

# **Development of moving object detection and tracking**

By

Muhammad Kamal Rizal bin Zainon

18581

Supervisor: Dr. Azrina Bt. Abd Aziz

Co Supervisor: Patrick Sebastian

Dissertation submitted in partial fulfillment of

The requirements for the

Bachelor of Engineering (Hons) (Electrical & Electronic)

JANUARY 2017

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

**THE DEVELOPMENT OF MOVING OBJECT DETECTION AND TRACKING**

by

Muhammad Kamal Rizal bin Zainon  
18581

A project dissertation submitted to the

Electrical & Electronics Engineering

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONICS)

Approved by,

---

(Dr. Azrina Bt. Abd Aziz)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

JANUARY 2017

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

---

MUHAMMAD KAMAL RIZAL BIN ZAINON

## **ABSTRACT**

Object detection is a process of locating and identifying the position of object in the frame of a video sequence. While object tracking is the process of locating the object of interest over time in the sequence of video frame. The steps of object detection and tracking is start with detection of object of interest, followed by tracking of the object over time and lastly the analysis of trajectory of tracked object. Object detection and tracking have been implemented mostly in the field of control, security and surveillance like vehicle navigation and human robot interaction. But, object detection and tracking is not perfect as problems still emerged such as dynamic background, for multiple objects and expensive computational hardware. This work explores the commonly used object detection and tracking technique. The suitable method for static camera was implemented in Matlab and OpenCV on a sample pedestrian video. In the end, the results of both software were shown.

## **ACKNOWLEDGEMENT**

First and foremost, thanks to God because He has given me strength and opportunity to complete this work just in time. Even with a lot of difficulties to complete the task, I still manages to finish it.

I would like to express my gratitude to Dr. Azrina Bt. A Aziz for his supervision, encouragement, suggestions and trust throughout the development of this work. I am grateful for to her for reading my writing and helpful comments.

I am also thankful to Sir Patrick Sebastian as the Co Supervisor for supporting the work and words of encouragement.

Without the support of my colleagues, this research would not be that much enjoyable for me. I am especially grateful to them for their contributions, cooperation and discussions.

## TABLE OF CONTENTS

	Page
LIST OF TABLES	7
LIST OF FIGURES	8
ABBREVIATIONS	9
1 INTRODUCTION	
1.1 Background of detection and tracking	10
1.2 Problem Statement	12
1.3 Objectives and Scope of Study	12
2 LITERATURE REVIEW	
2.1 Overview of Detection and Tracking	13
2.2 Current Detection Methods	15
2.3 Existing Tracking Methods	17
2.4 Challenges of Detection and Tracking	21
2.5 Proposed Methods	
2.5.1 Overview of Background Subtraction	22
2.5.2 Detection and Tracking Model	
2.5.2.1 Color space.	23
2.5.2.2 Conversion to grayscale	23
2.5.2.3 Adaptive Background Subtraction Model	24
2.5.2.4 Thresholding	25

	Page	
3	METHODOLOGY	
3.1	Dataset	28
3.2	Application Environment	28
3.3	Video Scenarios	29
3.4	Algorithm of Detection and Tracking	30
3.5	Project Flow	32
4	RESULTS AND DISCUSSIONS	
4.1	Background Subtraction Result	33
4.1.1	MATLAB Detection and Tracking Result	34
4.1.1	OpenCV Detection and Tracking Result	35-36
4.2	Result Analysis	37
4.2	Performance Evaluation	38-40
5	CONCLUSIONS AND RECOMMENDATION	41
	LIST OF REFERENCES	42-43
	APPENDICES	44-50

## LIST OF TABLES

Table		Page
1	Object detection methods	11
2	The process in object detection	13
3	Current object classification methods	15
4	Current detection methods	16
5	Current tracking methods	18-20
6	The challenges in detecting and tracking motion	21
7	Type of thresholding	26
8	The hardware and software used	28
9	Details of the sample videos used	29
10	The processing time of the experiments	33
11	The frame based metrics description	38
12	The calculation of the frame based metrics	39
13	The performance evaluation	40



## LIST OF FIGURES

Figure		Page
1	The application of object detection and tracking	10
2	An object tracking sample	12
3	Object detection process	13
4	Classification of tracking methods	15
5	An example of background subtraction	22
6	Color Space Illustration	23
7	Threshold Illustration	25
8	Block diagram of object detection and tracking	27
9	Flow of the detection and tracking algorithm	30
10	MATLAB output frames for sample 1 video	34
11	MATLAB output frames for sample 2 video	34
12	MATLAB output frames for sample 3 video	34
13	MATLAB output frames for sample 4 video	34
14	OpenCV output frames for sample 1 video	35
15	OpenCV output frames for sample 2 video	35
16	OpenCV output frames for sample 3 video	36
17	OpenCV output frames for sample 4 video	36
18	MATLAB false positive in sample 3	37
19	MATLAB false positive in sample 4	37

## ABBREVIATIONS

RGB	Red, green, blue
CMY	Cyan, magenta, yellow, key (black)
HSV	Hue, saturation, value
HSI	Hue, saturation, intensity
TN	True Negative
TP	True Positive
FN	False Negative
FP	False Positive
TG	Total Ground Truth
TF	Total Number of Frames

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

Today, motion detection and tracking play an important role in our life as they have been implemented in various applications like security cameras, traffic monitoring, object avoidance, and automatic guidance [14]. In fact, many other applications in the monitoring field and control field are based on motion detection and tracking. . The research on object detection and tracking gain attention as many problems regarding human safety and system monitoring is in need and can be solved with the deployment of this system. The commonly applications of object detection and tracking can be seen in Figure 1. The applications of moving objects detection and tracking can be seen in public places such as shopping malls, metro stations, airports and independent surveillance request.

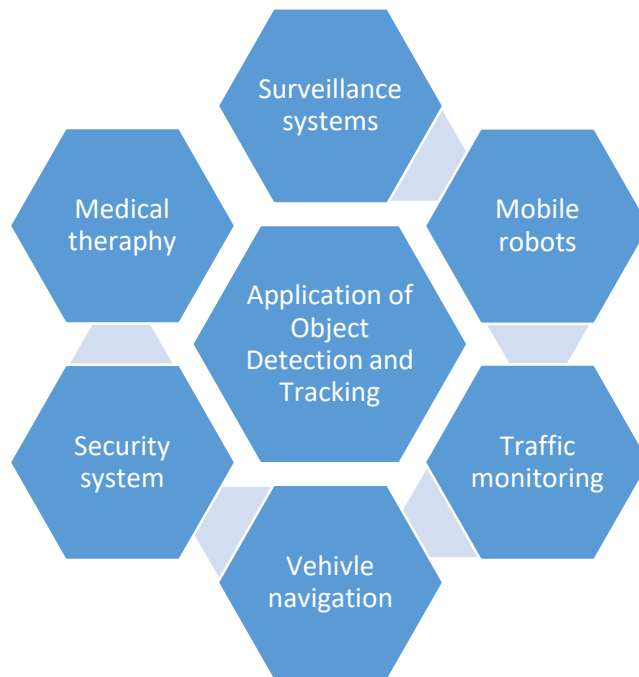


Figure 1: The application of object detection and tracking [14].

Object detection can be defined as the process to identify objects of interest in the video sequence and to cluster pixels of these objects. There are three available object detection methods which are frame differencing, optical flow and background subtraction. The detailed information of these methods can be referred on Table 1.

Table 1: Object detection methods.

Methods	Explanation
Frame differencing	Two consecutive images will be picked and compared to determine the motion of objects.
Optical Flow	The image optical flow field will be calculated. Then the clustering process will be done based on optical flow distribution characteristics of image.
Background subtraction	Background modelling will be made to yield a reference model. The video sequence will be compared with the reference model to determine object position throughout the frames.

Object tracking can be defined as the process of predicting the trajectory of the detected object in the sequence of images. The object will be tracked from they first appear in the frame to the last frame of video. Usually, the tracked object will be predicted and updated from frame to frame. The categories of object tracking can classified into point tracking, kernel tracking and silhouette tracking. In point tracking, the object will be featured as points and tracking will be detected from the previous knowledge of points like position and motion. While in kernel tracking, the appearance and shape of object will be featured when in the consecutive frames. The kernel shape can be rectangular, round or square. In silhouette tracking, an encoded information (appearance density or shape density) inside of the object region will be estimated to track object's position in frames. An example of object tracking can be seen in Figure 2.



Figure 2: An object tracking based on research [14].

## 1.2 Problem Statement

The current object detection and tracking techniques still experience the problems of noise, changes in light, shape, and occlusion. Hence, the design and development of a real-time detection and tracking of moving object are challenging as the designer needs to solve these problems while minimize the processing time and improving data accuracy into consideration factors. The system should be able to track and detect the object motion given these constraints.

In this work, the best method for object detection and tracking will be explored and studied. The selected method will be used to tackle two common challenges in object detection and tracking which is occlusion.

## 1.3 Objectives and Scope of Study

The objectives of this work are:

- To implement existing moving object detection and tracking for the static video.
- To explore the step needed for object detection and tracking for the static video.
- To implement the proposed detection and tracking methods in the Matlab and OpenCV.

This work will focus on the theory and current method of object detection and tracking for the static camera and moving object. First, Matlab will be used to give author experience and idea about the existing detection and tracking method available in the documentation. Then OpenCV will be used to compare the tracking result as it is more suitable to image processing and can be implemented in real application.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Overview of Detection and Tracking

Object detection can be defined as the process of clustering pixels of object of interest in the video. Detection of object is done by first identifying the common features like simple distance, position based features, shape, color and texture based features. Then the pixel features will be clustered to detect object motion. The process like pre-processing, segmentation, foreground and feature extraction will be done after this. The process of object detection can be seen in Figure 3 and detailed explanation of its flow can be referred in Table 2.

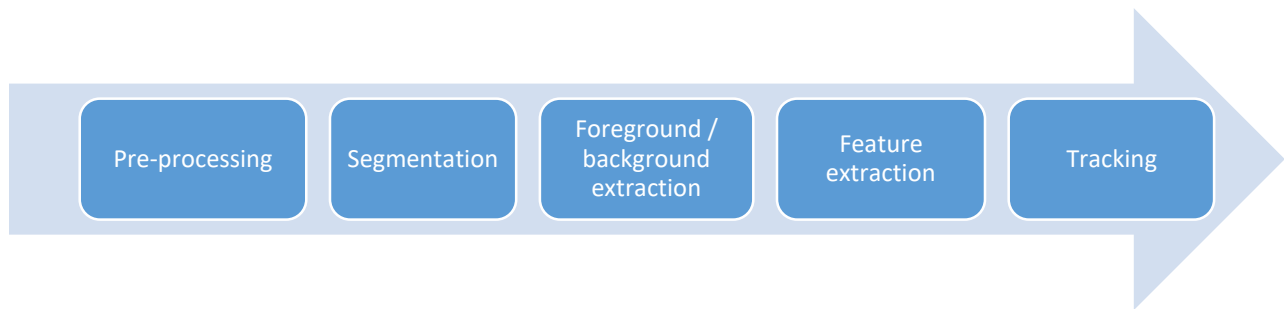


Figure 3: Object detection process.

Table 2: The Process in object detection.

Process	Explanation
Pre-processing	This process is done to yield the object of interest features.
Segmentation	The process of segmenting relevant information in the frame.
Foreground / Background Extraction	The object of interest and all other object in the background will be differentiated in this step.
Feature Extraction	Features like shape, color or texture will be extracted to determine object motion.

The tracking of the object can be defined as the ability of the system to track moving objects as well as its position, velocity and acceleration with the changes of images frame [8]. The system needs to be able to sense physical movement in a given area by measuring a change in speed or vector of an object. Besides, the tracking methods should be able to cope with the changes of environment like object motion, occlusion and orientation. Object tracking can be classified into point tracking, kernel tracking and silhouette based tracking as shown in Figure 4. The point tracking requires the object of interest point to be tracked in every frame while the other methods require detection only when the object first appear in the frame.

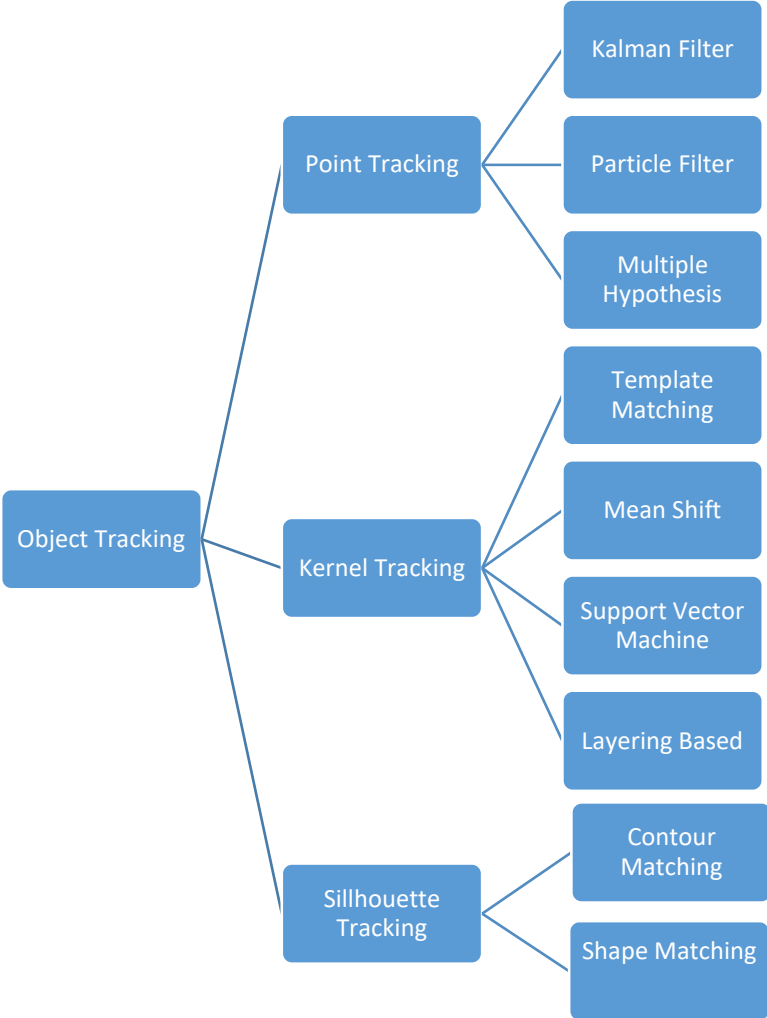


Figure 4: Classification of tracking methods.

## 2.2 Current Detection Methods

Detection methods focus on identifying and clustering pixels of object of interest in the video. A comparative study of object detection based on previous study is shown in Table 4. However the extracted information may be different multiple objects like vehicles, pedestrian and animals in the scene. Hence it is required to classify the detected objects to further track and analyze its behaviors. The classification of object is based on color, shape, motion and texture based. Detailed explanation of object classification is shown in Table 3.

Table 3: Current object classification method [6].

Method	Explanation
Color based	Accuracy of this method is high as the color in the scene usually constant throughout entire frame. In addition, it becomes a choice for designer as it has low computational cost and easy to be implemented. Color histogram will be used to express the specific color based on image. However, it not reliable in dynamic background where the external environment changes.
Shape based	The object features like edge or geometry can be specified to be extracted. The extracted data will be stored in histogram and object classification will be done in every frame.
Motion based	Its accuracy depend on residual flow of the images. This method is not reliable to classify non-moving human as they have low residual flow.
Texture based	Gaussian Mixture Model will be used to present color distribution and gradient orientation of the images.



Table 4: Current detection method [6].

Detection Method		Working Principle	Advantages and Disadvantage
Background Subtraction	Frame Differencing	The frame will be compared and subtracted with the reference frame based on certain threshold value.	The method is easy to be implemented. However, it very sensitive to external environment changes and need static object in the background [6].
	Approximate Median	Background frame will be calculated from the median of the buffered frame. Then foreground pixel is produced by comparing the background frame with the current frame based on pixel value.	The method does not require background modeling. The computational time may take longer than it need recent pixel values.
	Gaussian Mixture	An intensity histogram of current frame will be compared with the reference frame.	It requires low memory for computational. But, it cannot handle the object when there is noise.
	Running Gaussian average	Each color channel in the frame will be modeled and expressed for the intensity. The images will be compared based on mean and standard deviation for each channel.	This method is really suitable for real time applications. The computational time is longer than others.
Temporal Differencing		Differences images will be created by comparing several adjacent frames.	In detection moving object, this method is simple and easy [12]. But, some pixels of the moving object cannot be detected if there are some sudden changes in dynamic background.
Optical Flow		The vector characteristic of the moving object will be compared.	This method is better for moving camera. The drawback is the computational algorithms is complex and very sensitive to noise.

### **2.3 Existing Tracking method**

Object tracking can help to approximate the trajectory of object of interest in the video. It is needed to generate the path of the object in the sequence of images and may prove to be helpful when certain information need to be extracted. The information can be objects motion, orientation and pattern. In addition, tracking can be useful when there are noises in the scene. For example, in the present of occlusion, tracking methods can still track object position even the object of interest is overlapped with other object. A comparative study based on past research is shown in Table 5.

Table 5: Current tracking method [6].

Tracking Method		Working Principle	Capabilities
Kernel Tracking	Template matching	This algorithms is based on Region of Interest (ROI). The object features will tracked by comparing video with a reference images.	This method can cope up with tracking of single image and partial occlusion object. However the equivalent model must be defined first for each region of interest.
	Layering Based	This kernel based method is capable to track multiple moving objects fully occlusion object. The object's position will be calculated based on the foregoing motion and shape features	Multiple objects and full occlusion objects can be tracked. But it requires a specific limit model for each pixel.
	Mean Shift	The traits of the object like shape, color and texture will be defined first by a histogram. Then the matching of the traits will be done in image sequences.	It suitable for real time application as it have simple computation. However the iterations get into local easily.
	Support vector machine	Support vector machine will make use the positive and negative values in the video.	This method is useful for tracking single object and partial occlusions. Physical initialization and training must be defined first.

Tracking Method		Working Principle	Capabilities
Point Tracking	Kalman Filter	This algorithm used Optimal Recursive Data Processing algorithms. The estimation of object tracking within the video will be done by using feedback control system. The past, present and future of object can be located by using the system.	This method is capable to track point in noisy environment.
	Particle Filter	This method will implement texture, colors and contours to map the object. The object variable like noise and particles will be generated first before it make prediction and tracking.	It can produce good result for complex scene and occlusion object. The downside of this method is it requires big calculation. Hence it not suitable for real time application.
	Multiple Hypothesis	A few iterations will be made on the sequence of the images based on current hypothesis.	The entries of new moving object and exit of current object can be tracked by using this algorithms. The computational time depend on the memory provided.

Tracking Method		Working Principle	Capabilities
Silhouette Tracking	Contour Matching	The iteration of the primary contour from the previous frame will be tracked to the next frame. The approaches to define the contours are state space models and minimization techniques.	The tracked object can be modelled in this algorithms. But it requires time for state space estimation.

## 2.4 Challenges in Detection and Tracking

Although many methods have been created and implemented, the designer still struggle to create high accuracy, faster processing time and high performance detect and tracking system. The challenges lie in the types of detected object and the background situation. In same case, the detected object may change dramatically along with the background [6]. Because of this, the system usually is designed to work perfectly in specific condition. For example, it is not difficult to detect and track the same color pixels of the objects in the images. But the existence of another external environment with the same pixel colors can lead to false information that should be extracted.

The designer usually implements several methods to improve the accuracy of the data with the consideration of computational time. The challenges in designing high performance surveillance system are commonly based on object deformations, illumination and occlusion as explained in Table 6.

Table 6: The typical challenges in tracking and detecting object motion [6].

Challenges	Explanation
Illumination	This happens when there is sudden changes of light in the environment like transition of weather from sunny to cloudy.
Occlusion	The overlap of the tracking object with another object.
Camouflage	Camouflage may happen when the background share similar features with the detected object. This problem arise for temporal differencing method.
Motion	The jittering of the image database resulted from vibration of camera may results in false extraction data.
Object motion	In case of temporal differencing method, the object cannot be detected if its motion is slow. Besides, a ghost trail will be seen if the motion is too fast.
Shadow	The present shadow from foreground objects will make the classification and segmentation of the object difficult.

## 2.5 Proposed Method

Background subtraction has been chosen as they are appropriate for static camera and have good computational time. The absolute difference technique of background subtraction has been chosen for this work.

### 2.5.1 Overview of Background Subtraction

Background subtraction is a method that applies extraction between foreground image and background image for processing. The foreground image is the object of interest like pedestrian, vehicle and animal. Background subtraction method is commonly used in the detection and tracking from static cameras like surveillance camera and traffic monitoring. However, this method is not suitable for the real time application and performs worse if there is a change in the scene like illumination or weather. This may result in faulty output. An example of background subtraction based on frame differencing [15] can be seen on Figure 5.

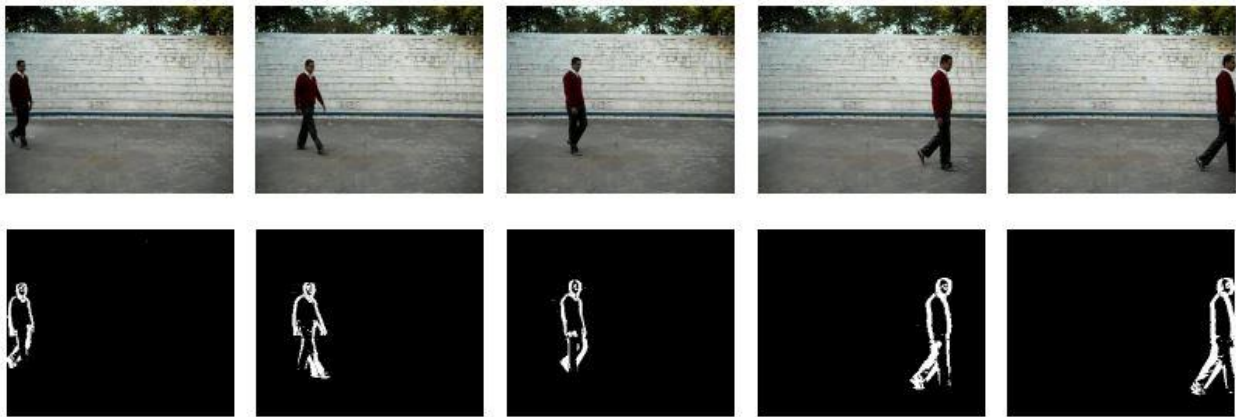


Figure 5: An example of Background Subtraction [15].

## 2.5.2 Detection and Tracking Model

### 2.5.2.1 Color space.

A color space is an abstract mathematical model which describe the range of colors as tuples of numbers. Color space is an elaboration of the coordinate system and sub-space where each color in the system is represented by a single dot. Example of color space include RGB, CMY, HSV and HIS. The value of pure red is (255, 0, 0), pure blue (0, 0,255) and pure green (0, 255, 0) [23].

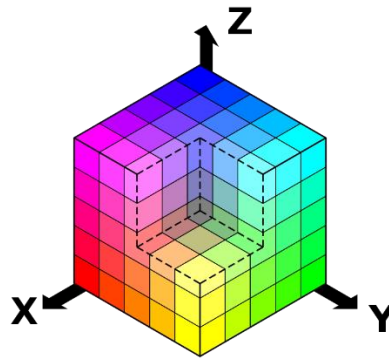


Figure 6: Color Space Illustration [23].

### 2.5.2.2 Conversion to gray scale

Color space conversion is the translation of the representation of a color from one basis to another. This can be done by converting an image from one color space to another color space. Gray scale images are rendered in black, white and gray in between. A weighted average of the red, green and blue will be computed to convert from RGB color space to its equivalent grayscale values [22].

Since the pure blue has the darkest color values followed by pure red and the pure green, it will have least weight compared to other. The grayscale weighted average,  $x$ , is given by the formula:

$$X = 0.299R + 0.587G + 0.11B \quad \text{Equation (1)}$$



### 2.5.2.3 Adaptive Background Subtraction Model

In this technique, the background subtraction will utilize a background model with the first few frames of video input. An absolute difference of the pixel value between every current frame and the reference frame is taken to find out the detection mask. The absolute difference output will be threshold and filtered to fill the empty hole in the pixels.

$$D_t(s) = \begin{cases} 1, & d(I_{s,t}, B_s) > \tau \\ 0, & \text{otherwise} \end{cases} \quad \text{Equation (2)}$$

Where  $D_t$  is the detection mask,  $I_{s,t}$  the time,  $d$  is the distance between  $I_{s,t}$ , the video frame and  $B_s$  the background pixel and  $\tau$  is threshold [20]. The equation (2) address every pixel at time  $t$  whether their color is different from the background or not based on the threshold value.

For the RGB color space, the background subtraction can be modelled based on equation (3),(4),(5),(6),(7),(8).  $\alpha$  act as the updating constant,  $d_0$ ,  $d_1$ , and  $d_2$  as detection mask and foreground pixel is detected by the thresholding of red, blue and green color [20].

$$B_{s,t+1} = (1 - \alpha) B_{s,t} + \alpha. I_{s,t}, \quad \text{Equation (3)}$$

$$d_0 = |I_{s,t} - B_{s,t}| \quad \text{Equation (4)}$$

$$d_1 = |I_{s,t}(R) - B_{s,t}(R)| + |I_{s,t}(G) - B_{s,t}(G)| + |I_{s,t}(B) - B_{s,t}(B)| \quad \text{Equation (5)}$$

$$d_2 = (I_{s,t}(R) - B_{s,t}(R))^2 + (I_{s,t}(G) - B_{s,t}(G))^2 + (I_{s,t}(B) - B_{s,t}(B))^2 \quad \text{Equation (6)}$$

$$d_\infty = \max\{|I_{s,t}(G) - B_{s,t}(G)|, |I_{s,t}(G) - B_{s,t}(G)|, |I_{s,t}(B) - B_{s,t}(B)|\} \quad \text{Equation (7)}$$

### 2.5.2.4 Thresholding

Thresholding is the process where the grayscale images will be converted into a binary mask. This is done to separate the detected object from the foreground image. The output of the Thresholding images that hold 1 value will be object of interest while the pixels that hold 0 will be the foreground images.

Thresholding process can be represented by using Figure 7. Let  $src(x,y)$  be pixels with intensity values of source image, the  $maxVal$  is the maximum pixel intensity and the threshold value is represented by blue line [21].

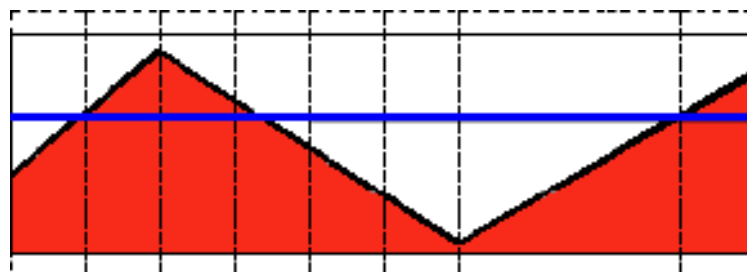
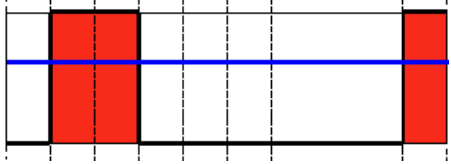

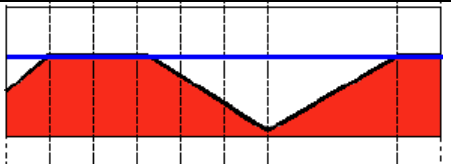
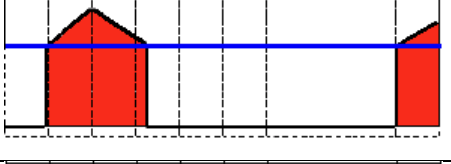
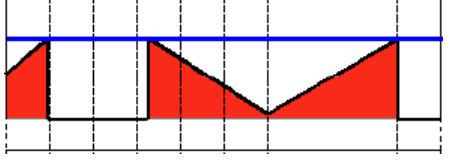


Figure 7: Threshold illustration.

There are five different types of Thresholding in OpenCV which can be referred at the Table 7 [21].

Threshold types	Graph	Threshold functions	Description
Threshold Binary		$dst(x, y) = \begin{cases} maxVal, & \text{if } src(x, y) > thresh \\ 0, & \text{otherwise} \end{cases}$	Pixel will be maxVal if pixel src(x,y) is greater than thresh, otherwise the pixels are zero.
Threshold Binary, Inverted		$dst(x, y) = \begin{cases} 0, & \text{if } src(x, y) > thresh \\ maxVal, & \text{otherwise} \end{cases}$	Pixel will be zero if pixel src(x,y) is greater than thresh, otherwise the pixel is maxVal.
Truncate		$dst(x, y) = \begin{cases} threshold, & \text{if } src(x, y) > thresh \\ src(x, y), & \text{otherwise} \end{cases}$	Pixel will be set to threshold if pixel value is greater than threshold, otherwise it is ignored.
Threshold to Zero		$dst(x, y) = \begin{cases} src(x, y), & \text{if } src(x, y) > thresh \\ 0, & \text{otherwise} \end{cases}$	Pixel will be src(x,y) if pixel is greater than thresh, otherwise it will be zero.
Threshold to Zero, Inverted		$dst(x, y) = \begin{cases} 0, & \text{if } src(x, y) > thresh \\ src(x, y), & \text{otherwise} \end{cases}$	Pixel will be zero if pixel src(x,y) is greater than thresh, otherwise it will be zero.

### CHAPTER 3

### METHODOLOGY

The datasets, hardware, software and video scenarios are presented in detecting and tracking moving object by using the proposed methods mentioned in Chapter 2. The computational time, accuracy, strengths and weakness of the proposed methods are discussed. The overall work of the detection and tracking is shown in Figure 3.1.

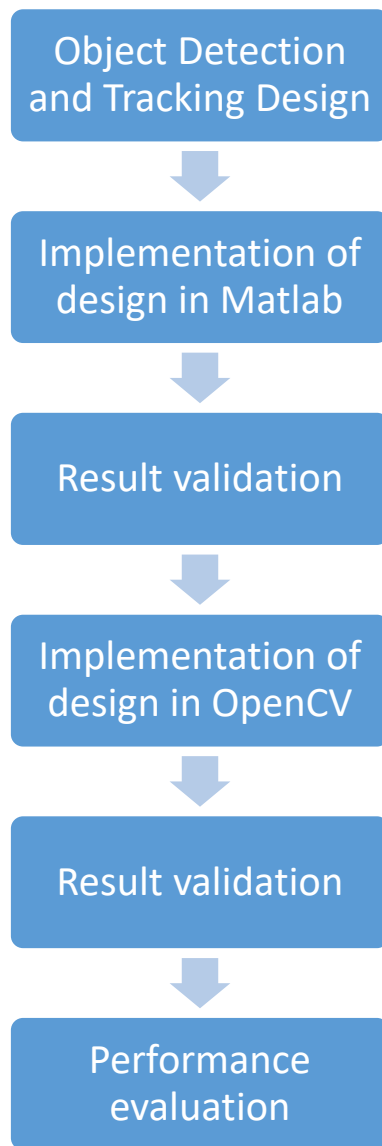


Figure 8: A block diagram of object detection and tracking.

### 3.1 Dataset

The samples videos [17] have been used in this study to evaluate the performance of background subtraction. The videos consisted of moving pedestrian in several scenarios which are indoor, outdoor, single moving object and multiple moving objects. Also, all the videos taken utilizes static camera with little noise in the background. This work focuses on the moving pedestrian video from the work [17] as it provides with the ground truth that will used for performance evaluation. Frame based metric method [18 ] will be used in the performance evaluation where the frame metrics like true positives (TP), false positives (FP) and false negatives between the ground truth and the experiment will be compared. The details of the videos used can be seen in the Table 3.2.

### 3.2 Application Environment

The proposed detection and tracking algorithms were implemented using Python version 3.5.3. The code was written by using Python language because of it robustness and the performance evaluation was done by using Matlab. The detailed hardware and software used is shown in Table 8.

Table 8: The hardware and software used.

Hardware	Processor: Intel i7-4500 Ram: 4 GB OS: Windows 10	The laptop specification used in detection and tracking of moving object.
Software	OpenCV 3.5.2	Open Source Computer Vision (OpenCV) is software with the focus on real-time computer vision [19].
	MATLAB	MATLAB (matrix laboratory) is a software used to express mathematical computational and implementation of algorithms.

### 3.3 Video Scenarios

The moving object detection and tracking will be done under several situations like in walking activity and waving hand activity. The experiment was done under several cases to ensure the algorithms able to work if they were to be implemented in real life. The experiment will consider the real life problem like changes in weather and detection and tracking of different or specific moving object. All frames in the experiment will be stored but only the first up to 100 frames will be compared with the ground truth data frames in sample 1 and sample 2 as they are for testing of the system. For the sample 3 and sample 4, all frames will be compared.

Table 9: Videos Environment

Video		Description	Frame size	Frame rate	Duration
Single moving object	1	Walking activity	640x480	25	27 sec
	2	Waving activity	640x480	25	15 sec
Multiple Moving object	3	Moving pedestrian in random direction	720x576	25	53 sec
Moving Object with occlusion	4	Car thief scene	1392x1040	14	1 min 46 sec

### 3.4 Algorithm of Detection and Tracking in OpenCV

The flow of the detection and tracking of moving pedestrians is shown in the Figure 9.

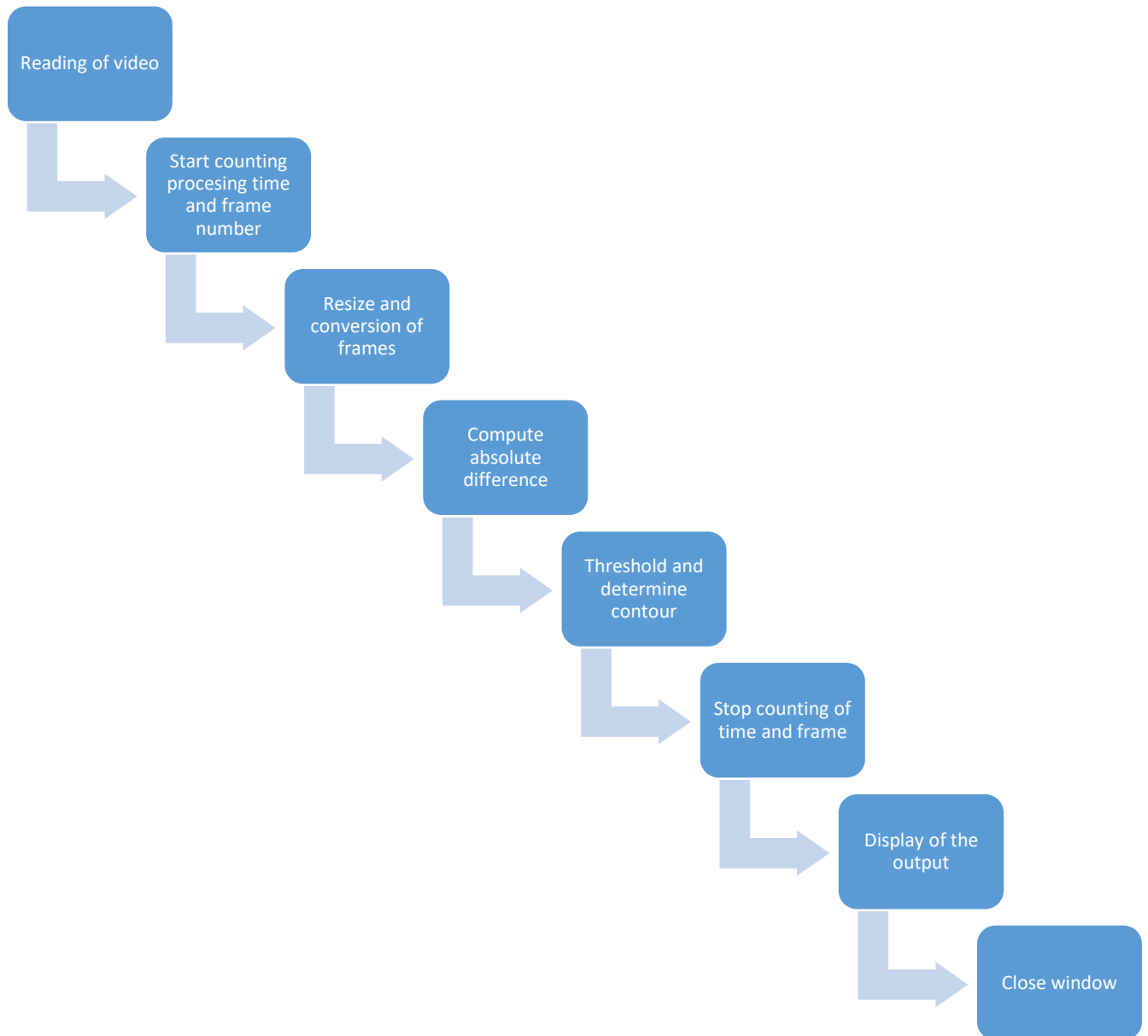


Figure 9: The flow of the algorithm used in detecting and tracking of moving object.

The detection and tracking algorithms started with the reading of a sample video. The user can define the video that wants to be used with its path. After that, start time will be recorded to record the timing. All videos will be resized to 640 x 480 and converted to grayscale color space to reduce processing time during detection and tracking moving objects. Then, a background model which is the first frame of the video will be defined. Absolute difference in terms of pixel value will be compared between the second frame and next frames of the videos until the last frame with the defined background model. Their frames result will be threshold to fill the holes between pixels. Bounding box will be defined and drawn based on the contour of threshold frames. Stop time will be done after this. The processing time will be computed based on the difference between start time and stop time. The video outputs consist of the absolute difference frames, threshold frames and bounding box frames will be shown at the end of the code. Also, the images for every process will be saved in their respective folder.



### 3.3 Project flow (Gantt chart)

Task \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project allocation	■	■	■											
Literature review research			■	■										
Familiarize with Matlab					■	■	◆							
Input data acquisition							■							
Object detection design								■						
Object classification design									■					
Motion tracking design										■				
Implement the previous design together										■	◆			
Data collection												■	■	
Analyze and discussion													◆	■

◆ : Milestones    ■ : Activities

Chart 1: Timeline for the project

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Background Subtraction Result

The experimental results for MATLAB are shown in the Figure 10, Figure 11, Figure 12 and Figure 13 respectively for a single walking pedestrian, multiple walking pedestrians and video with an occlusions. The result of OpenCV are shown in Figure 14, Figure 15, Figure 16 and Figure 17. These figures consisted of 3 rows which are the absolute difference frames, threshold frames, and detected frames. The difference frame is done by using the Equation (2) for the background subtraction method. A histogram value between the difference frames and the threshold frame is done by using threshold binary function in the Table 7 in Chapter 2. The computational time for this work is shown in Table 10.

Table 10: The processing time of the system. It was done in between after reading of the video and after showing the results.

Sample	Computational time (sec)
1	16.95
2	11.65
3	41.11
4	54.12

#### 4.1.1 MATLAB Detection and Tracking Result

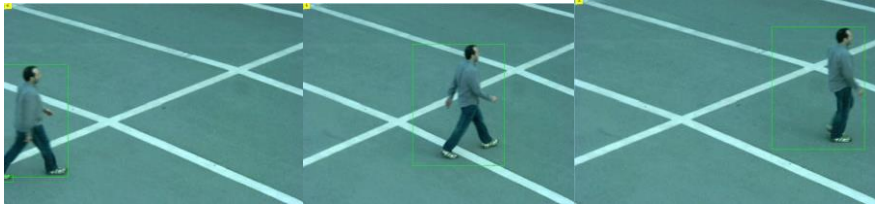


Figure 10: MATLAB result for the sample 1.



Figure 11: MATLAB result for the sample 2.



Figure 12: MATLAB result for the sample 3.



Figure 13: MATLAB result for the sample 4.

### 4.1.2 OpenCV Detection and Tracking Result

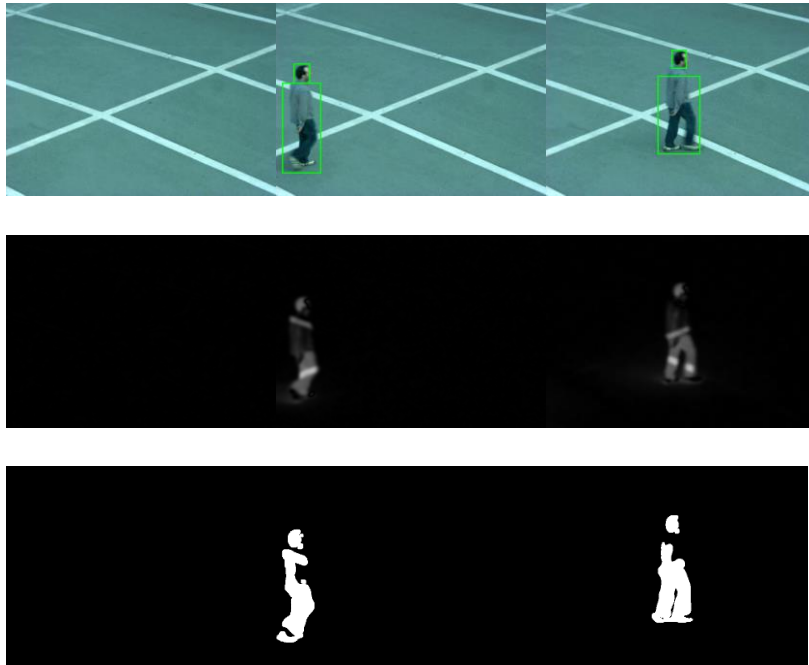


Figure 14: The frames for every process in the detection and tracking algorithm for the sample 1

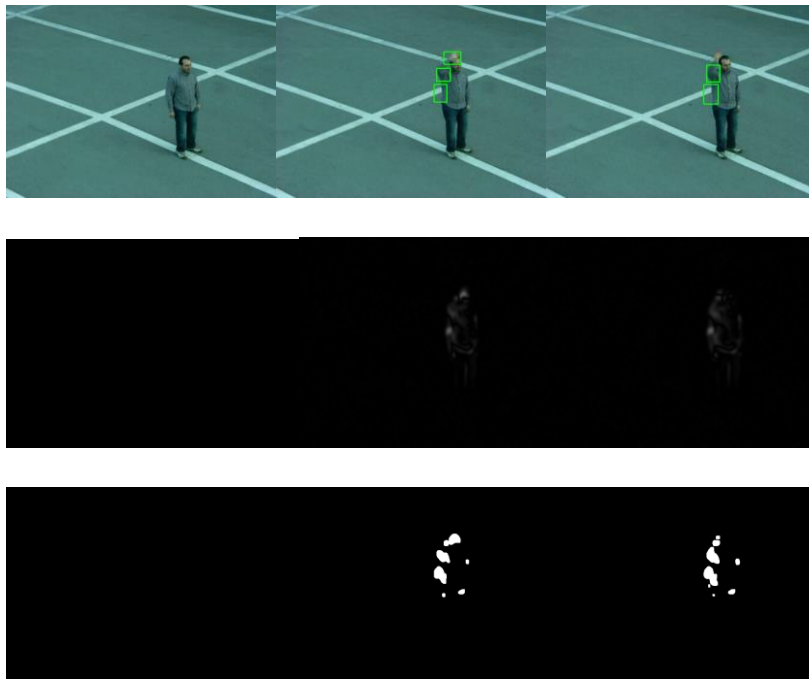


Figure 15: The frames for every process in the detection and tracking algorithm for the sample 2.



Figure 16: The frames for every process in the detection and tracking algorithm for the sample 3.

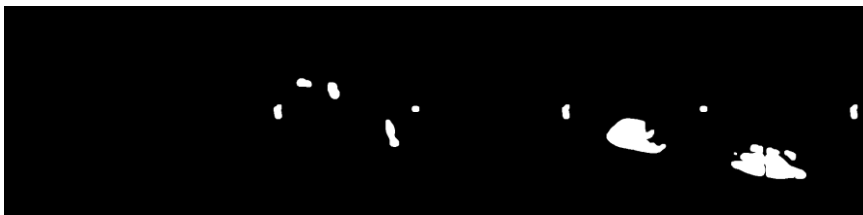
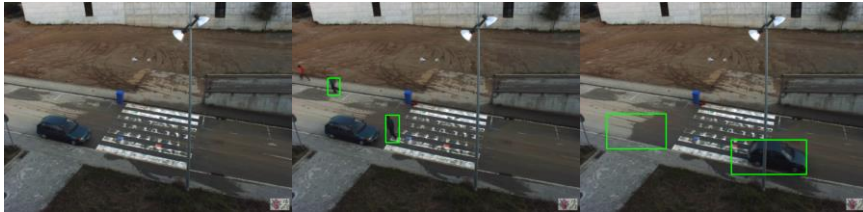


Figure 17: The frames for every process in the detection and tracking algorithm for the sample 4.

## 4.2 Result Analysis

Based on the detection and tracking result from MATLAB and OpenCV, the result from every cases have been compared. MATLAB able to detect the object of interest almost perfectly in sample 1 and sample 2 while in the sample 3 and sample 4, MATLAB likely to produce false positive frame. This caused when there was a high moving object like vehicle in the video frames which can be seen in the Figure 18 and Figure 19.

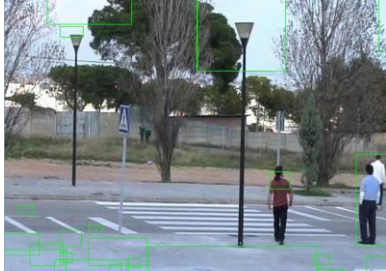


Figure 18: MATLAB false positive in sample 3.



Figure 19: MATLAB false positive in sample 4.

In OpenCV, the result for sample 1, sample 2 and sample 3 have high detection rate even when there was high moving object in the video sequences. However, in the sample 4, the system gave false positive frames when a car that supposedly remain stationary began moving. The system was still prone to error as it cannot update its background model when there was a change in the environment. The region of the stationary car at early frame will be identified as moving object as there were difference in pixels value.

In term of processing time, OpenCV able to produce shorter processing time than MATLAB. This caused from the conversion to grayscale and the resize of frame which helpful in reducing computation time.

### 4.3 Performance Evaluation

The evaluation of the object detection and tracking for every cases is done by frame-based metrics proposed on paper [18]. The evaluation of algorithms will be measured by comparing the individual frames of the system with the ground truth data for video sample. For example, the moving object position and size will tested with the video ground data for that particular frame. Then the whole sequenced will be averaged based on the individual frame data. The description and performance evaluation used in this work shown in the Table 11 and Table 12 respectively.

The detection for the sample is not perfect as shown in Table 13 as there other problem encountered in the system like camouflages, illumination and distance. For example, in the sample 3, the moving pedestrian share similar features with the background image at the end of video resulting in the false negative frames while in the sample 4, the detection and tracking algorithm cannot detect the moving pedestrian when they are away from the camera at certain distance. The problems is expected as the detection and tracking algorithm is not able to tackle these problems.

Table 11: The metrics used in the frame based evaluation [18].

Metrics	Description
True Negative (TN)	Number of frames where the ground truth and the system does not have any moving object.
True Positive (TP)	Number of frames where the ground truth and the system have the moving objects.
False Negative (FN)	Number of frames where the ground truth have at least one moving object while the system does not detect any moving object.
False Positive (FP)	Number of frames where the system have at least one moving object while the ground truth does not having any moving object.
Total Ground Truth (TG)	Total number of frames for the ground truth
Total number of frames (TF)	Total number of frames in the video sequence

Table 12: The following are calculated once all data in the table 11 have been recorded.

Metrics	Description	Equation
False Alarm Rate (FAR)	The number of false positives with respect to the total number of true positives and false positives. The likelihood of the algorithm to detect target correctly reported.	$FAR = \frac{FP}{TP + FP}$
Detection Rate (TP)	The number of true positives with respect to sum of the true positives and false negatives. The percentage of successfully detected targets.	$\text{Detection Rate} = \frac{TP}{TP + FN}$
Specificity	The true false detections relative to the total of false positives and true negatives. The likelihood of a negative response given the total number of actual negative detections.	$\text{Specificity} = \frac{TN}{FP + TN}$
Accuracy	The sum of true positives and true negatives relative to the true false. The measure of the actual performance of the system in detecting and tracking moving objects.	$\text{Accuracy} = \frac{TP + TN}{TF}$
False Negative Rate	The number of false negatives relative to the sum of the true positives and the false negatives. The likelihood of detected targets to be missed.	$\text{False Negative Rate} = \frac{FN}{FN + TP}$
False Positive Rate	The number of false positives relative to the sum of false positives and the false true negatives. The likelihood of the system to reject false positives.	$\text{False Positive Rate} = \frac{FP}{FP + TN}$
Positive Prediction	The number of true positives relative to the sum of true positives and false negatives. The percentage of detected object that are correct.	$\text{Positive Prediction} = \frac{TP}{TP + FN}$
Negative Prediction	The number of true negatives relative to the sum of false negatives and true negatives. The percentage of detected object that are incorrect.	$\text{Negative Prediction} = \frac{TN}{FN + TN}$



Table 13: The evaluation result

Metric	Sample 1	Sample 2	Sample 3	Sample 4
True Negative (TN)	14	75	243	435
True Positive (TP)	68	24	1073	936
False Negative (FN)	0	26	146	290
False Positive (FP)	19	23	124	276
Total Ground Truth (TG)	100	100	1324	1603
Total number of frames (TF)	100	100	1324	1603
FAR	0.21	0.48	0.10	0.22
Detection Rate	1 (100%)	0.48 (48%)	0.89 (89%)	0.76 (76%)
Specificity	0.42	0.76	0.66	0.61
Accuracy	0.82	0.99	0.99	0.85
Positive Prediction Value	0.78	0.51	0.89	0.77
Negative Prediction Value	1	0.74	0.62	0.6
False Negative Rate	0	0.52	0.11	0.23
False Positive Rate	0.57	0.23	0.33	0.38

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

This work presents a theory on current applications and methods on detection and tracking moving object. The important of the object detection and tracking have been explained as they help society in safety system and monitoring system to prevent unwanted accidents and ease human jobs. Besides, the existing methods to detect and track moving object have been compared from the other research paper, journal and book. Lastly, the sample videos was tested and evaluated for the MATLAB and OpenCV and shown in the Chapter 4.

The tested algorithm still far from perfect as their detection rate low when there was changes in the background environment. MATLAB will give false positive frames in sample 4 when there was high moving object in the video while OpenCV give false positive frames when the background model changes. A few suggestion can be implemented in the MATLAB and OpenCV to reduce false detection rate. For MATLAB, other method like optical flow may be used as they are more suitable for detecting high moving object. For OpenCV, the background model should be updated at certain interval time when there was a changes in the environment.

## References

- [1] Ahmed, S.M.A.A. and O.O. Khalifa. *Vision-Based Detection and Tracking of Moving Target in Video Surveillance*. in *Computer and Communication Engineering (ICCCCE), 2014 International Conference on*. 2014.
- [2] Babu, G.S. *Moving Object Detection Using MATLAB*. in *International Journal of Engineering Research and Technology*. 2012. ESRSA Publications.
- [3] Yang, K., Z. Cai, and L. Zhao, *Algorithm Research on Moving Object Detection of Surveillance Video Sequence*. *Optics and Photonics Journal*, 2013. **3**(02): p. 308.
- [4] Kulchandani, J.S. and K.J. Dangarwala. *Moving object detection: Review of recent research trends*. in *Pervasive Computing (ICPC), 2015 International Conference on*. 2015.
- [5] Prajapati, D. and H.J. Galiyawala, *A Review on Moving Object Detection and Tracking*.
- [6] Shaikh, S.H., K. Saeed, and N. Chaki, *Moving Object Detection Approaches, Challenges and Object Tracking*, in *Moving Object Detection Using Background Subtraction*. 2014, Springer. p. 5-14.
- [7] Singh, T., S. Sanju, and B. Vijay, *A new algorithm designing for detection of moving objects in video*. *International Journal of Computer Applications*, 2014. **96**(2).
- [8] Waykole, T.S. and Y.K. Jain, *Detecting and Tracking of Moving Objects from Video*. *International Journal of Computer Applications (0975–8887)*, 2013. **81**(18).
- [9] Kermani, E. and D. Asemani, *A robust adaptive algorithm of moving object detection for video surveillance*. *EURASIP Journal on Image and Video Processing*, 2014. **2014**(1): p. 1-9.
- [10] Patel, S.K. and A. Mishra, *Moving object tracking techniques: A critical review*. *Indian Journal of Computer Science and Engineering*, 2013. **4**(2): p. 95-102.
- [11] Patela, S., U.K. Jaliyab, and M.J. Joshib, *A Review: Object Detection and Object Tracking Methods*.
- [12] Chate, M., S. Amudha, and V. Gohokar, *Object detection and tracking in video sequences*. *ACEEE International Journal on signal & Image processing*, 2012. **3**(1).

- [13] Mohamed, S.S., N.M. Tahir, and R. Adnan. *Background modelling and background subtraction performance for object detection*. in *Signal Processing and Its Applications (CSPA), 2010 6th International Colloquium on*. 2010. IEEE.
- [14] Naeem, H., J. Ahmad, and M. Tayyab. *Real-time object detection and tracking*. in *Multi Topic Conference (INMIC), 2013 16th International*. 2013. IEEE.
- [15] Rout, R.K., *A survey on object detection and tracking algorithms*. 2013.
- [16] Faragher, R., *Understanding the basis of the Kalman filter via a simple and intuitive derivation*. *IEEE Signal processing magazine*, 2012. **29**(5): p. 128-132.
- [17]"Sample Video", *Iselab.cvc.uab.es*, 2017. [Online]. Available: <http://iselab.cvc.uab.es/files/Tools/CvcActionDataSet/index.htm>. [Accessed: 04- Mar- 2017]. [18]
- [18] Bashir, F. and F. Porikli. *Performance evaluation of object detection and tracking systems*. in *Proceedings 9th IEEE International Workshop on PETS*. 2006.
- [19]"OpenCV", *En.wikipedia.org*, 2017. [Online]. Available: <https://en.wikipedia.org/wiki/OpenCV>. [Accessed: 05- Mar- 2017].
- [20] Benezeth, Y., et al. *Review and evaluation of commonly-implemented background subtraction algorithms*. in *2008 19th International Conference on Pattern Recognition*. 2008.
- [21] "Basic Thresholding Operations." [Online]. Available: <http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html>. [Accessed: 09-Apr-2017].
- [22] "Basic Thresholding Operations." [Online]. Available: <http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html>. [Accessed: 09-Apr-2017].
- [23] "Color space," Wikipedia, 29-Mar-2017. [Online]. Available: [https://en.wikipedia.org/wiki/Color\\_space](https://en.wikipedia.org/wiki/Color_space). [Accessed: 09-Apr-2017].

## Appendix 1: OpenCV Detection and Tracking

```
# USAGE

# python motion_detector.py
# python motion_detector.py --video videos/example_01.mp4

# import the necessary packages
import argparse
import datetime
import time
import imutils
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt

count = 1

CV_CAP_PROP_FRAME_COUNT = 0

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

# if the video argument is None, then we are reading from webcam
if args.get("video", None) is None:
    camera = cv2.VideoCapture(0)

    #set fps

# otherwise, we are reading from a video file
else:
    camera = cv2.VideoCapture(args["video"])

    #original video size
```

```

width_ori = camera.get(3)
height_ori = camera.get(4)
ftotal = camera.get(7)
fcount = camera.get(CV_CAP_PROP_FRAME_COUNT)
print('original size= ',width_ori, 'x', height_ori)

#frame rat
fps = camera.get(cv2.CAP_PROP_FPS)

# Display the resulting frame
print ('frame rate = ',fps)
print ('total frame = ',ftotal)

# initialize the first frame in the video stream
firstFrame = None

#start time
e1 = cv2.getTickCount()

# loop over the frames of the video
while True:
    # grab the current frame and initialize the occupied/unoccupied
    # text
    (grabbed, frame) = camera.read()
    text = "Unoccupied"

    print ('frame =',fcount)
    fcount += 1

    # if the frame could not be grabbed, then we have reached the end
    # of the video
    if not grabbed:
        break

```

```

# resize the frame, convert it to grayscale, and blur it
frame = imutils.resize(frame, width=640, height=480)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (21, 21), 0)

# if the first frame is None, initialize it
if firstFrame is None:
    firstFrame = gray
    continue

# compute the absolute difference between the current frame and
# first frame

frameDelta = cv2.absdiff(firstFrame, gray)
thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

# dilate the thresholded image to fill in holes, then find contours
# on thresholded image
thresh = cv2.dilate(thresh, None, iterations=2)
(_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# get centroid
if len(cnts) > 0:
    # find the largest contour in the mask, then use
    # it to compute the minimum enclosing circle and
    # centroid
    s = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(s)
    M = cv2.moments(s)

    cx = int(M["m10"] / M["m00"])
    cy = int(M["m01"] / M["m00"])
    center = (cx, cy)
    print(center)

plt.plot(cx, cy, '.')

```

```

plt.ylabel('y coordinate')
plt.xlabel('x coordinate')
plt.title("the centroid of largest object contour")

# loop over the contours
for c in cnts:
    # if the contour is too small, ignore it
    if cv2.contourArea(c) < args["min_area"]:
        continue

    # compute the bounding box for the contour, draw it on the frame,
    # and update the text
    (x, y, w, h) = cv2.boundingRect(c)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    #text = "Occupied"

# draw the text and timestamp on the frame
#cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
             #cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
#cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
             #(10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

# show the frame and record
cv2.imshow("Security Feed", frame)
cv2.imshow("Thresh", thresh)
cv2.imshow("Frame Delta", frameDelta)

cv2.imwrite('frame%d.png'% count, frame)
cv2.imwrite('thresh%d.png'% count, thresh)
cv2.imwrite('delta%d.png'% count, frameDelta)
count += 1

```



```
#stop time
e2 = cv2.getTickCount()
time = (e2 - e1) / cv2.getTickFrequency()
print (time)

#plt.show()

# cleanup the camera and close any open windows
camera.release()

cv2.destroyAllWindows()
```

## Appendix 2: MATLAB Detection and Tracking

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...  
    'NumTrainingFrames', 50);
```

```
videoReader = vision.VideoFileReader('walk1.mpg');
```

```
for i = 1:150
```

```
    frame = step(videoReader); % read the next video frame
```

```
    foreground = step(foregroundDetector, frame);
```

```
end
```

```
se = strel('square', 3);
```

```
filteredForeground = imopen(foreground, se);
```

```
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
```

```
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
```

```
    'MinimumBlobArea', 150);
```

```
bbox = step(blobAnalysis, filteredForeground);
```

```
result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');
```

```
numCars = size(bbox, 1);
```

```
result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
```

```
    'FontSize', 14);
```

```
videoPlayer = vision.VideoPlayer('Name', 'Detected');
```

```
videoPlayer.Position(3:4) = [650,400]; % window size: [width, height]
```

```
se = strel('square', 3); % morphological filter for noise removal
```

```

while ~isDone(videoReader)

    frame = step(videoReader); % read the next video frame

    % Detect the foreground in the current video frame

    foreground = step(foregroundDetector, frame);

    % Use morphological opening to remove noise in the foreground

    filteredForeground = imopen(foreground, se);

    % Detect the connected components with the specified minimum area, and

    % compute their bounding boxes

    bbox = step(blobAnalysis, filteredForeground);

    % Draw bounding boxes around the detected cars

    result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');

    numCars = size(bbox, 1);

    result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
        'FontSize', 14);

    step(videoPlayer, result); % display the results

end

release(videoReader); % close the video file

```