Sudan University of Science and Technology
College of Graduate Studies

Motion Detection Algorithm Using Open Computer Vision

خوارزمية كشف الحركة بإستخدام المكتبة البرمجية المفتوحة للرؤية الحاسوبية

A thesis submitted in partial fulfillment for the award of the degree of M.Sc. in Computer and Network Engineering

Prepared by:

Mohammed Naeem Altyp Osman

Supervisor:

Dr. Ibtihal Haidar Gismallah Yousif

August 2020

الإستهلال

بسم الله الرحمن الرحيم

﴿ عَلَّمَ الإنْسَانَ مَالَمْ يَعْلَمْ (5) ﴾

العلق (5)

صدق الله العظيم

DEDICATION

To

The source of tenderness

My mother

To

All My family

To

Who Paved the way of science for me

my Teachers

To

All my friends

ACKNOWLEDGMENT

First, I would like to thank God Almighty and all his blessings and blessings to complete this research.

I would like also to thank Dr. Ebtihal Haidar for her unlimited support and guide me to the right path to complete this research.

Finally, I would like to thank my family and all my friends who helped me to complete this research.

Abstract

Monitoring areas such as security and surveillance are key areas in all aspects of our life in homes, roads, work areas, military zones, and other areas. Surveillance cameras have emerged to cover this field and have produced good results, but these cameras need people during the surveillance time to monitor the video from the cameras, making the monitoring process expensive and tedious. Motion detection system was designed for this purpose to detect any motion occurred in the area and then gives alarm signal to notify by the movement. This project was designed using the open computer vision (Open cv) library under Microsoft Visual Studio and tested with universal serial bus (USB) camera. The project gave good results especially in extracting the background and updating it by doing three tests, two of them using pre-recorded video files, the result of project was good because it detected almost all the movable targets in the video approximately (99.9%) detectable objects. The third test was a live video from webcam cameras on the laptop and another external USB camera connected to the computer and the project was tested and gave good results.

المستخلص

تعتبر مجالات المراقبة من المجالات الرئيسية في جميع جوانب حياتنا مثل المنازل ،الطرق ، مناطق العمل ،المناطق العسكرية ومناطق أخرى. لقد ظهرت كاميرات المراقبة لتغطية هذا المجال وحققت نتائج جيدة ، لكن هذه الكاميرات تحتاج إلى أشخاص خلال فترة المراقبة لمراقبة الفيديو من الكاميرات ، مما يجعل عملية المراقبة باهظة الثمن ومملة. تم تصميم نظام الكشف عن الحركة لهذا الغرض للكشف عن أي حركة حدثت في المنطقة ومن ثم يعطي إشارة إنذار للإعلام بالحركة. تم تصميم هذا المشروع باستخدام مكتبة رؤية الكمبيوتر المفتوحة. أعطى المشروع نتائج جيدة خاصة في إستخراج الخلفية وتحديثها عن طريق إجراء ثلاثة اختبارات ، إثنان منها باستخدام ملفات الفيديو المسجلة مسبقًا ، وكانت نتيجة المشروع جيدة لأنها اكتشفت جميع الأهداف المنقولة في الفيديو تقريبًا (99.9٪) أشياء قابلة للكشف. الاختبار الثالث كان فيديو مباشر من كاميرات كاميرا ويب على الكمبيوتر المحمول وكاميرا خارجية أخرى متصلة بالكمبيوتر بالناقل التسلسلي العام وتم اختبار المشروع وقدم نتائج جيدة .

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two Dimension
CAV	Component Analog Video
CCTV	Closed-Circuit Television
CV	Computer Vision
EPZS	Export Processing Zones
FPGA	Field-Programmable Gate Arrays
FT	Frequency-Tuned
GSM	Global System For Mobile
HDL	Hardware Description Language
HLS	High Level Synthesis
MAT LAB	Matrix Laboratory
MPEG-4	Moving Picture Experts Group
NTSC	National Television System Committee
PAL	Phase Alternating Line
PIC	Peripheral Interface Controller
PTZ	Pan–Tilt–Zoom Camera
QAM	Quadrature Amplitude Modulation
SECAM	Sequential Color with Memory
SIP	Spatial Image Partition
SSB-L	Subcarrier Sideband
STP	Spatiotemporal Projection
TMD	Temporal Motion Detection
TV	Television
UN	United Nation

USB	Universal Serial Bus
VSB	Vestigial Side Band
GUI	Graphic User Interface
OS	Operating System
V3	Version Three

CHAPTER ONE

INTRODUCTION

1.1 Overview

The video is a sequence of images changes with time. The image sometimes called frame in video concept so the video rate measure number frames per second which indicate the number of images taken in one second such as 25 frames per second and 30 frames per second, these two rates are the most common rates used in the video. Video signals were being analog or digital signals, analog signal usually used for transmitting or recording as TV video, but the digital signals are used on the video processing such as enhancement, compressing … etc. [1].

Analog video signals obtained by periodic sampling (scanning), there are two methods of scanning video progressive scanning and interlaced scan.

In progressive scanning method complete frame will be traced each second. The computer industry is an example of this method with time t=1/72 second for monitors [1].

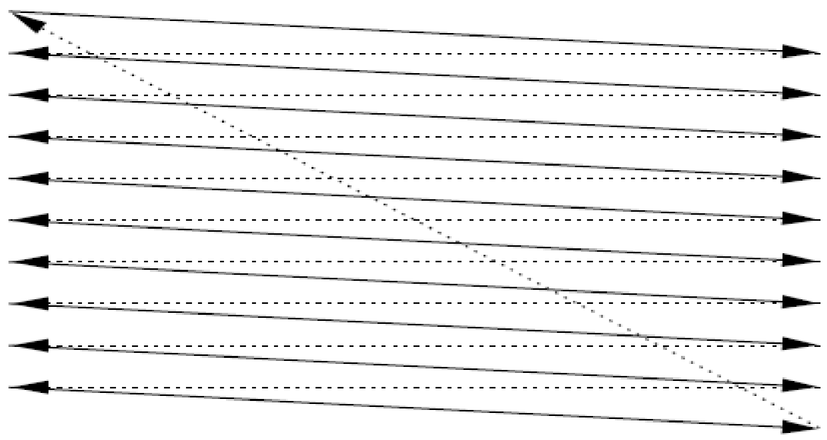


Figure 1.1: Progressive Tv Scanning

But the interlaced scanning method use the odd and even lines to scan the picture, that means this method will separate the picture (frame) into two parts odd parts and even parts, each part called field. Also, this method has a horizontal retraced to switch from field to another. TV video signals are the example of this method [1].

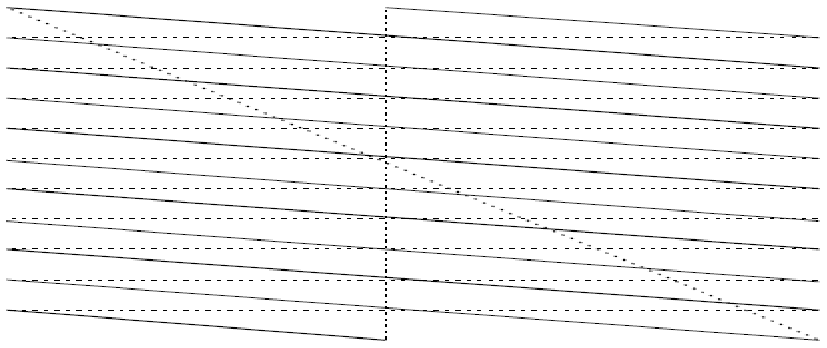


Figure 1.2: Interlaced TV scanning

1.2 Analogue Videos Standard

There are three standards for analogue video witch differ in image parameter and also differ in the method to process the image color. These can classified as follow.

1.2.1 Component Analog Video (CAV)

This standard uses Three signals to represent all components of video, one signal for luminance gray level (Y Signal), and two signal for chrominance colored signals (I & Q signals).

$$Y=0.3R+0.59G+0.11B \quad (1.1)$$

$$I=0.60R+0.28G-0.32B \quad (1.2)$$

$$Q=0.21R-0.52G+0.31B \quad (1.3)$$

This standard requires synchronization of three video components with time [1].

1.2.2 Composite Video

All components of video are collecting together with one signal in this standard, there are three signals format using this standard [1].

1.2.2.1 National Television System Committee (NTSC)

This standard used in North America and Japan, it is composite video standard it depends on interlaced video scanning method with 30 frames per second and 4:3 aspect ratio.

The signal of NTSC has 6MHz of bandwidth and it composed from Vesical Side Band modulation (VSB) for luminance signal(Y), quadrature amplitude modulation (QAM) for (Q) channel, subcarrier sideband modulation (SSB-L) for (I) channel and frequency modulation for the audio channel [1].

1.2.2.2 Phase Alternation Line (PAL)

European standards also use interlaced scanning method with 50 frames per second, 625 lines per frame, and 2:1 interlaced. The band width of signal is 8MHz, with a color subcarrier at 4.43 MHz above the picture carrier. the both chrominance signals (Q and I) are used Quadrature Amplitude Modulation (QAM). The main difference between PAL and SECAM in the integration color signals (chrominance) with the luminance signal [1].

1.2.2.3 System Electronic Color Avec Memoire (SECAM)

The chrominance signals Cr and Cb are transmitted alternatively on successive scan lines. They are used FM modulated on color subcarriers 4.25

MHz and 4.41 MHz for Cb and Cr, respectively. Since only one chrominance signal will be transmitted on line, there is no interference between the chrominance components [1].

1.2.3 S-Video (Y/C video)

combination between the composite video and the component analog video, which represent the video as two component signals, a luminance and a composite chrominance signal, The chrominance signal can be based upon the (I,&) or (Cr, Cb) representation for NTSC, PAL, or SECAM systems.
S-video is currently being used in consumer quality video cassette recorders and camcorders to give image quality better than the composite video [1].

1.3 Problem Statement

In several fields sometimes we need to detect the motion of objects in an important place to avoid many problems such as thieve and car traffic, monitoring place. In the field of using CCTV camera this feature does not support in security cameras, this feature will add new technique in security fields.

1.4 Objectives

The main objective of this thesis is to design motion detection algorithm from life video using computer vision environment open CV with increase accuracy.

1.5 Methodology

This project will use a motion detection algorithm by two parts, the first part is algorithm for extract the fixed background from the video and then apply this algorithm for recorded videos, life webcam video and live video from external camera to ensure this algorithm will extract real background video properly, then the second part applies the motion detection algorithm and deliver the output video as binary video (Black and white). This algorithm will apply by the computer vision library (Open CV) on visual studio C/C++ software figure 1.3 below explain the main steps to implement the propose solution.

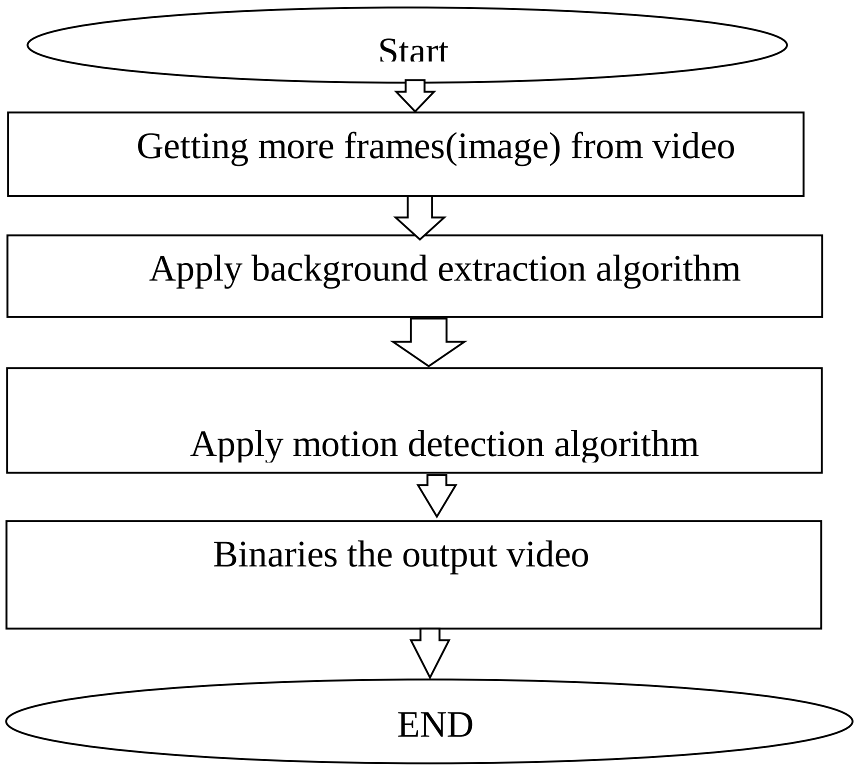

Start
Getting more frames(image) from video
Apply background extraction algorithm
Apply motion detection algorithm
Binaries the output video
END

Figure 1.3: Flow Chart for Motion Detection Algorithms

1.6 Thesis organization

The Research will be organized as a follows chapter2 discusses the literature review of video processing techniques. Chapter3 discusses the methodology that will apply in this project by more details (explain all algorithms will have applied in this project) and software and tools that will be used. Chapter 4 present the result of this project and finally chapter 5 present the conclusion and recommendation.

CHAPTER TWO

LITERATURE REVIEW

This chapter covers the related work for this research in different topics by using different methods to solve this problem by using different tools, software and algorithm.

2.1 Background

The idea of the research has been achieved by a lot of researchers and designer, for this project will used the different methods to extract the background video will be discussing that will expect to enhance the output of background video which will improve the final output, then border will be adding to input video (high quality) around the moving objects as output video in addition of the binary output video.

This project can use at security systems field which will give more reliable system, no need for human check the video at all time, activate the alarm system when the camera reaches the un wanted view and can generate record signal only if the motion occurred in monitoring place which will save many memory space bytes to store the monitoring video (low cost).

2.2 background Subtraction Methods

Video processing applications became very important in surveillance field, traffic flow, security systems and computer vision etc. Computer vision has the basic tools of detect live moving object. For improve the real time moving object detection, motion detection algorithm is applied based on the background subtraction algorithm with many steps grey level background video, color reduction technique is applied to both the background and input video, both filtered videos are subtracted using image subtraction, the gravity center of the object image is calculated and sent to a computer via serial interface and Sobel edge detection algorithm is used to identify motion object's edges [2].

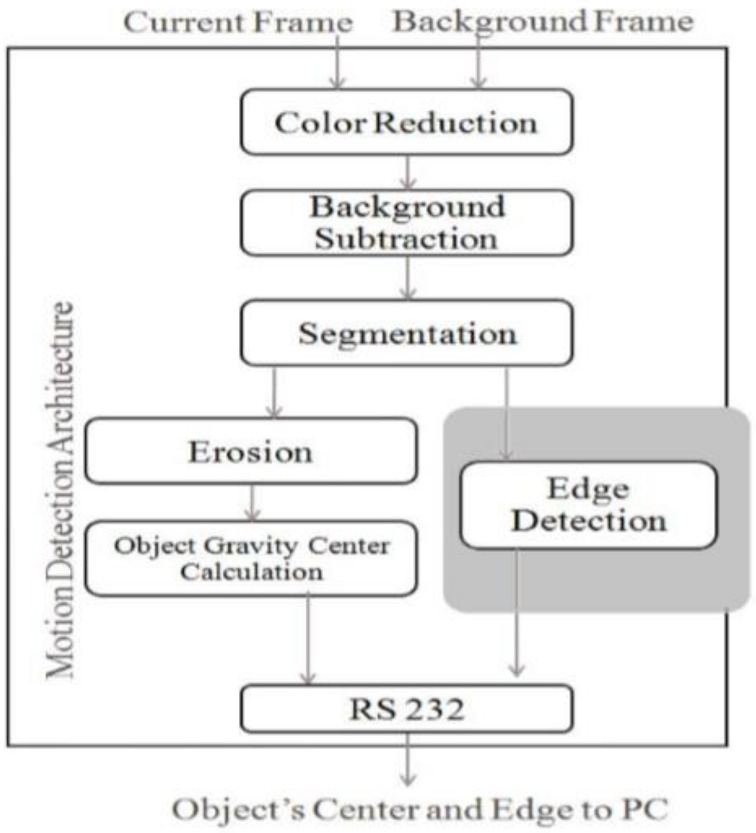


Figure 2.1: motion detection structure [2].

Detection algorithm takes a number of frames in order or to identifies moving pixels in the given scene. This thesis only focuses on Background Subtraction Algorithm. which uses as background reference video which is saved in memory. The original video is transformed into images and each image was compared with the background image to gives the moving pixels from the difference. There are two methods to extract background video, offline method (un-updated method) and online method (update method) if the offline method was applied the more error may be accrued even if the moving object becomes stationary this mean this became one element of the background. But if the online method was applied the background video was updated by averaging out the images in a period of time. This is a more effective method because in such case frequently moving the object, temporarily fixed objects etc., the background image doesn’t contain this target. Successive images are subtracted to identify moving pixels rather than using a fixed reference image. This method is able to adapt to dynamic scenery changes [2].

2.3 Sobel Edge Detection Algorithm

Sobel operator rub on a 2-D spatial domain for image. the aim of the Sobel is to find the absolute gradient scale at each pixels. Fig. 2 show Sobel mask with size 3*3.   Two (one pair) 3*3 mask is used by the Sobel edge detector. One mask is used for estimate the X direction and other is used for y direction. The size of the convolution mask is smaller than the actual image.

-1	0	+1
-2	0	+2
-1	0	+1

Gy

+1	+2	+1
0	0	0
-1	-2	-1

Gx

Figure 2.2: Sobel masks [2].

This algorithm applies this mask for the row to change pixel value and shift it to right for entire raw then go to another raw till complete all rows to detect the edge.

The main advantage of this approach is that it is very cost effective and simple, requires less computation. However, this approach faces difficulty in identifying object's shape resulting in difficult posterior recognition. Also it fails to identify stopped objects in the scene. In optical flow, a motion vector of each pixel is computed and entire image could be imagined as a vector field. The motion vector of each pixel represents the brightness of the pixel. The region of the image where brightness change is observed is considered as a candidate for moving object. This approach results in good performance, however this algorithm is very complex, as one more than image needs to be stored, thus resulting in higher memory requirements, in-turn resulting in high cost [2].

2.4 Frequency-Tuned (FT) algorithm

Motion detection and recognition of targets is an important in the field of video processing. A new Frequency-tuned (FT) algorithm has been used for extracting target dynamic saliency information from a mixture of Gaussian models, aiming at the misty effect of the traditional Frequency-tuned (FT) algorithm saliency map and the significant of feature map fusion. This algorithm makes innovative improvements from distance metrics and feature graphs. To solve the large computational complexity of traditional identification algorithms, the algorithm uses a Haar cascaded classifier with low computational complexity as a classification algorithm, and uses Open CV and Qt interface library to build an integrated multi-module system software platform to achieve single-target recognition and moving object. This algorithm achieved a good results and has significant effects on the detection and recognition of single-target motion and has high accuracy [2].

A moving object detection and identification mainly contain three functions: moving detection, tracking the targets and recognition. Now, according to the old method, there are two methods of different ideas for motion detection. The first type is based on the content of pixel location, this algorithm mainly focuses on the motion properties of the moving targets themselves, the image is segmented by means of edge detection, optical flow, etc., and then the spatial characteristics are used as the main criterion to estimate the moving object. The results of this method are more accurate, but the amount of calculation is very large; another kind of algorithm mainly uses the change of detection time as a basic criterion. This type of algorithm does not focus on the motion properties of moving objects but detects the change and invariant regions of the image mainly by the inter-frame difference method. the goal from it segments the moving target and the fixed background and ultimately achieve the effect of moving object detection. The biggest advantage of this method is that it reduces the computational complexity, but the accuracy is not good. This article is mainly based on the principle and mechanism of visual attention, applied to the detection and recognition of single target moving objects [3].

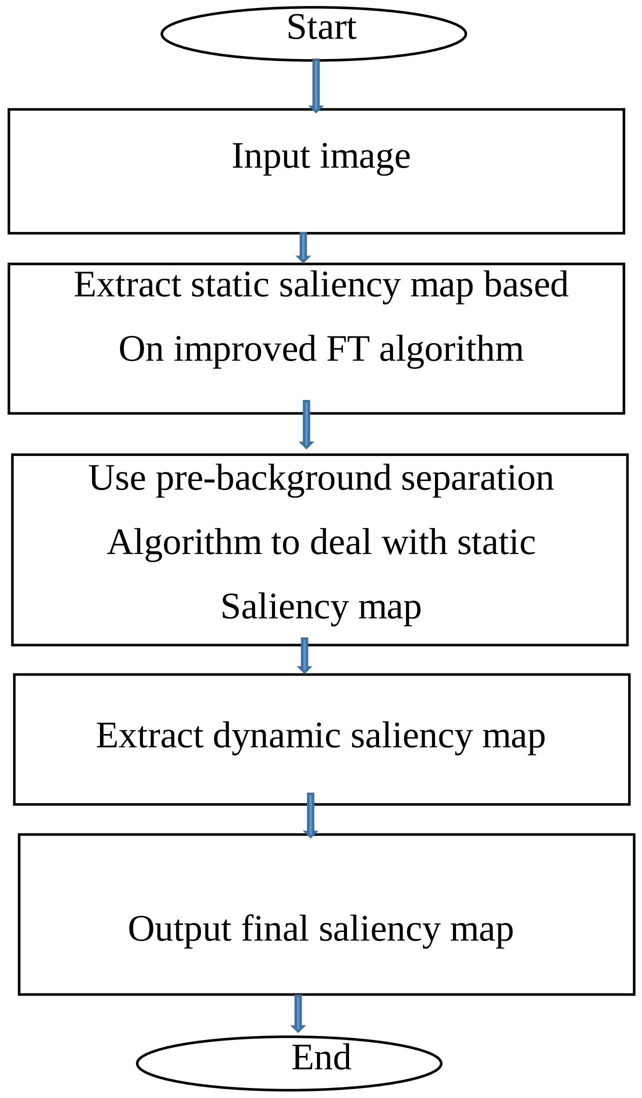

Start
Input image
Extract static saliency map based
On improved FT algorithm
Use pre-background separation
Algorithm to deal with static
Saliency map
Extract dynamic saliency map
Output final saliency map
End

Figure 2.3: Frequency-tuned (FT) algorithm

Firstly, an improved Frequency-tuned (FT) algorithm is proposed, which is improved from both the distance metric form and the feature graph fusion method. Then the improved Frequency-tuned (FT) algorithm is combined with the pre-background separation algorithm based on the mixture Gaussian model to extract the dynamic significant information. Then, the Hear cascade classifier with relatively simple computation as the basic algorithm for object recognition. Finally, the Open CV open-source computer vision platform was chosen as the basis for the implementation of the algorithm, and the Qt open source interface library was chosen as the platform for system integration, and it integrates human-computer interaction GUI interface module, detection algorithm module, identification algorithm module, network remote transmission module, and forms a complete system model. Through the actual test analysis of the image effects and the final frame rate and recognition error rate, it can be seen that the algorithm model can well complete the single-target moving object detection and identification tasks. The entire system has a wide range of applications and has a good prospect of practical engineering applications.

The progress advantages of networking, digitization and intelligence, embedded video surveillance system is broadly used in our life. In the current solutions, the difficult image processing algorithms cannot be achieved professionally in costly manner [4].

Design a embedded image processing system that involved image acquisition, processing and displaying in one set, but the image processing module and network transmission module is still not perfect [5].

Used Open CV to process the single image downloaded from the Web server, rather than the streams collected from camera in real-time [6].

Designed a networked video surveillance system based on B/S structure, this solution is very matured in embedded video surveillance, while there is no image processing module [7].

Based on the existed embedded video surveillance system platform, this paper adds the image processing module using efficient Open CV library functions, the live video streams captured from cameras can be handled effectively. Then take moving target detection algorithm as an example to verify this solution is feasible [8].

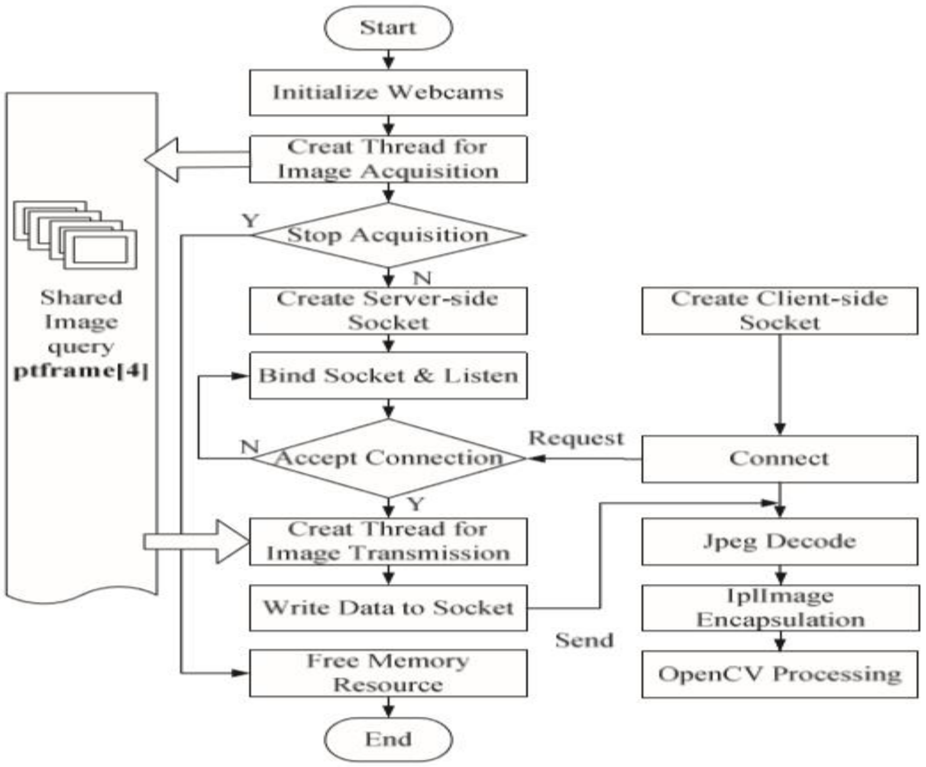


Figure 2.4: falling detection algorithm [8]

Falling detect system. In the proposed system, this design presents an automatic approach for detecting and recognizing falls people using video based technology. This design focus on the protection and assistance to the elderly people. The fall action is automatically extracted from the video, special information that can be used to give alarm or to make a suitable action whether the fall is really occurred. The main function of this work is to provide such a system which automatically detects the fall and intimate the respective authority. Proposed method uses background subtraction to detect the falling object and mark those objects with a boarder box followed by extracting the f information [8].

2.5 Morphological Transformation and segmentation method

This work aims to detect a vehicle from the high way traffic flow based on the morphology and wavelet transform method. Using image differences to get a background to transform the impact of light clearly, the corresponding easy and quick to update the background algorithm was used. This was achieved by background subtraction algorithm, and then images of the vehicles were the accurate detection of mathematical morphology and wavelet transform. A video object detection system was developed using visual (C++) v6.0 and Open CV. A background extraction, image filtering, binary image, morphological conversion, vehicle detection, and separation methods and stages have detected vehicle on the road from traffic flow. this system to identify the correct rate of more than 98 percent, satisfying the requirements of practical applications [9].

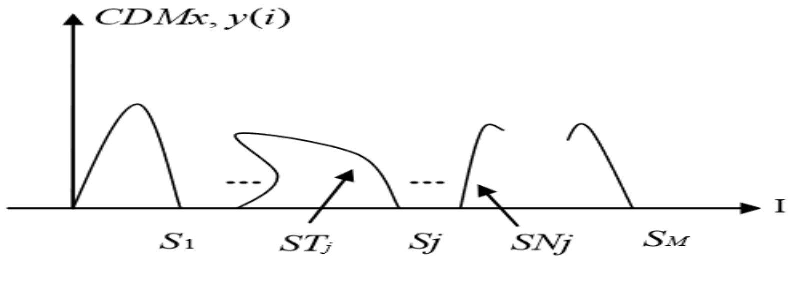


Figure 2.5: The Brightness Image Difference During Time Axis Change Function

The flowing equations represent the corresponding video background as (2.1). And video frequency background as in (2.2).

M (x, y) -(ST (x, y) +End (x, y))/2 (2.1)

B (x, y) =F (x, y, M (x, y)) (2.2)

ST (x, y), End (x, y) the statistic partition (starting and end), B (x, y) is reconstruction video frequency back ground [9].

2.6 Improved background subtraction algorithm

Motion detection and tracking function is based on the video stream processing of computer vision. Motion detection mainly used for object tracking, behavioral analysis, target recognition and identification, and so on. To achieve all these tasks must use the background extraction firstly, then easily can achieve these tasks, therefore enhances the motion detection algorithm timeliness, the accuracy and the integrity has theoretical value and the actual project significance. To detect moving object, the object position is very important to movement object tracking when needs to solve the question has the movement object classification, tracking function not effective when tracking the movement, the multi objective tracks and so on. Moving object detection and tracking algorithm was researched in this work. An improved background subtraction algorithm was proposed for this work [9].

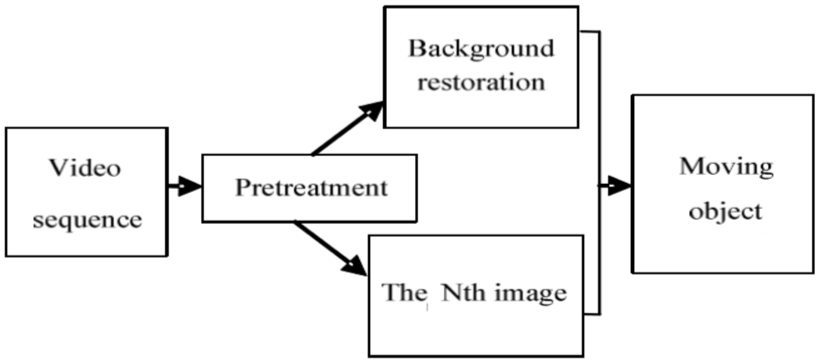


Figure 2.6: Improved Background Subtraction Algorithm [9].

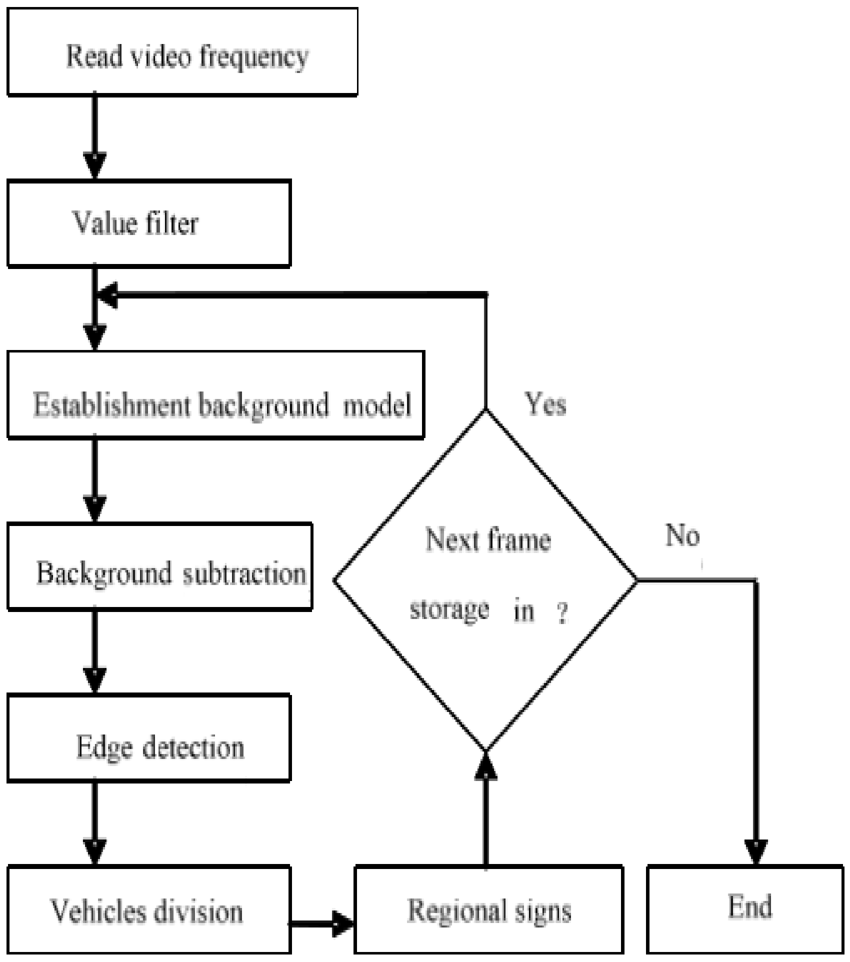


Figure 2.7: Vehicles Detection Algorithm [9]

2.7 Further Related Studies

In [10] the issue is detecting up normal sleep by sensing the patient motion using motion detection algorithm to determine sleep disorder. The MATLAB software was used to achieve this project, the image differences algorithm was found in MATLAB library (imabsdiff) this function requires two arguments as image and has one output image as absolute differences between the pixel's value. The output of this work was tested at a short time.

Because the designer found the processing of this algorithm using MATLAB was need more time so this project unable to process at a long time.

In [11] the issue object extraction combining with motion detection algorithm using MPEG-4, the project focused on the object rather than background so the output was doing not consist of background. Three steps were used for this project, spatial image partition(SIP), temporal motion detection (TMD) and spatiotemporal projection (STP) using global. Applied all three algorithms on MPEG-4 software to test project. The result of this project was effective to extract the object if there is one object appear in the video, but if the video contains more than object this project cannot process all objects so the project will need new techniques to avoid this problem.

In [12] the issue is detecting a salient object motion in the complex background area and recognize target motion from different motions on the monitoring area. The algorithm that was used for this project based on combining temporal difference image and binary block image which is calculated by the motion vector using the newest MPEG-4 and EPZS.

The result of this project is more effective in complex background area that consists of more than one moving object, but the target object must be moved faster than other objects. The limitation of this project depends on the type of moving objects and their speed if the target object was moving slowly or zigzag this algorithm cannot detect this object.

In [13] the aim of the work is to design the real-time motion detection algorithm using open CV library. The image segmentation preprocessing algorithm was used in this project using open CV library, and transform all picture from RGB colored picture to gray level, then using frame differences to detect the moving object by using threshold value. Through this threshold, applied to the whole image, pixels whose gray levels exceed the value, then set the value of this pixel (1) otherwise reset this pixel (0). The Result of this project is to detect only a red color object but did not detect any other color object. this project was designed for the only red color object so this project could not use in general application because it designs for specific applications.

In [14] the issue is design and enhance security system by motion detection algorithm using MATLAB and GSM connecting with a PIC microcontroller for the alert system. MATLAB software was used for video processing and apply the algorithm of detecting an object, the microcontroller for alarm system witch send alarm control signal to the GSM module to send alarm message when moving to detect. The output of this project depends on complexity and format of input video because the algorithm was used for motion detection based on try and error input value (tuning factor) to enhance the output.

In [15] the issue is detecting motion for security system using video processing on open CV. The methodology that was used in this paper is mathematical morphology using generic tool for open CV, Emgu CV with absolute differencing to get the object motion. Also, use image filtering for noise removal.

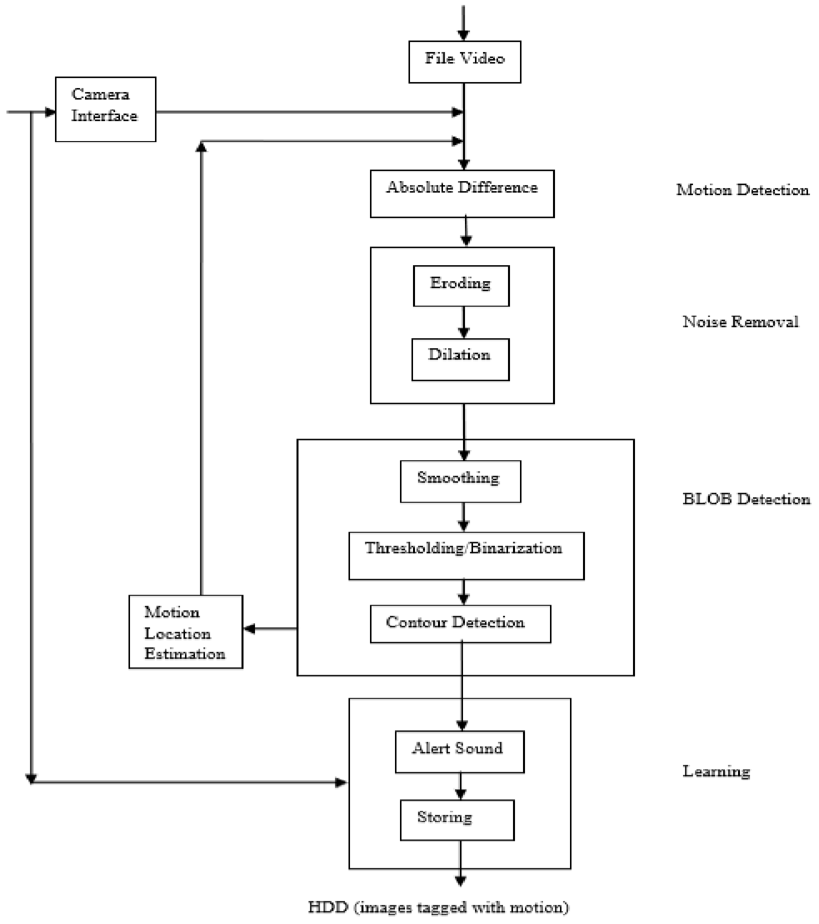


Figure 2.8: Motion Detection Uses Mathematical Morphology.

The result of this work is shown in figure bellow witch consist little noise in output binary image.

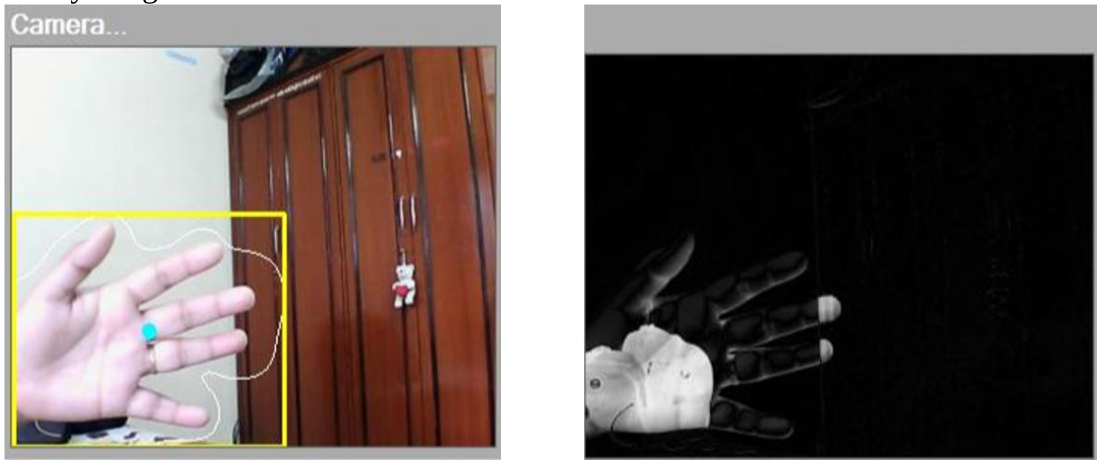


Figure 2.9: Detecting Object Causing Motion & Output of Absolute Differencing

The left image is the moving object surrounded by bounding yellow triangle and the right image it is binary output result there are some noise in the object, also the video was used in this project is file video, not a real time and the stationary background.

In [16] the issue is design the moving object detection and tracking system for security in important area. The methodology was followed on this work background subtraction algorithm and Foreground mask sampling. This paper proposes the use of cascade classifier for object detection.

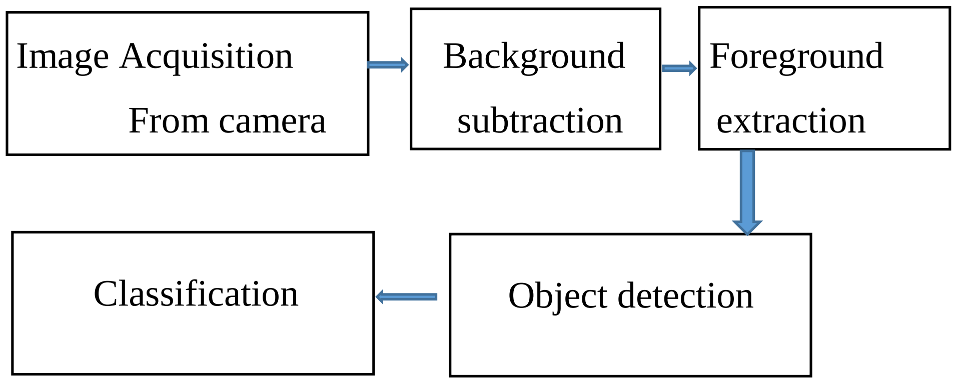


Figure 2.10: Object detection algorithm

This paper provided the function of moving object effectively using the background subtraction method but did not use video contain un stationary background to test the updated background.

In [17] the issue is design and implementation the motion detection algorithm using Raspberry Pi hardware an optical flow algorithm.

The methodology was used in this project is the optical flow algorithm using Raspberry Pi. The idea behind this project is to find the displaced object from an image using the Lucas Kanade algorithm. This algorithm works by comparing the first two successive image frame with that it guesses the direction of the displaced object. The output of this project worked well at a low-speed object but there is the delay was created from Lucas-Kanade algorithm estimates the motion of objects. The hardware was used in his project did not support all video format.

In [18] the issue is design motion detection algorithm to keep a more time to monitoring and system power. The method that used in this project is background subtraction and frame differencing technique. Background subtraction is a method that subtracts the current frame with background frame to get a moving object and frame differencing is a method that subtracts the current frame with the previous frame to get a moving object.

The monitoring system is usually applied to security parts, for example in investment, warehousing, workplaces, various public facilities such as airports, stations, until used in homes. The implementation of a checking system for production parts, for example, is applied in the manufacturing or industrial segments where management can monitor or monitor production events of worker’s laborers, control process instrumentation, machinery installations, and others. And of course, there are still many other objectives that underlie the implementation of the monitoring system. Thus, the use of a camera on a surveillance system is needed. But the problem is that the camera that is installed is always recording even though there is no movement or incident that occurs, as a result, there is a useless memory use. One other to handle this problem is to design software that can improve the efficiency of the camera.

What is meant by background is a number of pixels that are still images and do not move in front of the camera. The simplest background model assumes that all background pixel brightness changes freely depending on the normal distribution. Background characteristics can be calculated by accumulating several numbers of frames so that it will find the number of pixel values in location s (x, y) and square-square sq (x, y) which have values for each pixel location. While the foreground is all objects that exist besides the background and usually the foreground is there after the background is obtained [18].

Background subtraction is one of the important tasks that was first done in computer vision applications. The output from background subtraction is usually input which will be processed at a further level such as tracking the identified object. The quality of background subtraction generally depends on the background modeling technique used to take background from a camera screen. Background subtraction is usually used in the desired object segmentation technique from a screen and is often applied to surveillance systems [18].

The purpose of background subtraction itself is to generate a sequence of frames from the camera and detect all the foreground objects. A description of the existing approach to background subtraction is to detect foreground objects as differences between the current frame and the background image from the static screen. A pixel is said to be foreground if:

$$|\text{Frame } i – \text{Background} i| > \text{Threshold} \quad (2.3)$$

Another approach to getting a background is the Running average model. Compared to other models such as the Average Model or Median, this model is superior because it requires less memory than the other models. The running average model has the following formula:

$$Bi + 1 = \alpha * Fi + (1\alpha) * Bi \quad (2.4)$$

Where α is the level of learning, usually has a value of 0.05, while Bi is the background and F is a frame (image).

The result of this project was testing by two recorded videos, the first video is used the object is different from the background then the final result is clear, and the other video consists the similarity between the object and the background then the result has been corrupted [18].

CHAPTER THREE
ALGORITHM DESIGN

Introduction

In this research, the system of detection of moving objects was carried out using the background extraction algorithm and then use the motion detection algorithm and tested the system using three kind of videos such as:

- Two Recorded videos.
- Computer web cam video.
- External USB camera video.

3.1 Software and Tools

There are many software and tools can we chosen to implement this project. The software and tools will be explained in the following subsections.

3.1.1 Open CV Library

The algorithms were written on open CV version 3 library on visual studio version 12 using C++ language. Open CV was chosen because it specialized programs in the field of video processing in computer vision and it has more features than other programs in this field such as

- Open CV is open source computer vision library.
- Open CV support common programming language as C, C++, Python and supported by windows, Linux, and Mac OS X.
- Power full in field of image and video processing.
- After design the project code can use this code to generate HDL code using special program for hardware design.

3.1.2 USB Camera

This project was need external camera to do the real time test to verify the system designed as we need.

USB camera was selected because it easily to connect with computer throw USB interface, also the windows easily recognize it and available with suitable price.

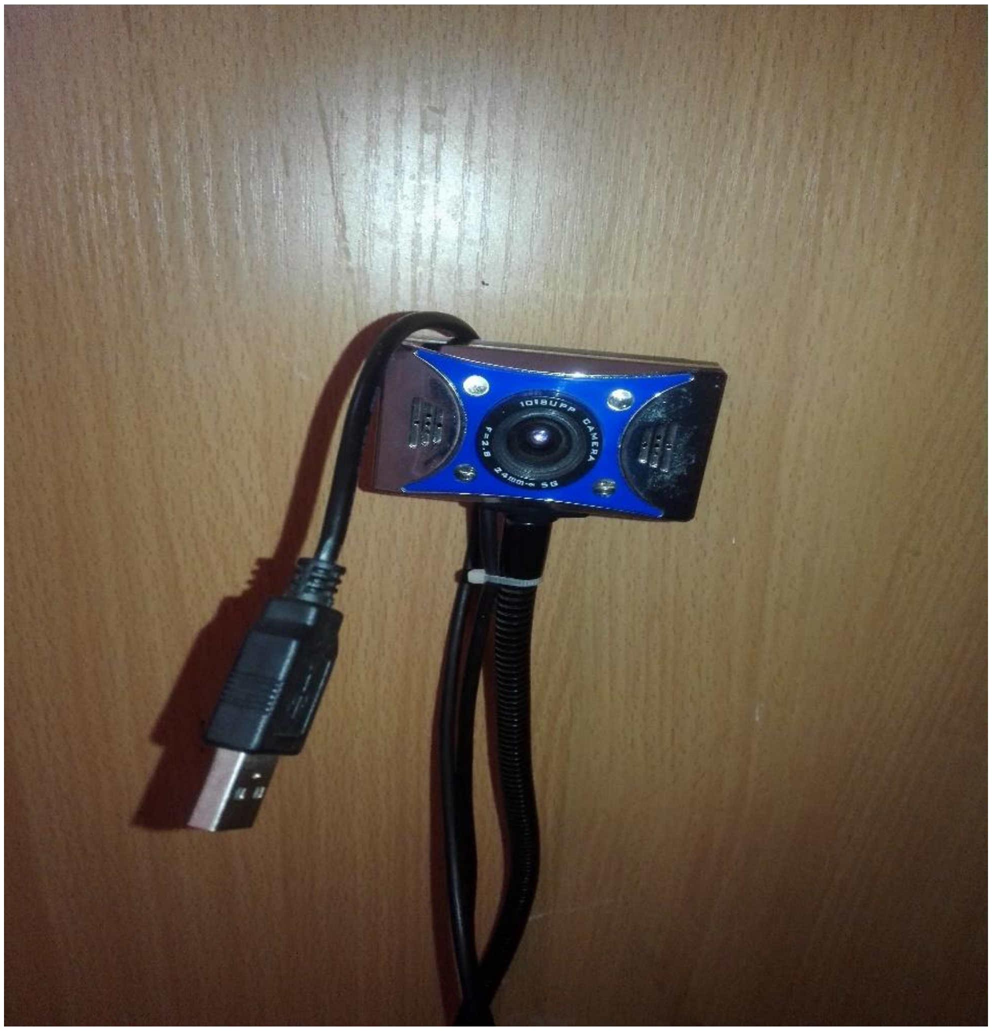


Figure 3.1: USB Camera

Table 3.1: The specification of selected camera

Parameters	Specification	Note
Camera Model	10 & SUPP camera	Fixed camera with flexible carrier
Frequency (frame rate)	30 frame per second	Standard frame rate
Focal number	2.8	(F. Number)
Focal lens	24 mm	Zooming lens
Video Format	PAL	Digital video
Interface	USB	2.0
Frame size	640X480	307200 Pixels

3.2 General System Description

In the general this system consists of three main parts

- Monitoring view: The target place that you need to monitor important places as work shop, homes, offices, high way traffic flow, etc…
- Camera: To get real time video then send this video information to the system.
- The System: May be computer or hardware system receive the real time video from camera then apply the algorithm to give the result as information as you like.
- Display unit: external screen or computer screen for display.

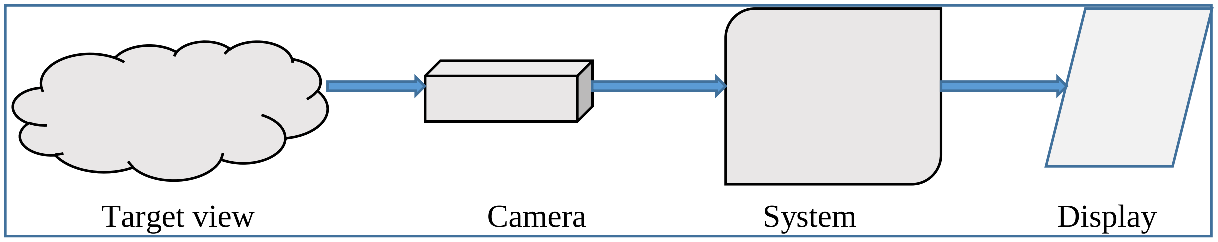
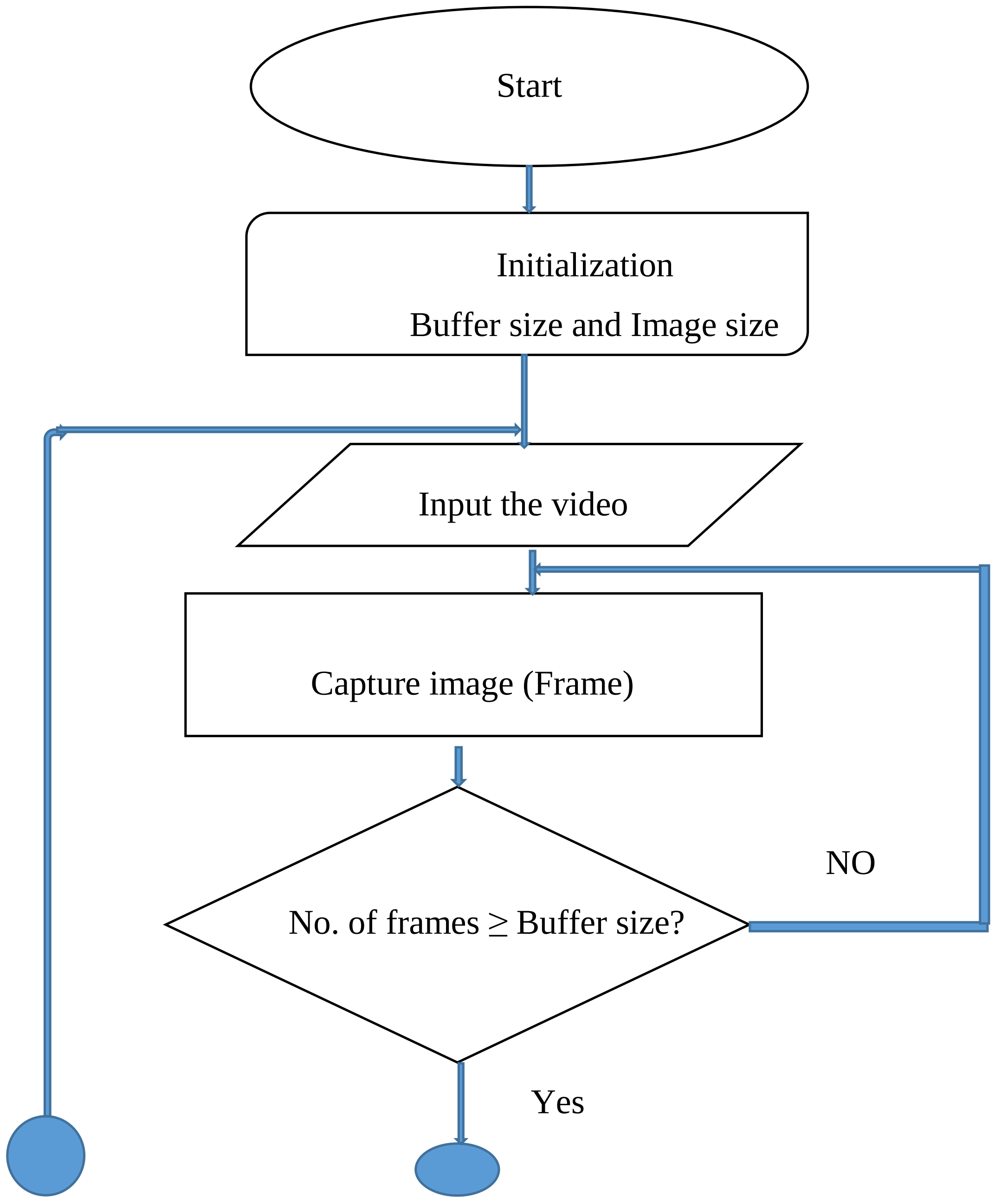


Figure 3.2: General System Description

3.3 Algorithmic Design

This part explains by more details the steps of design the project and all algorisms that were applied.



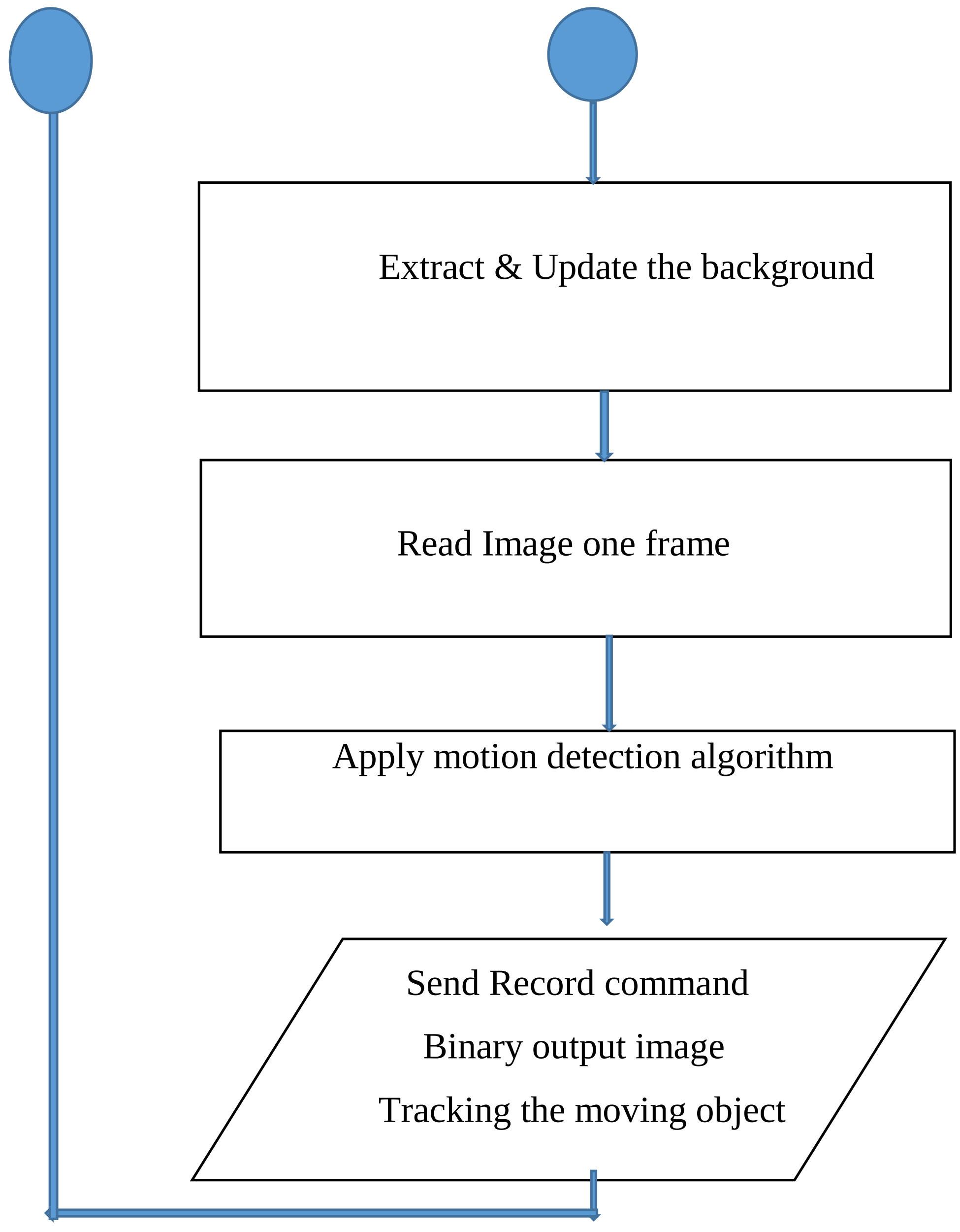

Extract & Update the background
Read Image one frame
Apply motion detection algorithm
Send Record command
Binary output image
Tracking the moving object

Figure 3.3: System Flow Chart Design

- Start: When run the program the system will start execute the design codes.
- Initialization: In this step all parameter is setting with initial value such as buffer size witch indicates for number of image captured from input vide, frame size includes the number of pixels in rows and number of pixels in column, and other variable if you need to initialized it.
- Input the video: There are three ways used to input video to the system
- Recorded video (already video saved in memory).
- Getting video from the laptop web cam.
- Getting video from external camera throw USB interface.
- Capture image: This process for getting one image (frame) from the input video and store it in the buffer to use in background extraction algorithm and motion detection algorithm.
- Comparison stage: This stage to compare the number of captured images with buffer size value:
- If the number of captured images equal the value of buffer size, then the background will be update to avoid the un stationary background problem, the next captured image will be passing to the motion detection algorithm direct, and the counter of the captured image will be reset.
- If the number of captured image did not equal the buffer size value, then the captured frame will be passing to the background extraction algorithm and motion detection algorithm and the counter of the captured frame will be increment.
- Extract and Update the background: In this stage the extraction background algorithm has been applied. The idea of this algorithm refers to comparison operation between the value of pixel in the position (x, y)

in the background frame with corresponding pixel in the same position in the new captured frame, then if the two value are matching to gather, then the possibility of this pixel in this position as background will increase, if you found the big different between two pixels in same position, then the possibility will decrease, then switch to the following pixel till reach the end of frame, then receive a new frame and apply the same operation on it till the number of frames reach the value of the buffer size.

This algorithm should be running all of time but the updated output will appear rapidly (smooth) each certain period of time. The initial background was getting from the first frame, then every frame would be coming should enhance the background till the number of frames equal to the buffer size at this moment the background must be clearest. After the number of frames equal the value of buffer size then each frame would be coming must compare with all previous frames to detect the changing if the background would change. When buffer size is big the output of background is more clearly but the rate of out but video will decrease

$$I = (\text{No.FC}-1) \,\%\, BZ \tag{3.1}$$

$$\text{Average Background}= \left(\sum_{I}^{BZ+I} F(I) + FC\right)/BZ \tag{3.2}$$

Where I the index of saved frame, No.FC the number of current frame, BZ is Buffer size, F(I) frame number I and FC the current frame.

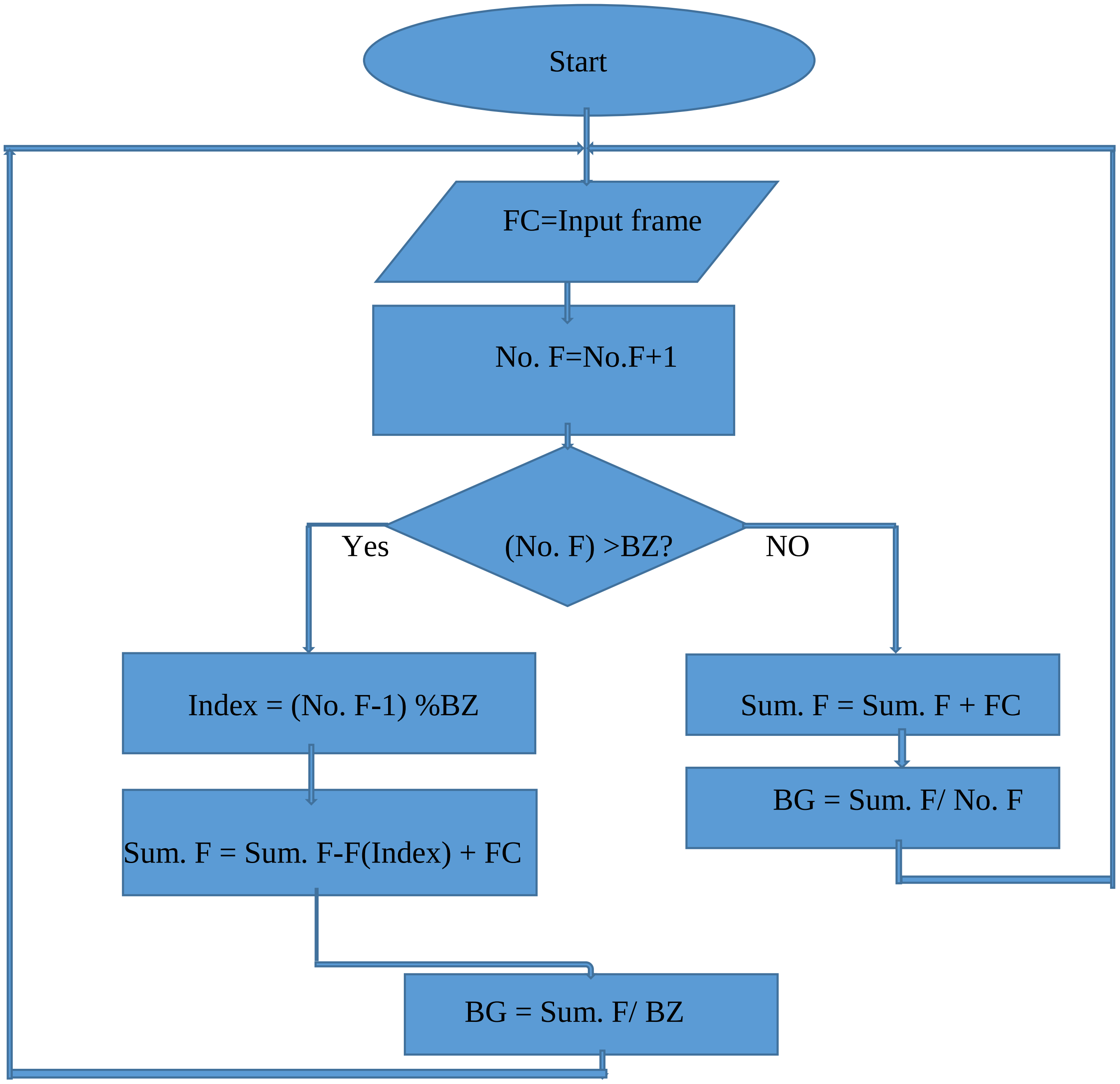


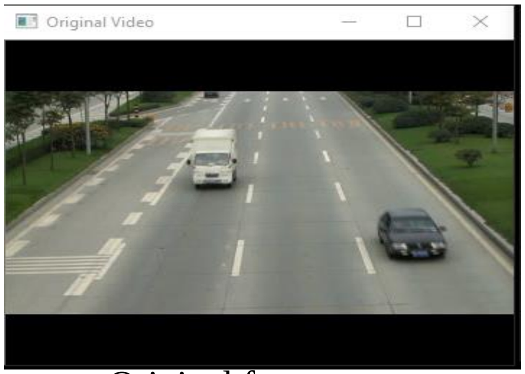
Figure 3.4: Background Subtraction Algorithm

CHAPTER FOUR

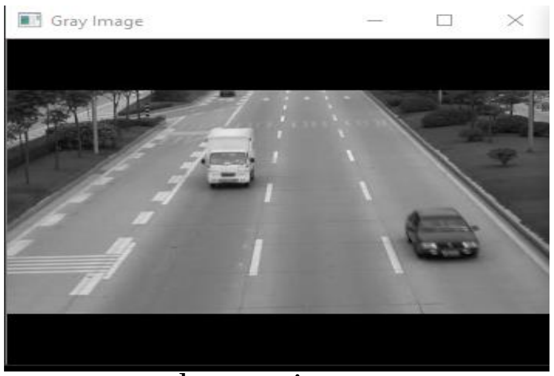
THE RESULTS

4.1 Average algorithm

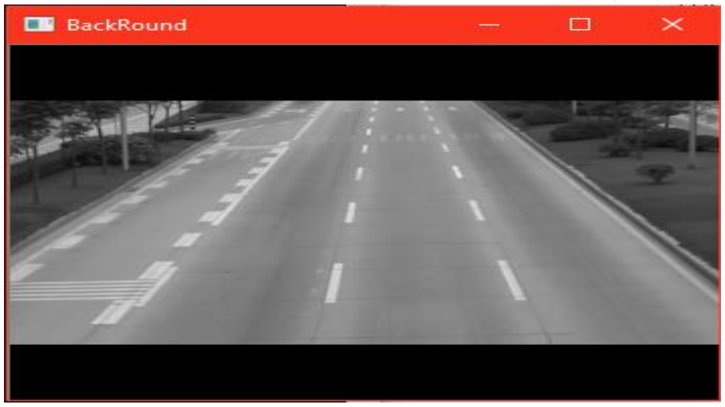
In the first test The algorithm was tested by un-updated background video, then the background video would have extracted well this algorithm effective on the fixed background view (Stationary background) because the background would take one time from input video this cannot make a delay on output video.



a. Original frame



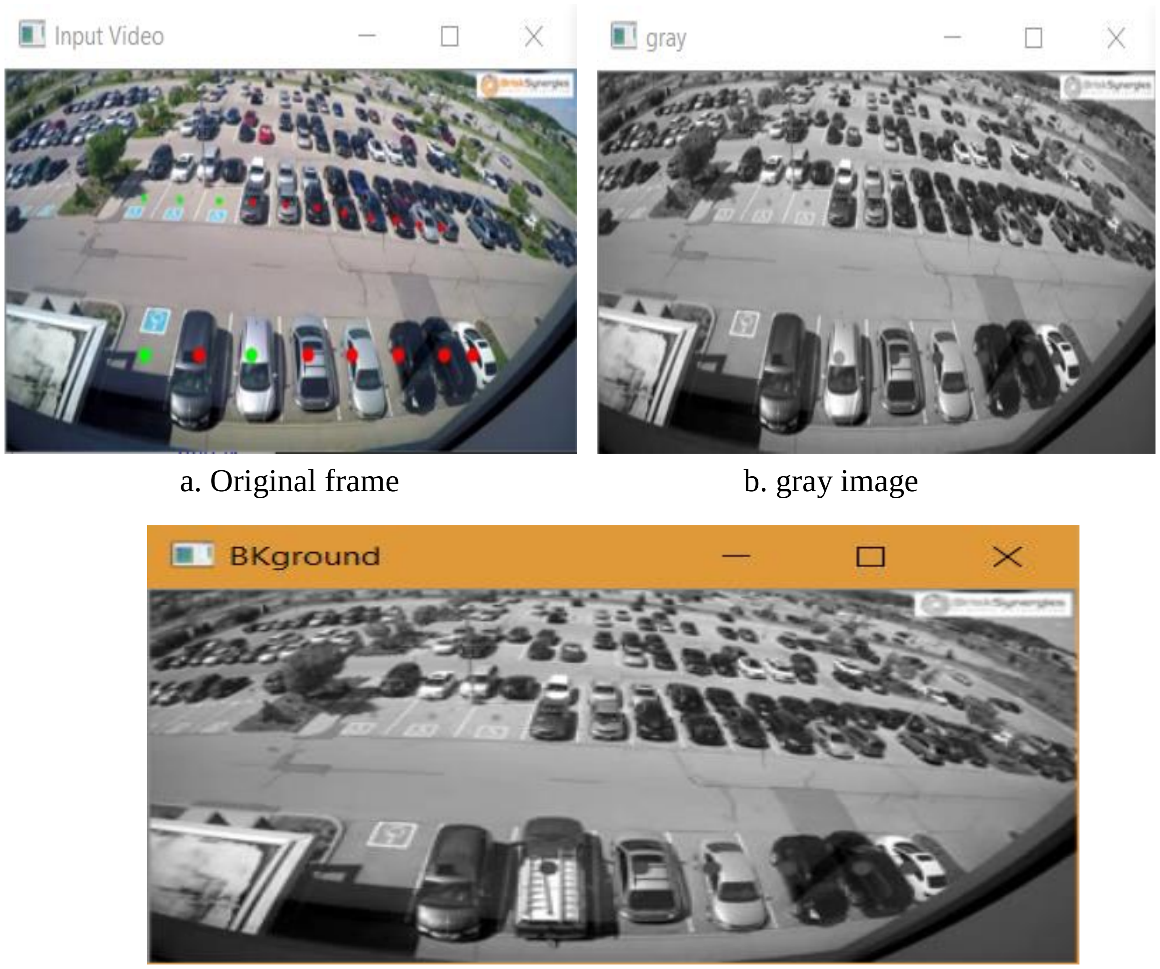
b. gray image



d. Background frame

Figure 4.1: Result of Average Algorithm with Un-Updating Background

In the second test another video test was tested with the same algorithm, but we found the background did not update well because the view of the background was changed by the moving object (appear or hidden) at a long time then this object became one part from background view.



a. Original frame

b. gray image

c. Background frame

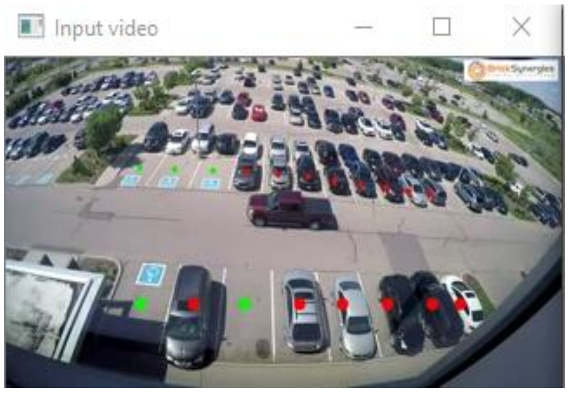
Figure 4.2: Result of Average Algorithm with Updating Background.

a. Input (original) video.
b. Gray scale for original video.
c. UN updated background frame.

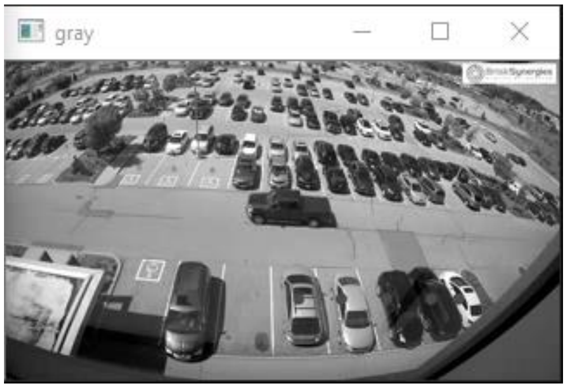
In picture c. the big car no. 2 appear in the background frame, but it doesn’t exist in the original frame. That means this care was left and another car was parking in the same place appear in the picture a.

4.2 Modified average algorithm

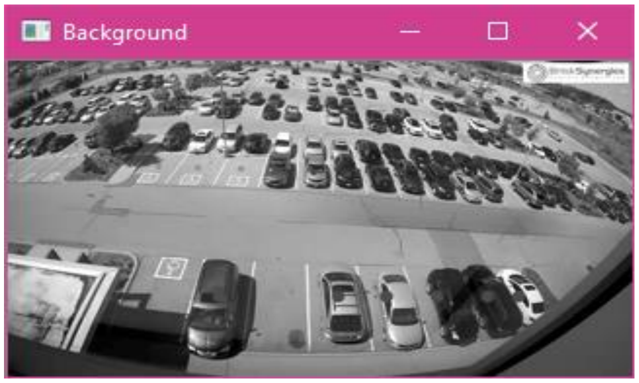
After modified the average algorithm the background was extracted and updated well from the average number of frames (buffer size) the figures below show the result of the modified algorithm.



a. original video frame



b. Gray video frame



c. Background frame

Figure 4.3: Modified Background Extraction Algorithm Results

In this case, there is a delay in the rate of output video because after the buffer size is full, any new frame would update the output background frame by adding all previous frames with the new frame then calculate the new average witch represent the updating background.

The result of background accuracy based on the value of buffer size as follow:

Table 4.1: Buffer Size with Background and Frame Rate

Buffer size	Background accuracy	Frame per second FPS
50	50%	90%
100	60%	80%
150	70%	70%
200	80%	60%
250	90%	50%
300	100%	40%
350	100%	30%
400	100%	20%

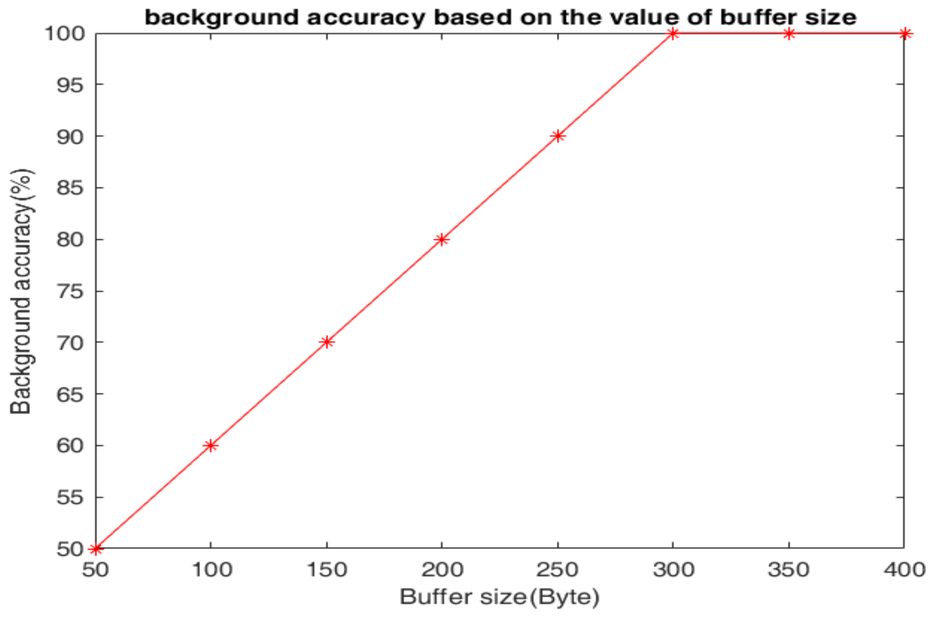


Figure 4.4: Relation Between Buffer Size and Background Accuracy.

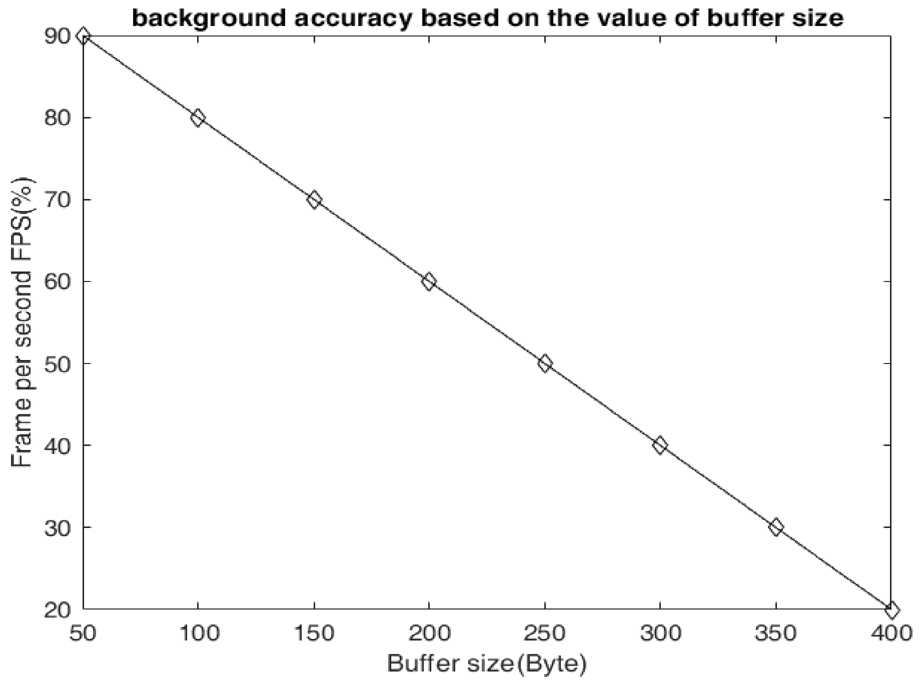


Figure 4.5: Relation between Buffer Size and Output Frame Rate.

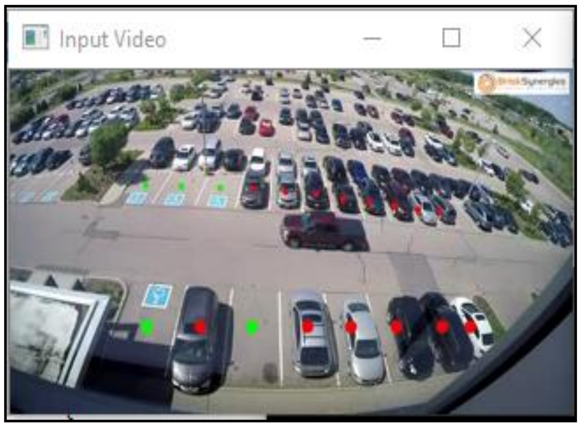
4.3 Motion Detection Algorithm

Motion detection algorithm using frames differencing from the new frame and background frame. the output appears as a binary video, the accuracy of the result based on the accuracy of the extracted background.

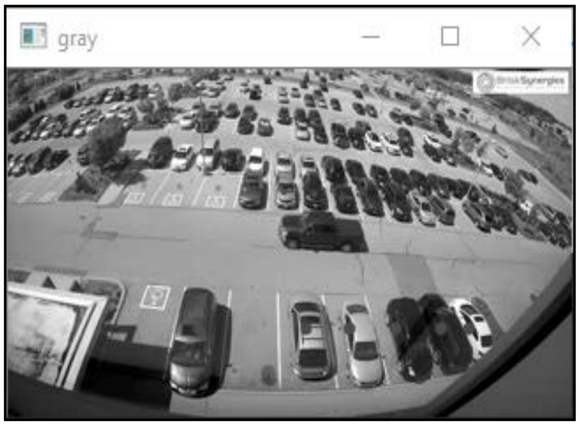
Figures below explain the result and steps of the motion detection algorithm from video file.

4.3.1 Motion Detection from Video File

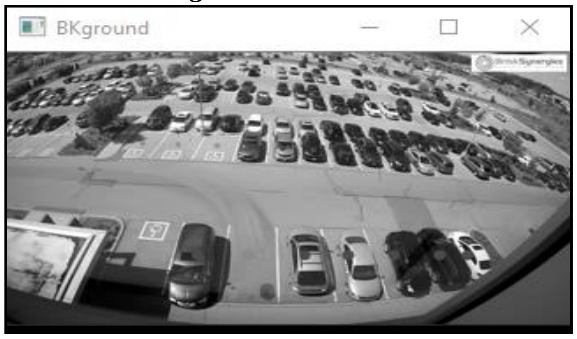
In this case, the algorithm was applied by recorded video, then the result depends on the stability of the background, the right number of the threshold for binary output and number of moving objects.



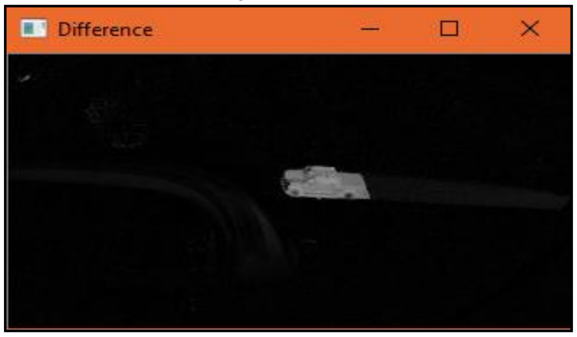
a. Original frame



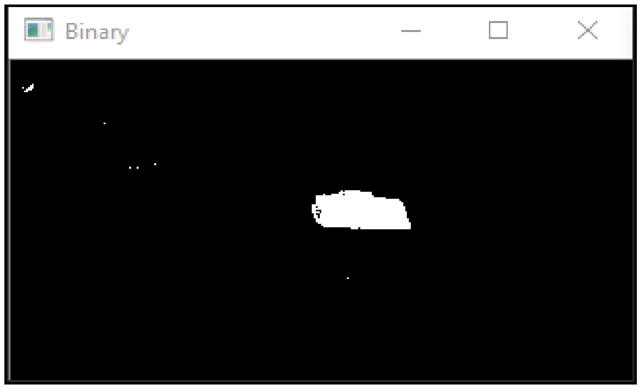
b. Gray video frame



c. Background frame



d. difference frame



e. Binary output frame

Figure 4.6: Motion Detection Results.

a. the original video.
b. the result of a grayscale video frame for processing.
c. the result of updated background video frame.
d. the result of differencing background and a new frame.
e. the result of detection as a binary frame.

Table 4.2: Threshold with Output Accuracy

Threshold	The object appearance	Accuracy
10	Bad	40%
20	Good	60%
30	Good	70%
40	Very good	90%
50	Good	60%
60	Bad	40%
70	Bad	30%

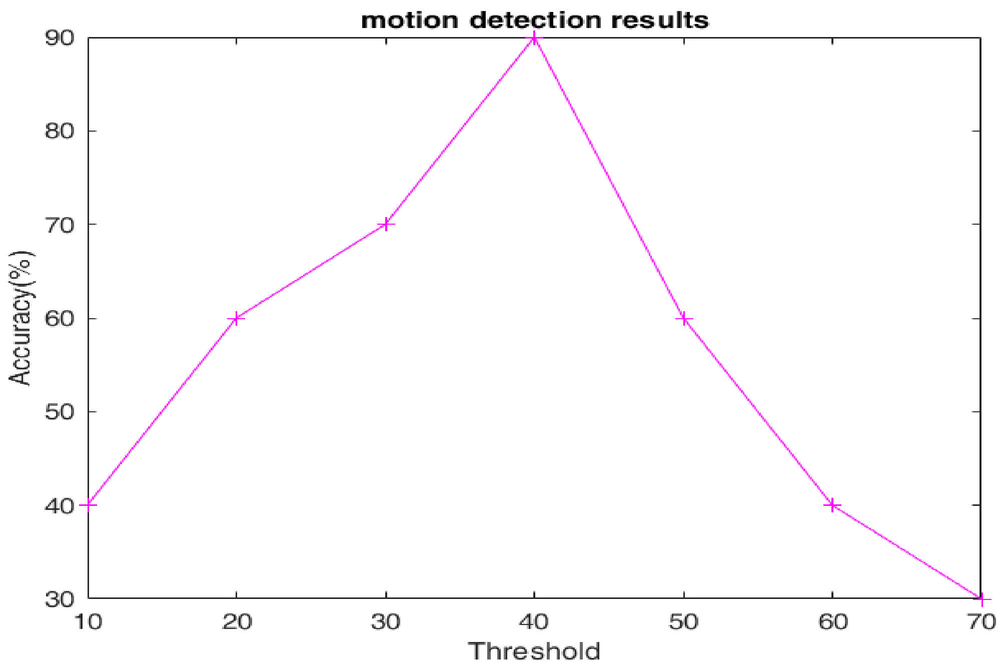


Figure 4.7: Relation Between Threshold and Output Accuracy

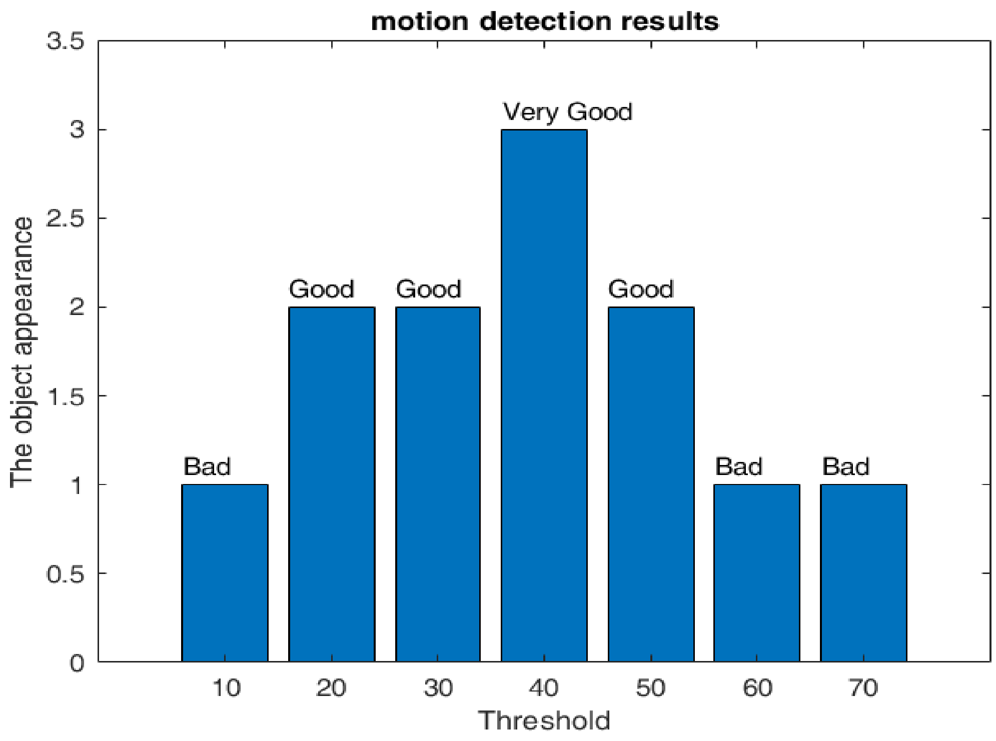
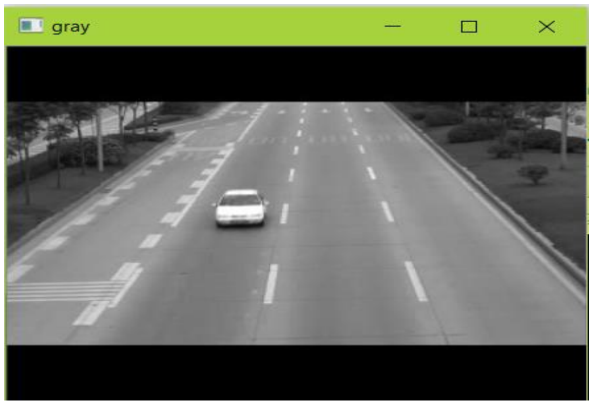


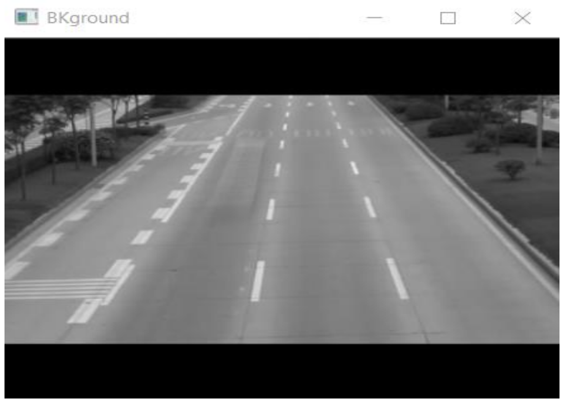
Figure 4.8: Effect Threshold for Quality of Output.

4.3.2 Bounding box feature (Tracking object)

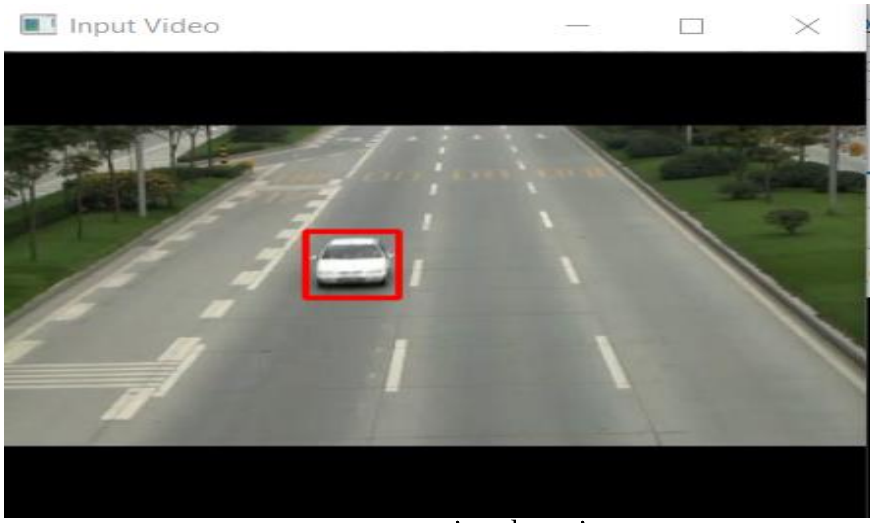
In this part, the output might be the original video with a bounding box around the moving object. This feature makes the algorithm more effective by tracking the moving object only without affected to the source video. All details of the original video are found but the bounding box appears only if the motion was detected.



a. original gray video



b. background frame



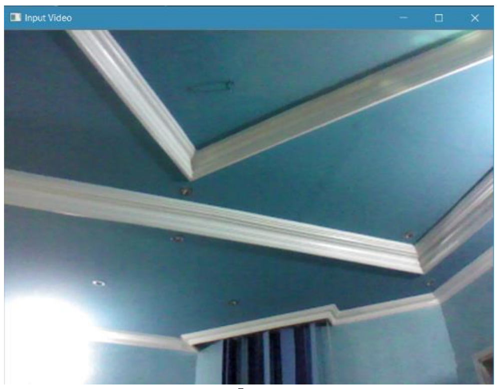
c. output motion detection

Figure 4.9: result of output with bounding box on original video

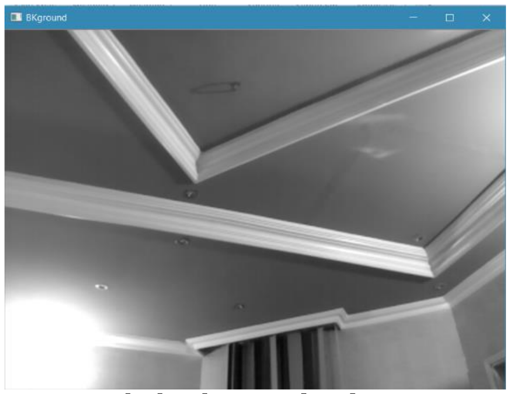
4.3.3 Motion Detection using Live Video

The quality output video based on camera resolution, the tested area (the number of moving object that you need to detect), the good tuning for buffer size to get a clear and updated background and also suitable threshold value to get clearance output video with object detection.

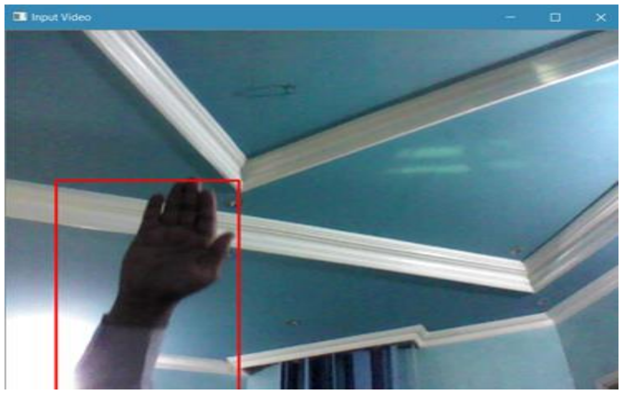
The buffer size tuning must be different based on the background updated, if the monitoring area has updating background slowly and the speed of the object is high, then the buffer size tuning value must be big value. And if the monitoring area has updating background slowly and the speed of the object is low, then the buffer size tuning value must be a small value. Because the time that needs to update the background effect on the output of the algorithm.



a. input video



b. background video



c. output of motion detection

Figure 4.10: The Result of Motion Detection by Using Live Video

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This research was talked about video processing topic in computer vision field witch discussed the motion detection algorithm to detect moving objects for several purposes such as security systems, surveillance fields, traffic flow and so on. Also after moving objects was detecting the project was creating the suitable action that you need after the unwanted motion was accrued, these actions such as generate alert signal to alarm system to inform this action was accrued, send the recorded signal to video recording system to start recording video for important actions only to save the memory space (reduce the cost) and make analysis for playback more easy, and also you can use with PTZ cameras to track object by sending control command to PTZ camera to make a suitable pan or tilt depending on motion direction.

This project was achieved by two algorithms, Background extraction algorithm based on average pixels' value, this algorithm was used to extract and update the fixed view from the input video (video file or camera live video). The test of this algorithm was succeeded with a different kind of video and gave the good results. This algorithm was used with another algorithm to achieve the moving object detection. Another algorithm is a motion detection algorithm based on frame differencing also was tested successfully with different videos. The two algorithms designed by open cv library under visual studio v12.

The final project was tested with the two video files and live video from the web camera and external USB camera. The two video files were used to test the extraction of the background (stationary and non-stationary) which gave the good results compared by previous work. The live video test from webcam and external USB camera was passed successfully and good results output were reached after test complete.

5.2 Recommendation

This project was designed by using Visual Studio as software only so I recommend to design this project as hardware by using HLS program, because this software can compatible with open cv code then burn the output bit stream file from HLS to FPGA, also you can use Raspberry Pi after download the open cv library on raspberry pi under Linux. This hardware design should make this project more effective and general use. Today Object recognition feature become very important more application also I recommend adding this feature to this project.

References

1- Tekalp, A.M., 2015. Digital video processing. Prentice Hall Press.

2- Gujrathi, P., Priya, R.A. and Malathi, P., 2014, August. Detecting moving object using background subtraction algorithm in FPGA. In Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on (pp. 117-120). IEEE.

3- Yu, L., Sun, W., Wang, H., Wang, Q. and Liu, C., 2018, August. The Design of Single Moving Object Detection and Recognition System Based on OpenCV. In 2018 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1163-1168). IEEE.

4- Guo-rong, Z., Chang-zhen, X. and Yan, Z., 2011, May. A method of embedded video surveillance based on OpenCV. In E-Business and E-Government (ICEE), 2011 International Conference on (pp. 1-4). IEEE.

5- Jia Jing-jing, Liu Ming-jie, Sun Kai. A Embedded Image Processing System Based on ARM [J].Control & Automation. 2009,25(8)

6- Zhang Li-ping, Zheng Wei-qiang. Image recognition and processing platform based onS3C2410 and embedded Linux [J], Science and Technology Information,2009 , No. 5

7- MENG Xu-xia, QIN Shao-hua, WANG Yan-mei,et al.Design and Implementation of Embedded Remote Video Monitoring System[J].Control & Automation,2009 辈 25(1-2).

8- Chamle, M., Gunale, K.G. and Warhade, K.K., 2016, August. Automated unusual event detection in video surveillance. In Inventive Computation Technologies (ICICT), International Conference on (Vol. 2, pp. 1-4). IEEE.

9- Lei, Z., Xue-fei, Z. and Yin-ping, L., 2008, December. Research of the real-time detection of traffic flow based on OpenCV. In 2008 International Conference on Computer Science and Software Engineering (pp. 870-873). IEEE.

10- Islam, M.Z., Nahiyan, K.T. and Kiber, M.A., 2015, December. A motion detection algorithm for video-polysomnography to diagnose sleep disorder. In Computer and

Information Technology (ICCIT), 2015 18th International Conference on (pp. 272-275). IEEE.

11- Yang, W., Lu, W. and Zhang, N., 2007, September. Object extraction combining image partition with motion detection. In Image Processing, 2007. ICIP 2007. IEEE International Conference on (Vol. 3, pp. III-337). IEEE.

12- Choi, Y.S., Zaijun, P., Kim, S.W., Kim, T.H. and Park, C.B., 2006, November. Salient motion information detection technique using weighted subtraction image and motion vector. In Hybrid Information Technology, 2006. ICHIT'06. International Conference on (Vol. 1, pp. 263-269). IEEE.

13- Image Processing and Object Detection. (2015). International Journal of Applied Research, p.4.

14- Thomas, A.M., Ashraf, A., Lai, L., Mathew, M.J. and Jayashree, M.J., 2011, July. Security enhancement using motion detection. In Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on (pp. 552-557). IEEE.

15- Balsaraf, S. and Joshi, U. (2013). Implementation of Motion detection and Tracking based on Mathematical Morpohology. International Journal of Computer Applications, 80(15), pp.22-28.

16- Sahasri, M. and Gireesh, C., Object Motion Detection and Tracking for Video Surveillance. International Journal of Engineering Trends and Technology (IJETT), pp.161-164.

17- C. Sureindar and Dr. P. Saravanan, Nov-2015. Motion Detection Using Optical Flow on Raspberry PI.

18- Irianto, K.D. and Ariyanto, G., 2009. Motion Detection Using OpencvWith Background Subtraction and Frame Differencing Technique.

APENDEX A:

SOURCE CODE

```
------------------------------------------------------------
#include<opencv2/opencv.hpp>
using namespace cv;
#define buffsize 100

int main ()
{
//For image.
Mat FrameCopy;
Mat Frame;
bool status = true;
VideoCapture Cap("320_180.mp4");
Mat BK[buffsize];
for (int i = 0; i<buffsize; i++)
{
BK[i] = Mat (Size (320,180), CV_8UC1,
Scalar (255));
}
int Width, Height;
int FrameCount = 0, BKF=0;//Time Window size is   initialized to
buffsize.
//For Video play.
if (! Cap.isOpened())
return -1;
Mat tmpimage (Size (320,180), CV_32FC1,
Scalar (0)) ;//Accumulated value for averaging.
Mat tmpimage_new (Size (320,180), CV_32FC1,
Scalar (0));
Mat tmpdata (Size (320,180), CV_32FC1, Scalar (0));
while (status)
```

```
{
Cap >> Frame;
if (Frame.empty())
break;
Width = Frame.cols;
Height = Frame.rows;
Imshow ("Input Video", Frame);
FrameCount++;
Printf ("FrameN0=%d\n", FrameCount);
cvtColor (Frame, FrameCopy, CV_BGR2GRAY);
imshow ("gray", FrameCopy);
if ((FrameCount-1) <buffsize)
{
Memcpy (BK [(FrameCount - 1) % buffsize].data, FrameCopy.data,
sizeof(uchar)*Width*Height);
Accumulate (BK [(FrameCount - 1) % buffsize], tmpimage);
tmpdata = tmpimage / (FrameCount*255);
imshow ("BKground", tmpdata);
waitKey (25);
}
else
{
if ((FrameCount-1) >= buffsize)
{
int idx = (FrameCount - 1) % buffsize;
memcpy(BK[idx].data, FrameCopy.data,
sizeof(uchar)*Width*Height);
for (int j = 0; j<buffsize; j++)
{
accumulate(BK[j], tmpimage_new);
}
tmpdata = tmpimage / (buffsize * 255);
imshow ("BKground", tmpdata);
waitKey (25);
    }}}
```

```
   return 0;}
```

APENDEX B:

SOURCE CODE 2

```
#include<opencv2\opencv.hpp>
using namespace cv;
#define Buffsize 50
int main ()
{
Mat FrameCopy, OldFrame;
Mat Frame;
bool status=true;
VideoCapture Cap("320_180.mp4");
Mat BK[Buffsize];
for (int i = 0; i < Buffsize; i++)
{
BK[i]=Mat (Size (320,180), CV_8UC1, Scalar (255));
// reserve a location in memory as gray scale image
}
int width, hight;
int Frame_Counter =0, BKF=0;
// For Reading Video from Camera
if (! Cap.isOpened())
return -1;
while (status)
{
Cap>>Frame;
If (Frame.empty())
break;
width = Frame.cols;
hight = Frame.rows;
```

```
Mat tmpimage (Size (width, hight), CV_32FC1,
Scalar (0));
MatimageF32(Size (width, hight), CV_32FC1,
Scalar (0));
Mat tmpdata (Size (width, hight), CV_32FC1,
Scalar (0));
Mat tmpdata8U (Size (width, hight), CV_8UC1,
Scalar (0));
Mat BGImg (Size (width, hight), CV_32FC1,
Scalar (0));
Imshow ("Original Video", Frame);
Frame_Counter ++;
Printf ("Frame Number %d\n", Frame_Counter);
cvtColor (Frame, FrameCopy, CV_BGR2GRAY);
imshow ("Gray Image", FrameCopy);
if (Frame_Counter <Buffsize)
{
BK[(Frame_Counter-1) %Buffsize].
convertTo (imageF32, CV_32F,1.0/255);
accumulate (imageF32, tmpimage);
tmpdata = tmpimage/Frame_Counter;
memcpy (BGImg.data, tmpdata.data,
sizeof(float)*width*hight);
BGImg.convertTo(tmpdata8U, CV_8U,255);
Imshow ("BackRound", tmpdata8U);
           waitKey (30);
}
else
{
if(Frame_Counter>=Buffsize)
{
int idx=(Frame_Counter-1) %Buffsize;
m*emcpy(BK[idx].data,FrameCopy.data,
sizeof(uchar)*width*hight);
for (int j=0; j< Buffsize; j++)
{
BK[j]. convertTo (imageF32, CV_32F,1.0/255);
Accumulate (imageF32, tmpimage);
}
tmpdata = tmpimage/Buffsize;
memcpy (BGImg.data, tmpdata.data,
sizeof(float)*width*hight);
BGImg.convertTo(tmpdata8U, CV_8U,255);
Imshow ("BackRound", tmpdata8U);
waitKey (30);
}
}
}
return 0;
}
```

APENDEX C:

SOURCE CODE 3

```
#include<opencv2/opencv.hpp>
using namespace cv;
#define buffsize 200
#define Threshold 40
#define USE_CAMERA
int main ()
{
//For image.
Mat FrameCopy;
Mat Frame;
//Defining a silhouette set
vector<vector<Point>> Contours;
vector<Vec4i> Hierarchy;
bool status = true;
#ifdef USE_CAMERA
VideoCapture Cap;
Cap.open(3);
#else
VideoCapture Cap("TestVideo.avi");
#endif
//End.
//get width of camera, the width is same to frame's width.
int Width = Cap.get(CV_CAP_PROP_FRAME_WIDTH);
//get height of camera, the height is same to
frame's height.
int Height = Cap.get(CV_CAP_PROP_FRAME_HEIGHT);
printf ("Width = %d \n Hight \n= %d", Width, Height);
```

```
Mat BK[buffsize];
for (int i = 0; i < buffsize; i++)
{
BK[i] = Mat (Size (Width, Height), CV_8UC1,
Scalar (255));
}
int FrameCount = 0;//Time Window size is initialized to
buffsize.

//For Video play.
if (! Cap.isOpened ())
return -1;

//Accumulated value for averaging.
Mat tmpdata (Size (Width, Height), CV_32FC1,
Scalar (0));

Mat tmpimage (Size (Width, Height), CV_32FC1, Scalar (0));

Mat BkImg8U (Size (Width, Height), CV_8UC1,
Scalar (0));

Mat Diff (Size (Width, Height), CV_8UC1,
Scalar (0));

Mat ResultBinary (Size (Width, Height), CV_8UC1, Scalar (0));

// get results after erosion
Mat erosion_mask (Size (Width, Height), CV_8UC1, Scalar (0));

//get results after dilation
Mat dilation_mask (Size (Width, Height), CV_8UC1, Scalar (0));
while (status)
{
Mat result (Size (352, 288), CV_8UC1,
Scalar (0));
Cap >> Frame;
if (Frame.empty())
break;
FrameCount++;
cvtColor (Frame, FrameCopy, CV_BGR2GRAY);
imshow ("gray", FrameCopy);
if ((FrameCount - 1) < buffsize)
{
Memcpy  (BK  [(FrameCount  -  1)  %  buffsize].data,  FrameCopy.data,
sizeof(uchar)*Width*Height);

Accumulate (BK [(FrameCount - 1) % buffsize], tmpimage);

tmpdata = tmpimage / (FrameCount * 255);
imshow ("BKground", tmpdata);
tmpdata. Convert To (BkImg8U, CV_8U, 255);
for (int n = 0; n < Height; n++)
for (int m = 0; m < Width; m++)
      {
uchar Diff8U = abs(BkImg8U.at<uchar> (n, m) -  FrameCopy.at<uchar>
(n, m));
Diff.at<uchar> (n, m) = Diff8U;
ResultBinary.at<uchar> (n, m) = Diff8U >= Threshold? 255: 0;
}
waitKey (25);
}
else
{
if ((FrameCount - 1) >= buffsize)
{
Mat tmpimage_new (Size (Width, Height), CV_32FC1, Scalar (0));


int idx = (FrameCount - 1) % buffsize;
memcpy(BK[idx].data, FrameCopy.data,
sizeof(uchar)*Width*Height);
for (int j = 0; j < buffsize; j++)
      {
accumulate(BK[j], tmpimage_new);
      }
tmpdata = tmpimage_new / (buffsize * 255);
imshow ("BKground", tmpdata);
tmpdata. convertTo (BkImg8U, CV_8U, 255);
for (int n = 0; n < Height; n++)
for (int m = 0; m < Width; m++)
      {
uchar Diff8U = abs(BkImg8U.at<uchar> (n, m) -
FrameCopy.at<uchar> (n, m));
Diff.at<uchar> (n, m) = Diff8U;
ResultBinary.at<uchar> (n, m) = Diff8U >= Threshold? 255: 0;
      }
waitKey (25);
      }
      }
imshow ("Binary", ResultBinary);

//1. Erosion operation.
erode (ResultBinary, erosion_mask, Mat (),
Point (-1, -1), 1);
//imshow ("erosion_mask", erosion_mask);

//2. Dilation Operation
dilate (erosion_mask, dilation_mask, Mat (), Point (-1, -1), 6);
//imshow ("dilation_mask", dilation_mask);

//3. Compute Contoures and size of object in binary image.
find    Contours    (dilation_mask,    Contours,    Hierarchy,
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE, Point (0, 0)); int size =
Contours. Size ();
```

```
//4. Display frame number and number of rectangles
printf ("FrameN0=%d size=%d\n", FrameCount, size);

//5. Show camera image with bounding box
for (int i = 0; i < Contours. Size (); i++)
{
Rect r = boundingRect(Mat(Contours[i]));
rectangle (Frame, r, Scalar (0, 0, 255), 2);
}
imshow ("Difference", Diff);
imshow ("Input Video", Frame);
}
return 0;
}
```