

MOVING OBJECT DETECTION FOR VISUAL SURVEILLANCE

APPLICATION

BY

KELVIN ONG CHUN YAUW

A REPORT

SUBMITTED TO

University Tunku Abdul Rahman

In partial fulfillment of the requirements

For the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER SCIENCE (CS)

Faculty of Information and Communication Technology (FICT)

(Perak Campus)

May 2018 Trimesters

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

Supervisor's name

Date: _____

Date: _____

MOVING OBJECT DETECTION FOR VISUAL SURVEILLANCE

APPLICATION

BY

KELVIN ONG CHUN YAUW

A REPORT

SUBMITTED TO

University Tunku Abdul Rahman

In partial fulfillment of the requirements

For the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER SCIENCE (CS)

Faculty of Information and Communication Technology (FICT)

(Perak Campus)

May 2018 Trimesters

DECLARATION OF ORIGINALITY

I declare that this report entitled “**MOVING OBJECT DETECTION FOR VISUAL SURVEILLANCE APPLICATION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this challenging project. A special gratitude I would like to give to my supervisor, Prof. Zen Chen, who provided me thought –provoking feedback and encouragement to engage in this Moving Object Detection for Visual Surveillance Application this project. It is my first step to establish a career in visual surveillance application system development. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

Motion detection is one of the most active research areas in computer vision application. Motion detection can be informative and valuable, and motion information is very useful for visual surveillance system. However motion detection analysis is still an open problem in computer vision owing to the dynamic background, shadow, illumination changes, and camera jitter under the uncontrolled environment. This study focuses on detecting scene changes in critical environments effectively and efficiently. In order to tackle the stated problems, an improved approach is proposed to deal with the critical environment problems in vision based motion detection. The goals to be achieved include (a) the background model is represented by codewords derived from the dominant quantized colors. (b) an image preprocessing technique is employed in order to solve the lighting change effect on the motion detection result by converting the RGB color space to the LAB color space. (c) an image post-process technique is performed to enhance the foreground mask result by filling up the broken moving objects for which the morphological operation finds hard to cope.

Table of Contents

TITLE PAGE	i
DECLARATION OF ORIGINALITY	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	vi
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	1-2
1.2 Background and Motivation	3-5
1.3 Objectives	6
1.4 Proposed Approach / Study	7
1.5 Highlight of The Contribution	7
CHAPTER 2 LITERATURE REVIEW	8
2.1 Object Detection Classification	8
2.2 Object Tracking Classification	8
2.3 Background Subtraction Technique	9
2.3.1 Statistical Model	9-10
2.3.1.1 Single Gaussian, SG	10-11
2.3.1.2 Mixture of Gaussian, MOG	11-14
2.3.1.3 Kernel Density Estimation, KDE	14-15
2.3.2 Cluster Models	15
2.3.2.1 Codebook Model	15-20
CHAPTER 3 SYSTEM DESIGN	
3.1 Introduction	21
3.2 Key Concept	22

3.2.1	Background Modelling by Codebook Approach Based on Multiple Code Words and Code Word Sharing	22-24
3.2.2	Lighting Effect on Color Intensities	25-26
3.3	System Flow Chart	27
3.4	System Pseudocode	28-29
CHAPTER 4	SYSTEM IMPLEMENTATION AND VERIFICATION	30
4.1	System Code Description	30-42
4.2	System Code Testing and Verification	43
4.2.1	Testing Core Function on Video Clip	43-44
4.2.2	Debugging tool for image colors conversion and display.	45
4.2.3	Testing Result on Background Modelling by Code book Approach Based on Multiple Code Words and Code Word Sharing	46-48
4.2.4	Testing Result On Lighting Effect of Color Intensities	49-50
4.2.5	Testing Result on Enhancement of Foreground Mask Image	51
CHAPTER 5	EXPERIMENTAL RESULTS AND ANALYSIS	
5.1	Introduction	52-53
5.2	Experimental results on CDNET 2014 Dynamic Background Data Set	54-55
5.3	Experimental results on CDNET 2014 Dynamic Baseline Data Set	56
CHAPTER 6	CONCLUSION	57
5.1	Highlight on any novelties and contributions the project has achieved	57
5.2	Summary of the research problems that have been considered and solved	58
5.3	Future Work	58
REFERENCES AND BIBLIOGRAPHY		59-60
APPENDIX		

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.0	Example of scene change detection under various conditions.	2
Figure 1.1	General visual surveillance system	3
Figure 1.2	General visual surveillance system for multiple cameras	4
Figure 2.0	Background subtraction example for two sequence frame in an 'office'.	9
Figure 2.1	Human body in 2-D model	10
Figure 2.2	Pixel intensity distribution example	11
Figure 2.3	Some example illustrates the pixel process of some of difficulties involved in real environments.	12
Figure 2.4	Proposed codebook color model	15
Figure 3.0	Structure of Code Book Model	22
Figure 3.1	Lighting Effect on Color Intensities	25
Figure 3.2	System Flow Chart Diagram	27
Figure 4.0	ROI Selection	43
Figure 4.1	Performing Background Modelling	43
Figure 4.2	Foreground Detection	44
Figure 4.3	Debugging Tool for image color conversion and display	45
Figure 4.4	Output result by code word approach (set1)	46
Figure 4.5	Output result by code word approach (set2)	47
Figure 4.6	Output result by code word approach (set3)	48
Figure 4.7	Output result on lighting effect of color intensities (set1)	49
Figure 4.8	Output result on lighting effect of color intensities (set2)	50
Figure 4.9	Output result on Enhancement of Foreground Mask Image	51
Figure 5.0	Experimental results on CDNET 2014 Dynamic Background Data Set	54-55
Figure 5.1	Experimental results on CDNET 2014 Baseline Data Set	56

LIST OF ABBREVIATION

CCTV	Closed Circuit Television
PTZ	Pan-Tilt-Zoom
SG	Single Gaussian
SGG	Single General Gaussian
MOG	Mixture of Gaussian
GMM	Gaussian Mixture Model
KDE	Kernel Density Estimation
OpenCV	Open Source Computer Vision

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

The ability of detecting moving objects or object in motion has become an essential necessity for the majority of computer vision application. However, develops a reliable application to implement moving object detection is still a challenge until today as motion detection can be quite tricky. Let put under consideration that the object detected will be moving at any time and any speed. They might be moving very fast and suddenly stay stationary, the detection have to follow the object. The detection need to as fast as possible so that the object is detected immediately when they comes into the detection area. It is difficult to detect an object when it is in a busy environment where there are a lot of foreground objects. Moreover, when the objects have similar properties with the background it makes it harder for detection. For human, human are able easily determine and obtain several of the information out, for example, foreground and background, motions and events. But computer is not like human, it does not have self-learning ability, therefore they are not able to detect and track the motion as “smart” as human.

Motion detection is considered a challenge task in computer vision field. *Toyama et al.(1999) and Bouwmans. (2014)* listed that there are several type of challenges for computer vision field and only few commonly challenges will introduced.

- **Dynamic background:** generally non-static and should be regarded as background based on their relevance. Several dynamic factors such as waving tree, water rippling and so on may produce false positive as the respective background pixel that move may move different position as compared to previous frames.
- **Gradual and sudden illumination change:** Gradual illumination change are experienced along sunlight changes as time elapses or the sun being covered by clouds in outdoor environment, whereas sudden illumination change experienced along light can sometimes be switched on or off in indoor environment. Different lighting condition placed in scene might affect the visibility of object or alter the object appearance that causes the object look different.

CHAPTER 1 INTRODUCTION

- **Sleeping object:** where a background made change by a moved object, for example if someone parks car and not moved for a long period time, the car should classified as part of background.
- **Cast shadow:** the shadow of the moving object that classified wrongly as part of foreground pixel that should eliminate.
- **Camera jitters:** caused by sudden background movement.

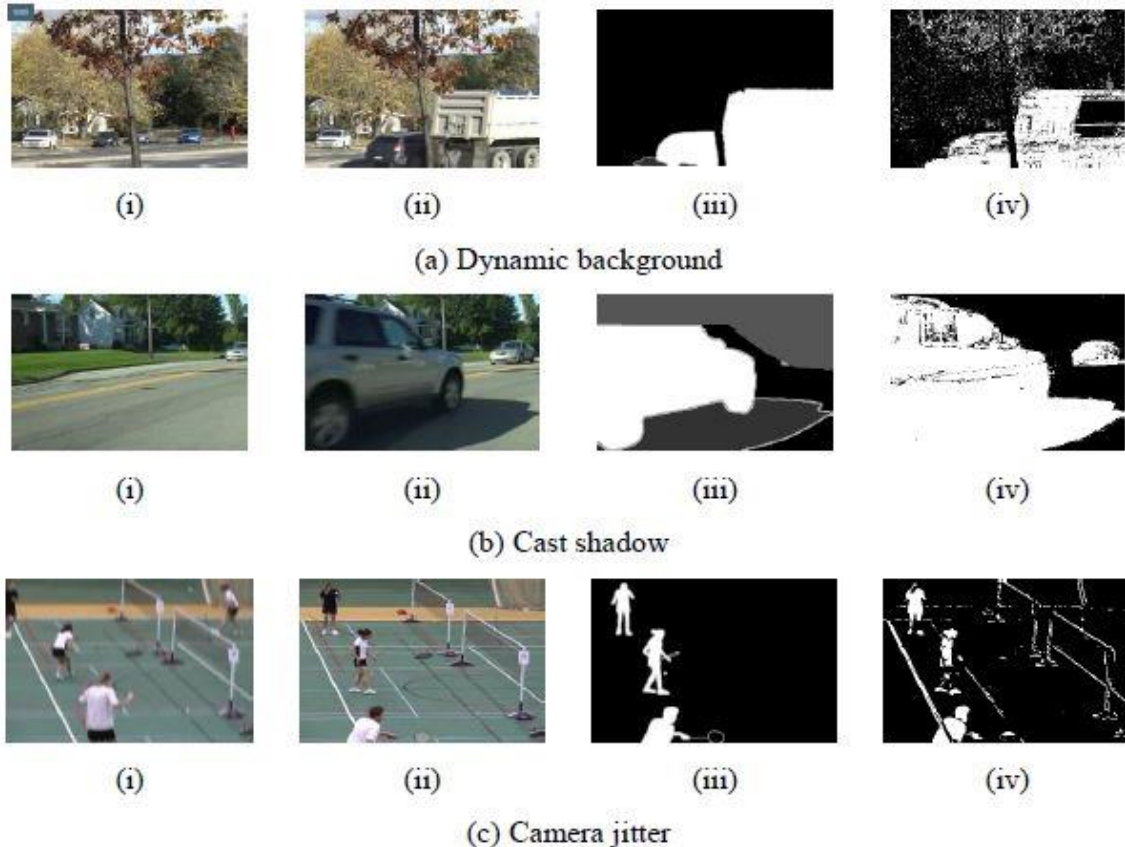


Figure 1.0: Example of scene change detection under various conditions.

(i) Background model, (ii) Input frame, (iii) Ground truth, (iv) False detection

1.2 BACKGROUND AND MOTIVATION

Computer or machine vision, digital image processing and pattern recognition are a field in artificial intelligence. It has become an extensive area of research on the images automatic analysis and image sequences due to its wide range of computer application such as intelligent visual surveillance, human-computer interaction, or video compression. Benefitting from the advances in those fields, various methods were developed to allow a machine or computer to implement such tasks like moving object detection and tracking, pattern recognition and other. One of the applications is the robotized video-based reconnaissance or commonly known as automated video- based surveillance system. In this study, moving object detection for visual surveillance application will be focused as our topic of interest.

Visual surveillance system is a system that monitors the behavior, activities, actions, and changing environment. In surveillance system, there are some common step, for instance, object detection, object association, or refer as tracking, and scene event understanding. The building block of visual surveillance system is indicated in **Figure 1.1** and will briefly introduce in Chapter 2 later.

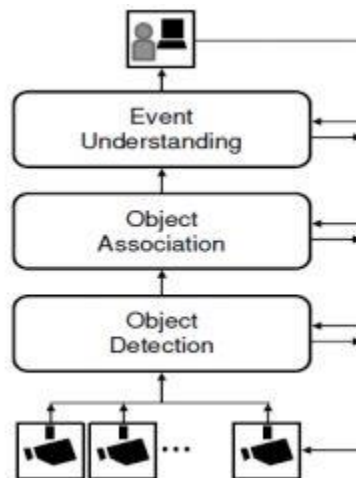


Figure 1.1: General visual surveillance system

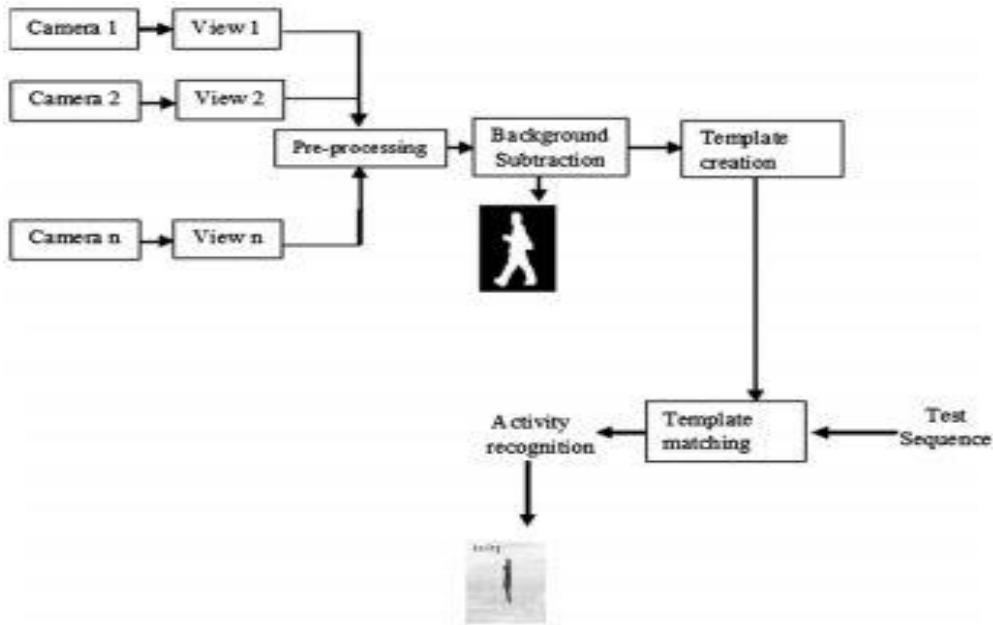


Figure 1.2: General visual surveillance system for multiple cameras

Motion detection has important roles in video surveillance system. It is become an own research topic that drawn much attention in last decade because of its raising threat in security, for example, assaults, intrusion, robberies or burglary in public area places such as indoor-outdoor, railway/bus station or airport, to mention only few of them. (Yadav and Singh, 2015).

Video surveillance system is aim to help human administrator monitor CCTV camera system and alerts them to deviate from normal behavior in the surveillance region without manual assistance. It bring the fundamental advantages, which an administrator able to monitor a huge number of camera with focusing his attentiveness to the critical place and times, due to the monitoring region where non-intriguing occasion are going on can assume a tedious task. Moreover, the information obtained through automatic video analysis method can utilized to help video administrator as well as legal authorities with retrieve the evidence proof from recorded video data, to control huge region video network in assignment and even for less technical and or specialized problem as assisting to protect the individual privacy in public place.

Nowadays, the usage of video surveillance system is not restrained for safety and security applications purpose. Video surveillance system can also being developed to analyzing the demographic distribution of a crowd, to provide advertising assessment and quality of service at

CHAPTER 1 INTRODUCTION

department store, traffic monitoring on highways purposes or even on houses area for intrusion prevention. Thus these phenomena results in increasing of number use of video surveillance system.

The purpose of this project is focus on the detection of objects in unrestricted environments monitored with the static video camera. It is to identify for each frame the set of pixels that are completely different from the previous frames. In the video surveillance domain, motion detection has been frequently used in order to segment foreground objects from the background. Foreground objects are the objects of interest in an automated surveillance system. The segmented foreground objects are then associated between frames in order to perform a scene analysis and detect events of interest. However, there are some backgrounds characteristics as dynamic or sudden illumination changes, which might make difficult the task of background modelling due to dynamic background will be recognized as foreground if there is insufficient of training process. Thus, there is a need to modify existing method to solving these problems from background because it is not meaningful to detect the background although there is have motion but it is not in the area of interest.

1.3 OBJECTIVES

The objectives of this project are:

- To detect and track the moving object
 - The system shall able to detect and sense any motions with focusing on object of interest only.

- To introduce an efficient algorithm for moving object detection in the surveillance system implementation
 - The system aim to allow user have a better understanding how proposed detection system can be applied

- To investigate the proposed methods able to overcome the challenges related to background modeling
 - The system shall able to perform in better way over the existing system to detecting scene changes in critical environments such as dynamic backgrounds, changes and moving objects.

1.4 PROPOSED APPROACH / STUDY

In this project, the moving object detection in input video is done by background subtraction method with using multiple dominant colors in codebook algorithm concept. By applying technique of background subtraction, the system shall first train with the video frames to model a background model by extracting at least one dominant colors with a given frequency coverage threshold. In the training phase, for each pixel from each training frame, the system reads its color and increments on its specific quantized code word. After finished reading all frames, for each pixel, the system sorts the codebooks and extracts the first number of most dominant code words to be as the background model of that particular pixel.

In the testing phase, the system tests each original video frame with the background model to distinguishing foreground and background pixels. The system should be able to generate foreground detection result for each video frame by comparing color difference of each pixel and each dominant code word in the background model.

1.5 HIGHLIGHT OF THE CONTRIBUTION

In this project, an improved approach is proposed to deal with the critical environment problems in vision based motion detection. The achievements include:

- 1) The background model is represented by codewords derived from the dominant quantized colors.
- 2) An image preprocessing technique is employed in order to solve the lighting change effect on the motion detection result by converting the RGB color space to the LAB color space.
- 3) An image post-process technique is performed to enhance the foreground mask result by filling up the broken moving objects for which the morphological operation finds hard to cope.

CHAPTER 2 LITERATURE REVIEW

2.1 Object Detection Classification [Szeliski, 2010]

Generic object recognition, or may refer as category-level object recognition, is considered to be one challenges in visual task in computer vision field [Szeliski, 2010]. For an instance of a particular general class, like, 'human', 'bird', or 'car', the task purpose to classify and initialize it through visual feature correctly. Research on detect object models and image location can be time-consuming for many computer vision applications. In order to reduce this complexity problem, surveillance system typically distributed the problem into two steps: first, the interest object is recognized, and another, the object that wanted to detect is initialized. Interest object are basically refer to those object presenting something changes in the observed scene or usually associated to moving objects.

2.2 Object Tracking Classification [Broida and Chellappa, 1986] [Tanizaki, 1987] [Yilmaz et al, 2006]

Object tracking is the activities of build parallelism among the objects that detected throughout the frames of a video sequences. Utilizing an object model and the motion they exhibit is needed in order to perform this task. Object models typically are the points, primitive geometric shape like ellipses, rectangle, articulated shape models as well as skeletons. The model of motion able to delimit based on the selected object mode utilized. For instance, a translational model can be used if an object is represented by point. Based on the domain application, assumption is made to constrain the tracking issue. As example, point-based tracking model is the popular choice to solve the tracking issue in the visual surveillance system. Therefore, Kalman method proposed by [Broida and Chellappa, 1986] and *Particle Filters* method proposed by [Tanizaki, 1987] are generally state estimation methods that used for computing the cost of given object association. A better understanding on the topic of tracking including appropriate use of image feature, motion model selection, and object detection can be found in [Yilmaz et al, 2006].

2.3 Background Subtraction Technique [Toyama et.al 1999] [Bouswans 2014]

Generally, motion detection can be conducted in many techniques. Background subtraction is the most frequent used application for motion detection in image processing. Therefore, background subtraction technique will be our topic of interest in this research.

Background subtraction is most commonly and widely used approach as a preprocessing step in setup with static cameras. This method consists in using a scene background model to detect foreground object through separating incoming frames. It attempt to detect the moved object through subtracting the current image with every pixel from a reference background image and compared it step-by-step with predefined object dataset. The basic of such way needed is to separating the moving object known as “foreground” from the static information known as “background”. This method is mostly fast and has low demanding in computational term.

Figure 2.0 indicates a scene where a person enters in an office and then consults a book. The empty office has been applied as background exemplary depiction whereas foreground mask has taken from the manually generated ground-truth.



Figure 2.0: Background subtraction example for two sequence frame in an ‘office’.
Left: frame number, Middle: background scene, Right: ground truth foreground mask

However, this method can be sensitive to some uncontrolled environment conditions as well as sudden illumination change, dynamic background and so on. An exhaustive introduction on background subtraction method with main issue that this application has to deal with, can be found in [Toyama et.al 1999] and [Bouswans 2014]. A few state-of-the-art background subtraction approaches overview and their respective performance can be found later on in thesis. (See2.3.1)

2.3.1 Statistical Models

Statistical models capture the distribution of pixel variations as a mean to build the background model. This model provides adaptive solutions to solve problem such as illumination changes and dynamic background. Several well-known statistical models are single Gaussian, mixture of Gaussian and Kernel Density Estimation will discuss in next.

2.3.1.1 Single Gaussian, SG [Wren et al 1997] [Elgammal et al. 2000] [Kim et al 2007]

At early years, [Wren et al. 1997] introduced a background subtraction methods named “Pfinder” (person finder) with adopt a Maximum a Posteriori Probability (MAP) method to detect and track body of human in 2-D model as shown in **Figure 2.1**.



Figure 2.1: (left) video input (color image), center: segmentation, right: a 2-D blob statistic model represents human body

[Wren et al 1997] modelled this 2-D region into low order statistic and each cluster of 2-D points has a mean and covariance matrix that shall denote as μ and K . The blob spatial statistics is interpreted as Gaussian model as shown in equation (2.0).

$$\Pr(O) = \frac{\exp\left[-\frac{1}{2}(O - \mu)^T K^{-1}(O - \mu)\right]}{(2\pi)^{\frac{m}{2}} |K|^{\frac{1}{2}}} \quad (2.0)$$

where O is a particular pixel intensity and m is the color dimension.

Furthermore, this is supported by [Elgammal et al 2000] where they stated that pixel intensities can be reasonably modelled with a Gaussian distribution when a series of frames are monitored in a static background. **Figure 2.2** depicts the sample pixel intensity histogram across a period time that can be approximated with a single Gaussian distribution.

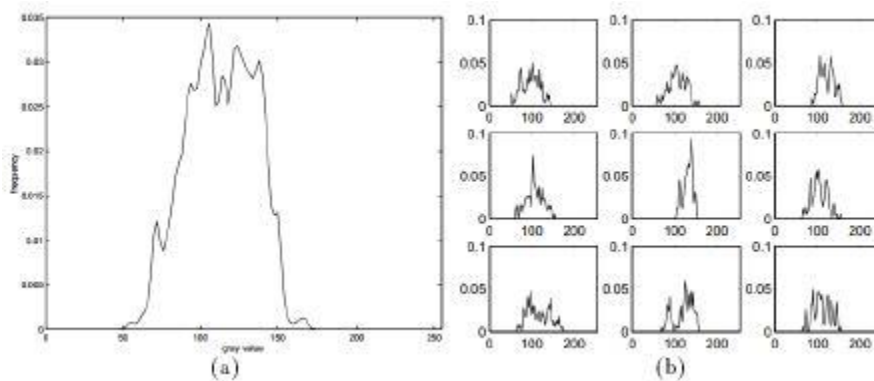


Figure 2.2: pixel intensity distribution example

Several of background subtraction methods are extensions of the Gaussian distribution model, for example, Single General Gaussian (SGG) model proposed by [Kim et al 2007] and Mixture of Gaussian (MOG) introduced by [Stauffer and Grimson.1999]. Nevertheless, these methods only able to capture the scene change when background is static or less dynamic compared to the foreground objects.

2.3.1.2 Mixture of Gaussian, MOG [Stauffer and Grimson 1999] [Zivkovic and van der Heijden 2006] [Shimada et al 2006] [Tan et al 2006]

[Stauffer and Grimson 1999] proposed method of Mixture of Gaussian (MOG) or may refer as Gaussian Mixture Model (GMM) that uses multi modeling to model a background pixel. Each Gaussian distribution in the MOG has its variance and this value is used to identify whether the Gaussian represent background or foreground.

A pixel process has been defined as a history of the intensity values of a pixel from frame 1 to frame t as represented by equation (2.1) and some examples of pixel process are shown in **Figure 2.3**.

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \quad (2.1)$$

where I is image sequences

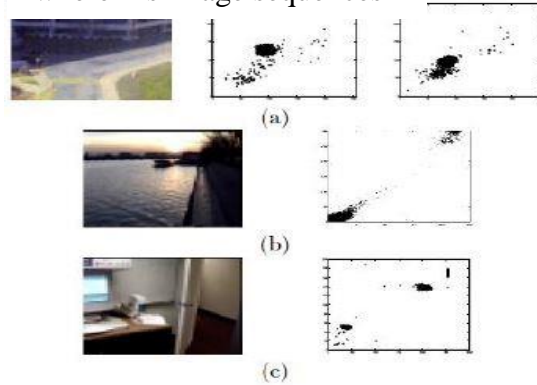


Figure 2.3: Some example illustrates the pixel process of some of difficulties involved in real environments.

The pixel process illustrations depicts that an adaptive system with automatic thresholds and a multi-modal representation are needed to handle difficulties involved in real environments. Thus, [Stauffer and Grimson 1999] modelled the pixel process into K Gaussian distribution and the probability of current pixel value can be calculated as:

$$P(X_t) = \sum_{i=1}^K w_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \quad (2.2)$$

where

- a) K is the total number of Gaussian distribution, usually set to 3 to 5
- b) $w_{i,t}$ is the weight estimation for the i th Gaussian in the mixture at time t
- c) $\mu_{i,t}$ is a mean value of the i th Gaussian in the mixture at time t
- d) $\Sigma_{i,t}$ is the covariance matrix of the i th Gaussian in the mixture at time t

Assuming the red, green and blue pixel values are independent and with same variances, the covariance matrix can be calculated using:

$$\Sigma_{i,t} = \sigma_i^2 I \quad (2.3)$$

- e) η is a probability density function (pdf) for Gaussian:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu)\Sigma^{-1}(X_t - \mu)} \quad (2.4)$$

CHAPTER 2 LITERATURE REVIEW

[Stauffer and Grimson 1999] used the expectation maximization (EM) algorithm to initialize the weight, mean and covariance matrix. When there is a new testing frame, a matching test will be made on every pixel and produce two type of result:

- a) Case 1: The input pixel matches to any K Gaussians. If the respective Gaussian is identified as a background, then the input pixel will be classified as background pixel.
- b) Case 2: The input did not match to any K Gaussians. The input pixel will be classified as foreground.

When the input pixel is classified as foreground, the least probable distribution is replaced with the current input pixel distribution as its mean value, an initial high variance, and low prior weight. Finally, the K distributions at time t will be calculated as below:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \quad (2.5)$$

where α : learning rate

M_k , is 1 for the model which matched and 0 for the remaining models

However, using fixed number of Gaussian for each of the pixel is not optimal in terms of detection and computation. [Zivkovic and van der Heijden 2006] introduced a method to improve the Gaussian Mixture Model by adapting the number of Gaussian for every individual pixel using Dirichlet distance. This method uses an exponential decaying constant α to update the background model recursively as shown below:

$$\omega_{i,t} = \omega_{i,t} + \alpha(o_i^t - \omega_{i,t}) \quad (2.6)$$

$$\mu_{i,t} = \mu_{i,t} + o_i^t \left(\frac{\alpha}{\omega_{i,t}} \right) \delta_m \quad (2.7)$$

$$\Sigma_{i,t} = \Sigma_{i,t} + o_i^t \left(\frac{\alpha}{\omega_{i,t}} \right) (\delta_m^T \delta_m - \Sigma_{i,t}^2)$$

$$\text{where } \delta_m = (X_t - \mu_{i,t}) \quad (2.8)$$

Furthermore, [Shimada et al 2006] introduced another background estimation method to control the Gaussians number dynamically. This method uses a threshold to integrate or delete the Gaussians. If the weight of the least probable distribution is less than the threshold, the respective Gaussian will be deleted and the remaining Gaussians will renormalize their weights. Otherwise, if the differences between the means of two Gaussians are smaller than a threshold, the two Gaussians will be integrated into one Gaussian. Moreover, [Tan et al 2006] proposed a modified online EM procedure to construct an adaptive - K Gaussian Mixture Model (AKGMM). This model adjusts the parameters at each pixel adaptively based on the complexity of the patterns using with a modified online EM procedure.

2.3.1.3 Kernel Density Estimation, KDE [Elgammal et al 2002]

[Elgammal et al 2002] define the probability density function (pdf) from a set of pixel data without any assumption on the underlying distributions with introduced a general non-parametric kernel density estimation technique.. The calculation of probability density function of a pixel can be estimated using a set of sample, $S=\{x_i\}_{i=1,2,\dots,N}$ with following on the equation

$$\Pr(x_t) = \frac{1}{N} \sum_{i=1}^N \prod_{j=1}^d K_{\sigma_j}(x_{tj} - x_{ij}) \quad (2.9)$$

where K_{σ} is a kernel function with bandwidth σ ,
 x_t is a d-dimensional color feature.

When choosing the Gaussian as the kernel, the estimated probability becomes:

$$\Pr(x_t) = \frac{1}{N} \sum_{i=1}^N \prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\left(\frac{(x_{tj}-x_{ij})^2}{\sigma_j^2}\right)} \quad (2.10)$$

Finally, by referring to the estimated probability, the pixel is classified as foreground pixel if $\Pr(x_t) < th$ where th is a global threshold for the whole image.

In term of practice, with go through to lookup tables, the computation cost of the probability estimation stated above can be reduced. However, the complexity of this method is still (N^2), so that this has become the bottleneck of the method. [Ianasi et al 2005] proposed a better

background tracking technique that grouping the strengths of parametric, namely, non-parametric and histogram based density estimation method in order to reduce the calculation of number of samples thus improve the estimation performance.

2.3.2 Cluster Models

Cluster models group the intensities of each pixel in the frame into clusters. Representative cluster models include K-mean algorithms and codebook (Kim et al., 2004). In the following, the codebook model is discussed in details.

2.3.2.1 Codebook Model [Kim et al. (2004)] [Ilyas et al 2009]

[Kim et al. (2004)] proposed a background clustering method to quantize each of the sample background pixel's value into codebook as a background model. The background pixel variation is able to be captured through a long period time by using quantization. The quantization is based on a color distortion metric corresponding with brightness bounds as shown in **Figure 2.4**.

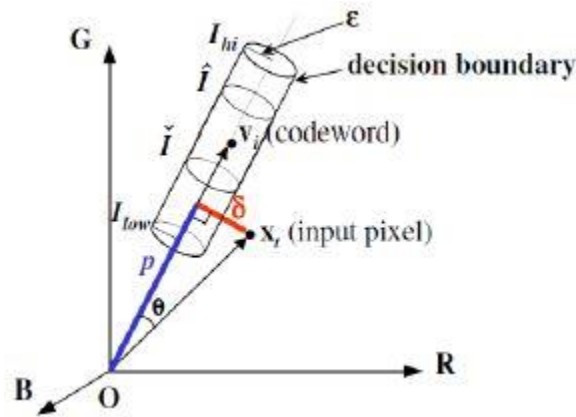


Figure 2.4: Proposed codebook color model - a color distortion and brightness distortion separation evaluation

Assuming there are C codebooks and each of it consists L codewords to represent different background model cluster for a particular pixel. Each codeword $c_i, i=1 \dots L$ consist the following items:

- a) RGB vector, $v_i = (R_i, G_i, B_i)$
- b) 6-tuple, aux_i , where aux_i contains the following variables:

CHAPTER 2 LITERATURE REVIEW

- \check{I}, \hat{I} : The minimum and maximum brightness of every pixel assigned to codeword respectively
- f : The frequency of the codeword occurrences
- λ : The longest interval during the training process that the codeword has not recurred and is define as maximum negative run-length (MNRL)
- p, q : The first and last access time that the codeword has occurred respectively

Since the variation of pixel value is mainly related to brightness, therefore it will lie along the principle axis of the codeword with low and high bound of brightness. Besides, the color distortion is interpreted as a brightness-weighted version in the normalized color space. For example, an input pixel, $x_t = (R, G, B)$ and codeword, c_i where $v_i = (\bar{R}, \bar{G}, \bar{B})$, the color distortion δ can be calculated using:

$$\|x_t\|^2 = R^2 + G^2 + B^2 \quad (2.11)$$

$$\|v_i\|^2 = \bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2 \quad (2.12)$$

$$\langle x_t, v_i \rangle^2 = (\bar{R}_i \cdot R + \bar{G}_i \cdot G + \bar{B}_i \cdot B)^2 \quad (2.13)$$

$$p^2 = \|x_t\|^2 \cos^2 \theta = \frac{\langle x_t, v_i \rangle^2}{\|v_i\|^2} \quad (2.14)$$

$$\text{Colour distortion, } \delta = \sqrt{\|x_t\|^2 - p^2} \quad (2.15)$$

Furthermore, the minimum and maximum brightness \check{I}, \hat{I} will be assigned to each of the codeword and a range of brightness intensity will be taken into account that limits the shadow level and highlight level. It will be calculated based on the minimum and maximum brightness as shown below:

$$I_{low} = \alpha \hat{I}, I_{hi} = \min \left\{ \beta \hat{I} \frac{\bar{I}}{\alpha} \right\}, \text{ where } \alpha < 1 \text{ and } \beta > 1$$

Normally, α is between 0.4 – 0.7, and β is between 1.1 – 1.5. (2.16)

Comparing with other methods, the codebook method only needs to find one of the codewords to detect the foreground. Foreground will be detected by matching the input image pixel with its respective codebook based on two conditions:

1. Color distortion, $\delta \leq \varepsilon^2$ where ε^2 is the detection threshold
2. Brightness = $\begin{cases} \text{True} & I_{low} \leq I \leq I_{hi} \\ \text{False} & \text{otherwise} \end{cases}$

If there is a match, the matched codeword will be updated by setting:

$$v_i = \left(\frac{f_i \bar{R}_i + R}{f_i + 1}, \frac{f_i \bar{G}_i + G}{f_i + 1}, \frac{f_i \bar{B}_i + B}{f_i + 1} \right)$$

$$aux_i = \langle \min\{I, \bar{I}\}, \max\{I, \hat{I}\}, f_i + 1, \max\{\lambda_i, t - q_i\}, p_i, t \rangle$$

(2.17)

Otherwise, a new codeword will be input into the respective codebook by setting:

$$v_L = (R, G, B)$$

$$aux_L = \langle I, I, 1, t - 1, t, t \rangle$$

(2.18)

After the update, a sorting will be done on the codewords to relocate the most recently updated codeword to the front in order to speed up the algorithm.

Furthermore, the minimum and maximum brightness \hat{I} , \bar{I} will be assigned to each of the codeword and a range of brightness intensity will be taken into account that limits the shadow level and highlight level. It will be calculated based on the minimum and maximum brightness as shown below:

$$I_{low} = \alpha \hat{I}, I_{hi} = \min \left\{ \beta \hat{I} \frac{\bar{I}}{\alpha} \right\}, \text{ where } \alpha < 1 \text{ and } \beta > 1$$

Normally, α is between 0.4 – 0.7, and β is between 1.1 – 1.5. (2.16)

Comparing with other methods, the codebook method only needs to find one of the codewords to detect the foreground. Foreground will be detected by matching the input image pixel with its respective codebook based on two conditions:

1. Color distortion, $\delta \leq \varepsilon^2$ where ε^2 is the detection threshold

$$\begin{cases} \text{True} & I_{low} \leq I \leq I_{hi} \\ \text{False} & \text{otherwise} \end{cases}$$

2. Brightness =

If there is a match, the matched codeword will be updated by setting:

$$v_i = \left(\frac{f_i \bar{R}_i + R}{f_i + 1}, \frac{f_i \bar{G}_i + G}{f_i + 1}, \frac{f_i \bar{B}_i + B}{f_i + 1} \right) \quad (2.17)$$

$$aux_i = \langle \min\{I, \bar{I}_i\}, \max\{I, \hat{I}_i\}, f_i + 1, \max\{\lambda_i, t - q_i\}, p_i, t \rangle$$

Otherwise, a new codeword will be input into the respective codebook by setting:

$$\begin{aligned} v_L &= (R, G, B) \\ aux_i &= \langle I, I, 1, t - 1, t, t \rangle \end{aligned} \quad (2.18)$$

After the update, a sorting will be done on the codewords to relocate the most recently updated codeword to the front in order to speed up the algorithm.

Moreover, codebook model is able to handle dynamic backgrounds, illumination changes and allow moving foreground objects in the training process. During the training process, a maximum negative run-length (MNRL) will be recorded for each of the codeword in order to remove those false codewords from the codebook later. A threshold $T_{\mathcal{M}}$ will be used for temporal filtering to create a new codebook \mathcal{M} without false background model as:

$$\mathcal{M} = \{c_m | c_m \in \mathcal{C} \wedge \lambda_m \leq T_{\mathcal{M}}\} \quad (2.19)$$

Usually, the threshold $T_{\mathcal{M}}$ is set to half of the number of training frames. As a result, although some codewords having a large λ , but it will still be removed from the codebook as it appears for a very short period.

[Kim et al 2005] also introduced two enhancements to the codebook model which are layered modelling and detection, and adaptive codebook updating. The layered modelling and detection is able to solve the wake up object problems. It is able to detect the foreground objects although a new background was obtained during the detection phase. During detection process, if the incoming pixel doesn't match with any of the codewords from the respective codebook, it will create a new codeword and add it to an additional model named cache. If the cache codewords stay longer than a threshold T_{add} , the cache codewords will be moved into the background model. Meanwhile, if the cache codewords did not access for a period of time and the time are longer than a threshold T_{delete} , it will be removed from the cache.

Another enhancement is to apply a learning rate to the update process in order to solve the illumination changes problem. The updating formula for the existed codeword will be replaced as:

$$v_i = \gamma(R, G, B) + (1 - \gamma)v_i \quad (2.20)$$

$$\sigma_i^2 = \rho\delta^2 + 1(-\rho)\sigma_i^2 \quad (2.21)$$

where γ and ρ are learning rates
 σ_i^2 is the overall variance of color distortion in the color model and it will be initialized when the algorithm starts.

Lastly, the color distortion formula will also be changed to:

$$\text{Colour distortion, } \delta = \frac{\sqrt{\|x_t\|^2 - p^2}}{\sigma_i} \quad (2.22)$$

CHAPTER 2 LITERATURE REVIEW

[Ilyas et al 2009] introduced to add a new parameter f into the codebook algorithm as a criterion for accessing, deleting, matching and adding a new codeword into the codebook. A frequency parameter f_c is also added for the cache codebook accessing. The modifications are as follows:

- 1) A new codebook \mathcal{M} without false background model as:

$$\mathcal{M} = \{c_m | c_m \in C \wedge (\lambda_m \leq T_{\mathcal{M}}) \wedge (f \geq f_{ref})\} \quad (2.23)$$

If the cache codewords stayed longer than a threshold T_{add} and the frequency of cache codewords $f_c \geq f_{re}$, then the cache codewords will be moved into the background model.

CHAPTER 3 SYSTEM DESIGN

3.1 INTRODUCTION

Our project topic title is Moving Object Detection for Visual Surveillance Application. We need an efficient algorithm for computer-vision program to track moving object in a video. Although many algorithm and method has been proposed, most of them have their shortcoming in terms of their accuracy, performance, memory requirement. In this project, the background subtraction method by applied multiple dominant color with codebook construction is chosen.

Normally, background subtraction method is used in static camera; hence, the background will be static (not moving). Background subtraction involved in building a background model. Then we can determine for each pixel, it is a background pixel or a foreground pixel. The entire foreground pixel can be classified as a “moving” object (because the background is static, hence the foreground can be said as “moving”). Hence, we are able to detect motion through this method.

In our implementation, we assign a (or a set of) dominant color to each pixel location. In training phase, we learn the background through accumulation of quantized color bin value across each frame of the training video for each pixel location. After training phase, the background model can be from using the color that occur the most across all the frame of training video for each pixel location.

At testing phase, for each pixel location, we obtain an absolute different of color value between the background model and video frame. If the different exceed certain threshold, it is marked as a foreground. The foreground is identified as “moving object”.

3.2 Key Concepts

3.2.1 Background Modelling by Codebook Approach Based on Multiple Code Words and Code Word Sharing

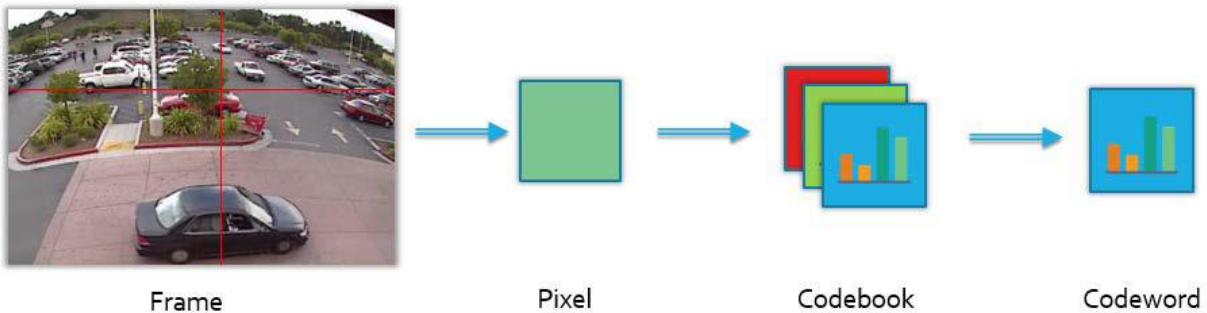


Figure 3.0 Structure of Code Book Model

In this project, the key idea of the system is to generate a background model by extracting the dominant codeword for each pixel and to generate a foreground mask results by comparing the original test frame to the background model. Usually, the whole process can be divided into two main stages, the training stage and the test stage.

In the training phase, first, the system divides all input training frames with user-specified ROI dimensions into output sequences of smaller frames. It helps reduce processing time and memory consumption when verifying the system process. By observing only a small portion of the output, it can help us to concentrate while conducting analysis on those small outputs.

The system employs a codebook model to handle each quantized color bin that will be incrementally updated with the original color vector retrieved from the training frame. A pixel represents a codebook and a codebook is constructed for each individual pixel in the training ROI. Each codebook contains a codeword with the total number of all possible quantized color bins. Similarly, each codeword represents a particular quantized color box and contains statistical data, such as the frequency of occurrence of the respective quantized color bin for that particular pixel. In the case where a null codebook is prepared, the system reads in the training frames of each partition to fill the codebooks. For each color vector read from a particular pixel, its original

CHAPTER 3 SYSTEM DESIGN

vector value is accumulated and its frequency of occurrence is increased in the matched quantized codebook.

As all pixel codebooks are updated, for each codebook, system is extracting a number of dominant codewords among all the others by sorting all codewords according to its frequency of occurrence. In the original system, only one dominant codeword will be extracted for each pixel, while other pixel words will be treated as the foreground color of that particular pixel. It works well on static background pixels with absolute dominant codewords, and all other codewords cannot exceed the codeword in terms of frequency. However, it becomes problematic when it comes to dynamic background pixels, which including interchangeably occupying multiple dominant colors on that particular pixel. Since the original system does not treat any other dominant codewords as a background, even if the frequency difference between the subsequent codewords and the first codewords is small, false detection often occurs on the pixels.

It is concluded that the background model of a single dominant codeword is less efficient against dynamic background scenes. Therefore, in our final system, we developed a background training model that comprises multiple dominant codewords for each pixel. The purpose of defining a frequency coverage threshold is to limit the extent to which the system should consider the current color bin to be dominant in the ordered codeword array. However, although the false positive on foreground detection is greatly reduced, but it is brings along side effect of increasing in false negative detection due to the suppression for the condition to categorize a pixel to foreground group has become stricter.

In order to solving this problem, codeword sharing is applied which this method separate the dynamic and static pixels based on defined threshold value of codeword. If the total number of codeword is greater than the given threshold, it will proceed to handle the dynamic background process with find the maximum codeword and synthesize the low value codeword value. Otherwise, it will handle the static background process.

Once the multi-layer background model is prepared, the system will enter the testing phase. The system reads in the test frame and obtains the original color vector from each pixel for background subtraction. The retrieved color vector from the test frame is compared to each

CHAPTER 3 SYSTEM DESIGN

major codeword for that particular pixel. To determine if it is foreground or background, the system calculates the mean vectors of all the original color vectors that fall within a particular dominant codeword, and finds the absolute difference between the colors vectors retrieved from the test frame. So far, a series of result indicating foreground and background pixels have been generated. To further improve the foreground mask, the system applies a flood fill method to the foreground mask to remove shadow and fill the blank space within the foreground mask component by expansion. It helps to improve the quality of the input bitmap connected to the foreground component to reduce the erroneous final output.

3.2.2 Lighting Effect on Color Intensities














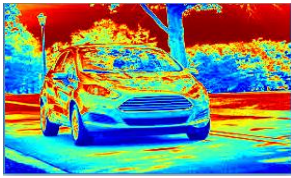
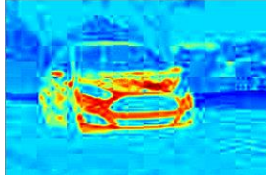
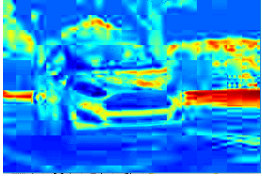
Original Image	Color component #1	Color component #2	Color component #3
	 B- Image	 G-Image	 R-Image
	 L- image	 A-Image	 B- Image
	 L – Image	 A –Image	 B- Image
	 L- Color Jet Map	 A- Color Jet Map	 B-Color Jet Map

Figure 3.1 Lighting Effect on Color Intensities

In this subsection, the main goal is to identify the optimal preference among the RGB and LAB color variation which affected by lightness problem. Light pixels can have greater fluctuation in its intensity values compared to dark pixels. The dark pixels provide the high uncertainty compared to bright pixels due to color ratios uncertainty is relevant to lightness. This cause the detection in the dark pixel region not stable and wrong detection clustered around the dark regions. Therefore lightness shall be considered as one of the factor in color ratio comparison.

CHAPTER 3 SYSTEM DESIGN

LAB color space is better instead of RGB color space in reducing false positive because LAB color space is more approximate to human vision whereas RGB color space is device-dependent. In the LAB color space, the L channel is independent with the color information and encodes lightness only. Else of the two channels are encode colors. In other words, L-distribution stand for lightness, A-distribution and B-distribution stand for color-opponent dimension. Not like in RGB color space, the information of color is split into three channels and these three channels are encodes the lightness information. Therefore, RGB color space lowers the sensitivity analysis of detection because much variance at a pixel is involving consideration on level of illumination, which is dark or bright. In order to solve the lightness problem such as shadow, the post processing by using flood fill to enhance the foreground mask image for shadow removal will be conducted.

3.3 System Flow Chart Diagram

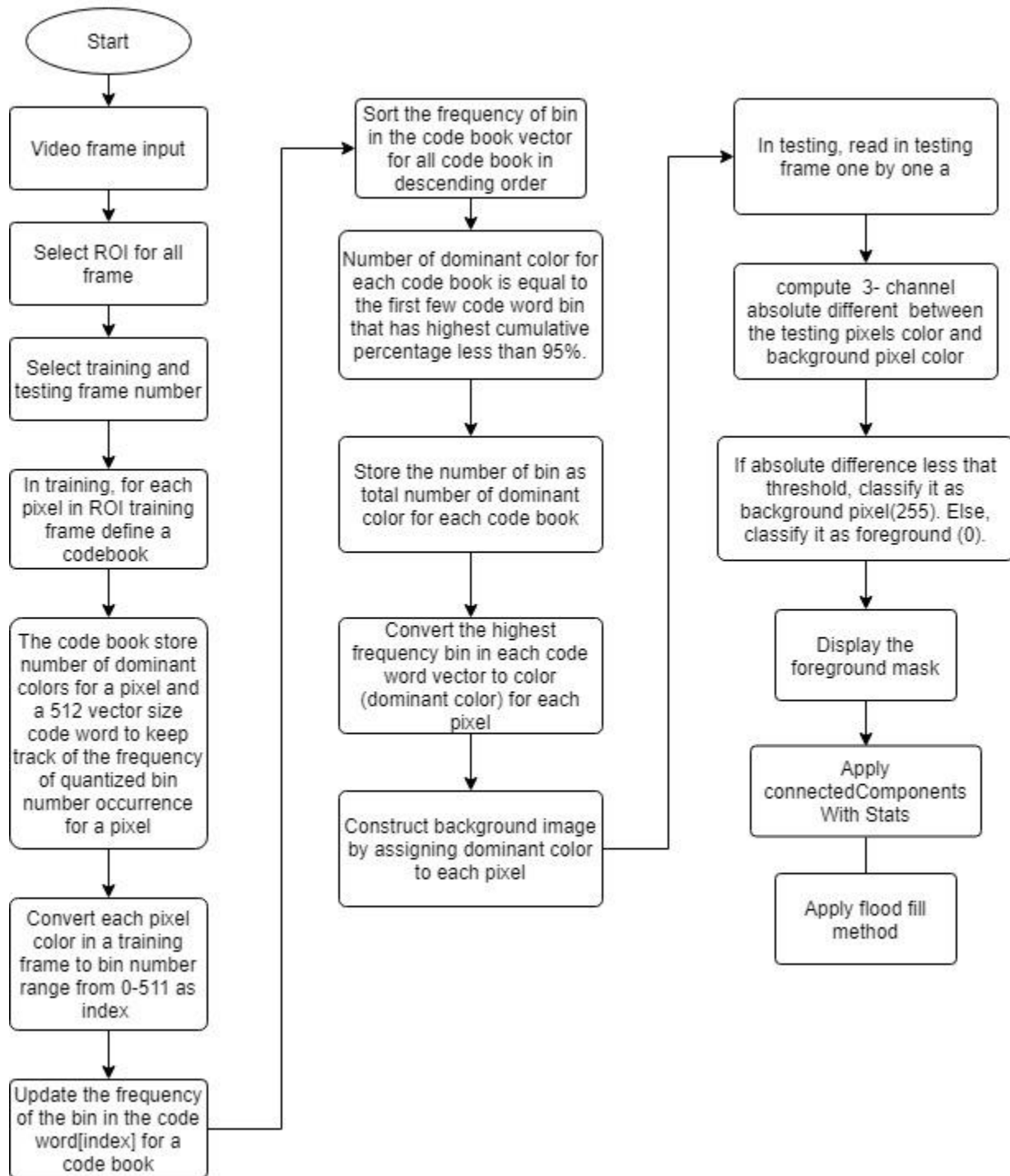


Figure 3.2 System Flow Chart Diagram

3.4 System Pseudocode

Pre - Processing Stage

STEP 1: Display the properties about the video and read in the first frame from video sequence

- a. Get the properties of the video sequence, $frame_nums \leftarrow$ total number of frames, $width \leftarrow$ frame width, $height \leftarrow$ frame height, $fps \leftarrow$ frame rate, etc. and display all the properties.
- b. $sampleFrame \leftarrow$ first frame

STEP 2: Determine the training phase and testing phase parameters with user key in in order to reduce the training and testing processing time.

- $trainStartFrameIdx \leftarrow$ Frame position of starting training frame
 $trainEndFrameIdx \leftarrow$ Frame position of ending training frame
 $testStartFrameIdx \leftarrow$ Frame position of starting testing frame
 $testEndFrameIdx \leftarrow$ Frame position of ending testing frame

STEP 3: Use the first frame to select ROI (region of interest). Crop all video frames sequence according to the selected ROI into sequential ROI images and copy into a vector with Mat type. After that, display the new ROI width and height of the frames.

Perform Background Modeling (Training Stage)

Step 1: Construct a codebook with the total size of frame, codebook [frame height * frame width] and it consist of a list of code word with different length. For each code word in codebook, it consists of the attributes including quantized color and frequency of code word occurs.

Step 2: Read in the ROI image of the training frame one at a time.

Step 3: Loop through every pixel of the ROI image and convert the colour of the pixel into bin number and go to the codebook for the corresponding pixel and update the frequency of the bin in the code word [bin number] vector with size 512.

Step 4: After finish reading the entire frame, sort the frequency of bin in the code word vector in descending order for each codebook

Step 5: The number of dominant colours for a code book is equal to the first few code word bins that have the highest cumulative frequency less than 95%.

Step 6: Construct the background image pixel by pixel by converting the highest frequency of bin in the code word of a codebook for the pixel to colour and assign the colour to the pixel of the background.

Step 7: Display the background model.

Foreground Detection (Testing Stage)

Step 1: Read in ROI image from the testing frame sequence one at a time

Step 2: Comparing the input pixel in current testing frame with the corresponding pixel background model by performing 3-channel absolute of color. If the condition is match, such pixel is classified as background; otherwise it is classified as foreground.

Step 3: Display the foreground mask result.

Post Processing – Stage

Remove the shadow from the foreground mask image

Step 1: Create binary image for triple value foreground masks (0 = foreground mask 0,255, 1 = foreground mask 128)

Step 2: Based on the binary image, construct the connected component and obtain the boundary box

Step 3: Based on each of bounding box, we plot the L* distribution color histogram.

Step 4: Based on the color histogram, we are differentiating there is single (foreground) or double different connected region (shadow and black body).

Step 5: To extract shadow region, scan the pixel in the boundary box whose binary image value =1. Use this pixel as seed point and apply the flood fill method.

Step 6: Repeat for this step for the multi-value foreground masks to remove the shadow pixel

Step 7: relabel the shadow pixels in fore mask image to become background (255).

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

4.1 System Code Description

The figure below is shown the overall structure and the flow of the program.

```

#include <iostream>
#include <string.h>
#include <ctime>
#include <math.h>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp> // OpenCV Window IO
#include <cmath>
#include <algorithm>

using namespace cv;
using namespace std;

struct CodeWAtt{ ... };

//***** global variable *****/
vector< vector<CodeWAtt> > CodeWAttriAtPixel;//CodeWAttriAtPixel[pixelcount][QcolorBin] ; Total pixel number unknown until training
Point VertexLeftTop(-1, -1); // Roi initial setting points
Point VertexRightDown(-1, -1);
Mat Roi;
vector<int>TotalNumCW;//dynamic memory using push_back;TotalNumCW[pixelcount]
int NumDominColors = 0;
float cumulativePercent = 0.0;
// 0508*****
bool flag_colorConver = false; // Normally bgr color space
bool flag_CWBorrowing = false; // Normally no borrowing
bool flag_KeyInROI = false;

int trainStartFrameIdx, trainEndFrameIdx;
int testStartFrameIdx, testEndFrameIdx;
//*****

// ***** prototype *****
int bgr2QColorBinIndex(Vec3b color);
Vec3b QColorBinIndex2bgr(int bin);
void onMouse(int Event, int x, int y, int flags, void* param);
void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train);
void testing_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_test);

//*****

//----- read a video and show the video info-----
int main(){ ... }

// ----- mouse to click for select Roi and use setMouseCallBack statement-----
void onMouse(int Event, int x, int y, int flags, void* param){ ... }

//***** TRAINING Stage *****/
void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train){ ... }

//*****TESTING Stage*****/
void testing_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_test){ ... }

// convert the color value of the pixel into 8bin color indices
int bgr2QColorBinIndex(Vec3b color){ ... }

//convert the 8bin color indices format value into normal color value
Vec3b QColorBinIndex2bgr(int bin){ ... }

```


In beginning, include the necessary library and the namespace.

```
#include <iostream>
#include <string.h>
#include <ctime>
#include <math.h>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp> // OpenCV Window IO
#include <cmath>
#include <algorithm>

using namespace cv;
using namespace std;
```

4.1.1 Data Structure and Classes

Then, define the structure or class needed for codebook based background subtraction algorithm which named as CodeWAtt. The main purpose of defining this structure is to hold the updated statistical data about a specific quantized color bin of a particular pixel during the training phase and to be used in the process of dominant codewords extraction and then background subtraction.

This structure has the following attributes and functions:

```
struct CodewAtt
{
    int QColorBinFrequency;//the number of the code word appearance
    int QColorBinIndex;// the quantized color bin index
    Vec3f original_color_mean; //mean_color_value
    Vec3f original_color_std_devi; // standard deviation_color_value

    CodewAtt()
    {
        QColorBinFrequency = 0;
        QColorBinIndex = 0;
        original_color_mean = { 0, 0, 0 };
        original_color_std_devi = { 0, 0, 0 };
    }
    bool operator< (const CodewAtt x) const
    {
        //operator overloading
        return this->QColorBinFrequency > x.QColorBinFrequency;
    }
};
```

In this structure, it is overload the < operator to let the program can sort the CodewAtt according to the frequency of color value will have higher sorting index during the training phase.

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

4.1.2 Global variables

Define the required global variables which list as shown as below.

```
//***** global variable *****/
vector< vector<CodeWAtt> > CodeWAttriAtPixel;//CodeWAttriAtPixel[pixelcount][QcolorBin] ; Total pixel number unknown until training
Point VertexLeftTop(-1, -1); // Roi initial setting points
Point VertexRightDown(-1, -1);
Mat Roi;
vector<int>TotalNumCW;//dynamic memory using push_back from NumDominantColors;TotalNumCW[pixelcount]
int NumDominColors = 0;
float cumulativePercent = 0.0; // cumulative frequency percent for dominant color.

bool flag_colorConver = false; // Normally bgr color space
bool flag_CWBorrowing = false; // Normally no borrowing
bool flag_KeyInROI = false; // Normally no roi key in

int trainStartFrameIdx, trainEndFrameIdx; // Starting and ending frame for training phase
int testStartFrameIdx, testEndFrameIdx; // Starting and ending frame for testing phase
```

4.1.3 Function Lists

The below shows the function list which will be used in the algorithm.

```
// ***** function list prototype *****/
int bgr2QColorBinIndex(Vec3b color);
Vec3b QColorBinIndex2bgr(int bin);
void onMouse(int Event, int x, int y, int flags, void* param);
void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train);
void testing_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_test);
```

4.1.4 Main Program Function

In main program, choosing the test video and read the first video frame for the purpose of ROI selection. Then, check whether the read is successful or not. If the video file not exists, the program will prompt an error. Otherwise, the program will display the video properties.

```
int main()
{
    // ...

    // ...
    string filename = "TCEMS with shadow.avi"; // 3264 , 4000 , 5500

    VideoCapture inputVideo(filename);
    if (!inputVideo.isOpened()) { // check if the video is valid
        std::cout << "Could not open the input video!" << endl;
        return -1;
    }

    int frame_nums = (int)inputVideo.get(CV_CAP_PROP_FRAME_COUNT), //get video information
        width = (int)inputVideo.get(CV_CAP_PROP_FRAME_WIDTH),
        height = (int)inputVideo.get(CV_CAP_PROP_FRAME_HEIGHT);
    double frame_rate = inputVideo.get(CV_CAP_PROP_FPS);

    std::cout << "Video name: " << filename << endl;
    std::cout << "Frame numbers: " << frame_nums << endl;
    std::cout << "Width: " << width << endl;
    std::cout << "Height: " << height << endl;
    std::cout << "Frame rate: " << frame_rate << " fps" << endl;
    std::cout << "Video length: " << (int)(frame_nums / frame_rate / 60) << "mins" << (int)(frame_nums / frame_rate) % 60 << "secs" << endl;
}
```

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

After choosing a video file, allow user enter the start and end training frame number. As the total training frame difference, the result of foreground detection will be difference also. Besides, user also need to enter the start and end testing frame number to allow user select their interest part of video sequence. Since the result accuracy of foreground detection is depends on the dominant color, thus the program is allow user to determine the suitable percentage value to classify the boundary between the dominant colors.

```
//*****  
//                               SELECT START AND END FRAME FOR TRAINING AND TESTING PHASE  
//*****  
  
std::cout << " Please enter both the starting and ending frame numbers for the training sequence: ";  
cin >> trainStartFrameIdx >> trainEndFrameIdx;  
std::cout << " Please enter both the starting and ending frame numbers for the testing sequence: ";  
cin >> testStartFrameIdx >> testEndFrameIdx;  
std::cout << "Please enter the cumulative percentage for the dominant colors: (0.0 ~ 0.99) ";  
cin >> cumulativePercent;
```

Lastly, user needs to draw a rectangle shape to select ROI image after the input of the video by calling the “setMouseCallback” function. After get the ROI image, the function will generate the video frames according the ROI image. A vector is created for to store the frames of data set for later use. There is consisting of two ways of selecting Region of ROI where one of way is using mouse to draw a rectangle to select the ROI image. Meanwhile, another way is user defined the rectangle properties such as the x-left position, y-top position, width and height. After that, press Esc key to process to training phase.

```
Mat image;  
vector <Mat> frames(frame_nums), frames_train(frame_nums), frames_test(frame_nums);  
  
if (flag_KeyInROI == false)  
{  
    inputVideo.read(image);//inputV >> image;  
    namedWindow("SelectRoi", CV_WINDOW_AUTOSIZE);  
    imshow("SelectRoi", image);  
    std::cout << "\n Drag the mouse's left key to select two points for roi and then release." << endl;  
    std::cout << "Press the esc key to quit the roi selection!\r";  
    setMouseCallback("SelectRoi", onMouse, NULL);  
  
    while (true)  
    {  
        if (VertexLeftTop.x != -1 && VertexRightDown.x != -1)  
        {  
            Mat RoiInit = image.clone();  
            rectangle(image, Rect(VertexLeftTop, VertexRightDown), Scalar(255, 0, 0), 1, 8, 0);  
            Rect WhereRec(min(VertexLeftTop.x, VertexRightDown.x), min(VertexLeftTop.y, VertexRightDown.y),  
                abs(VertexLeftTop.x - VertexRightDown.x), abs(VertexLeftTop.y - VertexRightDown.y));  
  
            RoiInit(WhereRec).copyTo(Roi);  
  
            cv::imshow("Roi", Roi);  
        }  
        if (cvWaitKey(33) == 27)  
        {  
            break;//break out of the while loop  
        }  
    }  
}
```

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

```
Rect WhereRec = Rect(min(VertexLeftTop.x, VertexRightDown.x), min(VertexLeftTop.y, VertexRightDown.y),
    abs(VertexLeftTop.x - VertexRightDown.x),abs(VertexLeftTop.y - VertexRightDown.y));

int trainFrameLength = trainEndFrameIdx - trainStartFrameIdx + 1;
inputVideo.set(CAP_PROP_POS_FRAMES, trainStartFrameIdx - 1);

for (int k = 1; k <= trainFrameLength; k++)
{
    inputVideo >> image;
    Mat RoiInit = image.clone();
    RoiInit(WhereRec).copyTo(frames_train[k]);
}

width = frames_train[1].cols;
height = frames_train[1].rows;
std::cout << "\nNew Roi Width: " << width << endl;
std::cout << "\nNew Roi Height: " << height << endl << endl;

int testFrameLength = testEndFrameIdx - testStartFrameIdx + 1;
inputVideo.set(CAP_PROP_POS_FRAMES, testStartFrameIdx - 1);
for (int k = 1; k <= testFrameLength; k++)
{
    inputVideo >> image;
    Mat RoiInit = image.clone();
    RoiInit(WhereRec).copyTo(frames_test[k]);
}
}

if (flag_KeyInROI == true)
{
    inputVideo.read(image);//inputVideo >> image;
    //int x = 96, y = 129, W = 4, H = 4;
    //int x = 62, y = 94, W = 44, H = 49;
    int x = 93, y = 12, W = 71, H = 43;

    cv::Rect WhereRec(x, y, W, H);
    Mat RoiInit = image.clone();

    RoiInit(WhereRec).copyTo(Roi);
    cv::imshow("Roi", Roi);
    waitKey(10);

    int trainFrameLength = trainEndFrameIdx - trainStartFrameIdx + 1;
    inputVideo.set(CAP_PROP_POS_FRAMES, trainStartFrameIdx - 1);

    for (int k = 1; k <= trainFrameLength; k++)
    {
        inputVideo >> image;
        Mat RoiInit = image.clone();
        RoiInit(WhereRec).copyTo(frames_train[k]);
    }

    width = frames_train[1].cols;
    height = frames_train[1].rows;
    std::cout << "\nNew Roi width: " << width << endl;
    std::cout << "\nNew Roi Height: " << height << endl << endl;

    int testFrameLength = testEndFrameIdx - testStartFrameIdx + 1;
    inputVideo.set(CAP_PROP_POS_FRAMES, testStartFrameIdx - 1);
    for (int k = 1; k <= testFrameLength; k++)
    {
        inputVideo >> image;
        Mat RoiInit = image.clone();
        RoiInit(WhereRec).copyTo(frames_test[k]);
    }
}
```

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

In the background training phase, the training frame is start from the user-define training frame number based on the main program. When the first training frame is read, the program will loop through every pixel of the frames which corresponds to the RoiWidth and RoiHeight to assign the color pixels into the 2- dimension CodeWAttr vector created. `int QColorBinIndex` is used to store the value of bins in each pixels and it is then stored in `QColorBinIndex` in the `CodeWAttr`. For every loop, the frequency of the `CodeWAttr` increases to indicate how many pixels are generated. Besides, the mean value of colours pixel and total of mean square of colours pixels is also calculated to compute the standard deviation of colours pixel.

```
void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train)
{
    std::cout << "*****Training Phase*****" << endl;

    std::cout << "\nflag_colorConver = " << flag_colorConver << endl;
    std::cout << "\nflag_CWBorrowing = " << flag_CWBorrowing << endl;
    std::cout << "\nflag_KeyInROI = " << flag_KeyInROI << endl;

    int start_s = clock();
    Mat img;
    float progress = 0.0;
    int barWidth = 35;
    VideoCapture inputVideo(filename);

    // ----- GET PIXEL COLOR -----
    //get a vector storing values of each pixel's color from all training video frames and used to generate the background model
    // resize codebook with length of area without constructor
    //For each codebook, assign 512 constructor of codeword
    vector<vector<CodeWAttr>> CodeWAttrAtPixel2((width*height), vector<CodeWAttr>(512, CodeWAttr()));

    int pixelCount = 0;

    int trainFrameLength = trainEndFrameIdx - trainStartFrameIdx + 1;

    std::cout << "\nTrain Starting Frame and Train Ending Frame are " << trainStartFrameIdx
        << ", " << trainEndFrameIdx << endl;

    //-----
    vector<vector<Vec3f>> original_color_squared_mean;// 1/8/2018
    original_color_squared_mean.resize((width*height));
    Vec3b pixel_color; // pixel_color is to display the training samples for debugging

    for (int i = 1; i <= trainFrameLength; i++)
    {
        img = frames_train[i];

        if (flag_colorConver == true) // change from bgr to Lab
        {
            cvtColor(img, img, CV_BGR2Lab); //Save LAB image only
        }

        pixelCount = 0;
        if (img.empty())
        {
            std::cout << "Frame " << i << " is empty" << endl;
            break;
        }

        if (flag_colorConver == true) // Now in the Lab space
        {
            cvtColor(img, img, CV_Lab2BGR); //Save RGB image only
        }
        imshow("Training Frames", img); // always in the bgr space

        if (flag_colorConver == true)
        {
            cvtColor(img, img, CV_BGR2Lab);
        }
        waitKey(1);
    }
}
```

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

```
//Construct the attributes for each quantized color which represents a code word
for (int y = 0; y < img.rows; y++)
{
    for (int x = 0; x < img.cols; x++)
    {
        //-----
        if (original_color_squared_mean[pixelCount].size() == 0)
            original_color_squared_mean[pixelCount].assign(512, { 0, 0, 0 });

        pixel_color = img.at<Vec3b>(Point(x, y));
        int QColorBinIndex = bgr2QColorBinIndex(img.at<Vec3b>(Point(x, y)));
        CodeWAtt &CodeWord = CodeWAttriAtPixel2[pixelCount][QColorBinIndex];
        //Construct the attributes for each quantized color which represents a code word
        CodeWord.QColorBinIndex = QColorBinIndex;
        CodeWord.QColorBinFrequency++;

        for (int chann = 0; chann < 3; chann++)
        {
            CodeWord.original_color_mean[chann]
                = ((CodeWord.QColorBinFrequency - 1) * CodeWord.original_color_mean[chann] + (img.at<Vec3b>(Point(x, y))[chann])) /
                CodeWord.QColorBinFrequency;

            original_color_squared_mean[pixelCount][QColorBinIndex][chann]
                = ((CodeWord.QColorBinFrequency - 1) * original_color_squared_mean[pixelCount][QColorBinIndex][chann] + pow((img.at<Vec3b>(Point(x, y))[chann]), 2)) /
                CodeWord.QColorBinFrequency;
        }
        // ...
        pixelCount++;
    }
}

for (int i = 0; i < CodeWAttriAtPixel2.size(); i++)
{
    //-----
    vector<CodeWAtt> &CodeWords = CodeWAttriAtPixel2[i];
    for (int j = 0; j < CodeWords.size(); j++)
    {
        for (int chann = 0; chann < 3; chann++)
        {
            CodeWords[j].original_color_std_devi[chann]
                = sqrt(original_color_squared_mean[i][j][chann] -
                    pow(CodeWords[j].original_color_mean[chann], 2));
        }
    }
}
}
```

After all training pixel codebooks are updated with all training frames, system starts to extract dominant code words for each training pixel to produce final background model. It will sort the code word vector first with the frequency (count) of that code word appears from high to low. In main program, user will enter the percentage value. The frequency of code word will continuing accumulate until the value is exceeds the user-define percentage value, the other color pixel which are not count into accumulate value will be remove. The number of sorted dominant colors is found. And by here, a background[i][j] model with multi-layer dominant codewords is produced.

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

```
// ----- CONSTRUCT BACKGROUND -----  
//By taking the vector storing all pixel's color  
//Compare the dominant color and original frame color at a particular px  
//If matches, construct background by using original frame BGR value  
Mat background = Mat::zeros(height, width, CV_8UC3);  
int x = 0, y = 0;  
  
float sum = 0.0;  
  
// For each pixel, find the sorted dominant colors  
for (int j = 0; j < CodeWAttriAtPixel2.size(); j++)  
{  
    sort(CodeWAttriAtPixel2[j].begin(), CodeWAttriAtPixel2[j].end());  
    // sorting the quantized color vector in the descending order  
  
    int quantization_step = 32;  
    int quantization_level = 256 / quantization_step;  
    for (int k = 0; k < pow(quantization_level, 3); k++)  
    {  
        sum += CodeWAttriAtPixel2[j][k].QColorBinFrequency;  
  
        if (sum > trainFrameLength * cumulativePercent)  
        {  
            NumDominColors = k + 1;  
  
            if (j == CodeWAttriAtPixel2.size() - 1)  
                std::cout << "\nSum, NumDominColors =" << sum << "\t" << NumDominColors << endl;  
  
            sum = 0;  
            break;  
        }  
        TotalNumCW.push_back(NumDominColors); // the only dynamic memory  
  
        /* ... */  
  
        background.at<Vec3b>(Point(x, y)) = QColorBinIndex2bgr(CodeWAttriAtPixel2[j][0].QColorBinIndex);  
  
        x++;  
        if (x >= width)  
        {  
            x = 0;  
            y++;  
            // in the end of the j-loop x reaches width and y reaches height.  
        }  
  
        /* ... */  
  
        /* ... */  
  
        /* ... */  
  
        /* ... */  
  
        /* ... */  
  
    }  
    CodeWAttriAtPixel = CodeWAttriAtPixel2;//the transfer is needed  
  
    std::cout << "\nTotalNumCW.size():" << TotalNumCW.size() << endl;  
  
    if (flag_colorConver == true)  
    {  
        cvtColor(background, background, CV_Lab2BGR);  
    }  
    imshow("background", background);  
    waitKey(1);  
}
```


CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

Proceeding to testing phase, the video frame jump to the user-define starting testing frame number and enter testing phase. The system initializes *CodeWAttrAtPixel* with background model returned from *CodeWAttrAtPixel2*. For each testing frame, colors vector is retrieved from each pixel and compared with the each dominant code words in the background model. Through background subtraction, the system categorizes whether a pixel is background or foreground, with a frame-by-frame, and pixel-by-pixel basis. At the end of testing phase, a sequence of foreground detection bitmap results shall be produced and displayed to user.

```
void testing_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_test)
{
    std::cout << "*****Testing Phase*****" << endl;

    Mat img;
    float progress = 0.0;
    int barwidth = 35;
    VideoCapture inputVideo(filename);
    Mat foreground = Mat::zeros(height, width, CV_BUC1);
    double x, y = 0.0;
    //int size_threshold = 50;
    int size_threshold = 64;
    int start_s = clock();

    //int testStartFrame = (int)(frame_nums* 0.7); // 2/8
    //int testEndFrame = (int)(frame_nums *0.85); //2/8
    int testFrameLength = testEndFrameIdx - testStartFrameIdx;

    std::cout << "\nTest Starting Frame and Test Ending Frame are " << testStartFrameIdx
        << ", " << testEndFrameIdx << endl;

    for (int i = 1; i <= testFrameLength; i++)
    { //loop through frames; i is better renamed as frameidx; not to use all frames!
        int pixelCount = 0; //for the new testing frame reset the pixelCount zero

        img = frames_test[i];

        imshow("Testing Frames", img);
        //imwrite("DayTimeHighway_1429_BGR.jpg", img);

        if (flag_colorConver == true) // Now in the Lab space
        {
            cvtColor(img, img, CV_BGR2Lab);
        }
    }
}
```


CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

```
for (int y = 0; y < img.rows; y++)
{
    //compare the color of foreground pixel and bg pixel
    for (int x = 0; x < img.cols; x++)
    {
        foreground.at<uchar>(y, x) = 0;
        // Initially each pixel set it as a fg (0)
        // Check if its color is similar to one of all code words? If yes, it is bg; otherwise, it is fg

        // ...

        for (int i = 0; i < TotalNumCW[pixelCount]; i++)
        {
            if (abs(QColorBinIndex2bgr(CodeWAttriAtPixel[pixelCount][i].QColorBinIndex)[1] - img.at<Vec3b>(y, x)[1]) < 30 &&
                (abs(QColorBinIndex2bgr(CodeWAttriAtPixel[pixelCount][i].QColorBinIndex)[2] - img.at<Vec3b>(y, x)[2]) < 30))
            {
                if ((QColorBinIndex2bgr(CodeWAttriAtPixel[pixelCount][i].QColorBinIndex)[0] - img.at<Vec3b>(y, x)[0])
                    > 0.20*(QColorBinIndex2bgr(CodeWAttriAtPixel[pixelCount][i].QColorBinIndex)[0]))
                {
                    foreground.at<uchar>(y, x) = 128; // shadow or black moving object
                    break;
                }

                else if (abs(QColorBinIndex2bgr(CodeWAttriAtPixel[pixelCount][i].QColorBinIndex)[0] - img.at<Vec3b>(y, x)[0]) < 30)
                {
                    foreground.at<uchar>(y, x) = 255; // background

                    break;
                }
            }
        }
        pixelCount++;
    }
}

if (((int)progress * 100) % 10 == 0)
{
    std::cout << int(progress * 100.0) << "%\n";
    std::cout.flush();
    progress = (float)(i + 0.0) / testFrameLength;
}

cv::imshow("Binary foreground mask", foreground);
cv::waitKey(1);
}

std::cout << "Done! " << endl;
std::cout << endl;
int stop_s = clock();
std::cout << "Time elapsed: " << int(((stop_s - start_s) / double(CLOCKS_PER_SEC)) / 60) << " min "
    << (int)((stop_s - start_s) / double(CLOCKS_PER_SEC)) % 60 << " sec" << endl << endl;

std::cout << "\nflag_colorConver = " << flag_colorConver << endl;
std::cout << "\nflag_CWBorrowing = " << flag_CWBorrowing << endl;
std::cout << "\nflag_KeyInROI = " << flag_KeyInROI << endl;
}
```

4.1.5 Function Declaration

Mouse Click Event Function

```
void onMouse(int Event, int x, int y, int flags, void* param)
{
    if (Event == CV_EVENT_LBUTTONDOWN) {
        VertexLeftTop.x = x;
        VertexLeftTop.y = y;
    }
    if (Event == CV_EVENT_LBUTTONUP) {
        VertexRightDown.x = x;
        VertexRightDown.y = y;
    }
}
```

This function is used to handle mouse event to let user select ROI. Once the ROI boundary is selected, set the start-coordinate of (x, y) and end-coordinate of (x, y) using Rect class.

C++ : void **onMouse** (int **Event**, int **x**, int **y**, int **flags**, void* **params**)

Parameters:

- Event - mouse condition
- x – the cols coordinate
- y- the rows coordinate
- flags – mouse condition
- void* param – other parameters.

Color Quantization Function

```
// convert the color value of the pixel into 8bin color indices
int bgr2QColorBinIndex(Vec3b color)
{
    int quantization_step = 32;
    int b = color[0] / quantization_step, g = color[1] / quantization_step, r = color[2] / quantization_step;
    return 64 * b + 8 * g + r;
}
```

Convert the BGR 3 channel color value into quantized bin color indices [0,512]. This function converts the color value of the pixel into 8 bin format and it takes a variable with type Vec3b as the only parameter.

C++: int **bgr2QColorBinIndex** (Vec3b **color**)

Parameters:

- color – Vec3b type to be converted into 8 bin format value.

Color Conversion Function

```
//convert the 8bin color indices format value into normal color value
Vec3b QColorBinIndex2bgr(int bin)
{
    int b = bin / 64, r = bin % 8, g = (bin / 8) % 8;
    return Vec3b(b * 32 + 16, g * 32 + 16, r * 32 + 16);
}
```

Convert the 8 bin color indices [0,512] into RGB color. This function converts the 8 bin format into color value of the pixel and it takes a variable with type integer as the only parameter.

C++: Vec3b QColorBinIndex2bgr (int bin)

Parameters:

- **bin** – 8 bin format value to be converted into Vec3b value.

Training phase function

```
void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train){ ... }
```

C++: void training_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_train)

Parameters:

- **filename** – Name of the video file to be loaded.
- **frame_nums** – Total number of frames of the video data set.
- **width** – ROI width of the frame of video data set.
- **height** – ROI height of the frame of video data set.
- **Frames_train** – A vector storing every frame of data set for training.

Training phase of the system is to find the background image or in another word background modelling for every pixel in all the data set frames. The output of this phase will be a vector consists of attributes needed for every pixels and a background model.

CHAPTER 4 SYSTEM TESTING AND IMPLEMENTATION

Testing phase function

```
void testing_phase(string filename, int frame_nums, int width, int height, vector<Mat> frames_test){ ... }
```

C++: void **training_phase**(string **filename**, int **frame_nums**, int **width**, int **height**, vector<Mat> **frames_test**)

Parameters:

- **filename** – Name of the video file to be loaded.
- **frame_nums** – Total number of frames of the video data set.
- **width** – ROI width of the frame of video data set.
- **height** – ROI height of the frame of video data set.
- **frames_test** – A vector storing every frame of data set for testing.

Testing phase of the system is to construct the binary foreground mask for every pixel in all the data set frames.

4.2 System Code Testing and Verification

4.2.1 Testing Core Functions on Video Clips

Generally, in motion detection, there are three common steps as following, unless specified for post- processing steps:

A. ROI Selection

First, read in a video and get the first frame from video sequence. The window namely “SelectRoi” which displayed the first frame capture from the video and will allow the user to select the ROI in rectangle border. Only ROI selected by user is being processed.

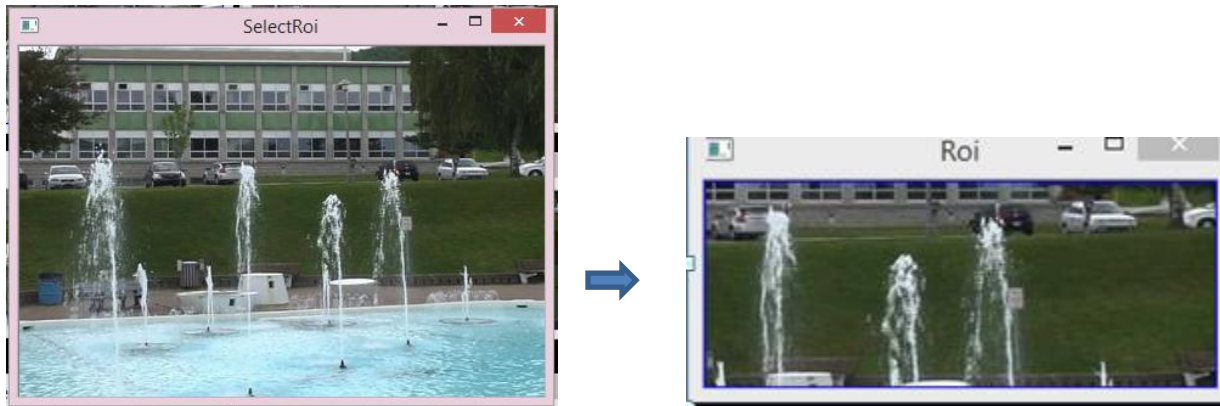


Figure 4.0 ROI selection

B. Perform Background Modeling (Training Stage)

Next, it will proceed to training phase. After the processing step, the background is constructed based on highest frequency of color index in each pixel.



Figure 4.1 Performing Background Modelling

C. Foreground Detection (Testing Stage)

Lastly, it will proceed to testing phase. And then, the foreground mask output result is generated.



Figure 4.2 Foreground Detection

4.2.2 Debugging tool for image colors conversion and display.

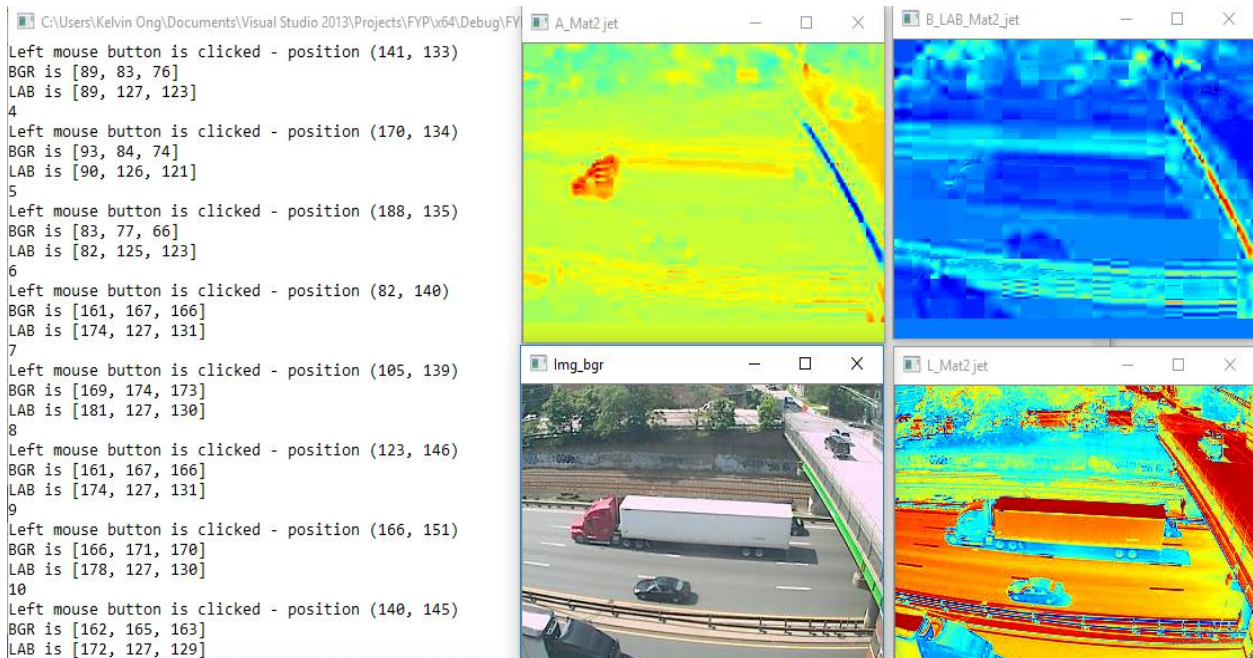


Figure 4.3 Debugging tool for image colors conversion and display

In this project, the debugging tool purpose to display the image colors conversion in order to view the difference of the image between L- distribution, A-distribution and B-distribution. Next, it is used to check the absolute difference of color intensity value for LAB color space. After that, the parameter will be set in the system based on the absolute difference of the color intensity value of the LAB color space. The reason of eliminating fixed difference threshold as implemented in system is that the variance in fluctuating intensity values between light and dark pixel is not identical. Light pixels can have greater fluctuation in its intensity values compared to dark pixels.

For instance, the road surface pixels have an A-distribution around under shadow and from 125 to 126 without shadow. Therefore, the A-distribution is roughly the same under the object or not under the object. On the other hand, the B -distribution are from 121 to 123 under shadow and from 129 to 133 without shadow. Therefore, the B-distribution is not quite the same under the object or not under the object. From the L distribution it can tell whether there is has shadow from (82 to 90) or without shadow (from 172 to 181).

4.2.3: Testing Result on Background Modelling by Code book Approach Based on Multiple Code Words and Code Word Sharing

Set 1: (fall2.avi)

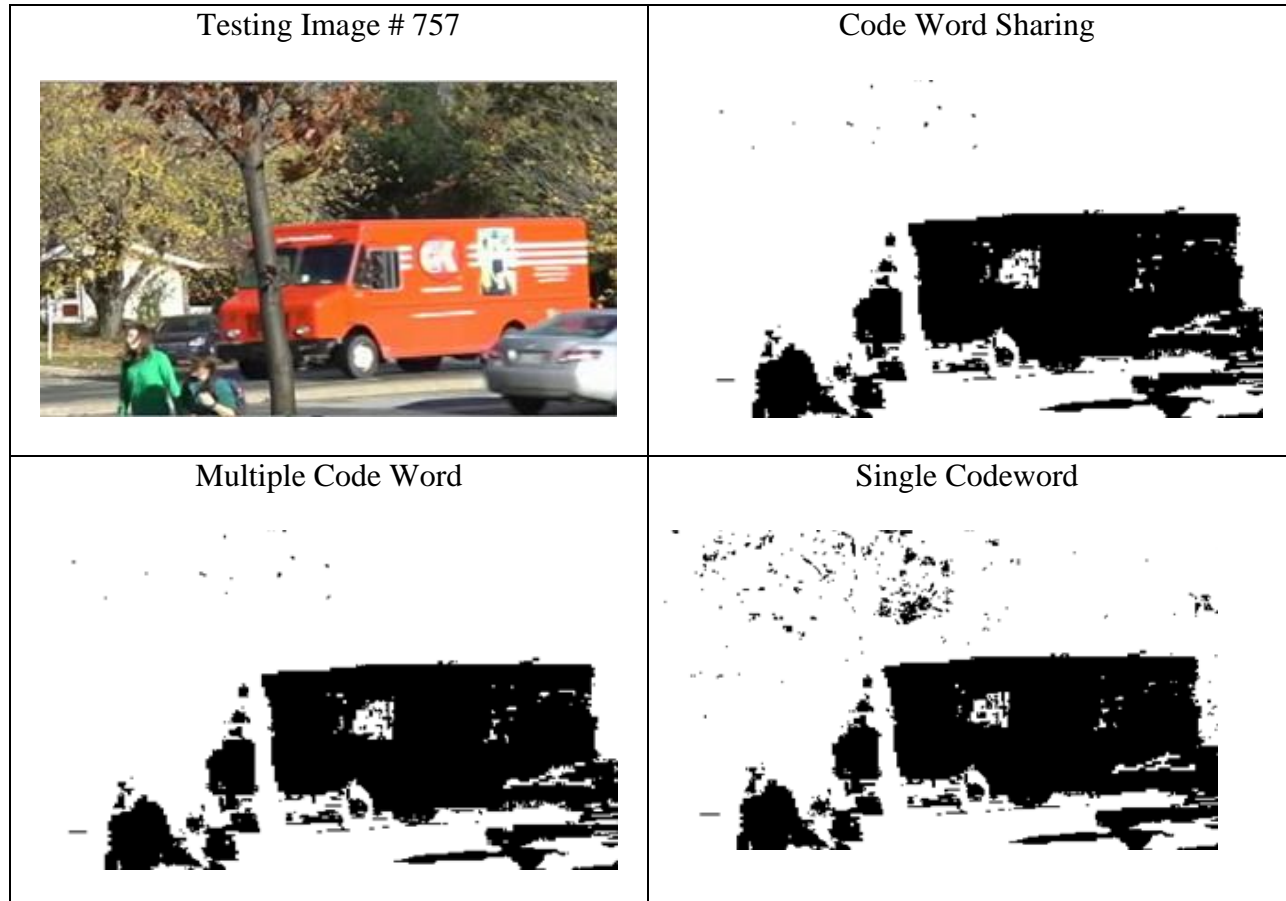


Figure 4.4 Output result by Code word Approach (set 1)

From the video above, we can see that the trees will be considered as foreground as it will move when it is blowing by wind. However, those trees should be considered as background as it is a dynamic background. It concludes that background model with single dominant codeword is inefficient in countering dynamic background scene. Therefore, the primitive model is replaced with the multiple dominant codewords model which efficiently reduced the occurrence of foreground noises by considering multiple dominant codewords. The refined systems successfully treat the moving trees as background with applied code word sharing model and the multiple code words model. The results generated by the refined system are significantly improved compared to the one generated by the primitive system. The false positives on foreground detection are significantly reduced.

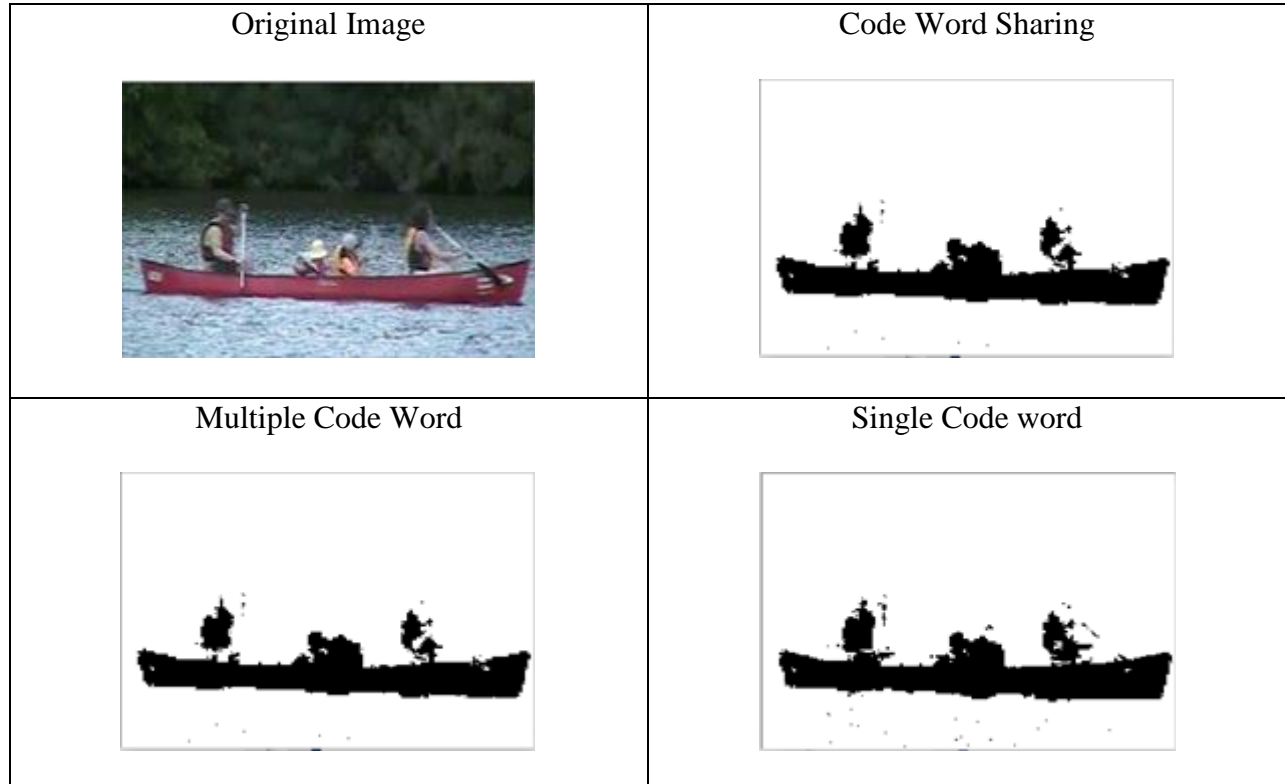
Set 2 : (canoe.avi)

Figure 4.5 Output result by Code word Approach (set 2)

Generally, it is normal for background scene to contain dynamic moving background objects and for the detection system to falsely recognize it as the true moving foreground objects. However, from the video above, we can see that the water ripple will be considered as dynamic background and it is considered as true detection. This is because the parameter of color vector and the number of threshold of codeword that have set is absolutely correct. Therefore, the false positives on foreground detection are greatly reduced. Next, the primitive model is replaced with the multiple dominant codewords model which efficiently reduced the occurrence of foreground noises by considering multiple dominant codewords. The results generated by the refined model are significantly improved compared to the one generated by the primitive model.

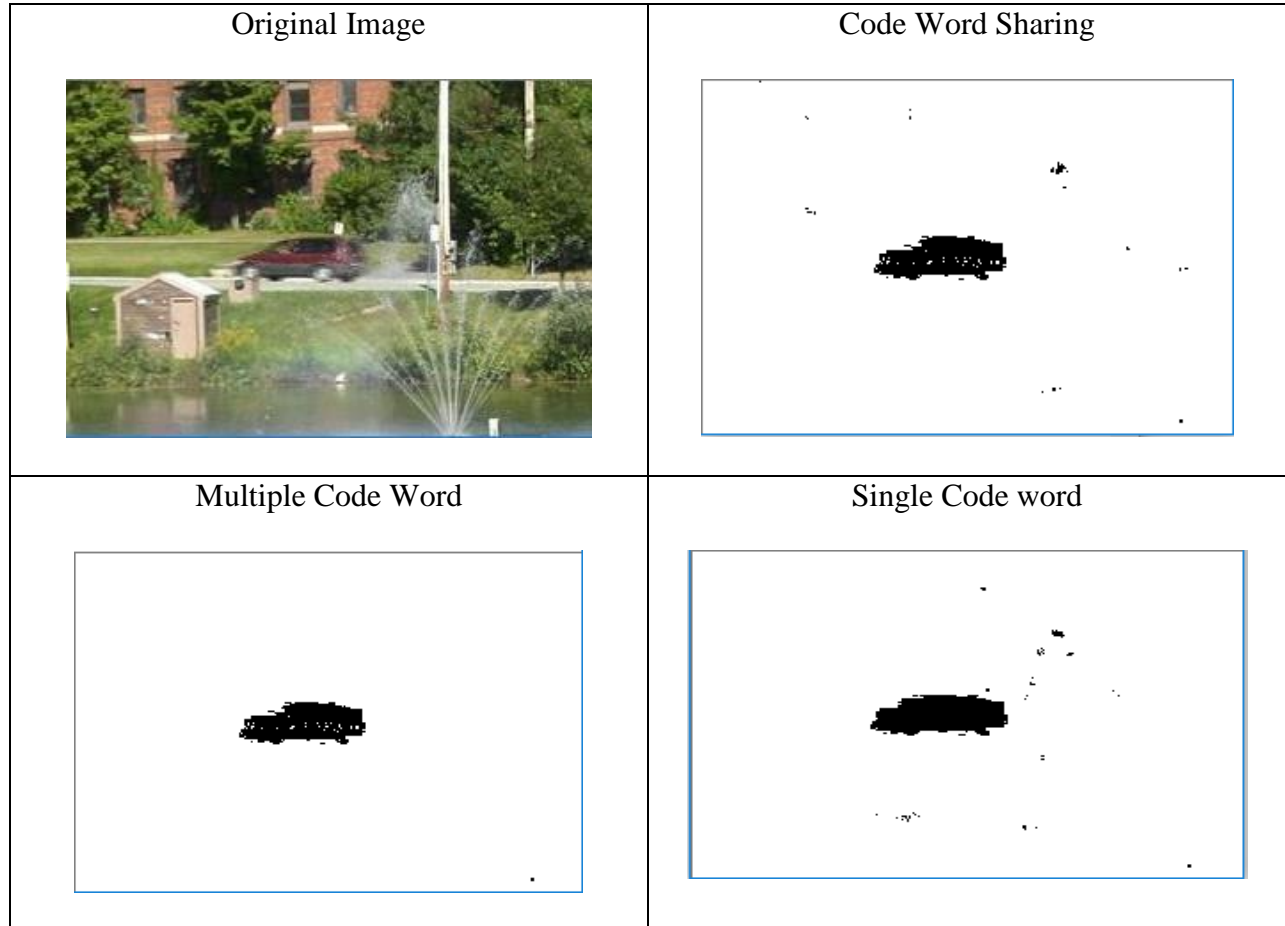
Set 3: (fountain02.avi)

Figure 4.6 Output result by Code word Approach (set 3)

From the video above, we can see that the water splashing will be considered as foreground as it is move. However, that water splashing should be considered as background as it is a dynamic background. It concludes that background model with single dominant codeword is inefficient in countering dynamic background scene. Therefore, the primitive model is replaced with the multiple dominant codewords model which efficiently reduced the occurrence of foreground noises by considering multiple dominant codewords. The refined systems successfully treat the water splashing as background with applied code word sharing model and the multiple code words model. The results generated by the refined system are significantly improved compared to the one generated by the primitive system. The false positives on foreground detection are significantly reduced.

4.2.4 Testing Result On Lighting Effect of Color Intensities

Set 1: (fall2.avi)

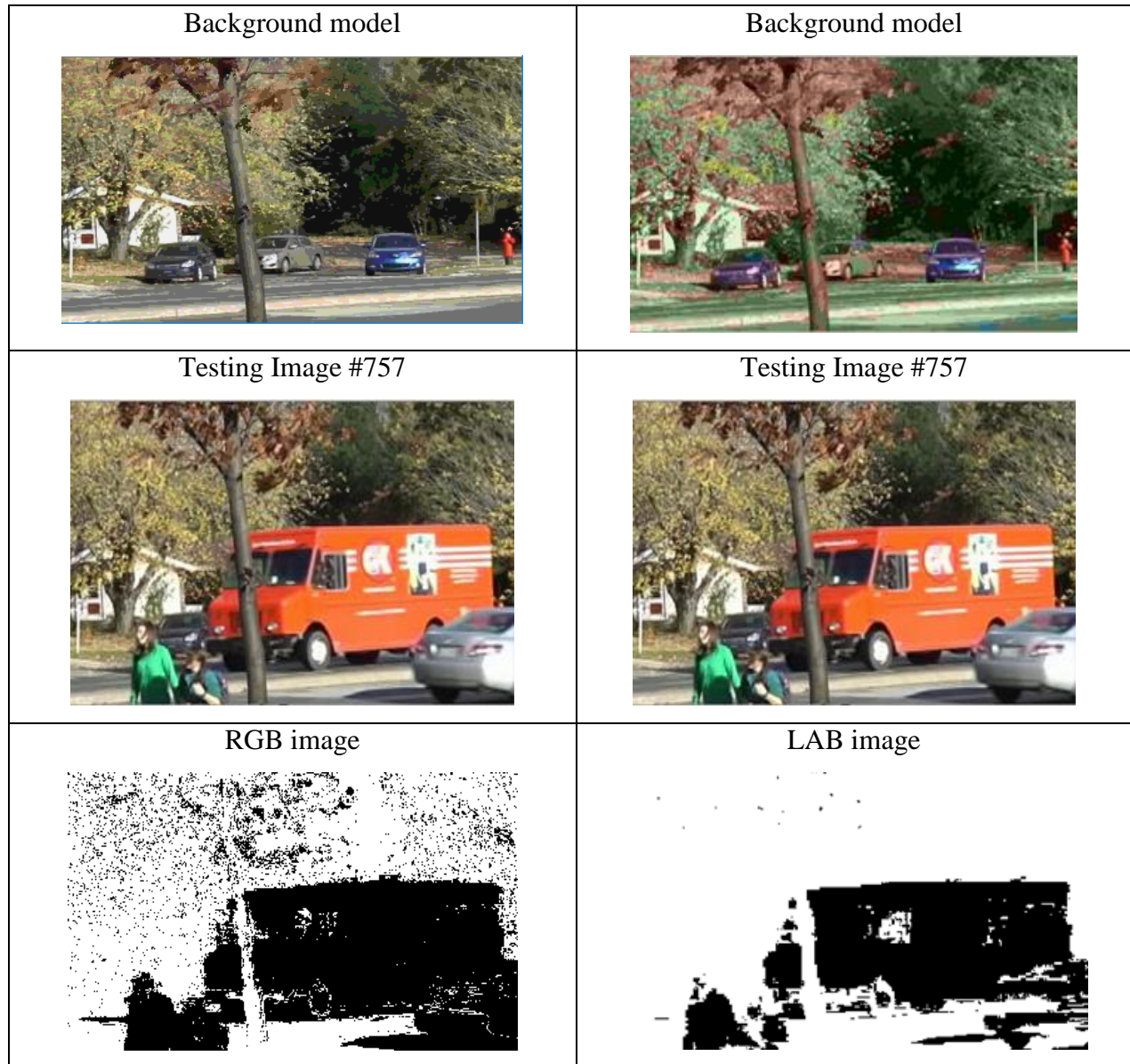


Figure 4.7 Output result on Lighting Effect of Color Intensities (set 1)

The video clip has background that contains the waving trees, will change the light intensity value which will lead to a lot of inaccurate classification and noise. The changes in light intensity of the waving tree affect the foreground mask result thus produce some noise at the background. As for observation, from the result, the LAB color space is more stable under the lighting change instead of RGB color space. The false positives on foreground detection are greatly reduced in LAB color space.

Set 2 : (canoe.avi)

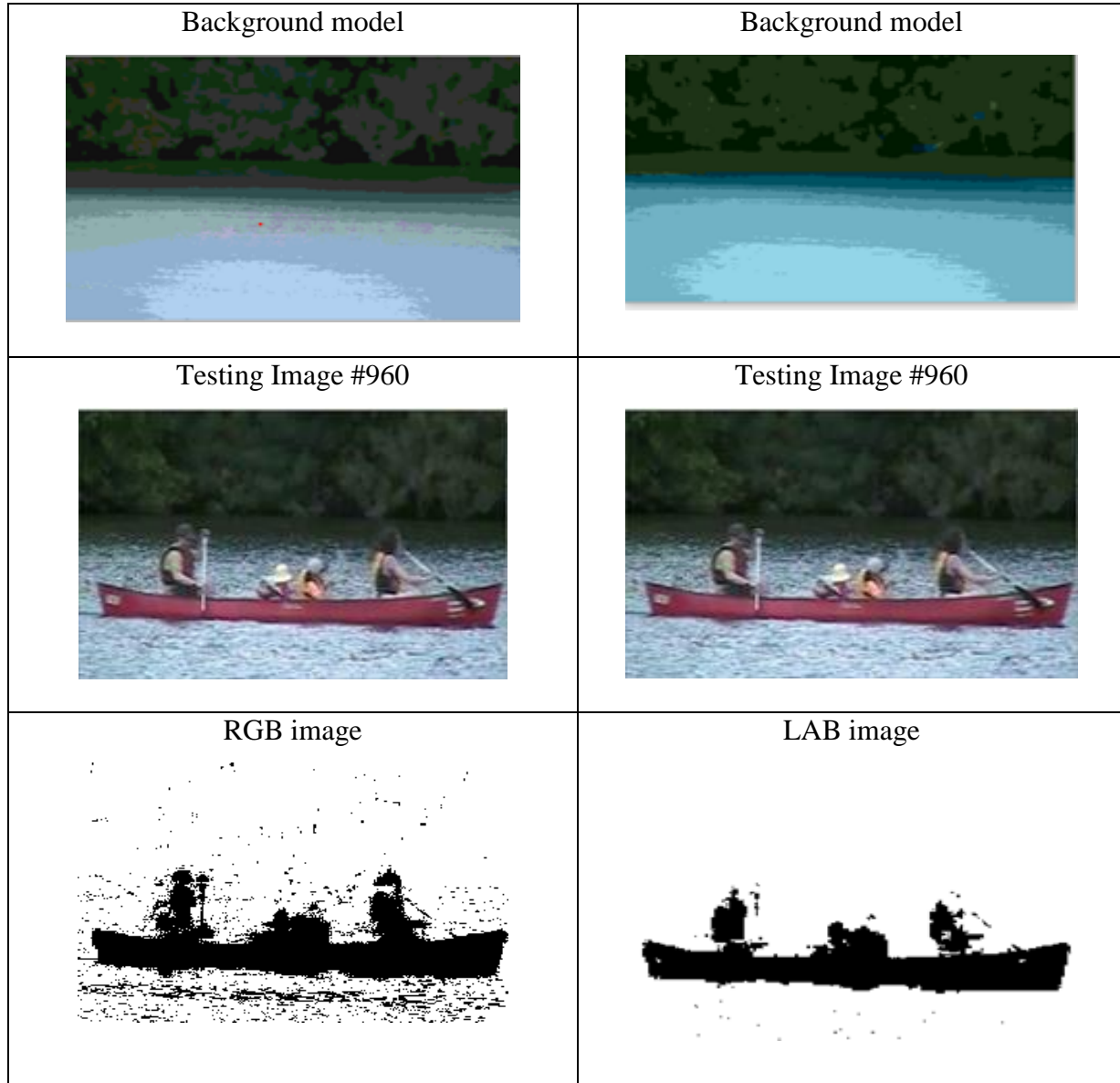


Figure 4.8 Output result on Lighting Effect of Color Intensities (set 2)

The video clip has background that contains the water ripples, will change the light intensity value which will lead to a lot of inaccurate classification and noise. The changes in light intensity of the water ripples affect the foreground mask result thus produce some noise at the background. As for observation, from the result, the LAB color space is more stable under the lighting change instead of RGB color space. The false positives on foreground detection are greatly reduced in LAB color space.

4.2.5 Testing Result on Enhancement of Foreground Mask Image

False-negative detection in foreground detection may increase because conditions that treat pixels as foreground become more stringent than ever. The system compensates for the rate of false negative detection and intersects with the effective suppression of foreground noise. From the resulting generated, as shown in the last section, foreground mask contain erosion and incomplete properties, and some may contain several fragments or broken pieces. Generally, to refine the resulting bitmap, the system applies a morphological transformation, which is dilation followed by erosion to extend the foreground components and join each of their fragments back to a complete component. However, this method is only able to apply on small fragment. In order to fill up the big broken pieces of foreground component, a flood fill method is chosen to apply.

The flood fill is a method for determining the region connected to a given point or node and used to fill the connected region with a different color. This method takes the most major three parameters which is a start node, a target color, and a replacement color. This algorithm keeps on checking for all nodes where the nodes that are connected to the start nodes likely with the target color and then change into replacement color. Then, this method will be applied into foreground mask in order to grow the broken piece of foreground back to a complete component.

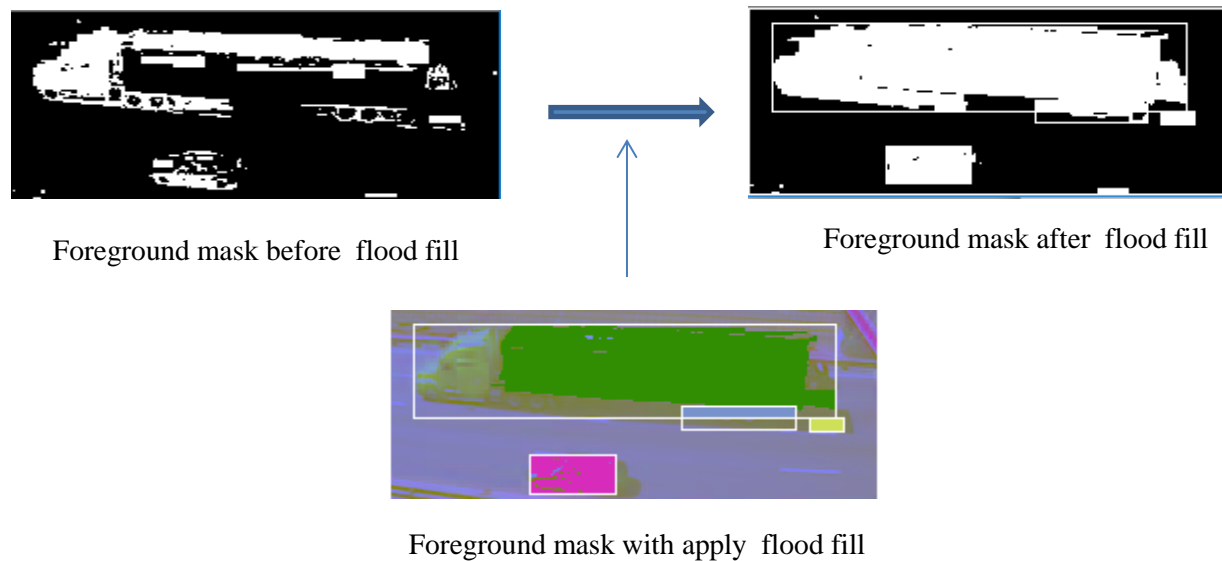


Figure 4.9 Output result on Enhancement of Foreground Mask Image

CHAPTER 5 EXPERIMENTAL RESULT AND ANALYSIS

5.1 INTRODUCTION

In this chapter, we are going to show a series of output generated by the improved system. The improved system were run on a machine equipped with processor Intel Core i5-7200U with clock speed 3.16 GHz, 4 GB of RAM, and Operating System of Windows 10.

In our training and testing phases, we had used different video data sets with different parameters such as total numbers of frame, frame per seconds (FPS), and video length and so on. Besides, some of these videos contain dynamic background. Below is the list of information regarding video data sets we had used:

- Video Name: WavingTrees.avi
 - Source: Provided by the lecturer Prof. Zen Chen
 - FPS: 30 frames/seconds
 - Length: 9 seconds
 - Video Size: 2.43MB
 - Frames size (width x height): 160 x120

- Video Name: parking.mp4
 - Source: Provided by the lecturer Prof. Zen Chen
 - FPS: 30 frames/seconds
 - Length: 3 minutes & 7 seconds
 - Size: 3.54 MB
 - Frames size (width x height): 320 x 240











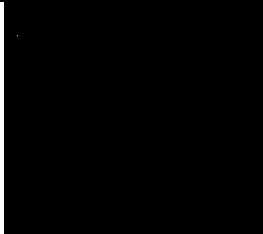

CHAPTER 5 EXPERIMENTAL RESULT AND ANALYSIS

- Video Name: highway.avi
 - Source: CDNET 2014 Data set
 - FPS: 30 frames
 - Length: 56 seconds
 - Video Size: 24.0MB
 - Frames size (width x height): 320 x 240

- Video Name: fauntain02.avi
 - Source: CDNET 2014 Data set
 - FPS: 25 frames/second
 - Length: 59 seconds
 - Video Size: 24.0 MB
 - Frames size (width x height): 432 x 288

- Video Name: fall2.avi
 - Source: CDNET 2014 Data set
 - FPS: 25 frames/second
 - Length: 1 minute
 - Video Size: 39.2 MB
 - Frames size (width x height): 480 x 320

5.2 Experimental results on CDNET 2014 Dynamic Background Data Set

Dynamic Background			
Video Name	Sample Result		
<p>fall02.avi</p> <p>training frame sequence : 1000-1500</p> <p>testing frame sequence: 500- 900</p> <p>Codeword coverage rate: 0.95</p> <p>Detection param: 100,10, 10</p> <p>FL param: 70,15,15</p>	 <p>Frame#500</p>	 <p>Frame#700</p>	 <p>Frame#900</p>
	 <p>Frame#500</p>	 <p>Frame#700</p>	 <p>Frame#900</p>
<p>fountain02.avi</p> <p>training frame sequence :1- 700</p> <p>testing frame sequence: 701- 1251</p> <p>codeword coverage rate: 0.95</p> <p>detection param: 100,10,10</p> <p>FL param: 70,15,15</p>	 <p>Frame#751</p>	 <p>Frame#1051</p>	 <p>Frame# 1251</p>
	 <p>Frame#751</p>	 <p>Frame#1051</p>	 <p>Frame# 1251</p>




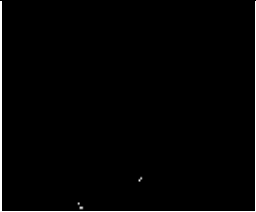


Dynamic Background			
Video Name	Sample Result		
WavingTrees.avi training frame sequence : 1-180 testing frame sequence: 181- 280 codeword coverage rate: 0.95 detection param: 100,10,10 FL param: 70,15,15	 <p>Frame#181</p>	 <p>Frame#251</p>	 <p>Frame# 271</p>
	 <p>Frame#181</p>	 <p>Frame#251</p>	 <p>Frame# 271</p>

Figure 5.0: Experimental results on CDNET 2014 Dynamic Background Data Set

5.3 Experimental results on CDNET 2014 Baseline Data Set



Baseline			
Video Name	Sample Result		
<p>highway.avi</p> <p>training frame sequence: 1-700</p> <p>testing frame sequence: 701- 1501</p> <p>codeword coverage rate: 0.9</p> <p>Detection param: 100,10,10</p> <p>FL param: 70,15,15</p>	 <p>Frame#701</p>	 <p>Frame#1101</p>	 <p>Frame#1401</p>
	 <p>Frame#701</p>	 <p>Frame#1101</p>	 <p>Frame#1401</p>
<p>parking.mp4</p> <p>training frame sequence : 1-2000</p> <p>testing frame sequence: 2001- 4001</p> <p>codeword coverage rate: 0.9</p> <p>Detection param: 100,10,10</p> <p>FL param: 70,15,15</p>	 <p>Frame#2601</p>	 <p>Frame#3001</p>	 <p>Frame#3701</p>
	 <p>Frame#2601</p>	 <p>Frame#3001</p>	 <p>Frame#3701</p>

Figure 5.1: Experimental results on CDNET 2014 Baseline Data Set

CHAPTER 6 CONCLUSION

As for the conclusion, in terms of foreground detection results and processing complexity, the completed system version brings some improvements compared to the original system of the previous project. For the test results, the completed system performed fairly well in detecting all common foreground moving objects from a dynamic background scene containing non-static background objects, with a lower false positive rate. As for system runtime performance, the background training phase has a similar runtime as the original system, and because of the deployment results enhancement tool, the test phase takes longer than the original system. In this project, it has applied these techniques to completed systems to improve the quality and output of the test results:

6.1 Highlight on any novelties and contributions the project has achieved

- 1) The background model is represented by codewords derived from the dominant quantized colors.
- 2) An image preprocessing technique is employed in order to solve the lighting change effect on the motion detection result by converting the RGB color space to the LAB color space.
- 3) An image post-process technique is performed to enhance the foreground mask result by filling up the broken moving objects for which the morphological operation finds hard to cope.

6.2 Summary of the research problems that have been considered and solved

With these techniques applied to the completed system, several motion and visual issues that could not be countered by the primitive system are now resolved:

- False detection on recognizing non-static background objects (waving tress and splashing water) is greatly reduced and the static background object able to be detected clearly.
- Inconsistency of variation between changing light and dark pixels are resolved by converting the RGB color space to the LAB color space.
- Foreground noises and broken pieces of foreground components are improved with applied the flood fill method which the morphological operation finds hard to cope.

6.3 Future Works

Due to the limited conducting time on research, some viable solutions could not discover and integrate them into the completed system. Therefore, the completed system is still susceptible to the following vision and motion issues. Therefore, it can consider as future work that needed for improvement.

- The shadow pixel from triple-value foreground need to be removed after flood filling operation
- To further evaluate the system with different variety of the video datasets
- Continue to discover different possibilities of parameters and further fine tuning the system preferences to seek for the optimal case for the system to perform the best.


REFERENCES

REFERENCES

1. Bouwmans, T., 2014. Traditional and recent approaches in background modeling for foreground detection: an overview. *Computer Science Review*, 11–12, pp.31–66.
2. Broida, T. J. and Chellappa, R. (1986). Estimation of object motion parameters from noisy images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(1):90–99.
3. Elgammal, A., Harwood, D., and Davis, L. S. (2000). Non-parametric model for background subtraction. In *Proceedings of the 6th European Conference on Computer Vision*.
4. Ianasi, C., Gui, V., Toma, C.I. and Pescaru, D., 2005. A fast algorithm for background tracking in video surveillance, using nonparametric kernel density estimation. *Facta universitatis-series: Electronics and Energetics*, 18(1), pp.127–144
5. Ilyas, A., Scuturici, M. and Miguët, S., 2009. Real time foreground-background segmentation using a modified codebook model. September 2009 IEEE, pp. 454–459.
6. Kim, Kyungnam, Chalidabhongse, T.H., Harwood, D. and Davis, L., 2004. Background modeling and subtraction by codebook construction. *2004 International Conference on Image Processing, 2004*, 291, pp.C237–C244.
7. Kim, K., Chalidabhongse, T.H., Harwood, D. and Davis, L., 2005. Real-time foreground–background segmentation using codebook model. *Real-Time Imaging*, 11(3), pp.172–185.
8. Kim, H. et al., 2007. Robust foreground extraction technique using gaussian family model and multiple thresholds. *Asian Conference on Computer Vision*. 2007 Springer, pp. 758–768.
9. Shapiro, L. G., & Stockman, G. C. (2001). *Motion from 2D Image Sequences*. Computer vision (pp. 254-255). Upper Saddle River, NJ: Prentice Hall.
10. Shimada, A., Arita, D. and Taniguchi, R., 2006. Dynamic control of adaptive mixture-of-gaussians background model. *2006 IEEE International Conference on Video and Signal Based Surveillance*. 2006 IEEE, pp. 5–5.
11. Stauffer, C. and Grimson, W. (1999). Adaptive Background Mixture Models for Real-time Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 246–252, Fort Collins, CO, USA.
12. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

REFERENCES

13. Tan, R., Huo, H., Qian, J. and Fang, T., 2006. Traffic video segmentation using adaptive-k gaussian mixture model. In: *Advances in Machine Vision, Image Processing, and Pattern Analysis*. Springer, pp. 125–134.
14. Tanizaki, H. (1987). Non-gaussian state-space modeling of nonstationary time series. *J. Amer. Statist. Assoc.*, 82:1032–1063.
15. Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and Practice of Background Maintenance. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, page 255, Los Alamitos, CA, USA. IEEE Computer Society
16. Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfunder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785.
17. Yadav, D. K. and Singh, K. (2015) ‘Object Detection in Dynamic Background for visual surveillance applications’, pp. 1601–1606.
18. Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38.
19. Zivkovic, Z. and van der Heijden, F., 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7), pp.773–780.



Universiti Tunku Abdul Rahman Campus, Perak
Faculty of Information, Communication and Technology

Kelvin Ong Chun Yauw 1506094

Moving Object Detection For Visual Surveillance System

Abstract

Video motion detection is the process of detecting a change in the position of an object relative to its surroundings in the video stream. Background subtraction is a most commonly used method to segment video streams into moving and background components. However, we need to handle problems such as lighting, shadows, dynamic background, and noise problems. Therefore, we will proposed a method to deal with the problems.

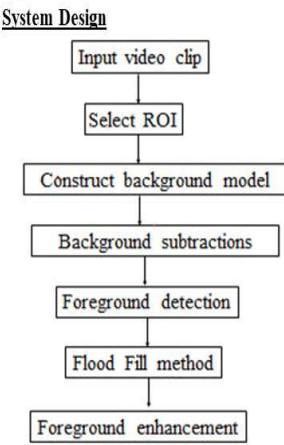
.Project Objective
To introduce an optimal solution for background model and background subtraction

To overcome challenges related to background modeling such as dynamic backgrounds, illumination changes and so forth.

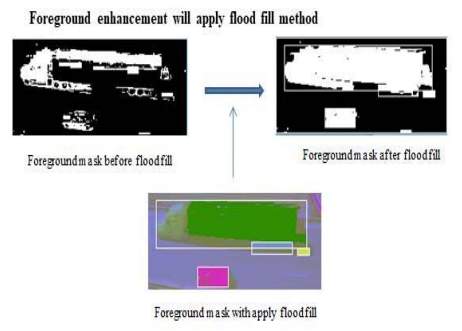
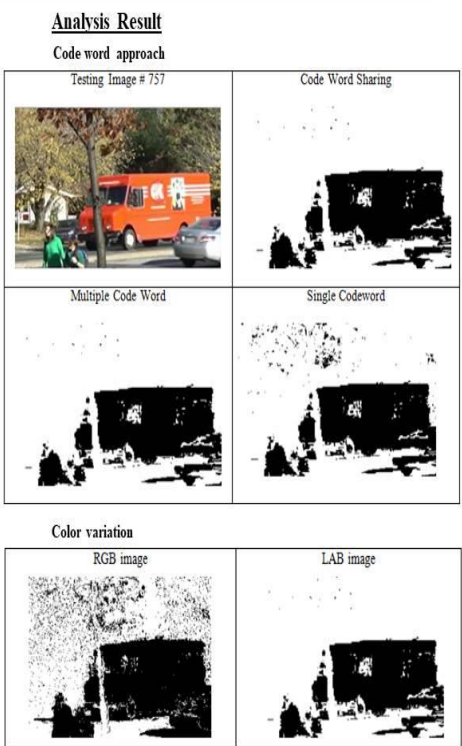
To reduce noises in background subtraction and enhance the foreground result..

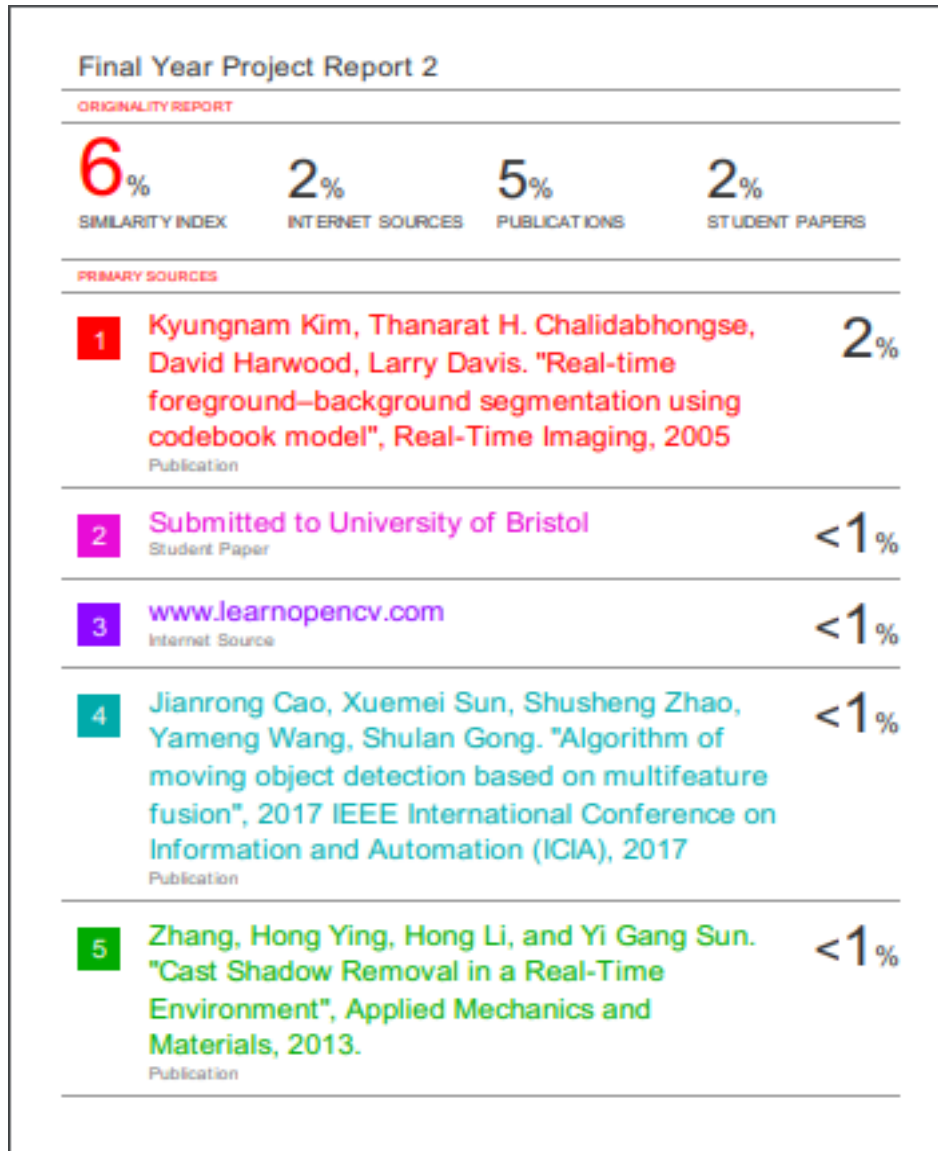
.Background of study
Codebook algorithm is the method that used in the system. Construct a background model with size of ROI frame selected and initialize Codebook in every pixel in background model. Read in all the training frame (with reduce to ROI) by looping through every single pixel and quantize its color value to store into a Codebook. There are many Codeword inside a Codebook, each of them is used to store a color value and the quantity of that color. Continue update the Codebook for every pixel when new training frame is read in. After stop training phase, assign 95% of dominant color (each dominant corresponding to a codeword) to every pixel in background model to generate the background. Reduce all testing frame to ROI frame. Read in all the testing frame by looping through every single pixel and check whether this pixel is belong to foreground or background. If that pixel belongs to foreground, set it to white color in foreground mask, if it belongs to background, set it to black color. Using flood fill method to grow the broken piece of foreground back to a complete component

- Technique Involved**
- Replaced single dominant color background model with multi-layered codebook background mode
 - Applies L*a*b color extraction measure to represent RGB color extraction
 - Background subtraction
 - Flood Fill method for foreground enhancement



Conclusion
We successfully developed a console application which can detect any moving object in either static background or dynamic background. Inconsistency of variation between changing light and dark pixels are resolved by using LAB color space. Foreground noises and broken pieces of foreground components are improved using flood fill transformation. However, there are still some of the problems that we can not solve are shadow removal and the foreground detected is not so nicely.





APPENDIX

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: _____ % Similarity by source Internet Sources: _____ % Publications: _____ % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: _____	
Parameters of originality required and limits approved by UTAR are as follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

 Signature of Supervisor
 Name: _____
 Date: _____

 Signature of Co-Supervisor
 Name: _____
 Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. <hr style="width: 80%; margin-left: 0;"/> (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. <hr style="width: 80%; margin-left: 0;"/> (Signature of Supervisor) Date:
---	---