VIDEO OBJECT TRACKING USING MOTION ESTIMATION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
in
Electrical Engineering

By

Himanshu Maharana (108EE051)
Dubasi Monika (108EE073)
Soumya Ranjan Sahoo (108EE075)

Under the Guidance of

Prof. Dipti Patra

Department of Electrical Engineering
National Institute of Technology, Rourkela
May 2012

Department of Electrical Engineering
National Institute of Technology, Rourkela
MAY 2012

CERTIFICATE

This is to certify that the thesis entitled "***Video Object Tracking Using Motion Estimation***" submitted by **Himanshu Maharana (108EE051), Dubasi Monika (108EE073) & Soumya Ranjan Sahoo (108EE075)** in partial fulfillment of the requirement for the award of degree ***Bachelor of Technology*** in ***Electrical Engineering*** to ***National Institute of Technology, Rourkela*** is a record of the candidate own work carried out by them under my supervision.

The matter embodied in this thesis is original and has not been submitted for the award of any of degree in any other Institute or College.

Date: 07-May-2012

Prof. Dipti Patra
Department of Electrical Engineering
National Institute of Technology, Rourkela

Department of Electrical Engineering
National Institute of Technology, Rourkela
MAY 2012

ABSTRACT

Real time object tracking is considered as a critical application. Object tracking is one of the most necessary steps for surveillance, augmented reality, smart rooms and perceptual user interfaces, video compression based on object and driver assistance. While traditional methods of Segmentation using Thresholding, Background subtraction and Background estimation provide satisfactory results to detect single objects, noise is produced in case of multiple objects and in poor lighting conditions.

Using the segmentation technique we can locate a target in the current frame. By minimizing the distance or maximizing the similarity coefficient we can find out the exact location of the target in the current frame. Target localization in current frame was computationally much complex in the conventional algorithms. Searching an object in the current frame using these algorithms starts from its location of the previous frame in the basis of attraction probably the square of the target area, calculating weighted average for all iteration then comparing similarity coefficients for each new location.

To overcome these difficulties, a new method is proposed for detecting and tracking multiple moving objects on night-time lighting conditions. The method is performed by integrating both the wavelet-based contrast change detector and locally adaptive thresholding scheme. In the initial stage, to detect the potential moving objects contrast in local change over time is used. To suppress false alarms motion prediction and spatial nearest neighbour data association are used. A latest change detector mechanism is implemented to detect the changes in a video sequence and divide the sequence into scenes to be encoded independently. Using the change detector algorithm (CD), it was efficient enough to detect abrupt cuts and help divide the video file into sequences. With this we get a sufficiently good output with less noise. But in some cases noise becomes prominent. Hence, a method called correlation is used which gives the relation between two consecutive frames which have sufficient difference to be used as current and previous frame. This gives a way better result in poor light condition and multiple moving objects.

Department of Electrical Engineering
National Institute of Technology, Rourkela
MAY 2012

ACKNOWLEDGEMENTS

It is with great reverence that we wish to express our deep gratitude towards **Prof. Dipti Patra** for her untiring support and valuable knowledge that helped us immensely to make this project a success.

Her astute guidance, invaluable suggestions, enlightening comments and constructive criticism always kept our spirits up during our work.

Our experience in working on this project has been wonderful. We hope that the knowledge, practical and theoretical, that we have gained through this project will help us in our future endeavors.

Himanshu Maharana (108EE051)
Dubasi Monika (108EE073)
Soumya Ranjan Sahoo (108EE075)
Date: 07-05-2012
Dept. of Electrical Engineering
National Institute of Technology, Rourkela

TABLE OF CONTENTS

LIST OF SYMBOLS

1.	$\boldsymbol{\sigma_w^2(t)}$	Intra class variance
2.	$\boldsymbol{\sigma_i^2(t)}$	Variance of a particular class
3.	$\boldsymbol{w_i(t)}$	Probability of two classes separated by a threshold.
4.	$\boldsymbol{\sigma_b^2(t)}$	Inter class variance
5.	$\mu_{\boldsymbol{i}}\boldsymbol{(t)}$	Mean of a class.
6.	**DCT**	Discrete cosine transforms
7.	$\boldsymbol{I_t}$	Intensity of a particular pixel
8.	$\boldsymbol{B_t}$	Intensity level of background
9.	$\boldsymbol{\tau}$	Predefined threshold
10.	$\boldsymbol{I_x}$	Derivative of the image intensity value along x axis
11.	$\boldsymbol{I_y}$	Derivative of the image intensity value along y axis
12.	$\boldsymbol{I_z}$	Derivative of the image intensity value along z axis
13.	$\boldsymbol{\alpha}$	Regularization constant
14.	$\mu_{\boldsymbol{T}}$	Mean intensity for the whole image
15.	**SAD**	Sum of absolute difference
16.	$\boldsymbol{IC_{ji}}$	Contrast change image
17.	$\boldsymbol{L_o}$	Object Luminance
18.	$\boldsymbol{L_B}$	Surrounding background Luminance
19.	$\boldsymbol{\sigma_L}$	Local standard deviation
20.	$\mu_{\boldsymbol{L}}$	Local mean intensity
21.	$\boldsymbol{M_{CM}}$	Local contrast salience map
22.	$\boldsymbol{MD^{[p,q]_{c_{ji}}}}$	Mask for contrast difference image between two contrast salience images.
23.	$\boldsymbol{I^{[p,q]_{c_{mi}}}}$	Contrast image of i^{th} image.

LIST OF FIGURES

CHAPTER I: INTRODUCTION AND REVIEW

1.1 INTRODUCTION:-

PROJECT OBJECTIVE: Object detection and motion estimation in a video file.

Video object tracking has got wide application in vision, security, observational issues in natural science and in various other fields. Video surveillance for security purpose is one of its major applications. Object tracking has high priorities in religious places, market buildings, courts, train stations and airports. Various other applications include military, astronomy, road traffic regulation, robotics, medical imaging. Air traffic control is a typical application of video object tracking, where aircraft are more or less continuously visible on radar, but in case the transponders are absent the identity is only revealed when the pilot reports by radio.

Video analysis basically involves three key steps:

- To detect the moving object under interest.
- Track the object from frame to frame
- Analyse the object tracks to know their behaviour.

In simple words, tracking may be defined as the estimation of the trajectory of a moving object in the image plane as it moves around the scene. Consistent labels are assigned to the tracked objects in each frame of a video. Further based on the tracking domain, a tracker can give useful information such as area, shape, and orientation of the object under interest.

Object tracking can be complex due to following reasons -

- noise in images,
- complex object motion,
- articulated nature of non-rigid objects
- scene illumination changes, and
- real-time processing requirements.

Tracking can be simplified by making some assumptions or imposing some constraints on the motion or appearance of the object. In almost all tracking algorithms, object motion is assumed to be smooth with no abrupt changes in between. Prior knowledge about the object size, number, appearance, shape and motion can also help in its tracking. A number of methods for object tracking have been proposed.

In our project we have basically worked on following methods-

a) Background Subtraction Method

b) Background Estimation Method

c) Optical Flow Method

d) Adaptive Contrast Change Detection

1.2 MODEL:-

1.2.1 THRESHOLDING:

Thresholding is based on clip-level or value to turn a gray-scale image into a binary image. It is the simplest method used for image segmentation. This method is carried out by first selecting a threshold value (or values in case of multiple-levels are selected) which is optimum. In industry several popular methods are used, including the maximum entropy method, Otsu's method (maximum variance) etc.

1.2.2 BACKGROUND SUBTRACTION:

We use only the successive I-frames for tracking in our algorithm and thereafter interpolate the object motion in the intermediate frames. We initially acquire a DCT image of an I-frame representing the background that is treated as the reference image. After that, all subsequent DCT images are compared with the reference image to segment the foreground object. Based on the model of the application the background image is created and is updated from time to time whenever there is a permanent change in the background. Details regarding Thresholding and Background Subtraction method have been described later.

1.2.3 BACKGROUND ESTIMATION:

In this technique the algorithm identifies the incomplete background as those pixels that do not belong to the foreground pixels. As the foreground objects move, the background estimation algorithm estimates more and more of the background pixels.

Once background estimation is completed by the program, the background is subtracted from each video frame to produce foreground images. This foreground image is converted to binary image. This is carried out by implementing thresholding and performing blob-analysis and other morphological closing on each foreground image. Then object tracking is carried out by another program.

1.2.4 OPTICAL FLOW:

The optical flow of a video frame is a field of motion vector per pixel or sub-pixel. Multiple methods allow computing the optical flow among which partial differential equation based methods, gradient consistency based techniques and least squared methods.

In this model we have used an optical flow estimation technique to get an estimation of the motion vectors in each frame of the video sequence. Then the required moving object is detected by a program block and converted into binary image. This is carried out by implementing thresholding and performing blob-analysis and other morphological closing on each foreground image. Then object tracking is carried out by another program.

1.2.5 ADAPTIVE CONTRAST DETECTION:

This method is effectively used basically for detection and tracking multiple moving objects on night-time lighting conditions. Normalized cross-correlation is used to remove effect of global changes in the lighting form one frame to another.

First of all potential moving objects are detected by using contrast in local change over time. Motion prediction and spatial nearest neighbor data association is used to suppress false alarm due to minor contrast change. After this program is executed we get a contrast image which contains the moving objects. This contrast image is converted to binary image. This is carried out by implementing thresholding and performing blob-analysis and other morphological closing on each foreground image. Then object tracking is carried out by another program.

CHAPTER II: METHODOLOGY

In computer vision, **segmentation** is the process which divides a digital image into multiple segments (sets of pixels, also known as super-pixels). The main aim of segmentation is to simplify and/or change the representation of an image frame into something which is more meaningful and easier to analyse. Segmentation of image is typically used to detect/locate objects and boundaries (lines, curves, etc.) in an image frame. More precisely, segmentation of an image is carried out to assign a label to each pixel in an image such that pixels with the similar label share certain visual characteristics.

2.1 THRESHOLDING:

Thresholding is the simplest method of image segmentation. Thresholding is used to create binary images from a gray-scale image. In the thresholding process, depending on their values individual pixels in an image, pixels are marked as "object" pixels if the value is greater than some threshold value (assuming an object to be brighter than the background) else the pixels are marked as "background" pixels.

There are various conventions such as threshold above, threshold below, threshold inside and threshold outside. In our case we have used “threshold above” convention. The value “1” is assigned to object pixel while value “0” is assigned to background pixel. Then a binary image is constructed by colouring each pixel according to the values assigned to them. Different thresholding techniques are available on the basis of information and algorithms.

Thresholding can be classified as bi-level and multi-level. In bi-level thresholding, the pixels are classified into two groups, one containing the pixels having gray levels above the threshold and the other with gray levels below the threshold. Multiple thresholds are present in multilevel thresholding [1]. Pixels are grouped having gray level within a threshold.

Many popular methods are used now a days including the maximum entropy method, Otsu's method [2] (maximum variance) etc.

Here we have briefly analyzed thresholding using Otsu's approach.

2.1.1 OTSU'S APPROACH:

An image is a 2D grayscale intensity function. It contains N pixels with gray levels from 1 to L. Let the number of pixels with gray level value **i** is denoted by $\mathbf{f_i}$, giving a probability of gray level **i** in an image of

$$\boldsymbol{p^i = f^i / N} \qquad \text{..... (3.1.1)}$$

While implementing bi-level thresholding of an image, the pixels are divided into two classes, C_1 with gray levels [1, …, t] and C_2 with gray levels [t+1, …, L]. Then, for these two classes the gray level probability distributions is given as:

$\mathbf{C_1: p_1/ \omega_1 (t), ..., p_t / \omega_1(t)}$ **and**

$\mathbf{C_2: p_{t+1}/ \omega_2 (t), p_{t+2}/ \omega_2 (t), ..., p_L/ \omega_2 (t),}$

Where,

$$\boldsymbol{\omega_1(t) = \sum_{i=1}^{t} p_i} \quad \text{.... (3.1.2)}$$

$$\boldsymbol{\omega_2(t) = \sum_{i=t+1}^{L} p_i} \quad \text{.... (3.1.3)}$$

Also, the means for classes C_1 and C_2 are:

$$\boldsymbol{\mu_1(t) = \sum_{i=1}^{t} ip_i / \omega_1(t)} \quad \text{.... (3.1.4)}$$

$$\boldsymbol{\mu_2(t) = \sum_{i=t+1}^{t} ip_i / \omega_2(t)} \quad \text{.... (3.1.5)}$$

Let **the mean intensity** is given by $\boldsymbol{\mu_T}$ for the whole image. This can be easily shown that

$$\boldsymbol{\mu_1\omega_1 + \mu_2\omega_2 = \mu_T \mu_1} \quad \text{.... (3.1.6)}$$

$$\boldsymbol{\omega_1 + \omega_2 = 1} \quad \text{. (3.1.7)}$$

Using discriminant analysis, Otsu defined the between-class variance of the thresholded image as

$$\boldsymbol{\sigma_B^2 = \omega_1 (\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2} \quad \text{....(3.1.8)}$$

For bi-level thresholding, Otsu verified that the optimal threshold t* is chosen so that the between-class variance σ_B^2 is maximized.

$$\boldsymbol{t^* = \text{Arg Max} \{\sigma_B^2 (t)\}} \quad \text{....(3.1.9)}$$

$$\boldsymbol{\text{for } 1 \leq t < L}$$

The previous formula can be easily extended to multilevel thresholding of an image.

Assuming that there are M-1 thresholds, $\{t_1, t_2, \ldots, t_{M-1}\}$, that divides the original image into M classes:

C_1 for [1, 2,…, t1], C_2 for $[t_1+1, \ldots, t_2]$, … , C_i for $[t_{i-1}+1, \ldots, t_i]$, …, and C_M for $[t_{M-1}+1, \ldots, L]$, the optimal value of thresholds $\{t_1^*, t_2^*, \ldots, t_{M-1}^*\}$ are chosen by maximizing σ_B^2 as follows:

$$\{t_1^*, t_2^*, \ldots, t_{M-1}^*\} = \text{Arg Max}\,\{\sigma_B^2\,(t)\,(t_1, t_2, \ldots, t_{M-1})\} \quad \ldots\ldots (3.1.10)$$

$$1 \leq t_1 < \ldots < t_{M-1} < L$$

where

$$\sigma^2 = \sum_{k=1}^{M} \omega_k (\mu_k - \mu_t)^2 \quad \ldots\ldots (3.1.11)$$

with

$$\omega_k = \sum_i p_i \quad \ldots (3.1.12)$$

$$\mu_k = \sum_i i p_i / \omega_k \quad \ldots\ldots (3.1.13)$$

The ω_k in Eq. (3.1.12) is regarded as the zeroth-order cumulative moment of the kth class C_k, and the numerator in Eq. (3.1.13) is regarded as the first-order cumulative moment of the kth class C_k; that is,

$$\mu_k = \sum_i i p_i \quad \ldots\ldots (3.1.14)$$

However, the Otsu method has its drawbacks. It is quite time-consuming owing to the enormity of computations involved. Nevertheless, substantial research in this area has resulted in a number of latest algorithms which are faster and reliable. We are deliberating such alternatives and shall eventually select one based on the requisite of the final objective of the project.

2.2 BACKGROUND SUBTRACTION:

Background subtraction **[3]** is a commonly used class of techniques for segmenting out objects of interest in a scene for applications such as surveillance. It compares an observed image with an estimate of the image if it contained no objects of interest. The areas of the image plane where there is a significant difference between the observed and estimated images indicate the location of the objects of interest. The name “background subtraction" comes from the simple technique of subtracting the observed image from the estimated image and thresholding the result to generate the objects of interest. Here we survey several techniques which are representative of this class, and compare three important attributes of them: how the object areas are distinguished from the background; how the background is maintained over time; and, how the segmented object areas are post-processed to reject false positives, etc.

Several algorithms were implemented to evaluate their relative performance under a variety of different operating conditions. With this, some conclusions can be drawn about what features are important in an algorithm of this class.

In our algorithm, we have used successive I-frames for tracking and thereafter we have interpolated the motion of the object in the intermediate frames. Initially we acquire a DCT image of an I-frame representing the background, which is used as the reference image. Then, all the DCT images are compared with the reference image subsequently to segment the foreground object. Based on the model of the application the background image is created and is updated from time to time whenever there is a permanent change in the background.

2.2.1 ALGORITHM USED:

A pixel is marked as foreground if

$$|\boldsymbol{I}_t - \boldsymbol{B}_t| > \tau \qquad \textbf{.... (3.2.1)}$$

where τ is a “predefined" value threshold. The process thresholding is followed by closing with a 3 X 3 kernel and the discarding of small regions. The background is updated as

$$\boldsymbol{B}_{t+1} = \boldsymbol{\alpha I_t} + (\boldsymbol{1} - \boldsymbol{\alpha})\boldsymbol{B_t} \qquad \textbf{.... (3.2.2)}$$

where the value $\boldsymbol{\alpha}$ is kept small to prevent the detection of artificial “tails" forming behind moving objects.

Two background corrections are applied:

1. If a pixel is marked as foreground for more than m of the last M frames, then the background is updated as $\boldsymbol{B}_{t+1} = \boldsymbol{I}_t$. This correction is designed to compensate for sudden illumination changes and the appearance of static new objects.
2. If a pixel change is frequent that it changes its state from foreground to background frequently, it can be masked out due to inclusion in the foreground. This is designed to compensate for fluctuating illumination, such as swinging branches of trees.

2.3 BACKGROUND ESTIMATION:

The model uses the *Background Estimation* **[4]** *Technique* that we specify in the Edit Parameters block to estimate the background.

Here we have used the following algorithm for background estimation.

2.3.1 ELIMINATING MOVING OBJECTS:

Initially this algorithm identifies the moving objects in the first few image frames and then labels the corresponding pixels as foreground pixels. Next, the algorithm identifies those pixels that do not belong to the foreground pixels as the incomplete background. The algorithm estimates more and more of the background pixels as the foreground objects move.

Once the background estimation is completed by the program, the background is subtracted from each video frame to produce foreground images. This foreground image is converted to binary feature image. This is carried out by implementing thresholding and performing certain morphological closing on each foreground image. Then object tracking is carried out by another block. The model locates the objects in each binary feature image using the Blob Analysis block. Then the Draw Shapes block is used to draw a green rectangle around the objects that moves beneath the white line. A counter is used in the upper left corner of the Results window to track the number of objects in the region of interest.

2.4 Optical Flow:

In a video frame, the field of motion vector per pixel or sub-pixel is called optical flow. There are many a methods for computing optical flow among which few are partial differential equation based methods, gradient consistency based methods and least squared methods.

The model used here is based on an optical flow estimation technique which estimates the motion vectors in each frame of a video sequence. Then, thresholding followed by morphological closing on the motion vectors produces binary feature images. The model further locates objects in each binary image using the Blob Analysis block. The Draw Shapes block draws a green rectangle around the objects that moves beneath the white line. The counter in the Results window tracks the number of objects in the ROI.

2.4.1 Need for optical flow:

In the area of optical flow research, motion estimation has developed as one major aspect. The optical flow field is more or less similar to a dense motion field which is computed in motion estimation. Optical flow can be used not only for the determination of the optical flow field itself, but also of its requirement in the estimation of the three-dimensional nature and structure of a scene, and the 3D motion of objects and the observer relative to the scene.

Robotics researchers have been widely using optical flow in areas as diverse as: object detection and tracking, movement detection, image dominant plane extraction, robot navigation, visual odometry etc. Moreover, Optical flow information has also been identified as a potential method for controlling micro-air vehicles.

The problems with the application of optical flow include that of inferring the motion and the structure of observer and the objects in a scene. In animal (and human) vision, motion knowledge and generation of mental maps of the environment structure are deciding components so the conversion of this inherent ability to a computer capability extremely critical in the field of machine vision.

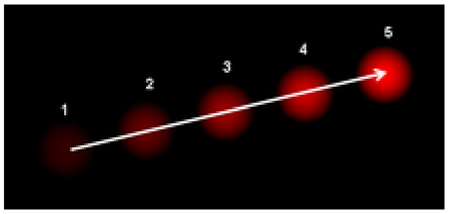


Fig. 1 Optical Flow of an Object

2.4.2 MATHEMATICAL DETAILS

For Optical flow estimation, there exist two well-known methods i.e. the Horn-Schunck [5] algorithm and the Locase-Kanade method. While the Horn-Schunck method (global method) is based on smoothness in flow over the entire image, the Locas-Kanade's method is a local method. Here we study in detail the Horn-Schunck method as it effectively minimizes distortions in flow thus giving solutions which show more smoothness.

The formula for optical flow is a global energy functional sought to be minimized for best results. For two-dimensional image streams, the function is as follows:

$$\boldsymbol{E = \iint \left[\left(I_x u + I_y v + I_t \right)^2 + \alpha^2 \left(\|\nabla u\|^2 + \|\nabla v\|^2 \right) \right] dx\, dy} \qquad \textbf{.... (3.4.1)}$$

where Ix, Iy and It are the derivatives of the image intensity values along the x, y and time dimensions respectively, $\vec{V} = [u(x,y), v(x,y)]^T$ is the optical flow vector, and 'α' is a regularization constant. Larger the value of α smoother will be the flow. This functional can be simplified by solving the associated Euler-Lagrange equations.

These are

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 0 \qquad \text{.... (3.4.2)}$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} = 0 \qquad \text{.... (3.4.3)}$$

where L is the integrand of the energy expression, which gives

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0 \qquad \text{.... (3.4.4)}$$

$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0 \qquad \text{.... (3.4.5)}$$

The subscripts denote partial differentiation, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ denotes the Laplace. In practice, however the Laplacian is approximated numerically using the method of finite differences, and may be written as $\Delta u(x, y) = \bar{u}(x, y) - u(x, y)$ where $\bar{u}(x, y)$ is a weighted average of u calculated in a neighborhood around the pixel at location ***(x, y)***. Using this notation the equations (3.4.4) and (3.4.5) may be written as

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \bar{u} - I_x I_t \qquad \text{.... (3.4.6)}$$

$$I_x I_y u + (I_x^2 + \alpha^2)v = \alpha^2 \bar{v} - I_y I_t \qquad \text{.... (3.4.7)}$$

which is linear in u and v and may be solved for each pixel in the image.

The solution must be updated once the neighbours have been update because the solution depends on the neighbouring values.

From the above discussion an iterative scheme is derived, given as:

$$\boldsymbol{u^{k+1} = \bar{u}^k - \frac{I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}} \quad \textbf{.... (3.4.8)}$$

$$\boldsymbol{v^{k+1} = \bar{v}^k - \frac{I_y(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}} \quad \textbf{.... (3.4.9)}$$

where the superscript **k+1** denotes the next iteration, which is to be calculated and **k** is result for the last calculated iteration. This is known as the Jacobi method and is applied to the large, sparse system arising when solving for all pixels simultaneously.

2.4.3 PROPERTIES OF THE HORN-SCHUNK:

The Horn–Schunck algorithm has its merits in the fact that it yields a high density of flow vectors, i.e. the flow information missing in internal parts of homogeneous objects is filled in from the motion boundaries. However, it has its demerits too. It tends to be more sensitive to noise than local methods.

2.5 ADAPTIVE CONTRAST CHANGE DETECTION

The method of adaptive contrast change detection [6] for video object tracking essentially involves integrating both the wavelet-based contrast change detector and locally adaptive thresholding scheme. This is preferred for night surveillance [7] and multiple colored objects [8] tracking. The first step includes computing the contrast in local change over time which is used to detect potential moving objects. This is followed by motion prediction and spatial nearest neighbour data association which helps to suppress false alarms.

2.5.1 MOTION DETECTION AND OBJECT TRACKING

A. CHANGE DETECTION AND SEQUENCE TESTER

A new change detector is used to detect changes in a video sequence, which is further divided into scenes to be encoded independently. It is then enhanced using a sequence tester to test each scene for high activity (arising due to motion) and thus decide whether or not another reference (key) frame is required. The performance of the suggested method is checked with the help of simulation results. This mechanism is based on segmentation of the video into scenes using a change detector wherein one or more key frames are chosen for each scene.

A scene is a sequence of shots that belong together semantically. According to the length and properties of the scene, the numbers and locations of the key frames are chosen. Each of the other frames is then differenced from the nearest key frame. In a scene, there exist two types of shot boundaries in videos: abrupt shot changes called "cuts", and gradual transitions between two different shots. The proposed scene cut detector here is simple and fast. The detector divides the whole video into independent sequences to be encoded separately which is accomplished in two steps. Firstly, the video as a whole is used to get the global cuts, thus dividing the video into sequences which is the output of the change detector (CD). Secondly, each sequence is investigated to ensure that there is not any high level of changes between the frames that may cause artifacts in the decoded video. The results are good since complexity involved here is low and this module in the code boosts efficiency further.

CHANGE DETECTOR (CD)

The change detector uses the normalized sum of absolute differences (SAD) between each two successive frames on a pixel-by-pixel basis. If this difference normalized difference is high, then an abrupt cut is present. The various conditions and effects are as follows:

- If the normalized difference is low, then cut type is continuous.
- If the normalized difference is in between, then cut type is gradual change.
- If the normalized difference is high, then cut type is abrupt change.

The change detector algorithm (CD) gave efficient results to detect abrupt cuts and helped divide the video into sequences. However in the case of long scenes (where motion occurs), there is a high level of changes between frames which must be considered which is because of the fact that the detector is based on the concept of differencing frames i.e. for each scene, a key frame is chosen and all other nearby frames are differenced from it. In case of long scenes with high level of changes between frames especially those with respect to the key frame, the encoded frame carries huge information which may be lost due to lousy encoding and decoding and there arises the need for another key frame.

Here, a key frame is chosen for every scene and all the other frames are differenced from it. The change detector algorithm thus helps to determine the gradual change and abrupt change.

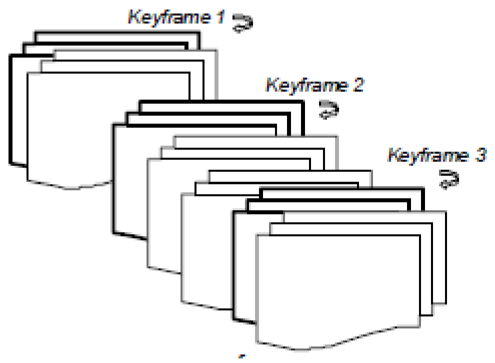


Fig. 2 Key frames

B. Algorithm: Object Detection and Tracking for Virtual Surveillance

The algorithm for object detection involves two steps. In the first step, local contrast computed over the entire image detects the object. In the second step, the detected objects are tracked and the falsely detected ones are removed using a feedback mechanism from the tracking algorithm. Here, we assume I as the image frame, R as the inter-frame relation which describes the similarity between two frames, T1 the contrast threshold and T the contrast change (CC) threshold. Contrast Change (CC) Image ICji is calculated between every two successive frames, which are deployed for object mask detection, which in turn is used for object tracking. The algorithm is described in the following section.

C. Content Detection based on Local Contrast Computation

To make detection more precise, we consider {I1, I2 … IM} to be an image sequence in which each image maps a pixel coordinate $x \in R1$ to an intensity or color $l(x) \in R^k$. where typically, k = 1 (for gray-scale images) or k = 3 (for RGB color images). Many methods are there to determine the contrast in an image.

One typical method is where luminance contrast is defined as the relative difference between luminance of the object, L_O and the surrounding background, L_B (i.e. as $\mathbf{C = (L_O - L_B)/ L_B}$), also called Weber contrast. Michelson defined contrast for elementary patterns as $C = (L_{max} - L_{min}) / (L_{min} + L_{max})$.

Here, we use a simple measure of contrast defined as the ratio of the local standard deviation σ_L of the image intensities to the local mean intensity μ_L.

$$C_L = \frac{\sigma_L}{\mu_L} \quad \text{.... (3.5.1)}$$

The local mean intensity $\mu^{(p,q)}$ of a (2p + 1) x (2q + 1) block of pixels is:

$$\mu^{(p,q)} = £_i£_j\, I(i,j)/\{(2p+1)(2q+1)\} \quad \text{.... (3.5.2)}$$

The local standard deviation $\sigma^{(p,q)}$ of the block of pixels is:

$$\sigma^{(p,q)} = [[££\, I(i,j) - \mu(p,q)]^2/\{(2p+1)(2q+1)\}]^{1/2} \quad \text{.... (3.5.3)}$$

Local contrast is related to entropy in a statistical sense. However, local contrast is simpler and faster to compute. From the above discussion it is clear that objects are detected only when the contrast exceeds a certain threshold. From the formula above, the mean can be highly random (from very low to very high) thus being of little use for detection. The standard deviation which can describe the local contrast to some degree is also not totally appropriate for reliable object detection. Reliable object detection can be achieved by thresholding the contrast level. In a video, some sub-images containing objects have very low contrast and thereby cannot be found by contrast detection alone. Also, there are some sub-images which might have a high contrast but without any moving objects. To solve this problem, the following formula is used which computes the mask for local contrast salience map M_{CM}:

$$\mathbf{MCM}\ (\mathbf{x},\mathbf{y}) = \begin{cases} \mathbf{1},\ \mathbf{if}\ C^{(p,q)}(\mathbf{x},\mathbf{y}) >= \mathbf{T} \\ \mathbf{0},\ \mathbf{if}\ C^{(p,q)}(\mathbf{x},\mathbf{y}) < \mathbf{T} \end{cases} \qquad \text{.... (3.5.4)}$$

$$\mathbf{I_{CM}\ (x, y) = I(x, y) \cdot M_{CM}} \qquad \text{.... (3.5.5)}$$

where $C^{(p,\ q)}(x, y)$ is the contrast image after local contrast computation, I(x, y) is the original image and $I_{CM}(x, y)$ shows the location of the interested moving object. It is observed that the interested moving objects and image structures (boundaries of illuminated areas) are detected by this step.

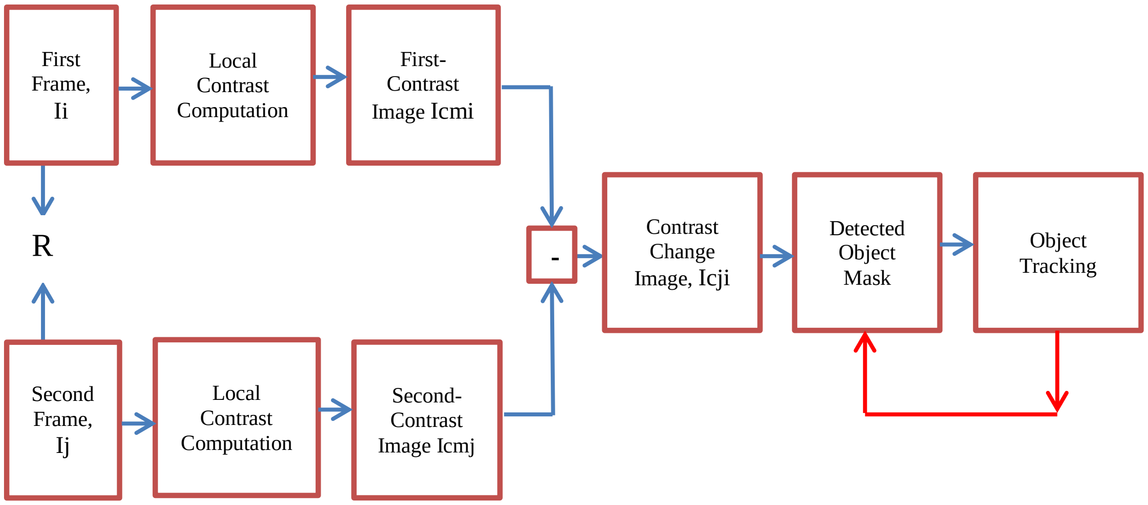

First Frame, Ii
Local Contrast Computation
First-Contrast Image Icmi
R
Second Frame, Ij
Local Contrast Computation
Second-Contrast Image Icmj
-
Contrast Change Image, Icji
Detected Object Mask
Object Tracking

Fig. 3 Algorithm Frame for Object Detection Using Adaptive Contrast Change

D. MOVING OBJECT DETECTION BASED ON CONTRAST CHANGE

The second step of the algorithm involves using the changes in contrast saliency to filter the results of the first step and thus, obtain the moving objects. The formula for this step is as under:

$$\mathbf{MD^{[p,q]}_{Cji} = T\left(\left| I^{[p,q]}_{Cmi} - I^{[p,q]}_{Cmj} \right| \right)} \qquad \text{.... (3.5.6)}$$

$$\mathbf{I^{[p,q]}_{Cji} = I_i \cdot MD^{[p,q]}_{Cji}} \qquad \text{.... (3.5.7)}$$

where $I^{[p,q]}_{Cmi}$ and $I^{[p,q]}_{Cmj}$ are contrast images obtained from the ith and jth image respectively, [p, q] is the window for contrast computation in the first step, $MD^{[p,q]}_{Cji}$ is the masking value for contrast difference image between two contrast images. T is the threshold to remove small changes caused by other factors such as noise or small variations in the light. The image $I^{[p,q]}_{Cij}$ takes large values in regions where the contrast is simultaneously high and changing.

E. LOCALLY ADAPTIVE THRESHOLDING

This class of algorithm involves calculating a threshold at each pixel, which is based on some local statistics like range, variance, or surface-fitting parameters of the pixel neighbourhood. In what follows, the threshold T (i, j) is denoted as a function of the coordinates (i, j) at each pixel.

2.5.2 Adaptive Threshold Change using Local Variance and Local Contrast

The method computes the threshold according to the local mean $\boldsymbol{\mu}$ (i, j) and standard deviation $\boldsymbol{\sigma}$(i, j) and a calculated window size of b×b comparing the gray value of the pixel with the average of the gray values in some neighbourhood window. If the pixel is significantly darker than the average, it is denoted as foreground; otherwise, it is classified as background. In the local method, the threshold is fixed at a mid-range value i.e. the mean of the minimum I_{low}(i, j) and maximum I_{high}(i, j) gray values in a local window. However, if the contrast C (i, j) = I_{high} (i, j) - I_{low} (i, j) is below a certain threshold then that neighbourhood is said to consist only of one class, print or background, which depends on the value of T (i, j).

CHAPTER III: RESULTS

3.1 Background Subtraction Method

3.1.1 Centroid Detection

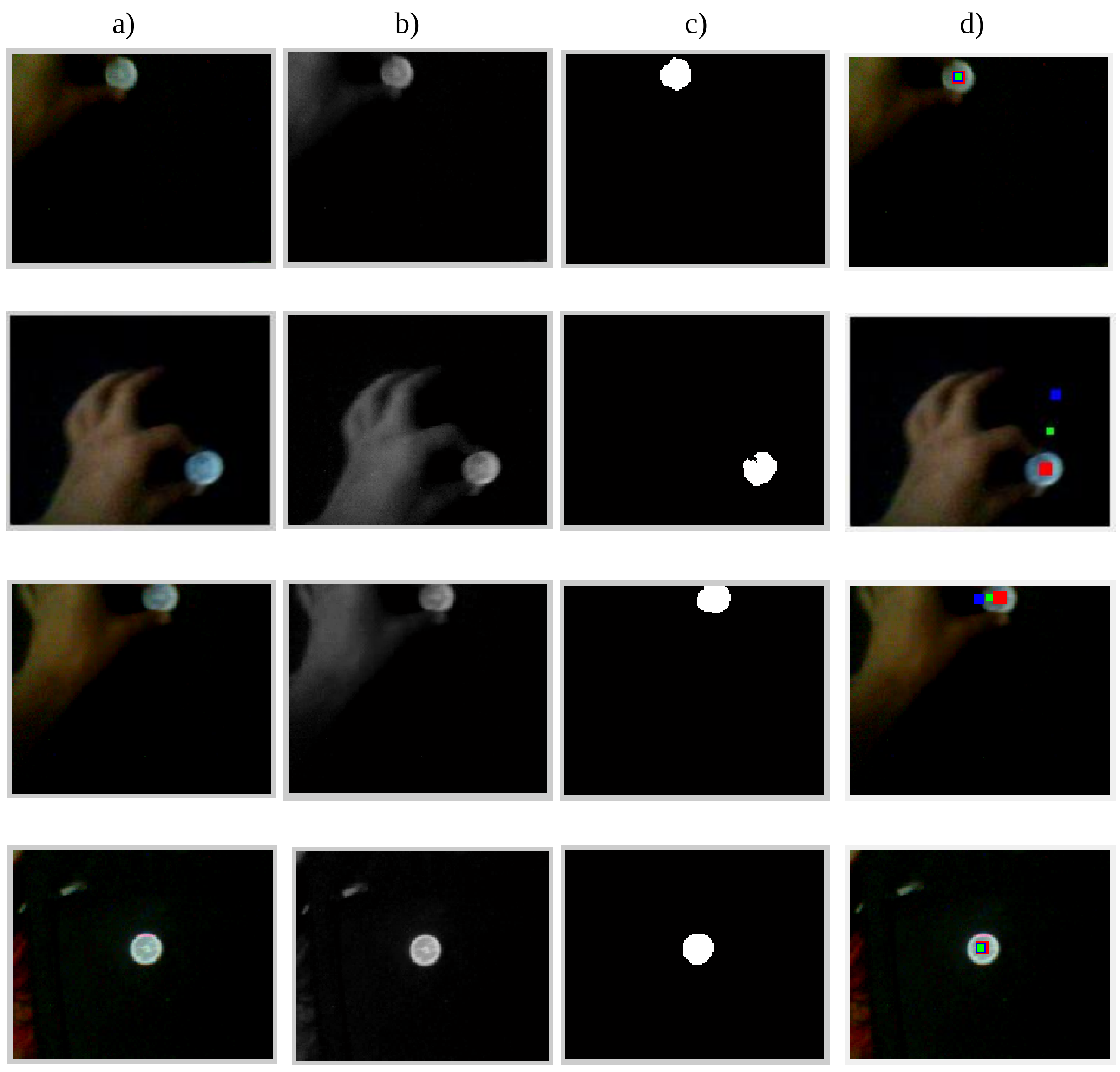


Fig. 4 Output of Background Subtraction Method (Centroid Detection)
a) Actual Image, b) Gray scale Image, c) Binary Image after Background Subtraction, d) Final Output with Centroid detection

3.1.2 Object Detection

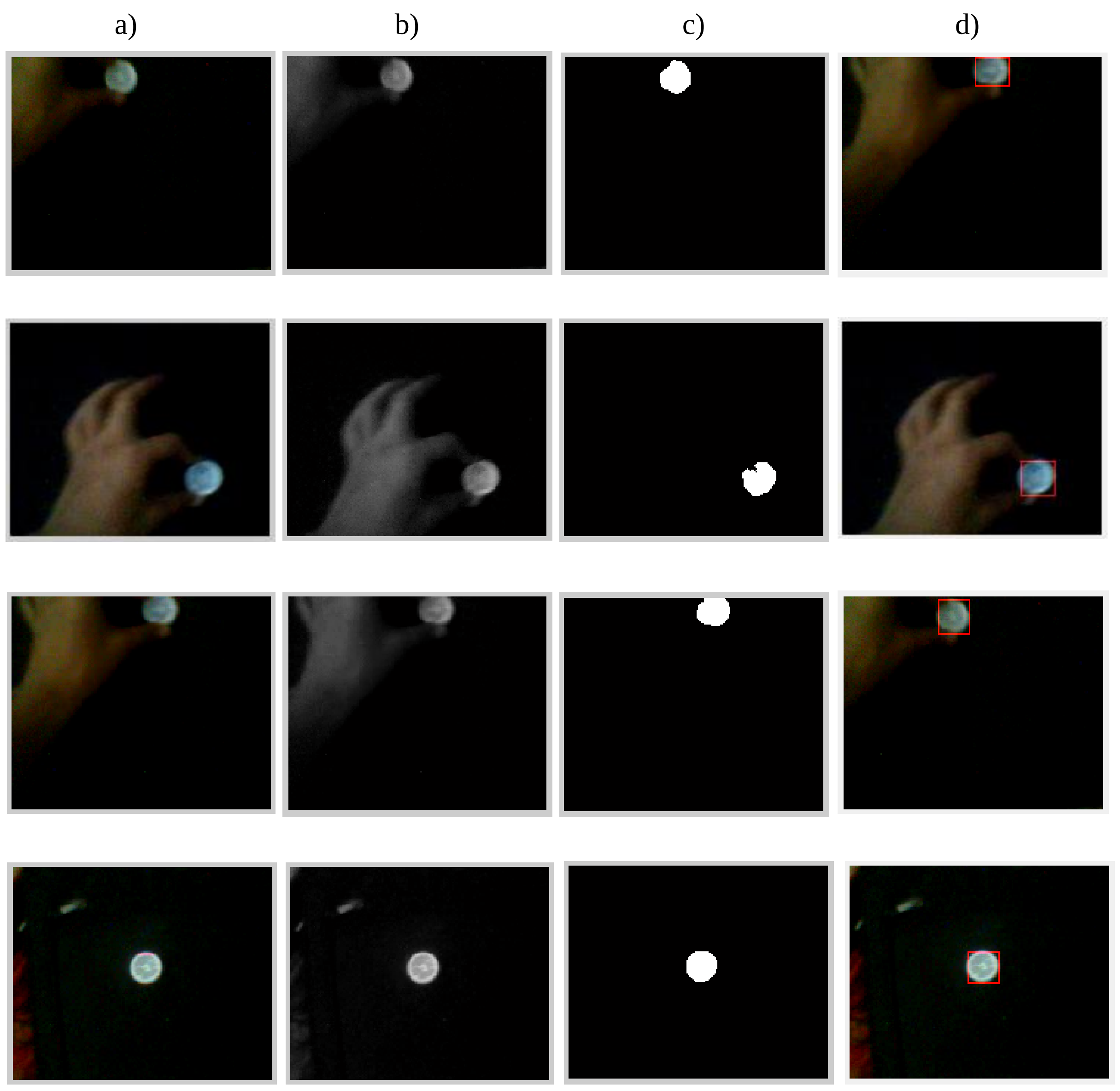


Fig. 5 Output of Background Subtraction Method (Object Detection)
a) Actual Image, b) Gray scale Image, c) Binary Image after Background Subtraction, d) Final Output

3.2 BACKGROUND ESTIMATION METHOD

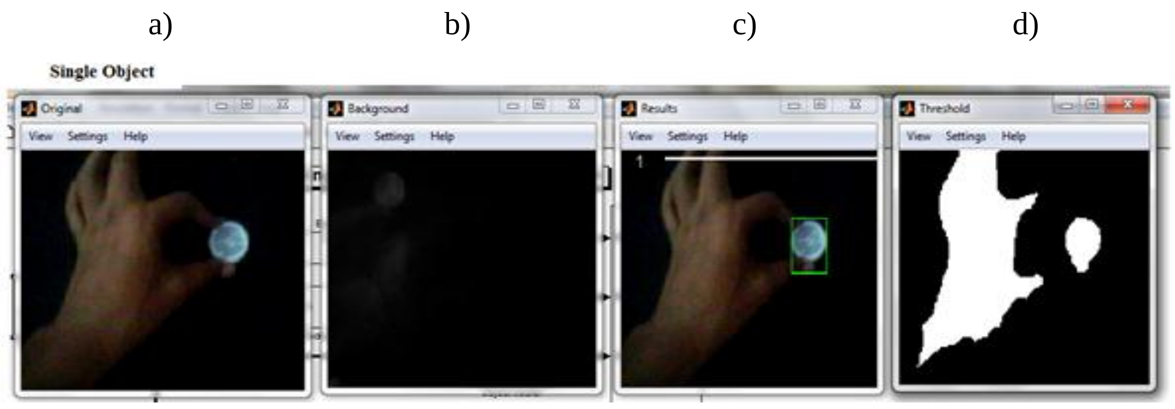


Fig. 6 Output of Background Estimation Method (Single Object)
a) Actual Image, b) Estimated Background, c) Final Output
d) Binary Image after Background Subtraction,

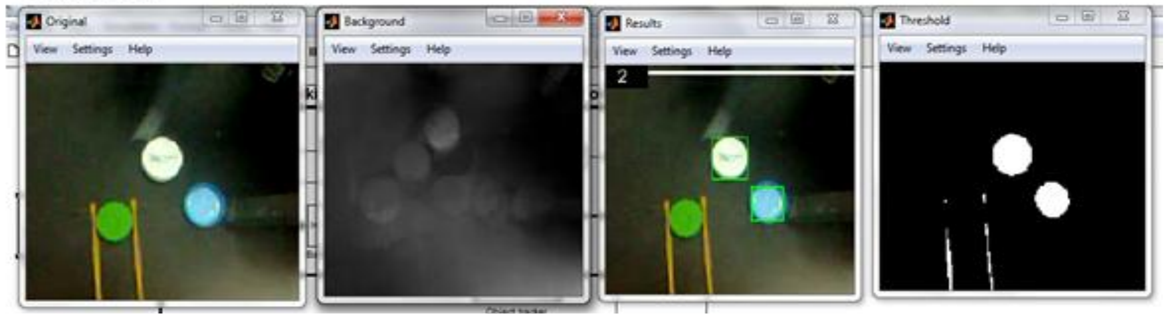


Fig. 7 Output of Background Estimation Method (Multiple Object)
a) Actual Image, b) Estimated Background, c) Final Output
d) Binary Image after Background Subtraction,

3.3 Optical Flow Method

a) b) c) d)

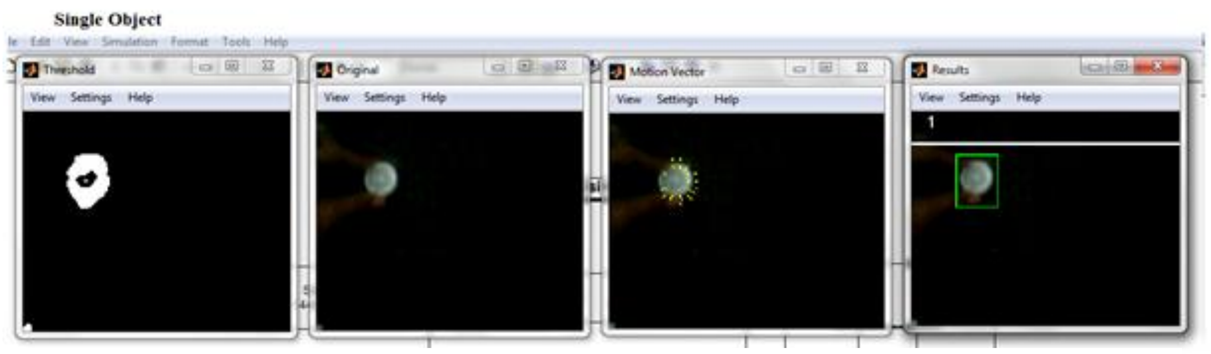


Fig. 8 Output of Background Estimation Method (Single Object)
a) Binary Image after Background Subtraction, b) Actual Image, c) Velocity vectors
d) Final Output

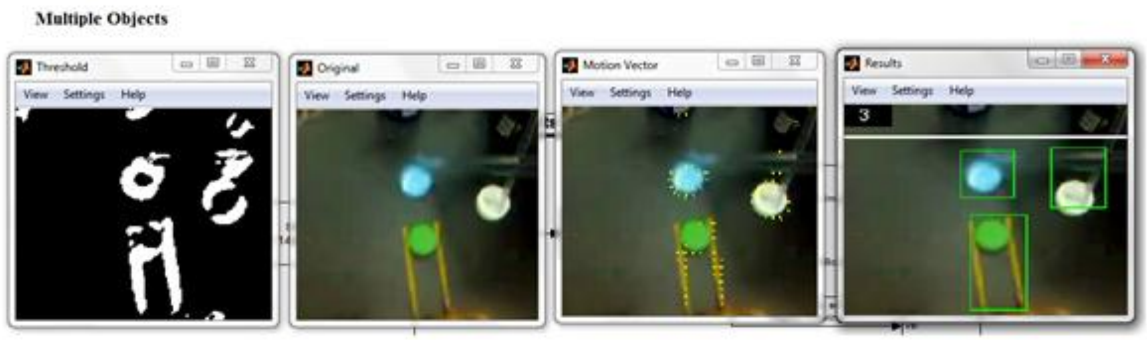


Fig. 9 Output of Background Estimation Method (Single Object)
a) Binary Image after Background Subtraction, b) Actual Image, c) Velocity vectors
d) Final Output

3.4 ADAPTIVE CONTRAST CHANGE METHOD

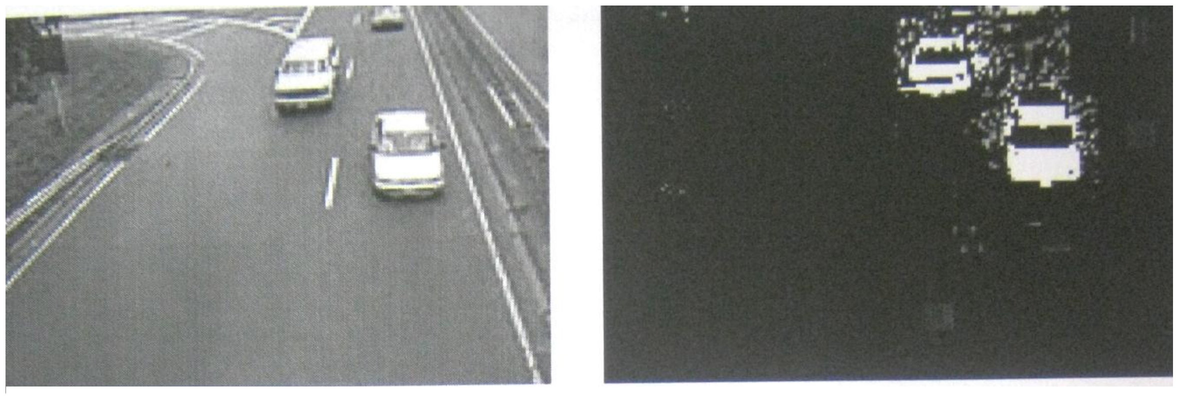


Fig. 10 Output for Light Object without Correlation
Actual Image and Masked Contrast Image for Frame No. 33

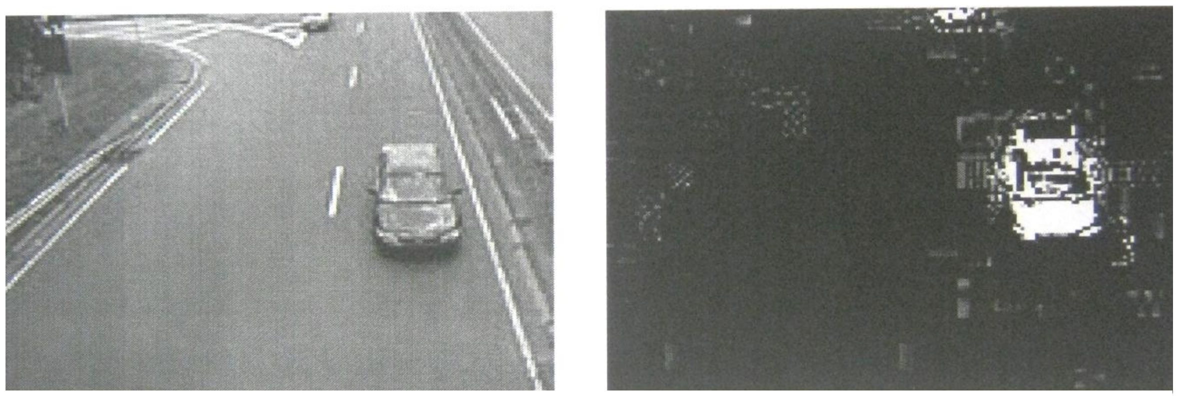


Fig. 11 Output for Dark Object without Correlation
Actual Image and Masked Contrast Image for Frame No. 49

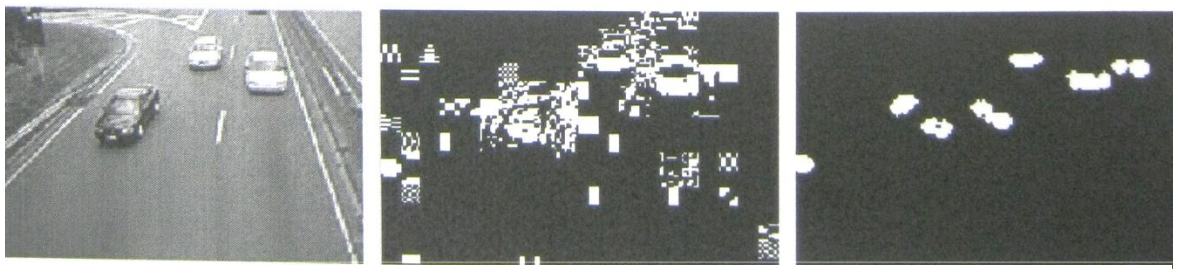


Fig. 12 Output for Dark Object without Correlation
Actual Image, Masked Contrast Image and Binary Image for Frame No. 68

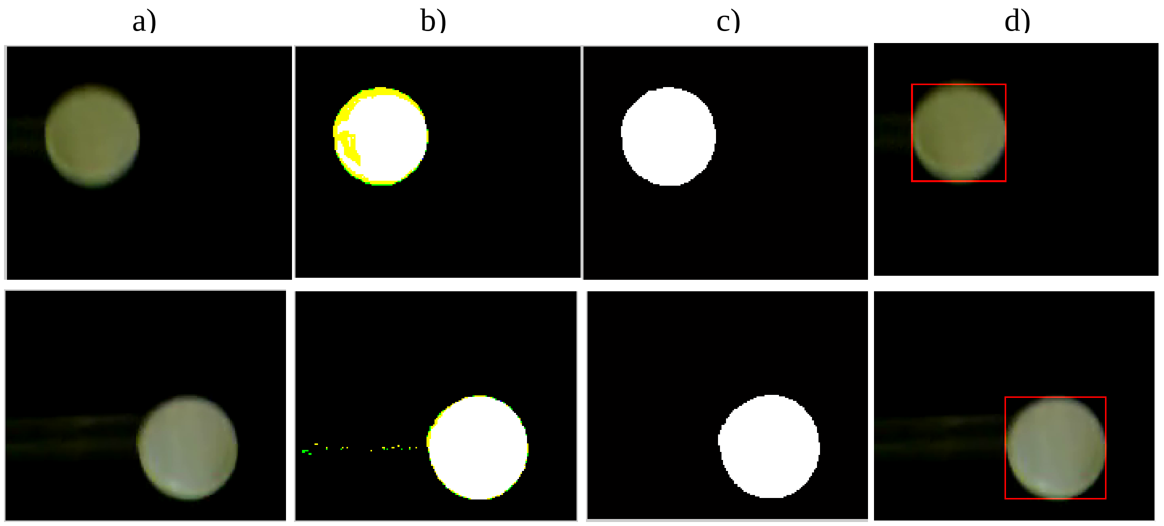


Fig. 13 Output for Single object in Night Light Condition with Correlation a) Actual Image, b) Image after Contrast change detection, c) Binary Image, d) Final Output

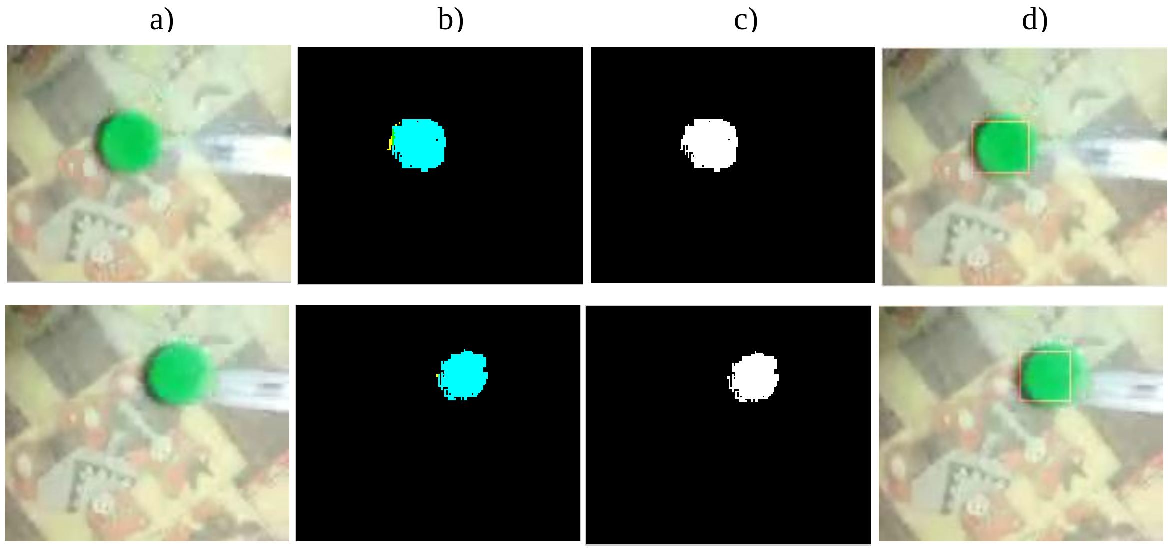


Fig. 14 Output for Single object in Night Light Condition with Correlation a) Actual Image, b) Image after Contrast change detection, c) Binary Image, d) Final Output

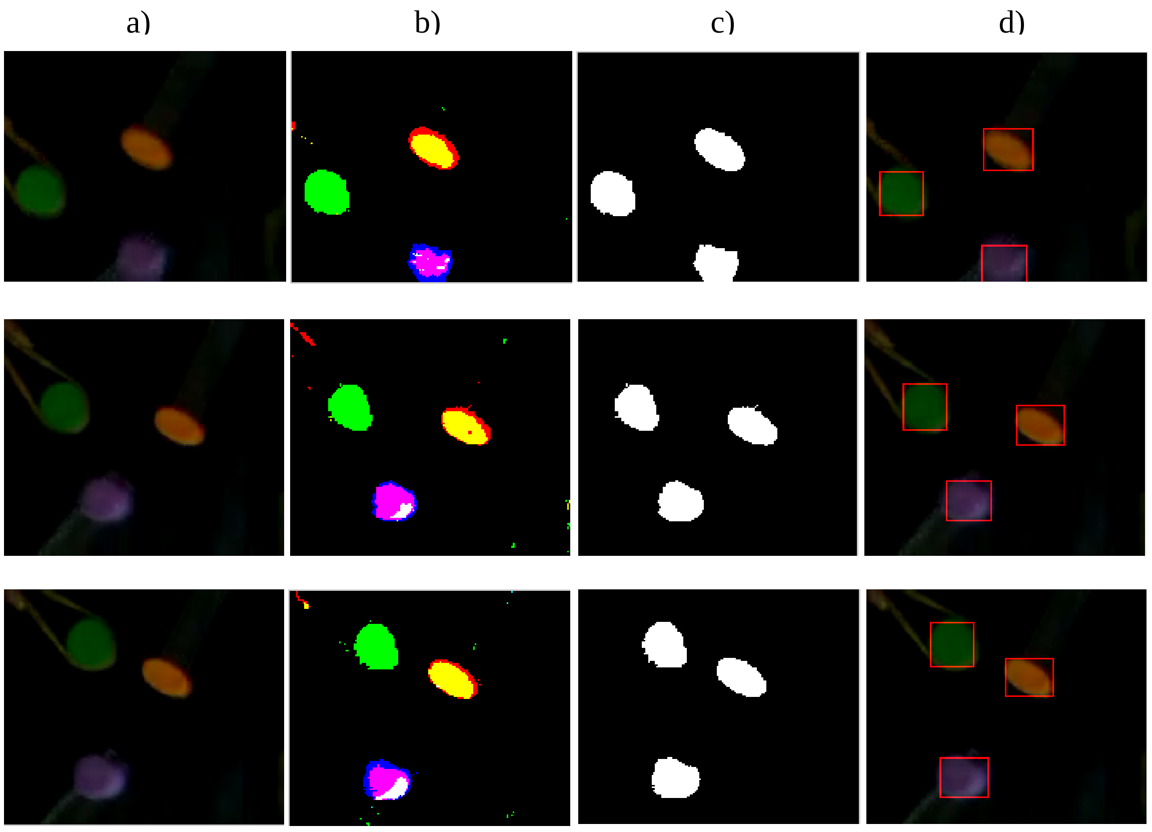


Fig. 15 Output for Multiple objects in Night Light Condition with Correlation a) Actual Image, b) Image after Contrast change detection, c) Binary Image, d) Final Output

CHAPTER IV: CONCLUSION

The objective has been to detect moving objects and thereafter, decide on objects of particular interest which would be tracked. While earlier we worked with object-intrinsic properties such as the centroid of a moving object in order to make a probable prediction of its immediate future motion, methods to detect a rectangular boundary for the object, then used background estimation method using Simulink models and got fair output for single object, but we did not obtain satisfactory results when the methods were worked with multiple objects. Further we made an attempt using the Optical Flow method wherein the Horn-Schunck algorithm for motion estimation was put into effect. The latest method of Adaptive Contrast Change Detection gave satisfactory results in sufficiently reducing the noise while detecting multiple objects. But in some cases it gives unwanted noise. Hence, we have used correlation which basically gives the relation between to frames having significant contrast change. Use of correlation has significantly improved the output and gives better result even with multiple moving objects. The approach seems to have efficient practical applications in poorly-lighted conditions such as night-time visual surveillance systems.

REFERENCES

[1] Liao Ping-Sung, Chen Tse-Sheng, Chung Pau-Choo, "A fast algorithm for multilevel thresholding", J. Inform. Sci. Eng. 2001, vol.17, pp. 713–727

[2] Otsu N., “A thresholding selection using gray level histograms”, IEEE IEEE Trans. Systems Man Cybernetics 1979, vol.9, pp. 62-69

[3] Aggarwal A., Biswas S., Singh S.,” Object tracking using background subtraction and motion estimation in MPEG videos”, ACCV 2006, vol.2, pp. 121-130

[4] Zhulin Li, Chao Xu, Yan Li, "Robust object tracking using mean shift and fast motion estimation", Intelligent Signal Processing and Communication Systems, ISPACS 2007, pp. 734 – 737

[5] Horn B.K.P., Schunck B.G., "Determining optical flow", Artificial Intelligence 1981, vol.17, pp. 185-203

[6] Sherin M. Youssef, Meer A. Hamza, Arige F. Fayed, **"**Hybrid wavelet-based video tracking using adaptive contrast change detection in night-time visual surveillance systems**",** WEC 2010, vol.2183, pp. 732 – 737

[7] Huanga K., Wanga L., Sana T., Tana T., Maybankb S., “A real-time object detecting and tracking system for outdoor night surveillance”, Pattern Recognition, ELSEVIER 2008, vol.41, pp. 432-444

[8] McKenna S. J., Raja Y., Gong S., “Tracking colour objects using adaptive mixture models,” Image Vision Comput 1999, vol.17, pp. 225–231

APPENDIX A:

MATLAB PROGRAM CODES FOR BACKGROUND SUBSTRACTION

1. CENTROID DETECTION USING BACKGROUND SUBTRACTION

Aim: To detect centroid of the moving object in a video file.

Code:

```
function [] = detectcen()

clc;
objectObj = mmreader('samplethr.avi');
get(objectObj)

nframes = get(objectObj, 'NumberOfFrames');
I = read(objectObj, 1);
taggedObjects = zeros([size(I,1) size(I,2) 3 nframes], class(I));
for k = 2 : 1 : nframes
    Backgroundimage1 = read(objectObj, k-1);
    Currentimage1 = read(objectObj, k);
    Backgroundimage2 = rgb2gray(Backgroundimage1);
    Currentimage2 = rgb2gray(Currentimage1);
    Backgroundimage2 = BackSub(Backgroundimage2);
    Currentimage2 = BackSub(Currentimage2);
    taggedObjects(:,:,:,k) = Currentimage1;

    stats = regionprops(Backgroundimage2, {'Centroid','Area'});
    c1 = centroid(stats);

    stats = regionprops(Currentimage2, {'Centroid','Area'});
    c2 = centroid(stats);

    width = 4;
    row = c2(1)-width:c2(1)+width;
    col = c2(2)-width:c2(2)+width;
    taggedObjects(row,col,1,k) = 255;
    taggedObjects(row,col,2,k) = 0;
    taggedObjects(row,col,3,k) = 0;

    width = 3;
    row = c1(1)-width:c1(1)+width;
    col = c1(2)-width:c1(2)+width;
    taggedObjects(row,col,1,k) = 0;
    taggedObjects(row,col,2,k) = 0;
    taggedObjects(row,col,3,k) = 255;

    c3 = (c1 + c2)/2;
    c3 = floor(c3);
    width = 2;
    row = c3(1)-width:c3(1)+width;
    col = c3(2)-width:c3(2)+width;
    taggedObjects(row,col,1,k) = 0;
    taggedObjects(row,col,2,k) = 255;
    taggedObjects(row,col,3,k) = 0;
end
frameRate = get(objectObj,'FrameRate');
implay(taggedObjects,frameRate);
end


function [c] = centroid( stats )

c = [2 2];
if ~isempty([stats.Area])
areaArray = [stats.Area];
[~,idx] = max(areaArray);
c = stats(idx).Centroid;
c = floor(fliplr(c));
end
end

function [I] = BackSub(I)
%set value for dark objects..
darkObjectValue = 93;
sedisk = strel('disk',2,8);
% Remove dark objects.
noDarkObjects = imextendedmax(I, darkObjectValue);
noSmallStructures = imopen(noDarkObjects, sedisk);
% Remove small structures.
I = bwareaopen(noSmallStructures, 60);
end
```

2. Object Detection using Background Subtraction Method

Aim: To detect the moving object in a video file.

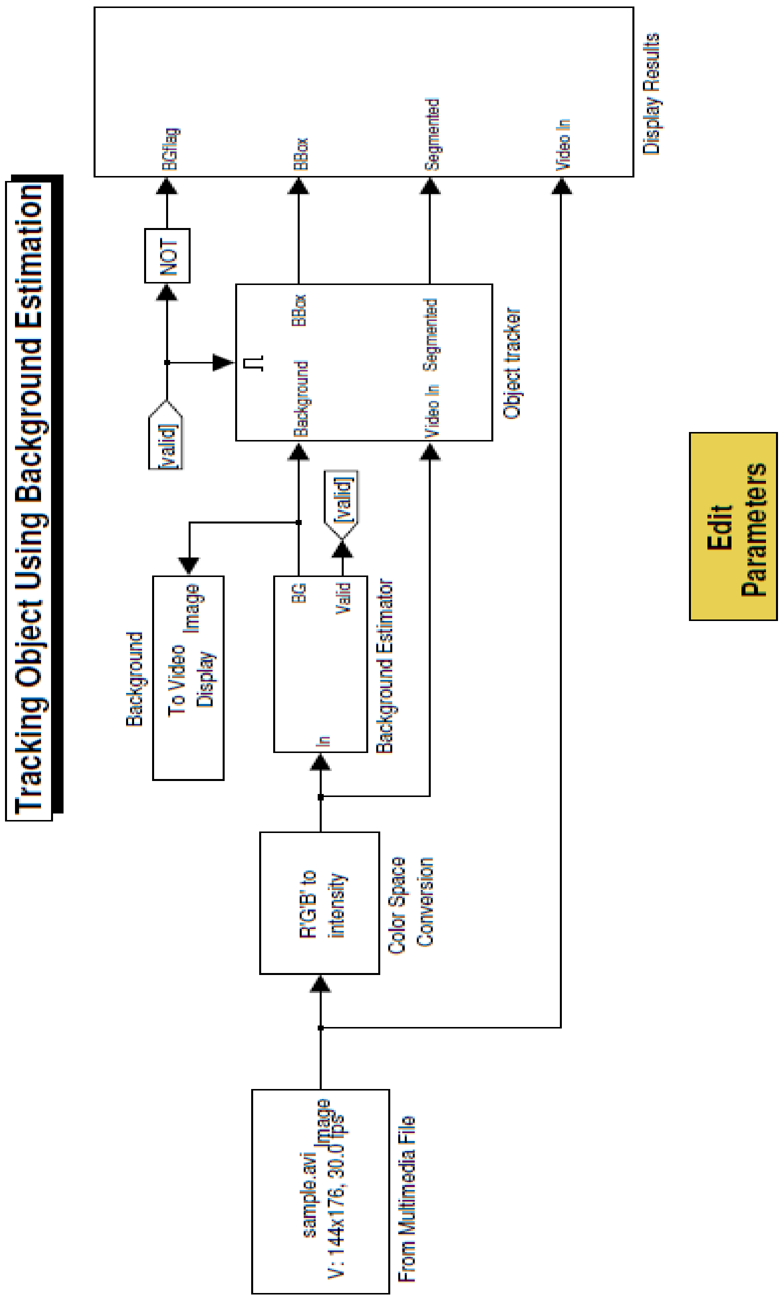
Code:

```
function [] = detectmov()
clc;
objectObj = mmreader('samplethr.avi');
get(objectObj)

nframes = get(objectObj, 'NumberOfFrames');
I = read(objectObj, 1);
ObjDetect = zeros([size(I,1) size(I,2) 3 nframes], class(I));

% Read Frames
for k = 2 : 1 : nframes
    Currentimage1 = read(objectObj, k);
    Currentimage2 = rgb2gray(Currentimage1);
% Perform Background Substraction
    Currentimage3 = BackSub(Currentimage2);
% Get the area and BoundingBox of each remaining object in the
frame. The

    ObjDetect(:,:,:,k) = Currentimage1;
%Use regionprops to get the region properties
    stats = regionprops(Currentimage3, {'Area', 'BoundingBox'});
%Bounding box around the image
    if ~isempty([stats.Area])
        areaArray = [stats.Area];
        [~,idx] = max(areaArray);
        [M ~] = size(stats(idx).BoundingBox);
        for h = 1:M
            for                          i                          =
ceil(stats(h).BoundingBox(1)):(floor(stats(h).BoundingBox(1))+floor(
stats(h).BoundingBox(3)))
                for                          j                      =
ceil(stats(h).BoundingBox(2)):(floor(stats(h).BoundingBox(2))+floor(
stats(h).BoundingBox(4)))
                    x1 = ceil(stats(h).BoundingBox(1));
                    y1 = ceil(stats(h).BoundingBox(2));
                    ObjDetect(y1,i,2,k) = 255;
ObjDetect(floor(stats(h).BoundingBox(2)+stats(h).BoundingBox(4)),i,2
,k) = 255;
                    ObjDetect(j,x1,2,k) = 255;

ObjDetect(j,floor(stats(h).BoundingBox(1)+stats(h).BoundingBox(3)),2
,k) = 255;
                end
            end
        end
    end
end

%Run the following commands to see ObjDetect in implay.
frameRate = get(objectObj,'FrameRate');
implay(ObjDetect,frameRate);
end

function [I] = BackSub(I)

%set value for dark objects..
darkObjectValue = 93;
sedisk = strel('disk',2,8);
% Remove dark objects.
noDarkObjects = imextendedmax(I, darkObjectValue);
noSmallStructures = imopen(noDarkObjects, sedisk);
% Remove small structures.
I = bwareaopen(noSmallStructures, 60);
End
```

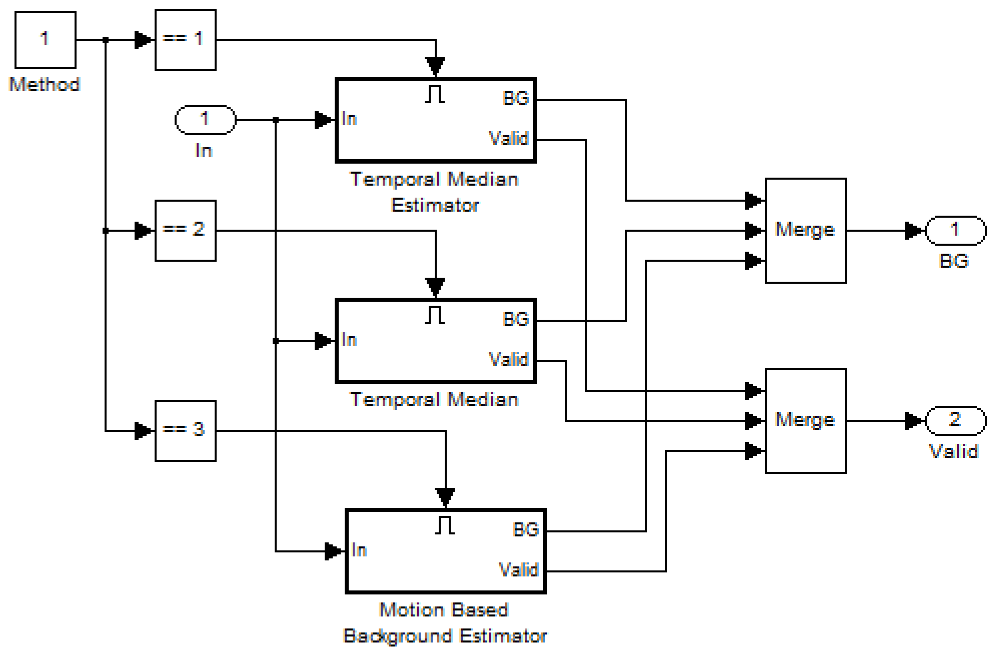
APPENDIX B:

SIMULINK MODEL FOR BACKGROUND ESTIMATION METHOD

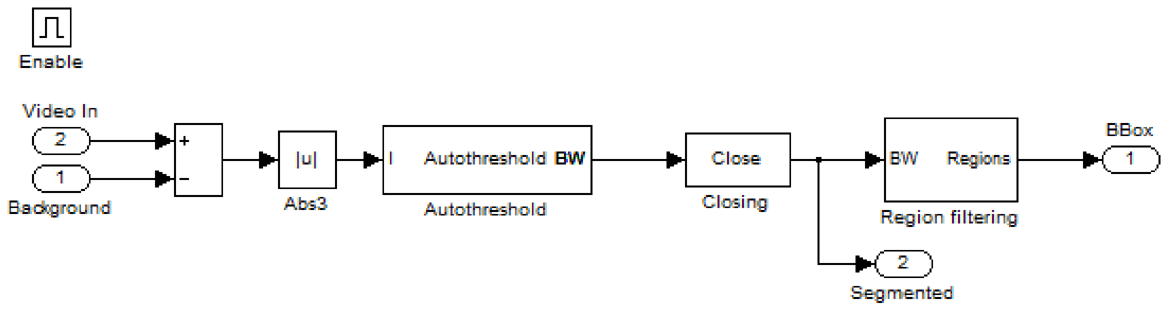
1. **OBJECT DETECTION USING BACKGROUND ESTIMATION**



i. Background Estimator



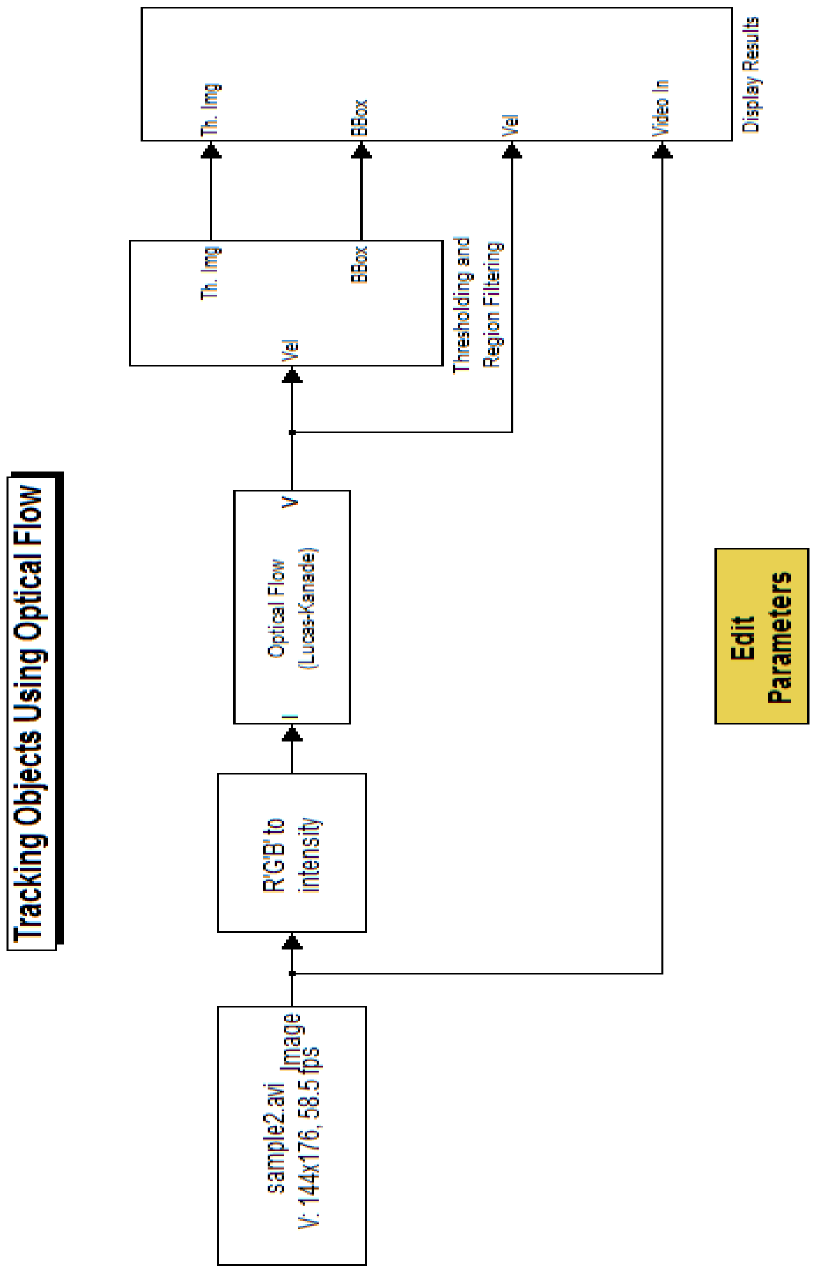
ii. Object Tracker



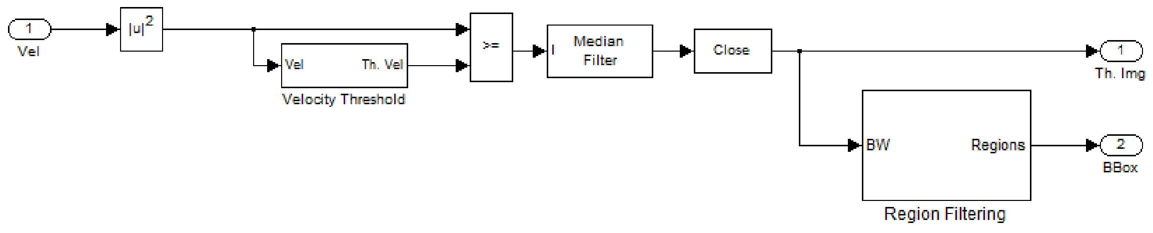
APPENDIX C:

SIMULINK MODEL FOR Optical Flow

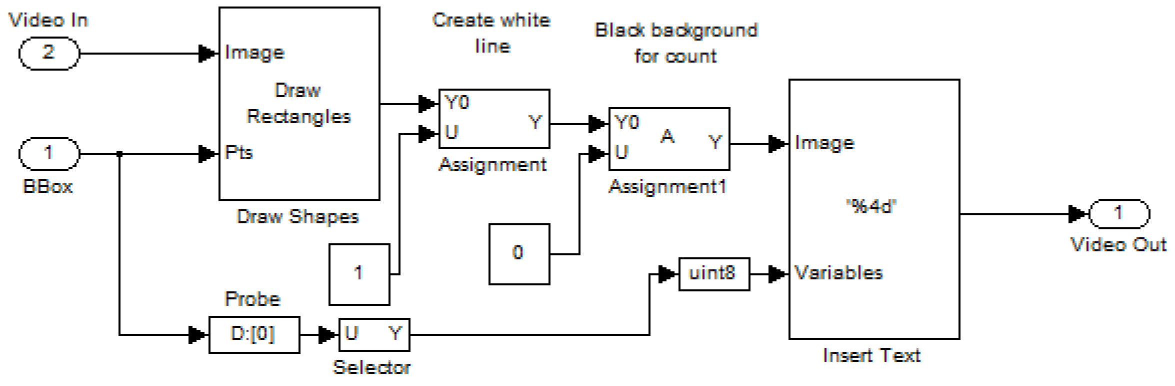
1. **OBJECT DETECTION USING Optical Flow**



i. Thresholding and Region Filtering



ii. Bounding Box(Display Results)



APPENDIX D:

MATLAB PROGRAM CODES FOR Adaptive CONTRAST CHANGE DETECTION

1. **OBJECT DETECTION IN NIGHT LIGHT USING ADAPTIVE CONTRAST CHANGE DETECTION**

Code:

```
function [ ] = Imgconchange( )

clc;
objectObj = mmreader('samplenn.avi');
get(objectObj)

nframes = get(objectObj, 'NumberOfFrames');
I = read(objectObj, 90);
S = size(I);
Icji = zeros(S);
ObjDetect = zeros([S(1) S(2) 3 nframes], class(I));

for k = 40:1:43
    Ii = read(objectObj, k);
    Ij = read(objectObj, k-1);
    R=Matrixcorr(Ii,Ij);
End

disp('Correlation Completed...');

k = R;
while(k<nframes)
    k=k+1;
    clc;
    disp(['Processing.... ', num2str(ceil((k/(nframes/100)))),'%']);
    Ii = read(objectObj, k);
    Ij = read(objectObj, k-R);
    Icmi = Contrastimage(Ii);
    %figure, imshow(Icmi);
    Icmj = Contrastimage(Ij);
    %figure, imshow(Icmj);

    for p =1:1:S(1)
        for q = 1:1:S(2)
```

```
            MDcji = 0.05*(abs(Icmi(p,q) - Icmj(p,q)));
            if(MDcji>0.02)
                MDcji = 0.01;
            else if(MDcji==0)
                    MDcji = 0.05;
                end
            end
            Icji(p,q,:) = Ii(p,q,:)*MDcji;
        end
    end

    DOV=450;
    if(max(max(sum(Ii == 0)))<80)
        for p =1:1:S(1)
            for q = 1:1:S(2)
                if((Icji(p,q,:)>=1))
                    Icji(p,q,:)=0;
                end
            end
        end
        DOV=300;
    end

    Icjif = BackSub(Icji,DOV);
    ObjDetect(:,:,:,k) = Ii;

    bw = bwlabel(Icjif,8);
    stats = regionprops(bw,'BoundingBox');

    [M ~] = size(stats);
    for h = 1:M
        for                                     i                                =
ceil(stats(h).BoundingBox(1)):(floor(stats(h).BoundingBox(1))+floor(
stats(h).BoundingBox(3)))
            for                                     j                            =
ceil(stats(h).BoundingBox(2)):(floor(stats(h).BoundingBox(2))+floor(
stats(h).BoundingBox(4)))
                x1 = ceil(stats(h).BoundingBox(1));
                y1 = ceil(stats(h).BoundingBox(2));
                ObjDetect(y1,i,1,k) = 255;

ObjDetect(floor(stats(h).BoundingBox(2)+stats(h).BoundingBox(4)),i,1
,k) = 255;
                ObjDetect(j,x1,1,k) = 255;
ObjDetect(j,floor(stats(h).BoundingBox(1)+stats(h).BoundingBox(3)),1
,k) = 255;
            end
        end
    end
end

disp('Processing Completed.. --> Output');

frameRate = get(objectObj,'FrameRate');
implay(ObjDetect,frameRate);
end


function [ RGBcm ] = Contrastimage( RGB )

S = size(RGB);
RGBc = Matrixcontrast(RGB);
RGBcmn = mean(RGBc,3);
RGBcm = zeros(S(1),S(2),3);

for i=1:1:S(1)
    for j=1:1:S(2)
        if(isnan(RGBcmn(i,j)) || isinf(RGBcmn(i,j)))
            RGBcm(i,j,:) = 0;
        else
            RGBcm(i,j,:) = RGB(i,j,:)*1;
        end
    end
end

RGBcm = mean(RGBcm,3);
end



function [R] = Matrixcorr(A, B)

S = size(A);
C = 0;
D = 0;
E = 0;
X=(S(1)*S(2));
```

```
for i=1:1:S(1)
    for j=1:1:S(2)
          C = C + A(i,j);
          D = D + B(i,j);
          E = E+(A(i,j)-C/X)*(B(i,j)-D/X);
    end
end

R = floor(abs(E/255)*pi^2);
end



function [ C ] = Matrixcontrast( A )

S = size(A);
B = Matrixmean(A);
C = zeros(size(A));
for i=1:1:S(1)
    for j=1:1:S(2)
          C(i,j)=(A(i,j)-B(i,j))^2;
    end
end

D = Matrixmean(C);
for i=1:1:S(1)
    for j=1:1:S(2)
          D(i,j)=D(i,j)^0.5;
    end
end

E = zeros(S);
for i=1:1:S(1)
    for j=1:1:S(2)
        E(i,j)=D(i,j)/B(i,j);
    end
end
end
```

```
function [ C ] = Matrixmean( A )

S = size(A);
B = zeros(S(1)+4,S(2)+4);
C = zeros(size(A));
for i=3:1:S(1)+2
    for j=3:1:S(2)+2
          B(i,j)=A(i-2,j-2);
    end
end

for i=3:1:S(1)+2
    for j=3:1:S(2)+2
        x=0;
        for k=i-2:1:i+2
              for l=j-2:1:j+2
                  x=x+B(k,l);
              end
        end
        C(i-2,j-2)=x/25;
    end
end
end



function [I] = BackSub(IMG,DOV)

I = rgb2gray(IMG);
sedisk = strel('disk',0);
noSmallStructures = imopen(I, sedisk);
% Remove small structures.
I = bwareaopen(noSmallStructures, DOV);
end
```