TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

WAQAS ABDULLAH

VISUAL ODOMETRY: FEATURE BASED TRACKING AND VELOCITY ESTIMATION BASED ON GROUND LOOKING CAMERA

Master of Science Thesis

Examiner: Professor Kalevi Huhtala
Dr. Reza Ghabcheloo
Examiner and topic approved by the Faculty Council of Engineering Sciences meeting on 06 March 2013.

Preface

The Master’s thesis work is carried out in the Department of Intelligent Hydraulics and Automation, Tampere University of Technology, Finland. First of all, I would like to thank The Almighty for giving me the courage and knowledge to do this research work. Secondly, I pay my deepest gratitude to Professor Dr. Kalevi Huhtala for providing me with the opportunity to work in the diversified and vivacious group of Mobile Machine and also for examining the work. Next, I would like to thank my mentor and supervisor, Dr. Reza Ghabcheloo, from the depth of my heart for not only introducing me to this topic but also for his long term support, guidance and advice throughout the research work. It could not have been done without his support, supervision and patience with my work, especially with my writing. I am also thankful to Dr. Mika Hyvönen, Mohammad Mohammadiaref and Antti Kolu for their positive feedback and motivation they had provided me during my work.

I am highly indebted to the people and Government of Finland for providing with the opportunity to study here and perform this research work. I am also thankful to all the people of the Intelligent Hydraulics and Automation group for providing me with such kind of conductive environment for the research work, especially my office colleagues for their kind support and helping hands.

I would also like to thank all my friends in Finland, Pakistan and everywhere else. It was due to their constant moral support and encouragement that I am able to perform this work. Special thanks to all my Pakistani friends in Tampere whose nice company, support and get-together never let me feel lonely and away from my home.

Last but not least, I would like to pay my deepest gratitude to my loving family especially my Parents (Shahnaz Mansha and Inayat Ullah Gohar) whose kindness, care, encouragement and unlimited love helped me during my studies here and made me able to do this work.

I would like to dedicate my thesis to my niece Zainab Duaa (late).

Tampere, Finland, February 05, 2013

Waqas Abdullah

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Machine Automation
ABDULLAH, WAQAS: Visual Odometery: Feature Based Tracking and Velocity Estimation Based on Ground Looking Camera
Master of Science Thesis, 64 pages, 6 Appendix pages
March 2013
Major subject: Mechatronics
Examiner: Professor Kalevi Huhtala and Dr. Reza Ghabcheloo
Keywords: camera, corner detection, corner matching, odometery, computer vision, robotics, optical flow.

The computer vision field has close relation with autonomous robotics which is in use for several purposes which include mobile robot vision tasks, robot navigation, motion trajectory, object detection and tracking and security surveillance tasks. There are many techniques available which are being used for completing these tasks but many of them lead to problems like low resolution, limited applicability and high capital investment. This thesis examines the way in which we could achieve results within some tolerance level of accuracy and low cost. Visual odometery is one of the main objectives of this thesis which is achieved with simple and practical method. The algorithm is developed to estimate the velocity using a corner detection technique based on ground looking camera.

The thesis is divided into three main parts. In the first part of the thesis, literature review and previous work done in the relevant field is explained. The theoretical background of the topic is also described in the first part of the thesis. Second part of the thesis demonstrates the development of the algorithm, pre and post processing and implementation of the algorithm. Last part of the thesis describes the different test environments where the developed algorithm is implemented. The test environments are further classified into two main categories. Conclusions, results, problems faced during the whole process and future tasks are also included in the last part of the thesis.

The study indicates that, selection of the right pre-processing parameters can enhance the results quality. At the same time by providing the appropriate illumination for the camera system can also increase the efficiency of the outcome. This research and developed algorithm has the potential to be used for further implementation at commercial level by changing some necessary parameters in the algorithm and implementation. This research could be more useful by implementing addressed future tasks in Section 5.3.2, in order to achieve higher efficiency in the results. Implementation of all necessary parameters explained in this thesis and by considering future tasks will make this research more effective and beneficial for the business.

CONTENTS

1. INTRODUCTION

1.1. Background

It is important in the field of autonomous robots to estimate the velocity of mobile robot within some tolerances in order to achieve higher-level motion and navigation tasks accurately. Image processing is one of the tools that are used for this purpose. The visual odometry process consists of three main stages: *feature detection, feature matching and image transformation*, as shown in the Figure 1.1.

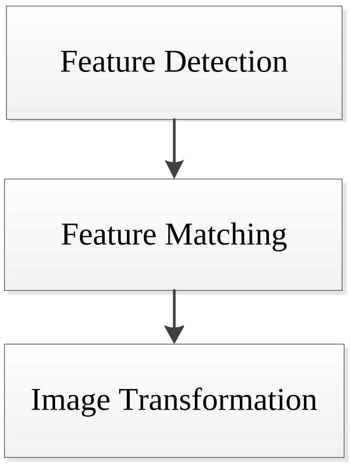


Figure 1.1: *Different stages for velocity calculation using image processing*

To calculate translation from on image to the other, the first feature detection. In this step a set of features e.g. corners, blobs are detected in both images. The most required property of the feature detection is repeatability, i.e. the method must find the same feature points in the processed images. Feature points are presented as feature vector (descriptors). Descriptors have to be unique and at the same time robust to noise. In the feature matching stage, the feature vectors are compared and matched between two images (current image and previous (second) image). The feature matching is based on a distance, e.g. the Euclidean distance [48] between the feature vectors. In the 3rd stage, image transformation is calculated from the result of feature matching.

This process is different from conventional “content-based image retrieval (CBIR)” [49], where the features have the same properties in all the images, e.g. red colour is considered as the point of relevancy between the images. So the difference is of colour relevancy and features detection in the process of CBIR and feature detection respectively.

Feature detection and feature matching have been the subject of research in different fields, namely mobile robot vision [50, 51], model based object recognition [52], panorama stitching [53, 54], video sequence matching [55], scalable object recognition [56].

Different types of image features are studied, for example, corners [57], interest-points [58], and moment invariants [59]. These all image features are declared as, depending upon their definition criteria. Corner points are the most significant features point in all of them. The "interest point" and "corners" are used equivalently in the literature but corners are also presented as the special subclass of interest-point [60]. Detail of corners is mentioned in Section 2.3.

1.2. Objectives

The main objectives of this thesis are as follows:

Visual Odometery: In an autonomous navigation, the position estimation of mobile robot is an essential part. Wheel encoder based odometery is the first method used for dead-reckoning for wheeled ground vehicles. Dead-reckoning is the process of determining the current position by using the data of previous position. In this process the error is accumulated to the current position. Despite of its error accumulation and error computation due to wheel slippage, it is still one of the main localisation sensing tools in the majority of ground vehicles [61]. Reason for the popularity of this method is simplicity and practicality [62]. In our case, calculation for odometery is based on visual data. Main idea is to find good feature points in one frame with corresponding features in the next frame and the transformation calculation from these features points. Method is carried out using single camera looking downwards and related work example is mentioned in [63].

Velocity estimation: Using idea of features detection in one frame and corresponding features in next frame, velocity estimation is also calculated in this thesis work. Distance between detected features in current frame and the corresponding features in the second frame is the key for the velocity calculation-using frame per second as a time factor.

Application development: In order to achieve the above mentioned objectives, the application is developed using MATLAB Simulink and tested with two test environments, i.e. real image with simulated motion and real image with real motion.

1.3. Dissertation outline

Chapter 2 describes the basic of terminology of image processing and fundamentals of corner detection. Another idea and technique which is used for the speed estimation is optical flow. An overview of optical flow algorithm is presented in chapter 3. Developed application using corner detection algorithm is presented in chapter 4. Chapter 5 contains implementations and practical results and conclusions for the proposed algorithm and is followed by the references of the research work.

2. FUNDAMENTALS OF CORNER DETECTION

2.1. Introduction

Recent years of research have witnessed major advancement in the field of autonomous mobile robot systems. Various techniques and concepts have increased the research interest in this area. Many techniques and methods which had been used in the field of machine vision requires matching of two consecutive image frames in order to extract information from them. Corner detection is one of the leading concepts of interest in the field of machine vision used for autonomous robot systems.

For example, it is possible to find out speed of the platform on which the camera is mounted, tracking of the object in the images, shape analysis, motion analysis etc. if feature matching is successful between two images frames. The matching of all the pixels from one frame to the next frame will be computationally very expensive; hence this is prohibitive for most of the applications. In order to overcome such situation some locations in the image are used to point out which are known as feature points or corner points. These corner points then execute association between two consecutive image frames. This will reduce the association point's computational time for further processing; however, corner detection time is still needed be to consider. Approximate reconstructions of the shape from corners are also one of the important aspects of corner detection that make it stand out among other techniques of machine vision [29-33].

2.2. Image

Image is acquired when the illumination coming from the source is reflected from the scene. Hence, image is the 2-D function relying on illumination source and reflection of the scene, which is captured and is mentioned in [64].

$$f(x,y) = i(x,y)\; r\;(x,y)$$

where, $x \, and \, y$ are (2-D spatial variables) denoting spatial coordinates on the image, $i(x, y)$ is the illumination function and $r\,(x, y)$ is the reflectance function. When the image is digitized, it is possible to represent it with matrix with m rows and n columns. Each element of this matrix is known as pixel [64]. Digital image can be, grey scale, colour and/or binary. In this thesis work, grey scale image is used because of the corner detection algorithm requirement. The grey scale image has the luminance intensity information only, without any colour information. If each pixel is represented by a b bit number, in this case each pixel in the image has certain value of intensity in the range of 0 to $2^b - 1$ [63]. A pixel in the image is the element, which contains small variations in intensity among themselves, and large variation from other groups of pixels. This pixel variation among themselves and from other groups of pixels is termed as object rising phenomenon [65].

2.3. Corners

Interest points and corner points are termed as same in the literature [60]. Guru et al [66] defined corners as the intersection of two contiguous, comparatively straight curve segments. A main criterion of confirmation for a point as corner is that the direction of the curve changes significantly and sharply. The corners are shown in following two image examples:

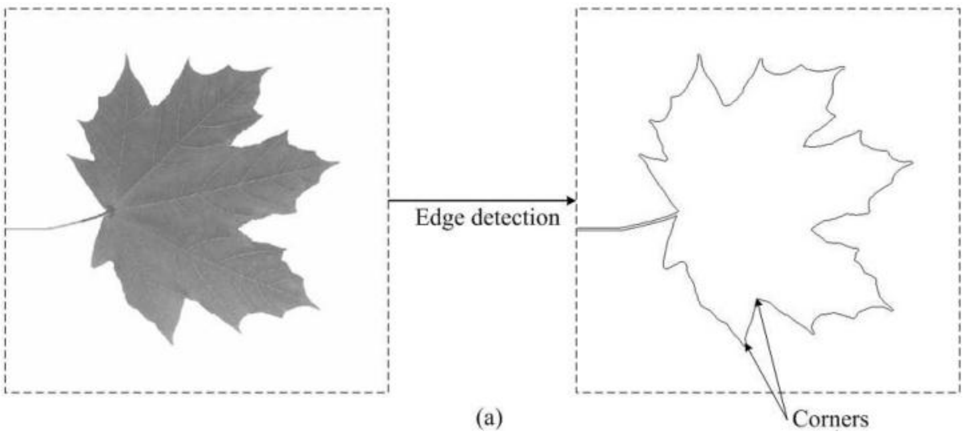


Figure 2.1: *(a) Leaf image available at [67] original image on the left hand side and edged image with corners on the right hand side [copied from source 16]*

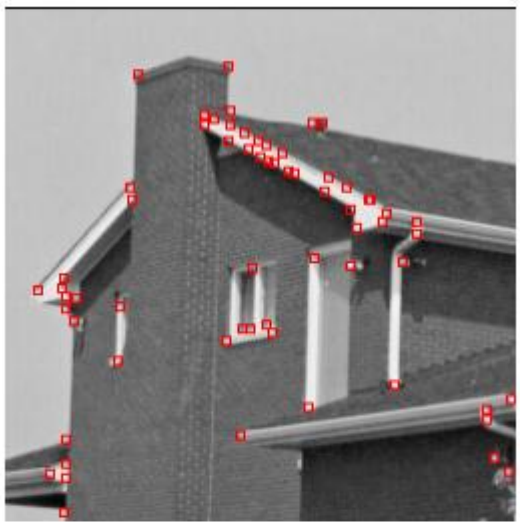


Figure 2.2: *Detected corners at test image [copied from source 11]*

2.4. Corner detectors

There are many different corner detectors proposed depending upon their definitions for corners point in an image. Some of the corner detectors find corners on the basis of local symmetry, some of them work to locate the corners and others find out the changes in the texture. Corner points are interesting points because they define the edges and came into existence when two or more edges of the two different or part of the same objects define its boundaries. The motions at edges are ambiguous by virtue of the aperture problem [34] but corners being 2D feature do not cause such ambiguity specifically in motion analysis.

2.4.1. Corner detectors requirement

Corner detector should meet the following criteria's in order to reveal correct result,

- True corners should be detected
- Removal of false corners and/or outliers
- Corner points should be well localized
- Corner detector should be robust with noise
- Corner detector should be computationally efficient

There are three main and popular corner detection algorithms available, which are namely as follows [71];

- Shi & Tomas detection method (minimum eigenvalue) [69-70]
- Harris & Stephen corner detection [8]
- Rosten & Drummond method (local intensity comparison) [10]

All of the above stated methods have their own significance with respect to the criteria they use to find the corner points.

2.4.2. Harris & Stephen corner detector

This is a popular corner detector in the field of machine vision because of its strong invariance to illumination variation, scaling, rotation and image noise [7]. The working principle of the detector is based on a matrix related to the local auto-correlation function of the signal and this function calculates the local changes of the signal from small amount of the moment of pixel in different directions. After auto-correlation operation is execution, the two eigenvalues are calculated for each pixel. From the eigenvalues a number or point is associated to each pixel in the image [9]. If this number or points is above certain threshold value, then the corresponding pixel is defined as corner. The whole method and criteria on which corners are allocated in an image is described briefly in [8].

2.4.3. Shi & Tomasi corner detector

Shi & Tomasi corner detector is entirely based on the Harris & Stephens corner detector but with some changes in the detection method and its criteria. Shi & Tomasi proposed direct use of eigenvalues instead of function manipulation with the points. The results are more accurate.

A score is associated to each pixel in the coordinate (x, y) on the image. The calculation of score in Harris and Shi & Tomasi is as follows;

For Harris & Stephens [9]:

$$R = detM - K(traceM)^2$$
$$detM = \lambda_1\lambda_2$$
$$traceM = \lambda_1 + \lambda_2$$

For Shi & Tomasi [9]:

$$R = \min(\lambda_1, \lambda_2)$$

where M denotes the covariance matrix or scatter matrix, K is an adjustable constant and known as sensitivity factor. R denotes the score, $\lambda_1\lambda_2$ are the eigenvalues. Notice that M is 2x2 positive definite matrix, thus the eigenvalues are non-negative values [35, 36]. From values of λ_1, λ_2 we can define whether a pixel is associated to an edge or a corner point. There are three cases to be considered [36]:

- If both $\lambda_1\ and\ \lambda_2$ are small, local auto-correlation function is flat and targeted image region is of approximately constant intensity. The change in the matrix M(x, y) is little in any direction.
- If one eigenvalue is high and the other is low local auto-correlation is ridge shaped. It shows little change in matrix M(x, y) in the direction of ridge and significant change in the orthogonal direction, which will indicate an edge surface.

- If both eigenvalues are high, then the local auto-correlation function is sharply peaked then the change in matrix M(x, y) in any direction will be significant, which indicates a corner.

2.4.4. Rosten & Drummond corner detector

It is an algorithm proposed by Rosten and Drummond for recognizing the features or interest points in an image. An interest point or feature in an image is a pixel which has a well-defined position and can be robustly detected. This corner detector is known as FAST corner detector where FAST is an abbreviation of Features from Accelerated Segment Test. Typically like the name of this corner detector it is recognized as fast corner detector as compared to other available corner detectors [10].

2.5. Corner detection steps

Most of the corner detectors follow same steps to detect interest or corner points in an image. Those corner detectors are as follows [11];

- Moravec
- Harris/Plessey
- Trajkovic and Hedley (4-neigbhours)
- Trajkovic and Hedley (8-neigbhours)

Mainly corner detection is done in three consecutive steps:

1. Corner operator application
2. Threshold application
3. Non-maximal suppression

Corner operator application: This is the first step in which a corner detector is supplied with an image as input. Corner operator requires few parameters to determine whether the pixel in an input image is a corner or not. The “cornerness measure [11]” is the measure which simply defines the degree to which corner operator consider the pixel as corner in an input image. All the corner detectors manipulate this first step and they only differ with each other on the basis of measure used for manipulation. The output image of this step is named as “cornerness map [11]”. Output image is an image with the same dimension as input image but it could be consider as processed version of the input due to corner operator’s application to each pixel of the input image.

Threshold application: Corner operators define the corners as local maximum in the output image (cornerness map) of the first step. At the second step there are many local maximum in the cornerness map which contain small cornerness measure and hence are not true corners. In order to avoid these un-true corners for further processing threshold application is used. All the local maxima under threshold value are set to zero and thus the image is considered as thresholded cornerness map. The threshold selection requires repeated judgments and error experimentation tests. The threshold value should be consider high enough to avoid un-true corners and low enough to hold true corners, therefore, trade-off policy needs to be adopted in this scenario. It is very rare to find ideal threshold value to have ideal corners in an image.

Non-maximal suppression: The non-maximal suppression is applied to define the location of the local maxima in the thresholded cornerness map. At this stage the cornerness map contains the non-zero values around local maximums which are declare as corners. Cornernerness measure for each pixel in the thresholded cornerness map is set to zero by non-maximal suppression if its cornerness measure is not larger than the cornerness measure of all other points within a certain distance. After non-maximal suppression application, the remaining points are non-zero points in the cornerness map and are true corners.

Following flow diagram shows how these steps are processed;

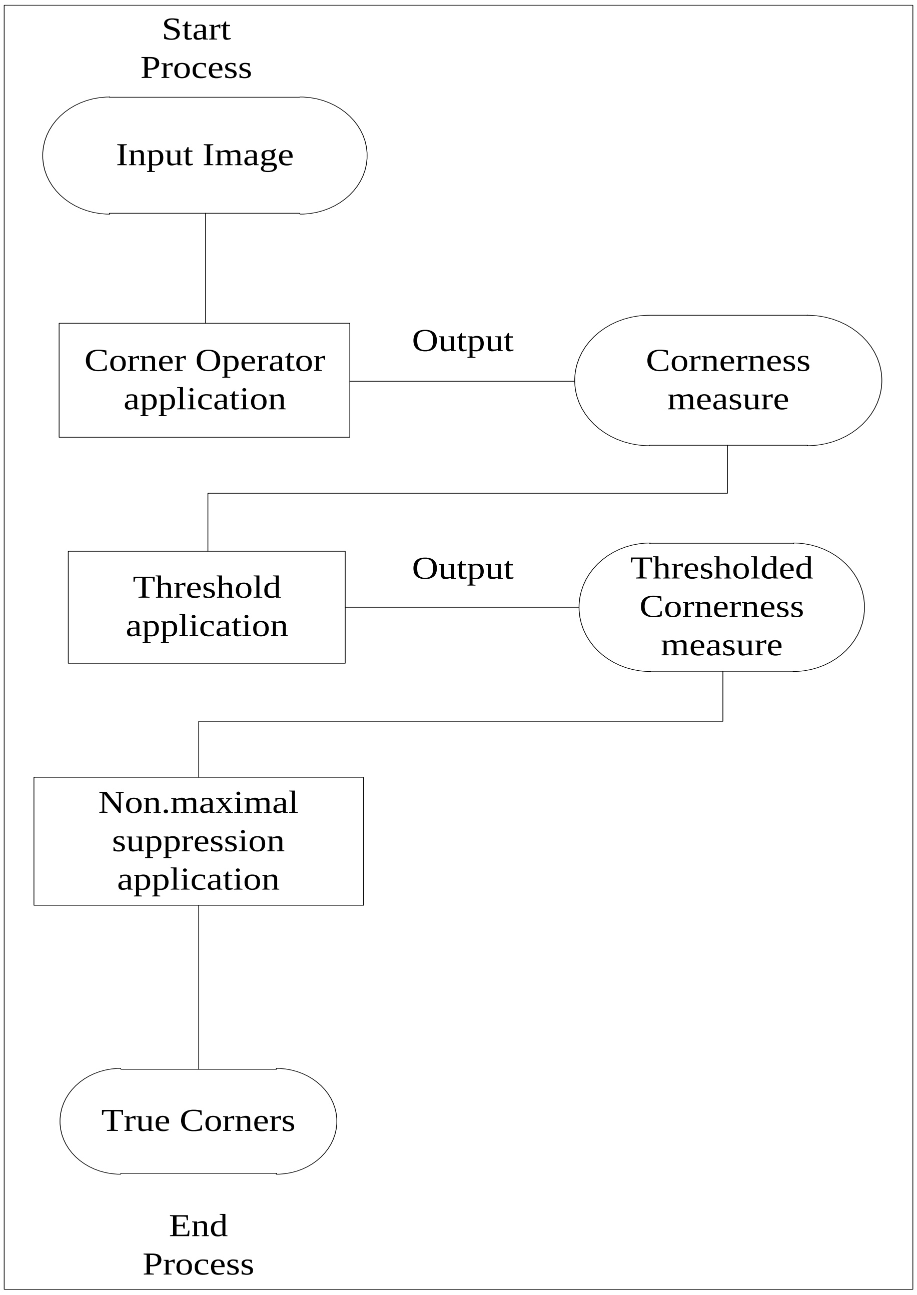


Figure 2.3: *Flow diagram for corner detection [11 modified]*

The visual presentation of each step of the corner detection is mentioned in [11] and is as follows:

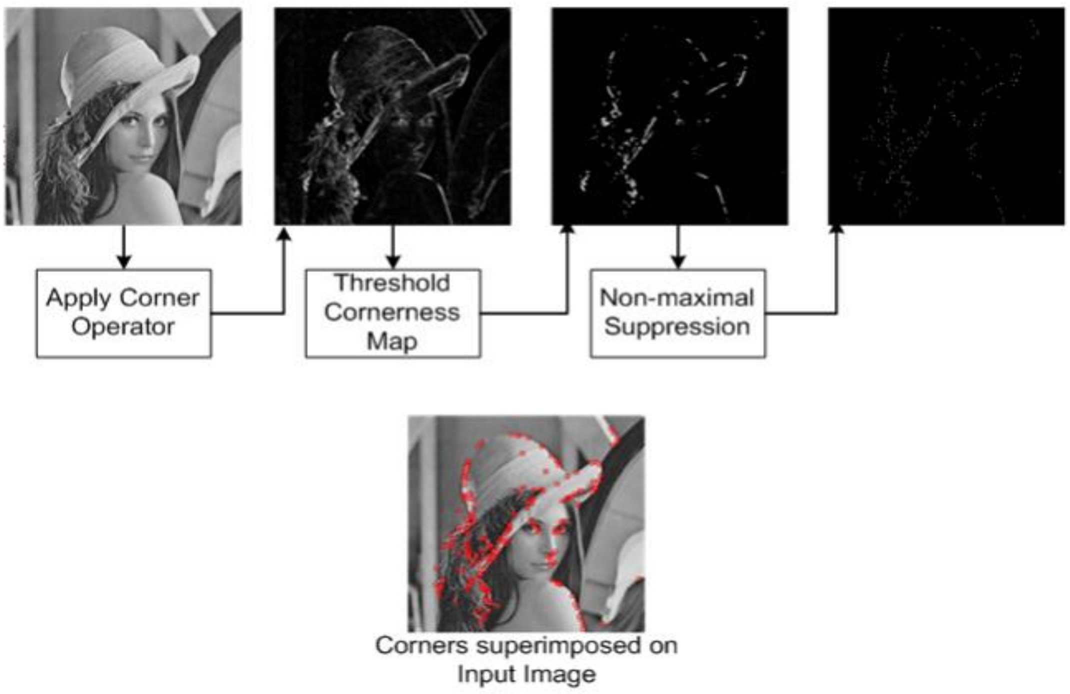


Figure 2.4: *Pictorial representation of corner detection step at Lena's image [11]*

2.6. Corner matching

The problem of corner matching between two images and at the same time, avoiding mismatches is an ultimate problem in machine vision application and it has been studied for couple of decades [13]. After an intensive research in this domain, there are noteworthy numbers of algorithms available for this purpose but no one is universal and reliable enough to survive with all situations and work environment [14]. The difficulties might be the corner detection and then the relative matching between two images. The solution for such problems might be computationally expensive and the algorithm could generates some false corners due to noise in the images and even algorithm might lose true corners due to the noise caused by the illumination and the corner strength which if presented with some numerical value could be tentative [15].

2.6.1. Corner point representations

Corner point representation is one of the important aspects in the corner matching. In corner matching each feature must be represented with its unique information. The more the information is unique, the less is the chance to have false matching at matching stage. The set of information about feature is known as descriptors.

The descriptors are categorized into two types and are as follows;

- Geometric information
- Neighbourhood intensity

Geometric representation of the feature consists of curvature, angle and the distance from its neighbouring corner points. Geometric representation is used in [37].

Neighbourhood intensity representation is based on pixel intensity values in a specific neighbourhood around each corner point. Neighbourhood representation is knows as local descriptors. Mikolajczyk and Schmid [25] presented the comparative study of local descriptors. Examples of such descriptors are differential invariants [38-40], central moment invariants [41], SIFT-based descriptors [42, 43] and steerable filters [44] among others.

2.6.2. Image matching Categories

The corner matching is generally divided into two categories that are "area based and feature based" corner matching [17, 18]. Following pictorial presentation will show the tree diagram for the corner matching subdivision categories.

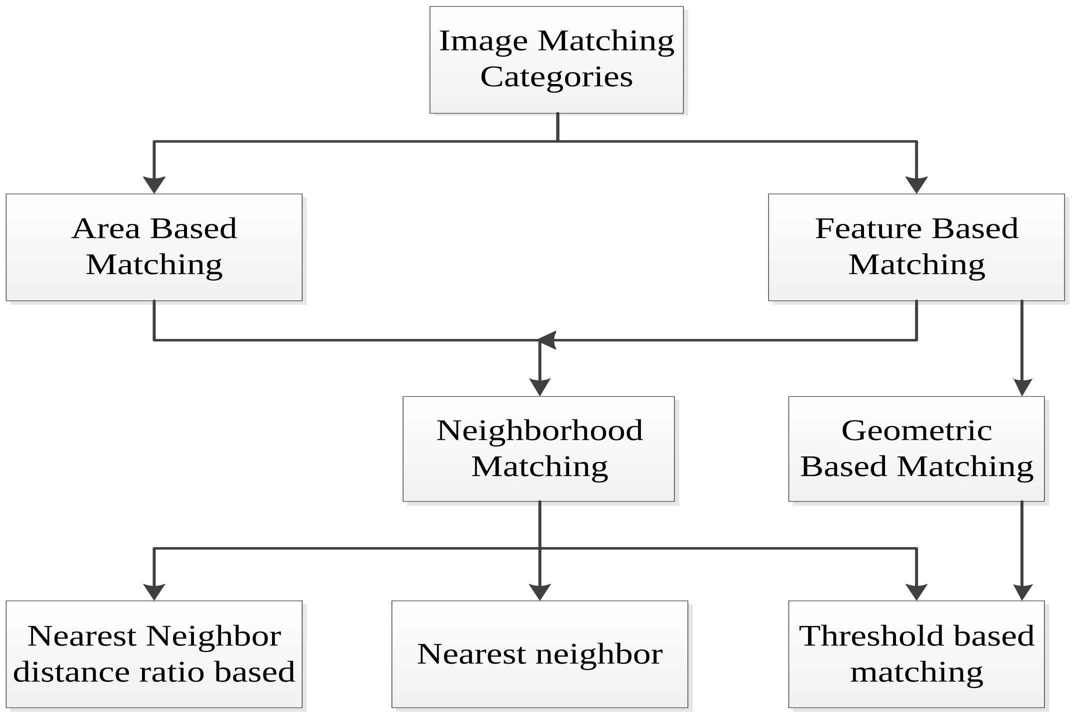


Figure2.5: *Image matching categories and sub-division [modified 16]*

2.6.2.1 Area based matching

This method will use grey patch area of the current image to relate with the previous image using cross-correlation or least-square technique. The area based matching algorithm assigns some value to the grey patch in the current image and tries to locate related value in the second image in order to have ultimate matching between two images. This method needs image pre-processing in order to avoid or minimize sensitivity to noise within the image. One of the drawbacks of this matching is that it is computationally expensive.

2.6.2.2 Feature based matching

Feature based matching entirely depends upon the features, e.g., corners, from the current image and relate those features to those in the second image. The correspondences between the features of two images rely on their descriptors information. This type of matching is more robust and reliable than their counterpart that is area based matching. But they could also be computationally expensive if large numbers of features have been encountered in the images.

2.6.2.3 Neighbourhood matching

Combination of aforementioned matching methods has also been used in the matching algorithms. They are named as neighbourhood based matching as shown in the Figure 2.5. This method uses information around the features for matching. Example of such implementation is present in [18-22]. One of the research articles [25] contains further sub-division of neighbourhood based matching techniques. In [25] Mikolajczyk and Schmid use descriptor information for matching. These matching techniques are as follows;

- *Threshold* based matching: In the threshold based matching, threshold is the decisive element for the selection of features matching. It works in a way that two features (feature from current and second image) are matched if the distance between their descriptors is less than certain predefined threshold. It is quite possible for the feature to have many matches in this strategy.

- *Nearest neighbour* based matching (NN matching): In the nearest neighbour based matching the decisive element is the distance between the two descriptors. If the features are corner points and named as C_1 and C_2 (corner points from current and second image) with descriptors D_1 and D_2 respectively. These descriptors will match if D_2 is the nearest neighbour of D_1 and the distance is beneath the threshold. Using this approach corner point should have at most one match.

- *Nearest neighbour distance ratio* based matching (NNDR matching): This technique is same as aforementioned technique. Certain threshold is defined before the matching. The threshold is applied for the distance ratio for the first two (1st and 2nd) nearest neighbour matches. The two corners C_1 and $C_{2,1}$ with descriptors D_1 and $D_{2,1}$ respectively will be matched if $\frac{|D_1 - D_{2,1}|}{|D_1 - D_{2,2}|} < T$, where $D_{2,2}$ is descriptor of $C_{2,2}$, 2nd nearest neighbour match-to C_1. This technique will also have at most one match.

All the three matching techniques are different from each other in a way that the first one is relying on threshold, second approach is relying on the distance between descriptors and the third matching method is deciding upon the distance ratio between descriptors.

2.6.2.4 Geometric based matching

Matching methods which do not use neighbourhood information around features are categorized as geometric point matching methods. Research papers [23-24] are the example of such implementation.

Geometric based matching use the set of correspondence element in order to decide the ultimate matching between the features. The set of correspondence descriptors consists of curvature, distance from the neighbour features, angle etc. The set of correspondence descriptors finally decide for best matches while depending upon the threshold value.

2.7. Removal of outliers

Outliers are the points which are true corners in an image and may lead wrong feature matching. In order to have true corner points in an image and then to match the true corners of current image to the second image, outlier's removal is necessary. RANSAC abbreviated as "RANdom SAmple Consensus" [26] algorithm is one the method for removing outliers. Example of RANSAC based outliers removal is noted in [27].

3. OPTICAL FLOW ALGORITHM, AN OVERVIEW

Optical flow is a concept, which describes the apparent motion of the objects in an image by the relative motion between an observer or eye or visual sensor (camera) and the scene or environment. Optical flow provides information about the environment of the image, which could be used for the control purpose. In particular, optical flow has the information of observer’s movement [5], distance travelled from one image to the next image, 3-D shape of environmental shape [6].

Optical flow is also one of the leading concepts of interest in the field of machine vision used for autonomous robot systems. Optical flow has a long history of research. This term was studied and introduced by Gibson J.J in 1950’s. He published this concept in 1950 [1] and later in 1960 [2]. In the history of research about optical flow most of the time it is argued that all the methods discovered some variation of the same subject. Mainly, the research in the field of optical flow revolves around two concepts, that is: brightness pattern and spatial arrangements [3]. The brightness pattern is typically derived from the study that surfaces normally persist over time and hence change in the position cause no change in the value of intensity of small region [4]. The spatial smoothness constraint (spatial term) reveals that neighbouring pixels generally belong to the same surface and so have almost the same image motion [3]. These terms in optical flow have a long history but despite of it, there have been few attempts to research about these terms [46]. Latest research [47] in the field of optical flow has provided image sequences with ground truth optical flow available to make this practical.

3.1. The Optical Flow field

The pattern created with the moment of the visual sensor (camera) relative to the environment is an optical flow field. It is represented by the instantaneous velocity field, where every vector corresponds to the optical motion of a point in the environment. Generally, rigid motion is decomposed into two components that are-translation and rotation which state that the pattern created with the relative moment of a visual sensor projected at the spherical shape object has two components: a rotational component depending upon the rotational flow due to the rotation of an observer whereas the translational component is due to translation of the observer. The following figure shows the phenomenon;

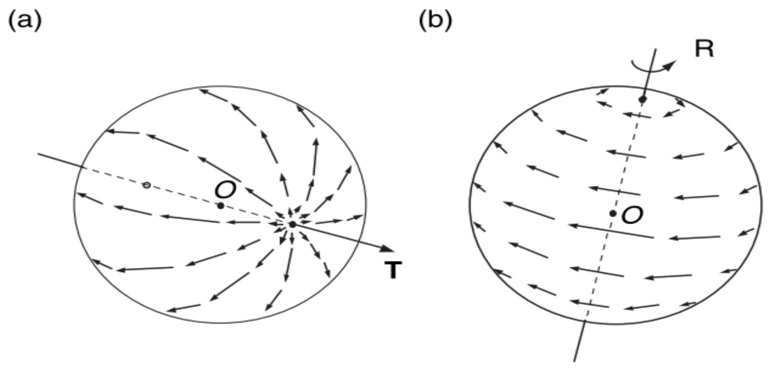


Figure 3.1: *(a) Translational component (b) Rotational component*

Above Figure 3.1 (a) is revealing the translational component of the optical flow field where an observer is making translation along the axis. Figure 3.1 (b) shows the rotational component of the optical flow field. This is due to the observer's eye rotation along the axis producing cylindrical flow along the spherical object without sources. These concepts are explained in detail below.

3.1.1. Translational component

It is important to consider an extraction and contraction of the focus with the moment of an observer while discussing translational component of optical flow. With the moment of an observer along the axis, focus usually expands in the direction of moment and contraction takes place in the opposite direction. The optical flow field pattern formed with the direction of the vector is dependent in the direction of moment of the observer. The environmental distance of the moment has no effect on the pattern of the optical flow field. This is the reason we can simply identify the direction of the moment by an observer. The distance from the matching point, velocity of the observer's moment and the angle from the focus of expansion reveals the magnitude of each vector. So, it is possible to reveal relative depth from the information delivered by the magnitude of the optical flow and from the moment of the observer.

It is very forthright to find out the direction of the optical flow field from the flow pattern of the translational component. In order to have consistent estimation for the focus of expansion the pair of vectors can be arranged in a triangular manner so that their point of origin could be determined and hence they will produce the focus of expansion.

3.1.2. Rotational component

The rotational component illustrates the flow pattern as solenoidal due to the rotary motion of the observer’s head or eye and in the case of camera the rotary motion due to rotation of the camera lens. Moreover, the parallel flow pattern is due to the yaw of the observer and rotation about the line of sight produce rotational flow pattern. This rotational pattern of the optical flow field brings information about the rotational moment of the observer as well as it is sovereign of the distance factor. The direction of the observer is possible to illustrate by the vector direction which is relying on the axis of rotation.

The observer’s rotation rate depends upon the vector magnitude which is possible to reveal with the visual angle from the rotation axis and the speed of rotation. This organized and related space of flow pattern in the optical flow field allows determining the direction and speed of the observer.

3.2. Optical flow algorithm

Optical flow is the apparent velocities of movement of brightness patterns which is distributed in an entire image.[1] Optical flow is always generated with the relative moment of the object and observer which becomes the reason for changes in the brightness pattern and hence produce optical flow. Horn [1] developed the algorithm or optical flow techniques, which are based on spatiotemporal difference from consequent images. Since, these methods are developed as differential method, region based matching method etc. are used for the optical flow calculation. It is possible to enlighten the spatial arrangements of the object viewed with optical flow concept and at which proportion these spatial arrangements are changing [2].

Optical flow algorithm for the velocity calculation of mobile robot is divided generally into following steps.

- Selection of tracking context
- Image acquisition
- Estimation of optical flow across the image sequence using Kanade-Lucas-Tomasi (KLT) algorithm for selection of features
- Selected features are then tracked in the later images
- Storing the results of feature tracking
- Replacement of the old image with the new image
- Velocity calculation

The optical flow algorithm calculates the changes in the moment between two frames of images which are taken at time t and t+Δt. [1]. In this method, suppose J is the grey scale image captured at time t and J+1 is the next image captured at time t+Δt.

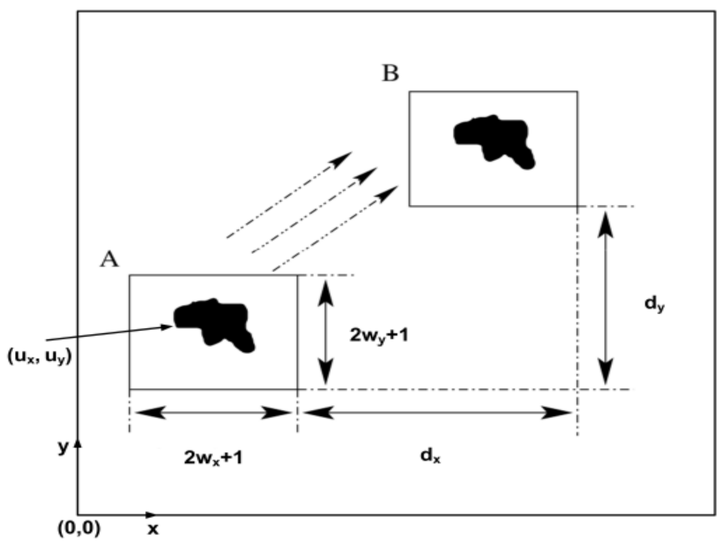


Figure 3.2: *Optical flow features [3]*

In Figure 3.2 (A) is the feature window in the grey scale Image J and (B) is the same feature window in the grey scale image J+1. The feature (A) has moved distances of dx and dy in X and Y coordinates respectively. Figure 3.2 is presenting the moment of the feature (A) and (B) between two sequences of images. The main objective is to find the displacement between features (A) and (B) and the the velocity vector is determined using optical flow output. Constraints like geometrical (shape) comparison, grey scale intensities comparison and/or feature from one image to be match with the feature in second image are the possible way to determine the optical flow pattern.

Calculation part starts with the calculation of pixels per centimetre just by moving the object manually at known distance. Then the pixel values have been converted to real world coordinates and general gain factor has been established.

Displacement from image J to J+1 is determined with feature tracking algorithm. The features in the image J are selected by the gradients calculation when the image is smooth and these gradients are one in x direction and one in y direction. Then these selected features are tracked entirely based on tracking context. The tracking context consists of “min-dist”, “min_eigenvalue” and “point_threshold”. Where, “min-dist” is the minimum distance between features, min_eigenvalue is the minimum allowable eigenvalue for the features need to be track. The eigenvalue is selected in a way that its minimum value remains above the threshold. These features need to be within the border (x-axis) and border (y-axis) of the image.

The displacement from x and y coordinates of features are being selected and then tracked which gives the optical flow d which is then divided by the time of frame per second to calculate the velocity of the mobile robot.

3.3. Results analysis and problems

The code for the optical flow algorithm is mention in Appendix 1. With provided code, the tracking context mention in Section 3.2, the image acquisition, selection of features, the tracking of selected features and velocity calculation has been done.

The test setup was same as it is explained in Section 5.3. The test was done in such a way that both the camera and the object remain at same place i.e. there was no moment. In an acceptable test results, there should be no change in the features position, neither in X nor in Y-coordinates. Following plots shows the position of features in X and Y coordinates separately.

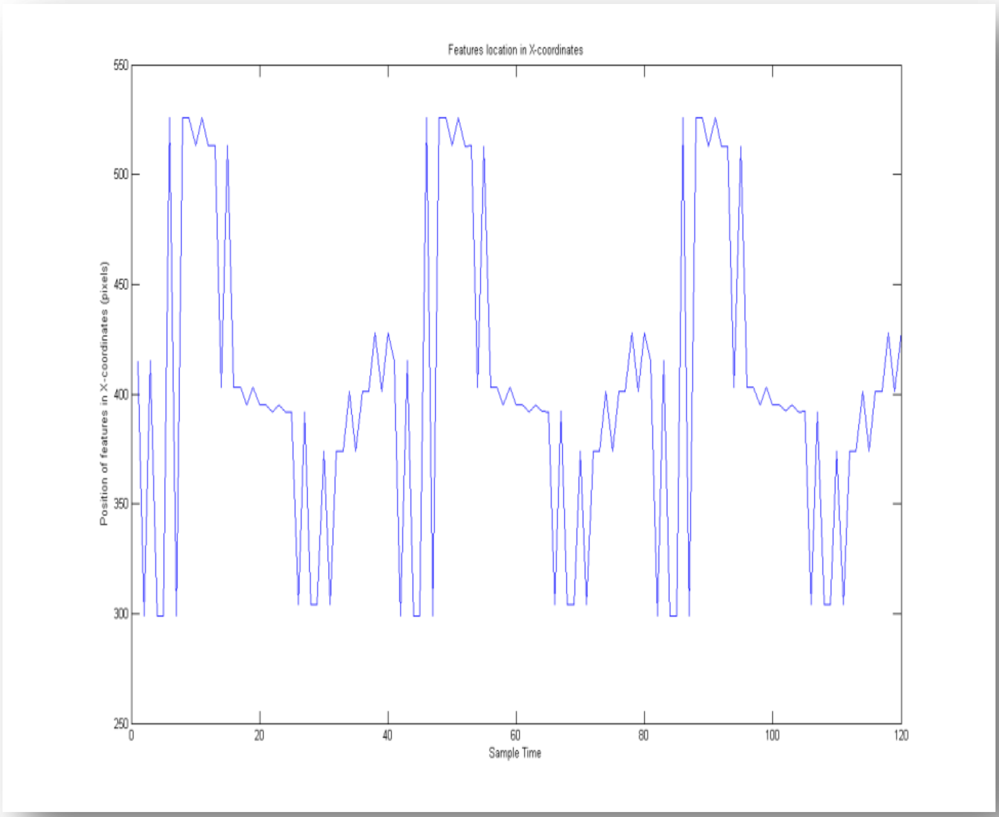


Figure 3.3: *Feature's position in X-coordinates*

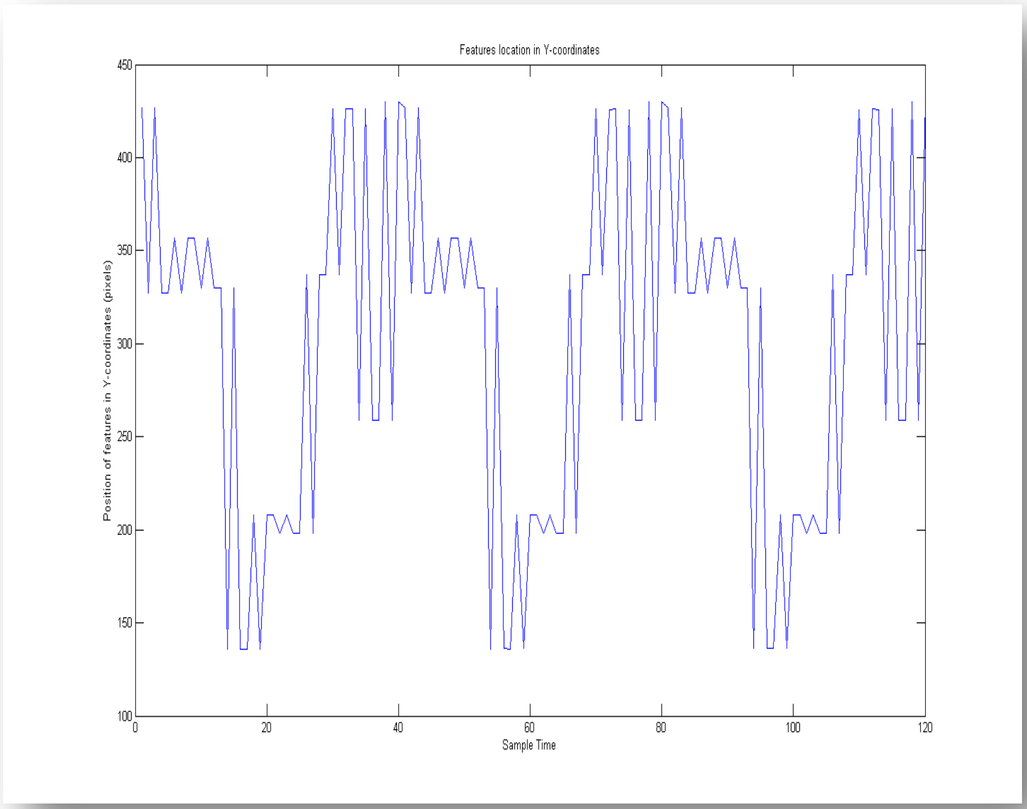
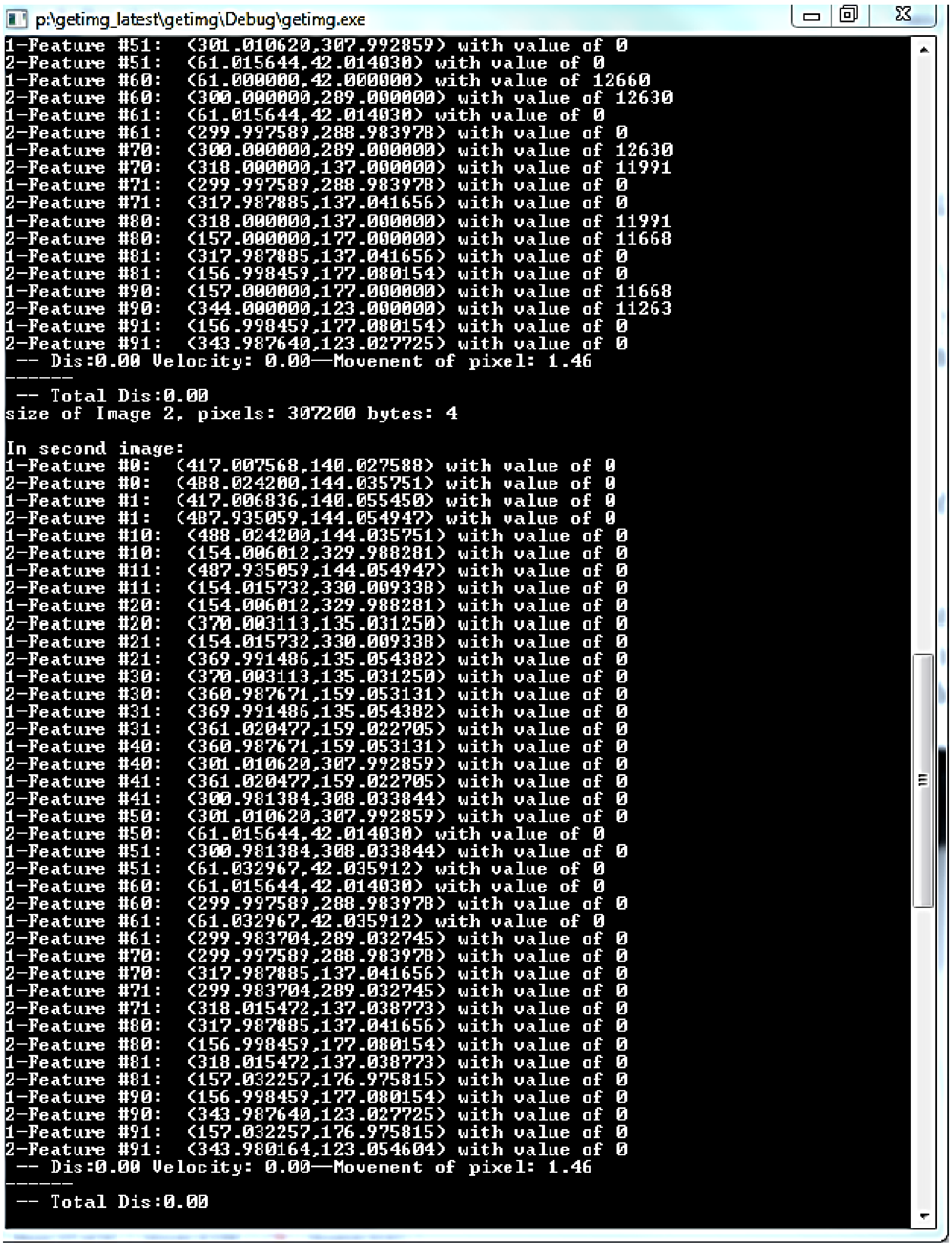


Figure 3.4: *Feature's position in Y-coordinates*



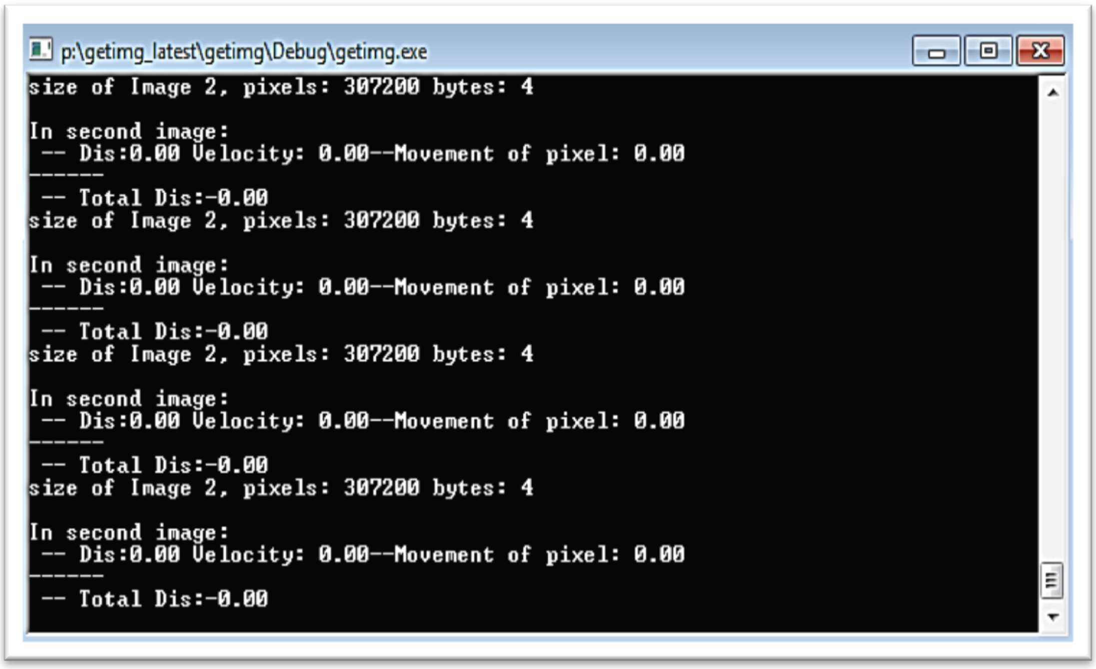
```
p:\getimg_latest\getimg\Debug\getimg.exe
1-Feature #51:  (301.010620,307.992859) with value of 0
2-Feature #51:  (61.015644,42.014030) with value of 0
1-Feature #60:  (61.000000,42.000000) with value of 12660
2-Feature #60:  (300.000000,289.000000) with value of 12630
1-Feature #61:  (61.015644,42.014030) with value of 0
2-Feature #61:  (299.997589,288.983978) with value of 0
1-Feature #70:  (300.000000,289.000000) with value of 12630
2-Feature #70:  (318.000000,137.000000) with value of 11991
1-Feature #71:  (299.997589,288.983978) with value of 0
2-Feature #71:  (317.987885,137.041656) with value of 0
1-Feature #80:  (318.000000,137.000000) with value of 11991
2-Feature #80:  (157.000000,177.000000) with value of 11668
1-Feature #81:  (317.987885,137.041656) with value of 0
2-Feature #81:  (156.998459,177.080154) with value of 0
1-Feature #90:  (157.000000,177.000000) with value of 11668
2-Feature #90:  (344.000000,123.000000) with value of 11263
1-Feature #91:  (156.998459,177.080154) with value of 0
2-Feature #91:  (343.987640,123.027725) with value of 0
 -- Dis:0.00 Velocity: 0.00—Movenent of pixel: 1.46
------
 -- Total Dis:0.00
size of Image 2, pixels: 307200 bytes: 4

In second image:
1-Feature #0:  (417.007568,140.027588) with value of 0
2-Feature #0:  (488.024200,144.035751) with value of 0
1-Feature #1:  (417.006836,140.055450) with value of 0
2-Feature #1:  (487.935059,144.054947) with value of 0
1-Feature #10:  (488.024200,144.035751) with value of 0
2-Feature #10:  (154.006012,329.988281) with value of 0
1-Feature #11:  (487.935059,144.054947) with value of 0
2-Feature #11:  (154.015732,330.009338) with value of 0
1-Feature #20:  (154.006012,329.988281) with value of 0
2-Feature #20:  (370.003113,135.031250) with value of 0
1-Feature #21:  (154.015732,330.009338) with value of 0
2-Feature #21:  (369.991486,135.054382) with value of 0
1-Feature #30:  (370.003113,135.031250) with value of 0
2-Feature #30:  (360.987671,159.053131) with value of 0
1-Feature #31:  (369.991486,135.054382) with value of 0
2-Feature #31:  (361.020477,159.022705) with value of 0
1-Feature #40:  (360.987671,159.053131) with value of 0
2-Feature #40:  (301.010620,307.992859) with value of 0
1-Feature #41:  (361.020477,159.022705) with value of 0
2-Feature #41:  (300.981384,308.033844) with value of 0
1-Feature #50:  (301.010620,307.992859) with value of 0
2-Feature #50:  (61.015644,42.014030) with value of 0
1-Feature #51:  (300.981384,308.033844) with value of 0
2-Feature #51:  (61.032967,42.035912) with value of 0
1-Feature #60:  (61.015644,42.014030) with value of 0
2-Feature #60:  (299.997589,288.983978) with value of 0
1-Feature #61:  (61.032967,42.035912) with value of 0
2-Feature #61:  (299.983704,289.032745) with value of 0
1-Feature #70:  (299.997589,288.983978) with value of 0
2-Feature #70:  (317.987885,137.041656) with value of 0
1-Feature #71:  (299.983704,289.032745) with value of 0
2-Feature #71:  (318.015472,137.038773) with value of 0
1-Feature #80:  (317.987885,137.041656) with value of 0
2-Feature #80:  (156.998459,177.080154) with value of 0
1-Feature #81:  (318.015472,137.038773) with value of 0
2-Feature #81:  (157.032257,176.975815) with value of 0
1-Feature #90:  (156.998459,177.080154) with value of 0
2-Feature #90:  (343.987640,123.027725) with value of 0
1-Feature #91:  (157.032257,176.975815) with value of 0
2-Feature #91:  (343.980164,123.054604) with value of 0
 -- Dis:0.00 Velocity: 0.00—Movenent of pixel: 1.46
------
 -- Total Dis:0.00
```

Figure 3.5: XY-coordinates values for selected and tracked features

The coordinate position in an image is shown in above figure. For example, 1-Feature # 90 ans 2-Feature # 90 is showing the position of the features in current and previous image. The position in XY coordinate of 1-Feature # 90 and 2-Feature # 90 is (156.99, 177.08) and (343.98, 123.02) respectively. Practically, moment in XY-coordinates was near to zero but still this algorithm was detecting large displacement of the position of the features.

The robustness of the algorithm was also not enough to cope up with the moment of camera. Algorithm's detection capability for the feature points with fast moment of the camera was also not enough to reveal result of the velocity. Hence, at the fast moment speed it shows no feature points detection. Following Figure, illustrate this problem;



```
size of Image 2, pixels: 307200 bytes: 4

In second image:
 -- Dis:0.00 Velocity: 0.00--Movement of pixel: 0.00
------
 -- Total Dis:-0.00
size of Image 2, pixels: 307200 bytes: 4

In second image:
 -- Dis:0.00 Velocity: 0.00--Movement of pixel: 0.00
------
 -- Total Dis:-0.00
size of Image 2, pixels: 307200 bytes: 4

In second image:
 -- Dis:0.00 Velocity: 0.00--Movement of pixel: 0.00
------
 -- Total Dis:-0.00
size of Image 2, pixels: 307200 bytes: 4

In second image:
 -- Dis:0.00 Velocity: 0.00--Movement of pixel: 0.00
------
 -- Total Dis:-0.00
```

Figure 3.6: *Zero detection of feature points at high-speed moments*

The incorrect distance measurement between the tracked features from image J to J+1 leads to the incorrect velocity calculation. Tracked features from the algorithm are as shown in the following Figure 3.7;

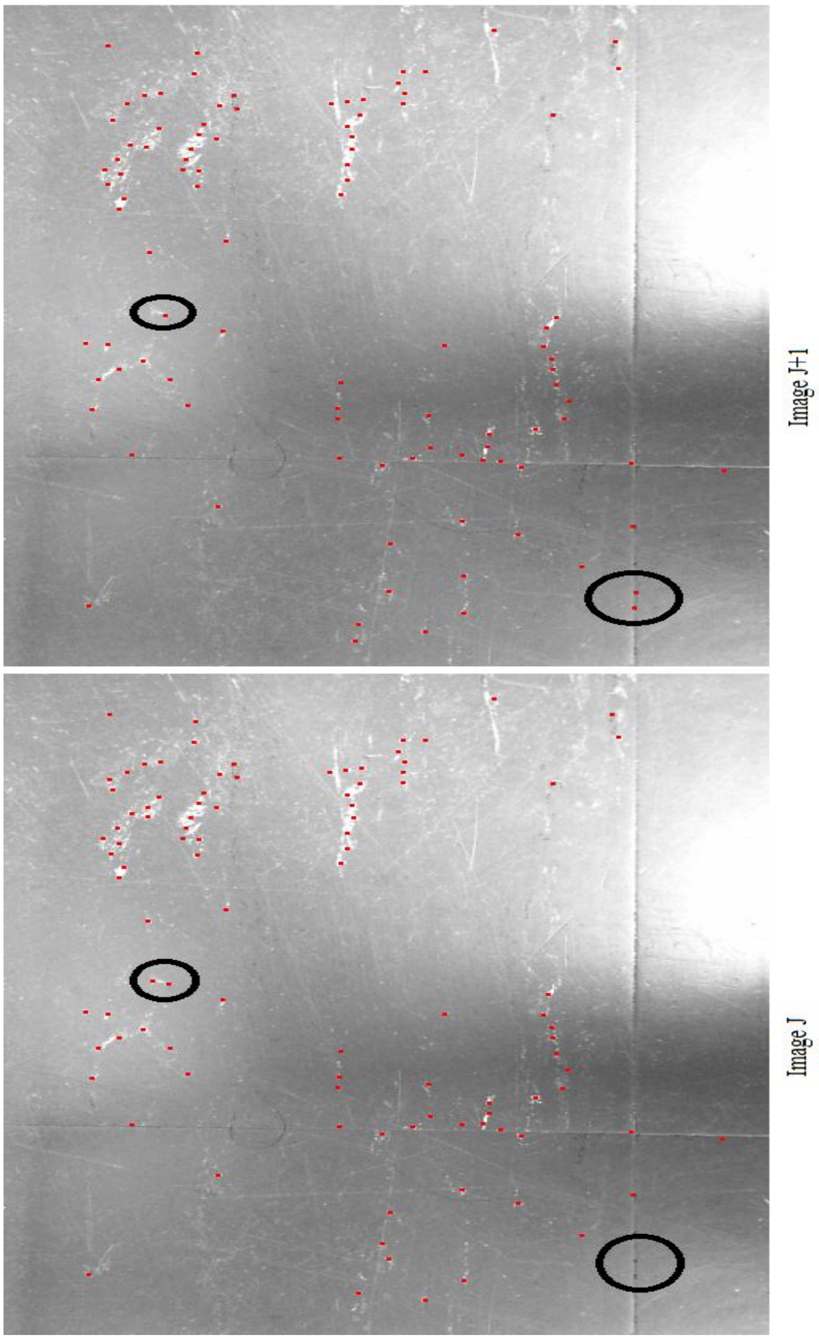


Figure 3.7: *Selected features in image J and J+1 when there was no moment, Red dots are features. Black circles shows those features, which are detected in image J and missed or not detected in image J+1 or vice-versa*

4. CORNER DETECTION ALGORITHM

This chapter is consists of theoretical and practical setup of the research work. This chapter will also highlight the pre and post processing algorithms. First, it is worthy to consider the standard coordinate system camera model, which is as follows;

4.1. Standard Coordinate System Camera Model

Camera coordinate system model gives the relationship between what appears on image plane and where it is located in 3D world. The standard coordinate system camera model is shown in Figure 4.1.

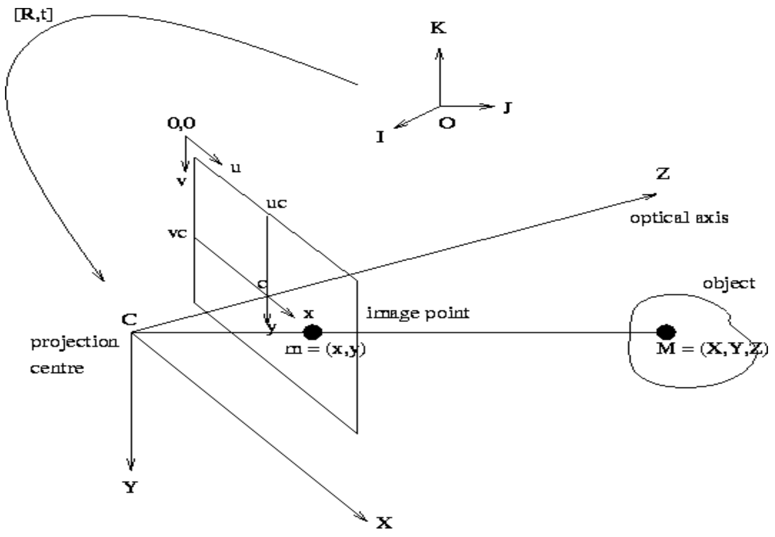


Figure 4.1: *The Standard Coordinate System Camera [4]*

In this model, light is entered the camera through a very tiny aperture. The geometry of the perspective projection is accurately captures by using this model [45]. The camera model has assumptions that the centre of projection of the image coincides with the world coordinates and z-axis (optical axis) is perpendicular to the terrain surface. It is also assume that image is placed in front of centre of projection as shown in Figure 4.1.

The camera model consists of intrinsic (image centre and focal length) parameters and extrinsic (location of the centre and 3D orientation of the projective method) parameters.

With an overview of the above mentioned standard camera model, it is mandatory to consider and implement camera calibration for intrinsic and extrinsic parameters. Camera calibration will help to have picture data on image coordinate for further analysis and results.

4.2. Camera Calibration

Camera calibration is also considered as pre-processing step of the work. Camera calibration is required for conversion of the image transformation estimation into measured body motions in the real world. Two sets of parameters namely, extrinsic and intrinsic parameters are the output of camera calibration part. Extrinsic parameters are not constant and are related to the camera orientation with respect to the setup environment. Intrinsic parameters are the parameters that are used to relate the pixel position of the conforming object in real space. Both these parameters are calibrated using Camera Calibration Toolbox using Matlab. They are important as they are used to determine the pixel position in real world. Those parameters are as follows;

- Focal length $f_c = [f_1 \quad f_2]$
- Principle point $C_c = [C_1 \quad C_2]$
- Skew coefficient $= \alpha_c$
- Distortion $K_c = [K_1\ K_2\ K_3\ K_4\ K_5]$

If the camera is assumes to have negligible skew coefficient and distortion then the $\alpha_c = 0$ and $K_c = [K_1\ K_2\ K_3\ K_4\ K_5] = 0$.

Camera calibration has been done using 50 images. All the images are captured using by the device “Microsoft LifeCam Cinema”. “Microsoft LifeCam Cinema” is also used in the later test experiments. Image acquisition has been done using different positions of the grid and camera. Some of the images used for calibration are as follows;

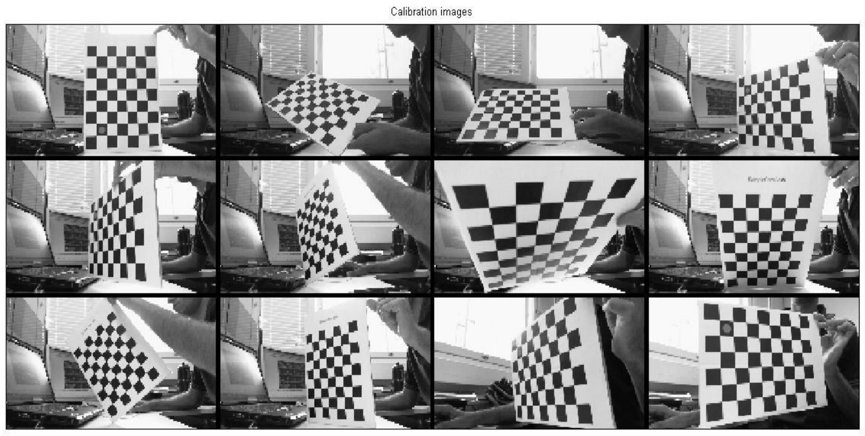


Figure 4.2: *Images used for camera calibration*

By following the steps provided at [12], calibration has been done. Following Figure 4.3 is showing the extraction of the grid corners using this above mentioned procedure.

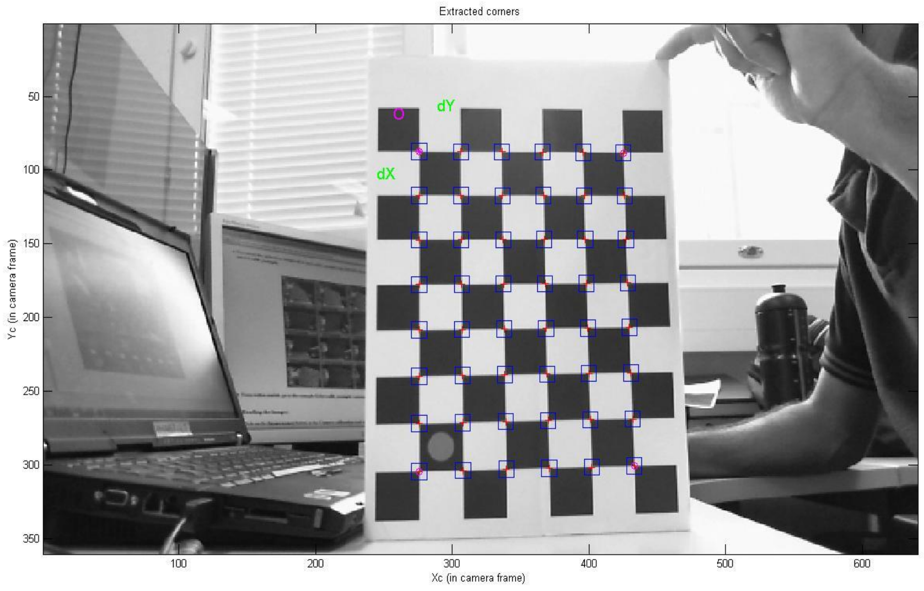


Figure 4.3: *Extracted corners at one of the calibration image*

With the following assumption of α_c and K_c, the data for the intrinsic parameters of the camera has following values as output;

Skew: alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion: kc = [0.00000 0.00000 0.00000 0.00000 0.00000] ± [0.00000 0.00000 0.00000 0.00000 0.00000]

Focal Length: fc = [493.59478 493.59478] ± [1.13663 1.13663]
Principal point: cc = [334.58054 174.66069] ± [0.74193 0.89226]
Pixel error: err = [0.18868 0.19682]

4.3. Camera Setup for Velocity Estimation

The camera setup for velocity estimation is as shown in the figure 4.4. The camera is looking downward towards the ground and mounted over the mobile robot. As it is assumed in the previous section that camera has negligible distortion, hence distortion parameter will become kc = |0 0 0 0 0 | and αc = 0.

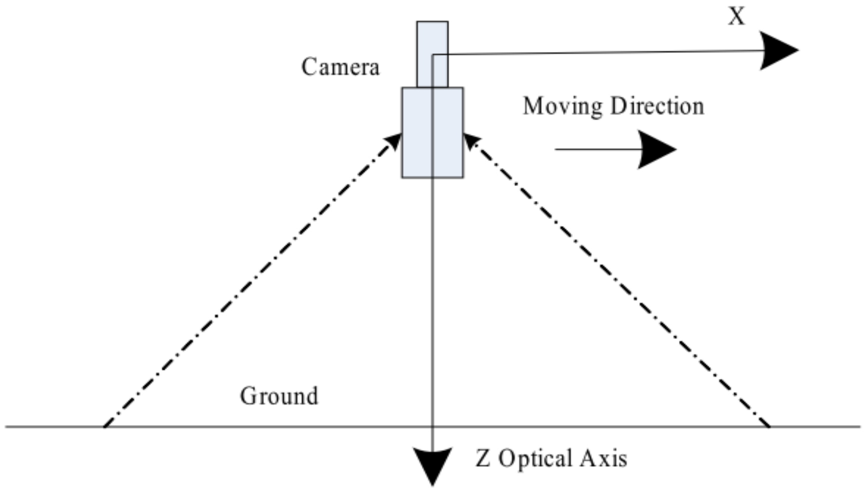


Figure4.4: *Camera Setup [3]*

On the basis of intrinsic, velocity component in image coordinate frame, minimum eigenvalue, principle point and distance between the features is calculated.

With the camera setup as shown in above Figure 4.4, it is necessary to do image pre-processing in order to have more reasonable and acceptable image for further processing.

4.4. Image Pre-processing

Image processing’s aim is to enhance quality of the image data and help to suppress the distortion and noise in order to have better image for further processing.

Image pre-processing is the initial step towards practical implementation of the algorithm and above mentioned setup. The autofocus of the camera is switched to fixed focus and configured according to the camera’s height requirement. It is very important to turn off the autofocus because with automatic setting of the focus, it is possible to have false corner points and so, the false results.

Illumination adjustment is one of the basic and another unique parameter in machine vision applications. Illumination parameter of the camera is also tuned automatically with the help of “imaqtool” of Matlab according to the height and test environment so that optimum level of illumination would help to find true corners.

Backlight compensation, exposure mode and white balancing is been set to automatic adjustment mode. Other parameters for pre-processing like saturation, sharpness and contrast mode are also been adjusted to a level to have an acceptable image.

All of these parameters are adjusted before commencing the main process.

4.5. Algorithm Implementation

The algorithm used for corner detection is been developed in Matlab/Simulink environment. The algorithm consists of the following steps;

- Image acquisition
- Colour space conversion
- Corner detection
- Corner matching
- Estimation of geometric transformation
- Velocities calculation

Entire Simulink Model is presented in Figure 4.5:

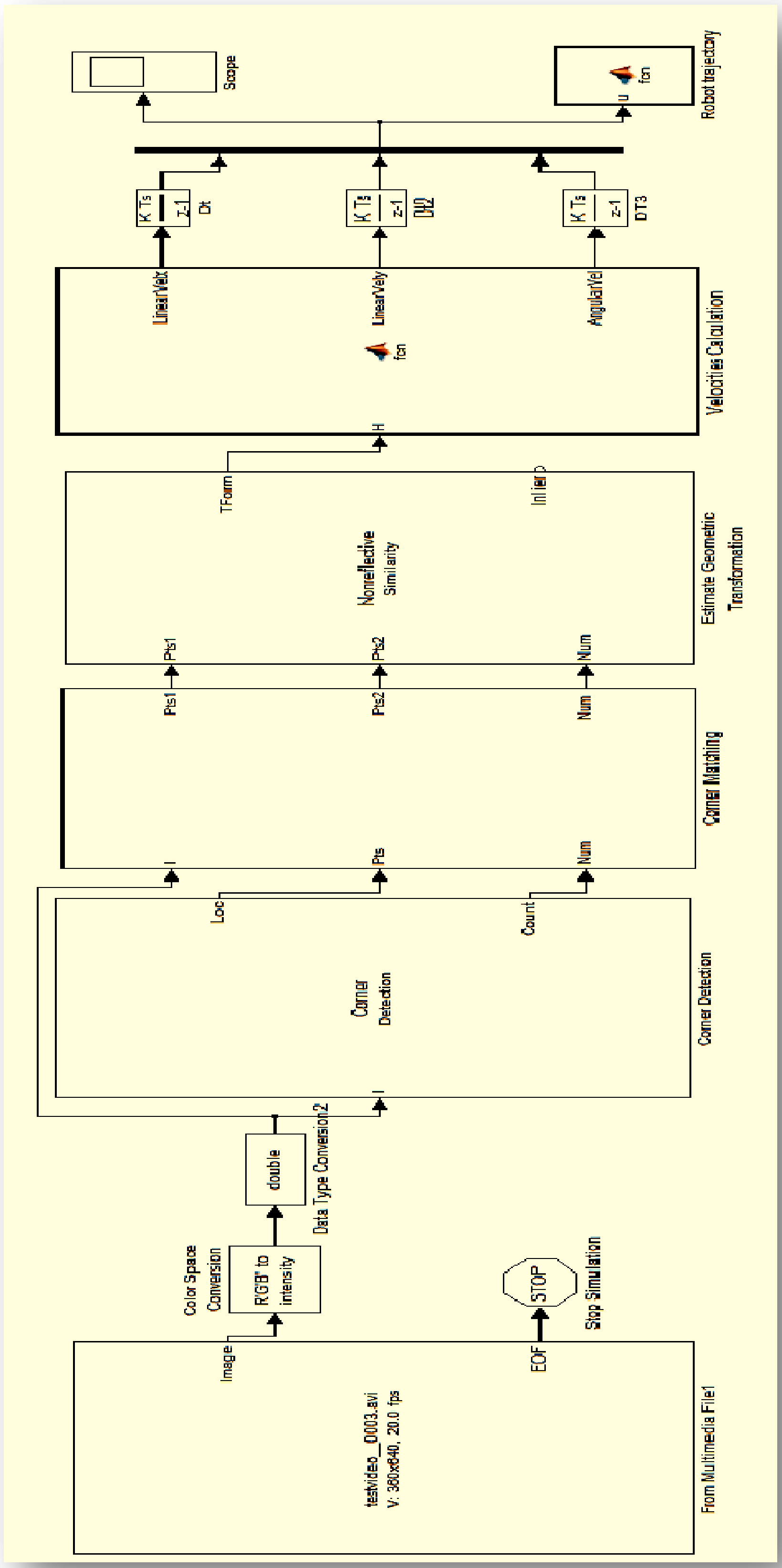


Figure 4.5: *Simulink Model for motion detection*

Detail code of the blocks like “corner matching”, “velocities calculation” and “robot trajectory” is presented in the Appendix 2.

4.5.1. Image acquisition

Image acquisition is the input of the algorithm. In order to capture images “Image Acquisition Tool” is been used. Before acquisition as described in section 4.4, image pre-processing is necessary to have better images in order to have better result. For this purpose all the parameters of pre-processing like, focus, saturation, white balancing, back light compensation, sharpness, contrast and exposure are set. White balancing, backlight compensation and exposure tuned to auto mode and rest of them are tune to manual mode. The image acquisition is done at 20fps (frames per second). The image resolution is set to 640x360 pixels. Very high resolution like 720x1280 or 1920x1080 pixels can reduce the computational speed of the algorithm and very low resolution like 320x240 can reduce the algorithm efficiency for finding feature points. So trade off policy was adopted and 640x360 pixels were selected. The colour space of the image is selected as RGB (Red, Green and blue channels).

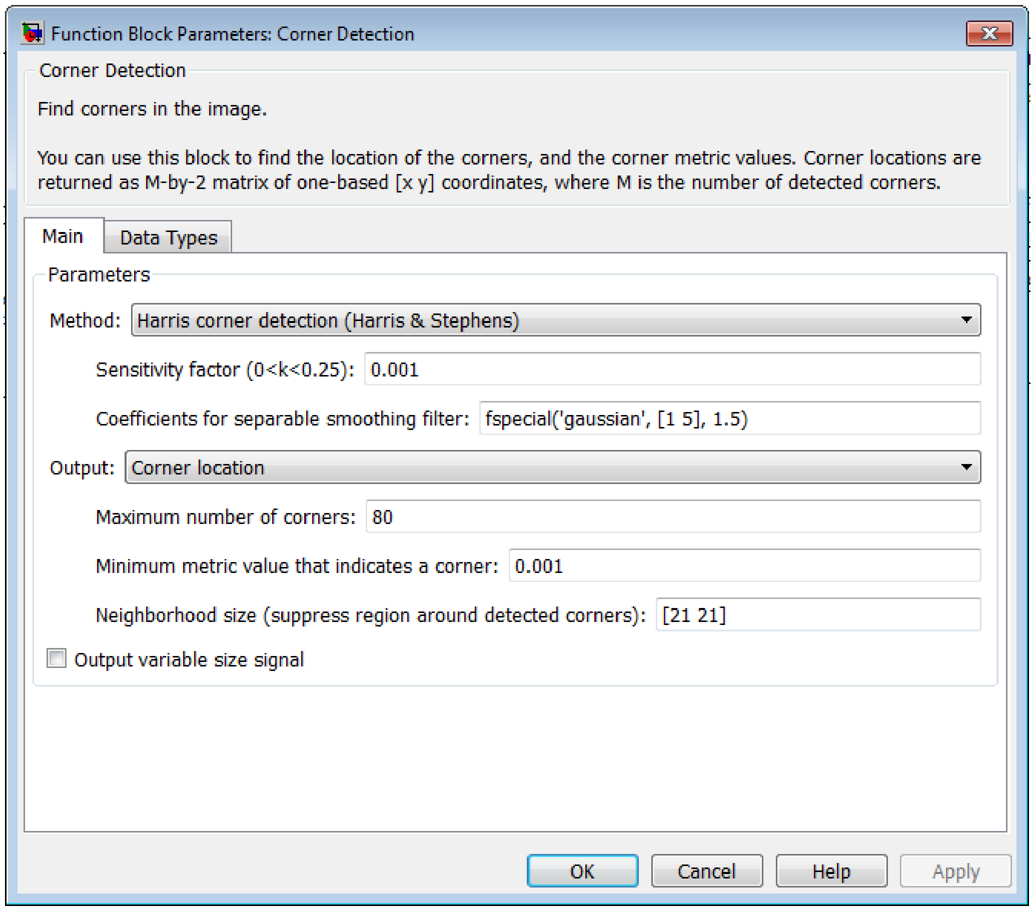
In case of saving the video stream at the test stage of the algorithm and for the camera calibration purpose; all the images are been acquired under these adjustments and save as “.avi” (uncompressed) file. The compressed feature is not been used because of the reason that it might loss some information from the image.

4.5.2. Colour space and data type conversion

Most of the machine vision algorithm works with grey scale image. Corner detection is also one of them and needs to have grey scale image for further processing. In order to convert RGB to grey scale image, colour space conversion block is been used. The data type that is supported by the corner detection algorithm is double and the data, which is coming as input, is single. So data type conversion from single to double is been also done using data type conversion block. After colour space and data type conversion, image is input to corner detection block.

4.5.3. Corner detection

Corner detection block is an important block of the algorithm. The input of this block is grey scale image with data type of double. Corner detection block finds corner points (feature points or point of interest) in an image. There are three methods as options available in this block for this purpose. Those methods namely are Harris corner detection, Shi & Tomasi corner detection and Rosten and Drummond. Details of these methods have been discussed in Section 2.2. Harris Corner detection is preferred among other available options because it is trade-off between accuracy and computation speed. The print screen for the corner detection parameters is as follows;


Function Block Parameters: Corner Detection

Corner Detection

Find corners in the image.

You can use this block to find the location of the corners, and the corner metric values. Corner locations are returned as M-by-2 matrix of one-based [x y] coordinates, where M is the number of detected corners.

Method: Harris corner detection (Harris & Stephens)

Sensitivity factor (0<k<0.25): 0.001

Coefficients for separable smoothing filter: fspecial('gaussian', [1 5], 1.5)

Output: Corner location

Maximum number of corners: 80

Minimum metric value that indicates a corner: 0.001

Neighborhood size (suppress region around detected corners): [21 21]

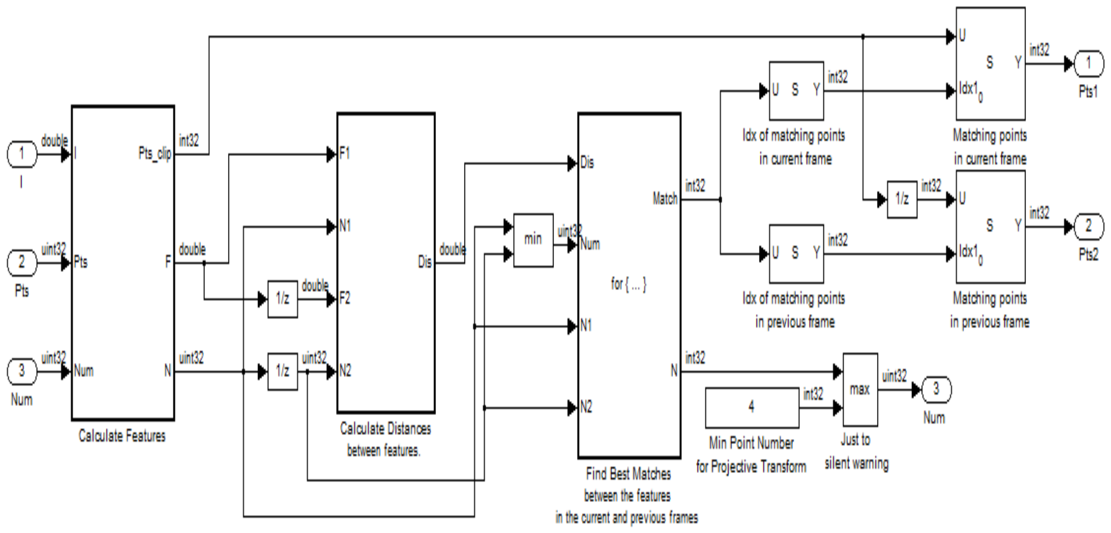
☐ Output variable size signal

Figure 4.6: *Corner detection parameters*

The sensitivity factor “K” is been adjusted between limits of 0<K<0.25. The smaller the value of sensitivity factor is, more likely sharper corners are detected as explained in Section 2.2. After several test runs the sensitivity factor “k” was adjusted to the 0.001. Maximum number of corner which algorithm needs to find in an input image can also be adjusted in this block and the number used here is 80 corners. The minimum metric value is the score R that indicates a corner, can be adjusted in this block. It was set to 0.001. Non-maximal suppression as explained in Section 2.3 is done with in certain range of neighbourhood around the corners. It is possible to tune this range in one of the options of corner detection block namely “Neighbourhood size”. It is required to define two element vector of positive odd integers that is [r c] where ‘r’ is the number of rows and ‘c’ is the number of column in the neighbourhood. With adjustment of all these parameters in the corner detection block we get corner location and number of corners (count) as output of this block. The corner location and count is the input of the corner matching block.

4.5.4. Corner matching

Corner matching is the block that is use to match the corner points from current image to the previous image. Detail code of the corner matching is block is mention in Appendix 2. The input of this block is the output of corner detection block. Detail of corner matching algorithm is discussed in section 2.4. In this algorithm the Neighbourhood matching is used for this purpose. Nearest neighbourhood based corner matching that is sub-category of neighbourhood matching is been used for this purpose. Distance is the decisive element for the selection of matching. All other matching options are explained in section 2.3. The output of this block is those matched points in current image and previous images that are name as Pts. 1 and Pts. 2 respectively. The third output "Num." is the number of corners been matched. These outputs are the input of next block, which is geometric transformation estimation. Print screen of the corner-matching block is as follows;



I
Pts
Num
double
uint32
Pts_clip
F
N
int32
Calculate Features
1/z
F1
N1
F2
N2
Dis
Calculate Distances between features.
min
Dis
Num
N1
N2
Match
for { ... }
Find Best Matches between the features in the current and previous frames
U S Y
Idx of matching points in current frame
Idx of matching points in previous frame
U
Idx1
S
Y
Matching points in current frame
Matching points in previous frame
Pts1
Pts2
4
Min Point Number for Projective Transform
max
Just to silent warning
Num

Figure 4.7: *Corner-matching blocks*

4.5.5. Estimate geometric transformation block

Estimate geometric transformation block is use to find the transformation matrix between number of point pairs between two images (current and previous). This block supports three types of transformations that is non-reflective similarity, affine and projective transformation. We are interested in Non-reflective similarity in this application. Non-reflective similarity algorithm contains four degrees of freedom. The input of this block is two pairs of points namely as Pts. 1, and Pts. 2. In this way, ultimate logical connection is established between corner matching and estimate geometric transformation blocks. One of the important functions this block performs is the removal of outliers as explained in section 2.5. With available options of RANdom SAmple Consensus (RANSAC) and Least Median Squares algorithm, RANSAC is been selected for finding and removal of outliers. Detail of RANSAC algorithm is available in [27] and [28].

With the selection of non-reflective similarity and RANSAC algorithm, it is possible to set the upper limit distance for the points that is called threshold for determining inliers. With this limit, a point can differ from the projection location of its associative points. It is possible to limit the percentage of random sampling performed by the algorithm for finding inliers from the input points. With the number of test runs the parameters of this block is fixed to the acceptable level in order to have optimum results. The output of this block is transformation matrix. This transformation matrix is been required by the velocities calculation and is working as input of velocities calculation block.

4.5.6. Velocities calculation

With transformation matrix as an input to the velocities calculation (Matlab function) block, velocities in X and Y direction and angular velocity is calculated. Hence, output of this calculation block is velocity vector (along x-y axis) and angular velocity (along z-axis). The result and test data is presented in the following chapter.

4.6. Visual presentation

In order to present the corners location in an image and robot trajectory, two blocks were added which are explained as follows:

4.6.1. Corners display

This block provides visual display of the corner points on the image. The input of this block consists of RGB image which is been provided before the conversion of image colour space to grey scale. The corner points are located with Pts 1 and Pts 2 for current and previous image respectively. Following figure will show their practical implementations which are translated images.

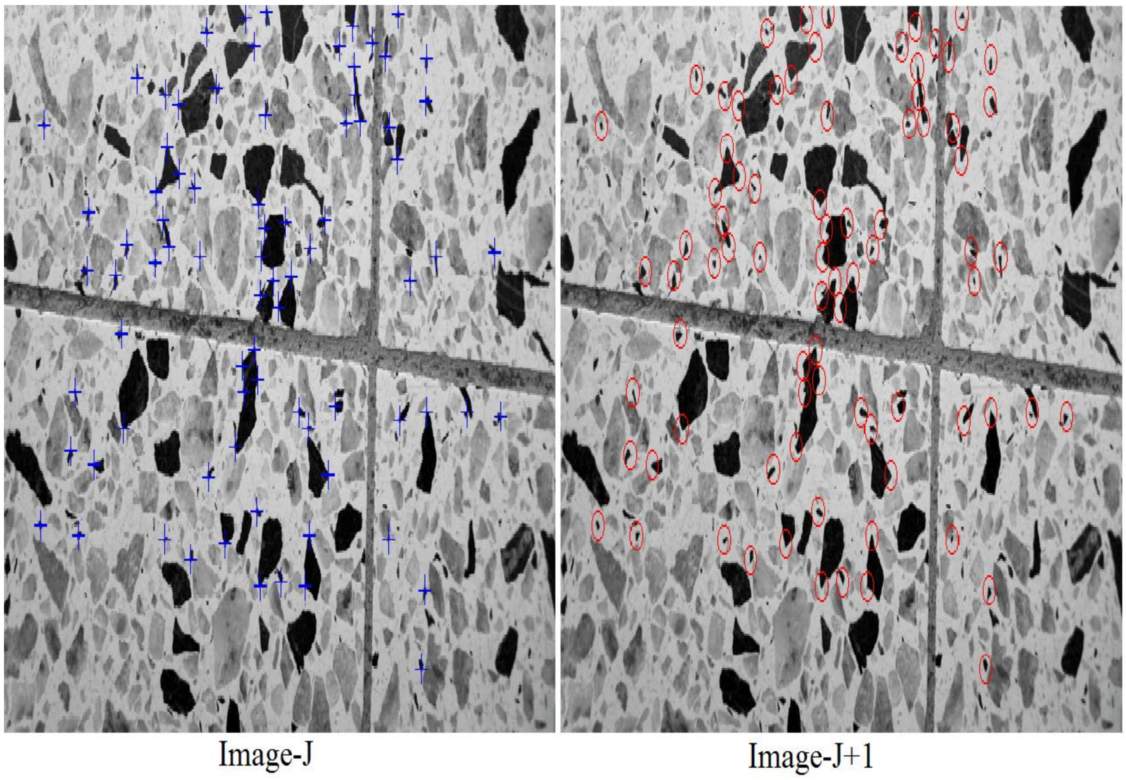


Figure 4.8: *Corner point's visual display with blue stars (image J) and red circle (J+1)*

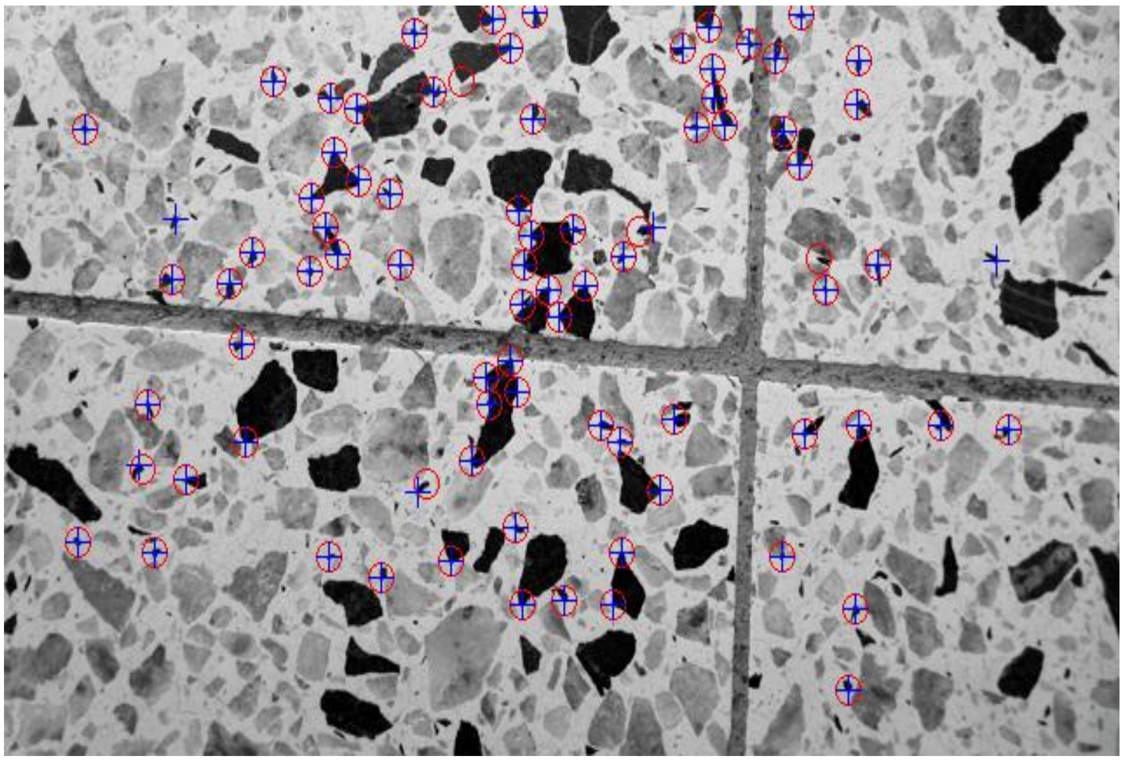


Figure 4.9: *Corner point's visual display when both corner points are completely overlapped with each other from current image (image J) to second image (image J+1)*

4.6.2. Robot position and trajectory display

In order to find robot position, “discrete time integrator block” is been added to the model and it is presenting the position of the robot versus time. With the plot of robot’s position, robot’s trajectory is been plotted as well. The test data of position and trajectory graph of the robot are presented in the next chapter.

5. RESULTS AND CONCLUSIONS

In this chapter, result of the algorithm is presented. The result we are presenting here are the outcome of the methods from the literature [7-9] and the methods mentioned in Section 2.2. Moreover, comparison and different problems while testing algorithm at different test setup is also mentioned in the chapter. Results are obtained by applying these methods in two test environments. Those environments are as follows;

- Real image with simulated motion
- Real test environment

5.1. Real image with simulated motion

Control test environment is consisting of MATLAB supported test setup. The main image is acquired using MATLAB toolbox “Imaqtool”. The motion of the image is simulated using translation commands in MATLAB. The input image has motion in a circular pattern. Motion of the input image is named as estimated output and it is used for the comparison to the actual output of the algorithm. Then the motion is detected using corner detection and matching algorithm as explained before in previous chapter 4.

The test image used for further processing is the part of the main image which is shown in Figure 5.1:

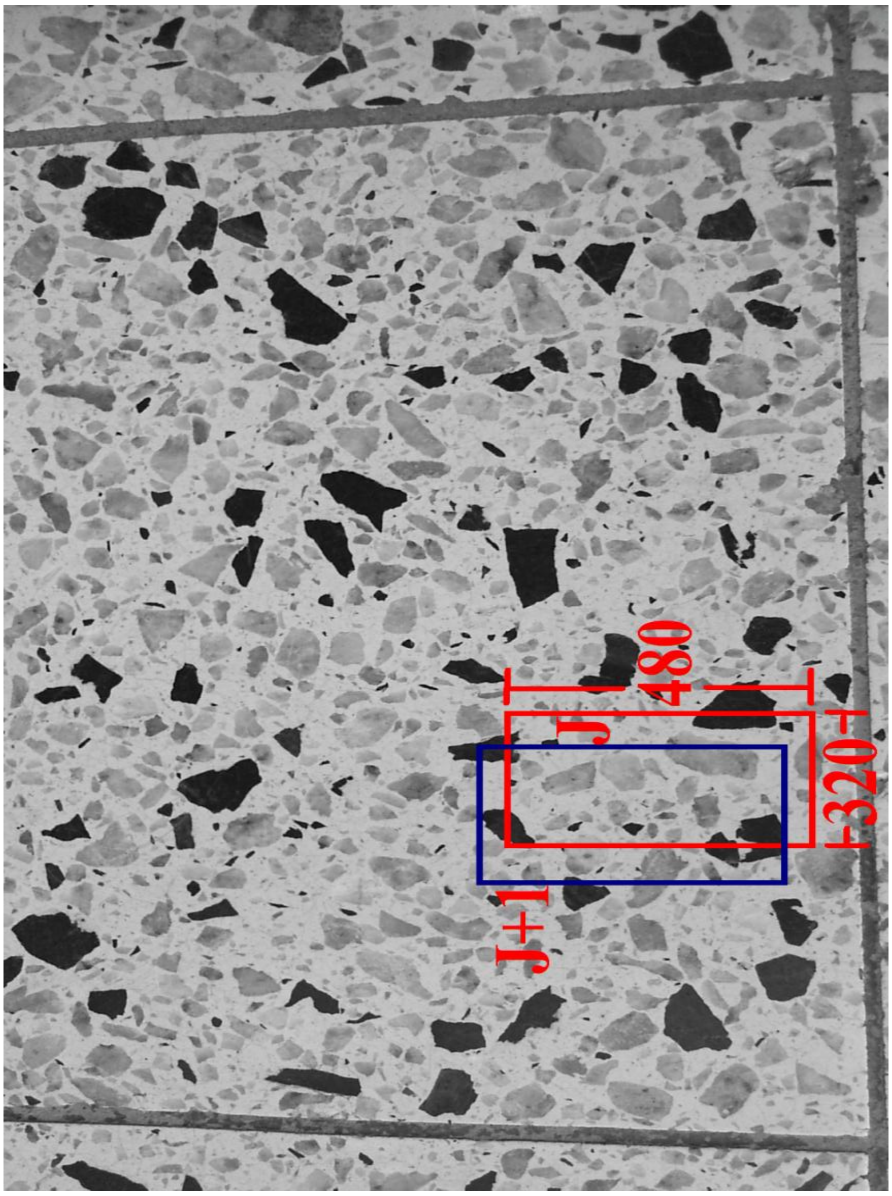


Figure 5.1: *Test image of the floor (image 'testpic') (dimensions 1920* x *1080 pixels²)*

The detected corner on the test image is shown in Figure 5.2:

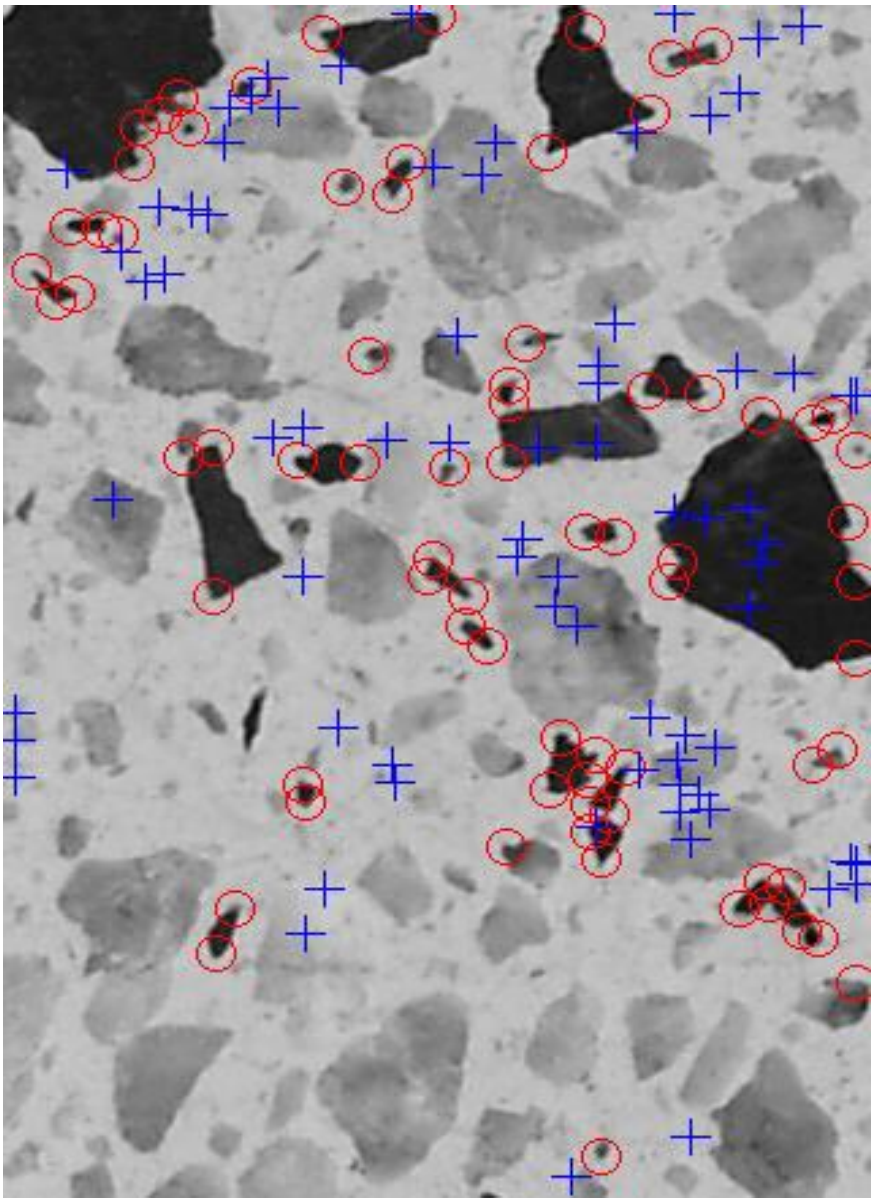


Figure 5.2: *(image 'testpic') Circles (red) are the corner points in image (J) followed by Plus sign (blue) in image (J +1) and both corner points are plotted on image J (dimensions 320* x *480 pixels²)*

Following plot (Figure 5.3 and 5.4) shows the comparison of an estimated vs. actual velocity with simulated test environment in X-Y directions.

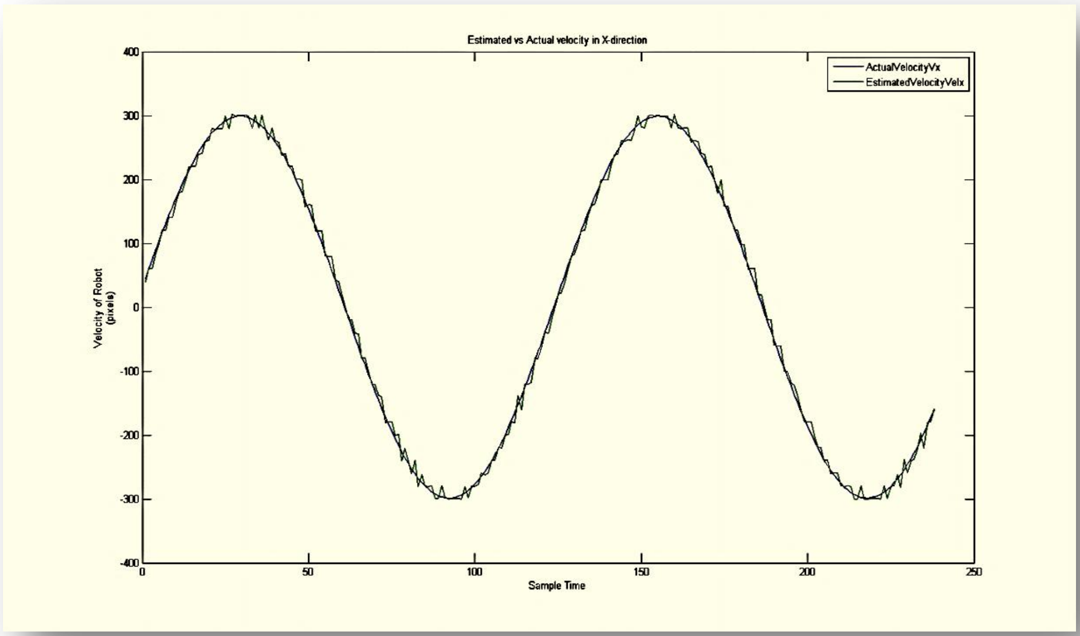


Figure 5.3: *Estimated vs. Actual Velocity in X-direction (image 'testpic')*

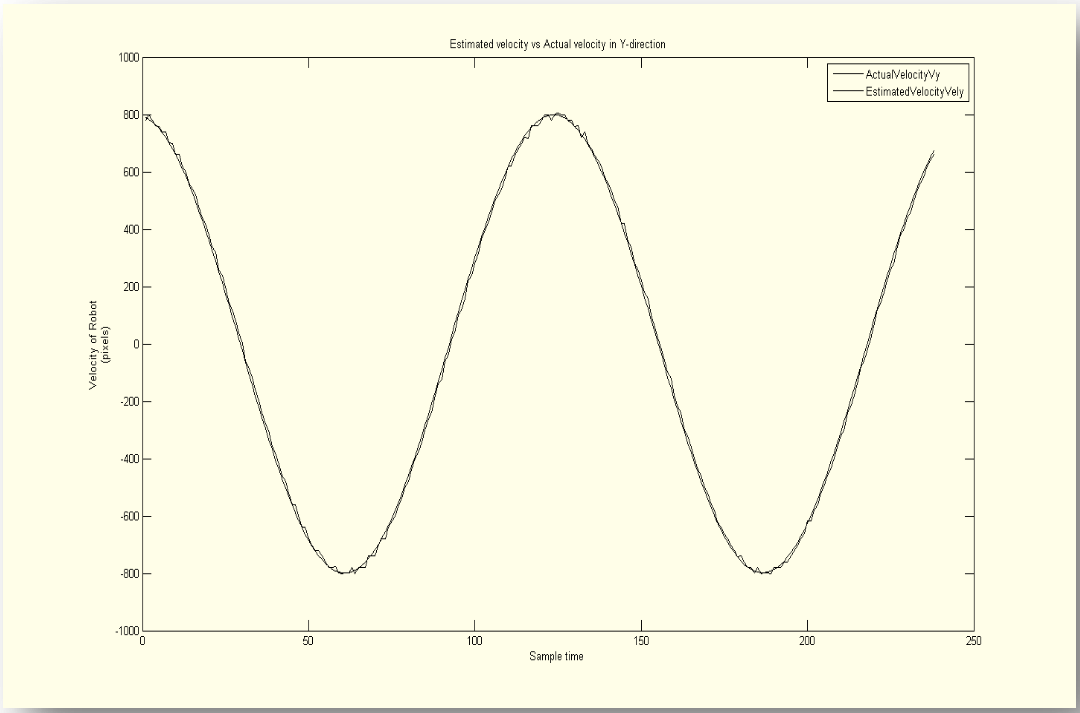
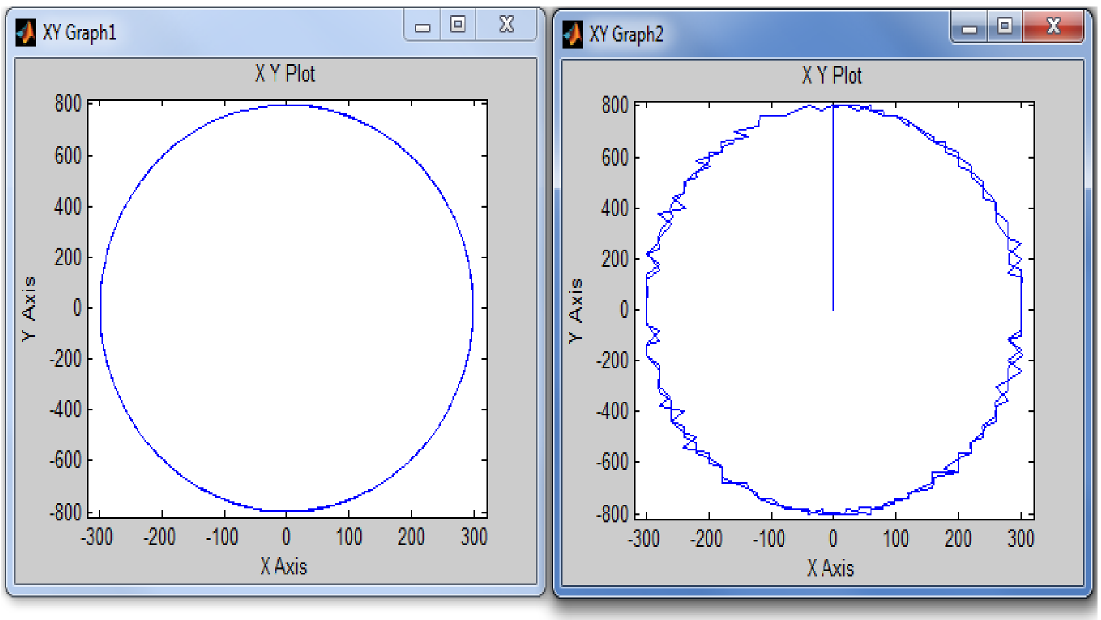


Figure 5.4: *Estimated vs. Actual Velocity in Y-direction (image 'testpic')*

Figure 5.5 shows the moment graph data of the estimated velocity, which is, compared with the actual graph data of the actual velocity.



Moment graph data for Actual Velcoity Moment graph data for Estimated Velocity

Figure 5.5: *Moment graph data (image 'testpic')*

The algorithm is processed with different images and error percentage and variance percentage is calculated for each image. The table that shows data for each image is as follows;

Input Image Name	Simulation Time Sec.	Moment direction	Average Error pixels	Average error %	Variance pixels sq.	Variance %
Testpic-1	12	X-direction	-0,629	-0,025	92,673	3,968
		Y-direction	-0,266	-0,005	274,208	4,626
Testpic-2 (testpic)	12	X-direction	-0,536	-0,023	91,782	3,929
		Y-direction	-0,242	-0,004	273,500	4,449
Testpic-3 (normal3)	12	X-direction	-0,523	-0,022	96,377	4,127
		Y-direction	-0,345	-0,0058	271,906	4,589
Testpic-4 (dark4)	12	X-direction	-0,400	-0,017	93,775	4,014
		Y-direction	-0,220	-0,004	275,544	4,649

Table 5.1: *Table showing the data for different images*

Major difference in all those above mentioned test images was illumination. The algorithm was tested with different illumination conditions in order to check the strength of the algorithm.

Testpic-1 and Testpic-2: Testpic-1 and Testpic-2 was the image having good illumination. Both the images have considerable difference in illumination parameters to each other. The average error % shows that difference of illumination has effect on results. Similarly the variance factor also reveals the difference of illumination. Because good and proper illumination helps to find true corners and it helps to have better results.
Testpic-3(normal3): This image has normal environment and illumination but also having two spots of light in the middle of the image. The average error% and variance% shows that this normal condition without any aid for the improvement of illumination has a lot effect on results if we compared to previous Testpic-1 and Testpic-2. The spot light in the middle of the image has also affected the results and similar problem was also faced in real tests and stated later in the Section 5.3.1.
Testpic-4(dark4): This image has less illumination than normal environment condition but no spots of light in the image. Average error% is less than the image (Testpic-3) having spots of light in it.

So after reviewing all the results by keeping in mind the illumination and environmental conditions of the images, it is concluded that illumination has also affect in simulated test environment. Spots due to reflection of light from the floor are also a reason of errors in calculation.

5.2. Real test environment

In the real test environment, the image was acquired using MATLAB toolbox “Imaqtool”. The entire prerequisite settings for the image acquisition are explained earlier in Section 5.5.1. Image pre-processing as explained in Section 5.4 is also adjusted using “imaqtool toolbox”.

The algorithm was tested with different test environment and conditions. The problems raised from very 1st to the last test, are mentioned later in the Section 5.3.

The test setup is shown in Figure 5.6.

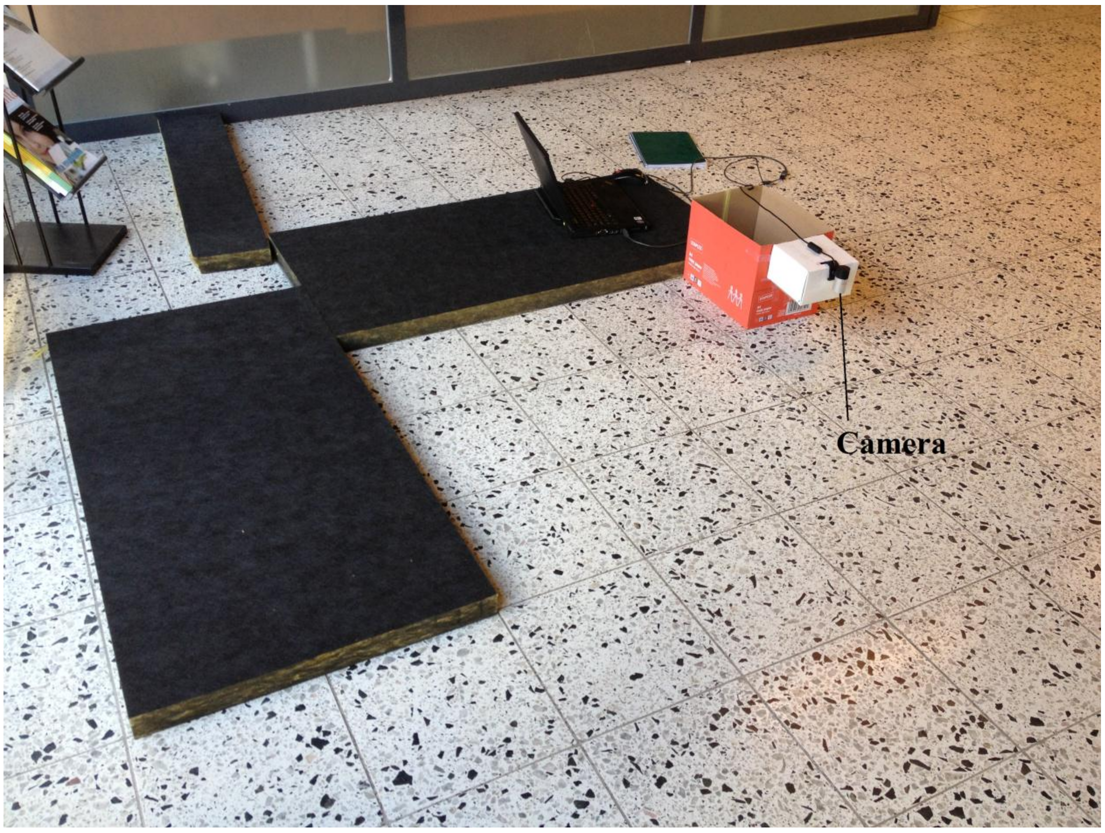


Figure 5.6: *Test setup*

Test setup was established inside the corridor near the glass door in front of information desk at Konetalo building in order to have rich and distributed illumination across the test surface. The camera was mounted with the small white box hanging away from the base red box at 90° as it is shown in Figure 5.6, the same principle as in Figure 4.4. In order to have corner points entirely from ground surface it was also made sure that camera should not have front side of the box in its frame of view. It is also possible to crop the image from entire frame of view in order to avoid the front end of the body.

The detected corners on the test surface are shown in the Figure 5.7. The dimensions of the image are 360 x 480 pixels^2.

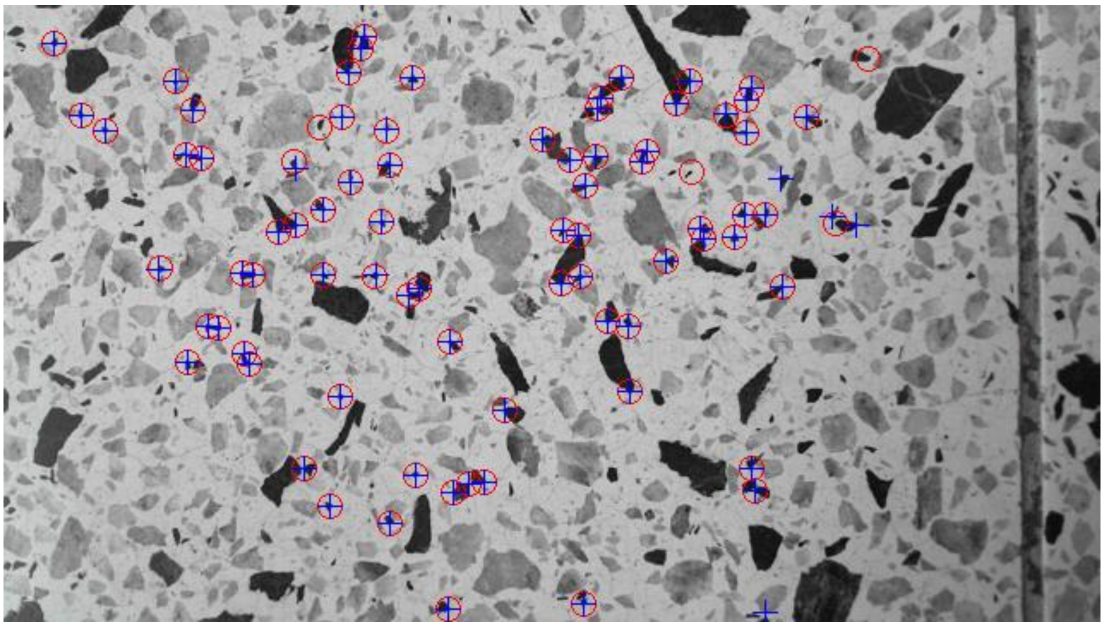


Figure 5.7: *Matched corner points in one of the test video*

The position of the robot after completion of one square round of 90 cm in X-direction and 90 cm in Y-direction (practically on ground), is shown in the following Figure 5.8. The starting XY-coordinate value is (-0.4517,-0.1416) and the last value of XY-coordinates at which the robot end its moments is (-2.377, 0.6431).

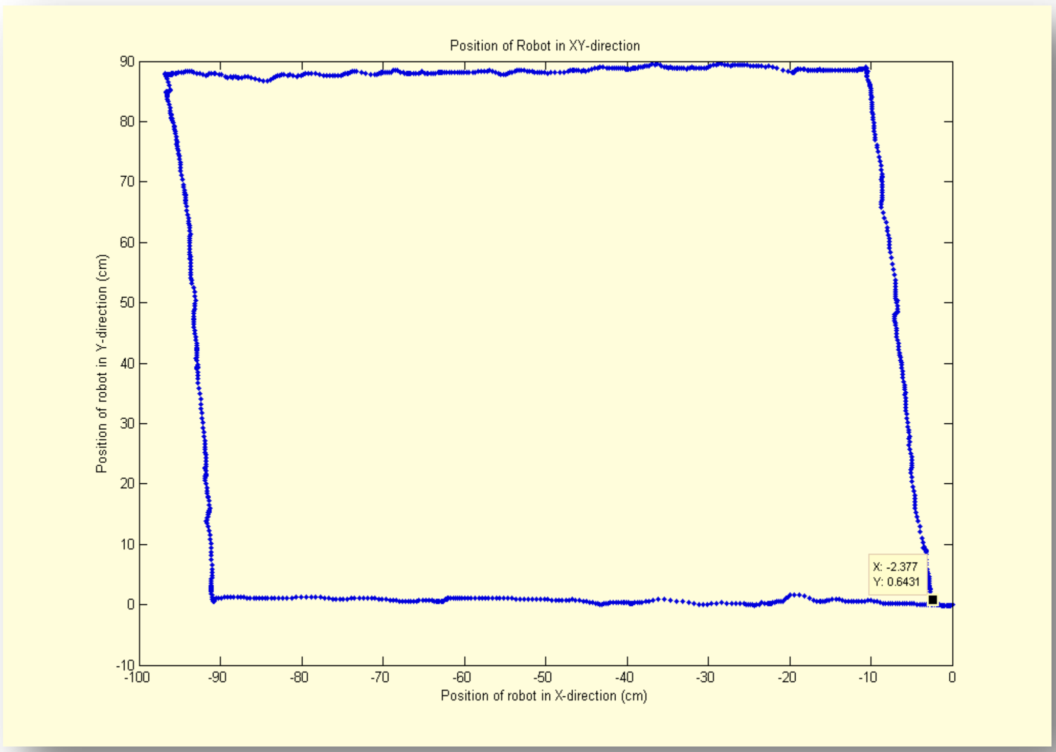


Figure 5.8: *Position data for the robot's moment*

In order to have a detail error analysis, the path of camera is divided into four parts as shown in the following Figure 5.9. It starts from the point (A) from where the robot starts it's travelling and ends its path at point named as end point as shown in the following Figure 5.9.

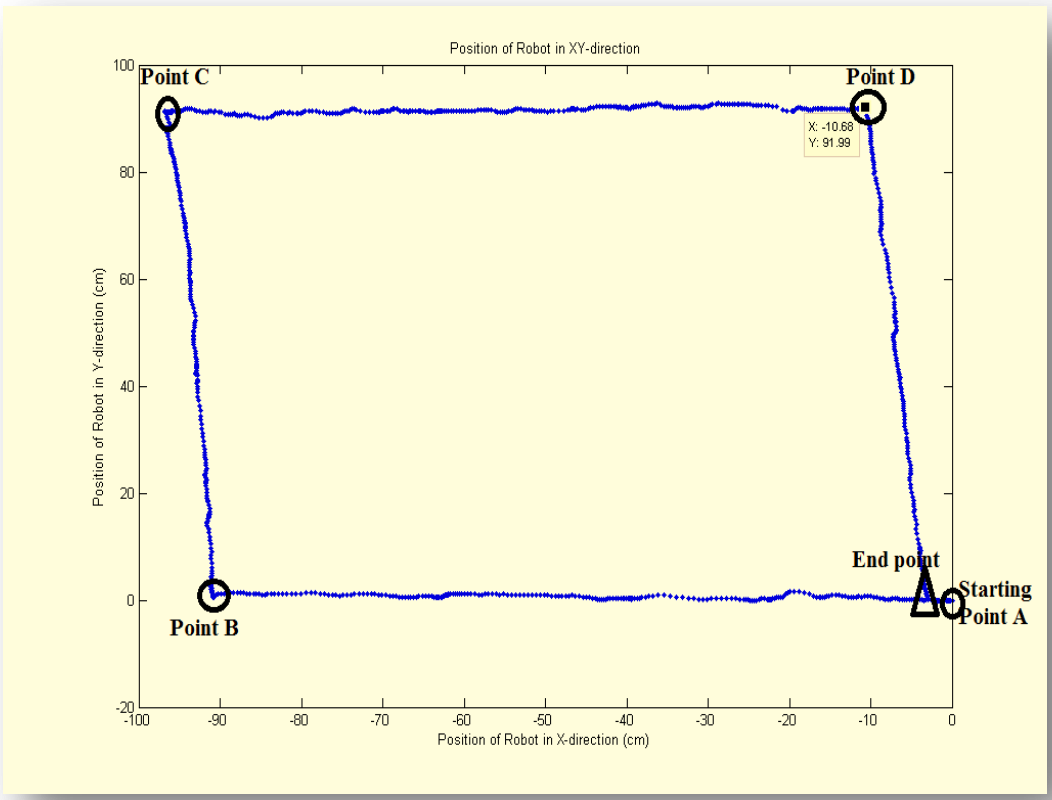


Figure 5.9: *Robot travelling path showing all start and ends*

The end point is highlighted in triangle just to make it different from other point on the trajectory.

The error analysis table is as follows:

Direction of moment	Start point (cm) coordinate value (x,y)	End point (cm) coordinate value (x,y)	Total motion (Actual) cm	Error value
X-axis	A (-0.4517, -0.1416)	B (-90.91, 1.083)	90	1.51%
Y-axis	B (-90.91, 1.083)	C (-96.75, 91.32)	90	2.67%
X-axis	C (-96.75, 91.32)	D (-10.68, 91.99)	90	4.3%
Y-axis	D (-10.68, 91.99)	End Point(-2.377, 0.6431)	90	2.9%

Table 5.2: *Error analysis*

The velocity plots for above-mentioned test in XY-direction are as follows:

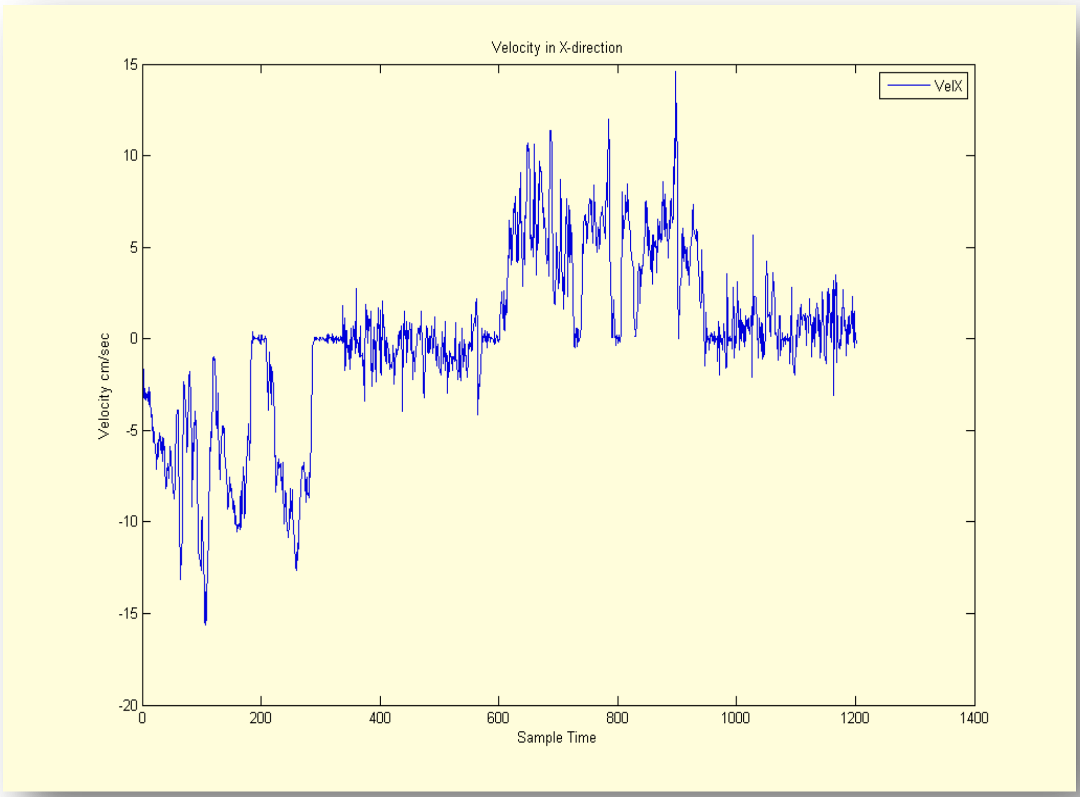


Figure 5.10: *Velocity in X-direction*

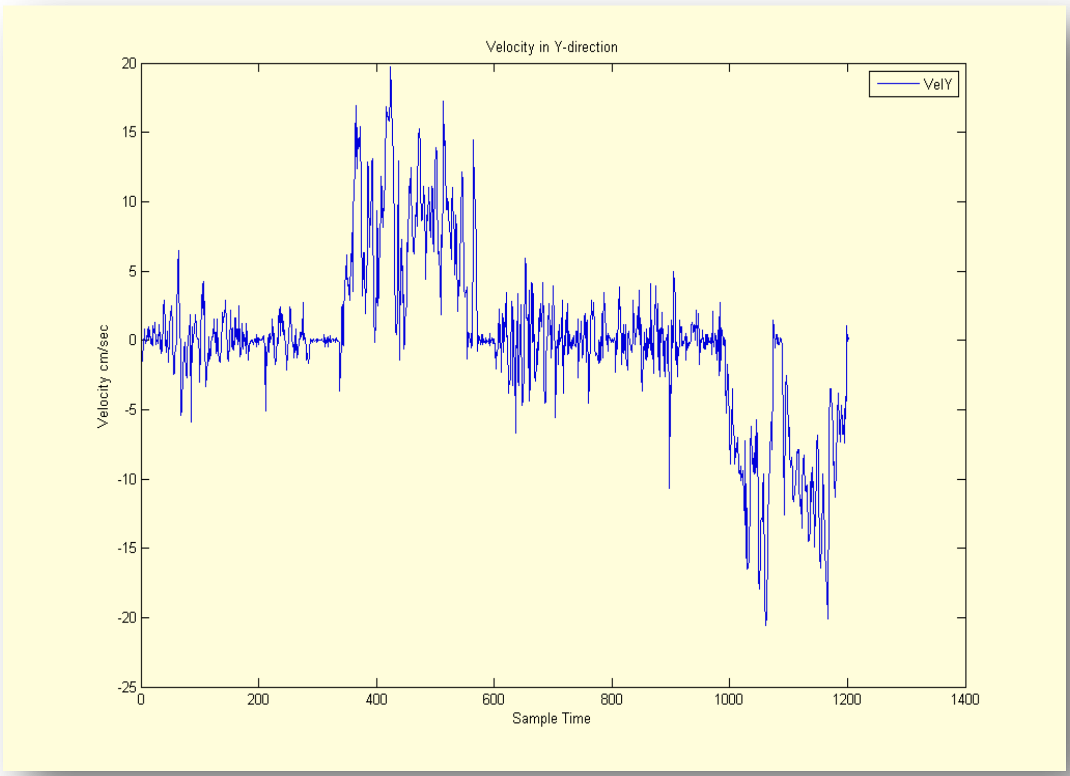


Figure 5.11: *Velocity in Y-direction*

All the three velocities for XY-direction and angular velocity are as shown in Figure 5.12.

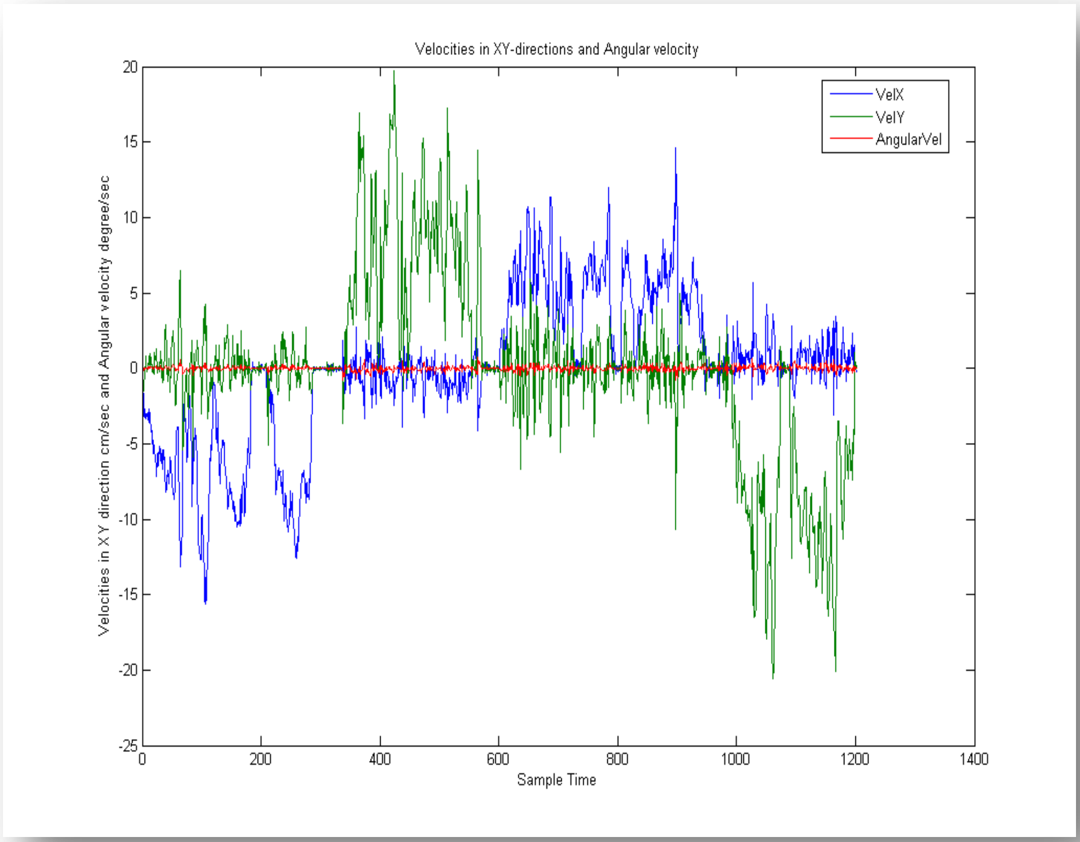


Figure 5.12: *VelX, VelY and angular velocity*

5.2.1. Heading angle accuracy test

In order to find the heading angle accuracy, one separate test was conducted. The test was conducted manually for moving the camera in a circular pattern path. In this test the camera was moving tangent to the path of moment. The whole test was consisting of moment in a complete one circle. The circular moment ended approximately at the same place where it is started from. By completing one rotation we can compare the heading angle estimated value with the actual value of the angle after one rotation. And by considering the difference between them, heading angle accuracy is calculated. Figure 5.13 shows the heading angle final value and angle error is given after that.

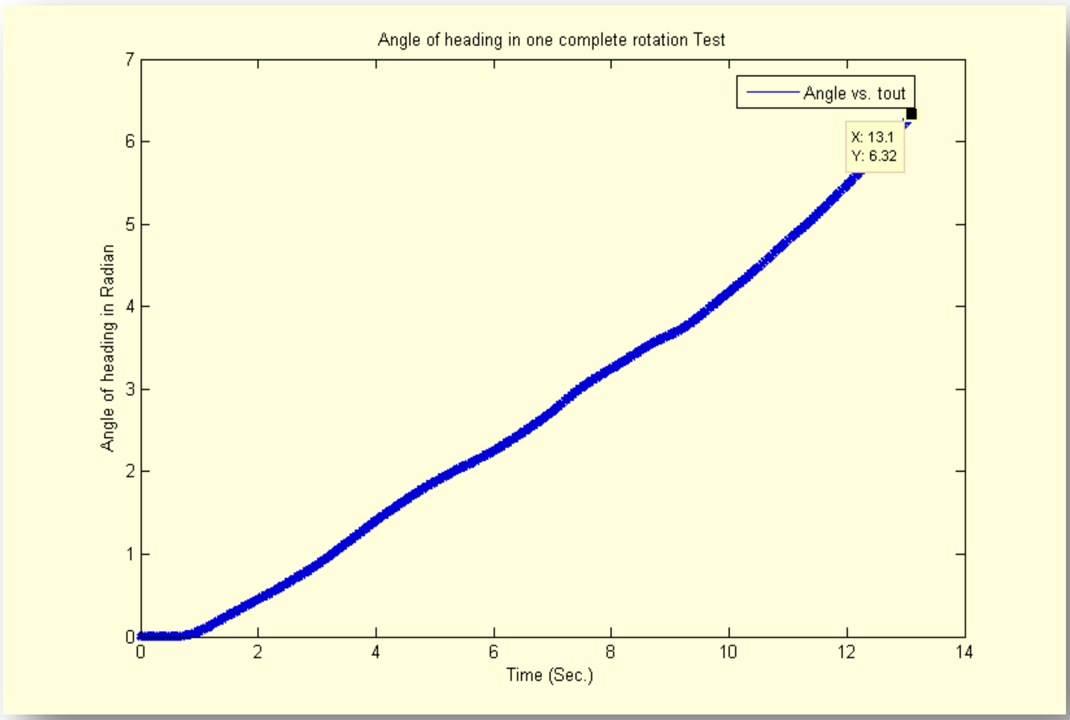


Figure 5.13: *Heading Angle moment in radians*

Heading angle accuracy plot is plotted with time (seconds) at X-axis and angle (radians) of heading on Y-axis. The heading angle accuracy is 0.5859%.

5.3. Discussions

In this section, all the problems and challenges that were faced during several phases of research work is mentioned.

5.3.1. Inside corridor test

The very first test of the algorithm is done inside the corridor. The test setup of the test is shown in the following Figure 5.9.

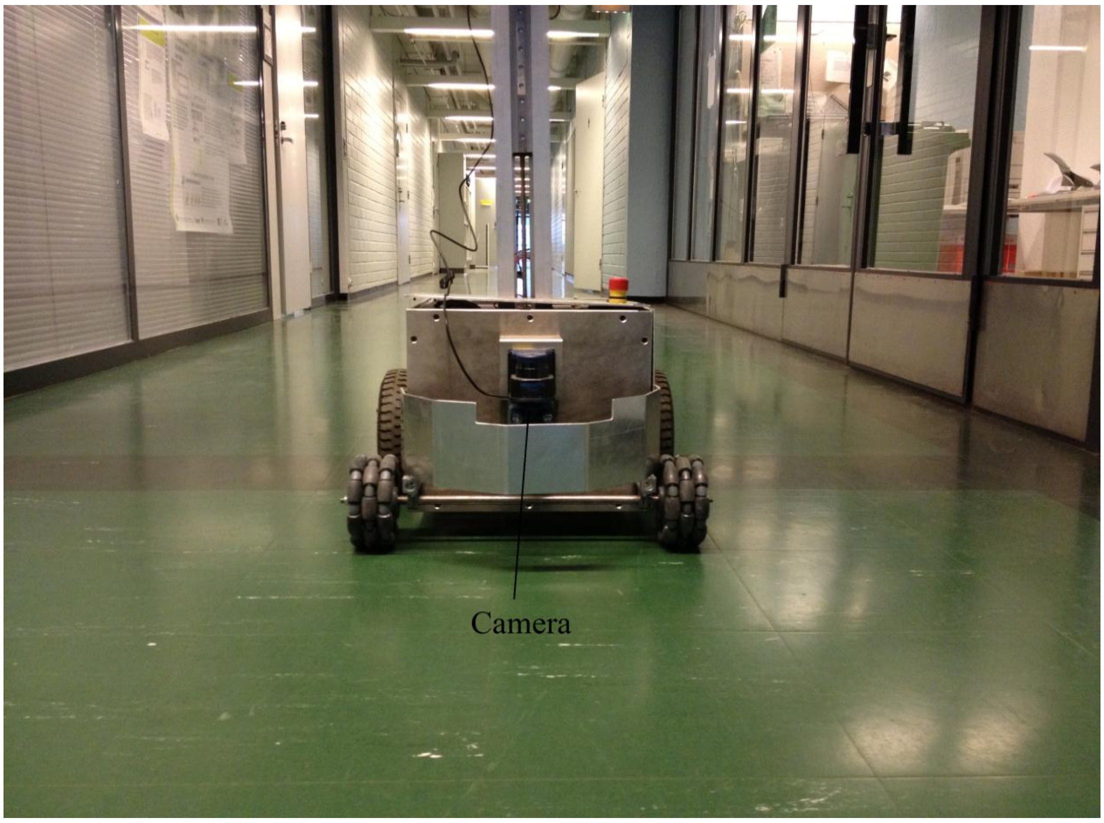


Figure 5.14: *The test setup and environment at inside corridor*

Autonomous robot was used for the test with the camera mounted at its front part as shown in Figure 5.9. Apparently illumination was looking fine. The algorithm was failed to detect corners in such illuminated environment because of few reasons; one of them was the spotlights, which accompanied the robot’s moment throughout the path.

Secondly, the dark shadows created by the front head of the robot and from the sides of the wall created the problem for the detection of corners. The corners were not detected at all in the dark places of the image.

Moreover, it was observed that corner points start moving with and around the spotlights by considering the edges of spotlights as a corner feature. See Figure 5.10.

As it is discussed in the Section 4.4 that focusing on the test surface is a very important factor in an image pre-processing. In the first test with “auto-focus” on, it was difficult for the algorithm to detect corner points as focus was changing all the time during the test.

The surface of the floor was highly reflective and the lights were reflected by it throughout the test, and this thing caused problem for the corner detection. These reasons are visible in Figure 5.15. The right parameters selection and with distributed light system installation along the camera, can reduce the surface reflection. Surface reflection is the particular problem with the test surface of our tests which were conducted inside the corridors.

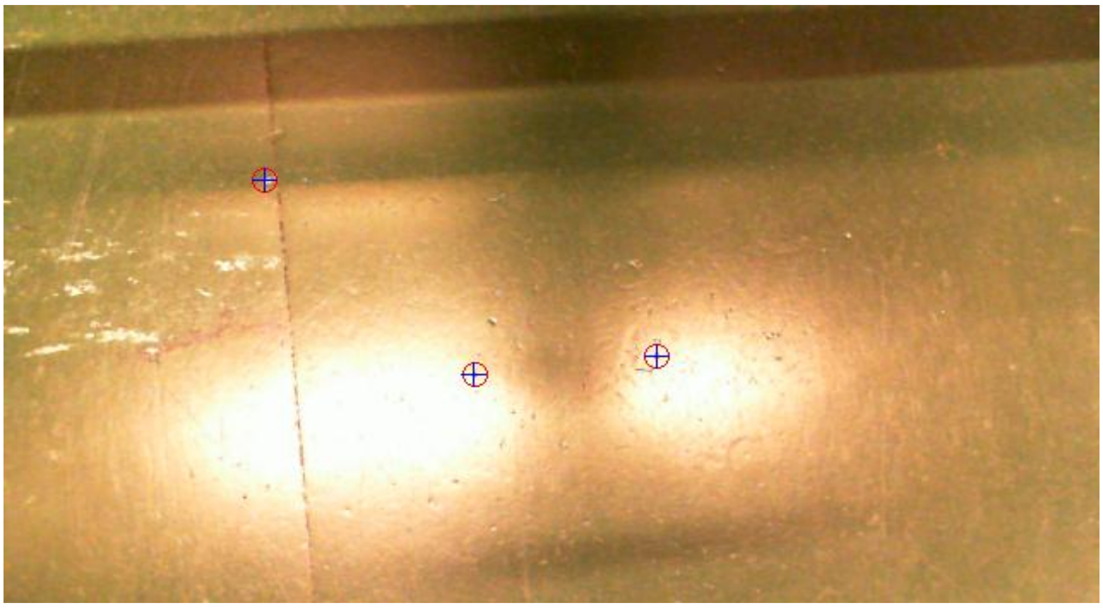


Figure 5.15: *Spotlights, un-distributed light and shadows are the reasons for almost zero corner detection*

From this test, it is concluded that spotlights and shadows are needed to be avoid in the future tests.

The test surface used in the future tests was very less reflective as shown in Figure 5.6.

The illumination problem was eliminated by using an environment where rich and distributed light was available as shown in Figure 5.6 and 5.7. The image pre-processing as described in detail in the Section 4.4, is very important before commencing of the tests. Image pre-processing options were available in MATLAB Imaqtool toolbox and were settled according to the environment.

5.3.2. Future work and suggestions

In the future work, it is possible to improve the results by doing some additional tasks. Those tasks options could be as follows:

- By using Kalman Filters, it is possible to reduce the current position error percentage and noise (random variations if there is any) of the algorithm.
- By using Kalman filters, inaccuracies of the estimation and efficiency of the algorithm could also be improved.
- Data fusion is another option to improve the results.

- Fusion of data from some absolute source and the current technology could improve the output of the algorithm.
- Illumination problems can be improved with some distributed light system, installed along the camera.
- In order to improve illumination problem, we can also use the camera looking in forward direction. Flat surface can be cropped from the frame of view for the tracking purpose while using same algorithm.
- Another solution to improve the algorithm results is to use the camera looking at ceiling but then these tests should be conducted inside the building.

REFERENCES

[1] Gibson, J.J., “The Perception of the Visual World” (Riverside Press, Cambridge, 1950)

[2] Gibson, J.J., “The Senses Considered as Perceptual Systems” (Houghton-Mifflin, Boston, MA. 1966).

[3] Deqing Sun, Stefan Roth, J.P. Lewis 3 and Michael J. Black “Learning Optical Flow” Proceedings of the 10th European Conference on Computer Vision, 2008.

[4] Black, M.J., Anandan, P. “The robust estimation of multiple motions: Parametric and piecewise-smooth flow field” CVIU 63, 75-104 (1996)

[5] Pepping, G. J. and Grealy, M. L. (ed.) “Closing the Gap”: The Scientific Writings of David N. Lee, Erlbaum” 2007

[6] Todd, J. T. “The Visual Perception of Three-Dimensional Structure from Motion” In: Perception of Space and Motion (eds. W. Epstein and S. Rogers), p. 201. Academic press, 1995.

[7] C. Schmid, R. Mohr, and C. Bauckhage., “Evaluation of interest point detectors” International Journal of Computer Vision, 37(2):151–172, June 2000.

[8] C. Harris and M.J. Stephens., “A combined corner and edge detector” In Alvey Vision Conference, pages 147–152, 1988.

[9] Available at http://www.aishack.in/2010/05/the-shi-tomasi-corner-detector/

[10] Edward Rosten, Reid Porter and Tom Drummond “Faster and better: a machine learning approach to corner detection” IEEE Trans. Pattern analysis and Machine intelligence, 2010.

[11] D.Parks, J.P. Gravel, “Corner detection” available at http://ftp.utcluj.ro/pub/users/nedevschi/AV/3_CornerDetectors/CornerDetection.pdf

[12] Jean-Yves Bouguet, “Camera Calibration Toolbox for Matlab” available at http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[13] I. -K. Jung and S. Lacroix, “A robust interest points matching algorithm,” in Proc. Int. Conf. on Comp. Vis., 2, 538–543, 2001.

[14] D. Zhou, G. Li, and Y. -H. Liu, “Effective corner matching based on Delaunay triangulation,” in Proc. Int. Conf. on Robotics and Automation, 3, 2730–2733, Apr. May 2004.

[15] E. R. Davies, Machine vision: theory, algorithms, practicalities, Academic Press, London, 1990.

[16] Mohammad Awrangjeb, “Contour-based Corner Detection and Robust Geometric Point Matching Techniques” thesis submitted for PHD at Monash University, Australia July 2008.

[17] D. M. Mount, N. S. Netanyahu and J. L. Moigne, “Efficient algorithms for robust feature matching”, Pattern Recognition, vol. 32, no. 1, pp. 17–38, January 1999.

[18] F. Zhao, Q. Huang and W. Gao, “Image matching by multiscale oriented corner correlation”, in Proc. Asian Conference on Computer Vision, vol. LNCS 3851, pp.

928-937, Hyderabad, India, January 2006.
[19] Y. Dufournaud, C. Schmid and R. Horaud, “Matching images with different res-olutions”, in Proc. International Conference on Computer Vision and Pattern Recog-nition, vol. 1, pp. 612–618, South Carolina, USA, June 2000.
[20] M. I. A. Lourakis, A. A. Argyros and K. Marias, “A graph-based approach to cor-ner matching using mutual information as a local similarity measure”, in Proc. International Conference on Pattern Recognition, vol. 2, pp. 827–830, Cambridge, UK, August 2004.
[21] F. Mokhtarian and F. Mohanna, “Performance evaluation of corner detectors using consistency and accuracy measures”, Computer Vision and Image Under-standing, vol. 102, no. 1, pp. 81–94, April 2006.
[22] F. Zhao and W. Gao, “Singular value decomposition based image matching”, in Proc. International Conference on Image and Graphics, pp. 192–195, Hong Kong, China, December 2004.
[23] D. P. Huttenlocher, “Fast affine point matching: an output-sensitive method”, in Proc. International Conference on Computer Vision and Pattern Recognition, pp. 263-268, Maui, HI, USA, June 1991.
[24] D. Zhou, G. Li and Y. Liu, “Effective corner matching based on delaunay trian-gulation”, in Proc. International Conference on Robotics and Automation, vol. 3, pp. 2730–2733, New Orleans, LA, USA, April-May 2004.
[25] K. Mikolajczyk and C. Schmid, ”A performance evaluation of local descriptors”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1615– 16309, October 2005.
[26] M.A. Fischler and R.C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography” Commu-nications of the ACM, 24(6):381–395, 1981.
[27] Bernd Kitt, Andreas Geiger and Henning Lategahn, “Visual Odometry based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme”, 2010 IEEE Intelligent Vehicles Symposium University of California, San Diego, CA, USA June 21-24, 2010
[28] Estimate geometric transformation block description available at http://www.mathworks.se/help/toolbox/vision/ref/estimategeometrictransformation.html
[29] ATTALI D. “R-regular shape reconstruction from unorganized points”. In Processing of ACM Symp. on Comp. Geometry (1997), pp. 248–253
[30] Ran Gal, Ariel Shamir, Tal Hassner, Mark Pauly, Daniel Cohen-Or, “Surface Reconstruction using Local Shape Priors” EUROGRAPHICS ’0x / N.N. and N.N., The Eurographics Association 2007.
[31] H.Hoppe, T. DeRose, T. Duchamp, J. Mcdonnald, and W. Stuetzle. “Surface reconstruction from unorganized points” SIGGRAPH’92. Computer graphics. 26(2):71-77. 1992.

[32] Philipp Jenke, Bastian Kr¨uckeberg, Wolfgang Straßer, “Surface Reconstruction from Fitted Shape Primitives” WSI/GRIS, University of Tuebingen, Germany, VMV 2008.

[33] Fernando de Goes, David Cohen-Steiner, Pierre Alliez,, Mathieu Desbrun, “An Optimal Transport Approach to Robust Reconstruction and Simplification of 2D Shapes” Volume 30 (2011), Number 5, Eurographics Symposium on Geometry Processing 2011.

[34] McDermott, J., Weiss, Y., Adelson, E.H., “Beyond junctions: nonlocal form constraints on motion interpretation” Perception, 2001, volume 30, pages 905-923.

[35] The material is available at the website with the link: http://www.seas.upenn.edu/~derpanis/steerable_filter_harris_corner_binomial_filter_lecture.pdf

[36] The material is available at the website with the link: http://undergraduate.csse.uwa.edu.au/units/CITS4240/Lectures/tracking.pdf

[37] D. Zhou, G. Li and Y. Liu, ”Effective corner matching based on delaunay triangulation”, in Proc. International Conference on Robotics and Automation, vol. 3, pp.2730–2733, New Orleans, LA, USA, April-May 2004.

[38] Y. Dufournaud, C. Schmid and R. Horaud, ”Matching images with different resolutions”, in Proc. International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 612–618, South Carolina, USA, June 2000.

[39] B. Platel, E. Balmachnova, L. M. J. Florack and B. M. ter Haar Romeny, ”Top-points as interest points for image matching”, in Proc. European Conference on Computer Vision, vol. LNCS 3951, no. 1, pp. 418–429, Graz, Austria, May 2006.

[40] C. Schmid and R. Mohr, ”Local grayvalue invariants for image retrieval”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, no. 5, pp. 530–534, May 1997.

[41] T. Tuytelaars and L. J. Van Gool, ”Content-based image retrieval based on local affinely invariant regions ”, in Proc. International Conference on Visual Information and Information Systems, vol. LNCS 1614, pp. 493–500, Amsterdam, The Netherlands, June 1999.

[42] Y. Ke and R. Sukthantar, ”PCA-SIFT: a more distinctive representation for local image descriptors”, in Proc. International Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 506–513, Washington, DC, USA, June-July 2004 .

[43] D. G. Lowe, ”Distinctive image features from scale-invariant keypoints”, Interntional Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, November 2004.

[44] W. T. Freeman and E. H. Adelson, ”The design and use of steerable filters”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 9, pp. 891–906, September 1991.

[45] Owens, R. [1997] \Pinhole camera model," online avaialable: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html#x1-30003

[46] Roth, S., Black, M.J. “On the spatial statistics of optical flow” IJCV 74, 33–50 (2007)

[47] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R. “A database and evaluation methodology for optical flow” In: ICCV (2007)

[48] H. Bay and T. Tuytelaars and L. V. Gool, ”SURF: speeded up robust features”, in Proc. European Conference on Computer Vision, vol. LNCS 3951, no. 1, pp. 404–417, Graz, Austria, May 2006

[49] G. Lu, Multimedia Database Management Systems, Artech House Inc., Norwood, MA, USA, October 1999

[50] T. T. H. Tran and E. Marchand, ”Real-time keypoints matching: application to visual servoing”, in Proc. International Conference on Robotics and Automation, pp. 3787–3792, Roma, Italy, April 2007

[51] J. Wang, H. Zha and R. Cipolla, ”Coarse-to-fine vision-based localization by indexing scale-Invariant features”, IEEE Trans. on Systems, Man and Cybernetics, Part B, vol. 36, no. 2, pp. 413–422, April 2006

[52] D. G. Lowe, ”Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, November 2004

[53] S. Alkaabi and F. Deravi, ”Iterative corner extraction and matching for mosaic construction”, in Proc. Canadian Conference on Computer and Robot Vision, pp. 468 -475, Victoria, BC, Canada, May 2005

[54] M. Brown, R. Szeliski and S. Winder, ”Multi-image matching using multi-scale oriented patches”, in Proc. International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 510–517, San Diego, CA, USA, June 2005

[55] Y. Caspi, D. Simakov and M. Irani, ”Feature-based sequence-to-sequence match-ing”, International Journal of Computer Vision, vol. 68, no. 1, pp. 53–64, June 2006

[56] D. Nister and H. Stewenius, ”Scalable recognition with a vocabulary tree”, in Proc. International Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2161–2168, New York, NY, USA, June 2006

[57] F. Mokhtarian and R. Suomela, ”Robust image corner detection through curvature scale space”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1376–1381, December 1998

[58] C. J. Harris and M. Stephens, ”A combined corner and edge detector”, in Proc. Alvey Vision Conference, pp. 147–151, Manchester, UK, August-September, 1988

[59] L. V. Gool, T. Moons and D. Ungureanu, ”Affine/photometric invariants for planar intensity patterns”, in Proc. European Conference on Computer Vision, vol. LNCS 1064, no. 1, pp. 642–651, Cambridge, UK, April 1996

[60] C. Schmid, R. Mohr and C. Bauckhage, ”Comparing and evaluating interest points”, in Proc. International Conference on Computer Vision, pp. 230–235, Bombay, India, January 1998

[61] J. Borenstein, H. R. Everett, and L. Feng, “Where am i? sensors and methods for mobile robot positioning,” University of Michigan, Tech. Rep., April 1996

[62] Navid Nourani-Vatani, Jonathan Roberts, Mandiam V. Srinivasan “Practical Visual Odometry for Car-like Vehicles” 2009 IEEE International Conference on Robotics and Automation Kobe International Conference Center Kobe, Japan, May 12-17, 2009

[63] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry for ground vehicle applications” Journal of Field Robotics, vol. 23, no. 1, pp. 3–20, 2005

[64] R.C. Gonzalez and R.E. Woods, “Digital image processing” Prentice Hall, Second edition, 2002

[65] I-M. Sintorn, “Segmentaion methods and shape description in digial images: Application in 2-D and 3-D microscopy” Swedish university of agricultural sciences

[66] D. S. Guru, R. Dinesh and P. Nagabhushan, ”Boundary based corner detection and localization using new ’cornerity’ index: a robust approach”, in Proc. Canadian Conference on Computer and Robot Vision, pp. 417–423, London, Ontario, Canada, May 2004

[67] Available at http://www.acclaimimages.com/ provided with persmission for research and non-commercial uses

[68] Available at http://www.petitcolas.net/fabien/ courtesy of University of Southeren California

[69] J. Shi and C. Tomasi "Good Features to Track," 9th IEEE Conference on Computer Vision and Pattern Recognition, June 1994

[70] C. Tomasi and T. Kanade "Detection and Tracking of Point Features" Pattern Recognition **37**: 165–168, 2004

[71] Linda G. Shapiro and George C. Stockman “Computer Vision” A book published in Feb. 2, 2001

APPENDIX 1: OPTICAL FLOW ALGORTIHM CODE

```
#include "pnmio.h"
#include "klt.h"

#include "stdafx.h"
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <iostream>
#include <iomanip>
using namespace std;

#define REPLACE

 // A Simple Camera Capture Framework
 int main() {


// Initialization of the KLT
unsigned char *img1, *img2;
KLT_TrackingContext tc;
KLT_FeatureList flOld;
KLT_FeatureList flNew;
KLT_FeatureTable ft;
int nFrames = 10;
int nFeatures = 100;
int ncols=640;
int nrows=480;
int i=0;
float d = 0;
float x = 0;
float v = 0;
float y = 0;
float y_tot=0;
float y_average=0;
float v_average=0;
float y_pixel=0;
float Y_tot=0;

tc = KLTCreateTrackingContext();
//tc->min_eigenvalue=1;


KLTPrintTrackingContext(tc);
flNew = KLTCreateFeatureList(nFeatures);
flOld = KLTCreateFeatureList(nFeatures);

ft = KLTCreateFeatureTable(nFrames, nFeatures);
tc->sequentialMode = TRUE;

// Capture first image before starting loop
CvCapture* capture = cvCaptureFromCAM( CV_CAP_ANY );
if ( !capture ) {
                      fprintf( stderr, "ERROR: capture is NULL \n"
           );
                      getchar();
                      return -1;
                 }
```

```
// Create a window in which the captured images will be presented
cvNamedWindow( "mywindow", CV_WINDOW_AUTOSIZE );

// Get one frame
IplImage* frame = cvQueryFrame( capture );
IplImage* frame1=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
cvCvtColor(frame,frame1,CV_BGR2GRAY);

/*
frame->dataOrder = 0;
cout << (frame->nChannels) <<  endl;
cout << (frame->depth) <<  endl;
cout << (frame->dataOrder) <<  endl;
cout << (frame->origin) <<  endl;
cout << (frame->width) <<  endl;
cout << (frame->height) <<  endl;
*/
if ( !frame ) {
                      fprintf( stderr, "ERROR: frame is null...\n"
);
                      getchar();
                      return 0;
           }


// grab the image data from IplImage
img2 = (unsigned char*)frame1->imageData;
img1 = new unsigned char[nrows*ncols];

//img1 = new unsigned char[frame->width*frame->height];

//printf("\nwidth :%d\n" ,frame->width );
//printf("\nheigth:%d\n" ,frame->height);


for(int j=0; j<nrows*ncols; j++){
                                 img1[j]=img2[j];
                                 }

//KLTSelectGoodFeatures(tc, img1, ncols, nrows, flOld);
KLTSelectGoodFeatures(tc, img1, ncols, nrows, flNew);
KLTStoreFeatureList(flNew, ft, 0);

//printf("\nIn first image:\n");
//for (i = 0 ; i < ft->nFeatures ; i++)  {
//printf("Feature #%d:  (%f,%f) with value of %d\n",
//         i, ft->feature[0][i]->x, ft->feature[0][i]->y,
//         ft->feature[0][i]->val);

printf("\nIn first image:\n");
for (i = 0 ; i < flNew->nFeatures ; i++)
{
printf("Feature #%d:  (%f,%f) with value of %d\n",
i, flNew->feature[i]->x, flNew->feature[i]->y,
flNew->feature[i]->val);
}

//KLTWriteFeatureListToPPM(flNew, img1, ncols, nrows, "feat1.ppm");
//KLTWriteFeatureList(flOld, "feat1.txt", "%3d");

Y_tot =0;
```

```
// Show the image captured from the camera in the window and repeat


while ( 1 ) {

// Get one frame
frame = cvQueryFrame( capture );
           if ( !frame ) {
                      fprintf( stderr, "ERROR: frame is null...\n"
);
                      getchar();
                      break;
                         }
IplImage* frame2=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
cvCvtColor(frame,frame2,CV_BGR2GRAY);


// Get the actual image data(pixel values)
img2 = (unsigned char*)frame2->imageData;
// Track feature between two adjacent
KLTTrackFeatures(tc, img1, img2, ncols, nrows, flNew);

KLTStoreFeatureList(flNew, ft, 1);
y_tot=0;
//conversion of pixels values to real life, 25pixels/cm
x=0.01/52;
// loop around the features
printf("\nIn second image:\n");
for (i = 0 ; i < ft->nFeatures ; i++)
{

           if((ft->feature[1][i]->x < 0.0) || (ft->feature[1][i]->y
< 0.0)){
           //std::cout << "skip" << std::endl;
           continue;
     }

printf("1-Feature #%d:  (%f,%f) with value of %d\n",
 i, ft->feature[0][i]->x, ft->feature[0][i]->y,ft->feature[0][i]-
>val);
printf("2-Feature #%d:  (%f,%f) with value of %d\n",
i, ft->feature[1][i]->x, ft->feature[1][i]->y,ft->feature[1][i]-
>val);

//printf("Feature #%d:  (%f,%f) with value of %d\n",
//i, flOld->feature[i]->x, flOld->feature[i]->y,
//flOld->feature[i]->val);

// TODO: calculate the difference in pixels between two features
// ,calculate the movement in meters
// , calculate the speed based on the movement in meters and the
framerate.

//movement difference of features between two images
d = ft->feature[0][i]->x - ft->feature[1][i]->x;
//distance moved in real life
y=x*d;
//velocity calcultaion, 0.1 is FramePerSecond
v=y/0.1;
y_tot += y;
```

```
                              }
y_average = y_tot/flNew->nFeatures;
v_average = y_average/0.5;
y_pixel = y_average/x;

cout<< " -- Dis:" << setiosflags(ios::fixed) << setprecision(2) <<
y_average<< " " << "Velocity: "<< setprecision(2) << v_average << "-
-Movement of pixel: "<< setprecision(2) << y_pixel<< endl;
cout<< "------"<<endl;

Y_tot += y_average;
cout<< " -- Total Dis:" << setprecision(2) << Y_tot << endl;


KLTStoreFeatureList(flNew, ft, 0);

#ifdef REPLACE
KLTReplaceLostFeatures(tc, img2, ncols, nrows, flNew);
#endif

KLTWriteFeatureListToPPM(flNew, img1, ncols, nrows, "feat1.ppm");
KLTWriteFeatureListToPPM(flNew, img2, ncols, nrows, "feat2.ppm");
KLTWriteFeatureList(flNew, "feat2.fl", NULL);      /* binary file */
KLTWriteFeatureList(flNew, "feat2.txt", "%5.1f");  /* text file   */

cvShowImage( "mywindow", frame2 );
// Do not release the frame!
//If ESC key pressed, Key=0x10001B under OpenCV 0.9.7(linux
version),
//remove higher bits using AND operator
int keyPress = cvWaitKey(10);
if ( (keyPress & 255) == 27 ) break;
//if ( keyPress == '\r' ) continue;

//KLTSelectGoodFeatures(tc, img2, ncols, nrows, flOld);
//KLTSelectGoodFeatures(tc, img2, ncols, nrows, flNew);

//img1=img2;
delete[] img1;
cout << "size of Image 2, pixels: "<<nrows*ncols<<" bytes:
"<<sizeof(img2) << endl;
img1 = new unsigned char[nrows*ncols];
for(int j=0; j<nrows*ncols; j++){
                                 img1[j]=img2[j];
                                }
   }

// Release the capture device housekeeping
cvReleaseCapture( &capture );
cvDestroyWindow( "mywindow" );

return 0;
 }
```

APPENDIX 2: CORNER DETECTION ALGORITHM

Corner matching detail code:

Following is the under the mask code for corner matching.

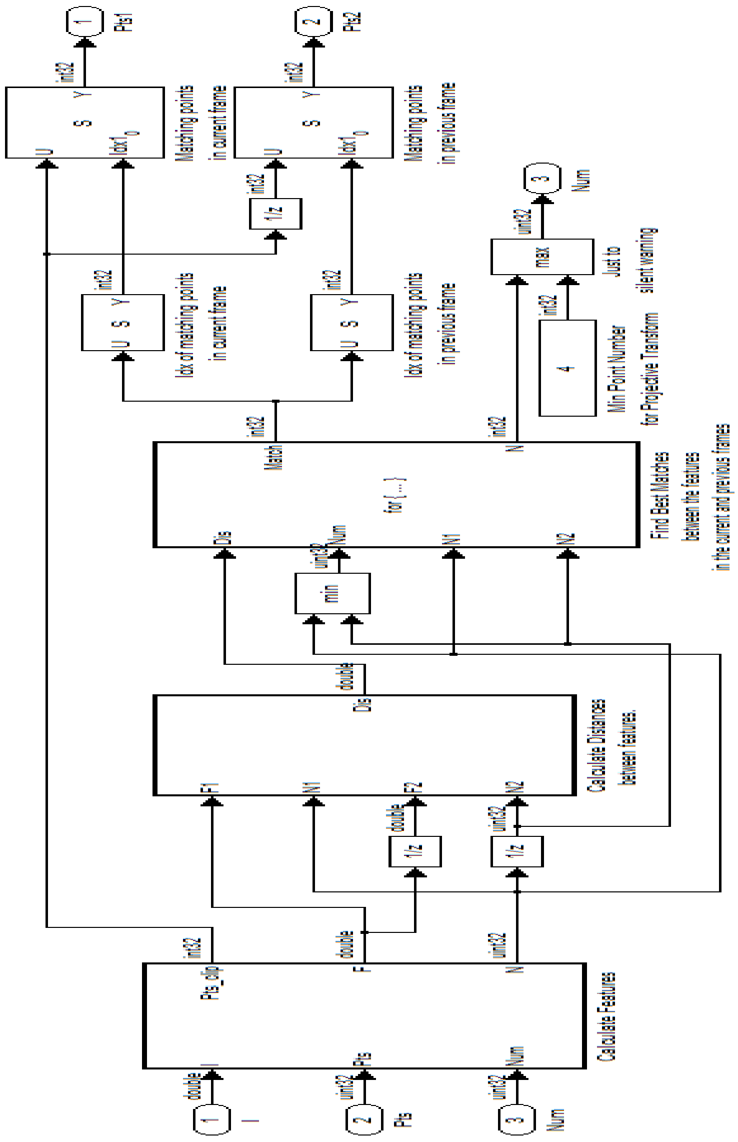

Pts1
Pts2
Num
Matching points in current frame
Matching points in previous frame
Idx of matching points in current frame
Idx of matching points in previous frame
max
Just to silent warning
4
Min Point Number for Projective Transform
for {...}
Find Best Matches between the features in the current and previous frames
min
1/z
Calculate Distances between features.
Calculate Features
I
Pts
Num

Figure: Corner matching under the mask code

Velocities calculation:

Following is the code of velocities calculation (Velocity in X and Y direction and Angular velocity).

```
function [LinearVelx,LinearVely,AngularVel] = fcn(H)
%#codegen

R=(H(1:2,1:2));
EE=eig(R);
AA=180/pi*(angle(EE(1)));
```

```
T=(H(3,:));

%LinVel=26*double(-norm(T)*sign(H(3,2)));
LinearVelx=double(H(3,2)*20);
LinearVely=double(H(3,1)*20);
AngularVel=AA;
```

Robot trajectory:

Robot trajectory display code is as follows.

```
function fcn(u)
%#codegen
coder.extrinsic('plot','hold');
plot(u(1),u(2),'.')
hold on
```