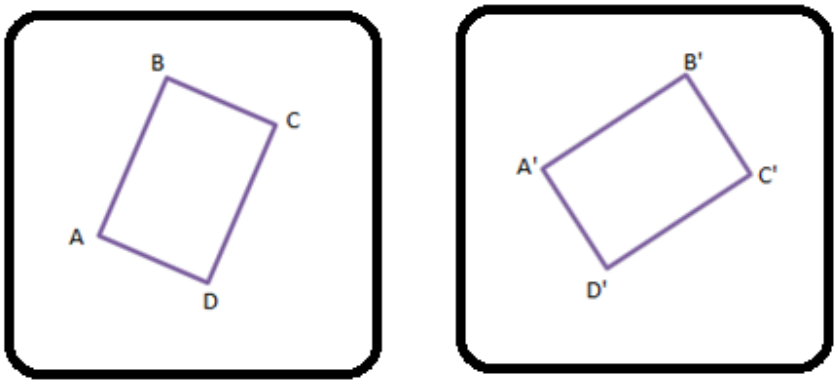


Figure IV: Affine Transformation of a parallelogram mapping into another parallelogram in two consecutive frames.

In order to satisfy the affine transformation, we have the following set of equations

$$\begin{aligned} x_a &= ax_A + by_A + c \\ x_a &= ax_B + by_B + c \\ x_a &= ax_C + by_C + c \\ x_a &= ax_D + by_D + c \end{aligned} \quad \text{and} \quad \begin{aligned} y_a &= dx_A + ey_A + f \\ y_a &= dx_B + ey_B + f \\ y_a &= dx_C + ey_C + f \\ y_a &= dx_D + ey_D + f \end{aligned}$$

Finding the affine transformation that exists between these parallelograms is equivalent to finding the coefficients {a, b, c, d, e, f}. Hence from the above equations, we have

$$\begin{bmatrix} x_{aA} \\ x_{aB} \\ x_{aC} \end{bmatrix} = \begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{ and } \begin{bmatrix} y_{aA} \\ y_{aB} \\ y_{aC} \end{bmatrix} = \begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix}$$

It is important to note that only 3 out of 4 equations mentioned above are needed to calculate the affine transformation. Coefficients 'c' and 'f' affect translation, 'a' and 'f' affect magnification and the combination affects rotation.

Appendix C

Back-Projection:

As mentioned in section 5.1, back-projection is a method used to estimate the error introduced by an algorithm by comparing the inverse transformation of a state space with the original state space on which the inverse transform is projected back.

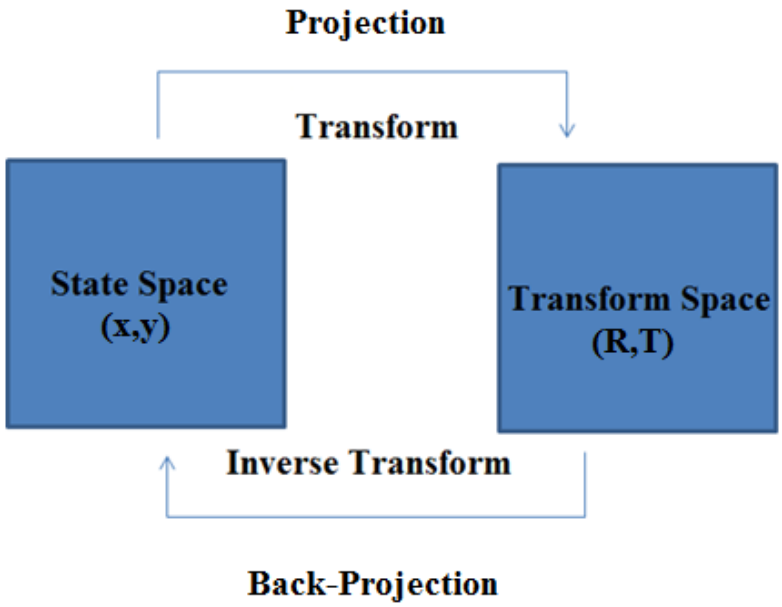


Figure V: Projection and Back-Projection space

In the current context, for an intensity function I(x,y), x and y denote the initial state space. That is a feature contained in I(x,y) has for instance state variables (x,y) representing the location of the feature. As the camera observes such a feature in its FOV, the PT is extracted. The corresponding differential rotation and translation vectors which denote the transform space can be extracted from this PT. These differential rotation and translation vectors are then utilised to calculate the corresponding inverse PT. This inverse perspective transform is finally made to re-project on the original feature. The error between the original feature location and the re-projected feature location for a certain interest point gives an estimate of the error introduced by the algorithm. Ideally the difference between the original marker and re-projected marker should be zero. In a practical case of trajectory estimation, as undertaken in this thesis, there will be a

series of measurements and projections. Back-projection can be used as a means of generating a scatter plot of the feature location in I(x,y) from which the statistics of the process errors can be determined.

Code

Good Features to Track code

Language: C
Libraries: OpenCV 2.1
IDE: Microsoft Visual Studio
Platform: Windows 7

```
int main (){
int i, corner_count = 100;
IplImage *dst_img1, *dst_img2, *src_img_gray;
IplImage *eig_img, *temp_img;
CvPoint2D32f *corners;
char imagePath[256] = "SecondImage.jpg";
printf("%s\n", imagePath);
dst_img1 = cvLoadImage (imagePath, CV_LOAD_IMAGE_ANYCOLOR | CV_LOAD_IMAGE_ANYDEPTH);
dst_img2 = cvCloneImage (dst_img1);
src_img_gray = cvLoadImage (imagePath, CV_LOAD_IMAGE_GRAYSCALE);
eig_img = cvCreateImage (cvGetSize (src_img_gray), IPL_DEPTH_32F, 1);
temp_img = cvCreateImage (cvGetSize (src_img_gray), IPL_DEPTH_32F, 1);
corners = (CvPoint2D32f *) cvAlloc (corner_count * sizeof (CvPoint2D32f));
cvGoodFeaturesToTrack (src_img_gray, eig_img, temp_img, corners, &corner_count, 0.01, 0.5);
for (i = 0; i < corner_count; i++)
cvCircle (dst_img1, cvPointFrom32f (corners[i]), 1, CV_RGB (255, 150, 0), 2);
printf("MinEigenVal corner count = %d\n", corner_count);
cvNamedWindow ("EigenVal", CV_WINDOW_AUTOSIZE);
cvShowImage ("EigenVal", dst_img1);
cvWaitKey (0);
cvDestroyWindow ("EigenVal");
cvReleaseImage (&dst_img1);
cvReleaseImage (&eig_img);
cvReleaseImage (&temp_img);
cvReleaseImage (&src_img_gray);
return 0;
}
```

Discrete Cosine Transform

Language: C
Libraries: OpenCV 2.1
IDE: Microsoft Visual Studio 2010
Version: 10.0.30319.1
Platform: Windows 7®

```
int main (){
int i;
IplImage* src = cvLoadImage( "image10.jpg");
IplImage* dst = cvCreateImage(cvGetSize(src),IPL_DEPTH_8U,1);
IplImage* gray = cvCreateImage(cvGetSize(src),IPL_DEPTH_8U,1);
IplImage* dstF = cvCreateImage(cvGetSize(src),IPL_DEPTH_32F,1);
IplImage* grayF = cvCreateImage(cvGetSize(src),IPL_DEPTH_32F,1);
while(1)
{
cvCvtColor(src,gray,CV_RGB2GRAY);
cvShowImage( "Source", src );
cvConvertScale(gray,grayF,1,0);
cvDCT(grayF,dstF,0);
cvConvertScaleAbs(dstF,dst,10,0);
cvNamedWindow( "dst", 1 );
cvShowImage( "dst", dst);
cvWaitKey(20);
}
cvWaitKey(0);
return 0;
}
```

HT Code For Image Sequence
Language: Matlab®
Version: 2012a 64-bit
Platform: Windows 7®
Libraries: Mexopencv[55]

```
clc;
clear all;
I  = rgb2gray( imread('image9.jpg') );
J = imfilter(I, fspecial('gaussian', [17 17], 5), 'symmetric');
BW = edge(J, 'Sobel');

%Perform Hough transform and show matrix
```

```
[H,T,R] = hough(BW);
imshow(imadjust(mat2gray(H)), [], 'XData',T, 'YData',R, ...
      'InitialMagnification','fit')
xlabel('\theta (degrees)'), ylabel('\rho')
axis on, axis normal, hold on
colormap(hot), colorbar

%# Detect peaks
P  = houghpeaks(H, 4);
subplot(2,1,1)

flag=[T(P(:,2)); R(P(:,1))]

plot(T(P(:,2)), R(P(:,1)), 'gs', 'LineWidth',2);
xlabel('\theta'), ylabel('\rho');
%# Detect lines and overlay on top of image
lines = houghlines(BW, T, R, P);
subplot(2,1,2);
imshow('image8.jpg'), hold on
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1), xy(:,2), 'g.-', 'LineWidth',2);
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
  plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
  plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
end
hold off

%%
figure(4), imshow('image9.jpg'), hold on
NL = length(P);
Nx = 640;Ny = 480;
for j=1:NL
theta = T(P(j,2))*pi/180; roe = R(P(j,1));
xo = roe*cos(theta);yo = roe*sin(theta);x1 = -sin(theta);y1 = cos(theta);
pp = zeros(4,3);
% Solve for x=1 intercept
t = (1-xo)/x1;
ya = yo+t*y1;
if ya>0 && ya < Ny
   pp(1,:) = [1,ya,1];
else
   pp(1,:) = [1,ya,0];
end
```

```
% Solve for x=Nx intercept
t = (Nx-xo)/x1;
yb = yo+t*y1;
if yb>0 && yb < Ny
   pp(2,:) = [Nx,yb,1];
else
   pp(2,:) = [Nx,yb,0];
end

% Solve for y=1 intercept
t = (1-yo)/y1;
xa = xo+t*x1;
if xa>0 && xa < Nx
   pp(3,:) = [xa,1,1];
else
   pp(3,:) = [xa,1,0];
end

% Solve for y=Ny intercept
t = (Ny-yo)/y1;
xb = xo+t*x1;
if xb>0 && xb < Nx
   pp(4,:) = [xb,Ny,1];
else
   pp(4,:) = [xb,Ny,0];
end

% Plot the line
ppp = [];
for jj = 1:4
   if pp(jj,3) == 1;
      ppp = [ppp;pp(jj,1:2)];
   end
end

plot(ppp(:,1),ppp(:,2),'LineWidth',2,'Color','green');

end
hold off;
%%
xy_int = [];
for j=1:NL
   for jj = 1:NL
```

```
      if jj~=j && P(j,2)~=P(jj,2)
         rj = R(P(j,1));
         thj = T(P(j,2))*pi/180;
         xoj = rj*cos(thj);
         yoj = rj*sin(thj);
         x1j = -sin(thj);
         y1j = cos(thj)
         Xj = [y1j,-x1j,x1j*yoj-y1j*xoj];

         rjj = R(P(jj,1));
         thjj = T(P(jj,2))*pi/180;
         xojj = rjj*cos(thjj);
         yojj = rjj*sin(thjj);
         x1jj = -sin(thjj);
         y1jj = cos(thjj)
         Xjj = [y1jj,-x1jj,x1jj*yojj-y1jj*xojj];
         XjXXjj = cross(Xj,Xjj);
         xi = XjXXjj(1)/XjXXjj(3);yi = XjXXjj(2)/XjXXjj(3);
         DD = [xi,yi]
         if xi>0 && xi<=Nx && yi>0 && yi<=Ny
           xy_int = [xy_int;[xi,yi]];
         end
      end
   end
end
figure(2);plot(xy_int(:,1),xy_int(:,2),'k.');grid on;
figure(1);hold on;
plot(xy_int(:,1),xy_int(:,2),'ko');
hold off;
```

HT Code For Real Time Video Sequence

Language: Matlab®
Version: 2012a 64-bit
Platform: Windows 7®
Libraries: Mexopencv

```
ptThresh = 0.05;
%hh = fspecial('gaussian',9,2);
hh=fspecial('gaussian', [17 17], 5);
obj = VideoReader('video5.wmv');
vid = read(obj);
frames = obj.NumberOfFrames;
for i=20:170
```

```
BW = imfilter(vid(:,:,i), hh, 'symmetric');
BW = edge(vid(:,:,i), 'Sobel');
%# Perform Hough transform and show matrix
[H,T,R] = hough(BW);
%# Detect peaks
P  = houghpeaks(H, 4);
figure(4), imshow(vid(:,:,i)), hold on
NL = length(P);
Nx = 640;Ny = 480;
for j=1:NL
theta = T(P(j,2))*pi/180; roe = R(P(j,1));
xo = roe*cos(theta);yo = roe*sin(theta);x1 = -sin(theta);y1 = cos(theta);
pp = zeros(4,3);
% Solve for x=1 intercept
t = (1-xo)/x1;
ya = yo+t*y1;
if ya>0 && ya < Ny
   pp(1,:) = [1,ya,1];
else
   pp(1,:) = [1,ya,0];
end

% Solve for x=Nx intercept
t = (Nx-xo)/x1;
yb = yo+t*y1;
if yb>0 && yb < Ny
   pp(2,:) = [Nx,yb,1];
else
   pp(2,:) = [Nx,yb,0];
end

% Solve for y=1 intercept
t = (1-yo)/y1;
xa = xo+t*x1;
if xa>0 && xa < Nx
   pp(3,:) = [xa,1,1];
else
   pp(3,:) = [xa,1,0];
end

% Solve for y=Ny intercept
t = (Ny-yo)/y1;
xb = xo+t*x1;
if xb>0 && xb < Nx
```

```
   pp(4,:) = [xb,Ny,1];
else
   pp(4,:) = [xb,Ny,0];
end
% Plot the line
ppp = [];
for jj = 1:4
   if pp(jj,3) == 1;
      ppp = [ppp;pp(jj,1:2)];
   end
end
plot(ppp(:,1),ppp(:,2),'LineWidth',2,'Color','green');
end
hold off;
xy_int = [];
for j=1:NL
   for jj = 1:NL
      if jj~=j && P(j,2)~=P(jj,2)
         rj = R(P(j,1));
         thj = T(P(j,2))*pi/180;
         xoj = rj*cos(thj);
         yoj = rj*sin(thj);
         x1j = -sin(thj);
         y1j = cos(thj)
         Xj = [y1j,-x1j,x1j*yoj-y1j*xoj];

         rjj = R(P(jj,1));
         thjj = T(P(jj,2))*pi/180;
         xojj = rjj*cos(thjj);
         yojj = rjj*sin(thjj);
         x1jj = -sin(thjj);
         y1jj = cos(thjj)
         Xjj = [y1jj,-x1jj,x1jj*yojj-y1jj*xojj];

         XjXXjj = cross(Xj,Xjj);

         xi = XjXXjj(1)/XjXXjj(3);yi = XjXXjj(2)/XjXXjj(3);
         DD = [xi,yi]

         if xi>0 && xi<=Nx || yi>0 && yi<=Ny
         xy_int = [xy_int;[xi,yi]];
         KK = unique(xy_int, 'rows');
         cc=sum(KK)/length(KK);
         end
```

```
        end
    end
end
figure(1);hold on;
plot(cc(:,1),cc(:,2),'k*');
hold off
pause(0.1);
end
```