Real World Object Detection as Markers for Autonomous Aerial Motion Planning Application Using Single Camera.

Author

SANA LIAQUAT

NUST201261248MCEME35512F

Supervisor

DR UMAR SHAHBAZ KHAN

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

JUNE, 2016

Real World Object Detection as Markers for Autonomous Aerial Motion Planning Application Using Single Camera.

Author

SANA LIAQUAT

NUST201261248MCEME35512F

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Mechatronics Engineering

Thesis Supervisor:

DR UMAR SHAHBAZ KHAN

Thesis Supervisor's Signature:____________________________________

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

JUNE, 2016

Declaration

I certify that this research work titled "*Real World Object Detection as Markers for Autonomous Aerial Motion Planning Application Using Single Camera"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

SANA LIAQUAT

NUST201261248MCEME35512F

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

SANA LIAQUAT

NUST201261248MCEME35512F

Signature of Supervisor

Copyright Statement

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

I would also like to express special thanks to my supervisor Dr. Umar Shahbaz Khan for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn’t have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

I would also like to express thanks to Dr. Atta Ur Rehman for his help throughout my thesis. I would also like to thank Dr. Rab Nawaz for his help throughout my thesis and also for Image Processing and Computer Vision which he has taught me. I can safely say that I haven’t learned any other engineering subject in such depth than the ones which he has taught.

I would also like to thank Brig Dr. Javaid Iqbal and Dr. Mohsin Islam Tiwana for being on my thesis guidance and evaluation committee,

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment

Abstract

This research proposes a single camera based depth estimation technique. The proposed technique takes images of walls in a room and detects objects of interest in a cluttered environment. Having detected different objects in a room the proposed technique calculates their areas. Based on training data and polynomial curve fitting approach the proposed technique estimates the distance of the camera from the objects. For a real world object one can determine a fixed equation which can then be used to find any random distance. The approach is efficient and can affectively be applied to any indoor navigation or motion planning algorithm. Based on the estimated distances from different objects the proposed algorithm estimates the accurate location of the camera (mounted on a robot) in a room. For detection, template matching technique is used. Algorithm compares the reference template with the objects of interest in a cluttered environment by using SURF (speeded up robust features). The proposed algorithm is tested on real world images and compared with the existing depth estimation techniques.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

The research work in this dissertation has been presented in two parts. First part is related to the data collection and curve fitting approach. The objective of this part is to calculate the area of the images taken from different distances and then this area is used in a curve fitting tool to produce a general term which can calculate the distance for any given area. The second part includes the proposed solution for object detection and experimental results.

1.1Background Scope and Motivation

The focus is to estimate accurately the depth of the objects in a room by using a single camera. Depth estimation has enormous number of applications in robotics, for instance missile tracking and teleoperations. These applications require high precision and control over the movement of the robot. This high precision can be achieved with a reliable acquisition and exploitation of data available in the environment around the robot. A robot can plan its movement with precision if it can recognize the objects in the environment and gather information from them. Therefore, calculating the distance between these objects and robot, can help the robot to plan its motion. An accurate estimation of a distance of robot from the nearby objects of interest can result in an precipice movement of the robot.

Different types of sensors have been proposed in the literature for depth estimation [1], [2], [3], [4], [5], [6], most common among them are the vision sensors because of the rich amount of information available in them [7]. Single camera mounted on the robot can only give a 2-D projection of the scene on the image plane. This real world 3-D projection onto the 2-D image plane project results in information loss and therefore it is difficult to estimate depth information with only 2-D information available. Additional camera is needed to estimate the 3-D coordinates of the object of interest. One possible technique widely used in the literature is to estimate the depth information by using multiple cameras. However, this requires an additional camera and additional processing of the data. In [8] estimates the camera parameters by using defocused images and sharp images.

Advantage of the proposed depth estimation technique is that it uses only a single camera. For estimating the depth of the object, exploit the fact that an object which is close to the camera looks bigger in the 2-D image plane and keeps getting smaller as the camera moves away from the object. Therefore, the area of an object on image plane is inversely proportional to the distance between the camera and the object. The proposed algorithm uses this information for depth estimation. Good amount of training data has been used to draw a curve for the relationship between distance and area of detected object. This curve is used to general incoming data to find distance of the object from the camera. Proposed method is divided into two major stages. In the first stage the objects of interest is detected, their features are extracted and based on the features classification is performed. Once classification is done the areas of these recognized objects is extracted. In the second stage curve fitting equation is used which is calculated through training data. The area of the detected object is used in the curve fitting equation to obtain its distance from the camera.

1.2 Obstacle Detection using Sensors

There are many sensors which are used to detect obstacles in the surroundings, each sensor works differently and has its own limitations and advantages. They can't be used by mobile robots for obstacle detection and path planning. One of the reasons for not using range sensors is that their range is limited, although the range of stereo vision is infinite but with the increase in distance, data becomes noisy and accuracy reduces. Sensors are unable to detect all objects in 3D scenario. It only detects objects in its front and not the one which are above ground e.g. chair, table etc. sonar sensor, laser range finder and many other are used for obstacle detection but vision sensors are most common among them. Kinect depth sensor was introduced by the Microsoft Corp as an input device for the Xbox 360 gaming, in it each pixel value represent the distance to the point [9].

1.2.1 Vision Sensors

Vision sensor is an image processing system which is optimized for a specific task. It captures images, evaluates off with image processing algorithms and then triggers a corresponding reaction. It includes interfaces for communication.

Figure 1.1: Vision Sensor

1.3 Object Detection Techniques

There are many techniques which are used for object detection. In it there are two images, one is reference image and other is target image. Reference image is template, which is the image of specific object we want to detect in a cluttered environment and target image is the image of a cluttered environment.

1.3.1 SIFT

It is scale invariant feature transform. It describes and detects local features in images. It is invariant to translations, rotations and scaling transformations. It transforms image data into scale invariant co-ordinates relative to local features. Steps in SIFT feature selection are scale space peak selection, key point localization, orientation assignment and build key point descriptor. Advantages of SIFT are its robustness to occlusion and clutter and its efficiency is close to real time performance. SIFT is used for the image alignment, 3D reconstruction, motion tracking, object recognition, indexing and database retrieval and robot navigation.

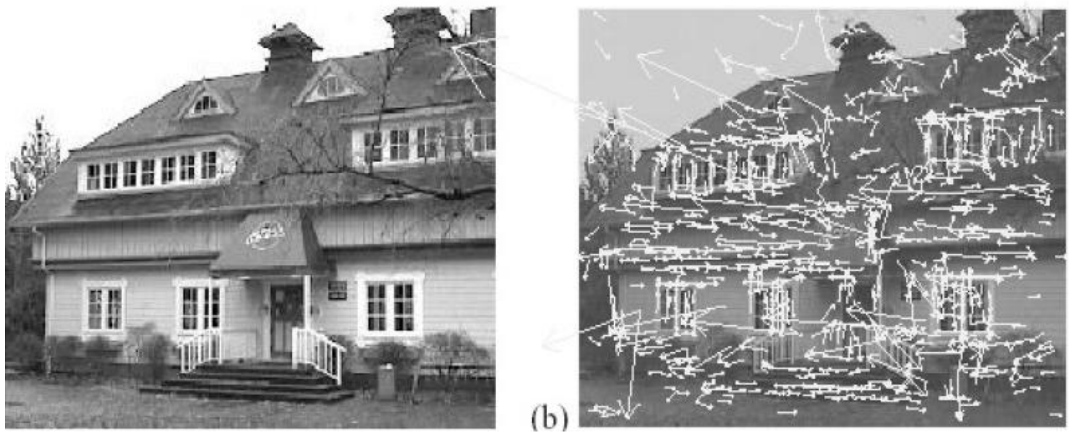


Figure 1.2: SIFT Feature Detection [10]

1.3.2 BRISK and FREAK

BRISK is a binary robust invariant scalable keypoints. It achieves comparable quality of matching at much less computation time. It is for high quality, fast keypoint detection description and matching. It is a scale and rotation invariant to a significant extent. FREAK is a fast retina keypoint. It is inspired by the human visual system and more precisely the retina [11]. Freaks are in general faster to compute with lower memory load and also more robust than SIFT, SURF or BRISK. Freak and brisk has been developed, analysis is required so it can immediately be used by android developers and security camera companies for applications such as panorama stitching, object detection, tracking and navigation [12]. Freak is more accurate than brisk. Freak and brisk generally did not perform well on pictures that had few key points and misclassified those. Brisk is a corner based detector/descriptor while SURF is blob based detector/descriptor, so you can't really get "correct" mapping that always makes sense.

1.3.3 SURF

SURF is a speeded up robust features [13]. It's a robust local feature detector used for object recognition and 3D reconstruction. It is a scale and rotation invariant interest point detector and descriptor. It is three times faster than SIFT and is good at handling images with blurring and rotation but not good at handling viewpoint change and illumination. It is more repeatable and distinctive. Whereas SIFT has detected more number of features compared to SURF but it is suffered with speed. SURF is fast and has good performance. SURF combines Hessian Laplace region detector with an own gradient orientation based feature detector. Instead of relying on Gaussian derivatives for its internal computation, it is however based on simple 2D box filters ("Haar Wavelets") shown in figure 1.3. Those box filters approximates the effect of the derivative filter kernels, but can be efficiently evaluated using integral images [14]. This evaluation requires the same constant number of loopups regardless of the image scale, thus removing the need for a Gaussian pyramid. SURF has been shown to achieve comparable repeatability as detectors based on standard Gaussian derivatives, while yielding speedups of more than a factor of five compared to standard Difference of Gaussian.

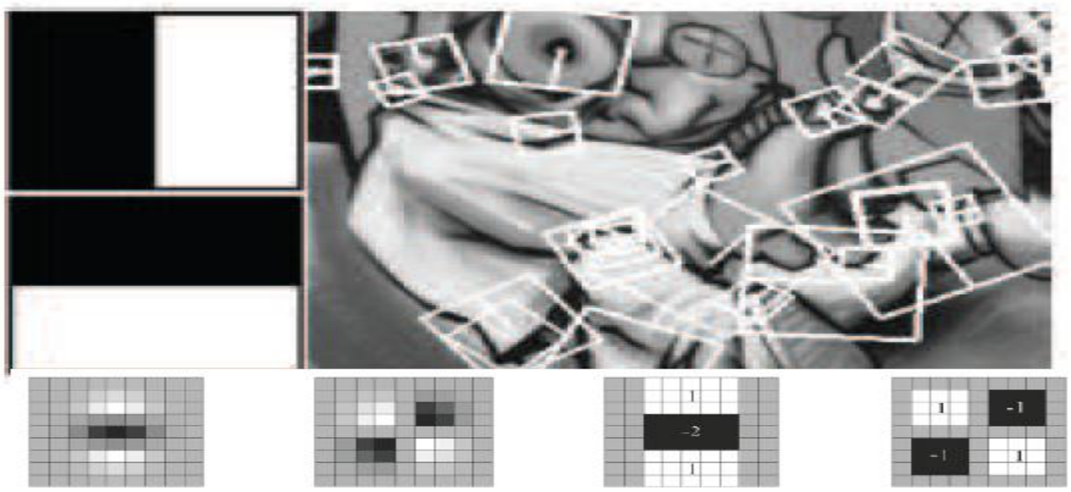


Figure 1.3: SURF detector and descriptor, computation is based on 2D box filter

1.3.4 Hessian Detector

It searches for image location that exhibit strong derivatives in two orthogonal directions. It is based on second derivatives matrix.

$$\mathbf{H\ (X,}\ \boldsymbol{\sigma}\mathbf{)} = \begin{bmatrix} \boldsymbol{Ixx(X,\sigma)} & \boldsymbol{Ixy(X,\sigma)} \\ \boldsymbol{Ixy(X,\sigma)} & \boldsymbol{Iyy(X,\sigma)} \end{bmatrix} \qquad \mathbf{(1.1)}$$

$I(X,\sigma)$ is the convolution of the image with second derivative of the Gaussian. I(x) is the integral image where X is the sum of all pixels in a rectangular area. The derivatives must be scaled by a factor related to the Gaussian Kernel σ_I^2.. The detector computes the second derivatives I_{XX} I_{YY} and I_{XY} for each image point and searches for the points where determinant of the hessians become maximal

$$\mathbf{det\ (H) = I_{xx}\, I_{YY} - I^2_{xy}} \qquad \mathbf{(1.2)}$$

In equation (1.2) det (H) is the determinant of the matrix in equation (1.1). Applying the non-maximum suppression using 3 X 3 window by computing a result image containing the hessian determinant values, the research is performed. Search window is swept over the image keeping only those pixels whose value is larger than the values of all 8 immediate neighbors inside the window. The detector returns all remaining locations whose value is above threshold θ shown in figure 1.4 below. The resulting detector responses are on corners and in strong textured images. Hessian matrix is used in SURF for keypoint detection and its determinant is used for selecting scale

1.3.5 Harris Detector

Laplacian of Gaussian is used for finding characteristic scale for a given image location, finding scale invariant and for edge detection.

$$\mathbf{L(x,\ \sigma) = \sigma^2\, (I_{xx}\, (x,\ \sigma) + I_{yy}(x,\ \sigma))} \qquad \mathbf{(1.3)}$$

Harris detector searches for the point ‘x’ where the second moment matrix ‘C’ around ‘x’ has two large eigenvalues.

$$\mathbf{C}(\boldsymbol{X},\boldsymbol{\sigma},\widetilde{\boldsymbol{\sigma}}) = \boldsymbol{G}\,(\boldsymbol{X},\widetilde{\boldsymbol{\sigma}}) * \begin{bmatrix} \boldsymbol{I}_X^2(\boldsymbol{X},\boldsymbol{\sigma}) & \boldsymbol{I}_X\boldsymbol{I}_Y(\boldsymbol{X},\boldsymbol{\sigma}) \\ \boldsymbol{I}_X\boldsymbol{I}_Y(\boldsymbol{X},\boldsymbol{\sigma}) & \boldsymbol{I}_Y^2(\boldsymbol{X},\boldsymbol{\sigma}) \end{bmatrix} \qquad \textbf{(1.4)}$$

‘C’ is the second moment matrix where ‘G’ is the Gaussian that takes role of summing over all pixels in a circular local neighborhood. Harris locations are more specific to corners. Harris points are preferable when we are looking for corners or when precise localization is required.

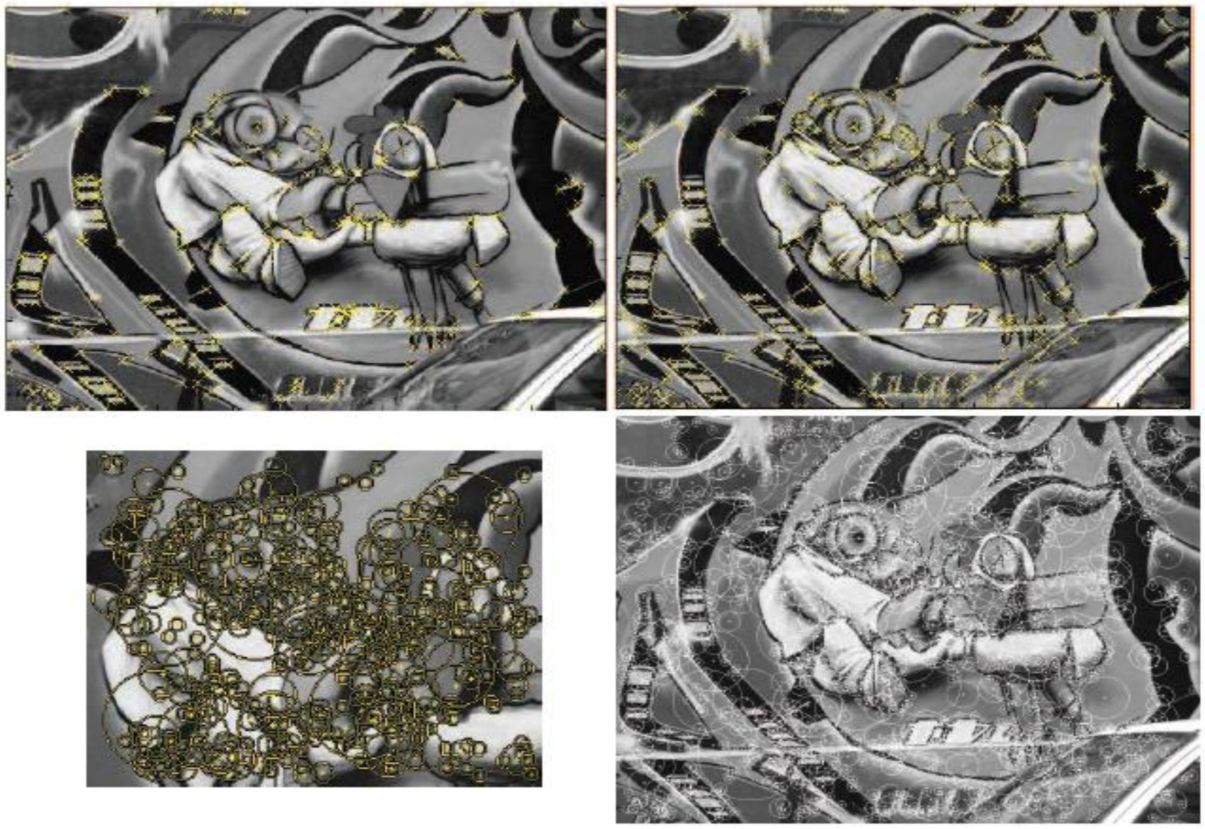


Figure 1.4: Example results of (top left) Hessian detector; (top right) Harris detector; (bottom left) Laplacian-of-Gaussian detector; (bottom right) Difference-of-Gaussian detector.

1.4 Depth Estimation

Different depth estimation techniques have been proposed in the literature for accurately estimating the distance of objects from the camera. Few of these widely used techniques are presented below.

1.4.1 Computational Stereo

Viewing the same physical point from two different viewpoints allows depth from triangulation. Much of geometric vision is based on information from two camera locations [15]. It is hard to recover 3D information from a single 2D image without extra knowledge. Stereo vision is ubiquitous in nature. Stereo involve problem of calibration, matching and reconstruction. Range of stereo vision camera is infinite but by increasing the distance, data become noisy and it reduces the accuracy. Computational stereo is shown in figure 1.5 below.

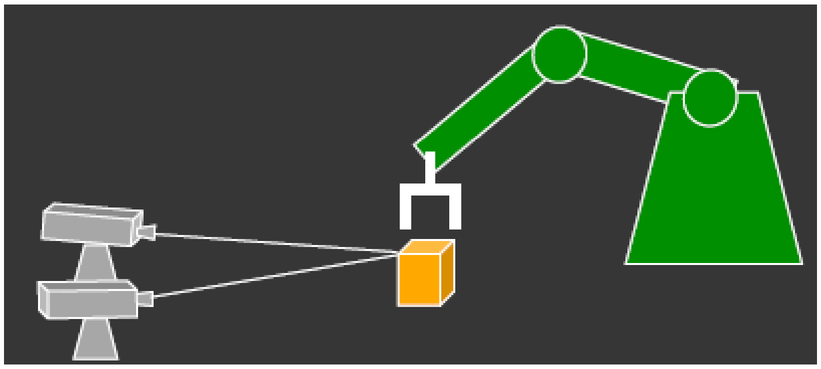


Figure 1.5: Computational Stereo

1.4.2 Stereoscopic Depth Estimation

Technique used for recording and demonstrating stereoscopic images. It creates a delusion of depth using two pictures taken at slightly different position. There are two possibilities for obtaining stereoscopic images, first is by using stereoscopic camera specifically designed for the particular task, it contains two lens mounted within a single camera. Second option is to use a system based on two single lenses joined together both designs are shown in figure 1.6 below

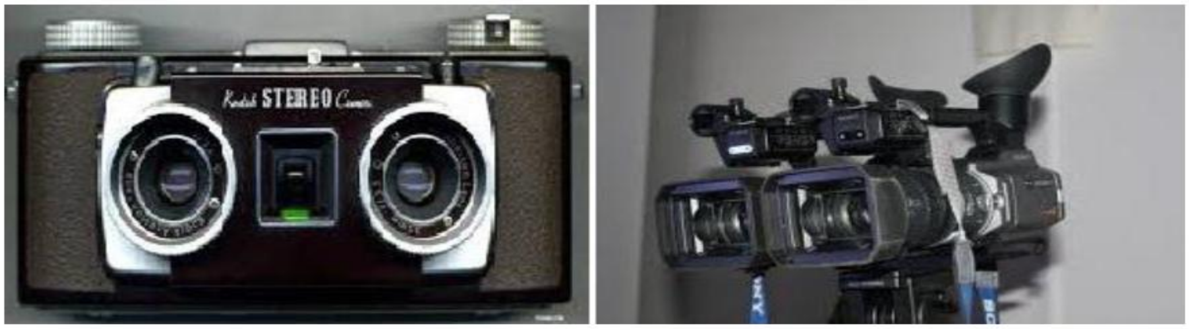


Figure 1.6: Stereoscopic camera (Left), system designed to acquire stereoscopic images (Right)

Distance is calculated from difference between the images taken from cameras, their focal length and distance between the positions of the lens [16]. It is assumed the lens should be horizontally aligned; pictures should be taken within the same environment and at the same instant. Any systematic or human error can be judged as shown in figure 1.7.

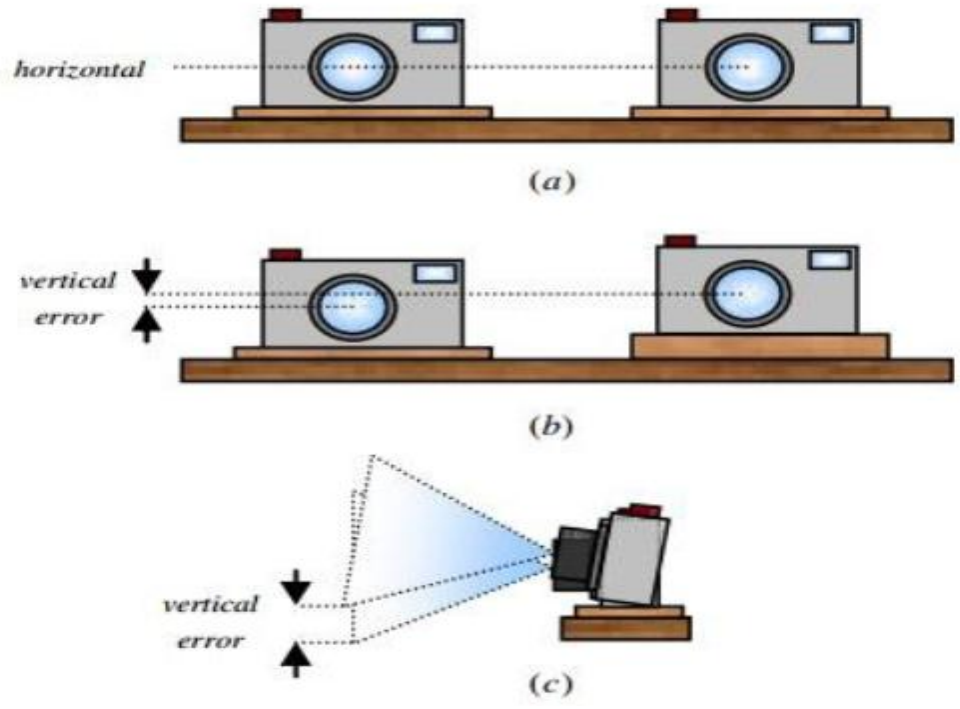


Figure 1.7 (a) Alignment of camera horizontally (b) and (c) Vertical error [16]

In (a) it shows the alignment of cameras horizontally, in (b) and (c) cameras show the vertical error of the two cameras. Distance of a particular object in an image can be calculated by trigonometric identities shown in figure 1.8

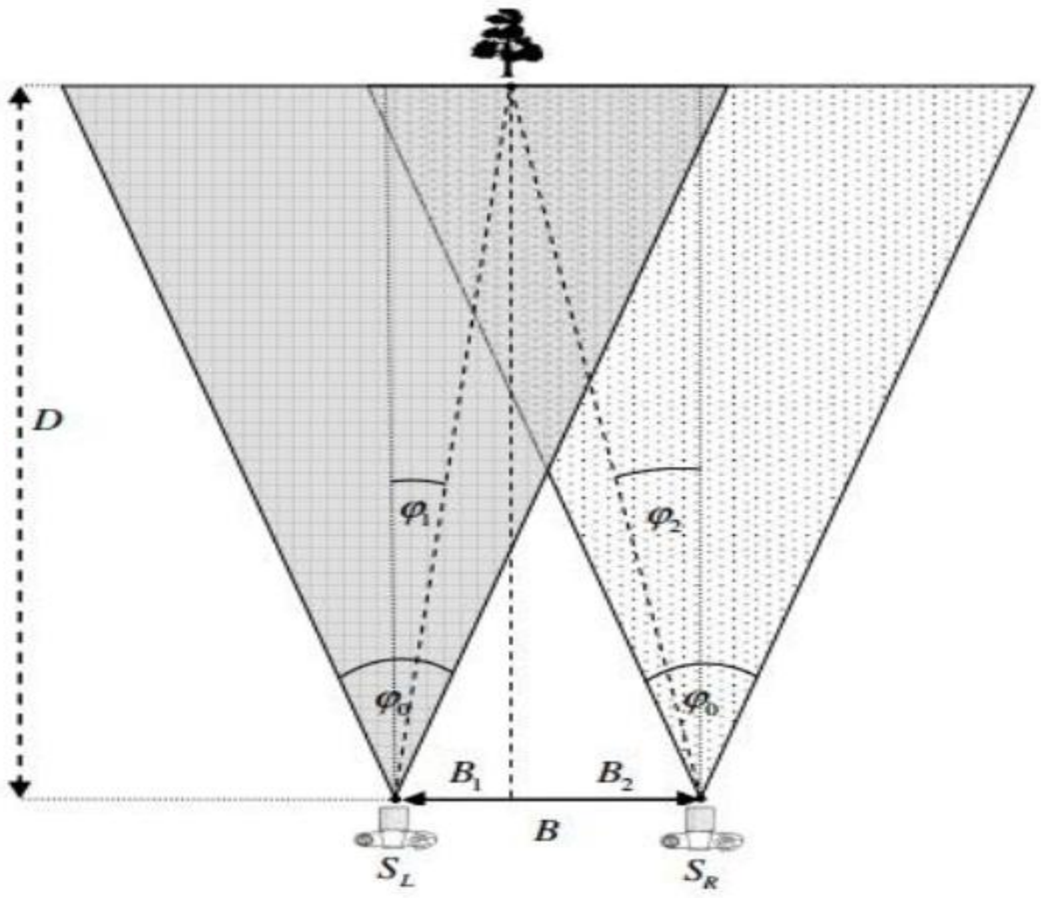


Figure 1.8: Stereoscopic image for distance estimation

$$\boldsymbol{B = B1 + B2 = Dtan\varphi 1 + Dtan\varphi 2} \quad \textbf{(1.5)}$$

$$\boldsymbol{D = \frac{B}{tan\varphi 1 + tan\varphi 2}} \quad (\mathbf{1.6})$$

Where ‘B’ is the distance between two lenses and φ is their respective angle. Ratio of these two parameters ‘D’ provides the estimated distance.

1.4.3 Markov Random Field

Most of the research work in 3D reconstruction is on binocular vision; beside just local features (geometric differences) other factors like haze, color, defocus, gradients; termed as monocular cues can also be utilized. Algorithm based on Markov Random Field (MRF) uses monocular cues for depth calculation.

Image is divided into small patches. Depth of each patch is estimated using absolute depth and relative features. Feature that capture texture variations, texture gradients and haze is selected. Texture energy is calculated using laws' mask [17]; Haze being the low frequency element in the color it can be obtained by low averaging filter, finally texture gradient is the convolution of intensity channel with neighboring edges as shown in the figure 1.9

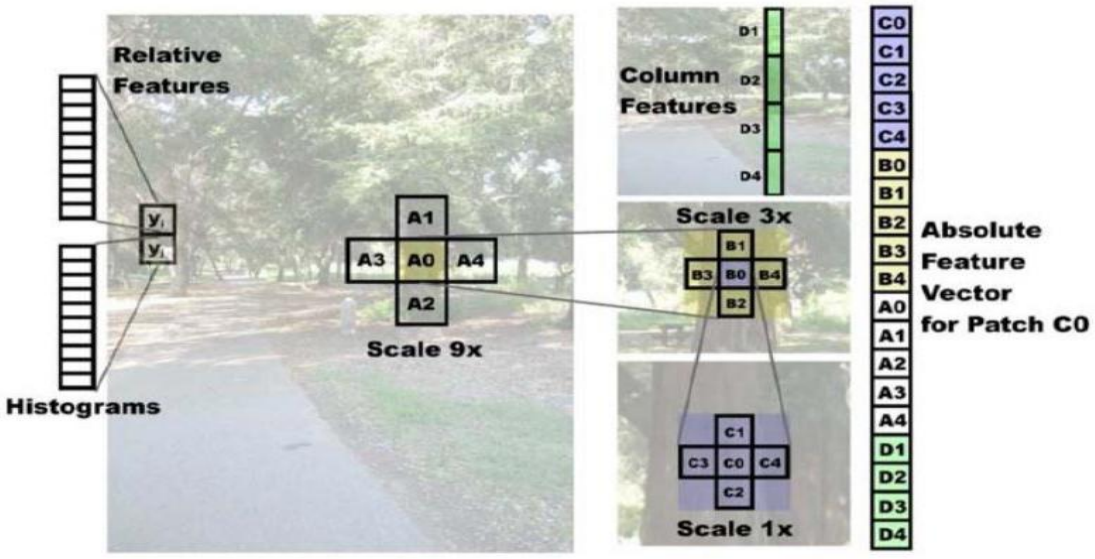


Figure 1.9: Absolute and relative depth features of a patch [18]

Depth of every patch depends upon the features of that patch and its neighboring patches. Therefore, probabilistic models can be used to drive the native relationship among the patches.

1.4.4 Monocular Cues

There are numerous monocular cues like texture variations and gradients, defocus, color and haze etc. it contains important depth information. Depth estimation from single image using monocular cues needs significant amount of prior knowledge. In it supervised learning is applied for depth estimation from single monocular images of unconstrained outdoor and indoor environments. Image is divided into small patches and depth estimation of each patch take place. Absolute features and relative features are used for absolute and relative depth estimation of patches [19].

1.4.5 Depth Estimation in Digital Imaging

Depth of a particular object in an image can be obtained from several ways. It can also be derived through theory of optics. Beam of reflected light from an object after passing through convex lens must converge at a focal point.

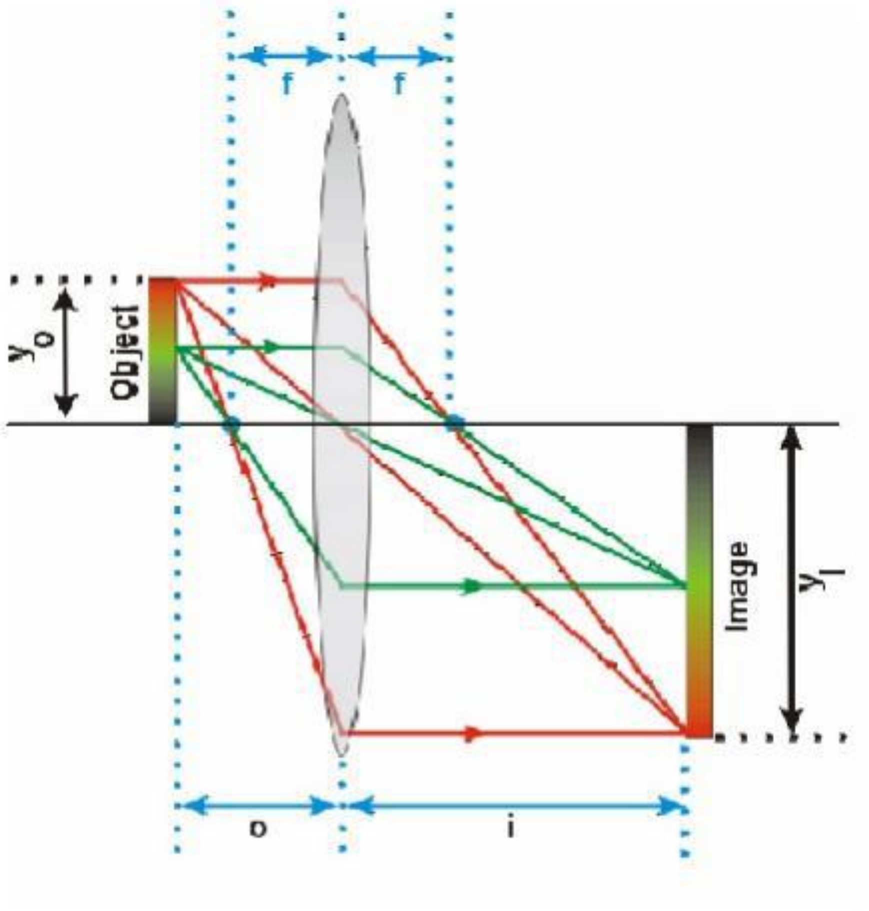


Figure 1.10: Theory of Optics [20]

Magnification of the object is defined in the below mentioned equation, which shows if Y_i is equal to Y_o, magnification will be 1.

$$M = \frac{Y_i}{Y_o} \qquad (1.7)$$

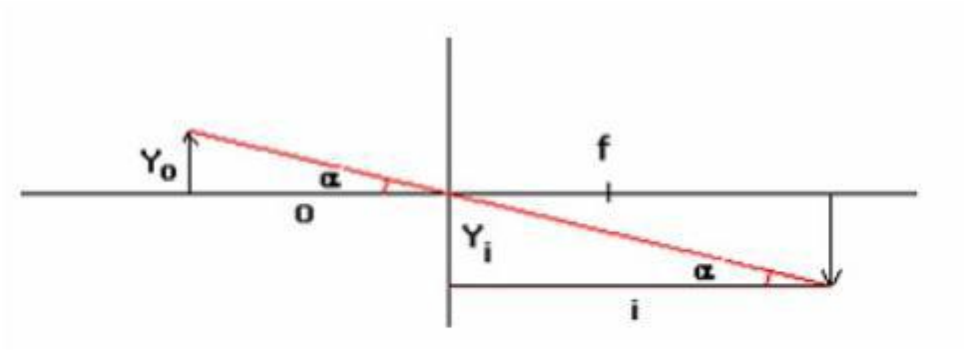


Figure 1.11: Depth Estimation using Theory of Optics

As shown in figure 1.11, point of consideration is 'O'. Simple trigonometric identities can be used to produce the desired result.

$$\tan\alpha = \frac{Y_i}{i} \tag{1.9}$$

$$\alpha = \tan^{-1}\left(\frac{Y_i}{i}\right) \tag{1.10}$$

Substitute and solve for 'O':

$$\tan\alpha = \frac{Y_o}{O} \tag{1.11}$$

$$O = \left(\frac{Y_o}{\tan\alpha}\right) \tag{1.12}$$

$$O = \frac{Y_o}{\tan\left[\tan^{-1}\left(\frac{Y_i}{i}\right)\right]} \tag{1.13}$$

$$O = \frac{iY_o}{Y_i} \tag{1.14}$$

1.5 Object Classification

There are several techniques currently being used for classification of certain objects in the image. It not only depends upon the technique itself, some intrinsic properties of the object also play an important role for the identification.

Theoretical model for optical flow fields (velocity of pixels in the image sequence) can be used for detection of obstacle during runtime [21]. In order to perform detection of obstacle image is divided into two layers. Pixels which match the optical flow model are in first layer and which do not match corresponds to the second layer. Novelty of detecting the obstacles is to separate them from the ground floor in the image sequence. Moving obstacles are recognized based on the motion in the sequence of gathered images. Initially, model of the environment is extracted then theoretical optical field model is compared with the actual video feed to observe the differences. Figure 1.12 shows the detected moving object, where highlighted red color depicts the contour of the detected moving obstacle.

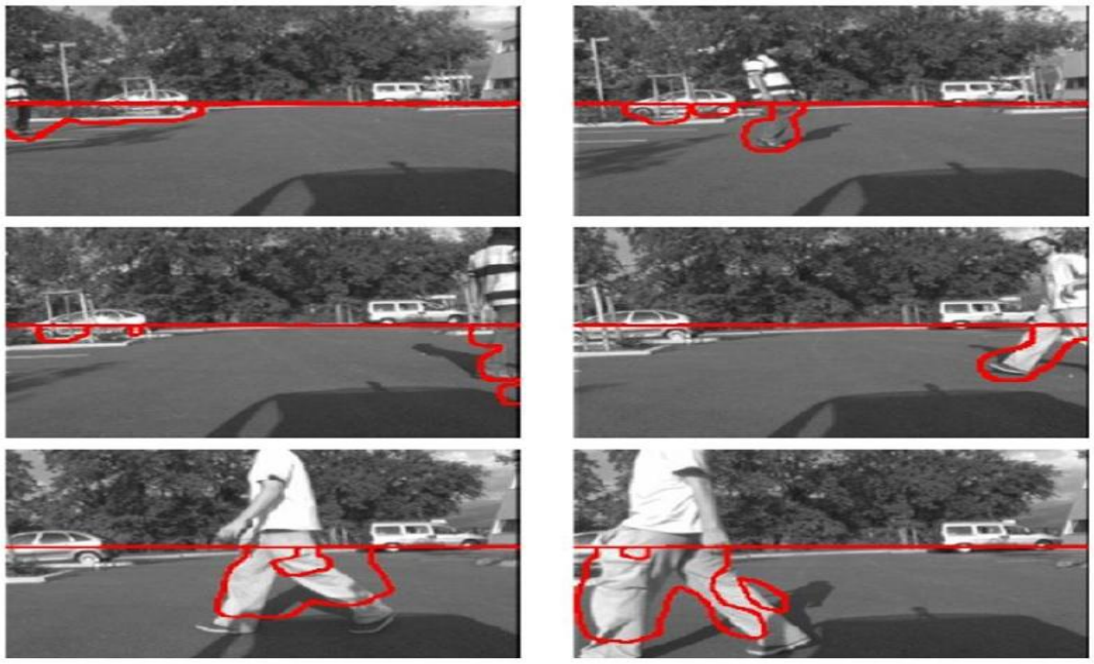


Figure 1.12: Detection of moving obstacles [21]

Once depth and classification of particular object is obtained successfully, position localization within the bounded work space can be worked out using simple mathematics. Autonomous aerial maneuvering has various applications (Robotics, Vehicles Automation, Military and Commercial use and industrial utilization). There are different vicinities where an autonomous indoor aerial surveillance is required, such as: surveillance of highly secured and sensitive areas, transportation of different objects from one point to another.

1.6 Autonomous Aerial Motion Planning

One application of depth estimation is autonomous aerial vehicles. These autonomous vehicles have advantages such as safety, lower emissions, reduced congestion and greater mobility [22]. Obstacle detection and real time planning of collision free trajectories are keys for the fully autonomous operation of micro aerial vehicles (MAVs) in restricted environments. Micro aerial vehicles (MAVs) are a famous choice of autonomous vehicle. This is due to their low cost, flexibility and choices of huge number of applications. These vehicles are used for aerial photography, inspection and surveillance missions [23].

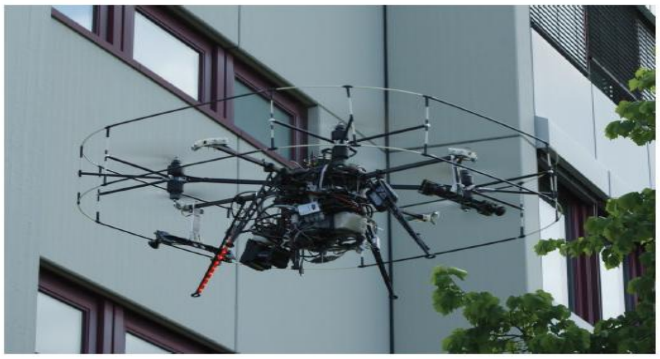


Fig 1.13: Micro Aerial Vehicle (MAV)

Micro aerial vehicles have the capability of automatic landing as well. Depth estimation and obstacle detection has enormous number of applications in unmanned aerial vehicles (UAV) and autonomous aerial robots.

CHAPTER 2: DATA COLLECTION

Depth calculation has significant importance in autonomous motion planning algorithm. System should be capable of detecting any object in its way and execute a timely decision to avoid collision. As discussed in the introduction that aim of the proposed algorithm is to assist a robot to plan its motion by recognizing the everyday objects in a known indoor environment. Most common objects in a room can be switch boards, wall clocks, doors and frames. Therefore, the proposed algorithm considers switch boards, frames, sceneries and wall clocks as objects of interest; however, the algorithm can be extended to consider other common objects as well. Proposed algorithm detects these objects in the room identifies their areas and then estimates the distances between object and the camera. To achieve depth valuation and object detection, the derived algorithm is divided in two major categories. Phase-I includes training data collection and using this training data we evaluate a curve fitting expression which can further be used as a generalized expression for calculation of distance of an object from the camera given the area of the object. In the second phase to estimate the area of objects we first detect them using object detection techniques and estimate their area and then use the area into curve fitting expression to calculate the distance of object from the camera.

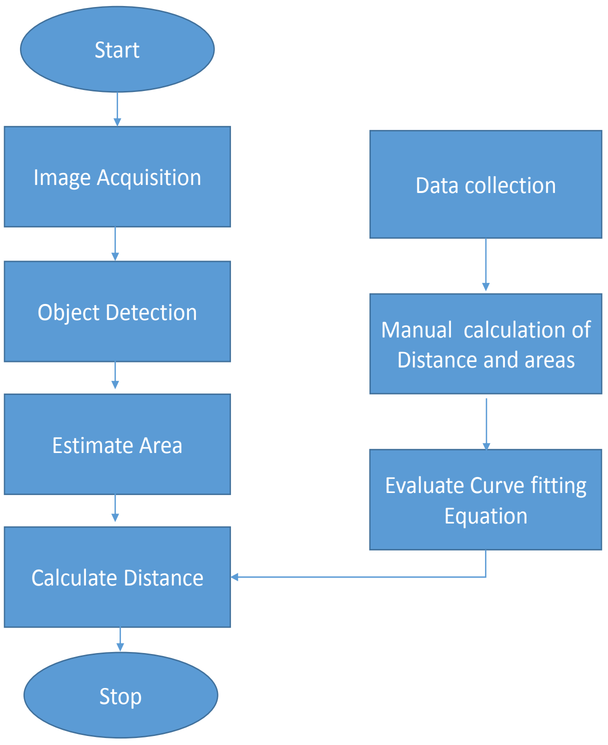


Figure 2.1: Flow chart representing general procedure

As shown in figure 2.1, the proposed depth estimation algorithm consists of following stages:

- Phase-I
    - Image acquisition
    - Training data collection
    - Evaluating curve fitting expression
- Phase-II
    - Object Detection
    - Estimating areas of detected objects
    - Distance calculation from the camera

2.1 Image Acquisition

Everyday routine camera is used for image acquisition. It is a good practice to take images in all lighting conditions. Different aspects of the real world conditions are considered while taking the training data. Firstly, to cover the changing illumination conditions, training data is taken in different illumination conditions. Multiple images of switch boards, frames, sceneries and wall clocks from different distances are taken for the purpose of training the system. These images are taken with an everyday routine camera. Images are taken in a cluttered environment in which objects are hanging on the walls with different orientations.

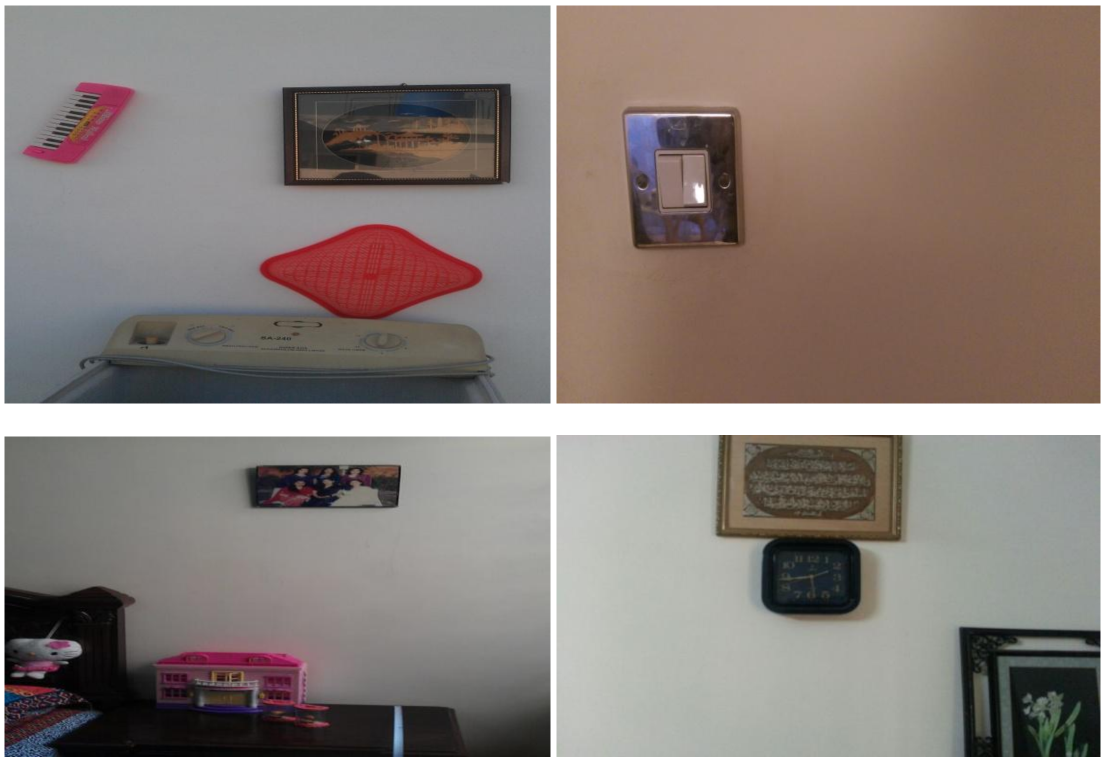


Figure 2.2: Sample images during image acquisition process

2.2 Distance Vs Area

To train the system multiple captured images of wall clock, switch boards, frames, art box and sceneries are taken from different distances, and their areas are manually calculated for training purpose. The idea is to generate a general term for estimating the distance of an object from the camera. This term should take area of an object as an input and should return an accurate distance between object and the camera. The general term is evaluated by using these multiple images and their manually calculated areas.

Table 2-1 Table 2-2 and Table 2-3 shows the manually calculated areas of the objects (Clock, Scenery and Art Box) with images taken from different distances. Data given in Table 2-1, 2-2, 2-3 can be used in a curve fitting tool to produce a general term which can calculate the distance for any given area.

Table 2-1: Covered Area by the object (Clock) versus distance from which the image is taken.

Serial No	Covered Area (pixels)	Distance (Feet)
1	235680	2
2	92400	3
3	73935	4
4	51072	5
5	37440	6
6	14762	7
7	10504	8
8	8649	9
9	7470	10

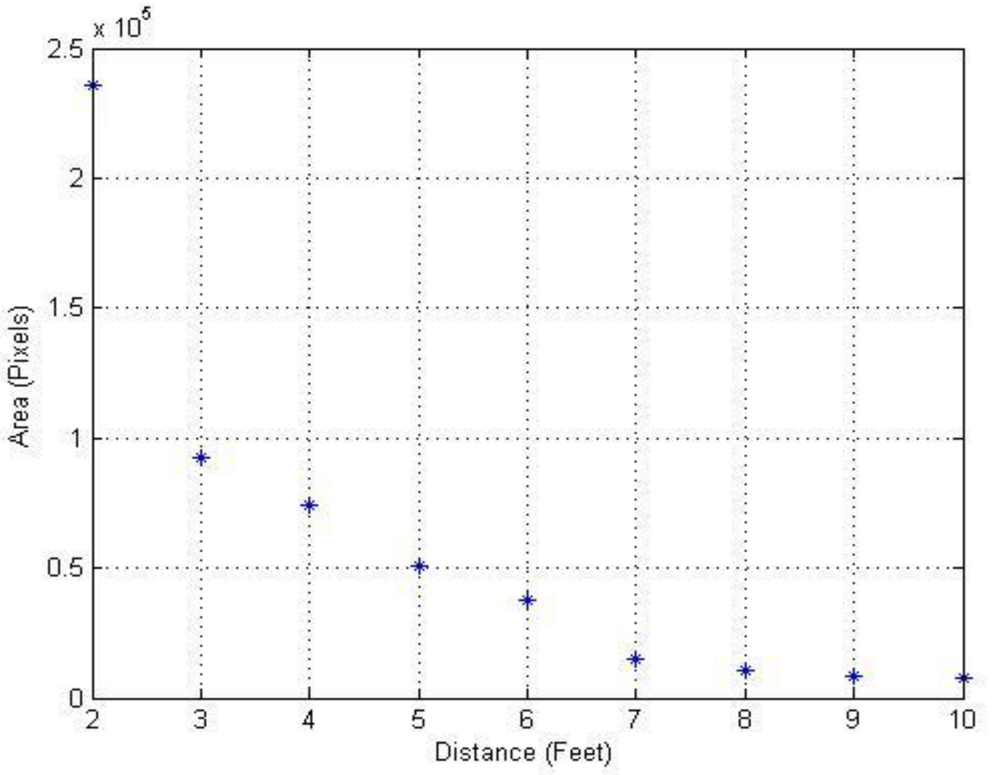


Figure 2.3: Plot of dataset versus covered area by the object (Clock).

Figure 2.3 contains a dataset of covered area by the object (Clock) in the 2-D image and respective distance from where the image is taken. In order to get a mathematical model which accurately demonstrates the pattern as shown in the figure 2.3 curve fitting is applied.

Table 2-2: Covered Area by the object (Scenery) versus distance from which the image is taken.

Serial No	**Covered Area (pixels)**	**Distance (Feet)**
1	660156	2
2	437600	3
3	382536	4
4	246000	5
5	159856	6

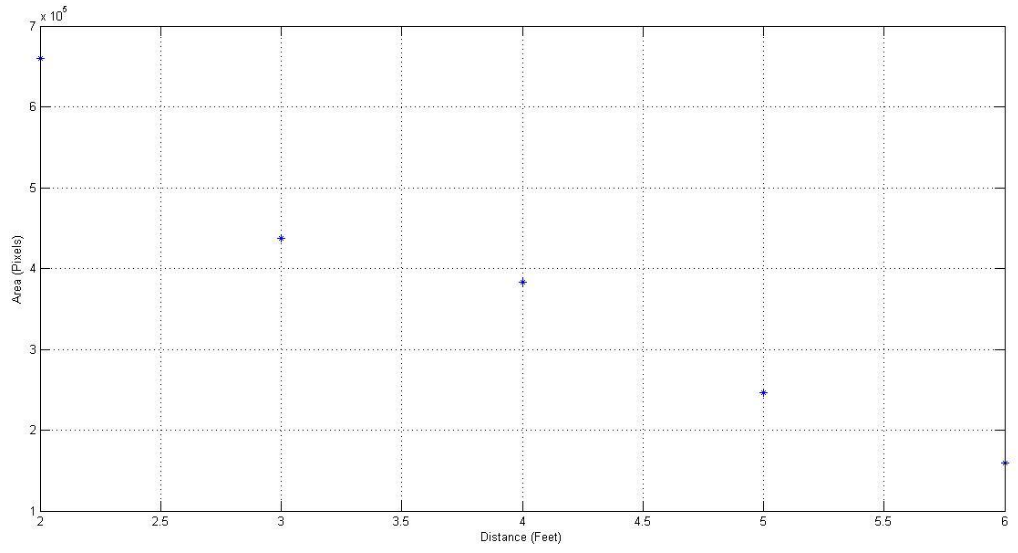


Figure 2.4: Plot of dataset versus covered area by the object (Scenery).

Table 2-3: Covered Area by the object (Art Box) versus distance from which the image is taken.

Serial No	**Covered Area (pixels)**	**Distance (Feet)**
1	854658	2
2	464130	3
3	305208	4
4	202311	5
5	156843	6
6	108108	7
7	89964	8
8	71262	9
9	53298	10
10	47763	11

11	36936	12
12	30429	13
13	26325	14
14	20160	15
15	19656	16
16	17550	17
17	14985	18

As observed from the values illustrated in the tables above covered area by the object at two feet is more as compared to the others and as the distance increases covered area decreases. Values shown against the column covered area in the table 2-1 is the arithmetical mean of atleast 10 observations.

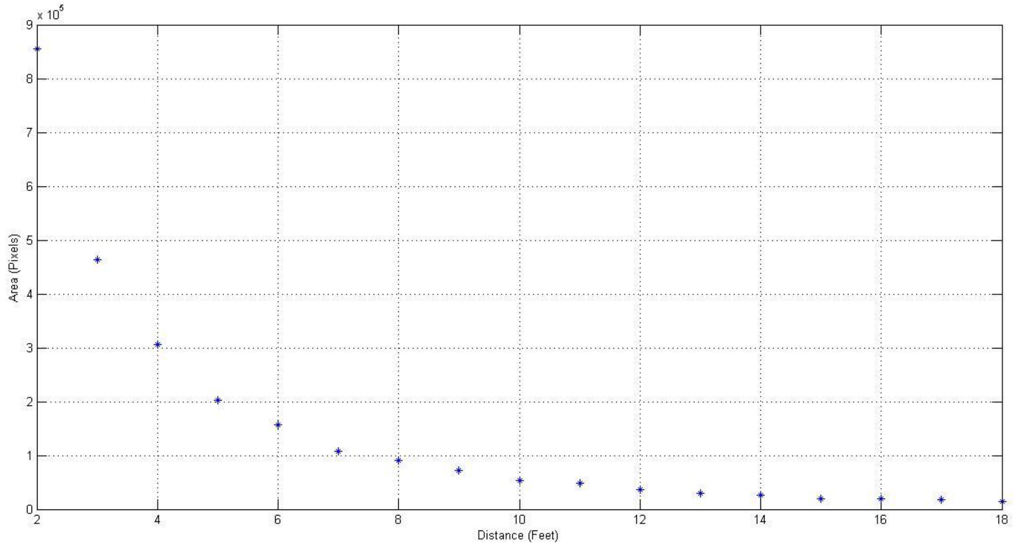


Figure 2.5: Plot of dataset versus covered area by the object (Art Box).

2.3 Curve Fitting

Curve fitting is capturing trend in data by assigning single function across the entire range [24]. There are many curve fitting tool which can be used for producing a general term [25]. Prominent techniques among the curve fitting techniques are quadratic polynomial, rational and power fit curve fitting techniques. In this section we will discuss some of the curve fitting and their characteristics [26] [27].

2.3.1 Quadratic Polynomial

Equation for quadratic polynomial is shown below

$$\boldsymbol{f(x) = \mathrm{p1} * \mathrm{x}^2 + p2 * x + p3} \qquad (\mathbf{2.1})$$

Where ;

p1 p2 and p3 are constants

x = covered area by object

f(x) = calculated distance from the object

In problems with many points, increasing the degree of polynomial fit. It does not always result better fit, it result in poorer fit of data. In those cases low order polynomial fit is used.

2.3.2 Rational Polynomial

Equation for rational polynomial is shown below

$$\boldsymbol{f(x)} = \frac{\mathbf{p1 * x^2 + p2 * x + p3}}{\mathbf{x^2 + q1 * x + q2}} \qquad (\mathbf{2.2})$$

Where ;

p1 p2 p3 q1 and q2 are constants

x = covered area by object

f(x) = calculated distance from the object

Rational function can model complicated structures with low degree in numerator and denominator, constant term in the denominator is usually set to 1.

Although rational curve fitting is better as compared to the quadratic polynomial but power fit can be used to obtain more adequate results.

2.3.3 Power Fit

Through experiments and by following the work of [28]. It is figured out the powered fit techniques best fits our data. General term for power 2 curve fitting tool can be written as

$$\boldsymbol{f(x) = \mathbf{a} * \mathbf{x^b} + \mathbf{C}} \qquad \textbf{(2.3)}$$

By using the curve fitting tool and using the data of Table I expressions for a, b and c comes out to be

a = 884.42 (-78.55, 247.4)

b = -0.1892 (-0.5714, 0.193) **(2.4)**

c = -6.169 (-29.93, 17.59)

x = covered area by the object

f (x) calculated distance from the object

Results obtained using power fit is meaningful; thus equation (2.3) is considered as a final model equation which will be used for depth estimation. Where 'a' 'b' and 'c' are constants 'x' represents covered area by the object and function 'f(x)' is corresponding distance from which the image is taken. Algorithm uses this equation as a baseline for distance evaluation. There are different other constraints which take part in the selection criteria for best curve fitting equation.

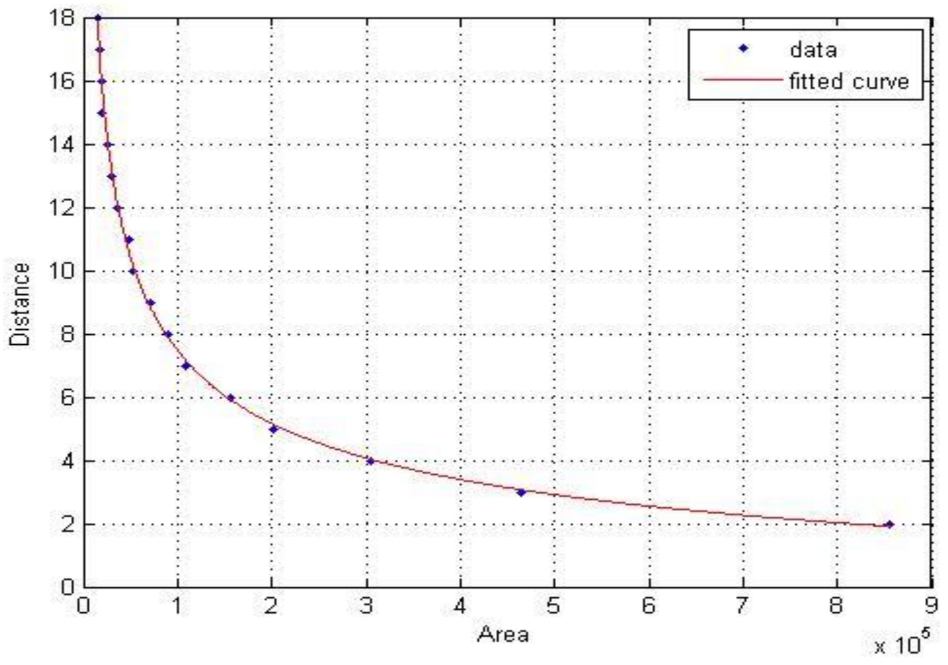


Figure 2.6: Curve Fitting (Power Fit).

With the calculation of curve fitting expression the first phase of the proposed algorithm completes. In the next phase which is discussed in the next chapter, objects of interest are detected, the areas are calculated and above derived curve fitting expression is used to calculate the distance of objects from camera.

2.4 Camera Calibration Using Formula

Camera is calibrated once for the given data, in that case there is no need of training.

(width of the object in meter/number of pixels of the object in columns) = (2 * distance1 * tand(Horizontal angle of view of camera/2))/number of pixels of the camera in columns

(height of the object in meter/number of pixels of the object in rows) = (2 * distance 2 * tand(vertical angle of view of camera/2))/number of pixels of the camera in rows

Final Distance = mean (distance1, distance2)

Height and width of the object is taken in meter, number of pixels of object are calculated in row and column. Number of pixels of camera are also taken in row and column.

To find angle of view following equations are used.

HTC M8 ultrapixel pixel size 2 micrometer, focal length f can be calculated

f = 28mm (27.54 to be exact)

sensor size 1/3

theta1 (horizontal angle of view of camera) = 2*invtan(d1/2f)

d1 = 2 micrometer * number of pixels (horizontal side, number of columns, 2688)

d2 = 2 micrometer * number of pixels (vertical side, number of rows, 1520)

theta2 (Vertical angle of view of camera) = 2*invtan(d2/2f)

Table 2-4: Comparison of error in depth using power fit and formula based calibration.

Serial No	**Error (Power Fit)**	**Error (formula)**
1	0.2686	0.8567

2	0.2706	1.4694
3	0.3681	2.1131
4	0.2534	2.6668
5	0.4932	3.3587
6	0.3612	3.9002
7	0.4830	4.5342
8	0.8514	5.2747
9	0.5592	5.7361
10	0.3913	5.9445

It shows error is comparatively more in formula calibration as compare to curve fitting technique; it is due to some error in the value of angle of view of camera. But by finding the exact angle of camera, results can be improved.

CHAPTER 3: OBJECT DETECTION

Image processing is done for the object detection. Set of images gathered in the data collection phase will be required. All captured images must undergo certain criteria before distance can be analyzed. Processing [29] [30] is purely based on input images and technique used for that improvement. Besides just improving the quality of the image, image is to be used for the distance estimation.

3.1 Pre-Processing

In order to overcome the problems occurred due to the low quality image, pre processing is done before feature extraction process with respect to the object detection rate. Pre processing of the captured image can give better results in recognition [31]. There are various techniques for image pre processing which can be used to enhance the captured image and increase the accuracy. De-noising, image normalization, filtering, histogram equalization, image resizing and cropping are the techniques to enhance image quality and improve recognition rate. Pre processing is done before feature extraction from the image. Input image is converted into gray image in the proposed algorithm.

3.1.1 Histogram Equalization

In the proposed algorithm, histogram equalization is used to improve the quality of images and to get a accuracy. Histogram equalization is an effective method to deal with varying light conditions in large dataset of images [32]. It is a functional method in stretching the range of gray levels and expands the contrast of the image. Histogram equalization is generally used to enhance the low contrast and appearance of the image by increasing the distance among the most frequently appeared intensity values through a non linear mapping function and is usually a histogram modification approach. In Figure 3.1, there are few images of cluttered environment after applying histogram equalization.

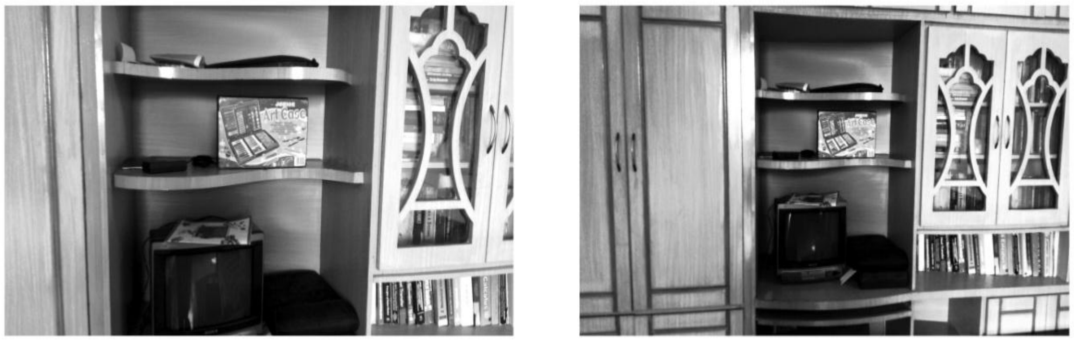


Figure 3.1: Histogram Equalization of input images

3.2 Object Detection

The proposed algorithm uses the template matching technique for detecting the required objects in a cluttered environment. A reference template of the objects is used to train the system and then features of the reference template are tried to match with the features extracted from a target image. Therefore, the first stage of the proposed algorithm requires feature extraction [33].

3.2.1 Feature Extraction

Speeded-up robust features (SURF) are used in the proposed algorithm to extract features from the reference and target images. The SURF extracts features from these two images and finds the correspondences between two images. The SURF algorithm work in three steps, the first step is detection of key feature points which corresponds to picking up the strong features points in the image. If we suppose the reference image is I_{ref} which corresponds to the object "art box" and the target image is represented as I_{target} corresponds to a clutter image which contains "art box" and other non-required objects. The first step of the SURF algorithm would detect strong points in I_{ref} and I_{target}. The second step of the SURF algorithm extracts features by using the features points, this is also known as feature descriptors extraction. Third and final step of the SURF algorithm matches the features between I_{ref} and I_{target}. This final step gives those points on the I_{target} which match features points of $I_{ref.}$

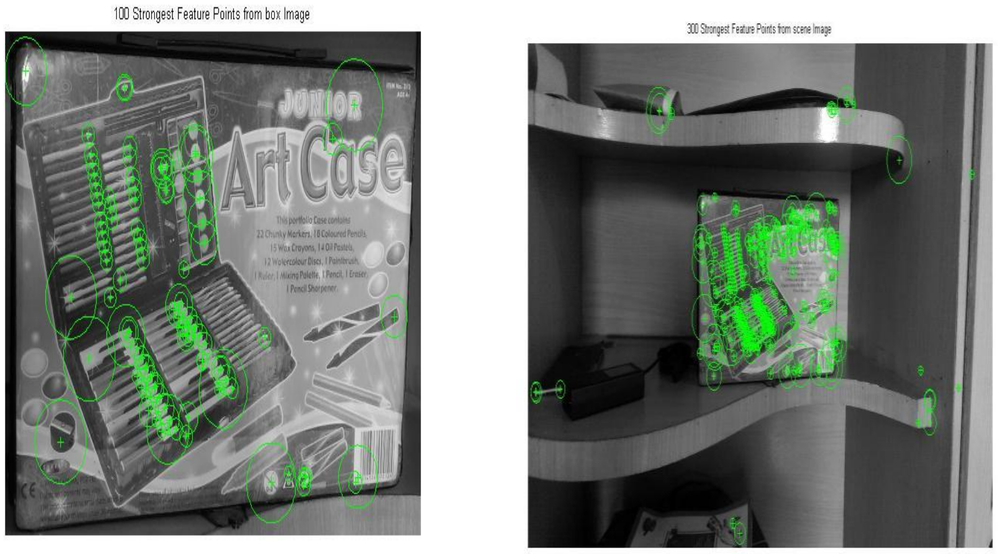

100 Strongest Feature Points from box Image
300 Strongest Feature Points from scene Image

Fig 3.2: Feature extraction of a reference and a template image

3.2.2 Locating Required Object

Once matched features between I_{ref} and I_{target} are obtained the next step is to locate the required object in I_{target} whose reference template is $I_{ref.}$ This is achieved by using the transformation relating the matched points while eliminating the outliers. To draw a boundary of the required object in I_{target}, polygon of the reference image I_{ref} is drawn and this polygon is transformed into the coordinate system of the target image. This transformed polygon tells the location of the required object in $I_{target.}$

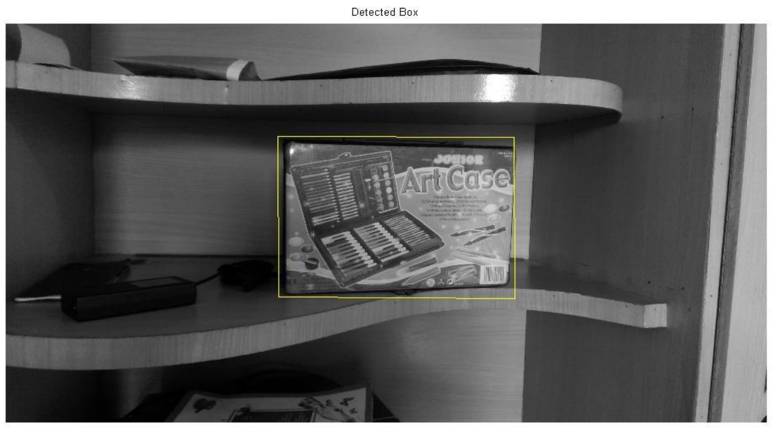


Figure 3.3: Object Detection in Cluttered Environment

3.3 Estimating Area Of Object

At this stage only rectangular objects are considered and the bonding box representing the required object in I_{target} is also a rectangle. The area of the required object calculated by using the pixel coordinates of the bounding box. This area is further used in the curve fitting equation to estimate its distance from the camera.

3.4 Estimating Distancing Using Area

Once the area of the object is obtained this area is used in equation (2.3) to calculate the distance of the object from the camera. Next section explains detailed experiments conducted to validate the performance of our proposed algorithm.

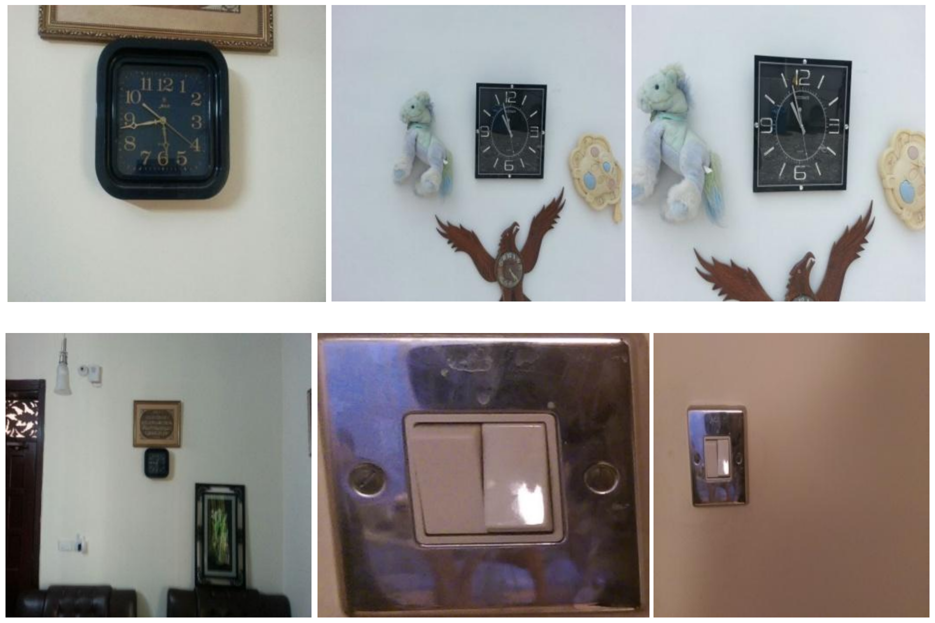


Figure 3.4: few of the images used as a training data, captured in cluttered environment in different lighting conditions.

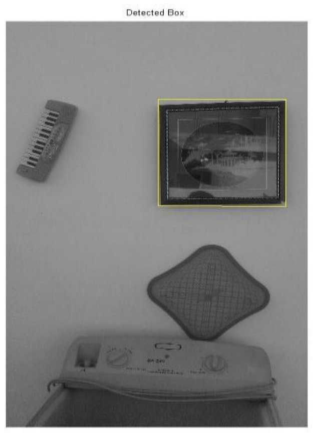
CHAPTER 4: EXPERIMENTAL RESULTS

To validate the performance of the proposed algorithm, it is tested on multiple objects.

4.1 Data Set of images

In this algorithm, large amount of data set is taken to make the system robust and in order to check the accuracy of algorithm. Maximum data set of 30 images is taken. Real world objects e.g. frames. Sceneries, clocks, switch boards, art boxes are used. SURF detected all real world objects with accuracy.

Different types of objects are detected and their distances from the camera are calculated through the proposed algorithm. Two different types of clocks are used both in a different lighting conditions; one indoor and the other outdoor, secondly a switch board, frames, sceneries, art box are detected through the proposed algorithm. To match the real world scenario all objects are in a cluttered environment.



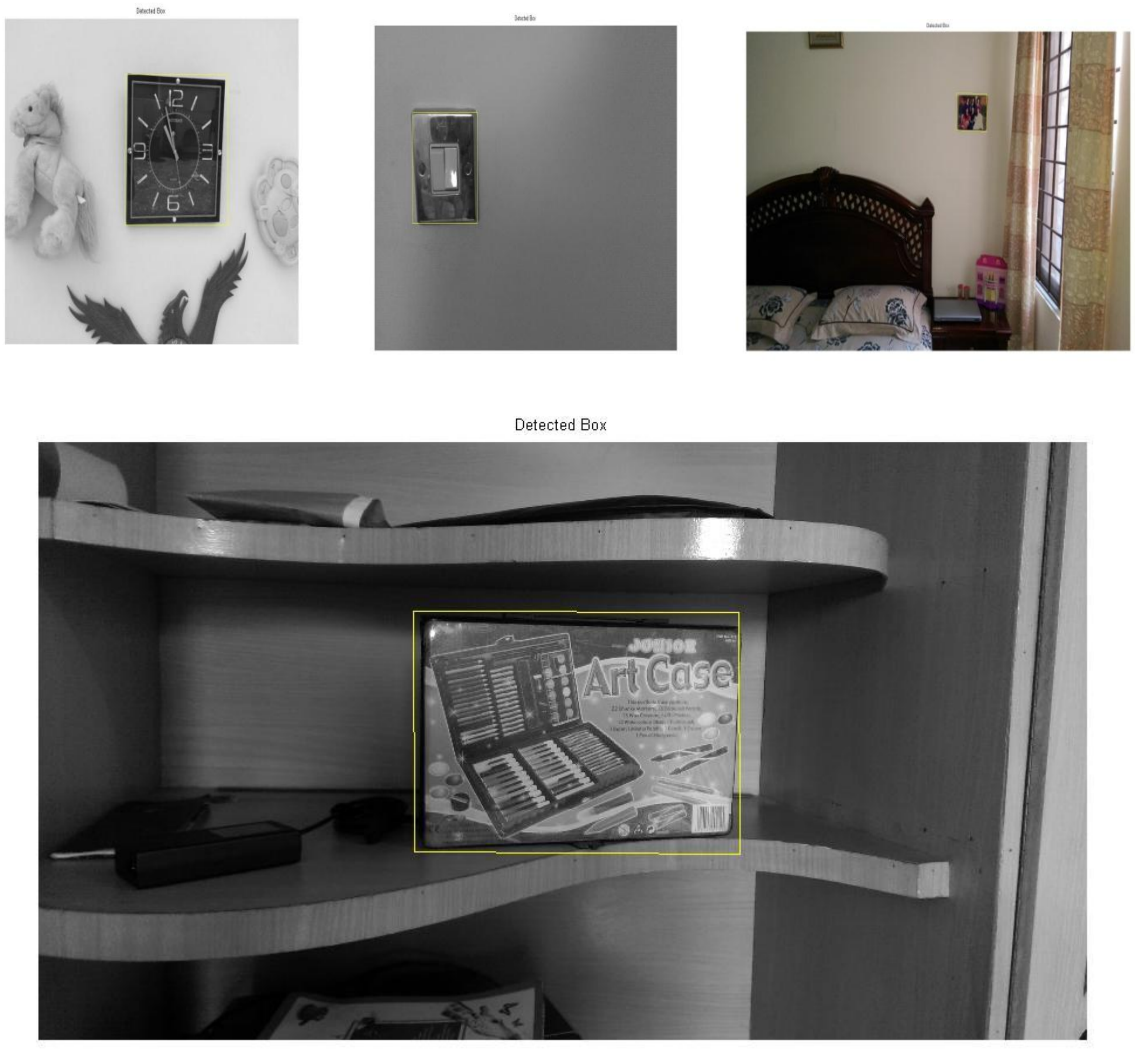


Figure 4.1: Sample images taken in different lighting conditions

In Figure 4.2 shown below, images of tilted and rotated objects are taken in a cluttered environment because SURF algorithm is rotation in variant. In that case the objects are also detected. Images taken from the corner of the room or from the side are also detected by the SURF algorithm which shows that the algorithm is robust and detects the objects in all scenarios.

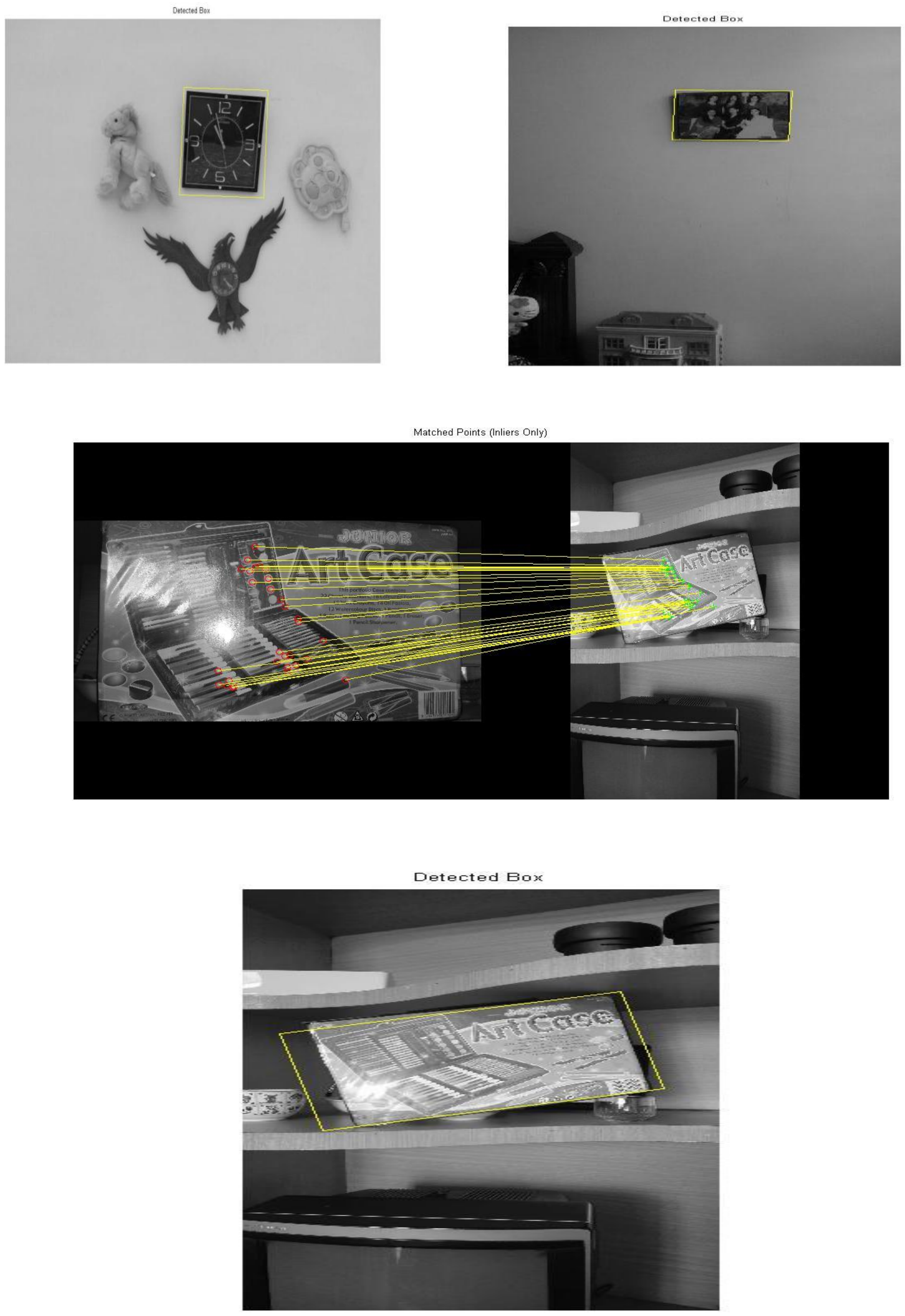
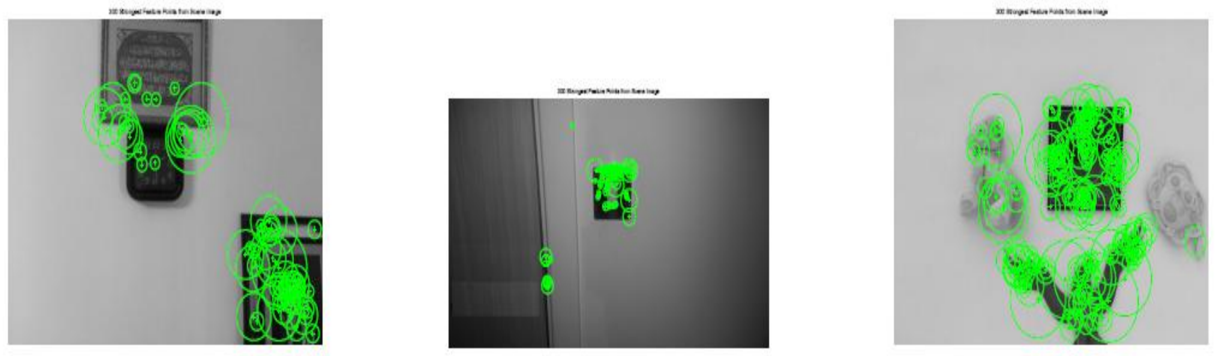


Figure 4.2: images of tilted and rotated objects.

4.2 Feature Extraction Results

Results presented in the figure 4.3 shows that the strong features points has successfully been chosen and these features points are represent by green circles.

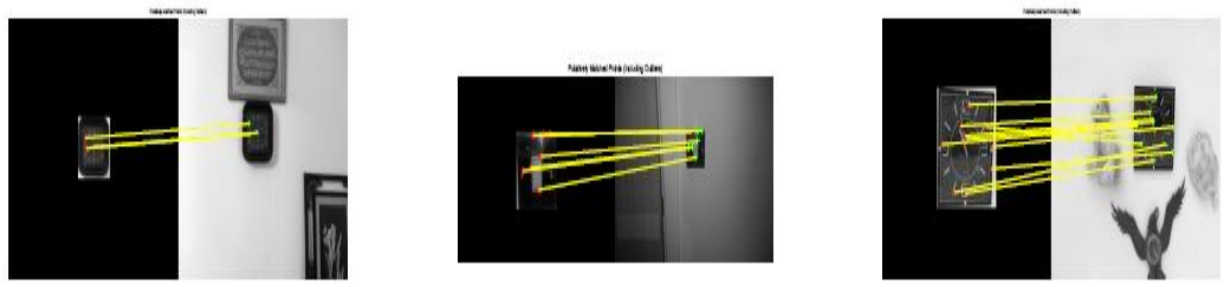


(a) Outdoor clock (b) Indoor switch board (c) Outdoor clock

Figure 4.3: Feature extraction

4.3 Features Matching Results

Results presented in figure 4.4 show that the proposed algorithm successfully matches a good number of features between the reference image I_{ref} and the target image I_{target}.



(a) Outdoor clock (b) Indoor switch board (c) Outdoor clock

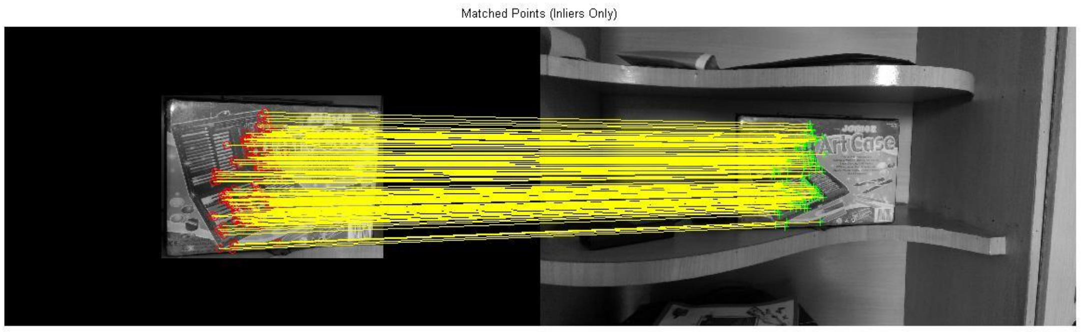
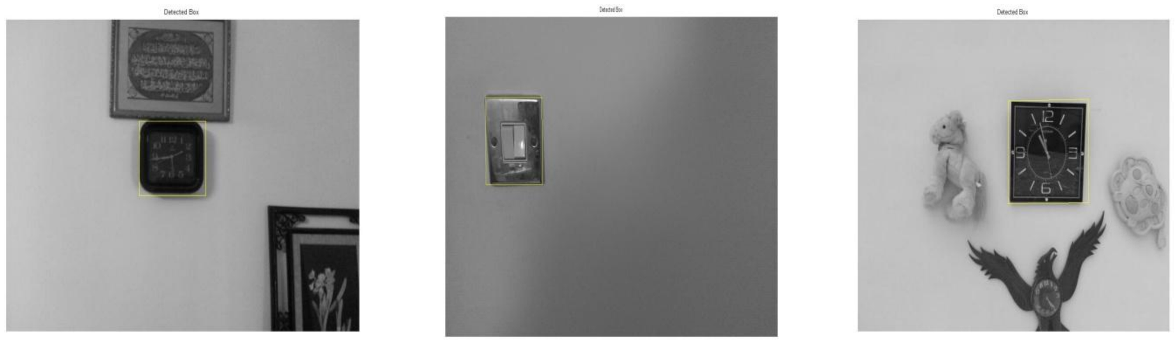


Figure 4.4: Feature matching

4.4 Object Detection and Estimating Area of Object

Once a match between the reference image and the target image is found, the next stage of the algorithm draws boundary of the detected object. These results are shown in figure 4.5. Yellow boxes around the detected objects show the successful detection of the required objects. The area of the object can now be easily calculated through the bounding box.



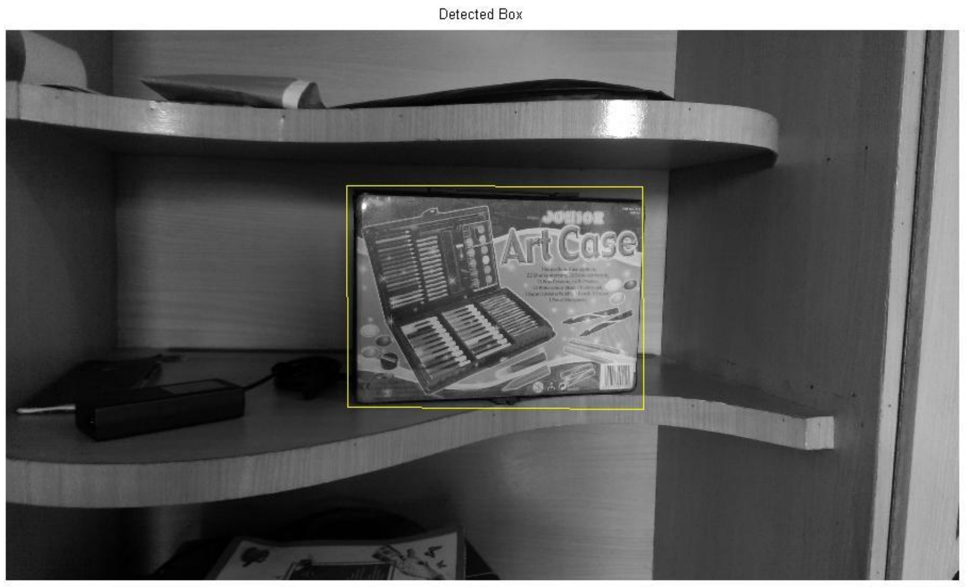


Figure 4.5: Few of the images used as a training data, captured in cluttered environment in different lighting conditions

4.5 Estimating Distancing Using Area

The final stage of the algorithm uses equations (2.3) and (2.4) to calculate the distance of the detected object from the camera. Distance of few of the objects e.g. clock, scenery, art box, whose images are taken from different distances are shown in Tables below.

Table 4-1: Estimated Covered Area vs Estimated Distance (For Clock)

Serial No	Estimated Covered Area (pixels)	Estimated Distance (Feet)	Actual Distance (Feet)	Accuracy (%)
1	235600	1.93	2	96.5
2	92772	3.13	3	87.4
3	74655	3.89	4	97.2

4	50476	4.84	5	96.8
5	39835	5.73	6	95.5
6	19004	6.68	7	95.4

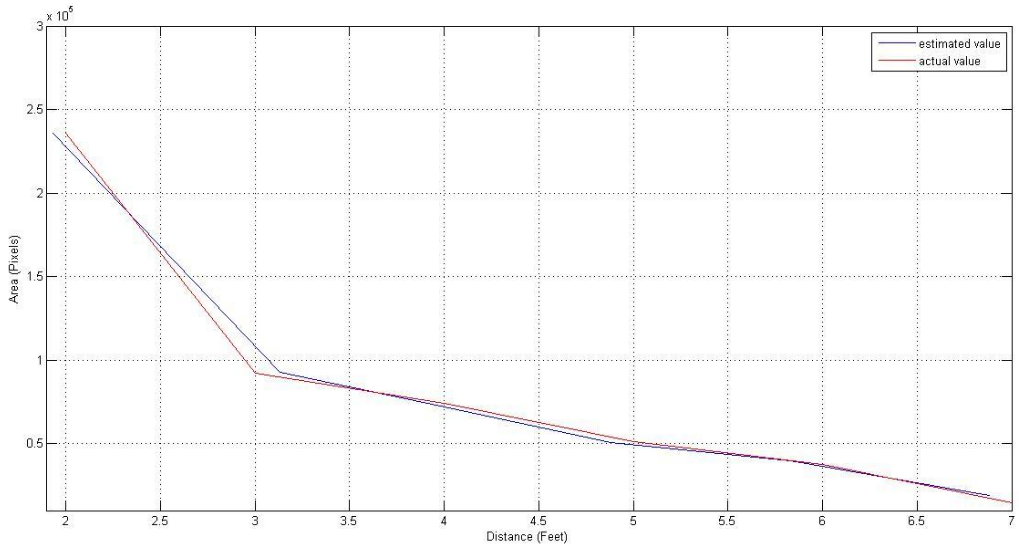


Figure 4.6: Plot for Comparison of Estimated Data and Actual Data for Clock

Table 4-2: Estimated Covered Area vs Estimated Distance (For Art Box)

Serial No	Estimated Covered Area (pixels)	Estimated Distance (Feet)	Actual Distance (Feet)	Accuracy (%)
1	920200	2.01	2	99.5
2	507086	3.14	3	95.3
3	338180	3.87	4	96.75
4	219760	4.82	5	96.4
5	174360	6.21	6	96.5

6	124860	7.29	7	95.8
7	99993	7.75	8	96.87
8	81051	8.74	9	97.11
9	66119	10.39	10	96.1
10	43815	11.42	11	96.18
11	39592	12.38	12	96.83
12	199708	7.81	13	60.07
13	131790	8.42	14	53.5
14	26994	15.40	15	97.33
15	23298	16.37	16	97.69
16	15065	17.44	17	97.41
17	11720	18.50	18	97.22

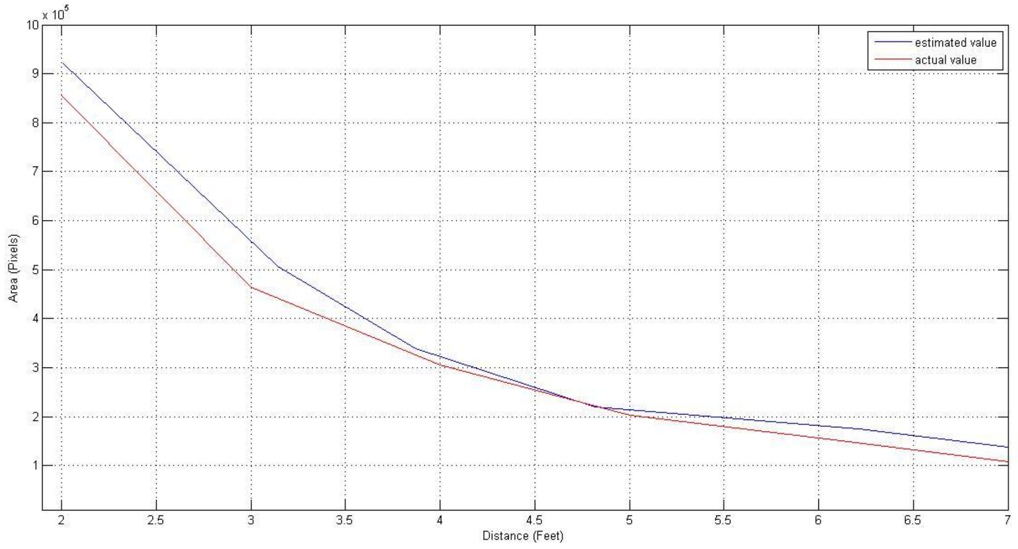


Figure 4.7: Plot for Comparison of Estimated Data and Actual Data for Art Box

Table 4-3: Estimated Covered Area vs Estimated Distance (For Scenery)

Serial No	Estimated Covered Area (pixels)	Estimated Distance (Feet)	Actual Distance (Feet)	Accuracy (%)
1	645855	2.06163	2	96.92
2	423200	3.14820	3	95.06
3	365460	4.1653	4	95.87
4	258860	5.2214	5	95.57
5	171046	6.3083	6	94.86

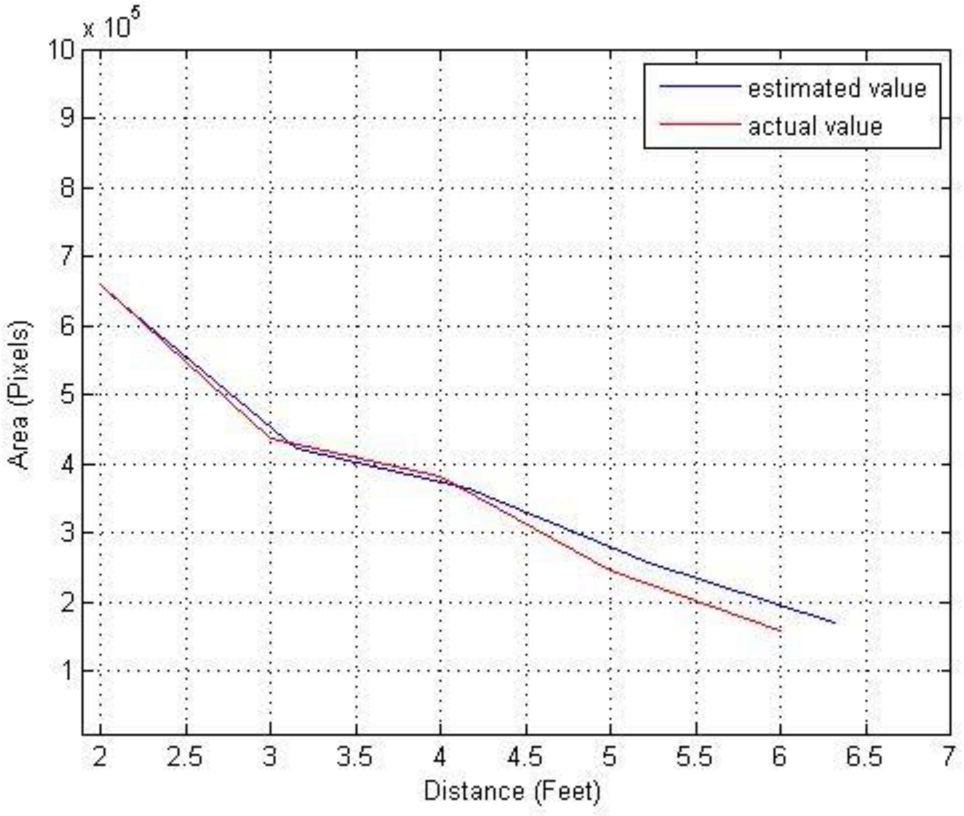


Figure 4.8: Plot for Comparison of Estimated Data and Actual Data for Scenery.

Figure 4.6, 4.7, 4.8 contains information of estimated data and actual data. It shows in all examples i.e. clock, art box, scenery, with the increase in distance, the area of the object decreases.

4.6Accuracy

Accuracy of this algorithm is between 80% to 90% with the dataset of 20 to 30 images, taken in different scenarios. It detects objects both in indoor and outdoor environment. Images taken from large distances in a cluttered environment and from the side of the room and from corner. Images taken at some angle are also detected by the SURF algorithm which shows robustness of the algorithm.

In Table: 4-2, it shows the accuracy of different objects in a real world environment. Objects e.g. switch board, clocks, frames, sceneries and art boxes etc are used as real world objects. By missed detections it shows for how many times it does not detects the object and by false positives it means for how many times it detects the wrong object.

Table 4-4 shows accuracy of different examples.

Serial No	**False positives (%)**	**Missed Detections (%)**	**Accuracy (%)**
Switch Board	10	5	85
Scenery	10	0	90
Photo Frame 1	5	5	90
Clock 1	5	0	95
Clock 2	3	6	91
Photo Frame 2	4	4	92
Art Box	2	4	94

The graph below in Figure 4.9 shows the plot of the error between the estimated distance of the object and the actual distance of the desired objects from the camera and frame number is the image taken from a specific distance. The error is plotted against the frame/image numbers.

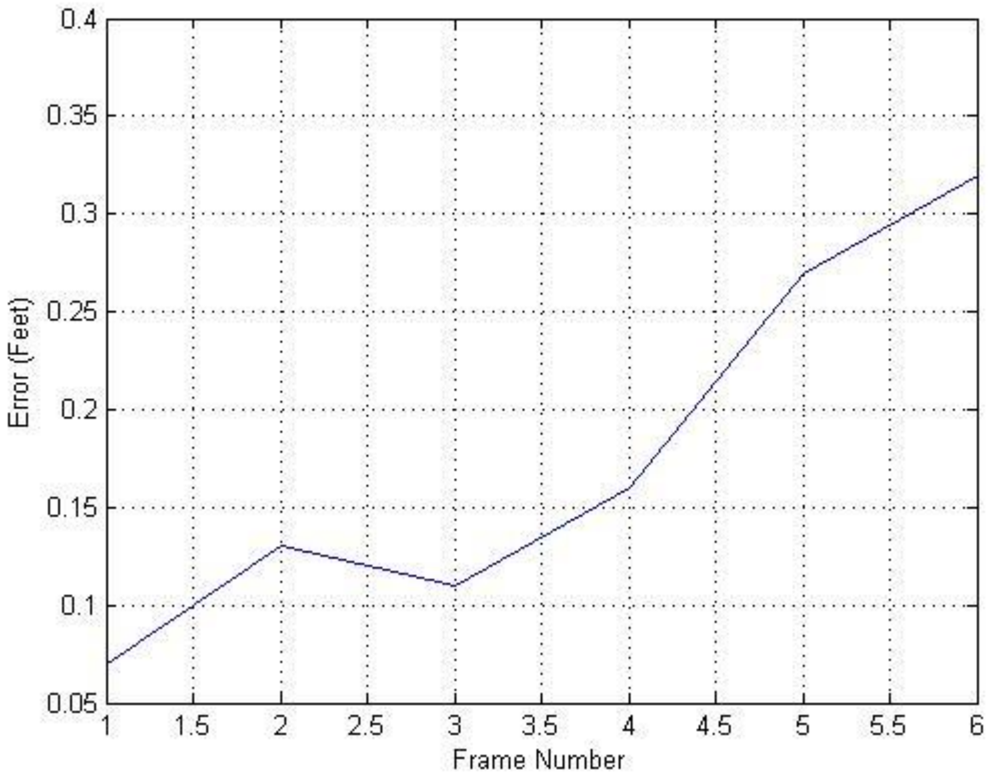


Figure 4.9: Plot of frame number versus error in feet.

4.7 Processing Time

On average the proposed algorithm processes image, detects required object and calculates its distance from camera in 3.45 seconds. This processing time is calculated for MATLAB implementation of the proposed algorithm. The algorithm was run on Intel core i7 3.00 GHz processor with other multiple applications running on the system. It shows that after down sampling of the image and by adding further preprocessing techniques and by implementation of the algorithm on C it is possible to achieve real time processing.

4.8 GUI Development

Graphical user interface is purely designed in the MATLAB [34]. One can easily access the GUI environment in the MATLAB. Guide initiates the GUI design environment tools that allow GUIs to be created or edited interactively.

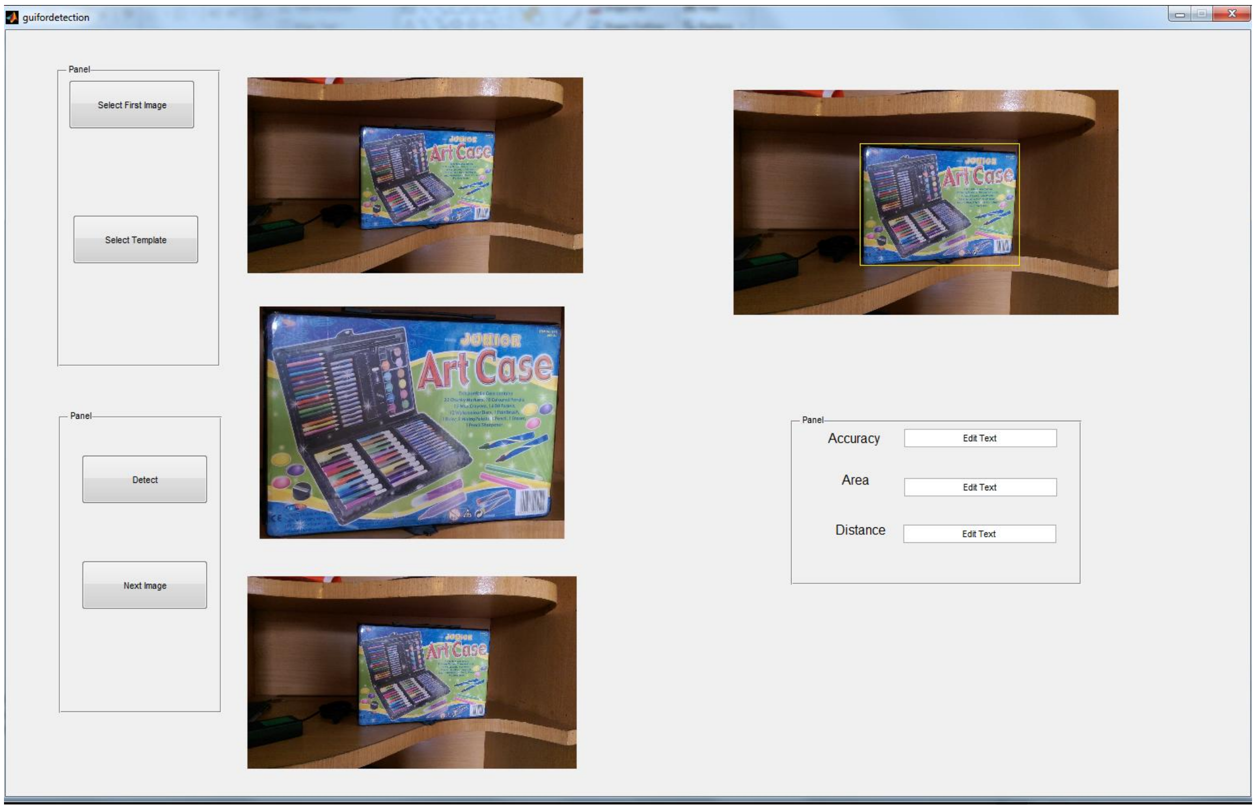


Figure 4.10: Graphical User Interface

In the Designed GUI figure 4.10, a user is required a template of the object to be detected and selects the first image of the sequence of images from which an object needs to be detected. And as an output the GUI displays the detected objects, accuracy, estimated area and distance of the detected object.

CHAPTER 5: CONCLUSION

Feature detection and extraction are well known techniques in computer vision and image processing. Already many algorithms have been developed in this field. Most of these algorithms are limited to certain frame of reference. Such as corner points, lines, binary features, boundary traces extraction or detection. Likewise, derived algorithm has its own novelty element. Algorithm has presented a robust solution for depth estimation. The proposed algorithm successfully detects the required objects in a clutter environment by successfully using the SURF tool. Results show the robustness of the algorithm even in the different lighting conditions. Successful evaluation of the distance of the detected objects from the camera is achieved through the power fit curve fitting technique, carefully captured training data and successful detection results. Robustness of the proposed algorithm is validated by the results.

The research shows that single camera depth estimation can be achieved using polynomial curve fitting approach. The approach is efficient and can effectively be applied to applications of indoor navigation or motion planning algorithms.

Future Work

To make the algorithm that it can detect everyday objects in real time using SURF. Secondly Algorithm should be capable of classifying dynamic obstacles.

REFERENCES

[1] S. Lee, J. Lee and J.Paik, “Simultaneous Object Tracking and Depth Estimation using Color Shifting property of a Multiple Color–filter Aperture Camera,” in Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, May 2011, PP. 1401-1404.

[2] A. Sabnis and L. Vachhani, “Single image based depth estimation for robotic applications,” in Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE, Sep 2011, pp.102-106.

[3] K. Sangjin. Eunsung, M.H. Hayes, and J.Paik, “Multifocusing and depth estimation using a color shift model-based computational camera,” Image Processing IEEE Transactions on, Vol. 21, no. 9, pp. 4152-4166, Sept 2012.

[4] M. Martinello and P. Favaro,”Depth estimation from a video sequence with moving and deformable objects,” in Image Processing (IPR 2012), IET Conference on, July 2012, pp. 1-6.

[5] T. Kuo, C. Hsieh, and Y.Lo, ”Depth map estimation from a single video sequence,” in Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on, June 2013, pp. 103-104.

[6] Z. Liu, J. Wang, P.E. Kee, and S. Sundaram, “Depth and normal vector identification of an unknown slope from a uav using a single camera,” in Control Conference (ASCC), 2013 9th Asian, June 2013, pp. 1-6.

[7] G. Bekey and J. Yuh, “The status of robotics” Robotics Automation Magazine, IEEE, vol. 15, no. 1, pp. 80-86, March 2008.

[8] A. Sabnis and L. Vachhani, “Single image based depth estimation for robotic applications,” in Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE, Sept 2011, pp. 102-106.

[9] KINECT DEPTH SENSOR EVALUATION FOR COMPUTER VISION APPLICATIONS Electrical and Computer Engineering Technical Report ECE-TR-6 M.R. Andersen, T. Jensen, P. Lisouski, A.K. Mortensen, M.K. Hansen, T. Gregersen and P. Ahrendt Aarhus University, Department of Engineering February 2012.

[10] SIFT – The Scale Invariant Feature Transform Distinctive image features from scale-invariant keypoints. David G. Lowe, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

[11] Alexandre Alahi, Raphael Ortiz, Pierre Vandergheynst “FREAK: Fast Retina Keypoint” (EPFL) Switzerland. 2012

[12] Cameron Schaeffer “A Comparison of Key point Descriptors in the context of Pedestrian Detection: FREAK vs. SURF vs. BRISK. Stanford University CS Department. 2012

[13] SURF : Speeded Up Robust Feature, Herbert Bay, Tinne Tuytelaars, and Luc Van Gool 2006.

[14] Kristen Grauman and Bastian Leibe “Local Features: Detection and Description” Excerpt chapter from Synthesis lecture draft: Visual Recognition.

[15] Gregory D. Hager “Perception and Sensing in Robotic Mobility and Manipulation” Laboratory for Computation, Sensing and Control Department of Computer Science Johns Hopkins University.

[16] Distance Measuring based on Stereoscopic Pictures, Jernej Mrovlje and Damir Vrancic. 9th Phd Workshop on Systems and Control, Young Generation Viewpoint, 1-3 Oct 2008, Izola, Slovenia.

[17] E.R Devies, Law’s Texture Energy in TEXTURE. In Machine Vision: Theory, Algorithms, Practicalities, 2nd Edition, Academic Press San Diego.

[18] “3-D Depth Reconstruction from a Single Still Image”, Ashutosh Saxena, Sung H. Chung, Andrew Y. NG, Computer Science Department, Stanford University, Stanford. 6th June 2007.

[19] Polynomial based depth estimation from a single image, Kaushik K. Tiwari.

[20] Undergraduate Physics Students’ Computing And Learning Environment (Physics Virtual Bookshelf), David M. Harrison, Department of Physics, University of Toronto

[21] Real Time Moving Obstacle Detection using Optical Flow Models, Christophe Braillon, Cedric Pradalier, James L. Crowley and Christian Laugier, Intelligent Vehicles Symposium 2006, June 13-15-2006, Tokyo Japan.

[22] Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions, Christos Katrakazas, Mohammad Quddus, Wen-Hua Chen and Lipika Deka, September 2015.

[23] Obstacle Detection and Navigation Planning for Autonomous Micro Aerial Vehicles, Matthias Nieuwenhuisen, David Droeschel, Marius Beul, and Sven Behnke , international conference on Unmanned Aircraft Systems (ICUAS), Orlando, USA, May 2014.

[24] Numerical Methods Lecture 5 - Curve Fitting Techniques, Computer Methods, Gurley.

[25] I. Coope, “Circle fitting by linear and nonlinear least squares “. Journal of Optimization Theory and Applications, vol. 76, no. 2, pp. 381-388, 1993.

[26] Numerical Methods of Curve Fitting, P.G. Guest, Philip George Guest, Cambridge University Press

[27] The Comparison Research of Non Linear Curve Fitting in Matlab and LabVIEW, Hao Wen, Beijing Changcheng Aeronaut. Meas. & Control Technol. Res. Inst., Aviation Ltd. Corp of China, Beijing, China, Jing Ma, Meiju Zhang, Guimei Ma, 24 – 27 June 2012

[28] M. U. Akhlaq, U. Izhar, and U. Shahbaz, “Depth estimation from a single camera image using power fit” in Robotics and Emerging Allied Technologies in Engineering (iCREATE), 2014 conference on , April 2014, pp. 221-227.

[29] The Image Processing Handbook, 2nd edition, Russ, Jhon C, Woods, Roger P.M.D

[30] Introductory Digital Image Processing: A Remote Sensing Perspective, Jensen, J.R., ISBN 0-13-205840-5

[31] Krishna Dharavath, Fazal Ahmed Talukdar and Rabul Hussain Laskar“Improving Face Recognition Rate with Image Preprocessing” Indian Journal of Science and Technology Vol 7(8) 1170-1175, August 2014.

[32] Isra’a Abdul-Ameer Abdul-Jabbar “Image Processing for Face Recognition Rate Enhancement” International Journal of Advanced Science and Technology vol. 64 (2014) pp. 1-10

[33] Sana Liaquat, Umar S. Khan, Ata-ur-Rehman “Object Detection and Depth Estimation of Real World Objects using Single Camera” in International Conference on Aerospace Science and Engineering (ICASE) 2015.

[34] Learning to Program with Matlab: Building GUI Tools, Craig S. Lent .

Appendix

Matlab Code

```
function varargout = guifordetection_new(varargin)
% GUIFORDETECTION_NEW MATLAB code for guifordetection_new.fig
%      GUIFORDETECTION_NEW, by itself, creates a new GUIFORDETECTION_NEW or
raises the existing
%      singleton*.
%
%      H = GUIFORDETECTION_NEW returns the handle to a new
GUIFORDETECTION_NEW or the handle to
%      the existing singleton*.
%
%      GUIFORDETECTION_NEW('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in GUIFORDETECTION_NEW.M with the given input
arguments.
%
%      GUIFORDETECTION_NEW('Property','Value',...) creates a new
GUIFORDETECTION_NEW or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guifordetection_new_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guifordetection_new_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guifordetection_new

% Last Modified by GUIDE v2.5 23-Mar-2016 05:50:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guifordetection_new_OpeningFcn, ...
                   'gui_OutputFcn',  @guifordetection_new_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before guifordetection_new is made visible.
function guifordetection_new_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guifordetection_new (see VARARGIN)

% Choose default command line output for guifordetection_new
handles.output = hObject;
cd('C:\Users\user\Desktop\final')
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guifordetection_new wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = guifordetection_new_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%clc;
%clear all;
%close all;
function[tform, boxPolygon,a,b,c] = detection_img(boxImagergb,sceneImagergb)
NofImage = 23;
sfram = 0;
% a =       248;
% b =     -0.469;
% c =     -0.2109;

       a =        1954;
       b =     -0.4676;
       c =      -1.447;

%for i = 1:NofImage
%tic;
%boxImage = imread(['C:\Users\uos\Desktop\shamma\images\template.jpg');
%I = im2bw(boxImage);
boxImage = rgb2gray(boxImagergb);
%boxImage = histeq(boxImage);
%boxImage = rgb2gray(boxImage);
%imshow(boxImage);
%figure, imshow(J)
```

```
%figure;
%imshow(boxImage);
%title('Image of a Box');

%Read the target image containing a cluttered scene.

%sceneImage = imread(['C:\Users\uos\Desktop\shamma\images\IMAG07\'
,sprintf(i+sfram),'.jpg']);
%sceneImagergb =
imread(['C:\Users\uos\Desktop\shamma\images\IMAG',sprintf('%0d',i+sfram),'.jp
g']);
sceneImage = rgb2gray(sceneImagergb);
%sceneImage_eq = histeq(sceneImage);
%figure;
%imshow(sceneImage);
%title('Image of a Cluttered Scene');
%figure;
%imshow(sceneImage_eq);

%Detect feature points in both images.

boxPoints = detectSURFFeatures(boxImage);
scenePoints = detectSURFFeatures(sceneImage);

%Visualize the strongest feature points found in the reference image.

%figure;
%imshow(boxImage);
%title('100 Strongest Feature Points from Box Image');
%hold on;
%plot(selectStrongest(boxPoints, 100));


%Visualize the strongest feature points found in the target image.

%figure;
%imshow(sceneImage);
%title('300 Strongest Feature Points from Scene Image');
%hold on;
%plot(selectStrongest(scenePoints, 300));


%Extract feature descriptors at the interest points in both images.

[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);

%Step 4: Find Putative Point Matches
%Match the features using their descriptors.

boxPairs = matchFeatures(boxFeatures, sceneFeatures);

%Display putatively matched features.
```

```
matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
%figure;
%showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
 %   matchedScenePoints, 'montage');
%title('Putatively Matched Points (Including Outliers)');

%Step 5: Locate the Object in the Scene Using Putative Matches

%estimateGeometricTransform calculates the transformation relating the
matched points, while eliminating outliers. This transformation allows us to
localize the object in the scene.

[tform, inlierBoxPoints, inlierScenePoints] = ...
    estimateGeometricTransform(matchedBoxPoints, matchedScenePoints,
'affine');
%Display the matching point pairs with the outliers removed

%figure;
%showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
 %   inlierScenePoints, 'montage');
%title('Matched Points (Inliers Only)');

%Get the bounding polygon of the reference image.

boxPolygon = [1, 1;...                                  % top-left
        size(boxImage, 2), 1;...                        % top-right
        size(boxImage, 2), size(boxImage, 1);... % bottom-right
        1, size(boxImage, 1);...                        % bottom-left
        1, 1];                    % top-left again to close the polygon

%Transform the polygon into the coordinate system of the target image. The
transformed polygon indicates the location of the object in the scene.

newBoxPolygon = transformPointsForward(tform, boxPolygon);

%Display the detected object.
%
% figure;
% imshow(sceneImagergb);
% hold on;
% line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
% title('Detected Box');
% xd = newBoxPolygon(2, 1)-newBoxPolygon(1, 1);
% yd = newBoxPolygon(4, 2)-newBoxPolygon(1, 2);
% Area(i) = xd*yd;
% Estd_Dist(i)=a*Area(i)^b+c;
%toc;
%end
```

Code for Camera Calibration Method

```
clc;
clear all;
close all;
%function[tform, boxPolygon,a,b,c] = detection_img(boxImagergb,sceneImagergb)
NofImage = 16;
sfram = 0;
% a =       248;
% b =     -0.469;
% c =     -0.2109;
       a =        1954;
       b =     -0.4676;
       c =      -1.447;
       dd_ground_truth = 2:18;
for i = 1:NofImage
%tic;
boxImagergb = imread('D:\formulaa\exammple6\template.jpg');
%I = im2bw(boxImage);
boxImage = rgb2gray(boxImagergb);
%boxImage = histeq(boxImage);
%boxImage = rgb2gray(boxImage);
%imshow(boxImage);
%figure, imshow(J)

%figure;
%imshow(boxImage);
%title('Image of a Box');

%Read the target image containing a cluttered scene.

%sceneImage = imread(['C:\Users\uos\Desktop\shamma\images\IMAG07\'
,sprintf(i+sfram),'.jpg']);
sceneImagergb =
imread(['D:\formulaa\exammple6\IMAG',sprintf('%0d',i+sfram),'.jpg']);
sceneImage = rgb2gray(sceneImagergb);
%sceneImage_eq = histeq(sceneImage);
%figure;
%imshow(sceneImage);
%title('Image of a Cluttered Scene');
%figure;
%imshow(sceneImage_eq);

%Detect feature points in both images.

boxPoints = detectSURFFeatures(boxImage);
scenePoints = detectSURFFeatures(sceneImage);

%Visualize the strongest feature points found in the reference image.

%figure;
%imshow(boxImage);
%title('100 Strongest Feature Points from Box Image');
%hold on;
%plot(selectStrongest(boxPoints, 100));
```

```
%Visualize the strongest feature points found in the target image.

%figure;
%imshow(sceneImage);
%title('300 Strongest Feature Points from Scene Image');
%hold on;
%plot(selectStrongest(scenePoints, 300));


%Extract feature descriptors at the interest points in both images.

[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);

%Step 4: Find Putative Point Matches
%Match the features using their descriptors.

boxPairs = matchFeatures(boxFeatures, sceneFeatures);

%Display putatively matched features.

matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
%figure;
%showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
 %   matchedScenePoints, 'montage');
%title('Putatively Matched Points (Including Outliers)');

%Step 5: Locate the Object in the Scene Using Putative Matches

%estimateGeometricTransform calculates the transformation relating the
matched points, while eliminating outliers. This transformation allows us to
localize the object in the scene.

[tform, inlierBoxPoints, inlierScenePoints] = ...
    estimateGeometricTransform(matchedBoxPoints, matchedScenePoints,
'affine');
%Display the matching point pairs with the outliers removed

%figure;
%showM0atchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
 %   inlierScenePoints, 'montage');
%title('Matched Points (Inliers Only)');

%Get the bounding polygon of the reference image.

boxPolygon = [1, 1;...                           % top-left
        size(boxImage, 2), 1;...                 % top-right
        size(boxImage, 2), size(boxImage, 1);... % bottom-right
        1, size(boxImage, 1);...                 % bottom-left
        1, 1];                   % top-left again to close the polygon
```

```
%Transform the polygon into the coordinate system of the target image. The
transformed polygon indicates the location of the object in the scene.

newBoxPolygon = transformPointsForward(tform, boxPolygon);

%Display the detected object.
%
% figure;
% imshow(sceneImagergb);
% hold on;
% line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
% title('Detected Box');
 xd = newBoxPolygon(2, 1)-newBoxPolygon(1, 1);
 yd = newBoxPolygon(4, 2)-newBoxPolygon(1, 2);
 Area(i) = xd*yd;
 distHN(i) = distanceHN(xd,yd)*3.28;
 Estd_Dist(i)=a*Area(i)^b+c;
 dd_diff_est(i) = dd_ground_truth(i) - Estd_Dist(i);
 dd_diff_HN(i) = dd_ground_truth(i) - distHN(i);
%toc;
end



% clc;
%
% clear all
%
% close all

function[dd] = distanceHN(xd,yd)

%Camera, Did you use back camera or front camera
H_angle = 88.6103*1.10;
V_angle =  57.7908*1.1;


tdh = tand(H_angle/2);
tdv = tand(V_angle/2);



N_pc = 2688;
N_pr = 1520;



%Template
W_m = 0.33;%meters, your values are wrong, the template has more width than
height. You are giving W_m = 0.035 and H_m = 0.22 awfully wrong
H_m = 0.22;



%Experiment
%pixels_detected = [1080    807;828 586;652 475;534 387];
```

```
%dd_ground_truth = [1 1.5 2 2.5]./3.28;


%[rr cc] = size(pixels_detected);


%for jj = 1:rr

    Nd_pc = xd;

    Nd_pr = yd;


    area_detected = Nd_pc*Nd_pr;


    d(1) = W_m*N_pc/(Nd_pc*2*tdh);
    d(2) = H_m*N_pr/(Nd_pr*2*tdv);

    dd = mean(d);


    %dd2(jj,1) = sqrt((1/(area_detected))*W_m*H_m*N_pc*N_pr*1/4*1/(tdh*tdv));


%end


%dd_diff = dd_ground_truth' - dd;%or use dd2


%rms_error = sqrt(sum(dd_diff.^2)/4);
```

Completion Certificate

It is certified that the thesis titled **“Real World Object Detection as Markers for Autonomous Aerial Motion Planning Application Using Single Camera”** submitted by registration no. NUST201261248MCEME35512F, NS Sana Liaquat of MS-74 Mechatronics Engineering is completed in all respects as per the requirements of MainOffice,NUST (Exam branch).

Supervisor: _____________

DR UMAR SHAHBAZ KHAN

Date: ____ June, 2016