

OBJECT TRACKING IN STEREO VIDEO FRAME SEQUENCES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

SERKAN KEFEL

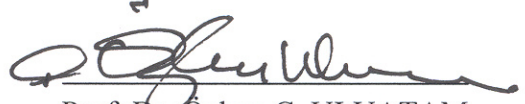
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE, 2008

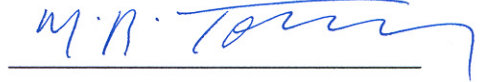
Title of the Thesis : **Object Tracking in Stereo Video Frame Sequences**

Submitted by **Serkan Kefel**


Approval of the Graduate School of Natural and Applied Sciences, Çankaya University


Prof. Dr. Özhan Ç. ULUATAM
Acting Director

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

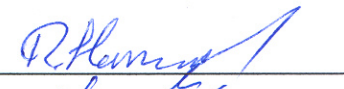

Prof. Dr. Mehmet R. TOLUN
Head of Department

This is to 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 Master of Science.

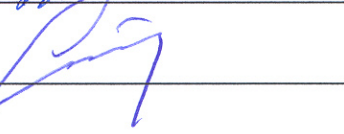

Asst. Prof. Dr. Abdül Kadir GÖRÜR
Supervisor

Examination Date : June 20, 2008

Examining Committee Members

Asst. Prof. Dr. Reza HASSANPOUR(Çankaya Univ.) 

Asst. Prof. Dr. Abdül Kadir GÖRÜR (Çankaya Univ.) 

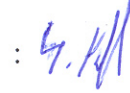
Dr. Ersin ELBAŞI (TUBITAK) 

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Serkan Kefel

Signature

: 

Date

: 20.06.2008

ABSTRACT

OBJECT TRACKING IN STEREO VIDEO FRAME SEQUENCES

KEFEL, Serkan

M.S.c., Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Abdül Kadir GÖRÜR

June 2008, 53 pages

Tracking in video refers to the process of locating moving objects in the following video frames. It is an important topic and has various application fields such as robotics, military, etc. In this thesis, moving object tracking in stereo video sequences is studied. Tracking is done by several processes. First, the object, that is going to be tracked, is detected, and extracted from its background. Then, a noise elimination process follows that, where we detect false candidates and eliminate them. After key features of the target object are gathered, we can track the object by searching these features in the following frame sequences.

Keywords: Stereo, Video, Object Tracking.

ÖZ

STEREO VİDEO KARELERİNDE NESNE İZLEME

KEFEL, Serkan

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi: Yard. Doç. Dr. Abdül Kadir GÖRÜR

Haziran 2008, 53 sayfa

Videoda izleme, nesnelerin takip eden karelerdeki konumlarının belirlenmesi işidir. Önemli bir konudur ve robotik, askeri vb. gibi değişik uygulama alanları mevcuttur. Bu tezde, hareket eden nesnelerin takip eden stereo video karelerinde izlenmesi çalışılmıştır. Takip etme, birkaç işlemden oluşur. Öncelikle takip edilecek nesne belirlenir ve arka planından ayrılır. Daha sonra görüntüde var olan gürültülerin ayıklanması işlemi yapılır. Bulunan anahtar kısımların gelen görüntü karelerinde bulunması ile nesne takip edilebilir.

Anahtar Kelimeler: Stereo, Video, Nesne İzleme.

ACKNOWLEDGEMENTS

I thank to my family for all their supports and patience during this thesis.

I thank to my supervisor, Asst. Prof. Dr. Abdül Kadir GÖRÜR and my co-supervisor, Asst. Prof. Dr. Reza HASSANPOUR for their guidance, advices, criticisms, encouragements and insights throughout the research.

I thank to the Management of Çankaya University for providing a peaceful and useful environment for completing the thesis.

I thank to my colleagues, Research Assistant S. Pelin GÜVENÇ and Research Assistant F. Serdar TAŞEL for their suggestions and comments.

TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTERS	
1. INTRODUCTION	1
1.1 Difficulties in Tracking	1
1.2 General Overview of the Thesis	3
2. BACKGROUND STUDY	4
2.1 Stereo Vision	4
2.2 Pinhole Camera	7
2.3 Pinhole Camera Model	7
2.4 Epipolar Geometry	10
3. ALGORITHMS AND METHODS USED IN THE THESIS	12
3.1 Algorithms and Methods	12
3.2 Results and Future Work	51
4. CONCLUSION	53
REFERENCES	R1

LIST OF TABLES

TABLES

Table 3.1	44
Table 3.2	45
Table 3.3	46

LIST OF FIGURES

FIGURES

Figure 1.1	Stereo Object Tracking System.....	3
Figure 2.1	Stereo Vision of Human.....	5
Figure 2.2	Stereo Vision of Computers.....	6
Figure 2.3	Principle of a Pinhole Camera.....	7
Figure 2.4	Geometry of a Pinhole Camera.....	8
Figure 2.5	Geometry of a Pinhole Camera.....	9
Figure 2.6	Epipolar Geometry.....	11
Figure 3.1	Histogram's of four types of image.....	14
Figure 3.2	Histogram Equalization.....	16
Figure 3.3	Histogram Equalization.....	17
Figure 3.4	Disparity Map.....	18
Figure 3.5	Background Image.....	19
Figure 3.6	Difference Image.....	20
Figure 3.7	Edge Image.....	22
Figure 3.8	Thresholded Edge Image.....	24
Figure 3.9	Contour Image.....	26
Figure 3.10	Noises.....	26
Figure 3.11	Dilation Operation.....	28
Figure 3.12	Erosion Operation.....	29
Figure 3.13	Opening Operation.....	30
Figure 3.14	Blob Extraction.....	31
Figure 3.15	Blob Extraction.....	32
Figure 3.16	Blob Extraction.....	33
Figure 3.17	Convex Hull.....	34

Figure 3.18	Boundary.....	36
Figure 3.19	Boundary.....	39
Figure 3.20	YCbCr.....	41
Figure 3.21	False Segmentation.....	43
Figure 3.22	YCbCr Segmentation.....	47
Figure 3.23	Kalman Filter.....	48

CHAPTER 1

INTRODUCTION

Tracking can refer to several different meanings in literature. For instance, tracking in education means separating children into different classes according to their academic ability [1], whereas tracking in typography refers to the process of uniformly increasing or decreasing the space between all letters in a block of text [1]. In this thesis, tracking in video is studied.

The process of locating a moving object in sequential video frames is called video tracking. It has a lot of application areas in computer vision such as video compression, robot technology, video surveillance, object recognition, video segmentation and so on. Because of this wide application area, there exists many algorithms about video tracking. However, they are still not perfect due to the complexity of construction of vision.

1.1 Difficulties in Tracking

Regardless of the algorithm, getting pixels from the camera or video frames, one at a time, into an array is always the first step. That array is a list of number which includes the location, and the relative levels of red, green, and blue light at that location. We perform algorithms on that array, such as segmentation.

Segmentation is one of the major topics in image processing and computer vision. It refers to the process of partitioning a digital image into multiple regions. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. It is mostly used to locate objects and its boundaries in an image. There exists several algorithms and techniques that have been developed for image segmentation, however, there is no perfect algorithm for the image segmentation problem [1]. Segmentation is also used in video tracking.

One of the major problems in video tracking is that many of the algorithms are computationally expensive whether they are monocular or stereo. The reason behind that is they first perform a spatial segmentation based on gray-scale values in order to determine its constituent regions or objects, and boundaries [2][3]. When we look at a moving object, we can directly segment the scene and detect the object, extract it from its background, and follow it in time easily and we can do all these tasks in real-time. However, these tasks, especially the segmentation process, can take a lot of time in computers.

In order to lower this computational expensiveness, we can only segment the parts that are moving and omit the other parts, since we are interested in moving objects.

Another difficulty that we face in tracking is the visibility of the video. In order to extract the moving objects and their key features from video frames, the visibility of the video should be clear. For example, in outdoor videos, the environmental details such as fog, rain, and snow can make extraction and segmentation process difficult.

1.2 General Overview of the Thesis

We started the thesis by introducing the general aspects of object tracking and the difficulties that we can face in tracking.

The following chapters include brief explanations about the background study, methods and algorithms used, and experimental results and conclusion.

The following figure 1.1 illustrates the steps of “Object Tracking in Stereo Video Frame Sequences” system.

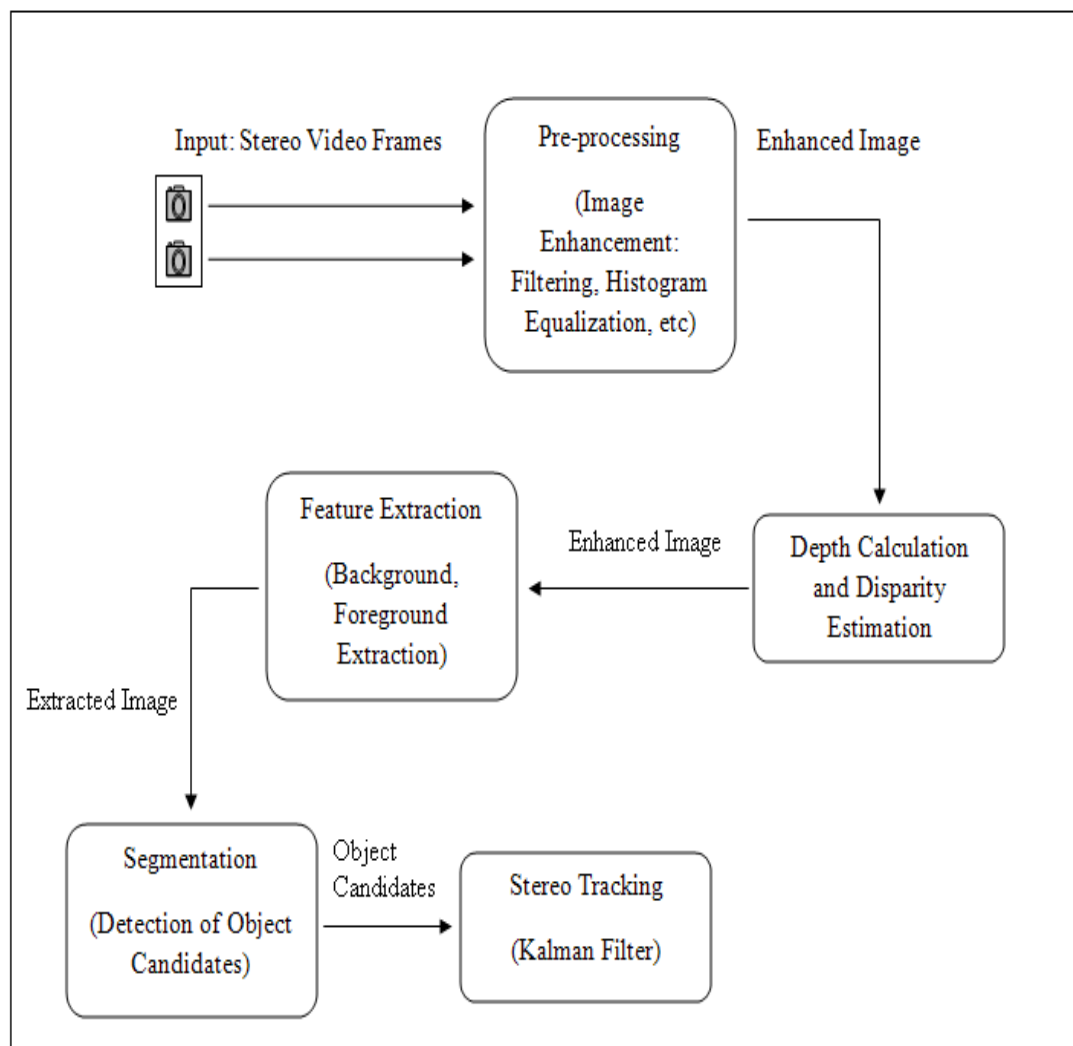


Figure 1.1: Stereo Object Tracking System

CHAPTER 2

BACKGROUND STUDY

In order to complete this thesis, we need to study some specific topics in computer graphics and image processing such as basic geometrical material that is essential for stereo and multiview image analysis as well as the basics of a camera. The following chapter contains brief information about these topics which are used in this thesis.

2.1 Stereo Vision

Every human being has two eyes which are located side by side in front of their head. Each eye takes a view of the same area from a different perspective due to the distance between them. The two views are almost same, but each eye picks up visual information that the other eye doesn't. That different visual information comes from our two eyes lead to slight relative displacement of objects (disparities) in the two monocular view of scene.

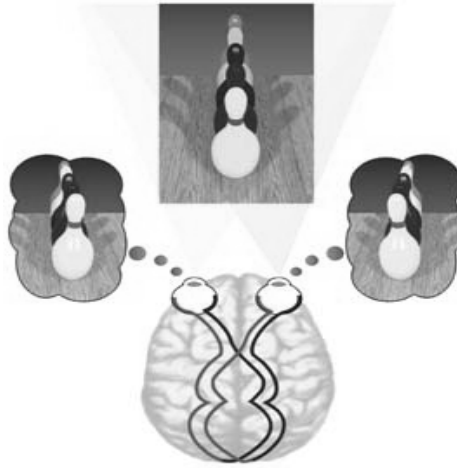


Figure 2.1: Stereo Vision of Human

In Figure 2.1, we can see the vision of human, and views of human eyes from two different perspectives. The two separate images that are captured from each eye are sent to the brain for processing. The brain combines the two images by matching the similarities between them and also adding the different information's coming from each view into a single image, where those differences makes the image a stereo, in other words three-dimensional image.

The word “stereo” comes from the Greek word “stereos”, which means solid or firm. With stereo vision, depth information is added and we can see objects as solid in three spatial dimensions, width, height, and depth, which makes the vision so rich and special. We can see where objects are in relation to our bodies with much greater precision especially when those objects are moving toward or away from us in the depth dimension.

In computer vision, stereo vision aims at reproducing the human ability to gather information on the three-dimensional structure and distances. The situation is pretty same as in the computer vision, where two cameras take the job of two eye. Two

cameras take the picture of the same scene, but again from a different perspective, exactly like our eyes. The two images that are captured from each camera are sent to a computer. The computer compares the images while shifting the images together over top of each other to find the parts that match, where the shifted amount is called disparity. The disparity, at which objects in the image best matches, is used by the computer to calculate their distance. Human eyes change their position according to the distance to the observed object. But for a computer, this task requires some complex geometrical calculations in Epipolar Geometry. Figure 2.1.2 shows the stereo vision of computers.

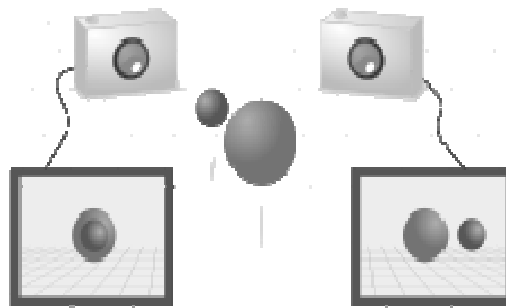


Figure 2.2: Stereo Vision of Computers

Stereo vision has long been researched, and several different algorithms were generated. Stereo vision is mostly used in robotics or unmanned vehicles to measure the distance between the object and the field of view, for purposes of path planning and obstacle avoidance.

The main advantage of the stereo vision is the depth and distance information that we gather from images.

2.2 Pinhole Camera

A camera without a glass lens is called Pinhole Camera. It has a very small hole on a thin material. The light rays from an object pass through that hole to form an image as described in the Figure 2.3.

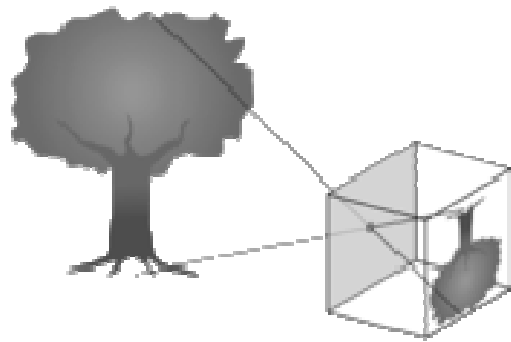


Figure 2.3: Principle of a Pinhole Camera

2.3 Pinhole Camera Model

In order to understand Epipolar Geometry and Stereo Vision, first of all, Pinhole Camera Model has to be learned, which describes the mathematical relationship between the coordinates of a three-dimensional point and its projection onto the two-dimensional image plane of a pinhole camera.

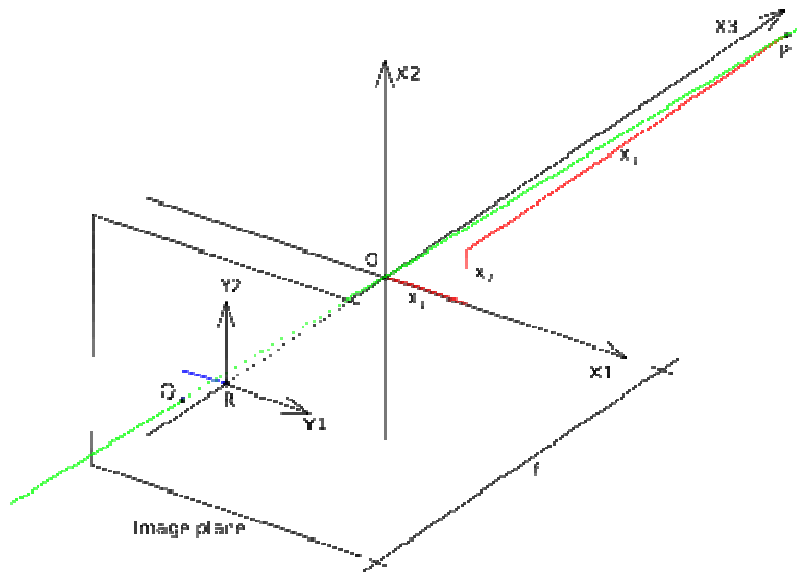


Figure 2.4: Geometry of a Pinhole Camera

Geometry related to the projection of a pinhole camera is shown in the figure 2.4. O represents the origin of a three-dimensional orthogonal coordinate system, and also the point where camera aperture is located [1]. $X1$, $X2$, and $X3$ refer to the three axes of the coordinate system. $X3$ axis is pointing in the viewing direction of the camera, which is called the *optical axis*, *principal axis*, or *principal ray* and the three-dimensional plane which intersects with the axes $X1$ and $X2$ is the front side of the camera, which is called *focal plane* or *principal plane* [1]. An image plane, which is parallel to the $X1$ and $X2$ axes and is located at the distance f , which is referred to as the *focal length*, from the origin O in the negative direction of $X3$ axis, exists where the three-dimensional world is projected through the aperture of the camera. The point R exists at the intersection of the optical axis and the image plane, which is referred to as the *principal point* or *image center* [1]. Point P shows a point at coordinates $x1$, $x2$, and $x3$ relative to the axes $X1$, $X2$, and $X3$. The green line shows the *projection line* of point P , which passes through the point P , the origin O , and the

point Q with coordinates y_1 and y_2 , where point Q is the projection of point P onto the image plane and is found by the intersection of the green line and the image plane [1]. The image plane is shown by the image plane is shown by the two-dimensional coordinate system with axes $Y1$ and $Y2$ with the origin R .

As we mentioned in the part 2.2, the aperture of the camera, through which all projection lines must pass, is assumed to be infinitely small. This point is referred to as *optical center*, *lens center*, *camera center*, *focus*, or *camera focal point* in literature [1].

The coordinate y_1 and y_2 of point Q is dependent on the coordinates x_1 , x_2 , and x_3 of point P . The following figure 2.5 illustrates this dependency. The figure is same as the Figure 2.4, but it is looking down in the negative direction of X_2 axis.

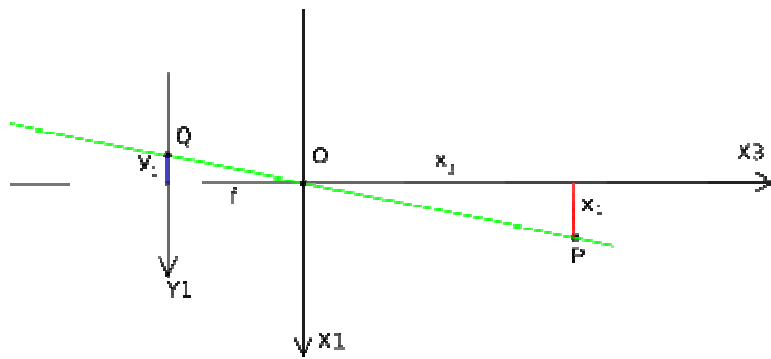


Figure 2.5: Geometry of a Pinhole Camera

By looking at the figure above, we can see two similar triangles where the green projection line form their hypotenuses. The following formulas can be derived according to the rules of similar triangles.

$$\frac{-y_1}{f} = \frac{x_1}{x_3} \quad (2.1)$$

and by looking in the negative direction of the X1 axis gives,

$$\frac{-y_2}{f} = \frac{x_2}{x_3} \quad (2.2)$$

These two formulas can be combined as,

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (2.3)$$

which describes the relation between the three-dimensional coordinates, x_1 , x_2 , and x_3 of point P and its image coordinates y_1 and y_2 of point Q in the image plane.

The mapping operation from three-dimension to two-dimension used in a pinhole camera is called *perspective projection*, and it is followed by a 180° rotation in the image plane [1]. The relative size of the projected objects depends on their distance to the focal point and the overall size of the image depends on the distance f between the image plane and the focal point.

2.4 Epipolar Geometry

The geometry of stereo vision is called epipolar geometry [1]. There are a number of geometric relations between the three-dimensional points and their projections onto the two-dimensional images when two cameras view a three-dimensional scene from two distinct points.

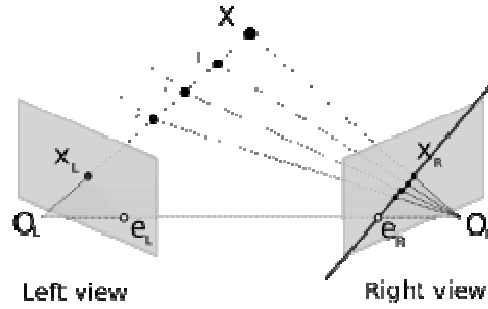


Figure 2.6: Epipolar Geometry

In figure 2.6, two pinhole cameras are looking at the point X , which is the point of interest for both cameras. O_L and O_R denotes the focal points of two cameras, and X_L and X_R denotes the projections of point X to the image planes of two cameras. Each camera captures a two-dimensional image of the three-dimensional world. The conversion from three-dimension to two-dimension is called *perspective projection* [1].

Because the two focal points of the cameras are distinct, each focal point is projected onto a distinct point into the other camera's image plane. These distinct points are called *epipoles* and they lie on the same single line with focal points O_L and O_R .

As seen from the figure 2.6, the line $O_L - X$ is seen as a point by the left camera because it is also in the same line with camera's focal point. But it is seen as a line in right camera's image plane. That line, $e_R - X_R$ in the image plane of right camera is called *epipolar line* [1]. The situation is same for the left camera, where $e_L - X_L$ is the epipolar line on left view.

The plane, that is formed by the points X , O_L , and O_R , is called *epipolar plane*. It intersects each camera's image plane at the points X_L , e_L , X_R , and e_R where it forms the epipolar lines [1].

CHAPTER 3

ALGORITHMS AND METHODS USED IN THE THESIS

The following chapter contains explanations of algorithms and methods used in the process of stereo tracking.

3.1 Algorithms and Methods

Before we begin to the tracking process, we have to do some operations on input video frames in order to simplify our job.

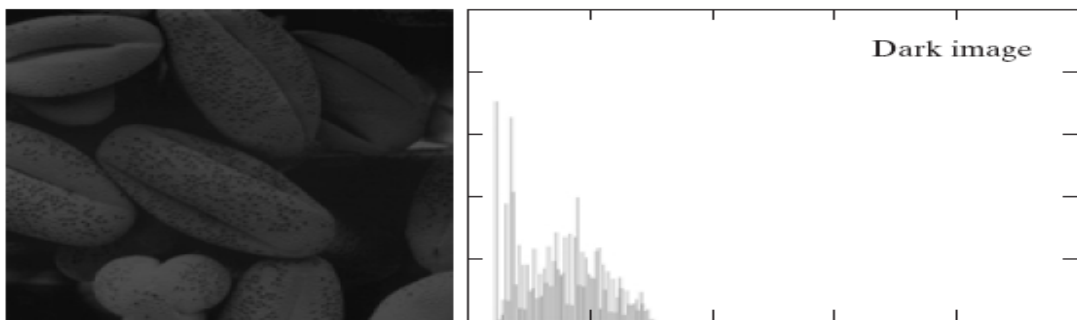
We mostly use gray-level color space in this thesis, so the first thing that we do is to convert the incoming frames which has a color space of RGB (RedGreenBlue) into gray-level. In order to do this, we use the following formula. Let Y denote the intensity of a gray-level image pixel, which is a number between 0 and 1, and R , G , and B are the Red, Green, and Blue amounts of the RGB image.

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \quad (3.1)$$

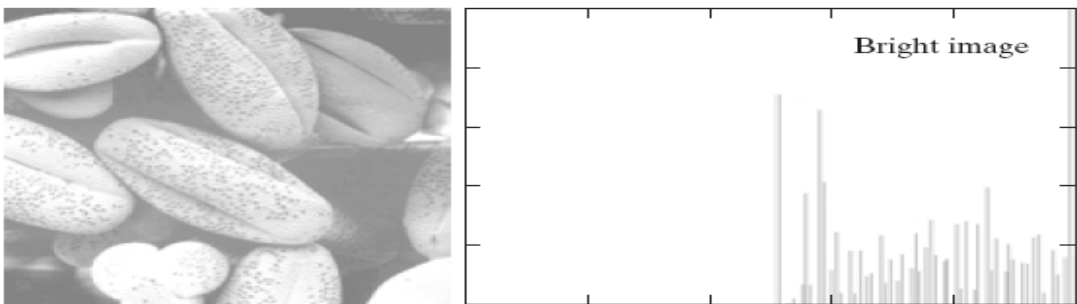
By using the formula above, we calculate the intensities of every single pixel in an image. Those intensity values give us the gray-level image.

After converting image from RGB color space to Gray, we need to adjust its contrast. In order to do this, we calculate the *histogram* of the image, and then apply *histogram equalization* [4].

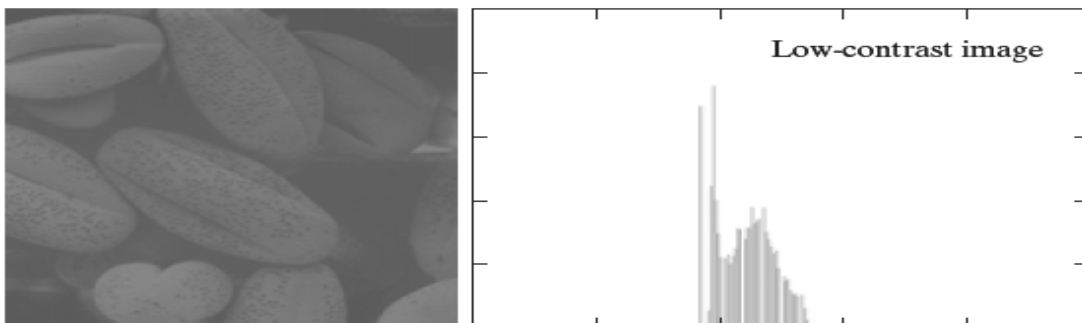
A histogram is the graphical version of a table that shows what proportion of cases fall into each of several or many specified categories. The histogram of a digital image with gray levels in the range of 0 and $L - 1$ is a discrete function $h(r_k) = n_k$, where r_k is the k th gray level and n_k is the number of pixels in the image having gray level r_k [11]. The following figure 3.1 illustrates four basic types of images; dark, light, low contrast and high contrast, and their corresponding histograms.



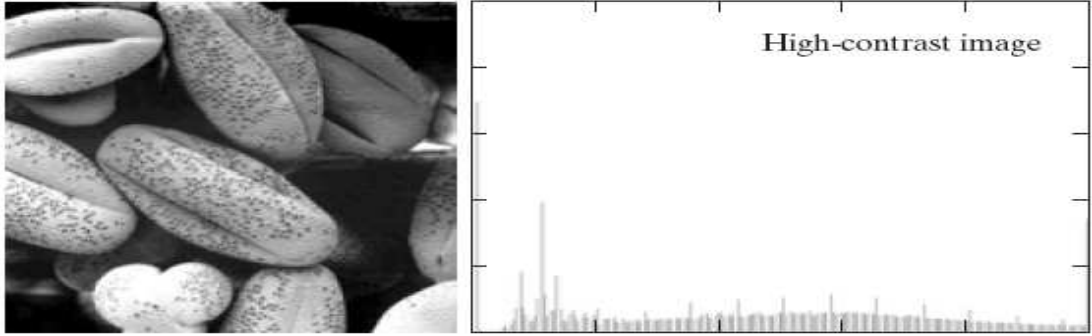
Part A of Figure 3.1.



Part B of Figure 3.1.



Part C of Figure 3.1.



Part D of Figure 3.1.

Figure 3.1: (Parts: A, B, C, D) Histogram's of four types of image.

Histogram equalization is the process of adjusting image contrast. It increases the local contrast of many images; especially the usable data of image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This method increases the contrast of areas that have lower local contrast to a higher contrast without affecting the global contrast by spreading out the most frequent intensity values. Consider a discrete gray-level image, and let n_i be the number of occurrences of gray-level i . The probability of an occurrence of a pixel of level i in the image is:

$$p(x_i) = \frac{n_i}{n}, i \in 0, \dots, L - 1 \quad (3.2)$$

L denotes the total number of gray-levels in the image, and n denotes the total number of pixels in the image, and p denotes the fact. Also, c is a cumulative distribution function corresponding to p , defined by:

$$c(i) = \sum_{j=0}^i p(x_j) \quad (3.3)$$

It is also known as the image's accumulated normalized histogram. The histogram equalization applies a transformation of the form $y = T(x)$ that will produce a level y

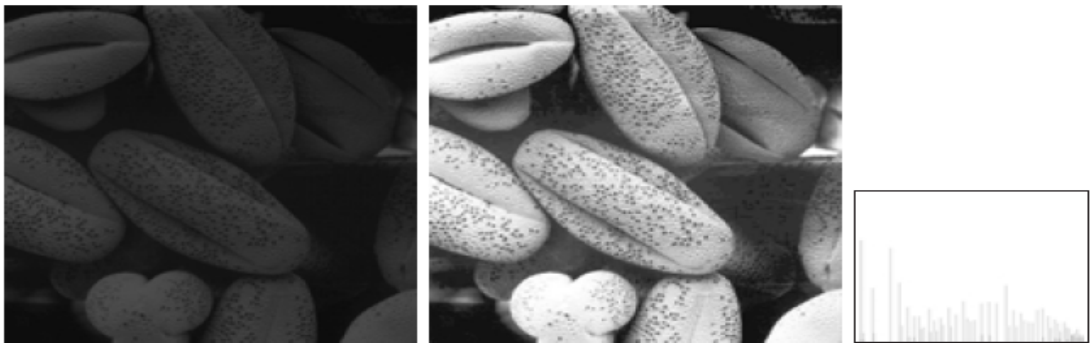
for each level x in the original image, such that the cumulative probability function of y will be linearized across the value range. The transformation is defined by:

$$y_i = T(x_i) = c(i) \quad (3.4)$$

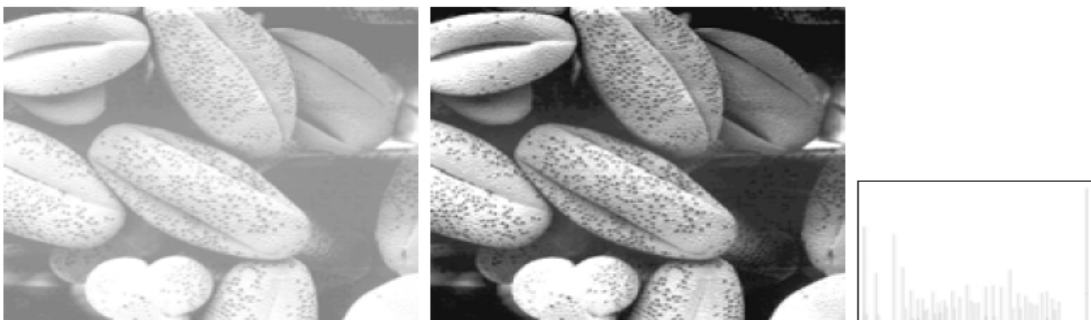
Since T has the value range of $0 - 1$, we have to map it to the original values by using the equation below:

$$y'_i = y_i \cdot (\max - \min) + \min \quad (3.5)$$

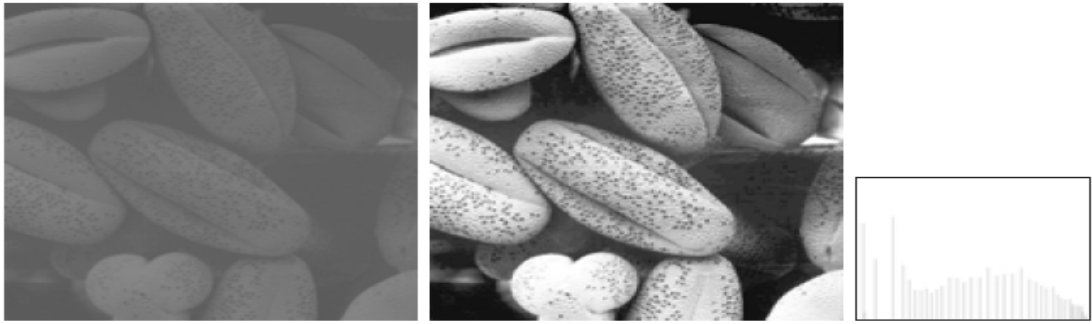
The following figure 3.2 illustrates the function of histogram equalization on the images in figure 3.1.



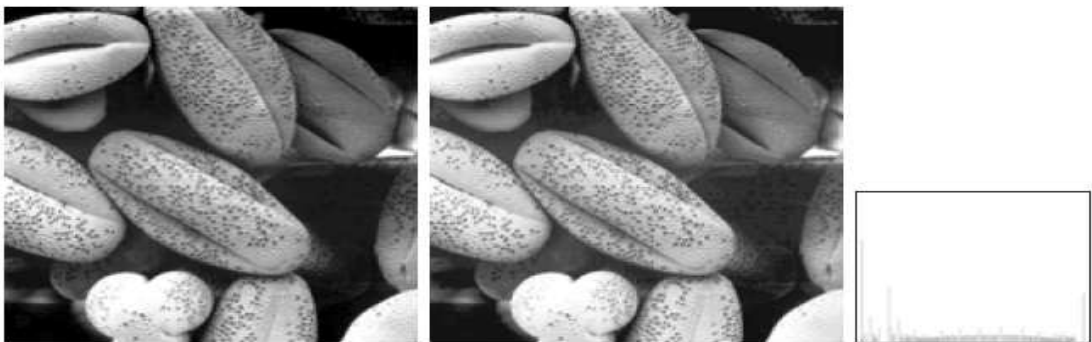
Part A of Figure 3.2.



Part B of Figure 3.2.



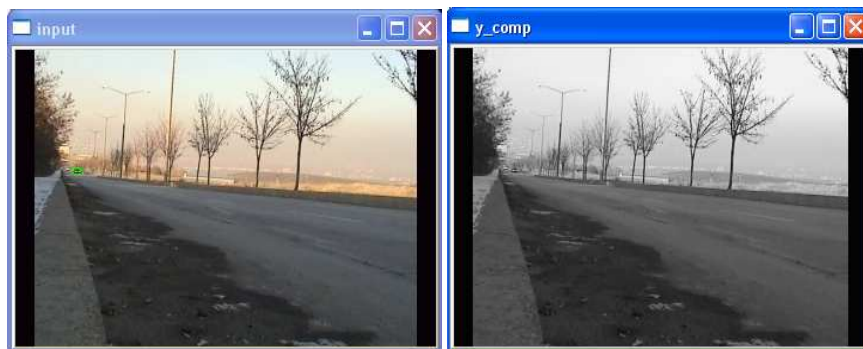
Part C of Figure 3.2.



Part D of Figure 3.2.

Figure 3.2: (Parts: A, B, C, D) Histogram Equalization.

The following figure 3.3 shows the function of histogram equalization on our sample data for tracking.



Part A and B of Figure 3.3.



Part C of Figure 3.3.

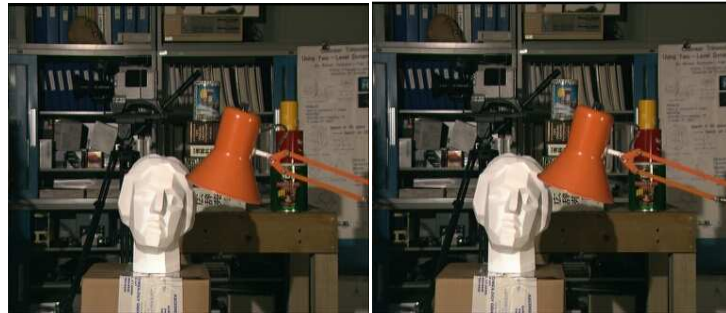
Figure 3.3: Upper-Left image (A) shows an input frame in RGB color space, Upper-Right image (B) is created after RGB to Gray conversion, and the bottom image (C) is histogram equalized form of gray image.

After enhancing the input frames, we need to calculate the depth information in order to distinguish object candidates from each other. To find depth information, we estimate disparities between left and right image of a stereo pair. In this thesis, a block-based matching algorithm is used as a disparity estimation technique. The algorithm is based on the similarities of the corresponding blocks between the left and the right images. In this method, one image, let's say left image, is chosen as reference image, and it is divided by a block unit, and then each block in this image undergoes the matching process with that of the other image based on a cost function, which is *minimum absolute difference (MAD)* for our study [5][6][7]. It simply compares corresponding pixels from each block, and calculates the sum of their difference. MAD is described by the following formulation:

$$\text{---} \tag{3.6}$$

A and B are the two blocks of size m and n in each of the stereo image pair. $A[p, q]$ is the value of the pixel in the p th row and q th column of block A, and $B[p, q]$ is the

value of the pixel in the p th row and q th column of block B. The lowest values correspond to the better matching results, so the block with the minimum MAD is chosen. The following figure 3.4 illustrates the extraction of disparity map from left and right images.



Part A and B of Figure 3.4.



Part C of Figure 3.4.

Figure 3.4: Upper-left image (A) is the left image of stereo image pairs, upper-right image (B) is the right image of stereo image pairs, and the bottom image (C) is the disparity map derived from the left and the right image.

After extracting the disparity map, we continue with the background and foreground extraction process. At the beginning of this step, we derive a still background image by simply taking average of a number of sequential frames using the equation below:

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y) \quad (3.7)$$

The figure 3.5 illustrates the function of image averaging in the test data that we used in this thesis.



Figure 3.5: Still Background derived by averaging 100 sequential frames.

Although we use image averaging in order to get the still background image, it is also used to remove noises in images or recover corrupted images.

The next step is the extraction of all moving pixels in the images. Since we tracking moving objects in a video, we don't have to struggle with the parts that are constant in every video frame, which will decrease the computational cost of later parts, and increase the speed of our algorithm. We accomplish this task by calculating the difference between every incoming video frame and the background that we derived. Simple formulation is as follows:

$$g(x, y) = f(x, y) - h(x, y) \quad (3.8)$$

where, g denotes the difference image, f denotes our extracted background and h denotes the incoming image frames.

The following figure 3.6 illustrates the difference operation in our test data.



Part A and B of Figure 3.6.



Part C of Figure 3.6.

Figure 3.6: Upper-left image (A) is the background image, upper-right image (B) is the input video frame, and bottom image (C) is the difference between background and the input frame.

After extracting all moving regions from incoming input frames, the noise elimination process begins. In this process, we try to eliminate fake candidates and derive the real object candidates from input frames.

First thing we do in this process is to retrieve the edges in the image. We accomplish this task by using sobel operator to the difference image that we calculated in the previous process. Sobel operator is used in image processing to detect the edges in an image. It is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. The Sobel operator is based on convolving the image with a small, separable, and integer valued mask or filter, and using this filter, it calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result of that operation shows us how abruptly or smoothly the image changes at that point, in other words, how likely it is that the part of the image represents an edge as well as how that edge is likely to be oriented. The operator uses two 3x3 kernels, one for horizontal changes and one for vertical changes, which are convolved with the original image to calculate derivatives.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \quad (3.9)$$

where, A is the source image, G_x and G_y are two images which contains horizontal and vertical derivations at each point respectively, and * operator denotes the two-dimensional convolution operation. We can calculate the gradient magnitude at each point by combining G_x and G_y as follows:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \quad (3.10)$$

We can also calculate the direction of the gradient using the equation below:

$$\Theta = \arctan \left(\frac{\mathbf{G}_y}{\mathbf{G}_x} \right) \quad (3.11)$$

where, for example, the result is 0 for vertical edges which is darker on the left side.

In the following figure 3.7 we can see the results of the Sobel operator application on our test data.



Part A and B of Figure 3.7.



Part C of Figure 3.7.

Figure 3.7: Upper-left image (A) shows the horizontal edges, upper-right image (B) shows vertical edges, and bottom image (C) shows combined edge image from horizontal and vertical edges.

As we can easily see from the figure 3.7, sobel operator finds object edges as well as a lot of other weak edges which we can call as noise. To remove those noisy

edges from the image, we use a binary thresholding. The thresholding algorithm simply specifies strong edges which has an intensity value greater than the threshold value defined before, and rounds them to an intensity value of 255, which is white, while removing all other edges which has an intensity value smaller than the threshold value by rounding them to an intensity value of 0, which is black.

$$P_i = \begin{cases} 0, & \text{if } P_i < \text{Threshold} \\ 255, & \text{if } P_i \geq \text{Threshold} \end{cases} \quad (3.12)$$

where, P_i is the intensity value of i th pixel.

The following figure 3.8 illustrates the application of the thresholding mentioned above on our test data.



Part A of Figure 3.8.



Part B of Figure 3.8.

Figure 3.8: Upper image (A) shows the application of sobel operator in images as mentioned in figure 3.7, and the bottom image (B) shows the application of thresholding on the sobel operator.

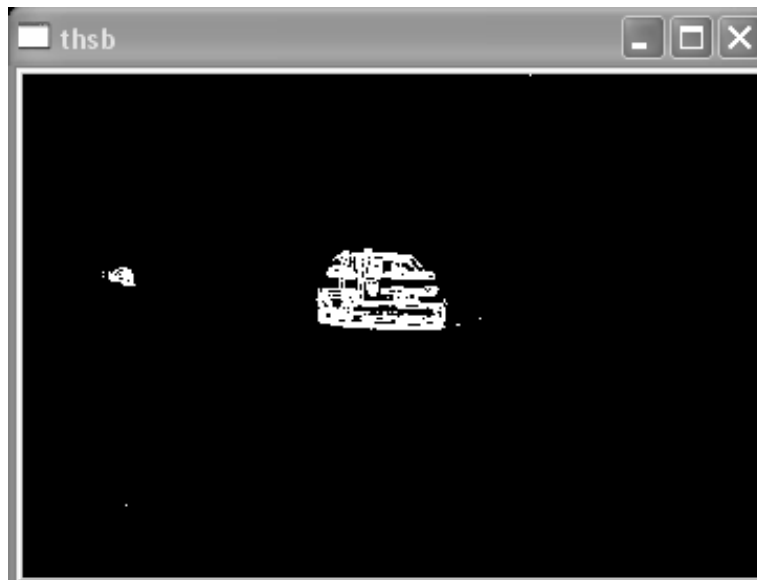
As we see from the above figure, the thresholding operation removes almost all the noisy edges from the image, and we have the edge pixels of the object candidate that is going to be tracked. However, edge detectors generally produce small, disjoint edge segments, which are mostly useless unless they aggregated into extended edges. So, in order to use these edge segments efficiently, some operations have to be done on them.

Contour extraction is an algorithm, which is designed to accomplish that task mentioned above. In literature, contour is a line connecting two points where they have similar properties. In image processing, contours are connected edges segments that are bounding a number of pixels [2]. By extracting contours on an image, we

find closed regions, which also will help us in segmentation, explained in later parts of this chapter.

In order to retrieve contours, two categories of methods are available; local methods, which extends edges by seeking the most compatible candidate edge in the neighborhood, but they suffer in big gaps, and global methods, which are successful in big gaps, but they are computationally expensive.

The figure 3.9 shows the extracted contours from the edges of figure 3.8. The extracted contours are filled by white pixels.



Part A of Figure 3.9.



Part B of Figure 3.9.

Figure 3.9: Upper image (A) is the thresholded sobel image as mentioned in figure 3.8, bottom image (B) shows the extracted and filled contours from the thresholded sobel image.

As we can see from the figure 3.9 above, some contours couldn't be extracted due to the huge gaps between edges, or undetected edges. There are also some salt like noisy pixels and pixel groups in the image after contour extraction process. Some of those noisy pixels are demonstrated in the following figure 3.10.

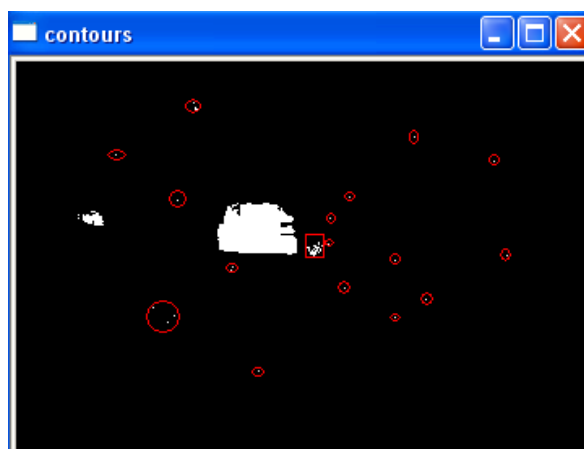


Figure 3.10: Noises

The figure 3.10 above shows the noisy pixels in a video frame after contour extraction process. Some of the noisy pixels are standing alone, as shown inside the red circles, and some of them are forming clusters as shown in the red rectangle. In order to remove those noisy pixels, we apply a morphological operation to image, called *opening* [4].

In image processing, opening is a morphological operation that generally smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions such as small islands and sharp peaks. The opening of a set A by structuring element B is denoted by $A \circ B$, and defined as:

$$A \circ B = (A \ominus B) \oplus B \quad (3.13)$$

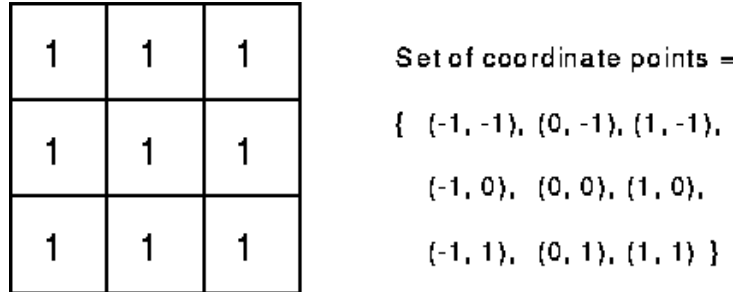
where, the \ominus symbol denotes the erosion operation and the \oplus symbol denotes the dilation operation. The opening operation A by B is the erosion of A by the structuring element B , followed by a dilation of the result by the structuring element B .

Dilation and erosion are two basic morphological operators in image processing. They are usually applied on binary images. The basic effect of dilation operation on a binary image is to gradually enlarge the boundaries of regions of foreground pixels, thus, areas of foreground pixels grow in size while holes within those regions become smaller. It uses two data pieces as inputs; the first one is the image that the dilation operation is going to be applied, and the second one is the structuring element or kernel that defines the effect of the operation on the input image. The following equation defines the dilation operation.

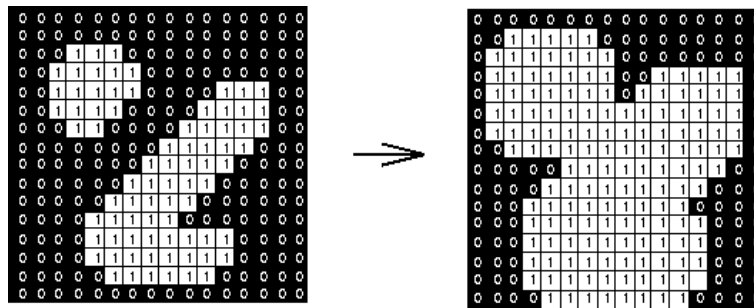
$$I \oplus E = \{z \mid [(\hat{E})_z \cap I] \subseteq I\} \quad (3.14)$$

where, I denotes the input image, and E denotes the structuring element.

The following figure 3.11 illustrates the dilation operation.



Part A of Figure 3.11.



Part B of Figure 3.11.

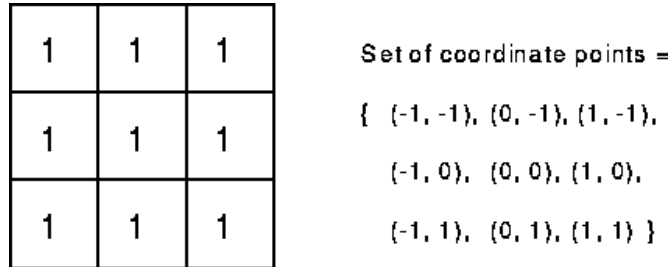
Figure 3.11: Upper image (A) is the 3x3 structuring element, bottom-left image is the input binary image, and bottom-right image (B) is the result of dilation operation on the input image using the given structuring element.

Erosion is the dual of dilation operation. The basic effect of erosion operation on a binary image is to erode away the boundaries of regions of foreground pixels, thus, areas of foreground pixels shrink in size while the holes within those areas become larger [4]. It uses two data pieces as inputs as well; the first one is the image that the erosion operation is going to be applied, and the second one is the structuring element. The following equation defines the erosion operation.

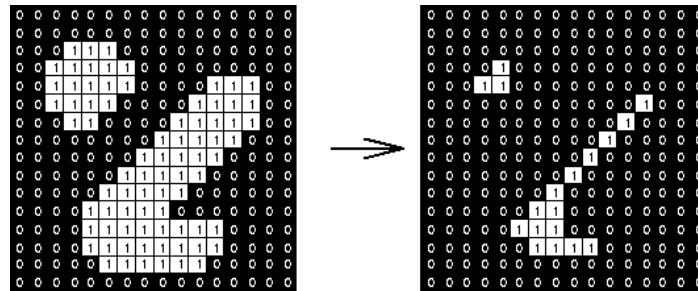
$$I \ominus E = \{z | (\hat{E})_z \subseteq I\} \quad (3.15)$$

where, I denotes the input image, and E denotes the structuring element.

The following figure 3.12 illustrates the erosion operation.



Part A of Figure 3.12.



Part B of Figure 3.12.

Figure 3.12: Upper image (A) is the 3x3 structuring element, bottom-left image is the input binary image, and bottom-right image (B) is the result of erosion operation on the input image using the given structuring element.

In this thesis, opening operation is used in order to eliminate those sharp peaks and small islands, which are noises, as seen from the figure 3.10. We used a 3x3 rectangular structuring element for dilation and erosion operations. The result of the opening operation is illustrated in the following figure 3.13.



Figure 3.13: Result of the opening operation, input image is shown in figure 3.10, and 3x3 structuring element is used.

The result of the opening operation can be seen from the figure 3.13. Most of the noisy pixels and pixel groups have been removed using opening operation. After removing the noises from the video frames, we have filled contours, most of them are being a parts of the object candidates.

In order to track an object in the following video frame sequences, we need to extract the deterministic features of the object, and then try to find these features in the next frame. Blob detection and extraction are methods of image segmentation that is mostly used in the fields of computer vision such as object recognition and tracking. Blob refers to a group of connected pixels forming a region in the image, that are either brighter or darker than their surroundings. Blob detection methods try to find those regions in a given input image. The main reason behind the usage of blob detection is that it provides complementary information about those regions which cannot be obtained by edge or corner detectors. It is also used in texture analysis and texture recognition as well as segmentation. Blob extraction is another

segmentation method that groups pixels in an image into discrete regions according to similar criteria. We can count, filter, and track blobs [1].

In our thesis, we extract blobs in process of segmentation. Because the blobs are formed of connected pixels, an object can be formed of several numbers of blobs. An example of blob extraction, using our test data, is illustrated in the figure 3.14.



Figure 3.14: Blob Extraction.

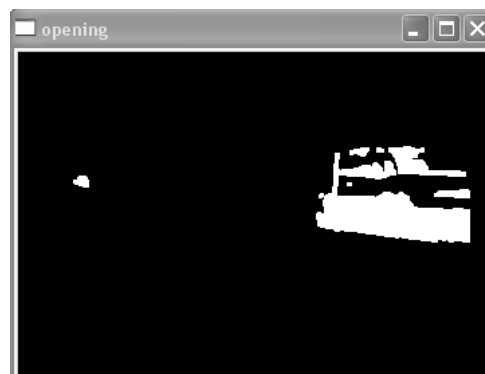
As we can see from the figure, three different blobs are found by the blob extractor. The one in the left part of the image, which is filled with purple, belongs to the object that is coming from back, the one in the middle, which is large in size, belongs to the object that is passing in front of the camera, and the rightmost blob which can be dark blue in color. The following figure 3.15 demonstrates the extracted blobs in gray-level image.



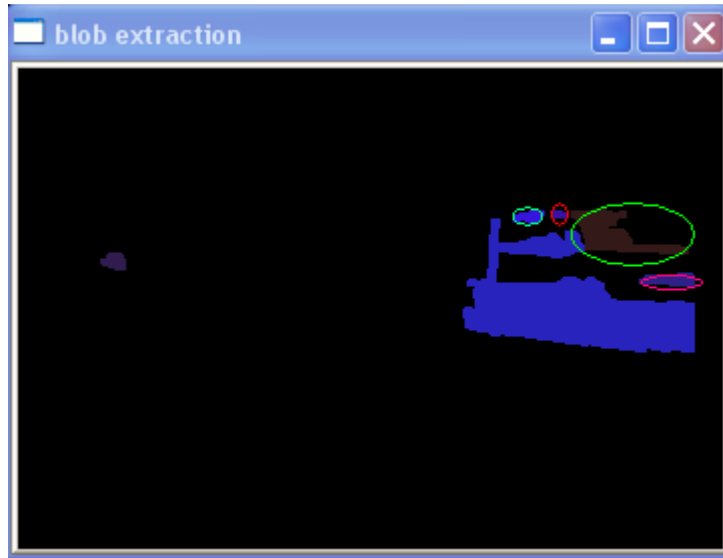
Figure 3.15: Blob Extraction.

The object that is coming from behind is showed with purple circle, the object that is passing in front of the camera is shown by the dark blue circle, and the third blob is shown by the red circle. We can say that, first and second blobs belong to the objects that can be tracked; however, the third blob is a false candidate, which is the shadow of the blue circled object. We can count that shadow as a noise that couldn't be eliminated by the noise elimination methods such as opening. Later parts of this chapter include a method to deal with the shadowy noises.

The following figure 3.16 shows the several numbers of blobs on a single object.



Part A of Figure 3.16.



Part B of Figure 3.16.

Figure 3.16: Upper image (A) is the output of opening algorithm, and the bottom image (B) is the output of blob extraction algorithm.

As we can see from the figure, when objects size is increased, the number of connected regions on it is also increases. The blob extractor finds five different blobs on the object, when it is about the leave the view area of the camera. The main blob having the largest area, which is not circled, is forming the body of the object. Other blobs, that are circled, forms the small parts of the object, and those blobs cannot be merged due to the large difference in characteristics. The black holes that are being in between those blobs are the parts that are hard to segment. After extracting those blobs, we can find object boundaries using them.

Object boundaries are lines in which the pixels of object lie within. We use *convex hulls* to find object boundaries.

Convex Hull of a point or pixel set S is the smallest number of convex set containing S [5]. Convex Hull of a point set X can be derived using the equation below [1].

$$H_{\text{convex}}(X) = \left\{ \sum_{i=1}^k \alpha_i x_i \mid x_i \in X, \alpha_i \in \mathbb{R}, \alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1, k = 1, 2, \dots \right\}. \quad (3.16)$$

where, k is an arbitrary natural number, numbers α_i are non-negative and sum up to 1, and the points x_i are in the set X . The following figure 3.17 is an example of convex hull.

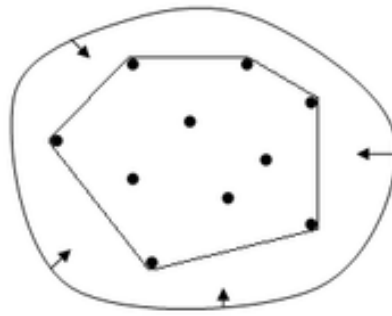
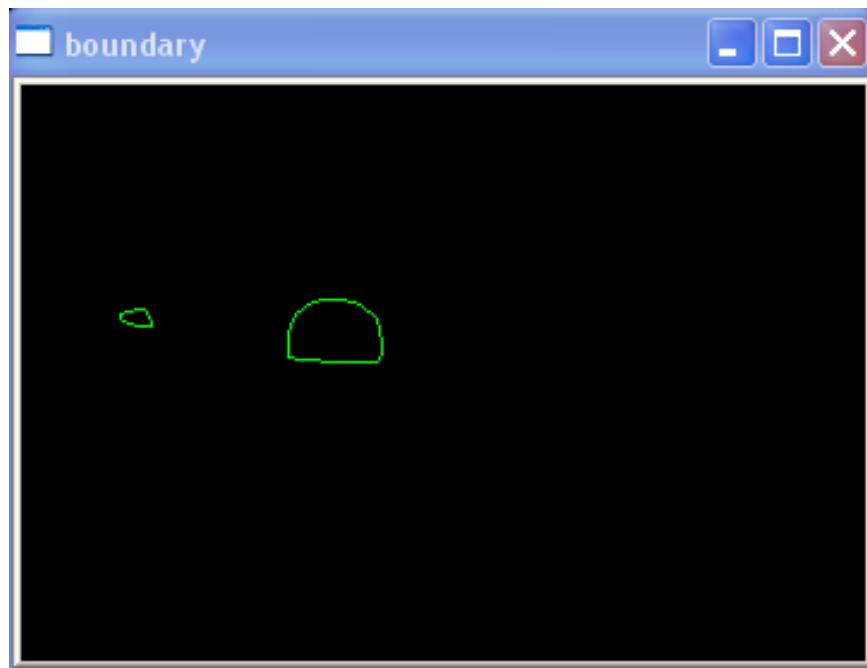


Figure 3.17: Convex Hull.

The blue lines are forming the convex hull for the points inside it in the example above. We can visualize a convex hull of a set by thinking of a rubber band stretched to encompass a given set of points, and when it is released, it will form the shape of the convex hull for that set of point, as shown in the figure 3.17. The figure 3.18 below shows the convex hulls that we gathered from the extracted blobs of objects.



Part A of Figure 3.18.



Part B of Figure 3.18.



Part C of Figure 3.18.

Figure 3.18: Upper image (A) shows the extracted blobs, and it is also the input to the method that finds convex hulls, center image (B) is the output of the convex hull method, bottom image (C) shows the drawn boundary in the color frame.

As we can see from the figure 3.18, convex hulls almost perfectly fits the objects. Also, by looking at the figure, we can see another false segmentation on the bottom part of the object. The background is threatened as foreground at there, because of the shadow of the object. The following parts of this chapter include the segmentation process done in this thesis.

Segmentation is the process of partitioning a digital image into multiple set of pixels, which is done in order to simplify the image representation and change it into something that is more meaningful and easier to analyze [1]. Image segmentation is one of hardest processes in the field of computer vision. Moreover, there is no general solution to the image segmentation problem among the many algorithms that have been developed. The segmentation technique that is used in this thesis is level-

set methods. It is derived to track moving interfaces. The central idea is represent the evolving contour using a signed function, where its zero level corresponds to the actual contour. Then, according to the motion equation of the contour, one can easily derive a similar flow for the implicit surface that when applied to the zero-level will reflect the propagation of the contour [1].

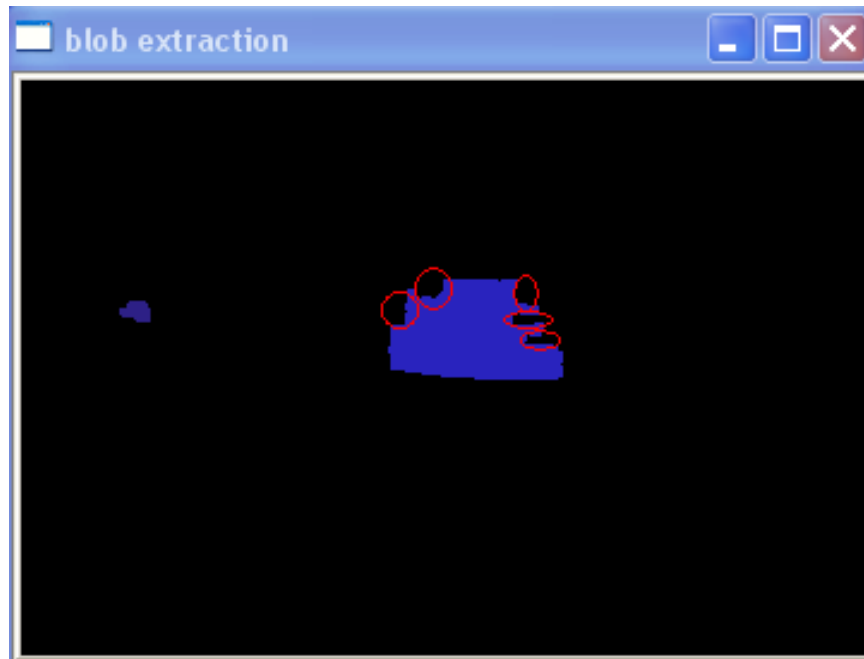
As we mentioned earlier, segmentation is a hard process for which, there is no general solution exists. Also, segmentation differs a lot according to the image that is desired to be segmented. The following figure 3.19 shows our segmentation process and the difficulties that we have faced during gray-level segmentation.



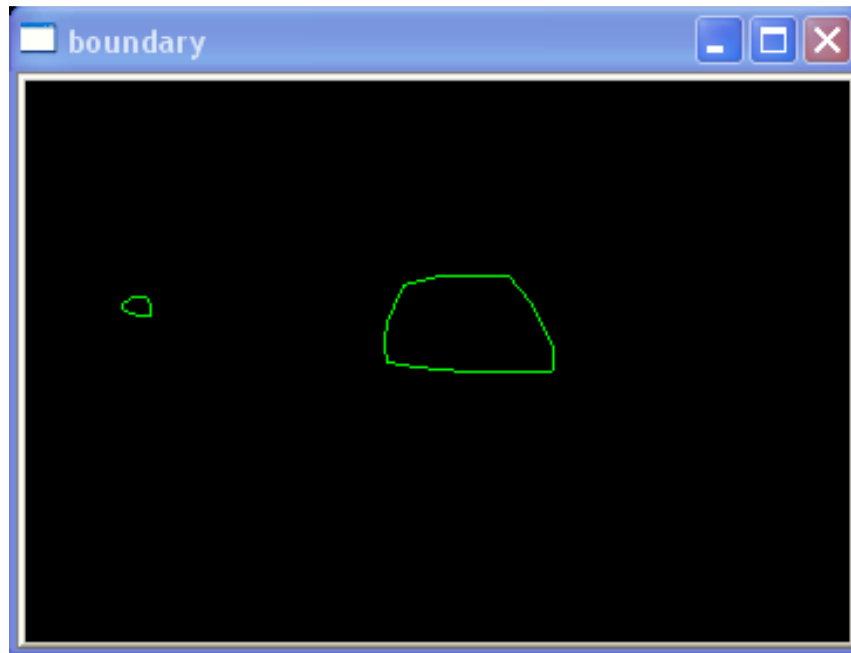
Part A of Figure 3.19.



Part B of Figure 3.19.



Part C of Figure 3.19.



Part D of Figure 3.19.



Part E of Figure 3.19.

Figure 3.19: (Parts: A, B, C, D, E) A is the reference image that we used as background, B is the current video frame, C shows extracted blobs, D shows convex hull, and E shows the boundaries of object in input frame.

In this thesis, we extract moving objects by taking the difference between the frame and our reference frame, in this case, a is our reference frame and b is our current frame. It can be clearly seen that, some parts of object are too similar to the background according to the intensity values, which are shown by the red circles in the image b. It is really hard to distinguish those parts from background due to the reason mentioned above. So, those parts are possible false segmentation candidates. The image c shows the segmentation results of the image gathered from the difference of image b from image a. As seen from the image c, the areas that are circled in image b could not be segmented, which are also circled in image c. However, when we find the convex hull of that blob, the areas that couldn't be segmented are omitted as seen from the image d. The last image, e, shows the convex hull over our object.

In order to strengthen the segmentation process, we also use YCbCr color space as well as gray-level color space. As clearly seen from the above examples, we cannot always trust, and so that decide by only looking at the intensity values of pixels.

YCbCr color space is mostly used in video and digital photography systems. Y denotes the luminance, and Cb and Cr are the blue and red chrominance components. The following figure 3.20 illustrates a color image, and Y, Cb, and Cr.

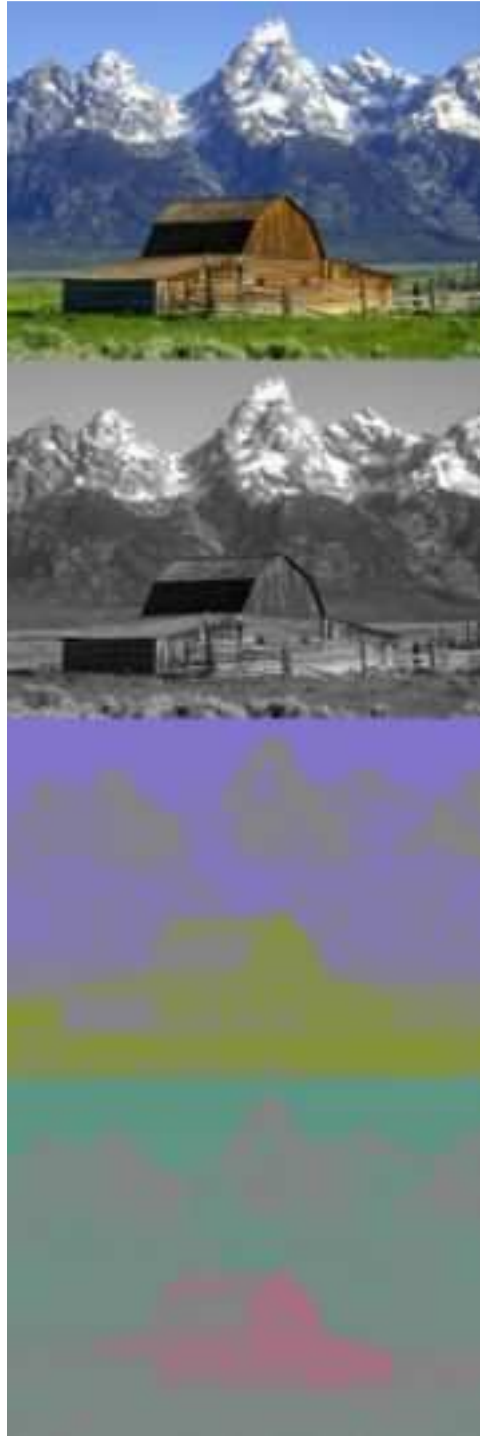


Figure 3.20: Color image, Y component, Cb and Cr components are shown respectively from top to bottom.

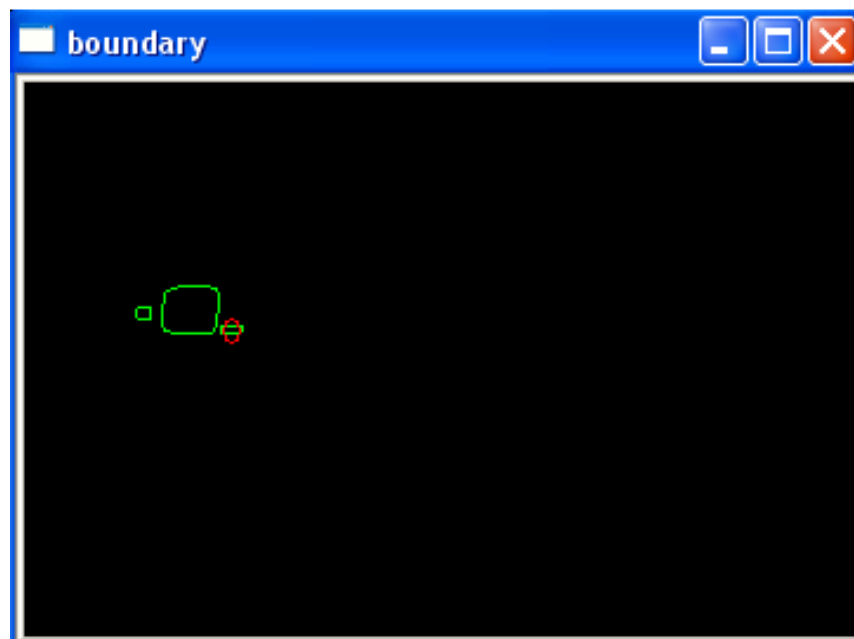
Y component of the image is same as the gray-level image of the color image. The brown building is represented by weak Cb, and strong Cr, green grass is represented by weak Cb and weak Cr, and blue sky is represented by strong Cr and weak Cb. The following formulation is used to make a convolution from RGB color space to YCbCr color space:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \quad (3.17)$$

$$Cb = (R - Y) * 0.713 + 128 \quad (3.18)$$

$$Cr = (B - Y) * 0.564 + 128 \quad (3.19)$$

In this thesis, we use Cb and Cr component of the YCbCr color space where segmentation using intensity fails. In our method, first, a static background image is formed in YCbCr same as the procedure used on gray-level background. Then, we check the Cb and Cr components of pixels that are segmented in gray-level, and compared them with the ones coming from background. The following figure 3.21 shows an example situation where segmentation fails by using intensity values.



Part A of Figure 3.21.



Part B of Figure 3.21.

Figure 3.21: Upper image (A) shows the boundaries of object candidates, false segmentation is shown by red circle, bottom image (B) is the demonstration of false segmentation on input video frame, where red circle shows the false segmentation.

In such cases mentioned in the above example, it is really hard to segment using intensity values. To solve this problem, first we calculate the standard deviation of C_b and C_r values of segmented points in the image. The following table 3.1 shows a section of the C_b and C_r values of background image and current frame.

Table 3.1: Shows the Cb and Cr distribution in between the background image pixels and current frames pixels, which is taken randomly from 5 sequential points.

Cb_current	Cb_background	Cr_current	Cr_background
.	.	.	.
.	.	.	.
.	.	.	.
114.000000	114.000000	143.000000	142.000000
117.000000	116.000000	142.000000	142.000000
117.000000	116.000000	143.000000	143.000000
113.000000	112.000000	143.000000	142.000000
113.000000	112.000000	143.000000	142.000000
.	.	.	.
.	.	.	.
.	.	.	.

We calculate a standard deviation according to the numbers given in the table 3.1 above, this deviation is used as a threshold value in distinguishing background and foreground pixels according. The following formulation is used to eliminate these false segmented pixels.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i = \frac{x_1 + x_2 + \dots + x_N}{N} \quad (3.20)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}. \quad (3.21)$$

where, x_i is the Cb and Cr values of segmented pixels, and N is the total number of segmented pixels in the image. σ is our threshold value that we use to distinguish foreground and background pixels. So after calculating the standard deviation, in other words threshold, we use the following method. A pixel is called as background pixel if both of the following criteria are satisfied:

$$\text{Cb_background} - \sigma < \text{Cb_current} < \text{Cb_background} + \sigma \quad (3.22)$$

$$\text{Cr_background} - \sigma < \text{Cr_current} < \text{Cr_background} + \sigma \quad (3.23)$$

The following table 3.2 shows a section of Cb and Cr values of false segmented points, in other words, both of the criteria above are satisfied.

Table 3.2: Cb and Cr values of false segmented pixels ($\sigma = 4.786789$).

Cb_current	Cb_background	Cr_current	Cr_background
.	.	.	.
124.000000	124.000000	132.000000	132.000000
124.000000	124.000000	132.000000	132.000000
127.000000	127.000000	132.000000	132.000000
127.000000	126.000000	131.000000	131.000000
124.000000	124.000000	134.000000	134.000000
.	.	.	.

By looking at the values inside the table 3.2 above, we can see that, at those points, Cb and Cr values are almost same for both background and current images. The following table 3.3 shows a section of Cb and Cr values of correctly segmented points.

Table 3.3: Cb and Cr values of correctly segmented points ($\sigma = 4.786789$).

Cb_current	Cb_background	Cr_current	Cr_background
.	.	.	.
114.000000	114.000000	145.000000	140.000000
111.000000	112.000000	147.000000	142.000000
112.000000	112.000000	147.000000	142.000000
114.000000	114.000000	145.000000	140.000000
113.000000	115.000000	145.000000	140.000000
.	.	.	.

The table 3.3 illustrates the correctly segmented points. Especially the Cr value varies a lot in those points. The following figure 3.22 shows those false segmented points and true segmented points in our test data.



Part A of Figure 3.22.



Part B of Figure 3.22.

Figure 3.22: Upper image (A) shows the boundaries of object, bottom image (B) shows true / false segmented points.

The white points in the bottom image of figure 3.22 show correctly segmented points, which are on the surface of the object, and gray points show the

false segmented ones. The red circle in the upper image of 3.22 shows the points that belong to the background, but segmented as a part of foreground.

After all these segmentation process, we have reliable features of the object that we are going to track, so we can begin the tracking process. Kalman filter is used for tracking process in this thesis.

Kalman filter is a powerful mathematical tool which is mostly used in the field of computer vision and computer graphics. We can define it as a recursive data processing algorithm that estimates the state of a dynamic system from a series of incomplete and noisy measurements [1]. It is optimal in the sense that it estimated error covariance. The following figure 3.23 illustrates the basic model of Kalman filter.

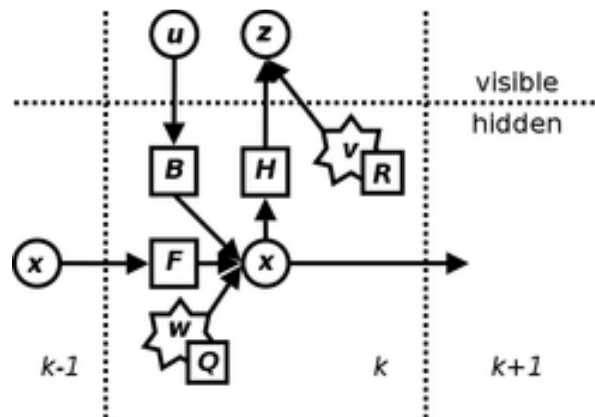


Figure 3.23: Model underlying the Kalman filter.

The Kalman filter assumes the true state at time k can be derived from the state at $k-1$ according to the following formula:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \quad (3.24)$$

with a measurement z_k of the true state x_k is made according to the following formula:

$$z_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad (3.25)$$

- F_k is the state transition model which is applied to the previous state x_{k-1} ,
- B_k is the control input model which is applied to the control vector u_k ,
- w_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k ,
- H_k is the observation model which maps the true state space into the observed space,
- v_k is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_k .

The formulas mentioned above are also modeled in the figure 3.1.23. An advantage of Kalman filter is that, because it is recursive, it doesn't require all previous data to be kept. Only the estimated state from the previous timestep and current measurements are needed for the filter itself in order to compute to current estimate. Kalman filter has two critical phases, predict and update. In predict phase, an estimation of the state at current timestep using the state estimation coming from the previous timestep is produced. In update phase, a new and more accurate estimate is produced again for the current timestep using the measurement information for the current timestep[1][8][9][10][11][12].

Predict:

Predicted State:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_{k-1} \quad (3.26)$$

Predicted Estimate Covariance:

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_{k-1} \quad (3.27)$$

Update:

Innovation or Measurement Residual:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad (3.28)$$

Innovation Covariance:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad (3.29)$$

Optimal Kalman Gain:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \quad (3.30)$$

Update State Estimate:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad (3.31)$$

Updated Estimate Covariance:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \quad (3.32)$$

where, $\hat{\mathbf{x}}_{n|m}$ denotes estimate of state X at time n given observations upto and including m.

As a summary of Kalman filter, we can say that it is the most popular optimal linear estimator, and have a lot of advantages as mentioned before. In our thesis, Kalman filter is used in order to get coordinates of specific points of our target object, which are extracted in earlier steps.

First of all, an initial estimate is done. Since we locate extracted features in the following frame sequences in time, we can give the coordinates of an extracted feature, let's say a corner, as initial estimate. We know every moving object has a

velocity, and from this information, we can say that the change in the position of extracted feature is directly related to the velocity. So, our state can be formed by the coordinates of the extracted feature, and the velocity of the object. After calculating the other parameters of Kalman, we make a prediction at a timestep in the future. This prediction gives us an idea about the current location of the feature. So we search the location given by the prediction with also including the variances calculated in the previous steps. So, with these information's, we can restrict the region that we look the feature inside. After finding the point that is most likely to be our feature point, measurement process has been complete. The next step is the calculation of Kalman gain. Kalman gain is calculated by combining the prediction and measurement. After calculating the Kalman gain, we can make a final state estimate which again shows us the region in which our feature point lie, in a more reliable way.

3.2 Results and Future Work

In this thesis, we try to track objects in following sequential video frames. The applied methods and their results show us that we have managed to track objects in time. However, our algorithm is not optimized, that's why, it is not in real-time due to the computational expensiveness lying on the nature of image manipulation algorithms.

The test material consists of eleven moving objects having various sizes. The objects that have small in size are difficult to track in all frames due to the difficulties

in tracking extracted key features. But the usage of Kalman filter helps us estimating the position of the feature.

As a future work, optimizations can be made in order to decrease computational delays, so that the algorithm can become closer to real-time.

CHAPTER 4

CONCLUSION

In this thesis, object tracking in video frame sequences has been done. After a series of segmentation process, object features have been extracted and tracked using the Kalman filter algorithm. The results show that moving objects can be detected and tracked successfully in this thesis.

REFERENCES

- [1] <http://en.wikipedia.org>

- [2] **WANG, Y., DOHERTY, J.F., VAN DYCK, R.E.** (2000) Moving Object Tracking in Video, *Applied Imagery Pattern Recognition Workshop, 2000. Proceedings. 29th*, 95-101.

- [3] **WANG, Y., DOHERTY, J.F., VAN DYCK, R.E.** (2000) Tracking Moving Objects in Video Sequences, *Proc. Conference on Information Sciences and Systems*, Princetown, NJ.

- [4] **GONZALEZ, R.C., WOODS, R.E.** (2002), *Digital Image Processing*, Prentice Hall, New Jersey.

- [5] **BAE, K.H., KOO, J.S., KIM, E.S.** (2003) A New Stereo Tracking System Using Disparity Motion Vector, *Optical Communications*, 1-3. Vol. 221.

- [6] **BIRCHFIELD, S., TOMASI, C.** (1999) Depth Discontinuities by Pixel-to-Pixel Stereo, *International Journal of Computer Vision*, 269-293. 35(3).

- [7] **KIM, J., HWANG, D., JEONG, H., SONG, C., LEE, K., LEE, M.** (1998) Development of Depth Extraction Algorithm for the Stereo Endoscopic Image, *Proc. of the Engineering in Medicine and Biology Society*, 884-887. Vol. 2.

- [8] <http://ourworld.compuserve.com/homepages/PDJoseph/kalman.htm>

- [9] <http://www.innovatia.com/software/papers/kalman.htm>
- [10] **GINHOUX, R., GUTMANN, J.S.** (2001) Model-Based Object Tracking Using Stereo Vision, *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 1226-1232. Vol. 2.
- [11] **KALMAN, R.E.**, (1960) A new Approach to Linear Filtering and Prediction Problems, *Transactions of the ASME – Journal of Basic Engineering*, 35-45. Vol. 82.
- [12] **WELCH, G., BISHOP, G.**, (2001), *An Introduction to the Kalman Filter*.
- [13] **BOVIK, A.** (2000), *Handbook of Image & Video Processing*, Academic Press, Canada.
- [14] **MAYBECK, P.S.** (1979) Stochastic Models, Estimation, and Control, *Automatic Control, IEEE Transactions on*, 868-869. Vol. 28.
- [15] **REN, J., AGGOUN, A., MCCORMICK, M.** (2003) A Novel Object Depth Estimation Algorithm for Integral 3D Images, *Visual Information Engineering, 2003. VIE 2003. International Conference on*, 198-201.
- [16] **RUSS, J. C.** (1999), *The Image Processing Handbook*, CRC Press LLC, Florida.
- [17] **SALINAS, R.M., AGUIRRE, E., SILVENTE, M.G.** (2007) People Detection and Tracking Using Stereo Vision and Color, *Image and Vision Computing*, 995-1007. Vol. 25.
- [18] Intel® Open Source Computer Vision Library, *Reference Manuals*.
- [19] **COMANICIU, D., RAMESH, V., MEER, P.** (2003) Kernel-Based Object Tracking, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 564-577. Vol. 25.

- [20] **CUEVAS, E., ZALDIVAR, D., ROJAS, R.** (2005), Stereo Tracking, *Technical Report B*.
- [21] **WONG, K. Y., SPETSAKIS, M. E.** (2002) Motion Segmentation and Tracking, *Proceedings of 15th International Conference on Vision Interface*, 80-87.
- [22] **XU, R.Y.D., ALLEN, J.G., JIN, J.S.** (2004) Robust Real-Time Tracking of Non-rigid Objects, *ACM International Conference Proceeding Series; Proceedings of the Pan-Sydney area workshop on Visual information processing*, 95-98. Vol. 100.