Visualization of 3D Object on Planar Screen Using View Angle

Zuhir Badr A. Badr

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
July 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

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

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

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

Asst. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Mehmet Bodur
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

The aim of this thesis is to develop and demonstrate a practical method to support 3D perception of stationary objects in a virtual space through the motion of a two dimensional projection image. The structure of a human eye is naturally equipped by some tools to perceive the depth from several hints such as the size of image compared to the its expected size, and the sharpness of the image at different focal lengths of the lens, the parallax difference in the images from the left and right eyes, and, if the image moves, by comparing the images at different view angles.

In this thesis, the movement of the observer is detected by a software using the video camera frames, and the expected 2D projection of the virtual objects is transformed for the detected position of the observer to support a depth feeling of the observer. The developed program is coded in MATLAB, to determine the position of a red marker that is attached to the head of the observer, to compose the transformation matrix that converts 3D corner points of the virtual objects to expected perspective projection for the determined view-angle, and to draw the projection on the screen for the observation. The code is written in a flexible form to work with any PC with a web-cam, and graphical screen. The implemented system is tested successfully comparing the views of a set of virtual geometric objects on a platform with respect to the view of similar objects physically on a test platform.

Keywords: Depth perception, Colour detection and tracking, 3D-visualization.

ÖZ

Bu tezin amacı sanal uzaydaki duran nesnelerin 3D algısını iki boyutlu izdüşümlerindeki hareket aracılığıyla destekleyen bir yöntem geliştirmek ve göstermektir. Insan gözü doğal olarak görüntünün büyüklüğüyle beklenen büyüklüğünü karşılaştırmak, görüntünün değişik odak derinliklerindeki keskinlik ve bulanıklığı, sağ ve sol göz görüntülerindeki fark, ve görüntü hareket ederse değişik gözlem açılarından görünüşünü analiz gibi derinlik algılamaya elverişli bir takım araçlarla donatılmıştır.

Bu tezde, gözlemcinin hareketleri bir yazılım sayesinde bir video kameranın yolladığı çerçevelerden algılanarak sanal nesnelerin belirlenen gözlemci yerine karşılık beklenen 2D izdüşümlerine dönüştürülerek, bu yolla, gözlemcinin nesneler hakkında bir derinlik duygusu oluşturulması sağlanmaktadır. MATLAB'da kodlanmak üzere geliştirilen program gözlemcinin başına iliştirilmiş kırmızı bir işaretin yerini belirlemekte, ve gözlemcinin bakış açılarını tayin ederek sanal nesnelerin 3D köşe noktalarının perspektif izdüşümü için gereken dönüştürme matrisini hesaplayıp ekrana 2D izdüşümünü çizmektedir. Kod, video kamera ve grafik ekran donanımlı herhangi bir PC de çalışacak esneklikte yazılmıştır. Uygulanan sistem sanal geometrik nesnelerin görünümlerini benzer nesnelerin fiziksel bir test platformundaki görüntüsüyle karşılaştırılarak başarıyla sınanmıştır.

Anahtar kelimeler: Derinlik algısı, Renk tespit ve izleme, Üç boyutlu-görüntüleme,

DEDICATION

Dedicated to My Family

ACKNOWLEDGMENT

First, I would like to thank ALLAH then my Mother, Brothers and sisters. And I express my gratitude for every one whom supported me during my education period.

Next I want to say many thanks to Asst. Prof. Dr. Mehmet Bodur, Prof. Dr. Majid Hashemipour, Asst. Prof. Dr. Adnan Acan, Husseyin Yetiner and Basma Al Gembry for their support and encouragement during my graduate studies.

I also would like to share my happiness with all my relatives and friends. Last but not least I dedicate my success to my late father who has been my constant and source of inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two Dimensional Image
3D	Three Dimensional Image
HSV	Hue-Saturation-Value
RGB	Red Green Blue
LB	Labelled Binary Image
S	Structuring Element
B	Binary Image
T	Threshold Value

Chapter 1

INTRODUCTION

1.1 3D Perception

With an aim to accomplish a 3D visual perception of an observer using a graphical 2D computer screen, this thesis starts with a short introduction on the perception of depth in a human visual system.

Human visual system (HVS) is equipped with several tools and methods to develop a cue for the perception of depth. In literature the following monocular cues of depth are commonly listed to contribute in decision of depth perception: i) texture, shading, and perspective properties are called pictorial depth cues ii) size constancy, iii) physiological cues of monocular eye structure such as sharpness at focus and blur at non-focused distances, iii) monocular movement cues also called parallax or kinetic depth effect [1].

HVS constructs a 3D mapping of the outside world mainly using the 2D images projected on the retinal surface of the left and right eyes based on binocular and monocular cues. It uses cues of depth to determine the depth feeling additional to the 2D retinal projection. The binocular cues are obtained from left and right eyes. The left and right eyes of a human visual system get the image of the same object in slightly different angles, which provides sufficient data to percept the depth of edges and corners on the object. This feature of the human visual system is known as the stereopsis, or stereo-vision. Perceiving depth by stereo-vision is called stereo effect. Along with the stereo vision, there is another component of depth perception, called the optical vergence. The binocular vergence depends on the angular displacement of the eyes to see the object at the centre of the retinal region. The diversion angle from a parallel position is called vergence angle, L. L=0 is obtained at an infinite distance, and larger L values correspond that the object is nearer [1].

The third major cue of depth is the kinetic depth effect. A rotational motion of the retinal image of a stationary object provides cues to HVS on the depth of the moving points. Together with stereopsis and vergence, the kinetic depth effect provides major information on the reconstruction of 3D mapping in HVS [1].

Among these three natural depth perception tools of HVS, the stereopsis method is commonly used in the 3D picture, 3D movie and the 3D media industries by using special eyeglasses that shows different pictures to left and right eyes. However, using these eyeglasses is not comfortable, and even may cause health problems for the eyes when they should be used for long hours. The binocular vergence method is not practical to be applied to available computer screens since the screen stays always at a constant, mostly about 50 cm distance. The only remaining depth cue is the kinetic depth cue, which requires rebuilding the image as a function of the relative rotational movement of the screen consistent to its perspective projection [1].

Literature points that, the kinetic depth effect and the binocular depth effect are mainly expected to be evaluated by a different mechanism, and a depth effect may be created by any of these cues [1]. This statement encourages the idea of developing a monocular depth perception cue method without using a special optical apparatus in between the screen and the eyes.

1.2 Object Detection and Tracking

The implementation of a system for a kinetic depth perception requires detection of the movement of the eyes of the observer. For this purpose, an image processing method is necessary to process the images which are captured from a video camera that is placed on the graphical screen. A real time object detection seeks the image of a target in the captured video frames. Along with the location of the target object in a single or a sequence of images, it also determines the changes in the size and position of the object. A special colour for the target provides easy and accurate detection of the target object by colour detection, and widely used in various applications nowadays, yet, it provides still an open research area to improve many parameters such as the accuracy and the speed of detection algorithms [3].

1.3 Description of Problem

The aim of this thesis is to provide the necessary depth cues by tracking the observers view angle, and drawing the 2D projections of the virtual objects accordingly. A literature survey on the kinetic depth perception indicates that binocular and monocular depth cues are processed separately, and combined in the Human Visual System to a 3D mapping of the observed objects without domination of one on the other. Consequently, developing kinetic depth perception by a software shall successfully create a 3D effect on the observers HVS whenever the observer changes the view angle of the screen. Along with the developed 3D perception, the proposed system may be successfully used for the tasks where an observer shall inspect the hidden parts of virtual 3D objects by looking them from different view angles.

1.4 Methodology

The methodology applied in this thesis is summarized by the following six items.

1. The eyes of the observer are assumed to be monocular for practical purposes. The literature states that the source of a depth cue has no effect on the depth perception when building the 3D mapping of the objects. Thus, we expect that the missing stereo optic depth cue shall not inhibit 3D mapping of HVS.

2. The movement of the object for a kinetic depth perception is necessary. This movement may be satisfied by the rotation of the objects even when the observer is stationary. But such a system may not be useful to work on a particular region of the objects. Another approach may be to determine the movements of the observer, and update the screen image for the new view angle of the observer. Most of the modern PC systems are equipped with a video camera system which is definitely suitable for this purpose.

3. The test of the depth perception requires a set of simple geometric objects. We prefer white objects to prevent extra depth cues other than kinetic depth cue by simplicity of the objects with plain white surfaces because any surface texture may develop texture and parallax based depth cues along with the kinetic depth cue. For simplicity of the implementation, the objects are described using the homogenous coordinate system in their own coordinate frames.

4. A red light on the cap of the observer provides the red target that is easily located and tracked at each captured frame of the video stream. The captured image is evaluated to get the red region in the image. The location of the region is converted to the view angle of the observer.

5. The view angle of the observer provides sufficient information about how many degrees of rotation about x and y-axes are necessary to update the 2D projection of objects. A translation of the 2D projection provides the perception of the placement of the objects on the virtual workspace.

6. A scene with multiple geometric blocks is composed physically similar to a 3D computer screen to compare the view of the virtual geometric blocks on the virtual workspace to avoid any mistakes in the evaluation of transformations corresponding to the view angle. The final test is carried comparing the depth feeling of the physical scene to the virtual one.

Comparing the physical kinematic depth perception against the virtual one may appear unfair because the vision of the physical scene provides a full set of depth cues including shadows and light conditions on the surfaces. However, if kinetic depth cue together with perspective appearance is sufficient for depth perception, the HVS of an observer may construct a 3D world in her or his mind watching the 2D drawing on the screen.

1.5 Organization of the Thesis

Chapter 1 introduced the perception of depth in a Human Vision System and proposed a novel method to build the kinetic depth perception on a computer screen observer. It also explained the methodology to composing and to test a kinetic depth perception mechanism using a typical personal computer screen and a video camera. The remaining chapters are organized in the following manner:

Chapter 2 discusses the colour detection and tracking process to locate the observer view angle starting with capturing an image from the video stream. It gives the details of image processing applied on the captured image to get the position and size of the red region attached on the hat of the observer.

Chapter 3 explains the process to convert the view angles to the homogenous transformation that calculates the view of the objects with the rotation of the view angles. It also provides a perspective scaling that provides extra depth perception even when the observer does not move.

Chapter 4 gives the details of implementation, testing and results of the tests for the depth perception by motion. Finally, Chapter 5 contains a discussion and conclusion about the implemented system.

In addition to the main text, this thesis contains the source code of the implemented program written in MATLAB.

Chapter 2

COLOR DETECTION AND TRACKING PROCESS

2.1 Introduction

This chapter explains the image capture and image processing sections of the developed system that supplies the location of the red marker attached on the head of the observer to determine the view angle of the observer. Further chapters give the details of the representation of the virtual objects, coordinate transformation of the object position and orientation to the 2D projection on the PC screen.

The proposed system requires capturing images from a video camera to process it for colour detection on real time. Figure 1 shows the overall block diagram of processes explained in this chapter.

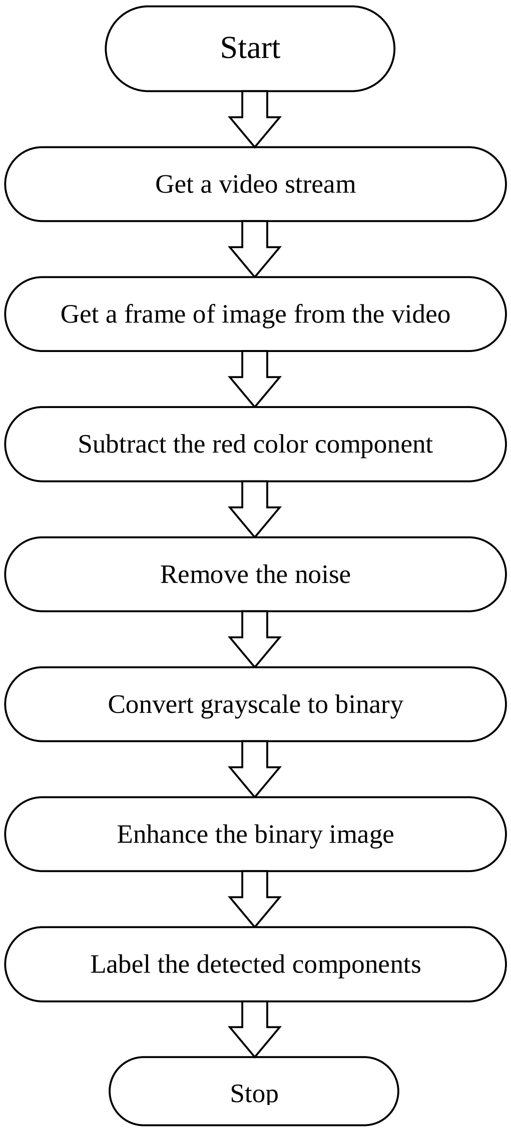

Start
Get a video stream
Get a frame of image from the video
Subtract the red color component
Remove the noise
Convert grayscale to binary
Enhance the binary image
Label the detected components
Stop

Figure 1. Block Diagram of Colour Detection and Segmentation Process.

2.2 Image Acquisition Objects

The video stream of a web camera attached on a personal computer is accessed through the image acquisition object created in Matlab Image Acquisition toolbox. The Matlab has a large library of video devices, and a function is available to detect the adaptor name, device ID, and the available video formats of the installed video camera device.

Matlab provides two image acquisition objects which are called video input object and video source object as seen in Figure 2 [4].

Video input object refers to the connection between the software 'MATLAB' and the hardware will be used, and that works as a container of the video source objects.

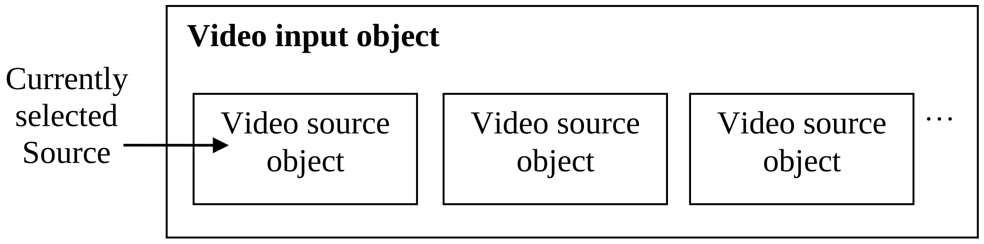


Figure 2. Video Input Object (From the Mathworks, by Permission)

Video source object refers to the number of video resources created by the acquisition toolbox. A video source might provide more than one data source, depending on the format of the source, but it is considered as a single source. The format is specified at the creation of video input objects by the sample code shown below [4].

```
Vid = videoinput ('winvideo', 1,'YUY2_640x480');

Set (vid, 'FramesPerTrigger', INF);

Set (vid, 'ReturnedColorspace', 'rgb')

vid.FrameGrabInterval = 1;
```

By typing '`vid`' in MATLAB command window, it is possible to display the format of the data input as seen in Figure 3.

```
>> vid

Summary of Video Input Object Using 'STARTEC 1.3MP
Webcam'.
Acquisition Source(s):  input1 is available.
Acquisition Parameters:  'input1' is the current selected
source.
  Continuous acquisition using the selected source.
     'YUY2_640x480' video data to be logged upon START.
     Grabbing first of every 5 frame(s).
     Log data to 'memory' on trigger.
Trigger Parameters:  1 'immediate' trigger(s) on START.
Status:  Waiting for START.
0 frames acquired since starting.
0 frames available for GETDATA.
```

Figure 3. Summary of Video Input Objects.

2.3 Acquiring the Frames

At the colour tracking and detection step of the image processing, we are required to have a single frame includes the data (Red Colour) that to be processed in further steps. The actual frames acquired from the camera (device) have been performed by the acquisition thread function Figure 4. The acquisition starts with `getsnapshot` function [5]. It keeps acquiring frames until it reaches the specified number at the acquisition call.

`Data = getsnapshot (vid);` where `'vid'` specifies information related to the device.

Figure 4. Acquired Frame.

The acquisition thread function contains two type of loops which are *thread message loop,* and *frame acquisition loop*.

Thread Message Loop is simply defined as main process loop of the acquisition thread function where the thread created by `OpenDevice` function followed by entering the thread message loop, and waiting the frame to be acquired where `StartCapture` function giving it right to start acquiring the frames as seen in Figure 5 [5].

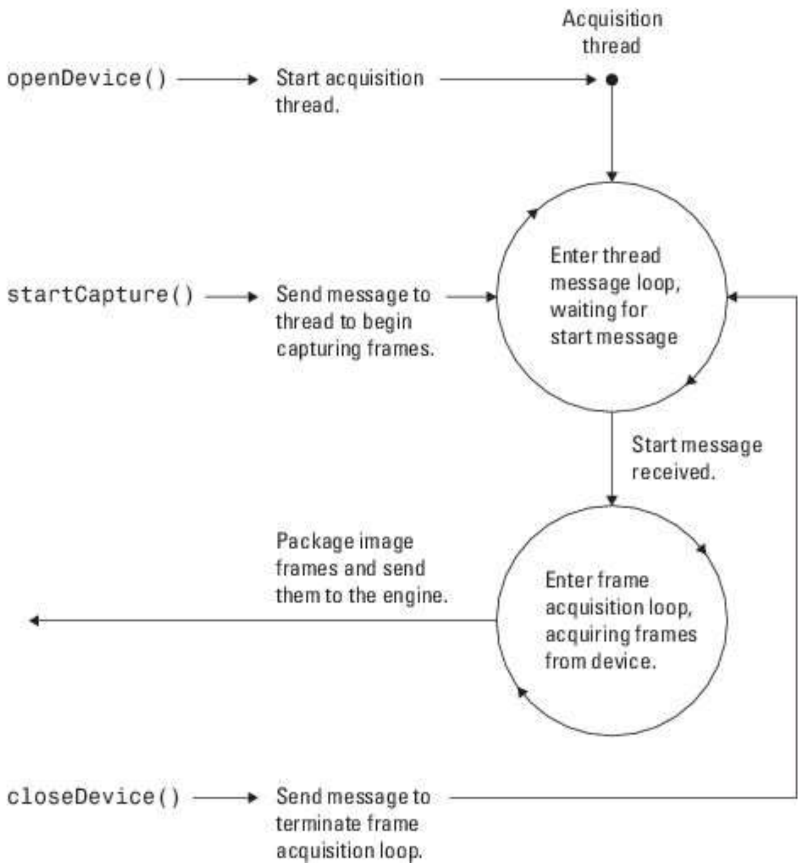

Acquisition thread
openDevice()
Start acquisition thread.
startCapture()
Send message to thread to begin capturing frames.
Enter thread message loop, waiting for start message
Start message received.
Package image frames and send them to the engine.
Enter frame acquisition loop, acquiring frames from device.
closeDevice()
Send message to terminate frame acquisition loop.

Figure 5. The Relation between the Adaptor Functions and the Acquisition Thread (From the Mathworks, by Permission).

Frame Acquisition Loop can be defined as link station between engine and *Thread Message Loop* where it receives the frames and sends them to the engine. It is responsible for all frames creating operations including status of the acquisition. It checks the number of frames that already specified has been acquired or not. And, it collects the frames from the device. It also configures status of the hardware trigger and controls the frame acquisition loop. In case of it needs to send frame to the engine, it works to create the frame object, filling the frame object by the acquired images, and logging the time of the acquisition [5].

2.4 Extracting Red Colour Component as a Grey-Scale Image

Processing a grayscale image is much faster than processing a colour image because each pixel of a colour image requires three times more information and accessing each one takes a considerable part of processing time. In the colour detection process, the red colour image is converted into a grey-scale image to reduce the information and speed up the processing [8].

The colour image can be easily converted to grey-scale image in MATLAB using `rgb2gray` (`ColorImage`) function which is based on the following transformation algorithm 1 [8].

$$I_{gray-scale}(n,m) = \alpha I_{colour}(n,m,r) + \beta I_{colour}(n,m,g) + \gamma I_{colour}(n,m,b). \quad (1)$$

Where (*n*, *m*) refers to output pixels at grey-scale and (*n*, *m*, [*r*, *g* or *b*]) refers to the channel of the pixel's colour. *r*, *g* and *b* indexed to red, green and blue respectively [6]. Since the red or any other colour component depends on illumination, they are normalized in between 0 and 1 by subtracting the luminance of grey-scale image from the red component as seen negative in Figure 7.

```
diff_im = imsubtract (data (:,:, 1), rgb2gray (data));
```

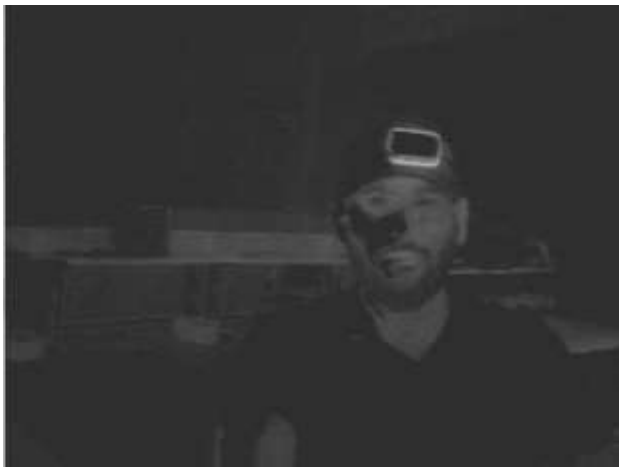


Figure 6. Grey-Scale of the Normalized Red Component.

Figure 6. Negative Grey-Scale of the Normalized Red Component.

2.5 Filtering Out Noise in the Image by Median Filter

This step of colour detection targets reducing the noise of the raw image by a process, which is called ‘Median Filter’. It is invoked by ‘`medfilt2`’ function in MATLAB. The Median Filter is a nonlinear statistical filter and considered as most common used filter. Median filter sorts the grey values of a *n.m* neighbourhood in natural numerical order, and sets value of the centre-pixel by mid value of the sorted list of all pixels in the neighbourhood (i.e, $(n.m+1)/2^{th}$ item in sorted list). The mask is typically a square of odd numbers like 3x3 or 5x5 to balance the upper and lower part of the list, and for example, for 3x3 median filter, the 5th of sorted 9 neighbour elements becomes the new centre-pixel value as seen in Figure 8 [7][8].

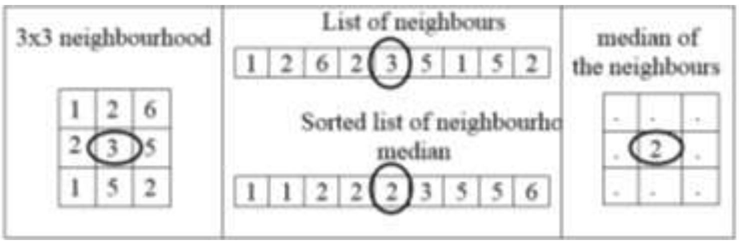


Figure 7. Median Filter Mechanism.

The output of Median Filter is denoted by:

$g(x, y) = med\{f(x-i, y-i), i, j \in W\}.$

Where $f(x, y)$ is an input binary image function and $g(x, y)$ is an output binary image function [7]. The ‘`medfilt2`’ MATLAB function is able to specify the number of neighbourhood (mask) parameter. In this thesis, [3x3] neighbourhood is used as shown in Figure 8. The filtered image is shown in Figure 9.

```
diff_im = medfilt2 (diff_im, [3 3]);
```

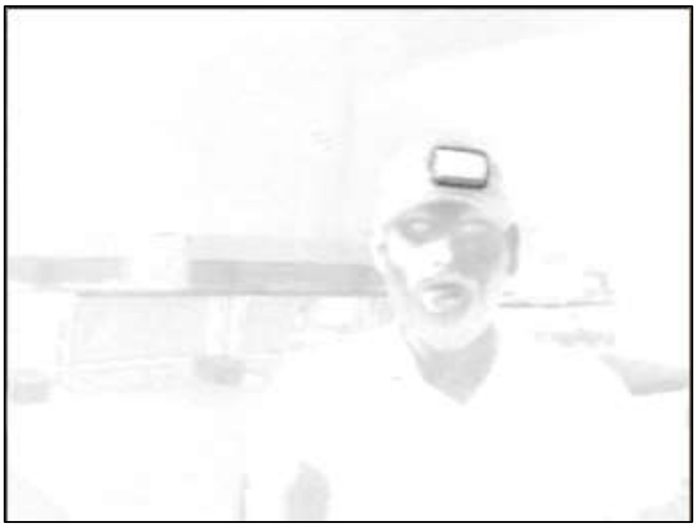


Figure 8. The Negative Normalized Red Component after Median Filter.

2.6 Colour Segmentation

After removing the noise of the image with a Median Filter, enhanced image is ready for further operations. The next operation is to strip off all unnecessary colours, objects and areas from the image. The image is transformed to a binary image by using “im2bw” MATLAB function, before the location of the red mark is determined.

2.6.1 Segmentation

Segmentation is a partitioning operation on a binary digital image to group the neighbouring pixels of the same kind as the sets of pixels. It works on a greyscale image and converts the image to a simpler, mostly binary image [12]. There are two types of segmentation: Complete segmentation that refers to the objects corresponding to original image objects. Partial segmentation refers to the objects which are not corresponding to original image objects. Segmentation may be obtained by different methods such as edge-based methods, and region-based methods, as well as global approaches.

2.6.2 Thresholding as a Segmentation

The simplest method for segmentation is called Thresholding segmentation. It is based on a threshold value. Thresholding segmentation converts the grey-scale image into a binary image g(x, y) which is 0 or 1, using algorithm 2.

$$g(x,y) = \begin{cases} 1 & if \quad f(x,y) \geq T \\ 0 & otherwise \end{cases} \tag{2}$$

The binary values are assigned depending on the value of the threshold. If the pixels' value of the grey-scale image is greater than the threshold the pixel will be assigned to 1 (White), otherwise will be assigned to 0 (Black) the following algorithm shows Thresholding algorithm [8].

```
For each pixel of I (i, j) within the image I
    If I (i, j) > threshold
        I (i, j) = 1
    Else
        I (i, j) = 0
    End if
End for
```

The Thresholding segmentation has two type of algorithms, global algorithm which uses only one threshold for all the image pixels which are used in our research through ‘im2bw’ MATLAB function. An adaptive algorithm which uses a variable number of thresholds for all the image that used to segment different colours from the same image. The key of Thresholding segmentation operation is finding the threshold value of each colour. Figure 10 represents the segmented image.

```
diff_im = im2bw (diff_im, T);
```

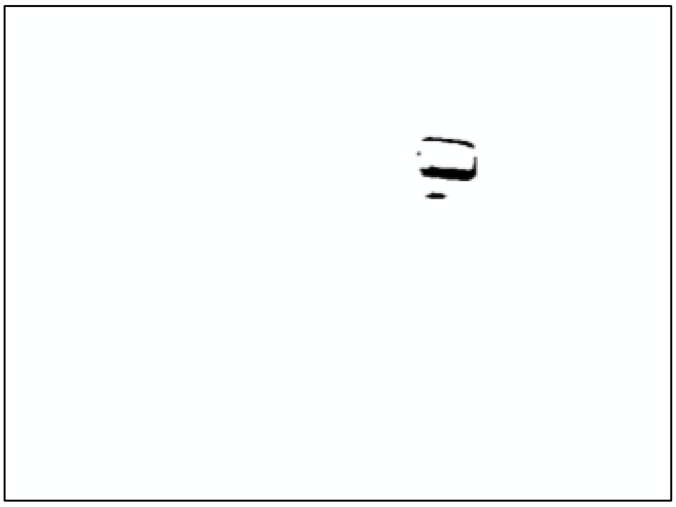


Figure 9. Negative Binary Image with Threshold 0.35 to Get Red Marker

2.7 Removing the Noise

The output of thresholding process might contain some impurities, in other word some noisy pixels in unwanted regions. To get rid of those noise pixels, the small objects or components should be removed from the image using `bwareaopen` function.

The bwareaopen function is based on a morphological binary operation which is called area opening operation. Area opening operation is obtained by specified number of dilation and erosion operations, which shrinks the region of the object by a specified number of pixels, and then enlarge it back to its original size. Shrinking a single pixel region deletes it from the image, so that the noise regions permanently disappear from the image [9].

2.7.1 Morphological Image Processing

Morphological Image processing methods are nonlinear transformations that can affect the shape and size of a binary region and reconfigure the structure of regions based on operations: *dilation*, *erosion* and set of compensations *opening*, *closing* and *boundary extraction* [10] [11]. This thesis uses area opening operation.

2.7.2 Binary Dilation and Erosion Operations

For both dilation and erosion operations, the main idea is sliding the binary image B on binary structuring element S similar to taking convolution across the image, and compare each pixel. The binary structuring element might have 4-or 8- active neighbours as seen in Figure 11 [12].

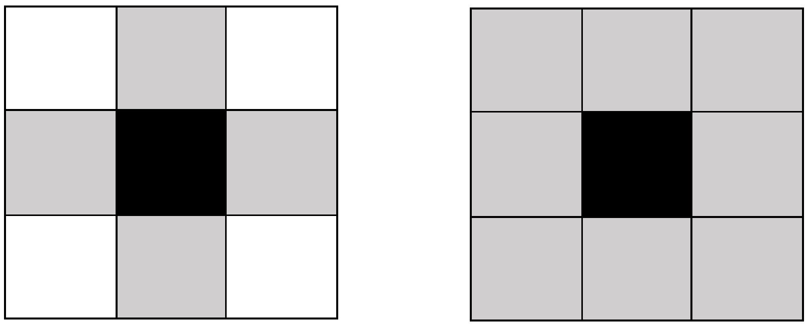


Figure 10. Shape of Structuring Element S, (A) 4- Neighbours, (B) 8- Neighbours

2.7.3 Binary Dilation Operation

The *binary dilation* operation is denoted by $B \oplus S$. Where B is a binary image, and S is a binary structuring element, while sliding the binary structuring element S across the binary image B, if there is a black pixel in B coinciding with the origin of S, the pixels in the image which is covered by the structuring element will be ‘black’. However nothing will change if the origin of S is coinciding with the ‘white’ pixel in B as seen in Figure [12-14]. In mathematical notation, the operation is expressed by:

$$B \oplus S = \cup_{a \in B} \{b + a \mid b \in S\}$$

2.7.4 Binary Erosion Operation

The *binary erosion* is denoted by $B \Theta S$ where B is a binary image and S is a binary structuring element, While sliding the S across B, if there is a black pixel in B coinciding with the origin of S nothing is done, however if a ‘white’ pixel in B falling on a black pixel in S, then the ‘black’ pixel in B is changed to white as shown in Figure (12-15).

In mathematical notation, the binary erosion operation is expressed by:

$$B \oplus S = \cup_{a \in B} \{x \mid x + bforeveryb \in S\}$$

2.7.5 Binary Opening

The binary opening operation used in our research with the aim to remove unneeded objects from the binary image which is the small components and pixels in the region and can be used for enhancing the binary image. The binary opening operation involves of erosion of the image by S then the output will followed by a dilation. The Binary Opening operation donated by $B \circ S.$ Figure (12-15).

In mathematical notation, the binary opening operation is expressed by:

$$B \circ S = (B \Theta S) \oplus S.$$

1	1	1	1	1	1	1	
			1	1	1	1	
			1	1	1	1	
		1	1	1	1	1	
			1	1	1	1	
		1	1				

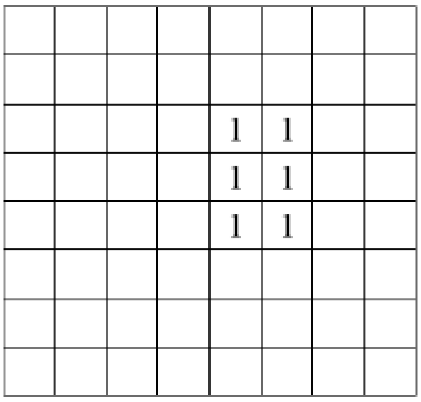
Figure 11. Typical Binary Image B.

1	1	1
1	1	1
1	1	1

Figure 12. Structuring Element 8-Neighbours.

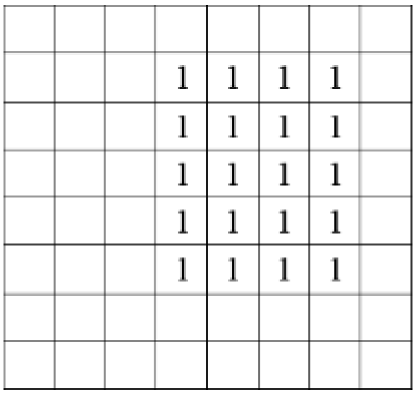
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1			

Figure 13. Binary Dilation B⊕S.



				1	1		
				1	1		
				1	1		

Figure 14. Binary Erosion B⊖S.



			1	1	1	1	
			1	1	1	1	
			1	1	1	1	
			1	1	1	1	
			1	1	1	1	

Figure 15. Binary Opening B∘S= (B⊖S) ⊕S

The output of bwareaopen function based on the previews operations will be presented by the Figure 17.

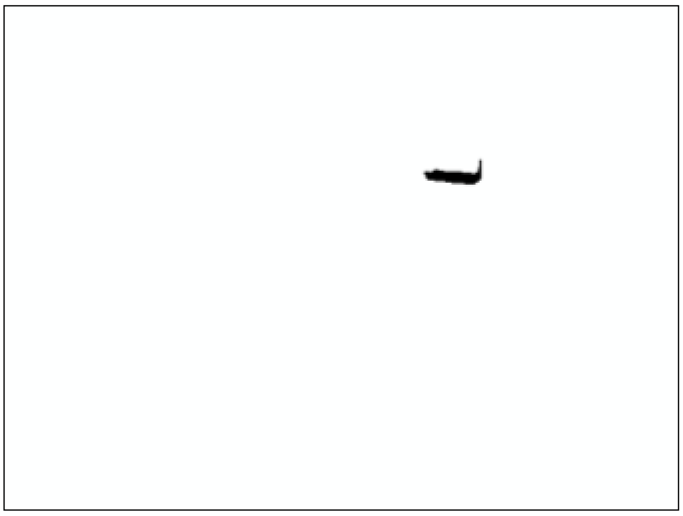


Figure 16. Negative Image for the Output of (Bwareaopen) Function

2.8 Labeling the Connected Components

Each pixel's value of the labeled binary image LB represent the label connected components, that's the simplest definition for the connected component labeling process. They are using the positive integer values of the pixels to label the connect components since it's much more convenient [14] [11].

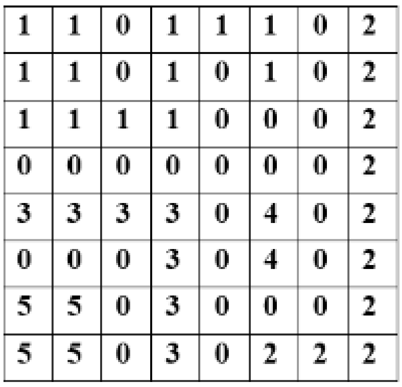
Many algorithms produced to label connected components depends on the size of the image and the ability to be stored in memory, since MATLAB stores matrix data in memory. Some of these algorithms scan the components one by one at a time across the image from left to right and top to the bottom Figure 19, an another algorithm designed which scan each two rows at a time also there is another algorithm works in parallel computing strategy for a big size of images [11]. The Algorithm used in MATLAB is A Recursive Labeling Algorithm which will be described.

2.8.1 Recursive Labeling Algorithm

RLA algorithm is set of procedures, and seeks to find the connected component (1-pixels) of an image B with (Maximum Number of Row + 1) and (Maximum Number of Columns + 1), in order to give an output labeled image LB Figure 20, the returned binary image based on one of two kind of scan-line orders either (four-neighbourhood) or (eight-neighbourhood) Figures (21-22) [11].

1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1

Figure 17. Binary Image.



1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

Figure 18. Connected Components Labeling.

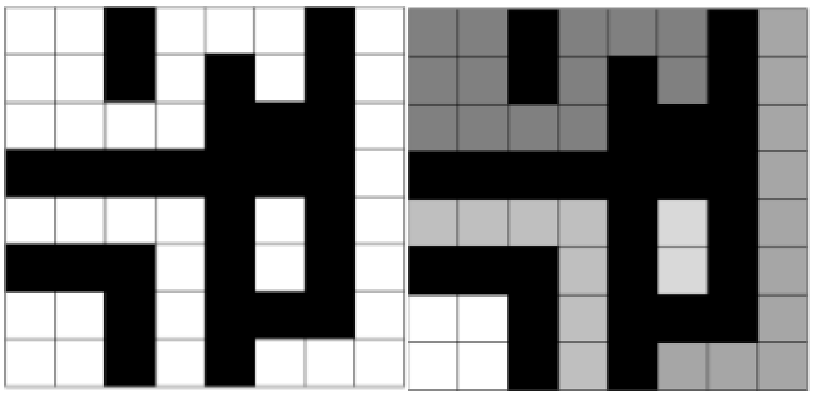
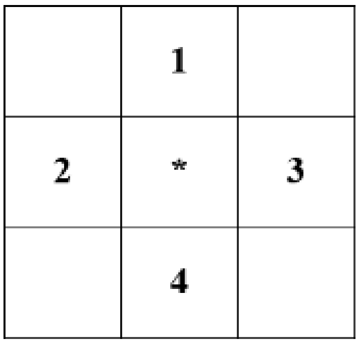


Figure 19. Binary Image and Labeling, Expanded For Viewing.



	1	
2	*	3
	4	

Figure 20. Four-Neighbourhood.

1	2	3
4	*	5
6	7	8

Figure 21. Eight-Neighbourhood.

2.8.2 RLA Mode of Operation

The first step of RLA algorithm is distinguishing the pixels (-1) from the component label (1) by negating the 1-pixels of a binary image using negate function, where the input a binary image B and output negated image later will be the labeled binary image LB. The next step is finding the pixels with value (-1) using same method for finding the connected components and their neighbours that have same value (-1) using searching procedure and giving them new label using 8-neighbors definition in our research, the neighbours (L, P) function used to return the position of all pixels' neighbours, algorithm 1 [11]. Which is to be represented later on in Figure 23.

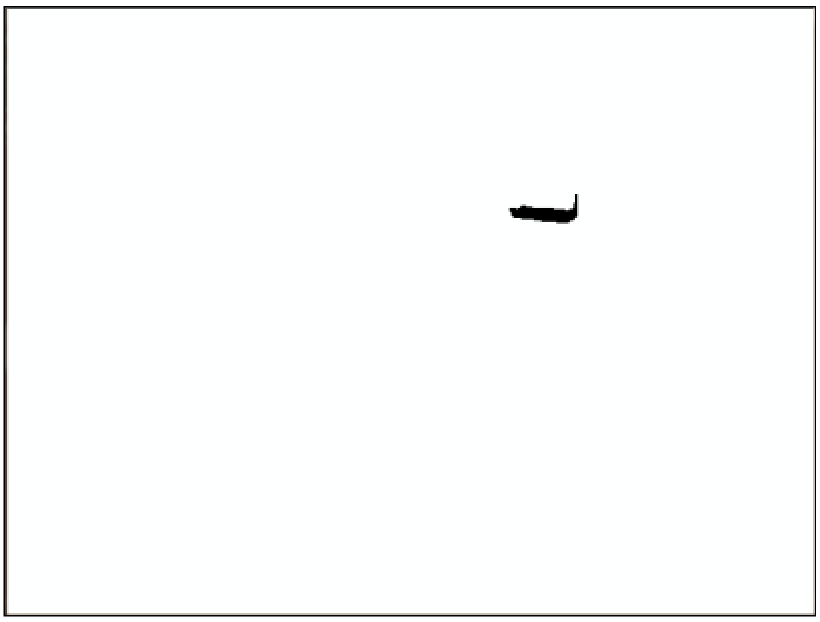


Figure 22. Negative Binary Image after Removing Noise.

2.9 Centroid

In last step, we have to obtain the (X, Y) coordinates from the region of the connected component of the binary image Figure 24, using “regionprops” MATLAB function. This region should be a numeric value which will be used in further processes. This process called (Moment and Algebraic Invariants), this approach has been improved since long time ago and used before introducing the first computer by (Ming Kuei Hu) in “1962”, and the Algebraic Invariants theory has been introduced by a mathematician called (David Hilbert) this theory used in many different image processing areas [15] [16].

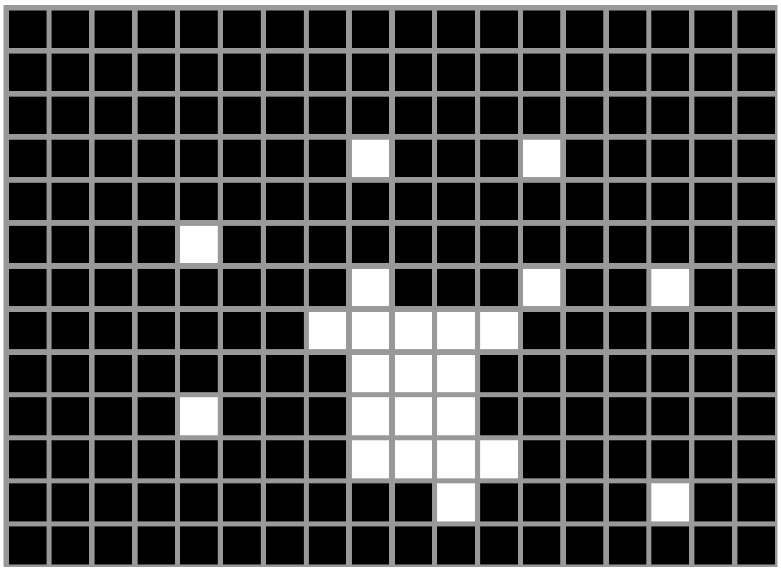


Figure 23. Binary Image with the Connected Component.

2.9.1 Mathematics of Moments

In Mathematics of Moment the image function known as *f*(x, y) . And the general order moment function represented by equation 3, this equation solves the functions which have only one variable [15] [16] [17].

$$\mu_{\eta} = \int_{-\infty}^{+\infty} (x-c)^{n} f(x)dx. \tag{3}$$

However in our interested case we are using the two-dimension (*x, y*) images which require to have two independent variables. The order of moment will be (*m*+*n*), where (*m, n*) are non-negative integer values (0, 1, 2…) and represented by equation 4 [15] [16] [17].

$$\mu_{m,n} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^m y^n f(x,y)\,dydx. \tag{4}$$

The central moment $\mu_{m,n}$ will be represented by equation 5, where c represent the point which is the order moment about it, and $f(c_x, c_y)$ indicates the centroid of the image function *f*(x, y).

$$\mu_{m,n} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} (x-c_x)^m (y-c_y)^n f(x,y)\,dydx. \tag{5}$$

The equation 3 can be represented by equation 6, in this equation they replaced the integration with summation to calculate the area of the binary image about point c which is (0, 0) [17].

$$\mu_{m,n} = \sum_{x=0}^{\infty}\sum_{y=0}^{\infty} (x-c_x)^m (y-c_y)^n f(x,y). \tag{6}$$

Since the point, we used to calculate the moment about it is (0, 0) we can remove each of (c_x, c_y) variables from the equation to be as equation 7 [15] [17].

$$\mu_{m,n} = \sum_{x=0}^{\infty}\sum_{y=0}^{\infty} x^m y^n f(x,y). \tag{7}$$

After substituting for each of m, n by zero the equation will be as equation 8 [15] [17].

$$\mu_{0,0} = \sum_{0}^{w}\sum_{0}^{h} x^0 y^0 f(x,y). \tag{8}$$

For x^0, y^0 can be removed from the equation because doesn't have any affection for the result since will be multiplied by either 0 or 1 which is the value of the image pixel. So the value of the pixel will be added to the moment equation 9 [17].

$$\mu_{0,0} = \sum_0^w \sum_0^h f(x, y). \tag{9}$$

2.9.2 Centroid

To find the centroid coordinates (x, y) for the calculated binary image area they using equation 10 [17].

$$\text{centroid} = \left(\frac{\mu_{1,0}}{\mu_{0,0}}, \frac{\mu 0{,}1}{\mu_{0,0}} \right). \tag{10}$$

However to simplified this formula they find each coordinate of (x, y) separately, where f(x, y) = 1 which means for all white pixels equations (11) (12) respectively.

$$sum_x = \sum \sum x f(x, y). \tag{11}$$

$$sum_y = \sum \sum y f(x, y). \tag{12}$$

Finally to find the average of each coordinate they divide each of coordinate's summation by the number of pixels equation (8). This methods has an advantage which is not sensitive to the image noise and disadvantage the centroid point might be not exact and shifted a little [17].

```
Stats = regionprops (BW, 'BoundingBox', 'Centroid');
```

The numerical outputs of color detection are in (x, y) format and it's obtained from the output binary image of the previews stage, that by applying the previews algorithms and mathematics formulas. Which represent the two coordinates of the observer's location. Where the centroid value are [428.09, 163.09], and the size of the bounding Box is [398.5, 147.5, 54, 23] figure 25, represents the center of the red color and the bounding box.

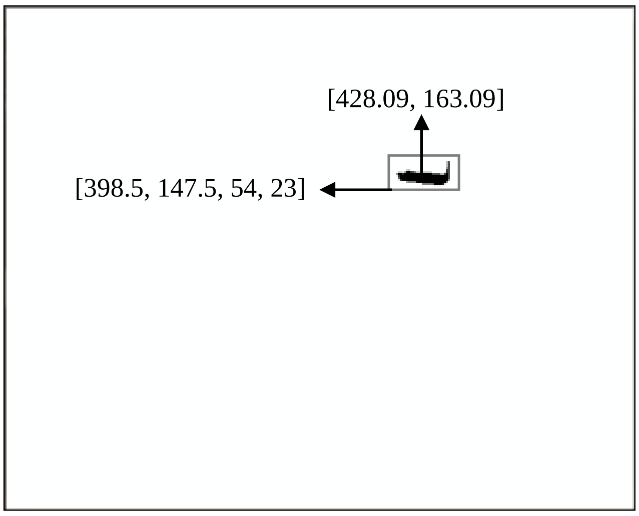


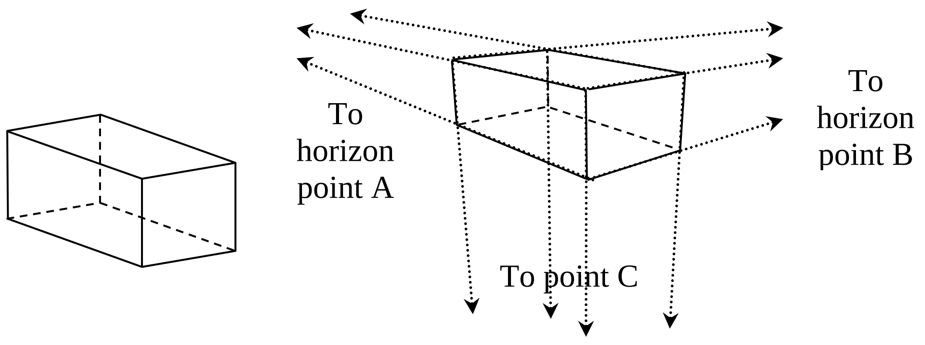
Figure 24. Center of the Red Color and the Bounding Box.

Chapter 3

VIEWING AND PROJECTION

3.1 Introduction

The three-dimension (3D) scenes based on two-dimension (2D) image plane uses projective geometry extensively and called planar geometric projection Where, the projective geometry of any object formed by the projectors 'Lines'. These projectors obtained when the projectors passed all the object's points. And getting the image will be formed by the intersection of these projectors, these projectors emitting from the center of projection 'single point'. There are two kind of projections called perspective projection and parallel projection Figure 26 [18] [19].



A) Orthographic Projection, B) Perspective Drawing.
Figure 25. Parallel Projection and Perspective Projection.

3.2 Planar Geometric Projection

3.2.1 Parallel Projection

Called parallel projection since all the projectors are parallel to each other, which means has an infinite center of projections. And the multi projections can illustrate the shape of the object Figure.20 B. The parallel projection produce unrealistic image since its preserves the length of the lines as well as it gives a uniform foreshortening. That's why the parallel projection extensively used in engineering drawing. The parallel projection can be divided into three types as following. [19].

3.2.1.1 Orthographic Parallel Projections

The orthographic parallel projection provide a realistic shape for an object. However, needs different views to describe any object, that depending on the complexity of the object. Orthographic parallel projection commonly used in engineering drawing [18].

3.2.1.2 Axonometric Parallel Projection

The axonometric parallel projection provides three-dimensional representation by viewing the three adjacent faces of any object in one view. However in the axonometric projection the distant and the close parts are represented at the same scale that affects the three-dimension representation view of axonometric projections and become distorted. As well as the axonometric parallel projection can't represent irregular, circles and complex shapes [18].

3.2.1.3 Oblique Parallel Projections

The oblique parallel projection combines each of orthographic and axonometric projections properties. The oblique projection solves the cylindrical and irregular shapes on the contrary of axonometric projection. Also, the oblique projections provide the observer three-dimensional representation by illustrating two adjacent faces of the object as well as representing the shape of any object as its [18].

3.2.2 Perspective Projection

The projection called perspective projection when the center of projection ‘point’ is finite, the perspective projection able to create an image equivalent to the image that created by eyes, since it represents any object as its can be seen by the observer. However it distorts the lines’ length and intersection angle, that’s why it’s not suitable for engineering drawing. The intersected coordinate axis’s classifies the perspective projection that might be one, two or three points of perspective Figure 28 [18] [19].

3.3 Homogeneous Coordinates and Matrix Representations

In homogeneous coordinates, any point in space with coordinates (x, y, z) can be represented in 4D to be as (x, y, z, 1), and the point in 4D with (x, y, z, w) coordinates can be represented in 3D. Where ($w \neq 0$) which will be in 3D as (x/w,y/w,z/w,w/w). The planar geometric projection of any point has represented in homogeneous coordinates can be expressed in a 4X4 matrix. Deriving the planar geometric projection require rotation, shearing, translation and perspective transformations. The rotation and shearing can be represented by 3x3 matrix equation (13). However, the translation and perspective can be presented by 4x4 matrix equation (14) [18].

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & p_1 \\ a_{21} & a_{22} & a_{23} & p_2 \\ a_{31} & a_{32} & a_{33} & p_3 \\ t_1 & t_2 & t_3 & 1 \end{bmatrix} \tag{14}$$

Where the (a_{ij}) the linear transformation, (t_i) represents the translation and (p_i) represents the perspective [18].

3.3.1 Projection Matrices

By reading the introduction of this chapter and understanding the type of projections and the differences between them in terms of selecting the projection planes as opposed to the rotations of the object, the projection matrix can be simply derived by three principle steps rotation, translation and projection [18].

Actually in this thesis we have used 'viewmtx (az, el, phi)' MATLAB function, this function produces the perspective view by applying different equation including rotation and perspective projection, however produces the perspective without translation with aim to produce a perspective view.

The 'viewmtx' MATLAB function can return different type of transformation such as orthographic transformation or perspective transformation either by specifying the point of the plot cube or without specifying it.

Where *az* is the *x* value, el is the y value and phi is the view angle value and all these values are in degree which require to converted to radians by multiply each by π and divide it by 180 for each.

3.3.2 Rotation

The transformation matrix formed by applying two rotations first about z-axis followed by rotation about x-axis equation 15, 16.

$$T = Rot(z,\theta)Rot(x,\theta).$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin\theta & 0 \\ 0 & \sin\theta & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$T = \begin{bmatrix} \cos(az) & \sin(az) & 0 & 0 \\ -\sin(el)*\sin(az) & \sin(el)*\cos(az) & \cos(el) & 0 \\ \cos(el)*\sin(az) & -\cos(el)*\cos(az) & \sin(el) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

3.3.3 Perspective Projection

Since the graphic system already scaled by w before mapping on the screen in the view matrix MATLAB function the perspective transformation generated by the equation 17. However, the perspective doesn’t appear on the frame. Where f is the distance of the observer from the screen, and phi is the view angle and we selected to be 8 in degree. The distance defined by $f = sqrt(2)/2/\tan/(phi*\pi/360)$.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \tag{17}$$

To overcome this problem, the result of transformation after the view matrix equation 18. Was scaled by the last entry of the output (w_i) equation 19.

$$T = \begin{bmatrix} x_i \\ y_i \\ z_i \\ w_i \end{bmatrix} \quad . \tag{18}$$

$$T = \begin{bmatrix} \frac{x_i}{w_i} \\ \frac{y_i}{w_i} \\ \frac{z_i}{w_i} \\ \frac{w_i}{w_i} \end{bmatrix} \tag{19}$$

Chapter 4

IMPLEMENTATION, TESTING AND RESULTS

4.1 Introduction

In this thesis, the system was built with MATLAB using acquisition toolbox. And since the main task of the proposed system is based on image processing to obtain the input data from LEDs with red colour Figure 27, it is essential for the designed system to have a Webcam either internal or external Webcam to be used as sensor to collect the data.

Figure 26. Red Color Flash.

4.2 Implementation of User Interface

The GUI of the proposed system has been implemented in MATLAB Figure 28,

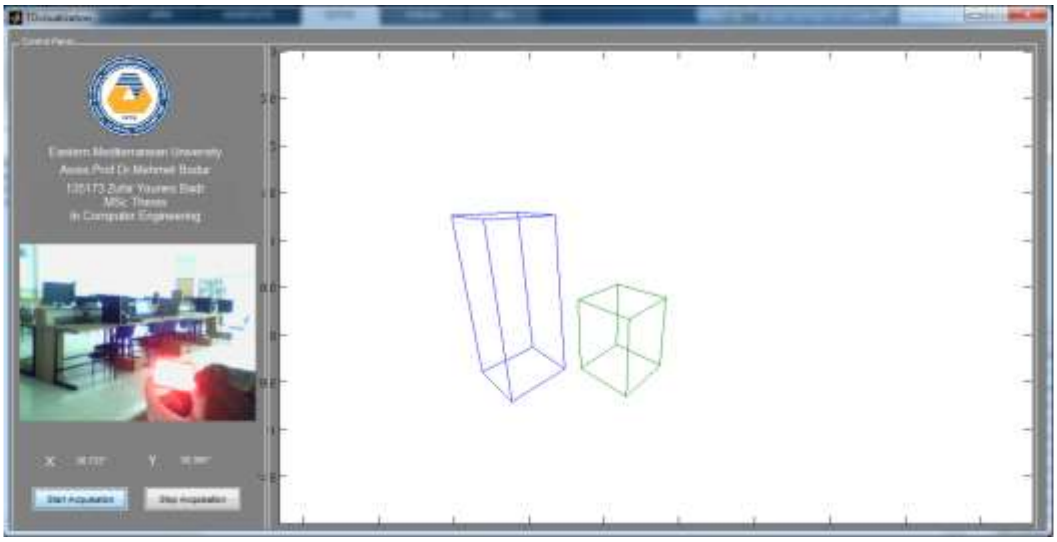


Figure 27. User Interface of the Implemented System.

As shown, the interface includes two MATLAB axes tools. The first axes is used to display the sequence of images ‘video’ in real time as acquired into the system by the webcam while the other axes is used to present the 3D image with rotation, translation and perspective. Inclusively, the interface contains two pushbuttons (Start Acquisition and Stop Acquisition), the Start Acquisition pushbutton is employed in order to start accruing the image data into the system, starting by specifying the image format which is followed by the acquisition and initialization of the number of frames and iteration of the process. Each frame is processed with the aim to separate the red colour from the pure image, if the image does not contain any red object, the next frame will be acquired until an object to be processed is found, otherwise system runs a required process to obtain the x, y value of the red object. These values will be scaled for use in the view matrix to add 3D effect into the 2D images and will be displayed on the GUI using the Text Boxes, the output of the view matrix will be displayed on the second axes tool. This process is repeated until the end of the iteration. The other pushbutton stops the image data acquisition to the system. Figure 31 represent the Flowchart of the data in the system.

The best way to clarify the perspective projection in adding 3D effect onto a 2D image understandably is through the application of translations, transformations and perspective projections equations on geometrical shapes, for this purpose we have designed two cubes with different heights as shown in Figure 29, which we have placed on a frame with different positions, in the general view at the middle of x, y coordinates where the first cube placed at points x, y = -1 and z= 0, the other cube has shifted a little bit for both of x, y coordinates to be x, y =0.2 and z= 0 Figure 30.

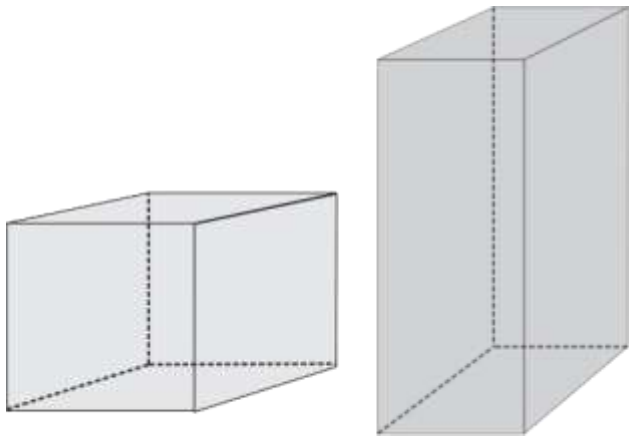


Figure 28.Two Different Cubes.

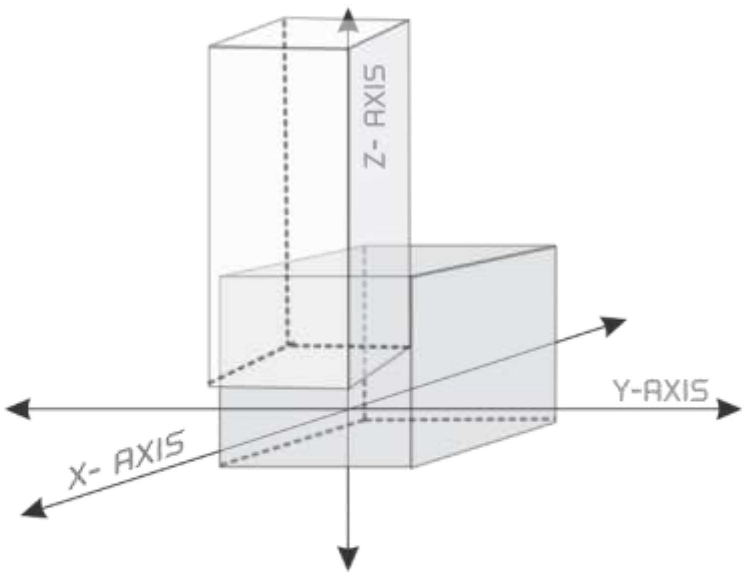


Figure 29. Cubes Placed On The Frame.

The implementation of the proposed system is explained by the following pseudocode.

```
1. Initialize $r \leftarrow red$ color; $\mathrm{T} \leftarrow 0.18$ $\phi \leftarrow 8$;
2. Loop for $\mathrm{i} = 0 \text{ to } 500$;
3.         $f(x, y) \leftarrow frame$; acquire frame from video stream
4.         if $r \in f(x, y)$
                   $f(x, y) = \alpha\, \mathrm{I}_{\mathrm{color}}(n, m, r) + \beta\, \mathrm{I}_{\mathrm{color}}(n, m, g) + \gamma\, \mathrm{I}_{\mathrm{color}}(n, m, b)$; c
                   $f(x, y) \leftarrow \mathrm{n}_{\text{grey-scale}}(m, n)$; get gray-scale of red component.
                   $\mathrm{g(x,\ y)} \leftarrow \mathrm{med}\{\mathrm{f(x - i,\ y - i), i,\ j} \in \mathrm{W}\}$; median filter.
                   if $\mathrm{g(x,\ y)} \geq \mathrm{T}$ then $\mathrm{g(x,\ y)} \leftarrow 1$; else $\mathrm{g(x,\ y)} \leftarrow 0$;
                   $\mathrm{g(x,\ y)} \leftarrow \mathrm{B} \oplus \mathrm{S} = \bigcup_{a \in | B}\{B + a \mid b \in S\}$;
                   $\mathrm{g(x,\ y)} \leftarrow$ recursive labeling algorithm;
                   $(x,y) \leftarrow \frac{\sum\sum\int x\, f(x,y)}{2}, \frac{\sum\sum y\, f(x,y)}{2}$;    (moment)
                   $(x,y) \leftarrow 45 - \frac{\left(\frac{60}{640}\mathrm{x} - 30\right)}{2}, 20 - \left(\frac{50}{480}\mathrm{y} - 35\right)$; (scale to view angle)
                   $[\mathrm{x, y, z, w}] = \mathrm{viewmatrix}(\mathrm{x, y}, \phi)$; (get the transformation matrix)
                   $[\mathrm{x}_j, \mathrm{y}_j, \mathrm{z}_j, \mathrm{w}_j] = [\frac{\mathrm{x}}{w}, \frac{\mathrm{y}}{w}, \frac{\mathrm{z}}{w}, \frac{\mathrm{w}}{w}]$; (perspective scaling)
                   plot(x, y);
                   i++;
           else go to Line-3 (to acquire a new frame)
5          End Loop
```

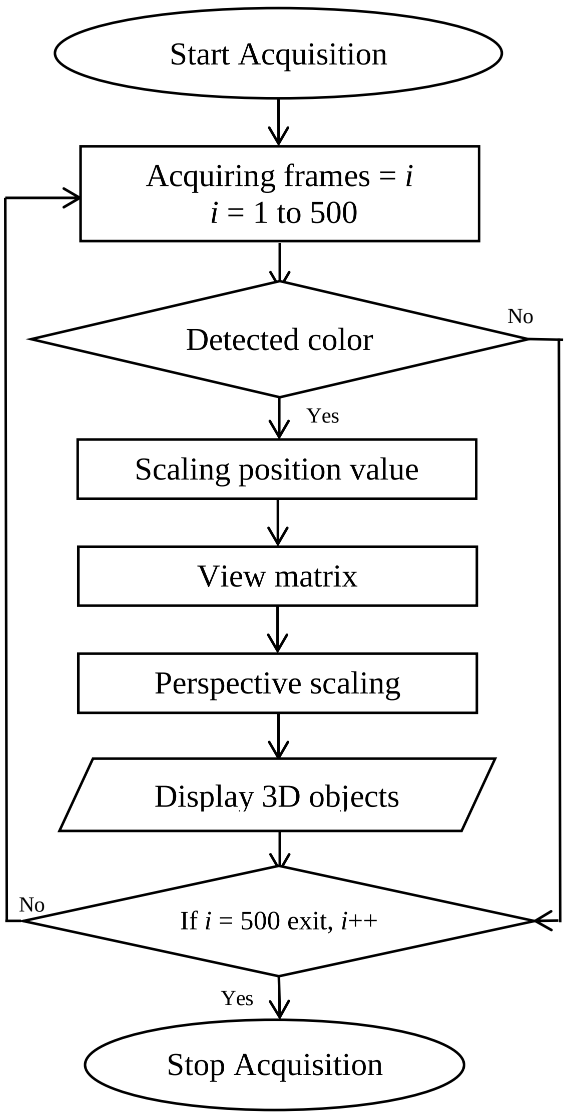


Figure 30. Flowchart of Implemented System.

4.3 Angle of View

The view angle of the observer has been scaled-down for both of x, y input to be $0 \leq x \leq 60$ and $0 \leq y \leq 60$, that by multiplying the pure value by 60 and divided by 640 for x direction equation18. And multiplying by 60 and divided by 480 for y direction equation 19 Figure 32. Where [640, 480] represents the maximum width and height values of the input video which has been specified at video source specifications step, we have been scaled because is the obtained values not suitable for the view matrix and much higher than what we need.

$$X_r = \frac{x_{pure} * 60}{640}. \tag{18}$$

$$Yr = \frac{y_{pure} * 50}{480}. \tag{19}$$

Actually the rotations that will be generated by the view matrix will be fixed to be between 30 and 60 degrees for x-axis and between 5 and 55 for y-axis, that to produce a view which is equivalent to natural and real view with inverse rotation to the observer movement to show the hidden face. That required us to scale the x and y values by subtracting the x-value from 30 and divided by 2 the output of this step will be subtracted from 45 for x-axis equation 20. For y-axis, we have subtracted the scaled value from 35 and the output of this step again subtracted from 20 equation 21.

$$X_{view} = 45 - \left[\frac{X_{input} - 30}{2}\right] \tag{20}$$

$$Y_{view} = 20 - [Y_{input} - 35] \tag{21}$$

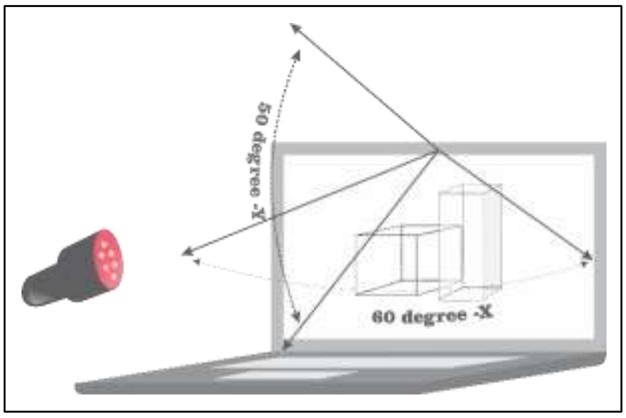


Figure 31. View Angle Bounds

The output of X_{view} and Y_{view} will be used as input to the MATLAB view matrix function which is we have discussed in the section (3.3.1) in the previous chapter.

4.4 Results

To evaluate the work done in this thesis we have built a real platform using carton with dimensions (40*40*25) cm, as well as we have built to cubes with different sizes which are similar to that ones we have built in MATLAB as it is shown in Figures (33-34), that to compare the real platform view with the MATLAB planar view.

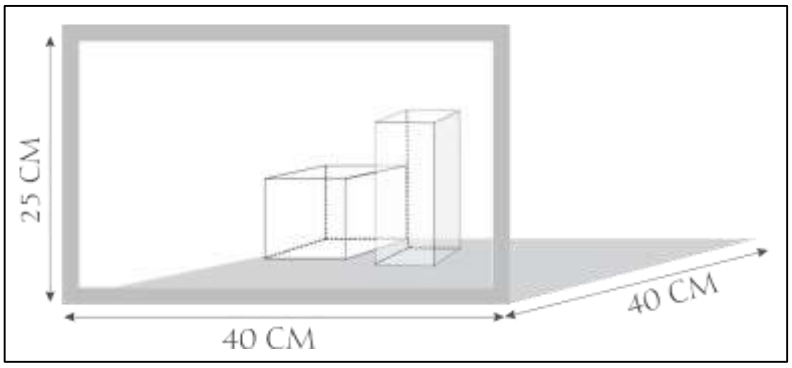


Figure 32. Platform Dimensions.

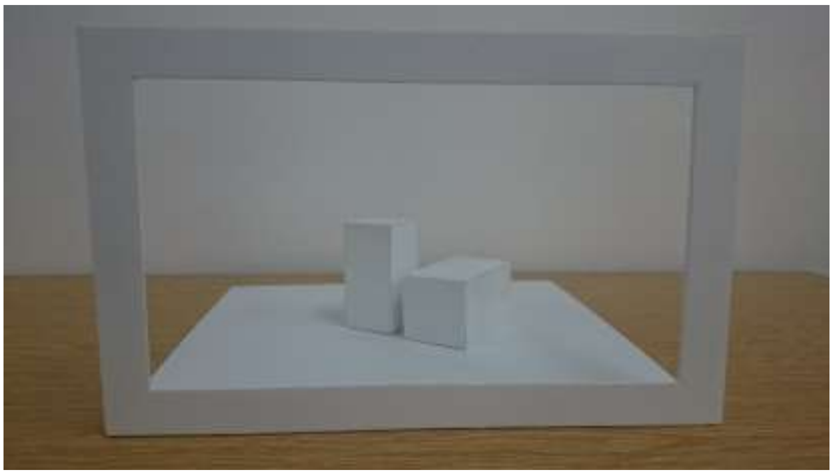


Figure 33. A Real Platform.

The implemented system gives similar view to the real view on the real platform and shows 3D effect on the 2D image as shown in Figures (35-36).

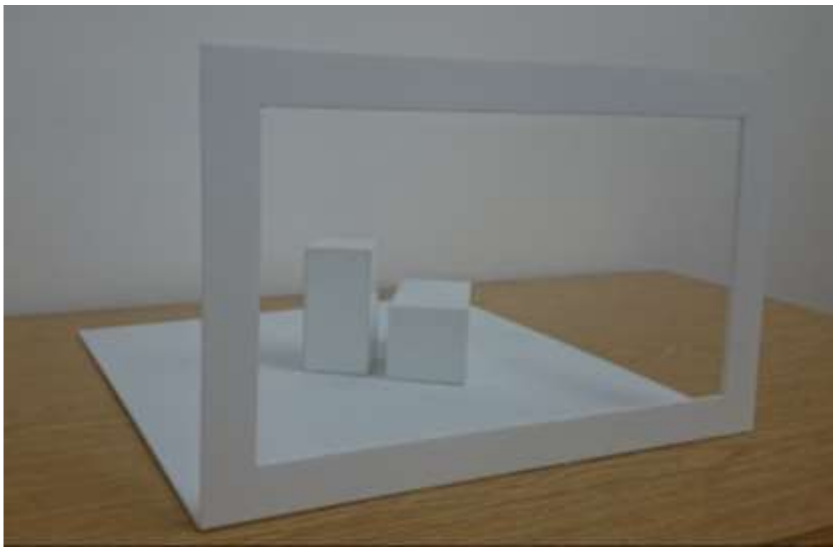


Figure 34. The Real 3D View.

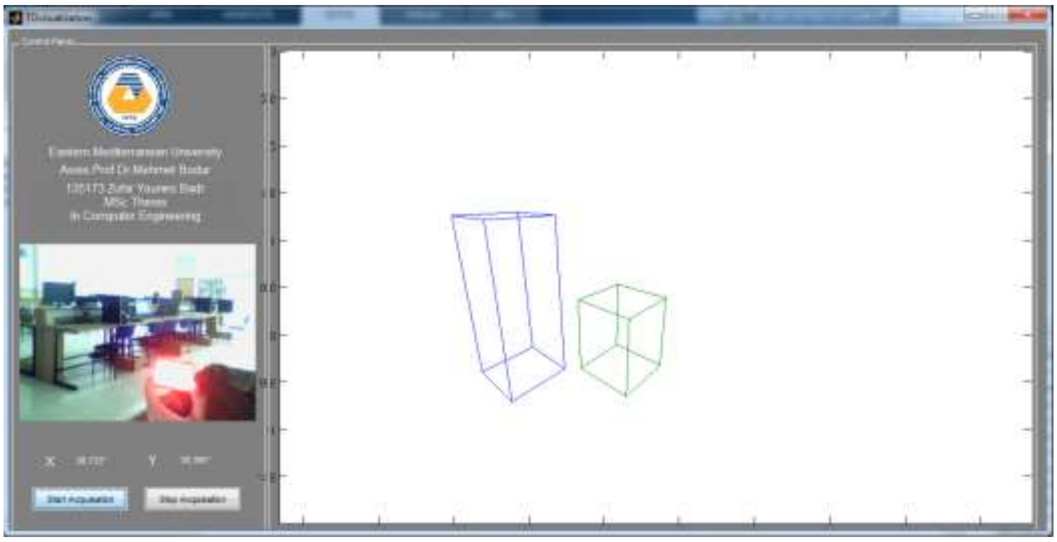


Figure 35. 3D View Created by the MATLAB.

Chapter 5

CONCLUSION

In this thesis, the prototype with aim to add three-dimension (3D) effect on two-dimension (2D) image has been successfully implemented to detect a red coloured marker in real time, and redraw the perspective appearance of a 3D object for the measured view angle of the marker. The implemented system calculates the position of the red coloured marker on the hat of the observer. Thereafter it uses the position information to calculate the homogeneous transformation matrix. The coordinates of the corners of an object is transformed by this transformation matrix to a 2D perspective view. The evaluation of the implemented system based on the comparison of the implemented view with a real platform that we have already constructed. The view of the implementation was exactly same as the real view.

In this thesis, the prototype has been implemented using MATLAB which is the multi-usage and commonly used software for geometric implementations with acquisition toolbox and GUI interface. Also, we have used the Webcam to act as a sensor to collect pure data that will be processed in further steps to obtain the numerical values.

The negative and positive errors as well as trembling performed by the red color detection and tracking algorithms caused some weakness and instability in the view, in addition to the system cannot work properly in case of there are more than one user ‘observer’, that can be considered as disadvantage in the performance of the implemented system.

5.1 Suggested Works

We can divide the suggested work into two points as following:

1. The red color detection and tracking based algorithm still in beginning research stages and needs more works and improvements to get high accuracy and stability.
2. Using an another technique based on biometrics algorithms such as eyes or face detection and tracking to obtain the view angle might be much better and suitable for the observer, which does not require holding any red light.

REFERENCES

[1] Kontsevich, L. L. (1998). Defaults in stereoscopic and kinetic depth perception. *Proceedings of the Royal Society of London B: Biological Sciences*, *265*(1406), 1615-1621.

[2] Lages, M., & Heron, S. (2010). On the inverse problem of binocular 3D motion perception. *PLoS computational biology*, *6*(11), e1000999.

[3] Guo, Z. (2001). *Object Detection and Tracking in Video*," Department of Computer Science, Kent State University.

[4] Avilable at http://www.mathworks.com/help/imaq/creating-image-acquisition-objects.Html

[5] Available at http://www.mathworks.com/help/imaq/adaptorkit/implementing-the-acquisition-thread-function.Html.

[6] Avilable at http://stackoverflow.com/questions/20360778/matlab-extracting-red-color-from-an-image.

[7] Zhu, Y., & Huang, C. (2012). An improved median filtering algorithm for image noise reduction. *Physics Procedia*, *25*, 609-616.

[8] Solomon, C., & Breckon, T. (2011). *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons.

[9] Available at http://www.cacr.caltech.edu/~cunha/bi199/three.html.

[10] Shen, S. (1993). Application of morphological image processing to texture decomposition.

[11] Shapiro, L., & Stockman, G. C. (2001). Computer Vision. 2001. *ed: Prentice Hall*.

[12] Available at http://elearning.vtu.ac.in/17/e-Notes/DIP/segmentation_DIP-SDG.pdf.

[13] Mai, L. (2010). Introduction to Image Processing and Computer Vision.

[14] Available https://nf.nci.org.au/facilities/software/Matlab/toolbox/images/bwlabel.html.

[15] Flusser, J. (2006, February). Moment invariants in image analysis. In *proceedings of world academy of science, engineering and technology* (Vol. 11, No. 2, pp. 196-201).

[16] Hu, M. K. (1962). Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, *8*(2), 179-187.

[17] Available at http://www.aishack.in/tutorials/image-moments/.

[18] Carlbom, I., & Paciorek, J. (1978). Planar geometric projections and viewing transformations. *ACM Computing Surveys (CSUR)*, *10*(4), 465-502.

[19] Thomson, R. A. (2006). The Direct3D Graphics Pipeline.