

OBJECT DETECTION AND PATH TRACKING

Submitted in partial fulfilment of the Degree of
Bachelor of Technology



May – 2014

Under the Supervision of
Dr. Pradeep Kumar

by

NISHIT GUPTA (101084)

SHIVAM GUJRAL (101095)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

Certificate

This is to certify that project report entitled “Object Detection and Path Tracking”, submitted by Nishit Gupta, Shivam Gujral in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Wagnaghat, Solan has been carried out under my supervision.

Date: 24.05.14



Dr. Pradeep Kumar
Associate Professor,
ECE Deptt. JUIT
Wagnaghat, Solan

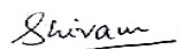
Acknowledgement

Knowledge, energy, and time are the resources in the completion of this project but the most requisite is the proper guidance of our respected mentor, **Dr. Pradeep Kumar** (Associate Professor, Department of Electronics and Communication) to whom we extend the sincere word of thanks, for his invaluable cooperation and help throughout the project. He acted as constant source of motivation throughout the development stage of the project

We would like to thank our H.O.D. **Dr. Sunil Bhooshan** and all the panel members for their valuable suggestion and guidance in all the seminars and viva-voce during evaluation.

We also express our obligation to all people who helped us directly or indirectly in the completion of this project. No thanks can counter our indebtedness to our parents and families who have been with us through every thick and thin, we thank them from core of our hearts.

Date: 24.05.2014



Shivam Gujral 101095



Nishit Gupta 101084

Table of Content

Certificate	i
Acknowledgement	ii
List of Figures	iv
Abstract	v

S. No.	Topic	Page No.
1.	INTRODUCTION	1
1.1	Overview	1
1.2	Basic Definitions	1
2.	OBJECT DETECTION	6
2.1	Object Detection Mechanism	6
3.	CANNY EDGE DETECTION AND IMPLEMENTATION	8
3.1	Introduction	8
3.2	Test Image	8
3.3	Canny Edge Detection Algorithm	9
3.4	Implementation	19
4.	OBJECT TRACKING	21
4.1	Introduction	21
4.2	Implementation	22
4.3	Problems	24
5.	CONCLUSION	25
	REFERENCES	26

List of Figures

S. No.	Title	Page No.
1.	Image Processing Model	4
2.	Importance of Image Processing	5
3.	Flowchart representing steps in object detection and tracking	7
4.	Test Image	9
5.	Flowchart of canny edge detection algorithm	10
6.	Smoothed Image	11
7.	Gaussian Filter in One-Dimension	12
8.	Gaussian Filter in Two-Dimension	12
9.	Sobel Operator in X-direction	13
10.	Sobel Operator in Y-direction	14
11.	Image with Gradient Magnitudes	15
12.	Non-maximum Suppression	16
13.	Double Thresholding	18
14.	Edge hysteresis	19
15.	Two Frames of video	19
16.	C.E.D on Frames	20
17.	Output	20
18.	Two frames showing Centroid	22
19.	Path traced by Moving Man	23
20.	Image showing Centroid (Another example)	23
21.	Path traced by Ant	24

Abstract

In this project we propose to use Image processing algorithms for the purpose of Object Detection and Tracking and implement the same using MATLAB.

We develop an object detection method using canny edge detection algorithm. The Canny edge detector is a very popular and effective edge feature detector that is used as a pre-processing step in many computer vision algorithms. It is a multi-step detector which performs smoothing and filtering, non-maxima suppression, followed by a connected-component analysis stage to detect true edges, while suppressing false non-edge filter responses. While there have been previous (partial) implementations of the Canny and other edge detectors on GPUs, they have been focussed on the old style GPGPU computing with programming using graphical application layers, but we programmed it in to MATLAB, a popular interactive simulation package often used by researchers.

The aim of motion tracking (the next step our project) is to detect and track the moving objects through a sequence of images. Motion tracking is not only useful for monitoring activity in public places, but is becoming a key ingredient for further analysis of video sequences. Information about the location and identity of objects at different points in time for instance is the basis of detecting unusual object movements (e.g. someone being mugged at an ATM) or coordinated activities (e.g. strategic plays in a football game).

CHAPTER 1

INTRODUCTION

1.1 Overview

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Object tracking is the process of locating a moving object in time using a camera. The algorithm analyses the video frames and outputs the location of moving targets within the video frame.

1.2 Basic Definitions

Image

An image may be defined as a two-dimensional function $f(x, y)$, where x and y are spatial (plane) coordinates and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When (x, y) and the amplitudes of f are all finite, discrete quantities, we call the image a **digital image**.

Pixel

The field of digital image processing refers to processing digital images by means of a digital computer. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements and pixels. Pixel is the term most widely used to denote the elements of a digital image.

In digital imaging, a **pixel**, or **pel (picture element)** is a physical point in an image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen. The address of a

pixel corresponds to its physical coordinates. LCD pixels are manufactured in a two-dimensional grid, and are often represented using dots or squares, but CRT pixels correspond to their timing mechanisms and sweep rates.

Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. The intensity of each pixel is variable. In color image systems, a colour is typically represented by three or four component intensities such as red, green, and blue.

The word *pixel* is based on a contraction of *pix* ("pictures") and *el* (for "element").

Image Processing

In imaging science, **image processing** is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

Image processing usually refers to digital image processing, but optical and analog image processing also are possible. The *acquisition* of images (producing the input image in the first place) is referred to as imaging.

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this manipulation can be divided into three categories:

1. Image Processing (image in -> image out)
2. Image Analysis (image in -> measurements out)
3. Image Understanding (image in -> high-level description out)

An image may be considered to contain sub-images sometimes referred to as regions-of-interest, ROIs, or simply regions. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve colour rendition.

Sequence of image processing:

Most usually, image processing systems require that the images be available in digitized form, that is, arrays of finite length binary words. For digitization, the given Image is sampled on a discrete grid and each sample or pixel is quantized using a finite number of bits. The digitized image is processed by a computer. To display a digital image, it is first converted into analog signal, which is scanned onto a display.

After converting the image into bit information, processing is performed. This processing technique may be Image enhancement, Image restoration, and Image compression.

Image enhancement:

It refers to accentuation, or sharpening, of image features such as boundaries, or contrast to make a graphic display more useful for display & analysis. This process does not increase the inherent information content in data. It includes gray level & contrast manipulation, noise reduction, edge crispening and sharpening, filtering, interpolation and magnification, pseudo coloring, and so on.

Image restoration:

It is concerned with filtering the observed image to minimize the effect of degradations. Effectiveness of image restoration depends on the extent and accuracy of the knowledge of degradation process as well as on filter design. Image restoration differs from image enhancement in that the latter is concerned with more extraction or accentuation of image features.

Image compression:

It is concerned with minimizing the number of bits required to represent an image. Application of compression are in broadcast TV, remote sensing via satellite, military communication via aircraft, radar, teleconferencing, facsimile transmission, for educational & business documents, medical images that arise in computer tomography, magnetic resonance imaging and digital radiology, motion, pictures, satellite images, weather maps, geological surveys and so on.

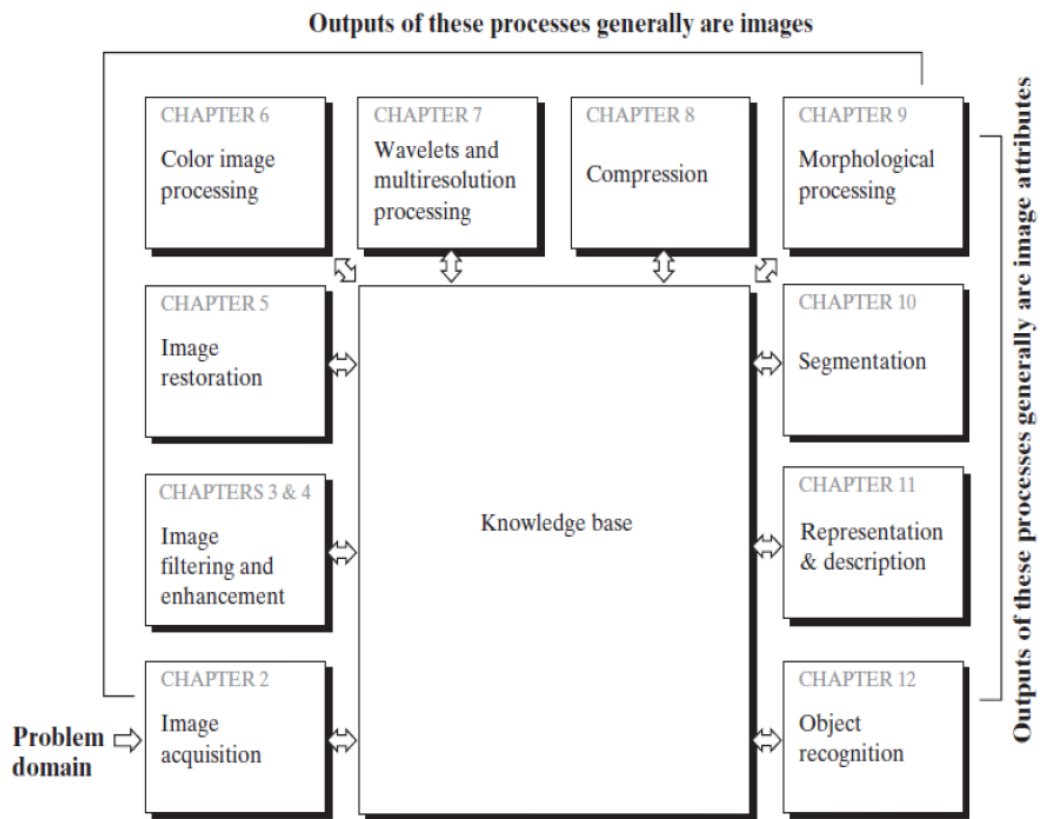


Figure 1.1: Image processing model

Digital camera images

Digital cameras generally include dedicated digital image processing chips to convert the raw data from the image sensor into a colour-corrected image in a standard image file format. Images from digital cameras often receive further processing to improve their quality, a distinct advantage that digital cameras have over film cameras.

Why Image Processing?

1. Among the rapidly growing technologies today, with its applications in various aspects of a business.
2. Forms core research area within engineering and computer science disciplines too.
3. Can commercial software do all the work?

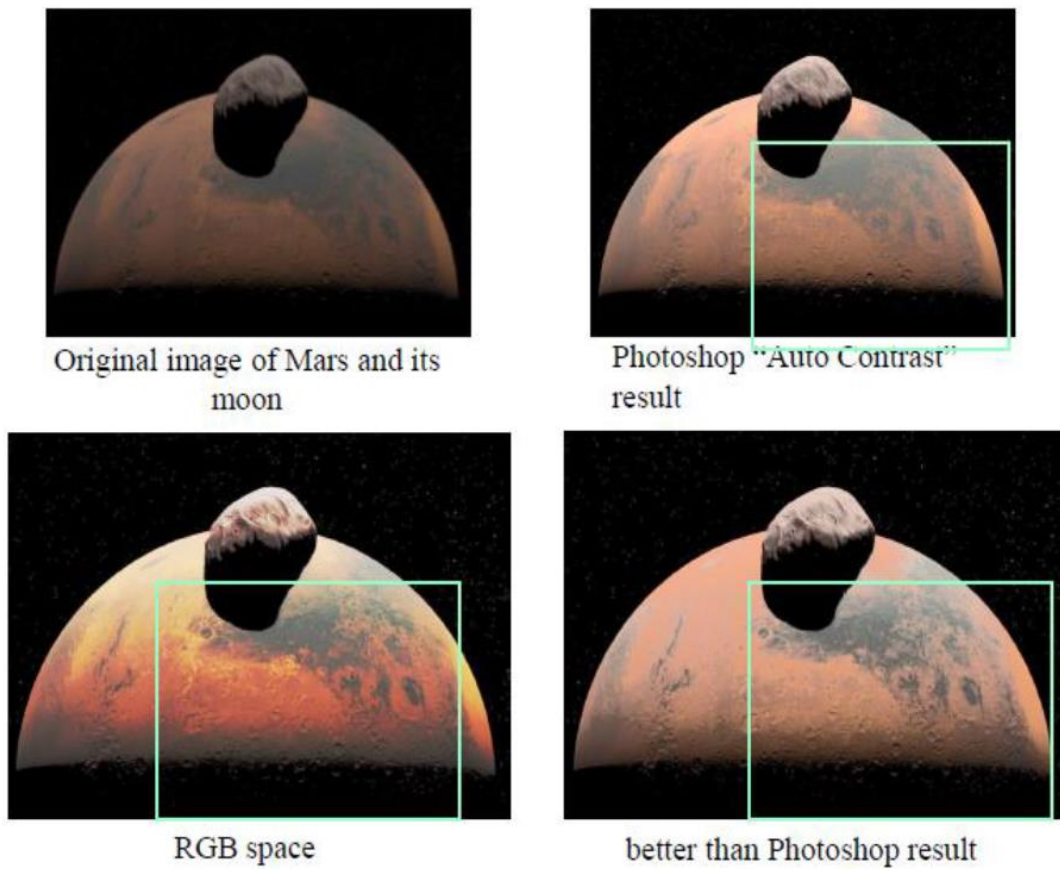


Figure 1.2: Importance of image processing

CHAPTER 2

OBJECT DETECTION

2.1 Object Detection Mechanism:

1. System environment
2. Image Acquisition
3. Frame Generation
4. Canny Edge Detector

2.1.1 System Environment

Our aim is to work in an unstructured environment. This provides greater system flexibility and portability but can make reliable segmentation more difficult. As this environment requires the need to distinguish the objects of interest from any other objects that may be present within the frame. This limitation may be overcome by restricting the target objects to saturated and distinctive colours to enable them to be distinguished from the unstructured background. Augmenting the unstructured environment with structured colour in this way is a compromise that enables a much simpler segmentation algorithm to be used. Another method to maintain the colour distribution is to keep the background environment a constant. This way, only the target is in motion and the system is able to track its motion in a 2-D frame.

2.2.2 Image Acquisition

The image capture is performed using a colour video camera which produces a stream of RGB pixels. The camera is mounted on a tripod stand with a fixed Background that contains the object to be tracked. A brief 5 - 10 second video is recorded in .avi format. We require a lower resolution as it will have a significantly higher acquisition rate for the observation of faster events. The size of the video frame is set to 512x512 pixels.

The video obtained is read in the computer using Matlab. The software processes the entire video and converts it into Image frames. Depending on the accuracy required

and computational capability of the System, the frames can be interlaced.

2.2.3 Frame Generation

The video is fed in the Matlab program. The program reads the .avi file and converts it to frames. Consider a 10 second video, let a total of 100 frames will be produced in RGB format. These frames are then stored as individual bitmap files (total of 100 files). The bitmap files are arranged in the order of their occurrence in the video. The first frame is selected as the Base - Background Frame. The remaining bitmap files are used for the process of Object Tracking.

2.2.4 Background and Object Identification

It is important that the object needs to be differentiated from the background. The colour elements must be eliminated and the recognition is done in gray scale. With a still environment, the background frame is selected as the first frame. Considering the 10 second video, the 60th frame is randomly selected as the Object frame - these two frames form the basis for Object Recognition. It gives information about the shape and size of the object.

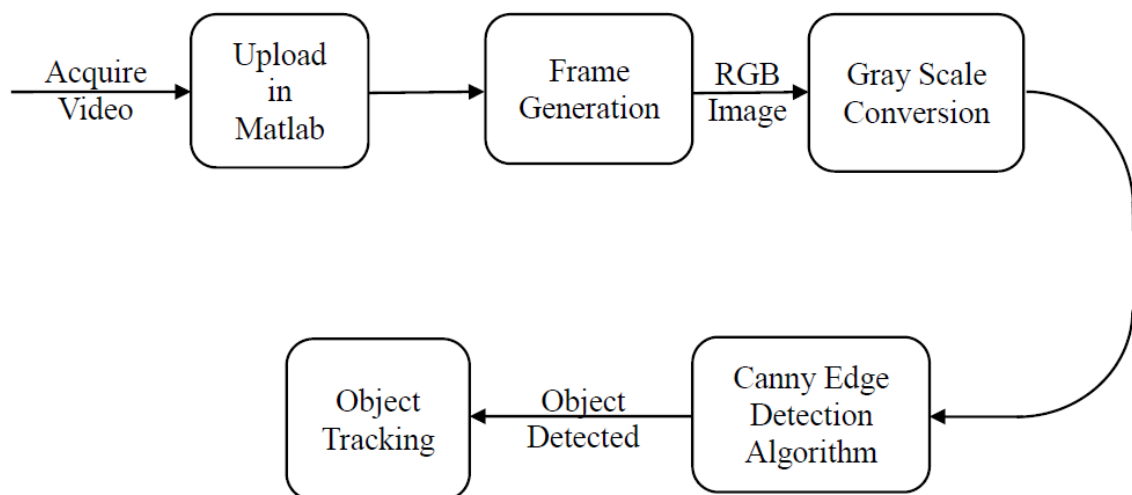


Figure 2.1: Flowchart representing steps in object detection and tracking

CHAPTER 3

CANNY EDGE DETECTION

3.1 Introduction

The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing. Several algorithms exist, and this report focuses on a particular one developed by John F. Canny (JFC) in 1986. Even though it is quite old, it has become one of the standard edge detection methods and it is still used in research. The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.

The aim of JFC was to develop an algorithm that is optimal with regards to the following criteria:

1. Detection: The probability of detecting real edge points should be maximized while the probability of falsely detecting non-edge points should be minimized. This corresponds to maximizing the signal-to-noise ratio.
2. Localization: The detected edges should be as close as possible to the real edges.
3. Number of responses: One real edge should not result in more than one detected edge

3.2 Test image

The image in Figure 3.1 is used throughout this report to demonstrate how Canny edge detection works.

The image has been pre-processed as described in the report “Ideas for Solution to the Pose Estimation Problem”

The pre-processing includes:

1. Determining ROI (Region of Interest) that includes only the object, and cropping the image to this region.
2. Conversion to gray-scale to limit the computational requirements.



Figure 3.1: Test image

3.3 Canny Edge Detection Algorithm

The algorithm runs in 5 separate steps:

1. **Smoothing:** Blurring of the image to remove noise.
2. **Finding gradients:** The edges should be marked where the gradients of the image has large magnitudes.
3. **Non-maximum suppression:** Only local maxima should be marked as edges.
4. **Double thresholding:** Potential edges are determined by thresholding.
5. **Edge tracking by hysteresis:** Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

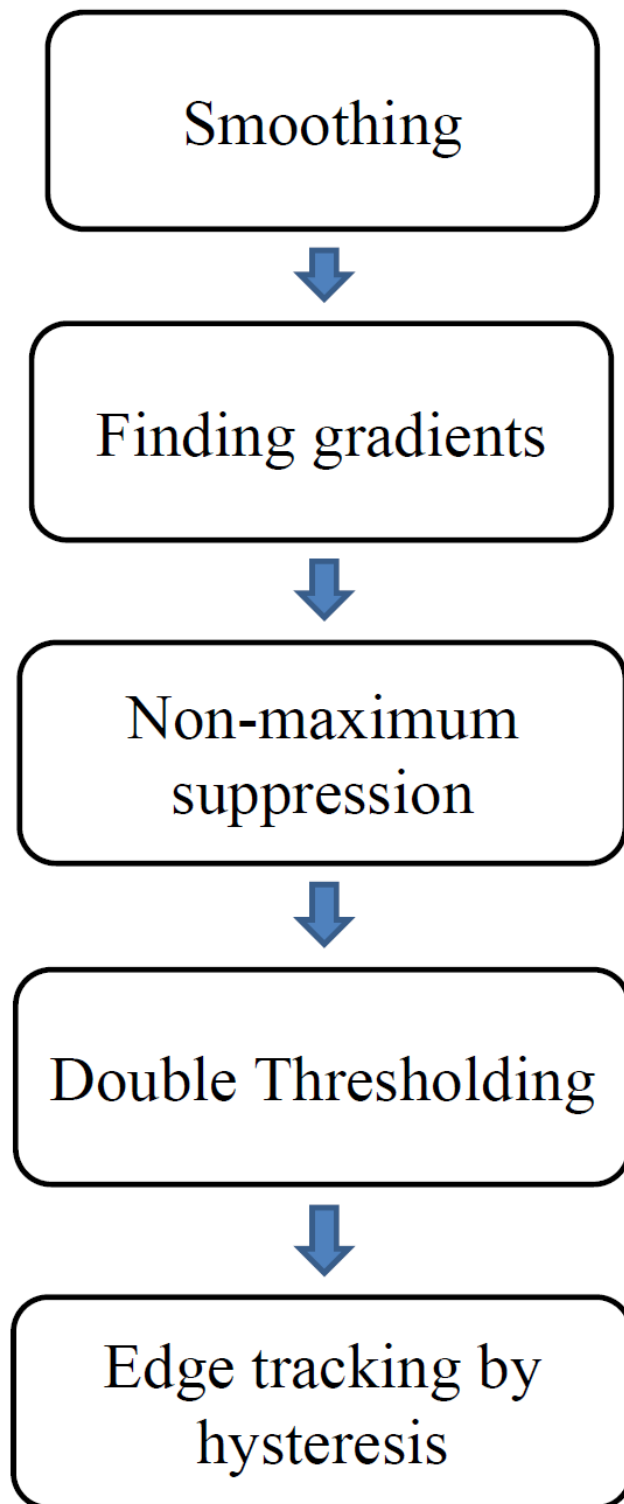


Figure 3.2: Flowchart of canny edge detection algorithm

3.3.1 Smoothing

It is inevitable that all images taken from a camera will contain some amount of noise. To prevent that noise is mistaken for edges, noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter. The kernel of a Gaussian filter with a standard deviation of $\sigma = 1.4$ is shown in Equation. The effect of smoothing the test image with this filter is shown in Figure 3.3.

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

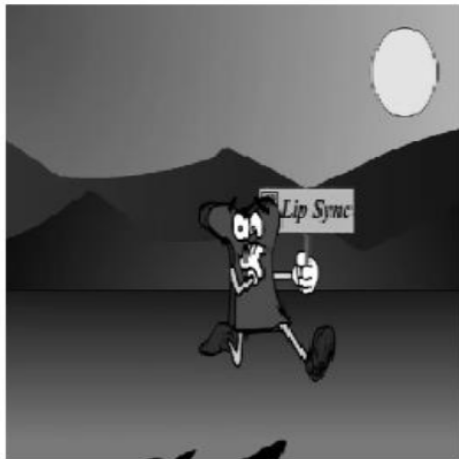


Figure 3.3(a): Initial Image

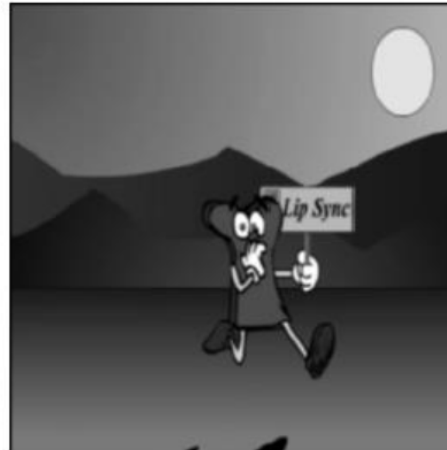


Figure 3.3(b): Smoothed image

Gaussian filters

One – dimensional Gaussian:

$$G_1(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

Two – dimensional Gaussian:

$$G_2(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

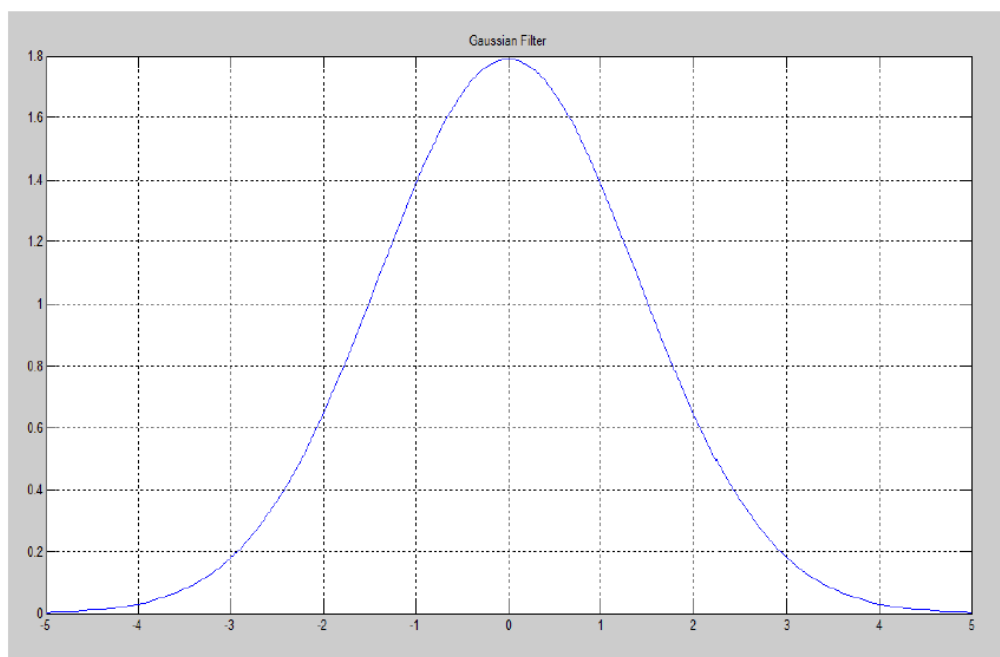


Figure 3.4: Gaussian Filter in One-Dimension

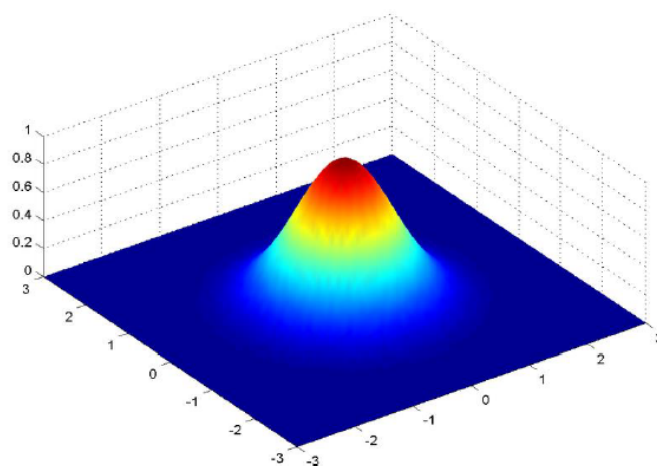


Figure 3.5: Gaussian Filter in Two-Dimension

3.3.2 Finding Gradients

The Canny algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. Gradients at each pixel in the smoothed image are determined by applying what is known as the Sobel-operator. First step is to approximate the gradient in the x- and y-direction respectively by applying the kernels shown in equations given below:

Sobel mask in x-direction:

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

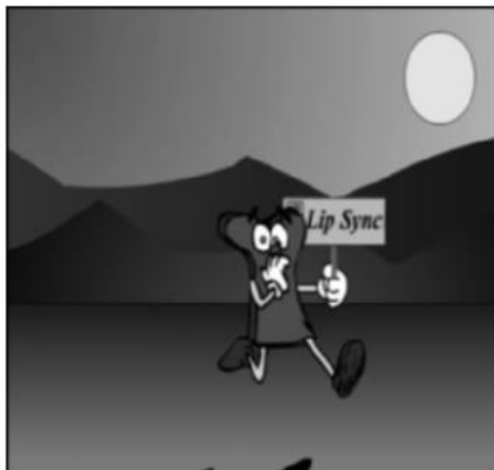


Figure 3.6(a): Smoothed image



Figure 3.6(b): Sobel operator in X-dir

Sobel mask in y-direction:

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{bmatrix}$$



Figure 3.7(a): Smoothed image



Figure 3.7(b): Sobel operator in Y-dir

Combined Image

The gradient magnitudes (also known as the edge strengths) can then be determined as an Euclidean distance measure by applying the law of Pythagoras as shown in Equation. It is sometimes simplified by applying Manhattan distance measure as shown in Equation to reduce the computational complexity. The Euclidean distance measure has been applied to the test image. The computed edge strengths is shown in figure 3.8

$$|G| = \sqrt{G_x^2 + G_y^2}$$

$$|G| = |G_x| + |G_y|$$

Where: G_x and G_y are the gradients in the x- and y-directions respectively.

It is obvious from Figure 3.8, that an image of the gradient magnitudes often indicate the edges quite clearly. However, the edges are typically broad and thus do not indicate exactly where the edges are. To make it possible to determine this, the direction of the edges must be determined and stored as shown in Equation

$$\theta = \arctan\left(\frac{|G_x|}{|G_y|}\right)$$



Figure 3.8: Image with Gradient Magnitudes

3.3.3 Non-Maximum Separation

Non-maximum suppression is an edge thinning technique.

The purpose of this step is to convert the “blurred” edges in the image of the gradient magnitudes to “sharp” edges.

Given estimates of the image gradients, a search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. In many implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbours in the gradient direction.

Basically this is done by preserving all local maxima in the gradient image, and deleting everything else. The algorithm is for each pixel in the gradient image:

1. Round the gradient direction to nearest 45° , corresponding to the use of an 8-connected neighbourhood.
2. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction. I.e. if the gradient direction is north ($\theta = 90^\circ$), compare with the pixels to the north and south.

3. If the edge strength of the current pixel is largest; preserve the value of the edge strength.

If not, suppress (i.e. remove) the value.

For example,

- if the rounded gradient angle is zero degrees (i.e. the gradient is in the north-south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **east and west** directions,
- if the rounded gradient angle is 90 degrees (i.e. the gradient is in the east-west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north and south** directions,
- if the rounded gradient angle is 135 degrees (i.e. the gradient is in the north east-south west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north west and south east** directions,
- if the rounded gradient angle is 45 degrees (i.e. the gradient is in the north west-south east direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north east and south west** directions.

Note that the sign of the direction is irrelevant, i.e. north-south is the same as south-north and so on.

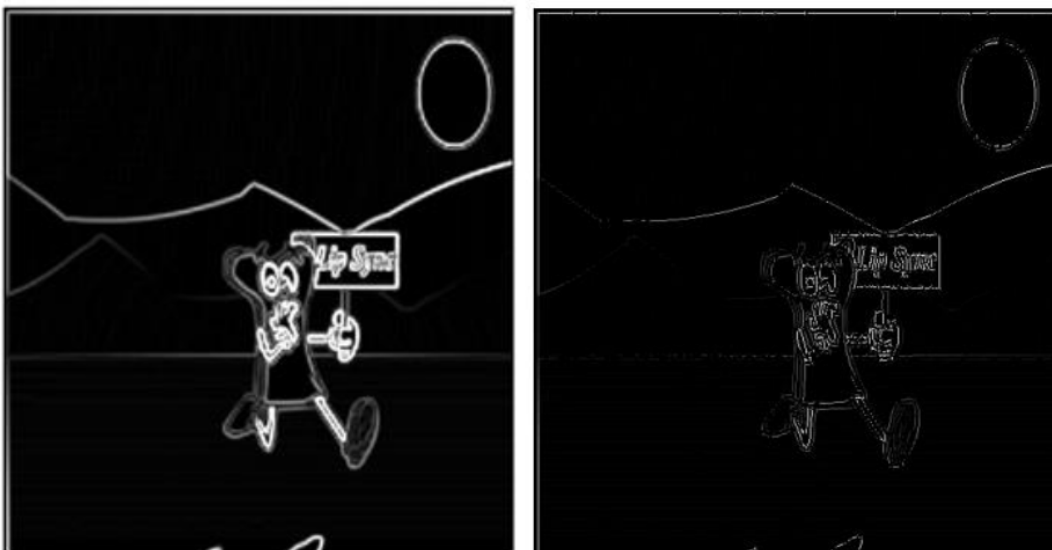


Figure 3.9(a): Gradient Magnitudes

Figure 3.9(b): Non-maximum separation

3.3.4 Tracing edges through the image and hysteresis

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

Thresholding with hysteresis requires two thresholds – high and low. Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients. Therefore we begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

Once this process is complete we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. From complementary output from the edge tracing step, the binary edge map obtained in this way can also be treated as a set of edge curves, which after further processing can be represented as polygons in the image domain.

3.3.4.1 Double Thresholding

The edge-pixels remaining after the non-maximum suppression step are (still) marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but some may be caused by noise or color variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger than a certain value would be preserved. The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak. The effect on the test image with thresholds of 0.05 and 0.15 upper and the lower threshold values respectively and is shown in the image below.

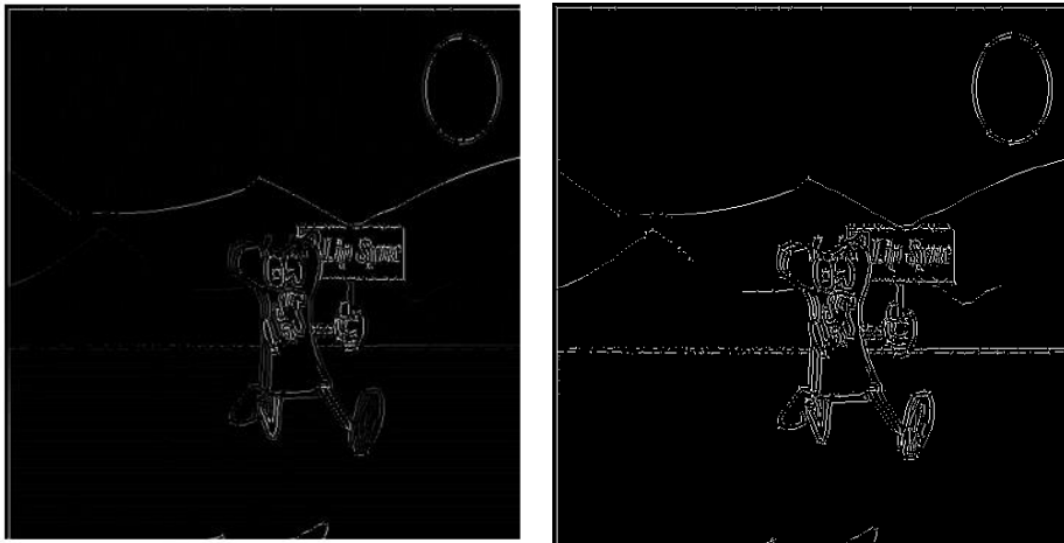


Figure 3.10(a): Non-maximum separation Figure 3.10(b): Double thresholding

3.3.4.2 Edge Hysteresis

Strong edges are interpreted as “certain edges”, and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/color variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges.

Edge tracking can be implemented by BLOB-analysis (Binary Large Object). The edge pixels are divided into connected BLOB’s using 8-connected neighbourhood. BLOB’s containing at least one strong edge pixel are then preserved, while other BLOB’s are suppressed. The effect of edge tracking on the test image is shown in the below image.

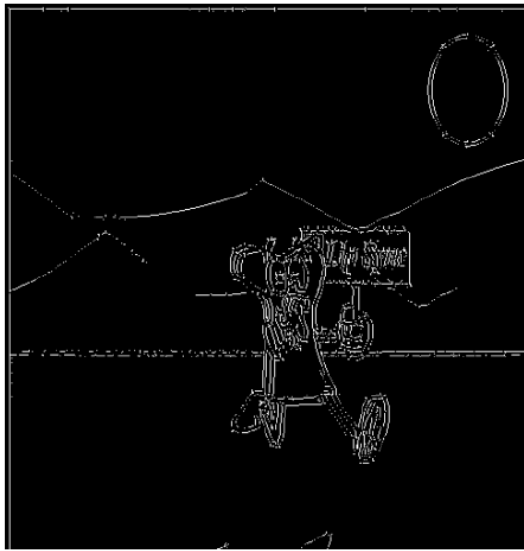


Figure 3.11(a): Double thresholding

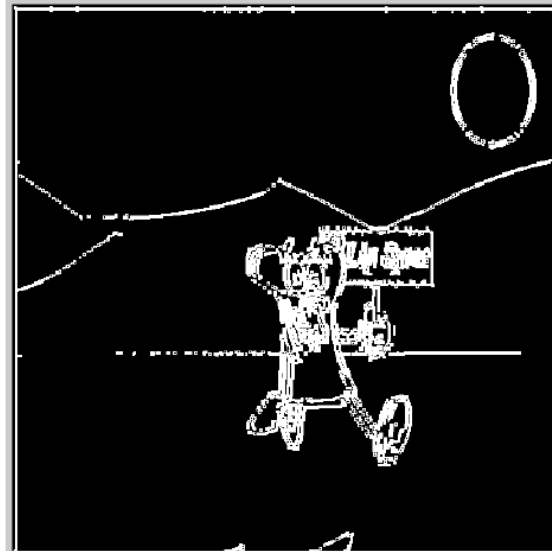


Figure 3.11(b): Edge hysteresis

3.4 Implementation

Figure 3.12 shows a frame with a background only (frame 30 in our video) and another frame having an object also (frame 60). These image are in RGB format.

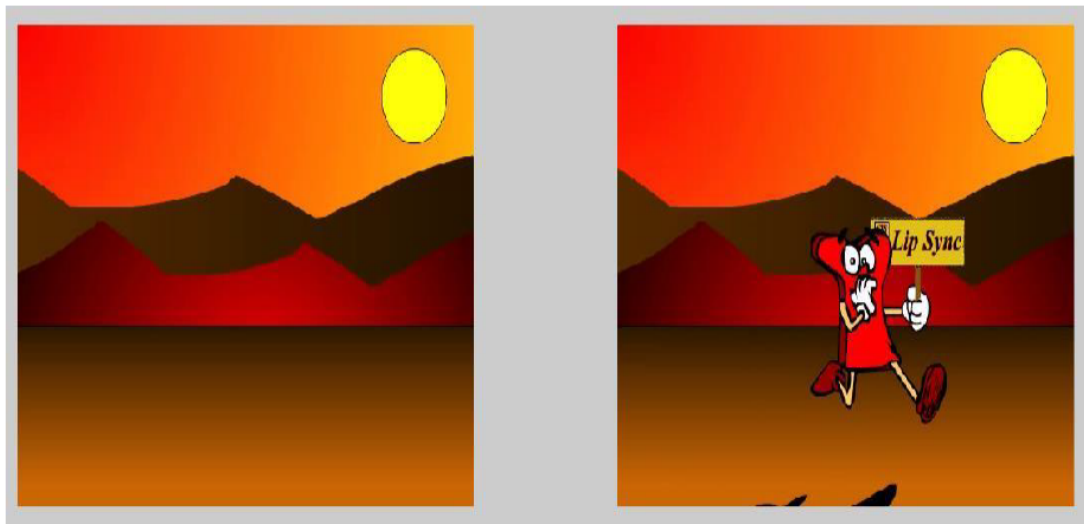


Figure 3.12

Figure 3.13 shows the canny edge detection algorithm applied to the above images separately. In first image the edges of sun, mountain and land have been detected. In the second image in addition to these, the edges of the object have also been detected.

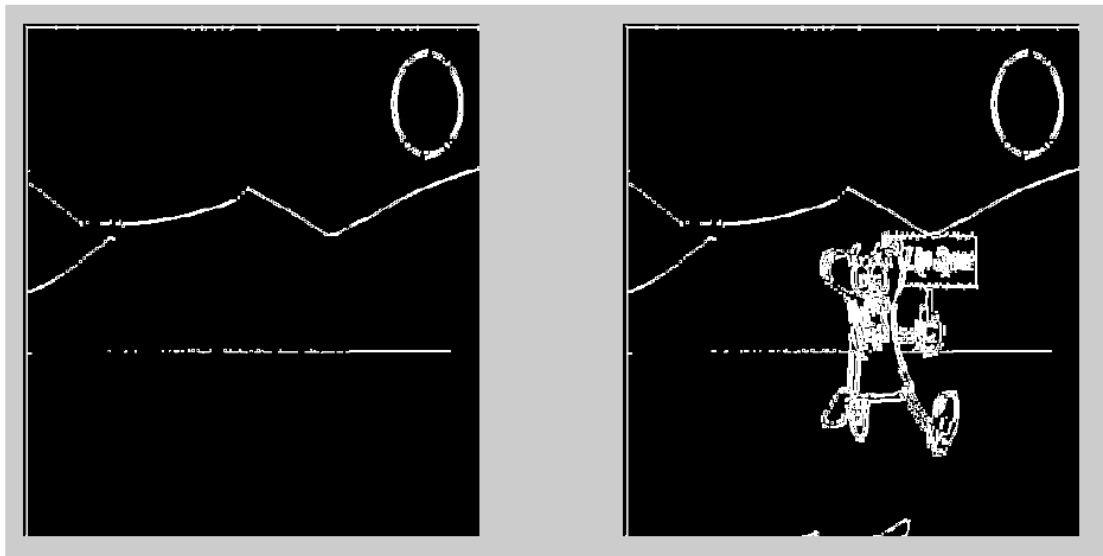


Figure 3.13

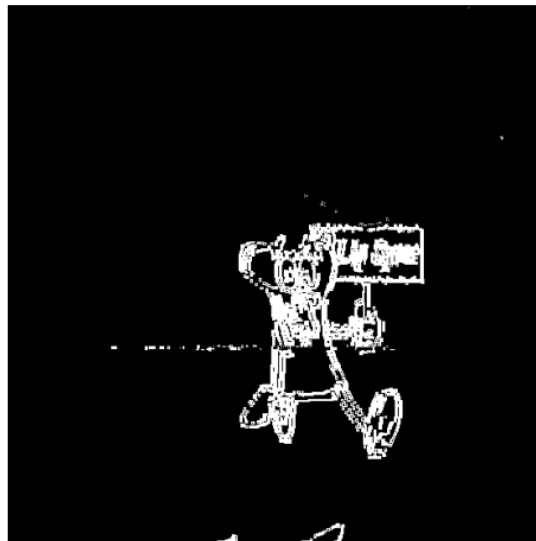


Figure 3.14

This figure shows the output of object detection by subtracting the images of figure 3.13 i.e. the background frame image (frame 30) is subtracted from the image having an object (frame 60).

CHAPTER 4

OBJECT TRACKING

4.1 Introduction

Object tracking consists of estimation of trajectory of moving objects in the sequence of frames generated from a video.

The aim of motion tracking is to detect and track moving objects through a sequence of images. Motion tracking is not only useful for monitoring activity in public places, but is becoming a key ingredient for further analysis of video sequences. Information about the location and identity of objects at different points in time for instance is the basis of detecting unusual object movements (e.g. someone being mugged at an ATM) or coordinated activities (e.g. strategic plays in a football game).

The implementation of object tracking system is based on following key concepts:

1. Detection of interested moving objects in a frame. Such objects are tracked from frame to frame.
2. Analysis of object tracks to recognize their behaviour and trajectory of object can be estimated.

Tracking of object (An Overview): The basic operation is to track the only interesting moving object irrespective of other moving objects. The object tracker module is used for this purpose, which keep track of interesting objects over time by locating the position of moving object in every frame of the video. The algorithm which we have implemented has flexibility to perform both task, object detection and to track object instances across frames simultaneously.

First of all tracking module will generate a template for all received centroid information and this template is used for the matching in next upcoming frame. Next, we match the template in next upcoming frame and calculate the centroid of matched area (i.e. the detected object) from the continuous video stream. Stream is appended to make an estimation of trajectory and meanwhile the template is updated with new matched region. The coupling of the template matching with the frame differencing

gives the good result for Object Tracking System (i.e. the same process or algorithm implemented in a better way).

Algorithm:

1. The video entered in the Matlab is divided into number of frames.
2. Each instant of frame containing object is subtracted from the background frame to obtain the object.
3. The centroid is calculated and represented by the 'v' shape in the image.
4. The path traced is successfully plotted by joining the centroid points as is shown in the upcoming results.
5. Trajectory is plotted whenever it is demanded.

The limitation with this tracking module is that all the centroid information received for tracking of objects should always be in camera view.

4.2 Implementation

In the first step of our implementation, the video is divided into a number of frames and centroid is calculated at each instant i.e. of each frame and we represent the centroid by 'v' shape as shown in figure 4.1.

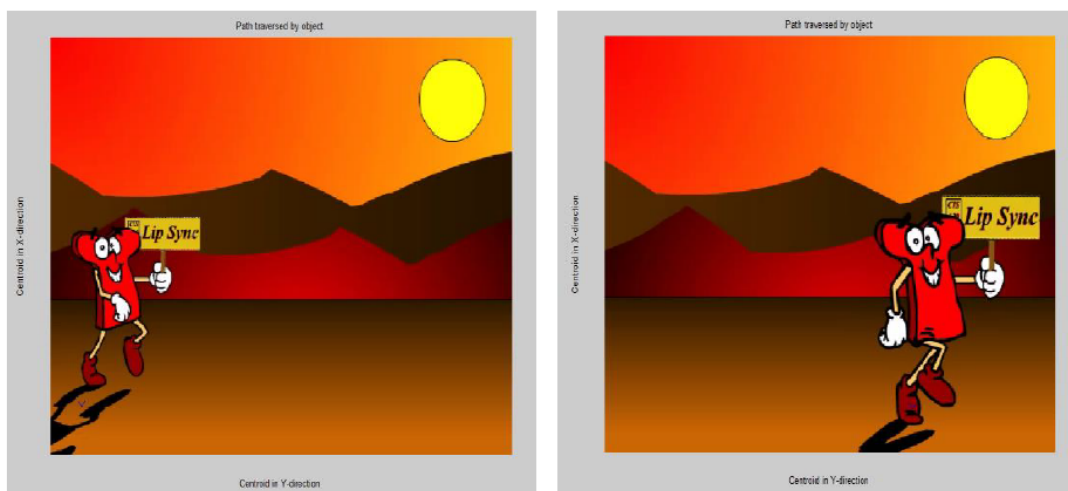


Figure 4.1: Image showing centroid

In the second step of our implementation, the centroid calculated in the previous step i.e. the centroid of detected object are taken into consideration and is plotted as shown in figure 4.2 by taking the coordinates in the x and the y direction.

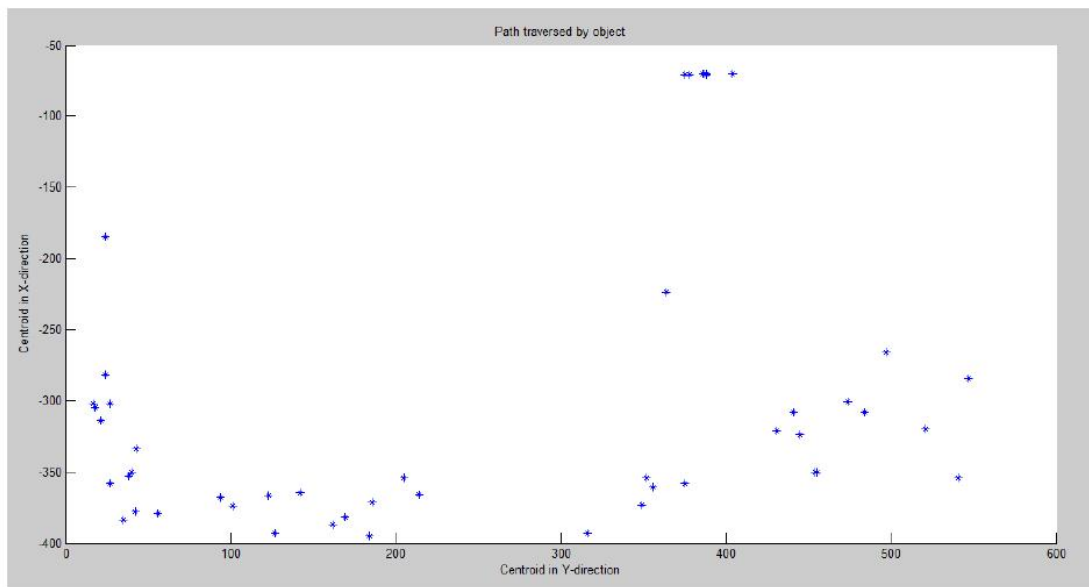


Figure 4.2: Path traced by Moving Man

In the next step of our implementation, the same process is repeated as is described in the previous two steps by taking a different example of an ant (in this case).

Figure 4.3 shows the centroid of the detected object at various instants which is represented by a 'v' shape.

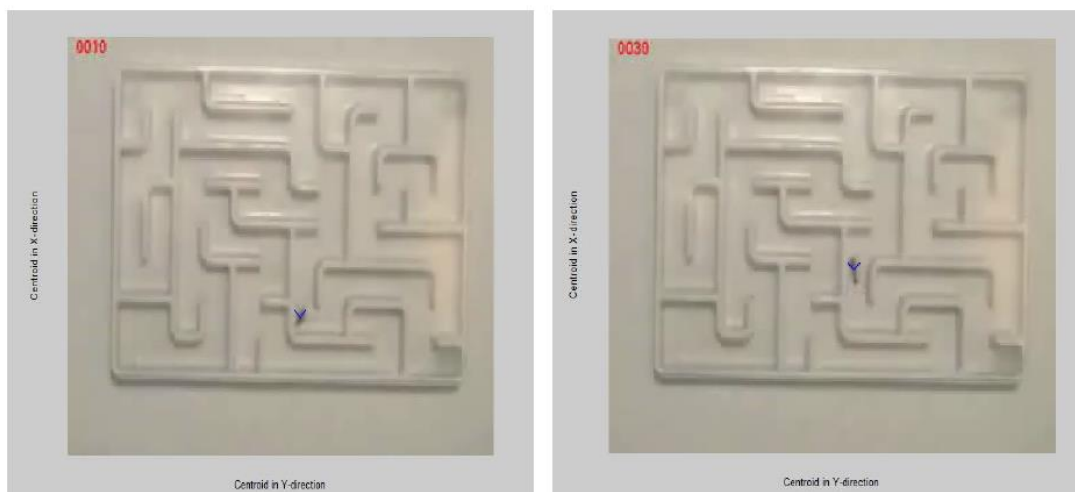


Figure 4.3: Image showing Centroid (Another example)

Figure 4.4 shows the path traced by an ant by calculating the centroids as done before. The path traced of the ant is with respect to the video which we have taken as an input.

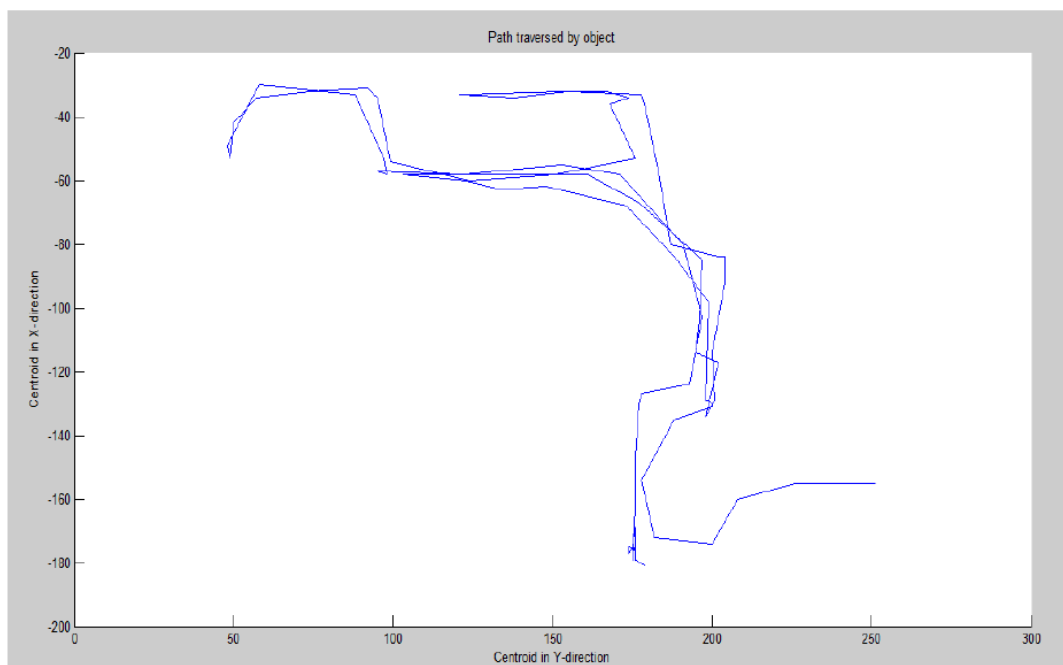


Figure 4.4: Path Traced by Ant

4.3 Problems

Tracking objects can be complex due to:

1. Loss of information caused by projection of the 3D world on a 2D image.
2. Noise in images.
3. Complex object motion.
4. Partial and full object occlusions, complex object shapes.
5. Scene illumination changes.
6. Shadows of moving object.

CONCLUSION

From this project we have concluded that the object can be detected efficiently by the use of the Canny edge detection algorithm and the path is successfully tracked by calculating the centroid of the object detected at each step.

The Canny edge detector is a very popular and effective edge feature detector. The edges obtained from this process are sharp as compared to various other techniques and the object detected using the Canny edge detection algorithm has much more SNR (signal to noise ratio) as compared to the object obtained by the subtraction algorithm. Information about the location and identity of objects at different points in time for instance is the basis of detecting unusual object movements (e.g. someone being mugged at an ATM). In our case the path of the moving man and the ant are successfully tracked.

REFERENCES

1. Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, 3rd ed., Pearson Prentice Hall, India, 2009.
2. G. Shrikanth and Kaushik Subramanian, "Implementation of FPGA based Object Tracking Algorithm", April 2008
3. Mathew George, C. Lakshmi, "Object Detection using the Canny Edge Detector", International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064, May 2013
4. Jyotsna Patil, Sunita Jadhav, "A Comparative Study of Image Denoising Techniques" International Journal of Innovative Research in Science, Engineering and Technology *Vol. 2, Issue 3, March 2013*, March 2013.
5. Zhong Guo, "Object Detection and Tracking Video", November 2001
6. John Canny. "A computational approach to edge detection. Pattern Analysis and Machine Intelligence", IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986