# Traffic Monitoring Using Image Processing

A thesis presented in partial
fulfilment of the requirements
for the degree of
Master of Engineering
in Information and Telecommunications Engineering

at Massey University,
Palmerston North, New Zealand.

**Massey University**

Rui Li
2007

## Acknowledgement

I would like to take this opportunity to thank those people who have guided and supported me in completing this thesis in last one and half years.

First of all, I would like to thank my supervisor Dr. Xiang Gui for his excellent supervision, supports and guidance. I am also grateful to my co-supervisor Dr. Ruili Wang for his supports and comments. Without their help, this thesis can not go this far.

Thanks to those staff in the Institute of Information Science and Technology (IIST) who gave me suggestions and helps. Thanks to all my postgraduate friends in IIST, Yiming, Frank and Mathew for their support.

Many thanks to my best friend Yiyi Tang who always gives me support, suggestion, and share his research experience with me. I thank my girlfriend Xuan Zhang for her support and inspiration, especially filled me with confidence when I lacked it.

Finally, I would like to thank my parents and my sister. I am grateful for their constant love and support, all the way across the Pacific Ocean.

# Traffic Monitoring Using Image Processing


## Abstract

Traffic monitoring involves the collection of data describing the characteristics of vehicles and their movements. Such data may be used for automatic tolls, congestion and incident detection, law enforcement, and road capacity planning etc. With the recent advances in Computer Vision technology, videos can be analysed automatically and relevant information can be extracted for particular applications. Automatic surveillance using video cameras with image processing technique is becoming a powerful and useful technology for traffic monitoring.

In this research project, a video image processing system that has the potential to be developed for real-time application is developed for traffic monitoring including vehicle tracking, counting, and classification. A heuristic approach is applied in developing this system. The system is divided into several parts, and several different functional components have been built and tested using some traffic video sequences. Evaluations are carried out to show that this system is robust and can be developed towards real-time applications.

# Contents

# List of Figures

# List of tables

# 1. Introduction

Traffic monitoring systems that are used for traffic management and planning have a long history. In early stage, people designed devices such as inductive loop detectors [1] or microwave detectors [2] to collect traffic data. With recent advances in computer vision and video technology, traffic monitoring using image processing becomes popular.

Nowadays, people have to deal with worsening traffic conditions in urban areas or even in suburb areas due to the fast development of cities. Long time ago, people tried to add more lanes or built motorways around the city centre to avoid congestion. Nevertheless, it becomes less and less possible to build more roads. Using the existing road and rail network more efficiently is the only solution. In order to achieve this goal, acquiring better and more accurate traffic data becomes more and more important.

Advances in electronics make powerful hardware available for acquisition and processing of images and videos at low costs. Automatic surveillance using video cameras with image processing technique is a powerful and useful technique for traffic monitoring. Traffic monitoring systems can be used in pavement analysis and design by measuring traffic volume and environmental conditions and evaluating system performance. It can be used in road design modification by classifying vehicles, recording traffic flows, detecting and predicting accidents. It can be used to support freeway or motorway management by analysing traffic volumes and tracking vehicles. Other applications include defining control decisions, providing travelling information for publics, supporting operations functions, planning special event and emergency management.

In this thesis, we propose a system that analyses image sequences captured by a camera, which can be mounted on a bridge over a road or installed near a traffic light to detect running vehicles, and obtain further traffic information for monitoring use. This system is robust and a real-time application can be developed based on the system developed.

## 1.1 Motivation

The work presented in this thesis has the potential to be immediately applied in the area of traffic monitoring using image processing. The developed algorithm can be incorporated with powerful hardware such as embedded system: Field Programmable Gate Array (FPGA), Digital Signal Processor (DSP) to make real-time monitoring come true. The required hardware involves a static camera mounted on a bridge or installed near a traffic light, a set of data transmission cables to transmit collected data, a FPGA or DSP board that is connected to a PC to process data, and the PC can be used to control and perform monitoring functions. If received data can be processed quickly enough to satisfy real-time requirements, the key frames showing any accident or danger need to be written out and stored. In this way the storage space for the whole video sequence can be reduced.

The main tasks of this system are to detect moving objects including pedestrians, bicyclers, motorcyclers, and vehicles, vehicle tracking, vehicle counting, and vehicle classification. The tracking function can track vehicles, and record the trajectories travelled by vehicles, which can be further used to calculate the speeds, accelerations of the vehicles. These can be used in speeding control. Through tracking, road hazards like unusual reversing or turning can be detected, and warning can be given out by writing out an image showing that scene. When these happen, abnormal trajectories will generate, the system can justify through analysis of the trajectories. For example, if one vehicle is trying to turn right at a corner that is forbidden to, the curvature of its trajectory will be greater than the predefined threshold, it is described as a dangerous vehicle.

The work can be extended to automatic security and surveillance use on/in buildings or on roads. The camera can be installed on the wall outside a building or at the entrance of a building to monitor the entrance of the building. The human bodies are detected and recognized by a computer, and these blobs can be tracked and recorded. Through further development, functions such as recognising different people and their actions can be added in. Action recognition is a key role in active surveillance. The algorithm can be further developed towards human computer interaction. A

camera is used to monitor and capture actions from the user, the computer system can analyse actions and give response.

## 1.2 Problem description

One stationary camera, mounted on a bridge over a road is used to monitor the whole road. The resolution of these images is 320 x 240 pixels due to the limitation of real-time. The original video sequences are in uncompressed RGB format, then a piece of software called hero (http://www.kingsoft.com) is used to sample the images from the video sequence at a rate of 20 frames per second. There are 8 bits to represent each colour component, which gives 256 steps of intensities. Then these images will be stored in a hard disk for further processing. Figure 1.1 shows one example traffic image. In our thesis, in order to develop and test our algorithm, we capture traffic videos from several different scenes, and some of them focus on pedestrians and bicyclers. Figure 1.2 shows other traffic images.


Figure 1.1 An example traffic image

(A) (B)

Figure 1.2 Two other traffic scenes (A) a vehicle is turning left at the corner (B) a vehicle at a corner that is merging to the main road.

There are many conditions to be considered in developing a traffic monitoring system. In image sequence capturing stage, the weather, lighting conditions of the outdoor scene, the position of the camera can be diverse. These would result in images of different quality. Environmental conditions are also considered, for example, some parts of the vehicles will be blocked by tress or buildings near the road, or even totally merged by them. The captured images contain a lot of information, such as vehicles, trees, pedestrians, traffic lights and so on. After detection, it is necessary to identify which objects are expected to be kept and which should be ignored. Waiting or stopping vehicles are also required to be kept monitoring instead of treating them as static objects.

In order to make real-time application possible, the algorithm is developed based on some assumptions to make the algorithm simple but robust. For example, in the background updating process, we assume that the monitoring area is small enough that the intensity change is the same for the whole area. Hence we only select four test areas for intensity change checking. In some situations, captured traffic frames could not satisfy the assumptions we made. For example, if cloud comes to cover only the area of testing areas, the updating still performs and makes the rest area darker than the actual situation. Although we make some changes in our algorithm to make the algorithm robust enough, some situations are still not solvable. For example the length and the shapes of shadows are changed by the movement of cloud. These changes affect the results of shadow handler. However, the developed algorithm is still good enough for real-time application.

After monitoring, the processed images will be added together to form an avi format video and play back. The flow chart of traffic monitoring using image processing in our research can be viewed as follow:

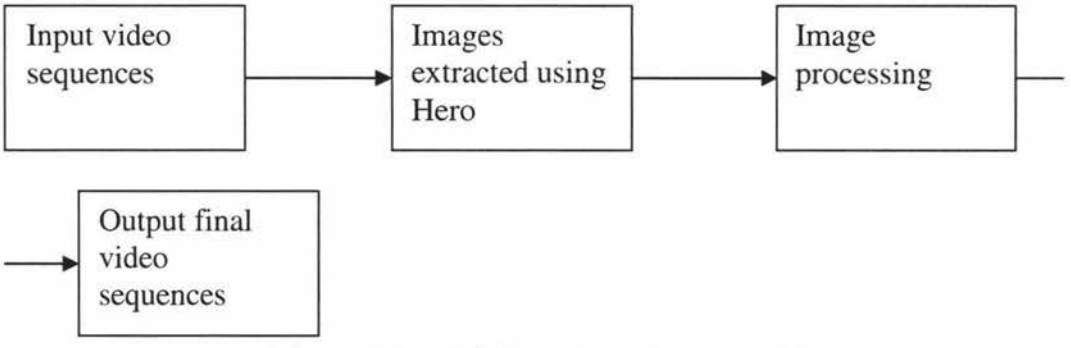



Figure 1.3 Brief flow chart of the algorithm

## 1.3 Block diagram of the algorithm

In this section we present the processes we used during the development of this thesis. Figure 1.1 shows the flow chart diagram of the processes.

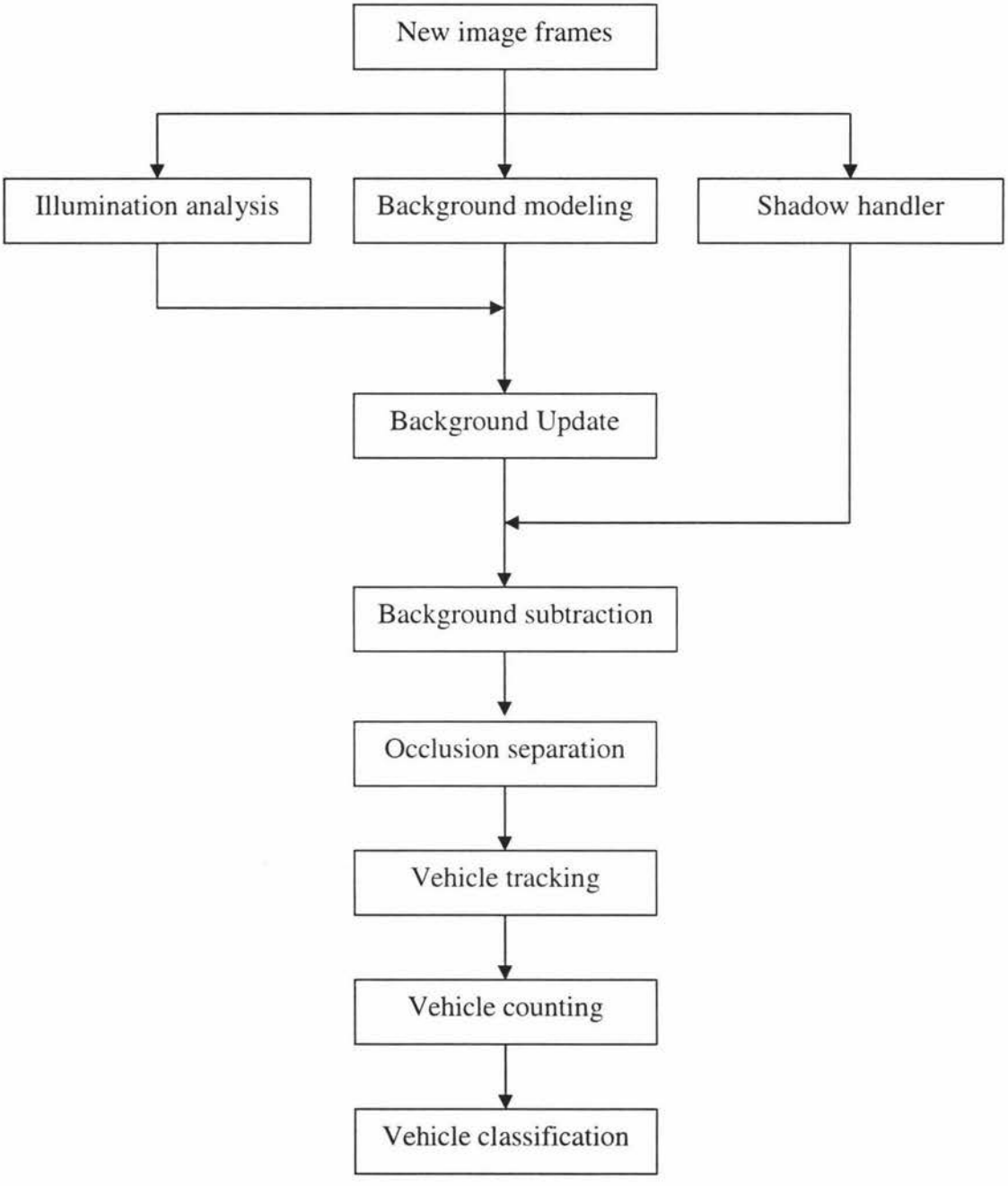



Figure 1.4 Processes of traffic monitoring

New video sequences that are captured by a camera are input to a PC. Then a piece of software called hero (or other software that can sample video sequences) is used to

get the image samples of the video sequences at a frame rate of 20 frames per second. These images sequences will be stored in a hard disc and then be input to the system.

The first step of the algorithm is to model and update the background image for background subtraction. The first 200 input image sequences (first 10 seconds of traffic scene) are used to model the initial background image using the averaging techniques. At the same time, each image is processed with illumination analysis. Illumination analysis is used to get the direction of the light source such as sunshine. This information is used define the direction of the shadow. Shadow handler is used to remove shadows in image sequences. Then these processed images are used to update the modelled background.

In the background updating stage, we use next 200 images to classify each pixel in each frame into either static background pixel array or dynamic background pixel array by comparing incoming images with the modelled averaged background image. Once all pixels are separated, averaging background updating is used on static background when necessary. At the same time, current dynamic background pixels are used to replace the original pixels at the corresponding places in the averaged background image if they are not covered by foreground objects.

Once an accurate background is built, this background image is subtracted from the current traffic image to obtain foreground objects. Through some other techniques such as edge detection, opening, or closing, only foreground objects including pedestrians, bicyclers, motorcyclers, and vehicle stay in the image. Rough contours of those objects can be extracted. Followed by the snake method [85 - 89], refined contours representing objects are constructed for tracking.

Our tracking algorithm is the combination of region-based tracking and contour-based tracking. Through energy minimization method (snake), we get accurate contours that bound detected objects. Then these contours are used to obtain the corresponding regions for detected objects. Region-based tracking is used to track all objects.

Before tracking, occlusion separation is used to separate two or more vehicles that are partially or totally occluded with each other. Tracking is separated into fast moving objects (vehicles and motorcycles) tracking and slow moving objects (pedestrians and bicyclers) by travelling speed. The tracking algorithm for these objects is the same, apart from the tracking frequency. Tracking is performed by recording the coordinates of the bounding boxes and the centroids of blobs that representing found objects. Histogram is used to separate two or more close vehicles. Objects are first classified into fast moving objects and slow moving objects by speed. For slow moving objects, single pedestrian and bicycler are defined as one category. Multiple humans and slow moving or stopping cars are classified by using the compactness of them.

## 1.4 Contributions

The following points are the major contributions from this research.

1. A complete traffic monitoring system that combines a set of effective and efficient methods, which has the potential to be developed to fulfil real-time processing, has been developed. Experimental results show that this algorithm is robust and effective.

2. An improved method that divides the background into static background and dynamic background for updating separately has been implemented. This method can provide accurate updating of background image with less computational load.

3. A modified approach that combines the benefits of region-based tracking and contour-based tracking to increase the efficiency and accuracy of vehicle tracking has been developed.

## 1.5 Functional overview of the system

The system has the following functions:

**a. Detection of vehicles**

Detecting all moving objects in the scene, including pedestrians, cyclists, and vehicles. Recognizing and separating vehicles from them according to the properties of vehicles such as speed, volume.

**b. Track vehicles in the scene and extract traffic data**

When a vehicle enters the monitoring area, its position (coordinates in the window) is recorded. The coordinates used in the tracking system are a relative set of coordinates. Tracking is conducted by recoding the change of coordinates according to some stationary objects within a static scene. Some other features such as histogram are used to distinguish closed vehicles. The coordinates of bounding box and centroid are recorded. Some functions like shadow detection and elimination, occlusion separation are used to reduce tracking errors. Stopped vehicles are required to be recorded, and be tracked again when they start to move.

Associated with tracking, traffic data such as velocity, acceleration, trajectories are recorded. Through the analysis of trajectory, any incorrect entries at corners or intersections are recorded and output. Through the speed measure, any speeding behaviours will be detected.

**c. Count vehicles in the monitoring area**

When a new vehicle enters the testing area, its coordinates will be recorded and the system treats it as a new vehicle and counts it. When it leaves the testing area, the system will automatically terminate tracking

**d. Classify vehicles**

All the detected and tracked vehicles will be classified into two different categories according to their properties like the length, width and compactness.

## 1.6 Scope of this thesis

This thesis is organised as follows:

**Chapter 2** presents a literature review in the areas related to our work including background modelling and updating, vehicle detection, shadow detection and elimination, occlusion separation, vehicle tracking, vehicle counting, vehicle classification, and segmentation. Also some materials related to our work such as colour space are discussed.

**Chapter 3** describes the algorithm we used to initialize, model and update the background. After the background subtraction, foreground objects are extracted and presented by contours. A snake operation is used to refine contours so that not much background information is included in the contours.

**Chapter 4** describes illumination analysis followed by shadow handler in our algorithm.

**Chapter 5** describes the tracking methods we developed in our algorithm. First, occlusion separation method is introduced. Through tracking, trajectories of vehicles are recorded and vehicles' information is obtained. These data will be further used to monitor traffic flow. Through tracking, vehicle counting can reach an accurate result. Vehicle classification is also discussed.

**Chapter 6** shows the results from the testing of the system using several different traffic videos. We use both quantitative and qualitative evaluations on the results. A discussion is also included.

**Chapter 7** draws the conclusion of this thesis. In this chapter, main findings and conclusions are presented, followed by a discussion of the future directions about our work.

# 2. Literature review

## 2.1 Introduction

Our research is in the area of computer vision. Related topics are background modelling and updating, image segmentation, vehicle detection, vehicle tracking, vehicle counting, and vehicle classification. We will present a literature review of these topics in the following sections.

## 2.2 Requirements for traffic monitoring system

An effective traffic monitoring system should be able to detect all kinds of objects such as trucks, buses, cars, motor cycles and pedestrian automatically. These detected objects are then be tracked. Through tracking, vehicle data can be extracted such as velocity, acceleration, and so on. The trajectories of vehicles are also recorded. These trajectories can be used in traffic control. The number of total cars passing through a road in a certain period should be counted. Through a classification process, they should be able to be classified into different groups, and the required information could be collected and recorded.

The system should be able to work in different weather and lighting conditions including sunny, rainy, cloudy, snowy, and at night. It should work at some situation with significant change in lighting such as during the sun-rise and sun-set. The system can work under any traffic conditions as free flowing, congestion, slow flowing. For example, stopping vehicles should not be recognised as a part of background, they should be tracked before they stop, and tracked again as they move. The last one and the most important one is that the system should work in real-time.

We are trying to meet all requirements mentioned above to make our algorithm robust enough and be suitable for real-time application. The algorithm can be divided into many steps, and each of them can solve a particular problem and prepare for the next step.

## 2.3 Early traffic monitoring systems

Traffic data acquisition can be conducted manually for a short period of time such as counting vehicles by staffs at a corner of a street or doing survey from pedestrians and drivers. However, for control or planning purpose, automatic techniques that can monitor the road continually without human operations have to be used. Automatic techniques using traffic sensors can be classified into intrusive such as inductive loop, and non-intrusive such as microwave, acoustic, radar, video, and infrared.

Before the use of computer vision in traffic monitoring, inductive loop detector [1] is used to collect traffic data and classify vehicles. This technique is still used in New Zealand now. Workers often installed two parallel lines on the road, when a vehicle passes through, the inductance of the inductive loop will change, and then the system will count the vehicle. This system is relatively easy to install and maintain.

Another popular approach is to use Microwave Vehicle Detector (MVD) [2]. Microwave energy is often in 10 GHz frequency range, MVDs send out a narrow beam of microwave energy and detect the change in frequency of reflected back radiation. Based on the Doppler principle, when a vehicle moves into a testing area, the energy will be reflected back at a little different frequency. Then this reflected back energy will be combined with a small part of the transmitted energy to construct a new signal at the different frequency. The detection of a vehicle can be made by this frequency change.

The main drawback of these two methods is the expense to install and maintain equipments. The other point is that using these two methods, stationary or slow vehicles may not be detected or would be missed.

## 2.4 Traffic monitoring using image processing

Video-based vehicle detection for capturing and processing traffic data, which is sometimes referred as a virtual loop [3], is simple, convenient, and can be updated quickly. Video-based traffic monitoring [3, 6, 7, 35, 51] becomes more and more popular nowadays.

Video-based traffic monitoring system is inexpensive when compared to intrusive traffic sensor. It can be installed in 15 minutes without causing traffic jam. Maintenance cost of video equipment is also low. Remote control through cables or wireless devices is possible and convenient. People can put any number of these virtual loops on the target roadways. The collected data can be backup or double-checked, which is beneficial for model development. They can provide wide range monitoring and data collection, which allow the analysis of traffic flow including vehicle tracking, vehicle counting, and vehicle classification. With the advanced hardware, real-time application can come true.

On the other hand, video-based techniques require considerable pre-pay costs for purchasing and installing equipment, and staff training. Video system itself is complex and changing rapidly. To reduce total costs, the selection of hardware for real-time application should be careful. However, the total cost for these types of systems are still low and acceptable to government and companies.

## 2.5 Vehicle detection

Detecting the existing vehicles on road is the first step of the traffic monitoring. The system should detect vehicles, pedestrians, and bicyclers only and ignore other unwanted objects.

### 2.5.1 Thresholding

The simplest approach in vehicle detection is thresholding [7]. Vehicles are compact objects that have different intensity from their background. If the right threshold level is selected, vehicles can be separated from their background through a threshold process. However, this technique is less effective, because it depends on the choice of threshold level. If the intensity of vehicles is similar to the background, false detection that classifies foreground pixels as background pixels and missed detection cannot be avoided. Adaptive thresholding and gray-level morphological operators are often used to improve this method.

### 2.5.2 Edge-based detection

The edge-features of objects can be used to detect vehicles. By applying morphological edge-detection schemes, vehicles are highlighted as complex groups of edges, while background has less edge content. Vehicles can be then represented by high edge complexity within the road area, which can be quantified through analysis of the histogram [3]. Edge-based vehicle detection is more effective than other background subtraction or thresholding approaches, because in diverse ambient lighting situations, the edge information is still significant [4]. In practice, the conventional gradient-based edge detection operations and morphological edge detectors are widely used [5]. The main drawback of edge-based detection is when the monitoring area is large, edges of vehicles will be too small to recognise.

### 2.5.3 Feature aggregation

Feature aggregation is another useful approach. First, the characteristic points on vehicles are detected, which are often the corners of the vehicles, and then those points will be aggregated regarding the vehicles' geometrical characteristics. These points are used to find out the vehicles. Feature aggregation can reduce the false detection rates and increase the detection robustness and reliability [7]. The main drawback of this approach is that the efficiency of detection and tracking performance depends on the number of feature points [6].

### 2.5.4 Optical flow

Utilizing optical flow is another approach to detect objects. A rigid object's appearance changes slightly during motion, while it changes dramatically at the regions where it moves in and/or moves out of the background. These changes can be used in calculating optical flow.

If we use $g(x,t)$ to stand the gray value recorded at the location $x$ at time $t$, the gray value at point $x-u\Delta t$ at time $t-\Delta t$ can be written as $g(x-u\Delta t, t-\Delta t)$. The optical

flow field $u(x,t)$ could be computed by mapping $g(x-u\Delta t, t-\Delta t)$ to $g(x,t)$. The optical flow field encodes the temporal displacement of the gray-scale structures within an image sequence. The information it contains is not only about the relative displacement of pixels, but also about the spatial structure of the scene [7]. This method is effective on small moving objects, but the main drawback is that it consumes much time on calculating the optical flow, and the inner points of a large homogeneous object cannot be featured with the optical flow.

There are many approaches have been proposed for estimating the optical flow field including gradient-based [8] [12], Feature-based [9] [13], correlation-based [10], multigrid method [11].

### 2.5.5 Background subtraction

The background subtraction approach is a classic technique for vehicle detection. The main idea is to subtract the background image from the current image so that the foreground objects can be detected. Through a threshold operation, only vehicles stay in the image. In practice, the effectiveness of this method depends on the accuracy of the background image, the updating of background image, and the selection of a suitable threshold value.

The background image is often modelled manually, for example taking a series of photos without vehicles, or being detected in real-time by forming a mathematical or exponential average of successive images [15]. Road background changes according to the time and weather. For example intensity changes significantly during sunrise and sunset, or when whether changes. Shadows cast by buildings and clouds also cause the background to change dramatically. If the background could not present the background correctly, false positive pixels and false negative pixels might occur. False positive pixels are defined as pixels that have been segmented as foreground, but actually correspond to background. False negative pixels are the pixels that correspond to the object, but have been segmented as background.

Zhang and Keltte [14] join the mixture of Gaussian (MOG) model with concepts defined by region level and frame level considerations to improve the detection of

moving objects. MOG method is robust and stable because multiple Gaussian distributions can be tracked simultaneously and a density function for each pixel can also be maintained. Multi-modal background distributions can be handled well by this approach. In their approach, foreground objects are extracted by using three processes called pixel process, frame process, and region process. Shadows are removed at the last step to finish the estimation of background image.

Fujiwara et al. [15] propose a method that uses the distribution of image vectors to obtain the sequential changes in the background. Then the Mahalanobis distances between the averages of image vectors and current image vectors is estimated to do the subtraction. "Mahalanobis distance is a distance measure based on correlations between variables by which different patterns can be identifid and analysed. It is often used to find the similarity of an unknown sample set to a known one" [15]. This method is more effective for real-world scenes.

### 2.5.6 Other approaches of vehicle detection

Beucher et al. [16] propose an approach for detecting vehicles using the gray values of traffic images. The gray value of road is detected and defined first. Vehicles are defined as regions with constant gray values. If any regions with constant gray values that are different from the one of the road are found in the region of interest, they are assumed as vehicles. This system detects vehicles well. However, due to its high computational load it is not suitable for real-time application.

Wan et al. [17] propose a multigrid identification of regions of interest method. Firstly, a hierarchy of traffic images at different resolutions is produced. Then a search begins from coarse to fine. Vehicles can be found because compact objects are different from background in low resolution while noise and small objects are likely to vanish at those levels.

Wavelet transform can be used in detecting moving objects. Subramaniam et al. [18] use a method that combines the information from Gabor and Mallat wavelet transforms to enhance the speed and accuracy of detection. Small, disconnected, and even openworked objects can be detected by this method. Through the Gabor-wavelet

analysis, a fast evaluation of image flow vectors with low spatial resolution can be obtained, and a histogram over the image flow field can also be produced. By using the Mallar-wavelet transform, the accuracy of object detection is increased.

## 2.6 Background modelling

Background modelling is very important to background subtraction approaches. On one hand, a good background should be robust against environmental changes and be sensitive enough to identify all moving objects. It should avoid detecting non-stationary background objects or moving shadows as moving objects, and it should react to any changes in background. On the other hand, background itself should require less resource for initialization.

As we are using a stationary camera to monitor the road, the background changes very slowly compared with the moving vehicles. When subtraction is employed, foreground objects are detected. However, background image will change significantly when illumination alters or new vehicles enter the monitoring area. Hence in some situation, the background image has to be re-modelled, or kept being updated to ensure it is as similar as possible to the current image.

In early stage, simple temporal and/or spatial smoothing technique such as Gaussian low pass filter is used to reduce the camera noise. Koller et al. [20] combine intensity values and spatial derivatives to form a single state space to track background with the Kalman filter. However, adding colour information and derived features increases the complexity for parameter estimations and also would bring problems in background maintenance [21].

The simplest way to create a background image for a scene is to average a selected image sequence without vehicles running or with few vehicles. This process is also called background initialization. However, sometimes it is difficult to find a moment with few vehicles running, some modifications and updating have to be incorporated into this method to increase the accuracy of the background image.

Background modelling can be classified into two categories, namely non-recursive and recursive [21]. Non-recursive approaches do not rely on the history of frames stored in the buffer hence it is more adaptive than recursive approaches. On the other hand, recursive approaches update the current background model according to each input frame recursively.

### 2.6.1 Non-recursive background modelling

The simplest non-recursive background modelling method is frame differencing [21]. The video frames at time $t$-1 is used as the background model for video frame at time $t$. It is difficult to detect slow moving objects, and when the colour of the object is similar to the background, it is difficult to distinguish them.

Another approach is to use filter in background modelling, such as median filter and linear predictive filter. In ref. [22], at each pixel location, the median of all the frames in the buffer can be used to define the background. In ref. [23], a linear predictive filter is used to estimate the current background in the buffer. These techniques use a single background estimate at each pixel location.

In Non-parametric model [24] the entire history $I_{t-L}$, $I_{t-L+1}$... $I_{t-1}$ is used to form a non-parametric estimate of the pixel density function $f(I_t = u)$:

$$f(I_t = u) = \frac{1}{L} \sum_{i=t-L}^{t-1} K(u - I_i) \qquad (2.1)$$

Here $K$ (.) is the kernel estimator that is chosen to be Gaussian. $I_t$ is the intensity of pixel, and $L$ is the number of video frames that are stored in the buffer. If $f(I_t)$ is smaller than some preset threshold, which means the current pixel $I_t$ is unlikely comes from the background distribution, can be defined as a foreground pixel. Using this method, multi-modal background distribution such as pixels from a swinging tree can be solved.

### 2.6.2 Recursive background modelling

Unlike the non-recursive techniques, the recursive techniques update a single background model derived from each input frame recursively for background estimation. Hence the current background model will be affected by the past input

frames. The main drawback of this method is that any errors in the background can remain a long time. However, exponential weighting and positive decision feedback that only uses background pixels for updating are used to reduce those effects [21].

McFarlane and Schofield [25] use a simple recursive filter to do the median estimation. If the input pixel is larger than the estimate, the running estimate of the median will be increased by one, and if it is smaller, the running estimate will be decreased by one. As a result, half of the input pixels are larger than the median, and half are smaller because this estimate eventually converges to a value.

Kalman filter, which tracks the evolution of a single value, can be used in recursive techniques for tracking the linear systems under Gaussian noise. Many versions of Kalman filter are used in background modelling. The difference is in the state spaces used for tracking. Examples are the use of luminance intensity [26], the use of intensity and temporal derivative [27], and the use of intensity and spatial derivatives [28].

The mixture of Gaussian (MoG) method tracks multiple Gaussian distributions simultaneously because MoG can keep a density function for each pixel. Also because MoG is parametric, the model parameters can be updated adaptively without maintaining a large buffer for video frames. One example is in ref. [29].

## 2.7 Background updating

During the training time, a background image is modelled. This background image will keep updating according to the change of intensity of the environment. On the other hand, the threshold level is changed associated with the resulting image. The most common two background updating techniques are averaging and selective updating.

The frame averaging technique takes the average of the previous background with the current frame. If a weighted average between the previous background and the

current frame can be formed, the background can be build through exponential updating [30].

On the contrary, the selective updating technique replaces the background with the current frame at the regions without any motion detected. The difference between the current and the previous frames is smaller than a threshold [30]. Selective updating can be performed in a more robust averaging form, where the stationary regions of the background are replaced by the average of the current frame and the previous background [31].

Hoose [30] proposes an equation to perform averaging of $N+1$ frames for real-time purpose.

$$B_{pt} = KB_{pt-1} + (1-K)C_{pt-1};\ 0 < K < 1 \text{ ;and } K = \frac{N}{N+1} \tag{2.2}$$

Where $B_{pt}$ is the updated background picture, $B_{pt-1}$ is the previous background picture and $C_{pt-1}$ is the previous frame of the scene. The rate of updating is decided by the value of $K$. When the value of $K$ moves towards 1, the updated background is closer to the previous one. Hence it can reduce the effect of the current picture. The suitable value of $K$ relies on the ambient lighting conditions and it is often changed manually. "The disadvantage of averaging technique is the quick response to the regions where there is a significant difference between the background and the current picture, while this difference is due to the objects [31]."

In the selective updating, ambient changes are often smaller than the changes caused by the moving objects. Only the scenes that are not covered by the moving objects are updated. At each selected pixel point, the current pixel values will be used to replace the background pixel values. The effectiveness of this method relies on the accuracy of the threshold value. Incorrect threshold value will cause misclassification of foreground and background pixels. Hoose [30] also presented selective updating as follow:

If $D_{pt} > T$

Then

$B_{pt} = B_{pt-1}$ (no update)

Else

$B_{pt} = C_{pt-1}$ (update) (2.3)

Where $D_{pt}$ is the difference image between the current image and the background image and $T$ is the threshold level. This differencing operation is usually selected as an absolute differencing operation. Vehicles that have brighter or darker gray value than the background can be detected.

Based on the work of Hoose [30], Fathy and Siyal [31] introduce a selective-averaging method for background updating. The ambient lighting variations are measured by the intensity changes between two consecutive frames. If in two consecutive frames, the difference of the corresponding pixels is less than the maximum expected variation of intensity, the updating is performed, which means no object is detected. This method takes the advantages of both averaging and selective techniques, and eliminates the disadvantages of them. It is less sensitive to the threshold selection technique. They also propose a dynamic threshold selection technique to make this approach more robust.

## 2.8 Shadow detection and elimination

Shadows are very common in outdoor scene and they can cause serious problems in image processing. Shadows occur when objects totally or partially occlude direct light from a light source. Shadows cast by objects together with objects themselves form distorted images. Shadow points can be misclassified as foreground points that cause the incorrect segmentation of objects from the background. Without shadow detection and elimination, object merging, shape distortion, and even losses may occur. Shadows are difficult to be detected because of the following two reasons. First, the colour information of shadow points are different from the background, which could be misclassified as foreground objects. Second, shadows have the same speed as the vehicles that could be identified as parts of the vehicles. In order to count and classify vehicles correctly, shadows have to be detected and removed.

A shadow has two parts: the self-shadow and the cast shadow. The self-shadow is the part of the object that is not illuminated by the direct light. The cast shadow is the area projected by the object in the direction of direct light [32]. Cast shadows are undesired and should be removed while self-shadows are parts of the objects and should be preserved. The difficulty to identify cast shadows and self-shadows is the intensity similarity between them. Moreover, if objects have similar intensity to the shadows, shadow removal will cause incompleteness in object shapes.

Jaing and Ward [32] propose a 3-step method to separate shadows. First, the threshold line method is used to extract the dark regions in images, which would include shadows such as self-shadows, cast shadows, and dark marks. Secondly, some features will be extracted if they are related to shadows, and these features will be used to analyse shadows. The last step checks the consistency of the features from the second step and resolves the conflicts between them. This method utilises both the intensity and geometry information in shadow identification.

Wu et al. [33] use the knowledge of the date and time of each image to determine the plausibility of shadow occurrence. Shadow processing is only performed if the current time of image is within the time that shadow would occur. In a set-up phase, all possible directions will be searched to detect shadows. Once shadows are found, the detection will continue for several frames to confirm the shadows.

In ref. [33 - 35], cast shadows and self-shadows are discriminated by the edges. The authors assumed that self-shadows contain more edges than cast shadows. According to this assumption, the boundaries are searched between the self-shadows and cast shadows along the estimated orientations of shadows. However, when cast shadows contain many edges, this approach will fail.

Scanlan et al. [36] proposed a method to eliminate the effect of shadows. In this method, input images are divided into blocks. The mean intensity of each block is then calculated, and the median of mean values is determined. A local ratio is computed by dividing the median with the local mean value of each block. After that, the local ratio is multiplied with the pixel values of each block. Shadows will be eliminated after the multiplication. However, using this method will reduce the

intensity contrast between image pixels. On the other hand, there is an existence of blocking effect.

Wang et al. [37] propose an illumination assessment technique to decide whether there are shadows in an image. Once shadows are found, their locations and orientations will be estimated. For each shadow, a numbers of points are sampled. Attributes of shadow are computed from these sampled points. Instead of removing shadows, the authors used three rules to recover object shapes on the basis of the information of object edges and attributes of shadow. Those three rules try to preserve bright foreground pixels, foreground pixels with attributes that are different from those of shadows, and foreground pixels nearby object edges.

Hsieh et al. [38] propose a method that uses the knowledge of lane dividing lines. In their approach, all lane dividing lines are detected first. These lane dividing lines' geometries provide useful information for shadow elimination. Background subtraction method is used to detect vehicles with or without shadows. Once vehicles are detected, a histogram-based method is used to detect different lines from video sequence. A line-based shadow modelling process is employed to eliminate shadows according to these lines.

In most situations, only one camera is used to monitor the road. Onoguchi [39] proposes a method that uses two cameras to monitor the surveillance areas so that the height information of moving objects can be obtained. The image from one camera is inversely projected to the road plane. This projected image is converted to the view from the other camera. Shadows on the road plane engage the same areas as in the converted image, while the areas of objects with different heights occupy different areas in these images from the road plane. By applying a subtraction, shadow areas can be removed.

## 2.9 Occlusion separation

In heavy traffic situation, the detection of a single vehicle is very difficult. Two or more vehicles may partially occlude with each other and may be detected as a vehicle cluster. For example two occluded short vehicles would be identified as a long vehicle, and when counting is performed, they will be counted as one. Occlusion

separation is often done together with tracking. Tracking information can reduce the error of separation.

Rad and Jamzad [40] presents a method to split occlusion in real-time. Each detected region is determined whether it belongs to a single vehicle or it is a merged one using three conditions. Those three conditions are: A. in two consecutive frames, if the centre points of two bounding boxes containing vehicles are very close to each other, then an occlusion may occur. B. if the width of a region is longer than the threshold, then there is more than one vehicle within that region. C. If the size of a region becomes much larger or smaller than that in the previous frame, then a wrong merge or split of region has occurred. Once an occlusion is found, the type of occlusion (horizontal, vertical, and diagonal) will be decided. For each type of occlusion, the corresponding distance will be calculated. For example, for a horizontal occlusion, the vertical distance from top to bottom sides of the bounding box is calculated. These values will be stored in two arrays A and B. Another array C is used to store the sum of A and B. After this, a split process will be conducted at the maximum value of C.

Jung and Ho [41] define two types of occlusions: explicit occlusion and implicit occlusion. Explicit occlusion is the situation that two or more vehicles appear separately, and then they are merged because of some occlusion conditions. Implicit occlusion is the situation that an original occlusion where two or more vehicles appear as a single object. The separation is used together with tracking. After merged blobs are separated, the type of occlusion is checked. If they are explicit occlusion, feature matching algorithm is employed to establish matches between the objects before and after occlusion. If the occlusion type is implicit, the occlusion reasoning algorithm is used to create the new trajectories of the separated objects. Then the algorithm is used again to connect the original trajectory with the new trajectories.

Heungkyu and Hanscok [42] use occlusion activity detection and object association algorithm to solve occlusion problems. By using the Kalman prediction equation, the occlusion activity detection method affords the occlusion status of next state. Then the status of current state is modified with this status. If an occlusion is found, an object association technique that uses a partial probability model is employed. The reliable centre points of occluded objects can be found using this algorithm.

Nevertheless, this method is not suitable in the case of high variation of illumination change and shadow effects because a tracking failure can be caused by the missing estimation of geometric information and an appearance model.

Zhu et al. [43] propose a Markov Random Field-Maximum a Posterior (MRF-MAP) framework for tracking vehicles with occlusion. The background scene and moving objects are treated as layers and MRF-MAP is used to model the correlations between consecutive images to get the layer label during occlusion. The MRF-MAP labelling can define the depth orders of occlusion related to the objects that are in the subsequent frames. After labelling, special parametric structuring elements that are based on the mathematic morphology theory are used to convert the contour into a series of feature points. Then these points are used to label the sites, which can reduce the computational load of labelling.

## 2.10 Vehicle tracking

A vision-based traffic monitoring system is required to track vehicles through the video sequence. Tracking can avoid multiple counts in counting application. Tracking can also provide other useful information for data extraction like calculating velocity and acceleration of vehicles. Furthermore, tracking can be used in solving occlusion problem, and refine the vehicle type in the classification applications. Hence tracking is the first and the most important step in traffic monitoring. There are four main types of vehicle tracking, namely model-based tracking, region-based tracking, contour-based tracking and feature-based tracking. In the next four sections, brief details of these four tracking approaches will be discussed. Other tracking approaches are discussed in 2.10.5.

### 2.10.1 Model-based tracking

Model-based tracking algorithms localize and recognize vehicles by matching a projected model to the image data. Koller et al. [44] present a useful 3-Dimesional (3-D) model-based tracking method. They use a cluster of moving image features to offer an estimate for moving regions in the image. Then the straight line edge segments that are extracted from the image are matched to the 2-Dimensional (2-D)

model edge segments using a hidden-line algorithm to determine their visibility. Mahalanobis distance of the line segment attributes is used to perform the matching between the image edge segments and model segments. This algorithm is data driven and depends on the accuracy of edge detection.

Kollnig and Nagel [45] present a method to estimate the vehicle pose by directly fitting image gradients to polyhedral vehicle models without an edge segment extraction process, which can extract more information from the image data. This approach can solve occlusion problem by textured objects. In ref. [46] they also propose an image-gradient-based algorithm in which virtual gradients in an image are produced by spreading the binary Gaussian distribution around line segments. Under the assumption that the real gradient at each point in the image is the sum of a virtual gradient and the Gaussian white noise, the pose parameters can be estimated using the extended Kalman filter [47].

Furthermore, Haag and Nagel [48] integrate the algorithm based on image gradients with an algorithm based on optic flow. This method combines edge element and optical flow estimates in the measurement process and a more consequent exploitation of background knowledge. They use a generic polyhedral vehicle model to perform tracking.

Tan et al. [49] use the ground-plane constraint to reduce the pose redundancy of 2-D image and 3-D model line matches. Hough transform is used with explicit probability-based voting models to obtain consistent matches and to spot the approximate poses. Furthermore, in ref. [50] the authors analyze the 1-dimensional (1-D) correlation of image gradients and determine the vehicle pose by voting. The algorithms are very suitable for real-time implementation.

In order to reduce the computational load, Foresti et al. [51] combine the recognition and tracking processes for autonomous vehicle driving. Recognition assigns semantic labels to detected objects and tracking establishes identity and pose correspondences between detected objects at different time instants.

Gloyer et al. [52] propose a 3-D model-based tracking system for freeway traffic monitoring. In their approach, during the modelling stage, a 3-D model of the background is built. This background can be used to compare current frame to find

out the foreground objects, and create a mapping between the detected vehicle locations on each frame to the 3-D space. In the difference image, vehicles can be represented by solid blobs, while noise only occupies a few pixels and can be eliminated in the following steps. For each blob, the position, area, and an estimate of its density are computed. During the tracking stage, each vehicle is isolated and tracked over many frames. Each frame is considered as a 3-D space, and those nearest blobs in the neighbouring frames are located and tracked, also an estimation of the expected position of the blobs in the next frame is made.

Model-based tracking has the following advantages. First, it exploits local visual cues because it is based on local image features. These cues can increase tracking accuracy. Second, the knowledge about the scene can be used to predict the hidden movement and acts of the objects and detect partial occlusions. The effects of outlier data introduced during tracking process are reduced. The disadvantages are the requirement of 3-D models and high computational cost. It is also unrealistic to expect to have all detailed models for all types of vehicles that could be found on road. Although researchers tried to reduce its computational load [50 - 51], real-time applications are still difficult to be performed.

### 2.10.2 Region-based tracking

In region-based tracking approach, a connected region in the image, also called a 'blob' is identified associated with each vehicle and then it is tracked over time using a cross-correlation measure. Typically, the process is initialized by the background subtraction technique [53].

Karmann and Brandt [54], propose a method that uses a Kalman filter-based adaptive background model, which allows the background estimate to evolve as the weather and time of day affect lighting conditions to track the vehicles. The main drawback is when vehicles are partial occluded it is difficult to segment them into individual vehicles [53]. Gupte [55] suggests a new approach to solve this problem. In this approach, region-based tracking is done by using an associated graph. An association graph is formed between the regions from the previous frame and those in the current frame that describes the relationship between them. Tracking problem can be

modeled as finding the maximal weight graph. The location, velocity and dimensions can be updated during tracking based on this graph.

Another system is the condition-monitoring system (CMS) Mobilizer system, which was supported by the Federal Highway Administration (FHWA) at the Jet Propulsion Laboratory (JPL) of America, Pasadena, CA, [56 - 57].

Region-based tracking only tracks a window of interest for each object and tracks the region as a whole. It has low dimensionality, and can provide comprehensive description of objects. Hence it is an efficient tracking strategy. By the collaboration of background subtraction, regions describing objects are easy to obtain. However, when vehicles are partial occluded, it is difficult to separate them into individuals. On the other hand, it matches the blob with candidate blobs in the next frame near the position of the blob, if the number of candidate windows is large, the computation load will be large.

### 2.10.3 Contour-based tracking

Contour-based tracking is the tracking strategy based on active contour models. Moving objects are represented by the outline of themselves as contours, which keep updating dynamically in successive frames. Koller et al. [58] propose an approach for detecting and tracking vehicles in road traffic scenes. Moving vehicles are presented by closed contours and a contour tracker based on intensity and motion boundaries is employed to track them. Motion and contour estimation is performed by linear Kalman Filters based on an affine motion model. Koller et al. [59] use closed cubic splines to represent contours, and the position and motion of them are estimated along the image sequence. Linear Kalman filters are used for estimation.

Dubuisson and Jain [60] proposed an algorithm to extract object contours of moving objects from the arbitrarily complicated scenes. They combined motion segmentation technique based on image subtraction with the colour segmentation technique based on the split-and-merge algorithm. Then these contours are tracked.

Contour-based tracking using Marginalized Likelihood Ratio (MLR) is another approach. These types of approaches allow for random and un-modelled shape

variations. In MLR, neighbourhood pixels' intensity is correlated if they are belonged to the object being tracked or the background. But they are uncorrelated if two of them are on the opposite sides of an object boundary. Pece [61] gives a detailed description of MLR contour-based tracking.

Because edges are usually well-defined and easy to get, contour-based tracking is robust, precise and has low computational load. Contours depend only on the objects shape, so we can track objects that have no feature points on the surface. However, edges are not distinctive features like points. Hence wrong edge may be followed during tracking. The inability to segment partially occluded vehicles is the problem. Also tracking precision is limited by a lack of precision in the location of the contour.

### 2.10.4 Feature-based tracking

Feature-based tracking tracks sub-features such as distinguishable points or lines on the object instead of tracking objects as a whole. According to the nature of selected features, feature-based tracking algorithms can further be classified into three subcategories namely global feature-based algorithms, local feature-based algorithms, and dependence graph-based algorithms.

Global feature-based algorithms use features like centroids, perimeters, areas, some orders of quadratures, and colors [62] to track vehicles. These algorithms can run at a high speed and with a high recognition rate. However, they cannot easily acquire the 3-D pose of vehicles and cannot deal effectively with occlusion and overlapping [48].

Local feature-based algorithms use features such as line segments, curve segments, corner vertices to track vehicles. These algorithms can partly handle occlusion and overlapping and have a low computational cost. However, they are unstable and easy affected by unrelated image structures. In ref. [53], after detecting the vehicles, only the corner features in detection zones that are located at the bottom of the image are used to track vehicles. These corner features are tracked using Kalman Filter to predict a given corner's location and velocity in the next frame.

Dependence graph-based algorithms use features such as a variety of distances and geometric relations between features [63] to track vehicles. These algorithms can

handle occlusion and overlapping. However, they are not suitable for real-time applications because of time-consuming searching and matching of graphs.

Tissainayagam and Suter [64] propose an object tracking technique based on the Bayesian Multiple Hypothesis Tracking (MHT) approach. Multiple hypothesis based tracking methods can provide reliable tracking results, because the MHT technique is a statistical data association algorithm that is composed by all the capabilities [64]. The *K*-best hypotheses [65] are employed in their approach. Once all the required feature points are found, the MHT-interacting multiple model (MHT-IMM) tracker is applied for tracking the key points through an image sequence. However, if the tracked objects occluded with each other, contour grouping process can be broken down.

Bevilacqua et al. [66] propose an algorithm that uses local and global objects' information to track vehicles. This approach can be treated as combining the benefits of both region-based tracking and feature-based tracking. Corner points of vehicles are extracted, and then they are used to track vehicles. Once no errors occur, these points are grouped into objects. The global information is used to improve object identification when dynamic occlusion occurs. It is used to guess object properties after splitting or missing, or even refine local information. This algorithm is good at solving static and dynamic occlusion, object splitting and missing.

The main advantage of feature-based tracking is that it can solve partial occlusion problem and different illumination problems. In the case of partial occlusion, some features of the moving vehicles are still visible. On the other hand, in different illumination conditions, some features like the corner of vehicles can be found. However, errors may accumulate and points may lose their corresponding features during tracking. The accuracy of tracking also depends on the number of tracking points.

### 2.10.5 Other approaches of vehicle tracking

Kamijo and Sakauchi [67] propose a Spatio-Temporal Markov Random Field model (S-TMRF) to track vehicles. Magee [68] combines a mixture of Gaussians colour

background model that is a multi-modal statistical model and a set of foreground models to estimate an object's position, velocity, size, and colour distribution. Rajagopalan and Chellappa [71] propose an algorithm that combines statistical knowledge of the 'vehicle class' with spatio-temporal information to track and classify vehicles. The tracker uses position co-ordinates and higher-order statistics (HOS)-based difference measurement values to estimate the corresponding position among frames.

Kim et al. [69] combine an adaptive threshold and wavelet-based neural network to track vehicles. The proposed system has low computational load, high localization accuracy, and is robust to noise. Adaptive thresholding is used on two consecutive frames to get the frame difference and extract the moving objects. Then the wavelet-based neural network is used to recognize vehicles from the moving objects. Vehicle tracking is performed by using the position coordinates and the wavelet features difference values in the consequent recognized vehicle regions.

Ha et al. [70] propose a neural-edge-based vehicle detection method for detecting and classifying vehicles. After an accurate background is obtained, the moving edges can be found by using Sobel edge detector. Information of vehicles is then extracted and input to the neural network. The individual vehicle is tracked within a tracking area on each lane. When a new vehicle is detected in the testing zone, it is added to the linked-list, which contains the information about it. The information and linked-list will be keep updating while the vehicle moving within the tracking zone. When it is detected as an exited vehicle, the number of vehicles will increase one.

### 2.10.6 Summary

Through the above literature review, we can find that model-based tracking is not suitable for our algorithm due to its high computational cost. Although more powerful hardware can solve this problem, the price of the system is still an important constrain. Due to the size of the tracking window and the area that is occupied by a vehicle, feature-based tracking is not considered as the tracking strategy because it is difficult to get enough feature points to present a vehicle. The main constrain on Kalman filter, which is only applicable to linear or nearly linear problems also make

it difficult to use in our algorithm. Neural-network based scheme also requires high computational load and does not fit the real-time application needs. Hence region-based tracking and contour-based tracking are selected as the tracking schemes in our algorithm.

## 2.11 Vehicle counting

Vehicle counting is very important in traffic monitoring system. Crowdedness of a given road at different time and date is useful in traffic control and road design. Vehicle counting that is always accomplished by vehicle tracking computes the total number of vehicles passing through the road during one period.

Soh et al. [72], propose a vehicle counting approach using a vehicle movement tracking method. The position of vehicles in each frame is represented by points on the image plane and counting is achieved by tracking of their movements. Vehicle counting is only performed by counting the newly incoming vehicles.

Perera and Harada [73] present a method based on frame comparison with the background and pattern matching. Background subtraction is used to detect vehicles firstly. A frame grabber grabs a frame from video sequence at a certain period. A small window from the frame is considered, and only pixels in that area will be processed. For each window, if the total number of pixels of one object satisfies the constrain, it will be regarded as one vehicle. When there are several windows defined, the system will track all windows using pattern matching to avoid multiple counts.

Jung and Ho [41] define tracing ranges as solid lines. If a new vehicle is detected in the detection area of the white solid line box, the tracking algorithm tracks the vehicle until the end of the white solid line and count. They also further determine the type of vehicle behaviour based on the velocity and trajectory of vehicles, and the geometrical information of lanes.

## 2.12 Vehicle classification

The aim of classification is to categorize vehicles into a sufficient large number of classes. Due to the types of vehicles are huge, it is impossible to obtain information of all types of vehicles. Due to the short process time, it is impossible to classify vehicles in details in one or two steps. Hence hierarchical classification methods are proposed that can quickly classify vehicles at a coarse granularity. Further classification will be performed according to the application needs such as doing off-line analysis on the extracted vehicles. On the other hand, the classification system should be able to classify objects that are similar in appearance. For example, the shapes and contours for a detected bicycler and a detected motorcycle look very similar through the extraction. The classification should include other criterions such as speed to separate them.

Liu and Zhou [74] propose an approach to classify vehicles in three stages. Firstly, objects are classified into vehicles and non-vehicles according to their speed. The non-vehicles will be stored in unidentified objects (UO) list. Then the compactness and aspect ratio of objects are calculated. According to the width of objects, UO list is further divided into unidentified big object (UBO) and unidentified small object (USO). Associated with the compactness and aspect ratio information, objects can be classified into human list and vehicle list.

Noll et al. [75], use two types of classifiers to perform classification, a Bayes-classifier and a multi-layer preceptor. First, correspondence between model and image features is established by an optimization algorithm, which is called constrained elastic net (CENET). Model feature position according to an iterative equation that optimizes an objective function is updated iteratively. Then a matching vector is derived and is used as input to the Bayes classifier, which is a neural net based on the correspondence.

In ref. [55], classification is conducted based on the dimensions of vehicles. This system classifies vehicles into two categories: cars and non-cars. The actual length and height of the vehicles are computed. Here, height is the combination of vehicle's width and height. A discrimination function that is calculated based on the mean and variance of samples is used to classify vehicles. The disadvantage is that in some

cases, trucks are only slightly larger than the average of cars, and some cars are only longer or wider than a truck, which will bring classification errors.

Hsieh et al. [80] propose an algorithm that uses the lane information and vehicle histogram to get better tracking and classification results. The information about lane-dividing lines and lane width is obtained at the initialization stage. A vehicle histogram is calculated by accumulating the number of vehicles passing a position. According to the extracted parameters, moving vehicles can be detected. Then the vehicles are tracked by a Kalman filter. Size and linearity of vehicles are used to classify vehicles in this approach. The size of the vehicles is not constant as vehicles moving. In order to solve this problem, the authors use the information of lane-dividing lines and lane width to normalize the vehicles in advance. Then an optimal classifier is used to divide vehicles into four different categories using the size and linearity information.

Other vehicle classification approaches can be found in many literatures. Kjellgren et al. [76] propose a method that uses radar measurements that includes Doppler registrations from a stationary radar to observe and classify vehicles. Abdelbaki et al. [77] proposed a laser intensity image based automatic vehicle classification method. Hussain et al. [78] use range sensors that can collect features of vehicles in the form of an $n$-dimensional vector that is suitable for classification to category vehicles. Nooralahiyan et al. [79] proposed a vehicle classification method by using acoustic signature of vehicles.

## 2.13 Segmentation

Segmentation is a fundamental problem in image analysis. According to some image's features, such as intensity, colour, textures, images are segmented into several disjoint regions. Then these separated regions will be analysed further. Segmentation can cause the problem of discontinuity. Hence it should have a balance between adherence to the noisy and incomplete data and smooth segmentation results, which is suitable for the subsequent analysis.

### 2.13.1 Colour segmentation

Colour segmentation is the process that divides the image into homogeneous regions using the colour information at each pixel. There are three common methods for colour segmentation, namely clustering, recursive region splitting, and split and merge algorithm [67].

Clustering is a method that finds homogeneous clusters of points in the 3D colour space and labels each cluster as a different region. The similarity between points to each other is defined by the Euclidean distance metric in the colour space [81].

Recursive region splitting is developed by Ohlander [82]. It takes a region of the image and determines a threshold in one feature by finding the best peak in the set of feature histograms (initially the entire image is considered). The subregions defined by this threshold are then further segmented, if necessary.

Horowitz and Pavlidis [83] developed the split and merge algorithm. This algorithm uses the maximum and minimum intensity values in a region for splitting and merging. In this algorithm, the image ($N$ x $N$ for instance) will be divided into many $z$ x $z$ squared images. If the sub-images are not homogeneous, they will be divided into 4 squared images, which are $z/2$ x $z/2$. If 4 adjacent of them are satisfied the homogeneous conditions, they will be merged into $2z$ x $2z$ blocks. Neighbour sub-images will be added to form a bigger region when they satisfy some criterion. When no more sub-images can be added to existing regions, a new region is created.

In most case, split and merge algorithm is better than the other two. Edge information is often used together with colour segmentation to produce better result.

### 2.13.2 Energy-based segmentation - snake

Other approaches used in segmentation can be classified into two categories: energy-based and watershed-based [84]. In energy-based approach, segmentation is obtained through minimizing the energy function. Two major terms are involved in this approach, namely data-driven and regularization. The data-driven term can be classified into two classes: contour-based (also called snake-based) and region-based.

The regularization term can impose the priori knowledge to make the segmentation result smoother.

Kass et al. [85] proposed a method called snake that uses image's gradient information as input data to minimize contour energy. This method can balance maximal edge evidence with minimal curvature. The final form of a contour can be influenced by feedback from a higher level process. Initially, with the help of an edge detector, a contour is generated through some processing, and then it would be placed near the edge of objects. When snake algorithm is applied, the image forces draw the contour to the edge in the image. The number of iterations can either be pre-defined by user, or be determined by the gradient of the image. As a result, a local minimum can be obtained, which can be further used in next set of operations.

The active contour model can be represented as a vector $V(s) = (x(s), y(s))$, where $s$ is the arc length. A contour can be represented as:

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{im}(v(s)) + E_{con}(v(s))ds \tag{2.4}$$

$E_{int}$ is the internal energy of the contour due to bending or discontinuities. $E_{im}$ is the image forces that due to various events such as edges or terminations. $E_{con}$ is the external constraints [88].

The internal energy can be represented as:

$$E_{int} = (\alpha(s)\,|\,v_s(s)\,|^2 + \beta(s)\,|\,v_{ss}(s)\,|^2)/2 \tag{2.5}$$

The first term is curve, if there is a gap in the curve, it will result a large value. The second term is continuity, if the curve bends rapidly, the value will be large. The value of $\alpha$ and $\beta$ at a point determine the extent of contour such as stretching or bending at that point. If $\alpha$ is 0, a discontinuity can occur, and if $\beta$ is 0, a corner can develop. Techniques of variational calculus are employed to minimize energy contour [88].

However, the variational techniques require several optimal conditions. Although these conditions are all necessary, they are usually not sufficient. Amini et al. [86 - 87] mentioned some of the problems that would occur when using this approach, and they proposed a new method that uses dynamic programming to determine energy minimizing for the contour. This method includes hard constraints that can not be

infringed. The first-order and second-order continuity constraints in the internal energy representation are called soft constraints, which are not satisfied absolutely, but only to a certain degree. This approach is good for optimization because it bypasses local minima and allows for enforcement of hard constraints on the solution within a natural and straightforward structure. Constraints can make the contour more controllable, and hence more optimal. For energy minimization of the active contours, the time-delayed discrete dynamic programming algorithm is used. The main drawback for this approach is the high computation load, which is $O(nm^3)$. Here $n$ is the length of the contour and $m$ is the number of possible choices at each stage. The storage requirement also increases from $n \times m$ to $n \times m^2$ memory elements when considering the second order term.

Williams and Shah [88] proposed a greedy algorithm that can reduce the computation load to $O(nm)$ and allows the inclusion of hard constraint described by Amini et al. [86]. Here $n$ is the points that are allowed to move to any point in a neighbourhood of size $m$ at each iteration. However, this algorithm can not guarantee a global minimum.

In this approach, the quantity being minimized can be written as

$$E = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds \tag{2.6}$$

The first and second terms are the first-order and second-order continuity constraints, which correspond to the internal energy. $E_{image}$ is the image quantity such as edge strength. The external energy can be included, but it is not considered in their algorithm [88].

The first term can be calculated as $|V_i - V_{i-1}|^2$, which can cause the curve to shrink. This term can minimize the distance between points, and also contributes to the problem of points bunching up on strong portions of the contour. Thus the algorithm uses the difference between the average distances between points, $\bar{d}$, and the distance between two points as $\bar{d} - |V_i - V_{i-1}|^2$. Minimum value will occur if points having distance near the average. The value of average distance will be updated after each iteration [88].

The second term that is the curvature can be obtained using formula $|V_{i-1} - 2V_i + V_{i+1}|^2$. This continuity term can force the points to be spaced relatively evenly. This term is normalized by dividing the largest value in the neighbourhood and giving a value from 0 to 1 [88].

The third term $E_{image}$ is the image force that is the gradient magnitude (mag). The Sobel approximation is used to determine the image gradient, which are two arrays: Cx=[-1 0 1; -2 0 2; -1 0 1] for x coordinates and Cy=[1 2 1; 0 0 0; -1 -2 -1] for y coordinates. After finding all 8 neighbour gradients, the maximum (max) and the minimum (min) gradients are determined. Then (min-mag) / (max-min) is used for the normalized edge strength term. This gradient magnitude term is negative so that points with large gradient will have small values. In order to avoid divide by zero error, the minimum value allowed in the denominator is 5 [88].

Classical snake models will meet problems when the boundaries of the edge are discrete, because these models rely on the edge function and depend on the gradient. Chan and Vese [89] proposed a method that can detect objects whose boundaries are not necessarily defined by gradient. They treat the energy minimizing as a particular case of the minimal partition problem. Using this method, the original image does not need to be smoothed.

### 2.13.3 Watershed segmentation

On the other hand, watershed segmentation that uses mathematical morphology method, is quite different from snake approach. Each gray level image contains morphological gradients, which can be thought as a topographical surface. The watershed algorithms divide this topographical surface in different catchment basins by the watershed lines.

There are two classical methods to achieve watershed segmentation: the rainfalling simulations and the immersion or flooding simulations. Flooding simulation is easy to perform than the rainfalling simulation. It can be imagined as filling the catchment basins from the bottom. Each catchment basin is represented by a minimum, and the altitude minima will be filled first. If all catchment basins are filled, all minima are

under the water level. Two catchment basins would merge if further immersion continued, a dam would be built all the way to the highest surface altitude and the dam represents the watershed line [90].

The main drawbacks of the watershed algorithms are that they are high sensitive to noise, which results in an oversegmentation and makes the contours not smooth that make tracking difficult [90]. Oversegmentation occurs because every regional minimum, even if tiny and insignificant, forms its own catchment basin. Therefore, a fusion process, a marker preprocessing or low pass filter is required to modify the segmentation.

One solution is to remove the minima that are too shallow. This can be done by suppressing all minima in the intensity image whose depth is less than a threshold value. Then by applying a Gaussian low pass filter to blur the image, the sensitivity can be reduced. After these two procedures, the watershed lines can be displayed later on the original image.

Another approach is proposed by Salembier [91], which uses a binary partition tree to organize the oversegmented regions. In their algorithm, once regions are obtained from the initial partition, a binary partition tree is used to represent them. Then the marker and propagation is applied on the binary partition tree. The propagation processes are based on similarity between neighbouring regions.

## 2.14 Colour space

In gray scale image, a scalar gray level is assigned to a pixel. While in colour images, a pixel is always defined by three values corresponding to the tri-stimuli, for example red, green, and blue, hence a colour vector is assigned to a pixel. Using colour image in detecting foreground objects and shadow can provide better results.

Color images are often displayed by changing the intensity of the three primary colors: red, green, and blue at each pixel of the image. Different combination of these three channels can present all possible shades. There are many color spaces are currently

used in computer vision, such as RGB, HSV, XYZ, $YC_rC_b$, and rgb. These color spaces can be shown in mathematical relationship with RGB color space.

RGB is the most commonly used color space for the television system and pictures because it has three primaries, and it has relatively low computational load. It has some drawbacks such as high correlation among the R, G, and B components, which makes it not suitable for color segmentation and analysis. High correlation means if the intensity changes, all the three components will change accordingly. However, by using other techniques, the effects of high correlation can be reduced.

HSV color space describes any color using three quantities: hue, saturation, and value. Hue is the color type, such as red, green and so on, which ranges from 0 to 360 degree. Saturation is the purity of the color, which ranges from 0 to 100%. Value is the intensity value, which ranges from 0 to 100%. HSV is similar to the way humans tend to perceive colour.

XYZ color space is the fundamental color space of the Commission International d'Eclairage (CIE) and it is the standard based on colour matching experiments on human observers. In XYZ color space, the brightness of the colour is presented by $Y$, and the chromaticity of a color is then specified by the two derived parameters $x$ and $y$, which are the functions of all three tristimulus values $X$, $Y$, and $Z$.

$$\text{Here } x = \frac{X}{X+Y+Z} \text{ , and } y = \frac{Y}{X+Y+Z} \tag{2.7}$$

The $X$ and $Z$ tristimulus values can be calculated back from the chromaticity values $x$ and $y$ and the $Y$ tristimulus value:

$$X = \frac{Y}{y}x,\ Z = \frac{Y}{y}(1-x-y). \tag{2.8}$$

$YC_bC_r$ is a color space used in video systems. Y is the luma component and $C_b$ and $C_r$ are the blue and red chroma components. Luminance is the weighted sum of the linear RGB components of a color video signal, proportional to intensity, luma is the weighted sum of the non linear R'G'B' components after gamma correction has been applied, and thus is not the same as either intensity or luminance. This color space is robust to bright light background and good for detecting shadows.

The last one is rgb color space, which is the normalized RGB color space using the following equations:

$$r = \frac{R}{R+G+B}\ ,\ g = \frac{G}{R+G+B},\ b = \frac{B}{R+G+B} \tag{2.9}$$

This color space is insensitive to bright light area. However, errors in classification would happen when foreground objects passing over the bright light background. It could not detect shadows but good at detecting foreground objects. In many tests, *rgb* color space always can have an averaged result compared to other four color spaces. Hence it is always rejected when selecting a color space.

Some literatures [92 - 93] argue that $YC_bC_r$ is a suitable color space used for foreground objects segmentation and shadow detection. Using this color space, false classification caused by shadow and highlights can be solved. $YC_bC_r$ also can provide good detection of shadows. RGB color space on the other hand has false classification in bright light background, and when lighting condition is not very bright, some foreground pixels will be misclassified as shadows. However, by adjusting the lighting condition, RGB color space can give high true detection rate for connected components detection.

## 2.15 Summary

In this chapter, a review on the literature in our research areas is presented, which demonstrates the state-of-art techniques in the system we proposed.

The review consists of 5 parts. We first state the requirements of a real-time traffic monitoring system. The second part is about vehicle detection, approaches based on vehicle detection, background modelling and updateing are discussed. In this part, we compare the advantages and disadvantages of all the approaches. The third part is about occlusion seperation and shadow handler, which are used to reduec tracking errors. The fourth part is the main part of our research that is vehicle tracking, counting and classification. Many approaches in these three topics are presented. Through the brief discussion, combination of contour-based and region-based tracking is selected as our tracking method. The last part is the examination of some

related techniques such as segmentaion and color space. Details about energy-based minimisation method – snake is given.

Through comparison and analysis, suitable methods are selected and modified for our algorithm development.

In the next chapter, background modelling and updating, and object detection are presented.

# 3. Background modelling and vehicle detection

## 3.1 Introduction

The background subtraction approach has been selected to be used in our vehicle detection system, due to its effectiveness and low computational load when compared to other approaches. However, the effectiveness of this approach depends on the quality of a background image, the updating of the background and the selection of the threshold level. Our objective is to design a method that is suitable for outdoor application. This method should be able to be applied in different weather and illumination situations.

Our approach consists of three steps. In the first step, a background is initialised using the averaging technique. The second step is to model a new background, which is based on the averaged background. Updating is performed at the same time in order to make the new background image as close to the current image as possible. The third step is to perform a background subtraction, in order to obtain an image which only contains vehicles.

In the following sections, these three steps are described.

## 3.2 Background initialisation

As mentioned previously, the input video sequence is sampled at a frame rate of 20 frames per second. The input video sequence for background initialisation is 10 seconds in duration, which gives 200 consecutive images. These images are in 8-bit RGB colour space, which contain 256 intensity levels in red, green and blue respectively.

Firstly, these input images are processed by a Gaussian filter [21] to reduce noise. A Gaussian filter is a smoothing filter that provides the least amount of spatial blurring for any desired amount of random noise reduction. It operates with a convolution kernel that is a Gaussian function. It can provide no overshoot to a step function input while maximising the rise and fall time. Hence it is very popular in the pre-processing

step in image process to reduce the image noise. Then they are averaged together in order to obtain the first background image.

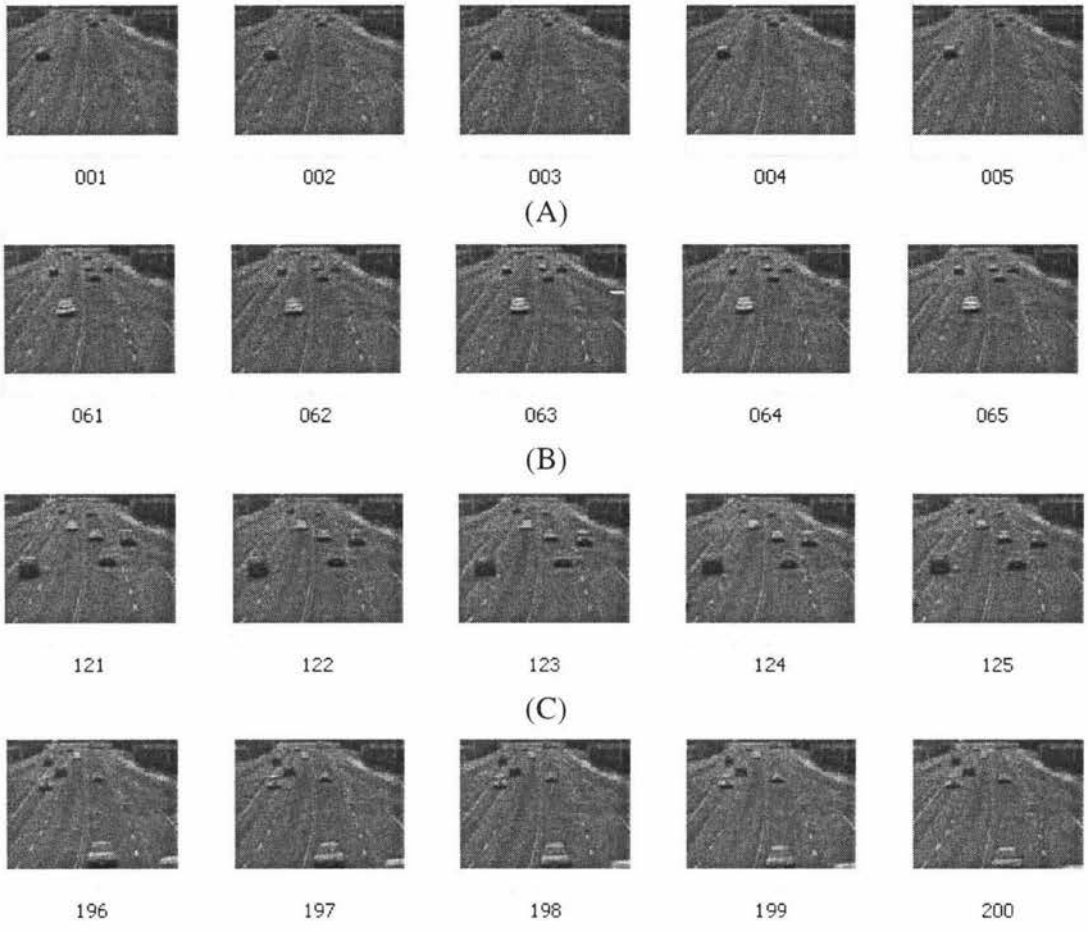



Figure 3.1 input image sequences (A) 001-005, (B) 061-065, (C) 121-125, (D) 196-200.

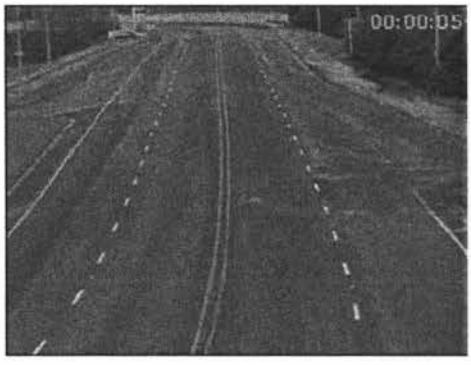



Figure 3.2 averaged background image

Figure 3.1 shows some images of the first 200 input images, such as image 001, 061, and 121 etc. Figure 3.2 shows the averaged image. From figure 3.2, we can see that, through averaging, there are no more moving objects on the road. However, the background image is becoming blurred and some useful information, particularly the texture information is missing. Image texture is a function of the spatial variation in pixel intensities. In image processing, we often make assumption about the uniformity of intensities in local image regions. However, this uniformity does not exist. For example, the surface of a black sedan contains variations of intensities that form certain repeated patterns called visual texture. After Gaussian filter that is equal to smoothing process and image blurring, the intensity variations will be reduced. Hence the texture information is lost in the image. The image is more uniform than before. If we perform background subtraction on this image, some false detection will happen. Hence the background image has to be updated in the next step, in order to obtain a more accurate background image.

During the background initialisation stage, it is necessary to obtain the following information:

Firstly, it is necessary to know the calibration of the road being monitored. When a camera is mounted on a bridge, we know the angle between the camera and the road. Through the calibration, the actual distance represented by one pixel at a different area can be confirmed. For example, when a pixel is near the camera, it can only present 0.25 or 0.5 metre. When it is far away, it can be 3 or more metres. This calibration information can be stored in a distance database for further use. According to the speed limit of the road, the actual pixels one vehicle can pass without speeding can be calculated. For example, on a 60 km/s limit road, a vehicle can only pass 16.7 metres per second, and 0.83 metres per frame. In the 0.25 metre per pixel area, if the vehicle passes 3 to 4 pixels on average, the system will treat it as running at normal speed. If the vehicle passes more than 4 pixels on average, the system will treat it as speeding. This set of data can also be stored in a distance database.

The calibration is actually done by measuring the length of the road with marks on every 50 meters, and then capturing image with these marks. Then number of pixels between two marks is computed through the image. This calibration contains some

errors, such as one mark is at the half of a pixel, for example 62.5, the system will round it to 63. The number of pixels for next 50 meters will be increased by one. However, this rough calibration is enough for monitoring use. Because for the speed measurement, the average speed for every 400 meters is used, not the instant speed.

The second information required is the curvature of the road dividing lanes. A piece of code is used to detect the lane lines in the background. The curvature can then be calculated when the curve occurs on the line. Lane lines are the thin, continuous and straighter lines when observed from the image. When background images are blurred, after removing big blocks that represent grass, bridge or hills in the image, land lines can be extracted. The curvature is calculated by getting the gradient from every 3 pixels.

## 3.3 Background updating

Background updating can be divided into two parts: static background updating and dynamic background updating. Static background is the area that no vehicle passes through, such as the roadsides, grass or hills. Dynamic background is the area that vehicles are actually moving on. After the background image is modelled, background updating starts.

For the static background, we assume that the variance of corresponding pixels between each image and the averaged background image is less than 5%. Comparison is conducted between each image and the averaged background image. The modelling procedure is as follow:

1. At first, the check symbol, which indicates pixels that may belong to a dynamic background, is set to false. If the difference between one pixel in current image and the corresponding pixel in the average background image is less than 5%, this pixel will be copied and put into the corresponding pixel location in the new background image.

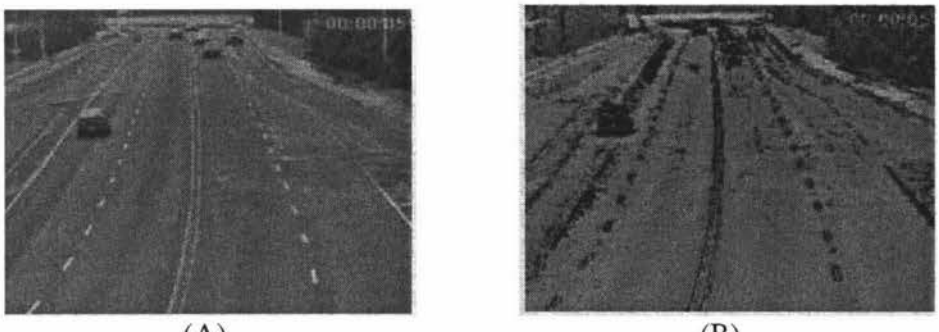
2. For a new coming image, if one pixel satisfied step 1 but the corresponding pixel location is already occupied by another pixel from previous frames, then the values of the new pixel will replace the old one.

3. If the difference is greater than 5% and the corresponding point in the new background is not zero, then the value of this point in the new background will be reset to zero, and a mark would be put to indicate that this point probably belongs to a dynamic background. The check symbol for this pixel on a new background will be set to true. The values of that point of current frames can be stored in an $N$ x 200 dynamic array, which is called a dynamic check array. The coordinates of that point will be stored in an $N$ x 2 dynamic array called a dynamic check coordinates array. The value of $N$ depends on the number of points that have a difference greater than 5%.

4. If the check symbol is true, the values of corresponding pixels of a new coming image will be stored in the dynamic check array. Step 2 and 3 will then be performed.

5. After searching if all the values of one row are quite similar, except one or two large different values, this point can be treated as a static background point. These break points could be caused by flicker, or illumination change in a very short time. If the values are fluctuated, or there are many different, large values, this point will be treated as a dynamic background point.

Figure 3.3 shows the result of image 001 without vehicles on. Figure 3.4 shows the final static background image after procedures.



(A) (B)

Figure 3.3 Image 001 without vehicles on

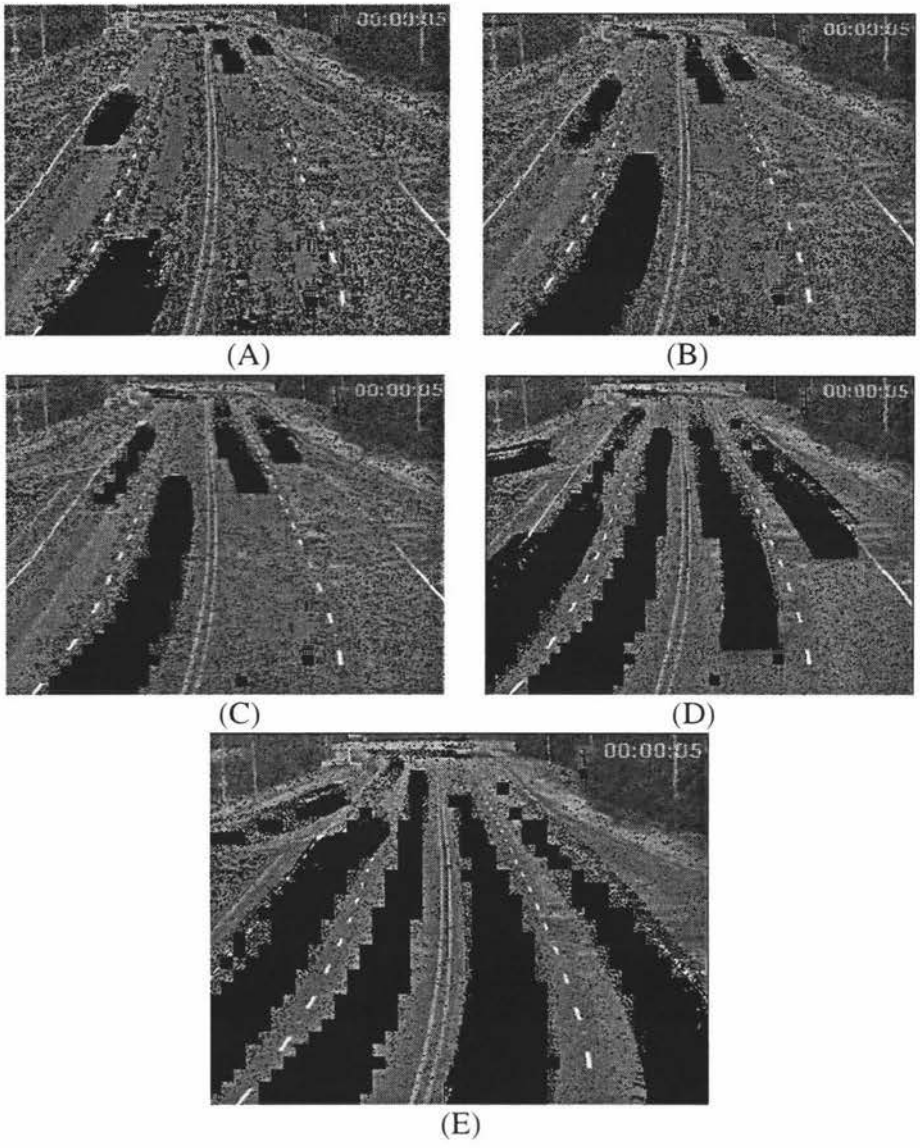



Figure 3.4 Final Figure of static background image (A) first 30 frames, (B) first 60 frames, (C) first 90 frames, (D) first 150 frames, (E) final image.

Once the static background is built, the coordinates of these pixels will be recorded in an array named 'static background array'. Static background pixels can be updated together by copying the pixel values from the current image, if foreground objects do not cover them, or by using a weighting factor. This can be done once every ten minutes, or more if the intensity change is not significant.

The processing for static background modelling takes about 30 seconds. But there are not many changes happen and the current image is very similar to the image 30

seconds before. Once the static background is built, in the following 10 seconds, the next 200 images are used to build the dynamic background. An array called 'dynamic background array' is used to store the values of the dynamic background pixels. Only the pixels that do not belong to the static background array are checked.

1. The 'done' signal that indicates updating is finished is set to false. If the difference between one pixel and its corresponding pixel in the average background image is less than 5%, then values of this pixel will be put into the corresponding pixel location in the dynamic background image.

2. For a new incoming image, if a pixel satisfies the 5% criterion and the corresponding location is already occupied by one value, the values of this pixel will replace the old one.

3. At the end of the search, if all the pixels of dynamic background are updated, the done signal will be set to true. If there are some points that are still empty, values of those pixels from the averaged background image will be copied to fill all the holes. This can occur if the traffic is very busy and if vehicles, or their shadows, occupy parts of the road during the initialisation and modelling stages.

4. If the done signal is not set to true, updating of dynamic background pixels will continue. This updating is only performed on pixels that are not updated, and it continues until all the pixels are updated.

The static background array and the dynamic background array will be updated differently, depending on the different situations with different methods. Figure 3.5 shows the modelled background image. It can be noticed that this image is not blurred in the same way as the average background image, and the intensity of each pixel is very similar to the current image. Some texture features can be also found in this new background image.

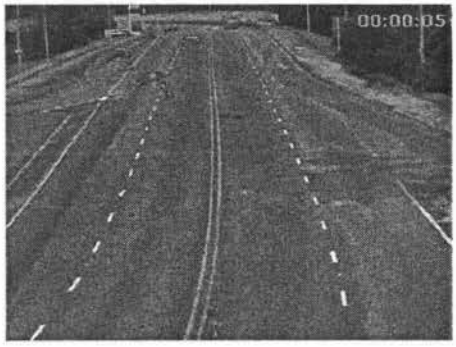



Figure 3.5 New modelled background image

The determination of updating for the static background can be achieved by observing the intensity changes from four testing squares, which have been selected by the user. Since the monitoring area of the road is relatively small, we can assume that the illumination changes are the same for the whole background. During a predefined time frame, the intensity changes of pixels in those four squares will be checked. The difference in intensity of these four squares, between the current image and the background image, is calculated. If the difference is greater than the threshold value, an update will be performed. If the difference is less then the threshold, no update occurs.

$X(I,j,t)$ can be used to represent one pixel at point $(x,y)$ at time $t$. $X(I,j,t)$ can be treated as a random process, because the background changes very slowly in an image sequence. The reason for this is that 20 frames per second are captured and changes only occur several times per 1000 frames in most time. However, $X(I,j)$ can still be considered as a random variable in a short period of time. Hence $X(I,j)$ satisfies the Gaussian distribution [94].

$$P(X(i,j)) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

We can regard the mean value $\mu$ as the feature point at point $(x,y)$ of the background, although most of the feature values have little difference with $\mu$.

If we use $L$ and $W$ to represent the length and width of the image, then background updating for channel $c$ of the $n^{th}$ frame image, and up to $N$ frames can be represented using the following equation:

$$BG(i,j,c)=\frac{1}{N}\sum_{n=1}^{N}I_n(i,j,c),\ 1\le i\le W,1\le j\le H,c\in\{R,G,B\} \tag{3.2}$$

However, this method requires large memory space and it is not suitable for real-time application, because all the last frames cannot be saved in memory for computing. Therefore, in our logarithm, averaging updating [30] is selected to update the static background, due to its effectiveness and low computational load.

$$B_{pt}=KB_{pt-1}+(1-K)C_{pt-1};\ 0<K<1\ \text{;and}\ K=\frac{N}{N+1} \tag{3.3}$$

Here $B_{pt}$ is the updated background picture, $B_{pt-1}$ is the previous background picture and $C_{pt-1}$ is the previous frame of the scene, the value of $K$ determines the rate of updating. The appropriate value of $K$ depends on the ambient lighting conditions.

An iterative method is used to update the corresponding pixels that are covered by vehicles, for dynamic background array. Those pixels covered by vehicles will be updated in the next few frames until they are not covered. The old static background is updated using the averaging method. This updating process will continue until the new background image is built. Basically, the system will check the intensity of the pixels in the testing squares once every ten minutes for twenty frames, and record the intensity result in another array. If the intensity changes, that are a measure of the ambient lighting variations between the current frame and the last frame in the last check is significant (greater than 15%), the system will consider updating the background. The system will continue to compare the intensity value in the following 15 frames and if the difference is still significant, an update will be performed. If the intensity changes are close to the last check, the background will be kept the same. The reason for this continuous comparison is because sometimes noise can produce a flicking of an unwanted signal, which can cause large intensity changes in several consecutive frames. The use of averaging updating can reduce the effect that misclassifies an object as background. Figure 3.6 shows the four test squares.

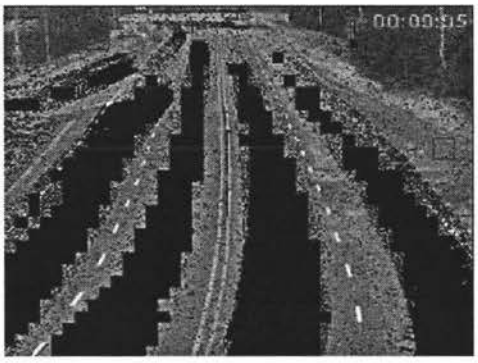


Figure 3.6 Four test squares

The predefined time is set by using the date and time information from the local area. It is convenient to obtain information from the local city council about the sunrise and sunset times and the number of sunshine hours. This information can be stored in an array. A real-time clock can be found in every PC-system. If the real-time is close to the predefined time from the array, an update check will be performed.

Furthermore, this information is also used to provide the prediction of the direction of shadows in the later section.

In order to eliminate the sudden change of brightness, we use the brightness compensation with the 'region of no moving objects' method proposed by He, Liu and Li [94]. We assume that the monitoring area is small enough, so only the brightness change in those four testing squares is checked. If there exists no disturbance of moving objects then the brightness changes of the testing squares can reflect the changes of the whole field. If we use $\mu_C$ to stand for the mean brightness of the current frame and $\mu_B$ to stand for the mean brightness of the background image, then the brightness change in the region with no moving objects is $\mu_C - \mu_B$.

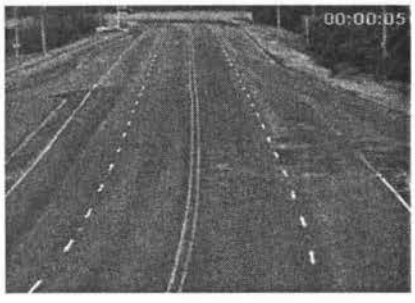
The averaging updating can be modified as:

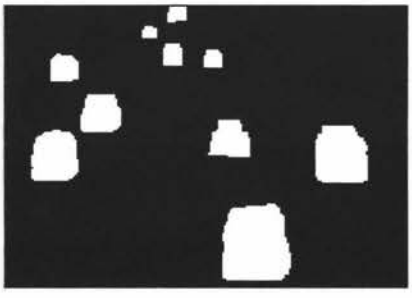
$$B_{pt} = KB_{pt-1} + (1-K)C_{pt-1} + \mu_C - \mu_B \; ; \; 0 < K < 1 \text{ ;and } K = \frac{N}{N+1} \qquad (3.4)$$

Figure 3.7 shows the result of using brightness compensation.


(A)


(B) (C)



(D) (E)

Figure 3.7 the result of vehicle detection after brightness compensation (A) original image, (B) background before using brightness compensation, (C) detected vehicles, (D) background after brightness compensation, (E) detected vehicles.

## 3.4 Background subtraction

A background subtraction method is used to subtract the background image from the current image, in order to obtain the absolute difference between them. Absolute difference means that if the intensity value of one pixel in the current image ($C(x,y)$)is greater than the corresponding pixel in the background image ($B(x,y)$), then the difference value ($D(x,y)$) is :

$D(x,y)=C(x,y)-B(x,y);$ (3.5)

On the other hand, if $B(x,y)$ is greater than $C(x,y)$, then

$D(x,y)=B(x,y)-C(x,y);$ (3.6)

In a grey scale image, this difference can be obtained by directly subtracting two grey scale values. In colour image, it is simple to acquire the values in red, green and blue channels without any transformation. Hence, in our algorithm, we analyse the chromaticity in RGB colour space and define the colour difference as

$$I_D(x,y)=\max\{|R_c(x,y)-R_b(x,y)|,|G_c(x,y)-G_b(x,y)|,|B_c(x,y)-B_b(x,y)|\} \quad (3.7)$$

Let the image sequence be $I(x, y, t)$ where

$x = 1, 2, \ldots W$

$y = 1, 2, \ldots H$

$t = 1, 2, \ldots N$

Here $W$ is the width of image, $H$ is the height of image, and $N$ is the number of frames in the sequences.

For each background image, the following parameters are calculated:

(1) Mean of the pixel intensity values:

$$\mu(x,y)=\frac{\sum_t I(x,y,t)}{N} \quad (3.8)$$

(2) Standard deviation of the pixel values:

$$\sigma(x,y)=\frac{\sqrt{\sum_t\left[I(x,y,t)-\mu(x,y)\right]^2}}{N} \quad (3.9)$$

For each image frame $f(x, y)$, the information will be separated into three channels: $f_R(x,y)$, $f_G(x,y)$, and $f_B(x,y)$. The mean and standard deviation are also separated into three channels: $\mu R(x,y)$, $\mu G(x,y)$, $\mu B(x,y)$, and $\sigma R(x,y)$, $\sigma G(x,y)$, $\sigma B(x,y)$.

$$\text{If } |R(i,j)-\mu R(i,j)| < c\times\sigma R(i,j) \quad (3.10)$$
$$|G(i,j)-\mu G(i,j)| < c\times\sigma G(i,j)$$
$$|B(i,j)-\mu B(i,j)| < c\times\sigma B(i,j)$$

At this point, the pixel is defined as a background pixel. In the above equation, c is a constant value, and the default value is set to 5 in the system. The value can be changed according to the actual situation. In our system, the average background image can be thought as the mean image $\mu$. Equation 3.10 can be treated as the difference between the current image and the average background image. Then the background pixels, foreground pixels and shadow pixels can be determined.

The remaining pixels are either foreground pixels or shadow pixels and another criterion is used to classify them [92].

$$\text{If } \begin{cases} 0.9 < \dfrac{R(i,j)-\mu R(i,j)}{G(i,j)-\mu G(i,j)} < 1.1 \\ \textit{and} \\ 0.9 < \dfrac{G(i,j)-\mu G(i,j)}{B(i,j)-\mu B(i,j)} < 1.1 \end{cases} \quad (3.11)$$

Now the pixels $(i, j)$ are defined as shadow pixels and the remaining pixels are defined as foreground pixels. The results are seen in three different channels: $R$, $G$, and $B$. These three channels will then be combined into one image and result in the corresponding binary image.

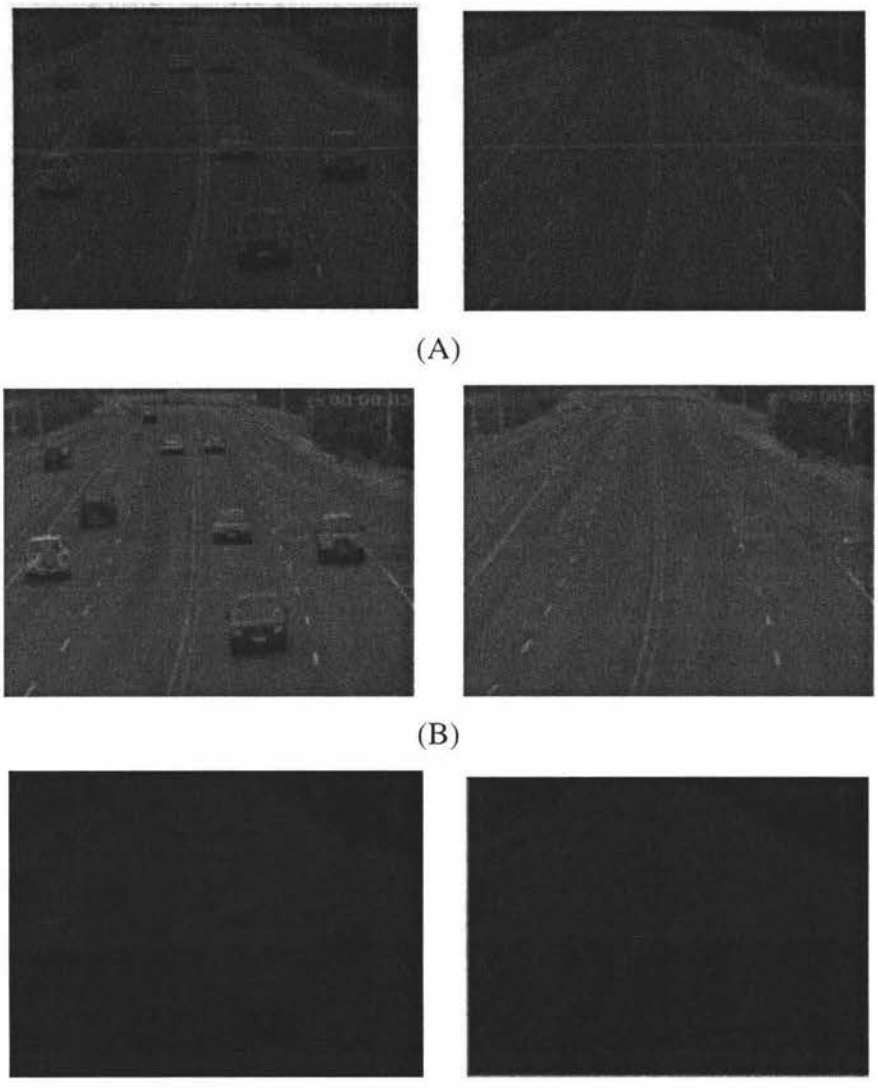
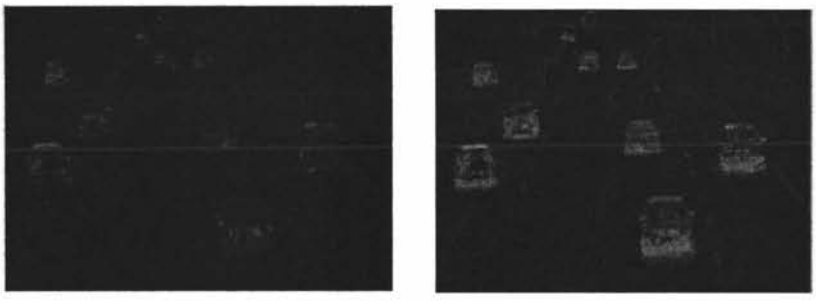


Figure 3.8 Traffic image and its background image in (A) red channel, (B) green channel, (C) blue channel

Figure 3.9 Difference image in (A) red channel, (B) green channel, (C) blue channel

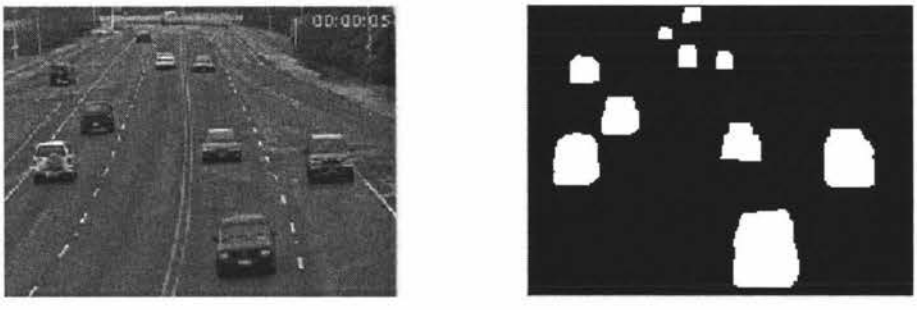


(A) (B)

Figure 3.10 (A) Combination of difference from three different channels, (B) conversion to gray scale image.

Figure 3.8 shows the three separate channels of the current image and their background image. Figure 3.9 shows the different images between the current images and their background image. It can be noted that some pixels are more significant in the red channel than in the other two channels, and some are more significant in the green or blue channels. Figure 3.10 (A) shows the combinational image from three of those channels. The system compares three values from the three channels of a pixel and selects the largest one. This can enhance the contrast between the foreground objects and background. Figure 3.10 (B) shows the converted grey scale image.

Figure 3.11 shows image 151 and the vehicles found in this image. It can be seen that only vehicles remain on the resulting image. The background modelling and updating scheme provide a satisfactory result.



(A) (B)

Figure 3.11 Result of found vehicles, (A) Image 151, (B) found vehicles

In order to secure better segmentation result, we use the method proposed by Dubuisson and Jain [60]. Figure 3.12 shows a moving ramp edge on three

consecutive frames. If a ramp edge successively moves from a position $f_1$ (in frame 1) to another position $f_3$ (in frame 3), the difference between frame 1 and frame 2 is the solid curve, and the difference between frame 2 and frame 3 is the dashed curve in figure 3.12 (B). The bold curve in figure 3.12 (B) shows only the meaningful information, where both frame differences are at the location of the ramp edge in the middle frame. If there is no motion, or if there is no strong edge at a given pixel, the difference at that pixel is not meaningful.

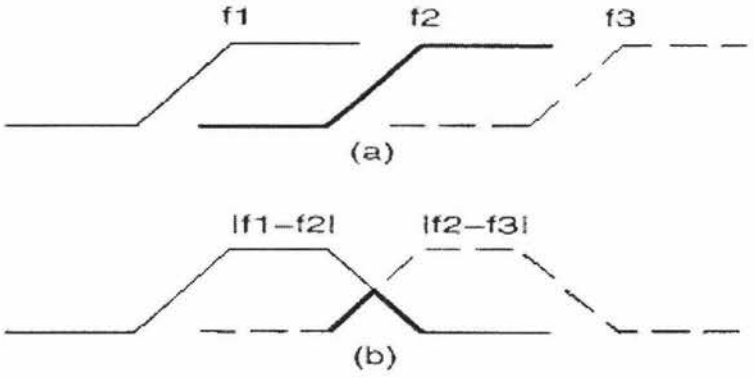


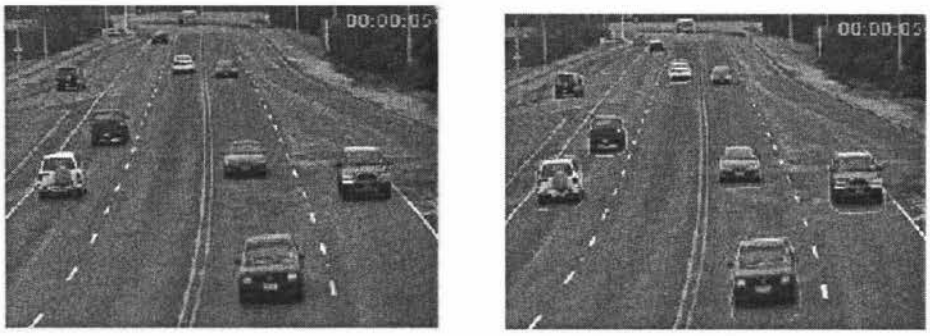

Figure 3.12 A moving ramp edge: (a) positions in frames 1, 2, and 3; (b) frame differences and extraction of the moving edge.

In their approach, the difference between frame 1, 2, and 3 are computed and the total difference is recorded.

$$d(i, j) = max[dR(i, j), dG(i, j), dB(i, j)] \tag{3.9}$$

where $dR$, $dG$, and $dB$ are the difference images. In our approach, we take the difference from three consecutive images, and compare them with the background image, in order to obtain the total difference.

Figure 3.13 (A) shows image 152, and figure 3.10 (B) shows the detected contour of all the moving vehicles. It is clear that all vehicles are enclosed by the contour.



(A) (B)

Figure 3.13 (A) image 152 (B) detected contours of all moving vehicles.

In our approach, some morphological operations are used, such as 'top-hat' and 'bottom-hat' operations, in order to enhance the contrast. An edge detector is also used to discover the edges of moving vehicles. These edges can be used to enhance the results of vehicles detection.

Opening and closing are two basic morphological operations in image processing. The opening module performs a dilation routine that grows the current white image, followed by an erosion routine that shrinks the current white image. The closing module performs an erosion routine followed by a dilation routine. The opening operation can separate objects that are connected in a binary image, while the closing operation can fill in small holes.

'top-hat' and 'bottom-hat' are two types of morphological operations. 'top-hat' image contains peak of objects and 'bottom-hat' transforms image shows the gap between objects. 'top-hat' filtering is the equivalent of subtracting the result of performing a morphological opening operation on the input image from the input image itself, which results in the display of just the pixels that close the object. 'bottom-hat' filtering is the equivalent of subtracting the result of performing a morphological closing operation on the input image from the input image itself, which results in the display of just the connection points between close objects. 'top-hat' and 'bottom-hat' are often used together to enhance the contrast of an image by adding the original image to the 'top-hat' filtered image, and then subtracting the 'bottom-hat' filtered image.

Edge detection is one of the most useful low-level image processing techniques and it is used as the first step in many computer vision applications. Edges can be defined as a 'discontinuity' in some image attributes and usually they are the brightness in grey-scale images. Edges are treated as features that can be detected without knowledge of the objects in the world that caused them. The use of edge detection can reduce the amount of data to be processed and preserve useful structure information relating to the objects' boundaries. Well-known edge detection methods for grey scale images, such as the Sobel gradient, the Laplacian operator and the Canny edge detector, are widely used.

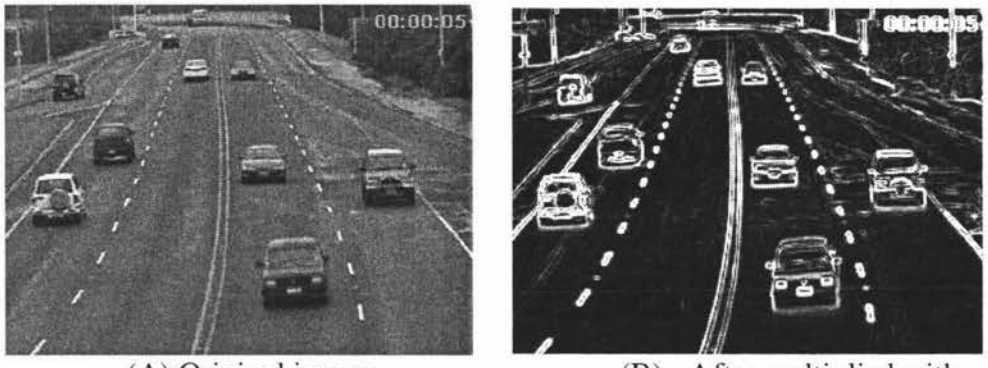
The Sobel edge detector is chosen in our approach. Because the following reasons. The Canny edge detector requires smooth procedure by Gaussian filter, which makes the intensity difference between objects and background smaller. The Canny edge detector uses the local maxima of gradient values and allows some edges to describe the inner details. However, we do not require these inner details in vehicle extraction. Laplacian edge detector is highly sensitive to the background noise. When noise occurs in the image, it becomes useless because noise interferes the detection of objects. Our application is outdoor scenes, noise cannot be avoided.

The Sobel operator performs a 2-D spatial gradient measurement on an image, so it emphasizes regions of high spatial gradient that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input gray scale image. The Sobel edge detector uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. Figure 3.14 shows the result of the original image, multiplied by Sobel approximation.

The adaptive thresholding method is used to secure the threshold level, before edge detection and segmentation. The conventional thresholding method sets a threshold value for all intensity values. This is not a sufficient method because it cannot solve the illumination changes. The adaptive thresholding method we used in our algorithm changes the threshold value dynamically over the image. In an actual implementation,

the threshold level is set 10% lower than the computed value. The reason for this is that using a lower value can preserve more edge information, so that more completed edges can be obtained.

On the other hand, we use open operation followed by hole-filling, which first disconnect jointed isolated pixels, and then eliminate them. Hole-filling fills holes in the grayscale image. A hole is defiend as an area of dark pixels surrounded by lighter pixels. This function brings the intensity values of dark areas that are surrounded by lighter areas up to the same intensity level as surrounding pixels. Actually, it removes regional minima that are not connected to the image border. The intensity values in an image can be thought as the elevations in the topographical map. The areas of high intensity and low intensity in an image can be thought as peaks and valleys in topographical terms. These peaks and valleys are the maxima and minima of an image. An image can have multiple regional maxima or minima but only a single global maximum or minimum.




(A) Original image

(B) After multiplied with Sobel approximation

Figure 3.14 the result of original image multiplied by Sobel approximation

## 3.5 Snake operation

In order to secure better contours that bound detected objects tightly, we first tried to use a watershed segmentation method to get watershed lines as a reference contour. Then we tried to use an energy minimisation method to force the contour to move towards objects. Figure 3.15 shows the rough watershed segmentation result.

As mentioned before, each gray level image contains morphological gradients, which can be thought as a topographical surface. The watershed algorithm is an approach that uses mathematical morphology method to divide this topographical surface in different catchments basins by the watershed lines. There are two classical methods to achieve watershed segmentation: the rainfalling simulations and the immersion or flooding simulations. The solution we selected is the watershed based on immersion simulation. The main drawbacks of the watershed algorithms are their high sensitivity to noise, resulting in an oversegmentation and the contours are not smooth that make tracking difficult [90].

In an image, every small fluctuation in intensity represents a regional minimum or maximum. However, only the significant minima or maxima are useful in segmentation not the smaller minima or maxima that are caused by texture. The minima of an image can be obtained by finding the valleys of all the connected areas in intensities. This can be done by looking for the smallest value in every connected areas, which is the regional minima. In order to solve the oversegmentation, we remove the minima that are too shallow, by suppressing all minima in the intensity image, whose depth is less than a threshold value.

A Gaussian low pass filter is applied follow the suppression of minima to blur the image, so that the sensitivity can be reduced. This is equivalent to convolve the image with a Gaussian or normal distribution. After the convolution, each pixel's value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value), and neighboring pixels receive smaller weights as their distance to the original pixel increases. The Gaussian low pass filter can preserve the boundaries and edges well while blur the texture information.

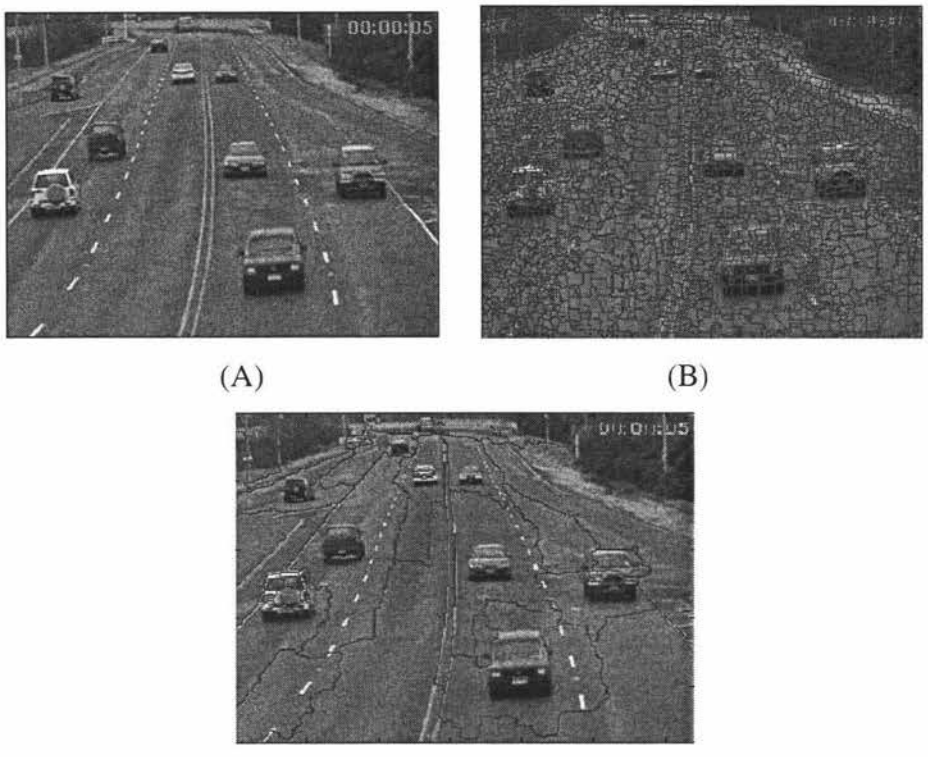


Figure 3.15 Watershed segmentation on traffic image (A) original image (B) Oversegmentation (C) Watershed after suppressing all minima and passing low pass filter

However, after testing the results of our vehicle detection approach, we found that the detected contours were sufficient enough to perform energy minimisation, in order to obtain better contours. Hence, in our algorithm, we only use a snake operation [84 - 88].

After obtaining the rough contours of the vehicles, the snake algorithm is performed on the rough contours to get better results. Unlike watershed, snake is the energy-based approach, which obtains segmentation through minimizing the energy function. Two major terms are involved in this approach, namely data-driven and regularization. The data-driven term can be classified into two classes: contour-based (also called snake-based) and region-based. The regularization term can impose the priori knowledge to make the segmentation result smoother.

The active contour model can be represented as a vector $V(s) = (x(s), y(s))$, here s is the arc length. A contour can be represented as:

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{im}(v(s)) + E_{con}(v(s))ds \tag{3.10}$$

$E_{int}$ is the internal energy of the contour due to bending or discontinuities. $E_{im}$ is the image forces that due to various events such as edges or terminations. $E_{con}$ is the external constraints.

The internal energy can be represented as:

$$E_{int} = (\alpha(s)\,|\,v_s(s)\,|^2 + \beta(s)\,|\,v_{ss}(s)\,|^2)/2 \tag{3.11}$$

The first term is curve, and the second term is continuity. The value of $\alpha$ and $\beta$ at a point determine the extent of contour such as stretching or bending at that point. If $\alpha$ is 0, a discontinuity can occur, and if $\beta$ is 0, a corner can develop.

The snake algorithm we developed is based on Williams and Shah's fast algorithm [88] used to calculate snake. This algorithm is a greedy algorithm that can reduce the computation load to $O(nm)$ and allows the inclusion of hard constraint described by Amini, Tehrani, and Weymouth [86].

In this approach, the quantity being minimized can be written as

$$E = \int(\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds \tag{3.12}$$

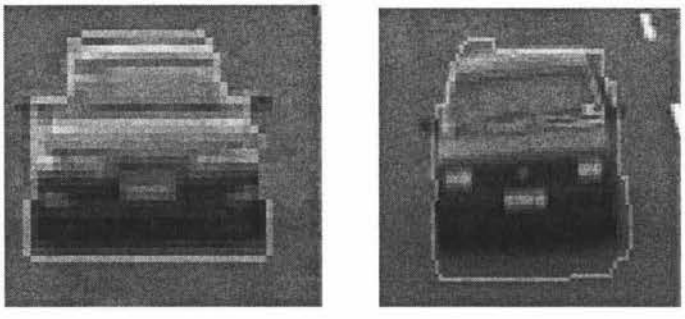
The first and second terms are the first-order and second-order continuity constraints, which are corresponded to the internal energy. $E_{image}$ is the image quantity such as edge strength. The external energy can be included, but not considered here.

The first term can be calculated as $|V_i - V_{i-1}|^2$, which can cause the curve to shrink. The second term can be obtained using formula $|V_{i-1} - 2V_i + V_{i+1}|^2$. This continuity term can force the points to be relatively evenly spaces. The third term $E_{image}$ is the image force that is the gradient magnitude (mag). After finding all 8 neighbour gradients, the maximum (max) and the minimum (min) gradients are

determined. Then (min-mag) / (max-min) is used for the normalized edge strength term. More details can be found in section 2.12.2.

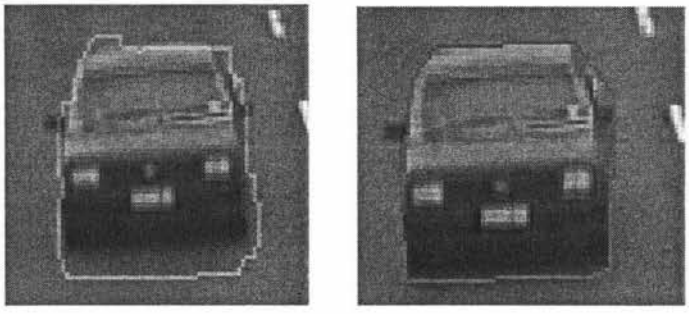
This algorithm only goes one pixel inside in every iteration. Therefore, in some situations, the iteration time will be long. But on the other hand, some contours are already well bound objects and they only require one or two iterations to secure better contours. In our algorithm, we process vehicles one by one, which ensures every vehicle is processed successfully and no iterations are wasted. Figure 3.16 shows the good result contour and bad result contour before the snake algorithm. Figure 3.17 shows the result of the snake on the contour of a vehicle. It can be seen that, after the snake operation, the car is nearly all bounded by the contour with only a small amount of non-useful information such as the background. These contours can then be used for tracking.

The iteration time can be set by the user or adjusted by the system itself. In our algorithm, if 90% of the points on the contour could not move further inside by one pixel, the system will stop the snake operation and output the contour. This can then reduce the computational load.



(A) (B)

Figure 3.16 Examples of good contour and bad contour, (A) good contour (B) bad contour



(A) (B)

Figure 3.17 One example of contour after snake, (A) original contour (B) contour after snake

## 3.6 Summary

A modified background modelling method that separates the background image into a static and a dynamic background and updates them separately is proposed in this chapter. The original background image is built using the first 200 images by getting the average image of them. The separation can significantly reduce the computational load of the updating for the whole background image. Updating is conducted by continuously using the pixels in current image to replace the corresponding pixels in the original background image. The requirement of background updating is checked through 4 test squares.

Dubuisson and Jain's approach [60] is used to get the maximum useful difference through 3 continuous images. The criterion of background or foreground pixels is based on the maximum difference from the three base channels in red, green and blue. Once the contours that represent the detected objects are obtained, snake operation is used to refine these contours so that they can represent vehicles more accurately.

Some useful image processing techniques such as opening, closing, edge detection, bottom-hat and top-hat that are used to enhance the results of modelling and updating, are also introduced in this chapter.

In the next chapter, we introduce the methods we used in illumination analysis and shadow handler, such as road structure extraction, the determination of illumination

and the directions of shadows, and how these are applied to eliminate shadows using shadow seeds methods. The result of shadow handler can provide good quality images for further processing such as tracking, counting and classification.

# 4. Illumination analysis and shadow handler

Shadows bring many problems in processing traffic images, such as causing partial occlusion, or joining several vehicles into one. If we cannot separate vehicles from shadows, subsequent processes on vehicle images become more difficult. Illumination analysis can obtain the directions of light sources, for example, sunshine. Once the direction of a light source is known, operations on shadow areas can be performed to get the locations of the shadows. A shadow handler can then separate the vehicles from their shadows.

The remaining sections of this chapter will discuss the methods we use in illumination analysis and the shadow handler.

## 4.1 Road structure extraction

The tracking area in our algorithm is only the middle 80% of the scene. Therefore, only the road structure of that part of scene is required to be determined. We only consider lane dividing lines on the scene and we assume that most of the vehicles are driving within the area between two lane lines, and therefore only parts of the body will cross these lines. If shadows are detected over the line, then they will be eliminated.

Firstly, a gradient-based edge detection (Sobel edge detector in our algorithm) is used to find the road structures in the image. Since road structure components have rich edge information, whilst the road surface does not, this process can extract only the structure. These detected edges are connected together, in order to obtain the complete lane lines using different morphological operations, such as opening, closing, spurring etc. These plausible lines can separate road surface into different parts. Figure 4.1 shows one example of lane separation. In figure 4.1 (D), it can be seen that only the lane lines remain in the scene.

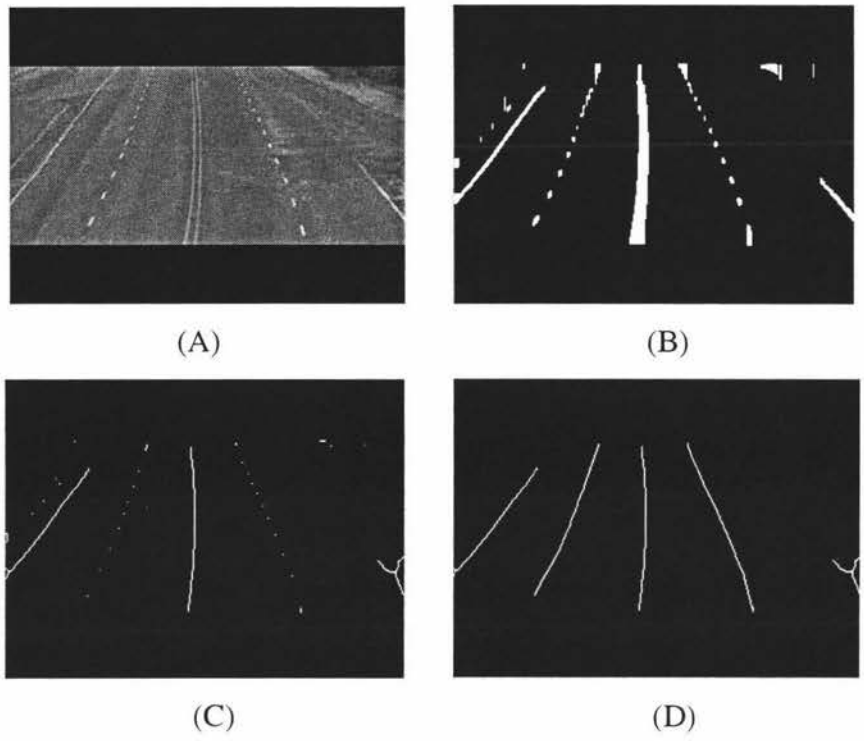



Figure 4.1 Road structure extraction: (A) background image, (B) detected lane lines (C) main lines and dots representing lines, (D) connected components

## 4.2 Illumination analysis

Shadows' direction and position vary with the direction of the light source. For example, shadows are at the left hand side of the vehicles in the morning, in front of the vehicles at noon, and at the right hand side in the afternoon. If we know the direction of the light source, we can know the position of shadows. On the other hand, if we analyse shadows, we can know the position of shadows and the direction of light source. Once we determine the position of shadows, we only need to perform shadow separation on that direction, which can then reduce the computational load.

We define eight directions in our algorithm, which is east, south east, south, south west, west, north west, north and north east. We assume that the monitoring area is small enough so that the directions of light source for all vehicles in the testing window are the same. Therefore, we only select two vehicles from the left lane and two vehicles from the right lane, in order to discover the direction of the light source.

Through averaging of the first 200 images, a rough background model is obtained. We then use a background subtraction method to get several foreground vehicles. These vehicles, including their shadows, will be cropped out.

The first step is to determine whether shadows are in existence. We define the length and width for several types of vehicles, such as sedans and trucks. The length, width, area and aspect ratios for all detected vehicles are calculated. If a vehicle moves regularly, its centre will be very close to the centre of the lane that it moves on, and most of its body will stay in the lane. Therefore, for each blob, if any parts of it occupy the area across the lane lines, these parts will be cut off. Parts of the blob that cross the lines can be caused by shadow or occlusion by other vehicles. However, through the use of lane lines, we only concern about blobs within the lane.

All vehicles within the lanes, through the combination of area and aspect ratio, are roughly classified into several categories. In each category, if the width for one blob is longer than the threshold level, shadow may occur. Then the aspect ratios of these vehicles are compared with the defined values. If the blob is irregular, the aspect ratio is different from the predefined one.

Figure 4.2 shows one example of a vehicle with shadow. The area of this blob is 1157, which is within the range of sedans. The left part of the blob crosses the lane line, which is part of its body, and the right part of the blob crosses the lane line, which is part of its shadow and both of them are cut off. Then the aspect ratio of this is calculated, which is 1.5319. This value is greater than the defined value for a sedan which is 1.3 and hence shadows may occur.

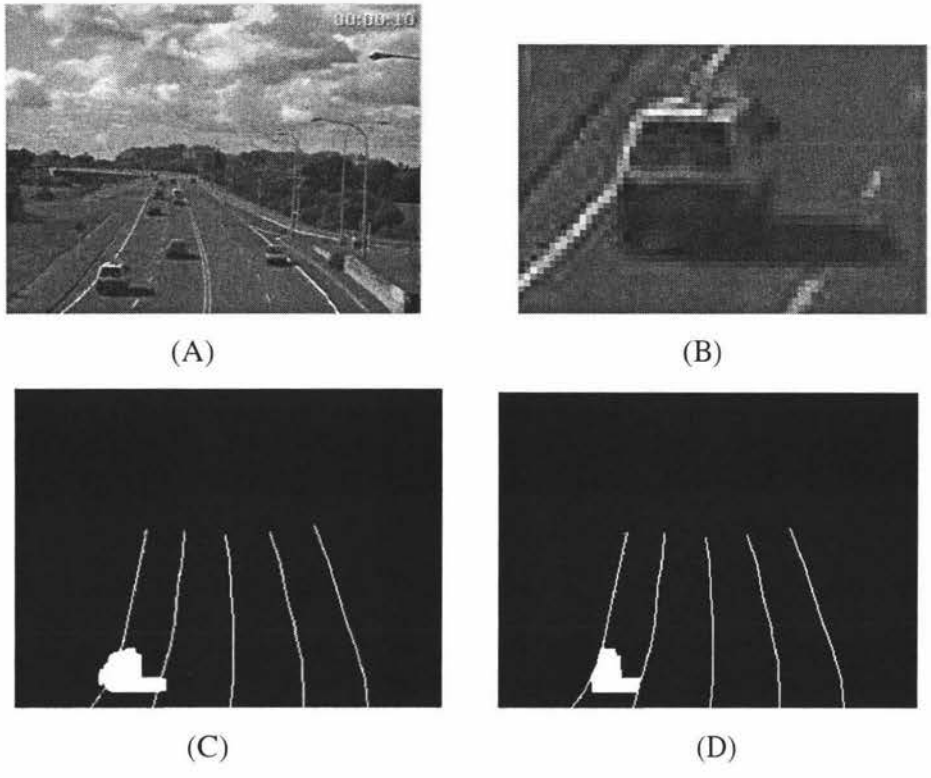


Figure 4.2 One example of shadow determination. (A) Original image, (B) enlarged detected vehicle, (C) blob with detected lane lines, (D) blob after cutting unwanted parts.

A Sobel edge detector is applied to detect the edges of the cropped images. Shadows contain fewer edges than vehicles. The direction of shadows can be defined in two steps. Firstly, shadows are often narrower than vehicles. The shape of each blob is checked. If one part of the blob is significantly narrower than others, shadow may occur. Secondly, the edge points are counted column by column and also row by row. In the shadow area, there are only two or three edge points, but several in the vehicle area.

Figure 4.3 shows one example of the illumination analysis. From 4.3 (D), we can see that the right part of the blob is significantly narrower than the left part of the blob. When edge points' counting is operating, there are only two points on the right part area. Hence, the direction of the illumination can be defined as left. In the following time period, only the right shadow will be considered.

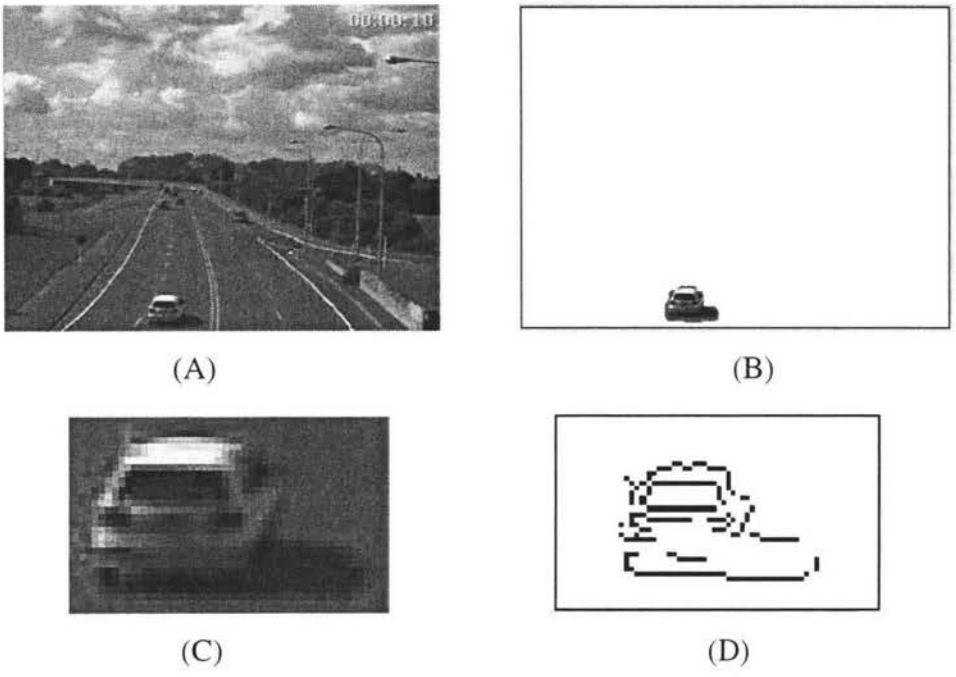


Figure 4.3 One example of illumination analysis (A) original image, (B) detected vehicle, (C) enlarged detected vehicle, (D) blob of vehicle with shadow

## 4.3 Shadow handler

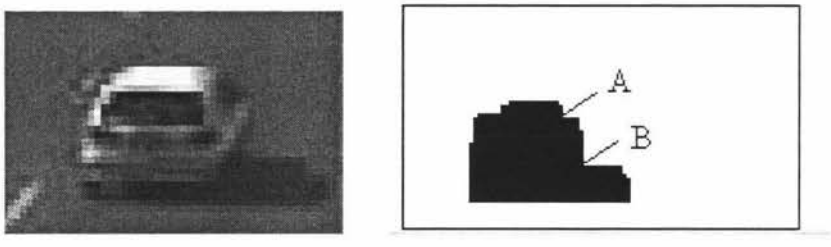
A shadow handler is used to detect and remove shadows in traffic image processing, particularly on sunny days. A traffic image may contain vehicles with or without shadows. Another situation is multiple vehicles connected by shadows.

We perform a road structure extraction to obtain the road structure relating to the scene. This information can be used to separate shadows over lane dividing lines. Through illumination analysis, we can decide whether shadows exist within the monitoring area. If there are no shadows within the scene, a shadow handler is not required. If shadows are found, a method is used to determine the direction and orientation of the shadows, and the corresponding background pixels are used to replace the shadow pixels.

For a detected vehicle with shadow, with knowledge of the illumination analysis, we only perform shadow handling on the shadow side. For example, if the direction of the light source is determined from the left hand side, the shadow handler only operates on the right hand side of the blob.

If shadows occur on the right hand side of vehicles, for each blob, a bounding box is used to bind it with 3 pixels away from the contour at the left hand and right hand side. The distances from the right boundary of the bounding box to the right side of the vehicle contour are calculated. These distances are stored in an array. These values will be compared with each two consecutive pixels. If the difference between two adjacent points is greater than 5 pixels, the latter point is defined as a 'candidate seed for shadow'.

Figure 4.4 shows one example of candidate seeds of shadow. In 4.4 (B), two seeds A and B are found. One of them is the true seed of the shadow.




(A) (B)

Figure 4.4 candidate seeds of shadow detection (A) original image, (B) detected shadow seeds.

In [95], the authors suggest a method to identify the false shadow seeds. We use their method to find out the true shadow seeds.

In the shadow regions, the red colour component decreases and blue colour component increases. Hence, the normalised red and blue colour components are used as features.

$$r = \frac{R}{(R+G+B)} \tag{4.1}$$

$$b = \frac{B}{(R+G+B)} \tag{4.2}$$

Normally, the intensity of background is higher than the shadow in the same region, and the relationship between background and vehicles is uncertain. Therefore, the luminance is calculated by the average of those three colour components.

$$L = \frac{(R+G+B)}{3} \tag{4.3}$$

After calculation, three features of the candidate seeds ($b$, $r$, $L$) are compared with the corresponding points in the background image. Through comparison, true shadow seeds are found.

Once all true shadow seeds are determined, the mean value of these seeds in red, green and blue are calculated. For each neighbour point of the seed points, if the difference of the mean values of these points in each colour component is less than 10% of the corresponding seed points, it is defined as shadow point. This procedure will iterate until all shadow points are obtained. These points are then merged together to form the shadow region.

As mentioned in section 3.4, shadow pixels can use another criterion for classification [92]:

$$\text{If} \begin{cases} 0.9 < \dfrac{R(i,j) - \mu R(i,j)}{G(i,j) - \mu G(i,j)} < 1.1 \\ \textit{and} \\ 0.9 < \dfrac{G(i,j) - \mu G(i,j)}{B(i,j) - \mu B(i,j)} < 1.1 \end{cases} \tag{4.4}$$

Then the pixels (i, j) are defined as shadow pixels.

This calculation is performed through the background subtraction. All the points receiving values satisfy the above criterion are recorded in an array. Once the shadow pixels are obtained, they will be double checked with the pixels found through background subtraction.

If all the shadow pixels are determined, these pixels will be replaced by the corresponding pixels from the background image. Figure 4.5 shows one example of removing shadow from one vehicle.

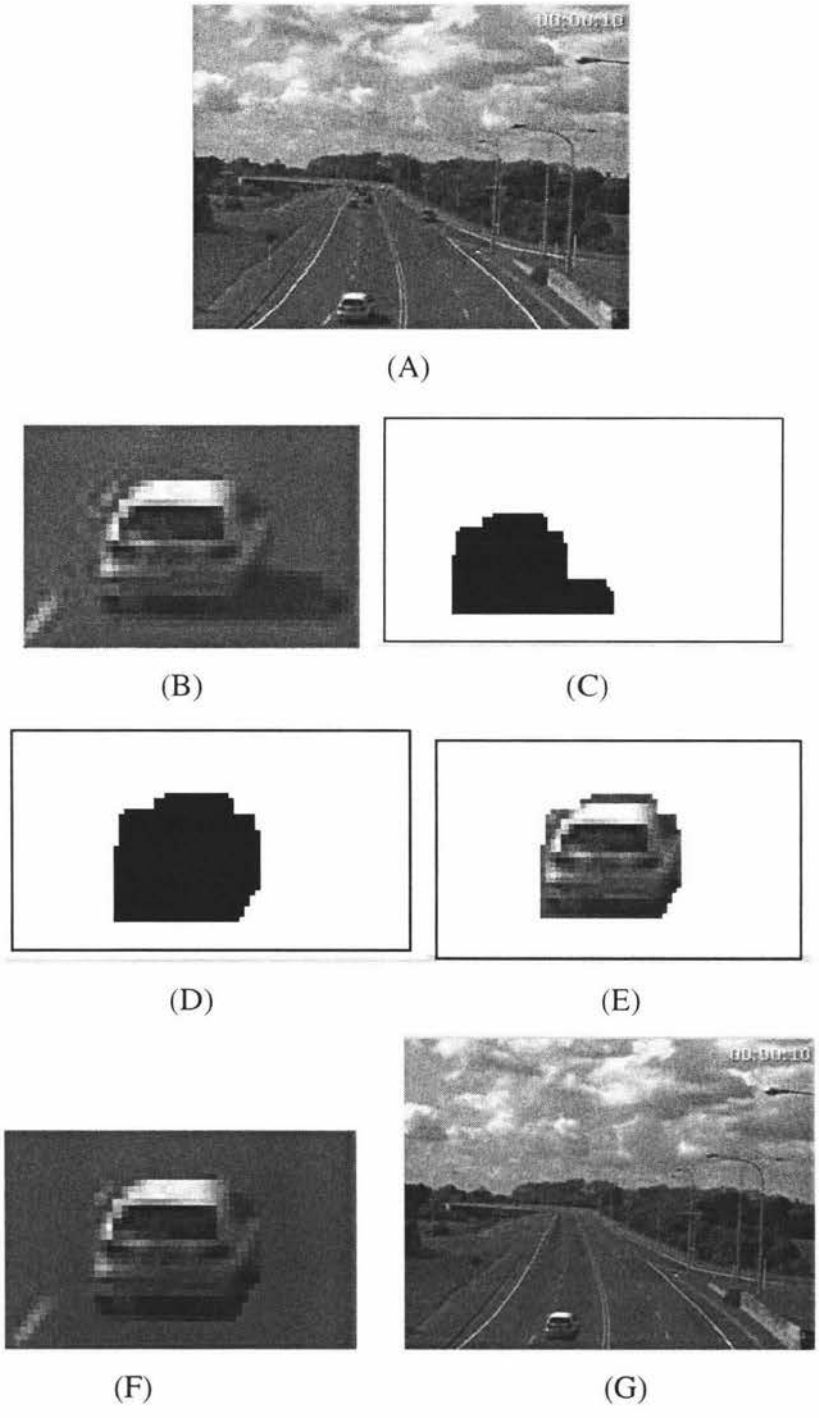


Figure 4.5 One example of shadow removal (A) original image, (B) detected vehicle with shadow, (C) blob, (D) blob without shadow, (E) mask image, (F) & (G) vehicle without shadow on background image.

## 4.4 Summary

Road structures can provide useful information to separate and eliminate shadows, especially when shadows are cross the lane lines. Road structures can be found by using edge detector since they have rich edge information.

Illumination analysis can be used to define the directions of shadows. Once the direction and position of the shadows are confirmed, shadow separation is performed only on those shadows.

Shadow handler uses the information obtained from road structures and illumination analysis to cut irregular shadows firstly. After that, candidate seeds for a shadow are detected. By using the colour information features, true shadow seeds are determined. The neighbour points will be compared with true shadow seeds to find out whether they are shadow points or not.

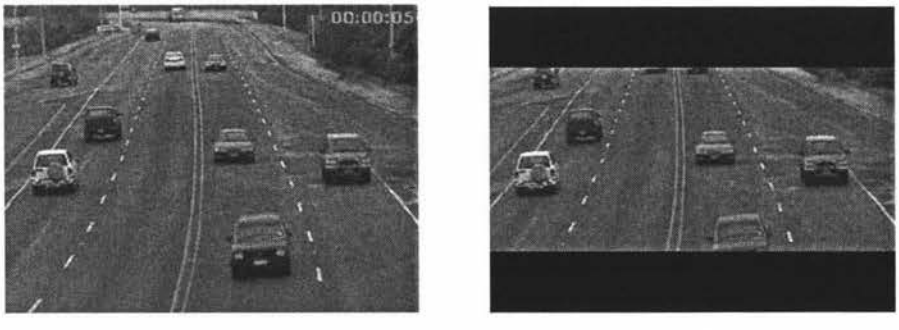
In the next chapter, we describe the methods we use in vehicle tracking, counting and classification.

# 5. Vehicle tracking, counting, and classification

After vehicle detection, there are a number of vehicles with a closed contour which are presented as 'output'. In order to obtain traffic information such as the number of vehicles, the velocity, the acceleration and the types of vehicles, tracking has to be performed first. Tracking can provide useful information and help the system to avoid multiple counting on one vehicle, and this increases the accuracy of the following operations. Based on the results of tracking, further analysis can be employed. The subsequent sections of this chapter will discuss how we track, count, and classify vehicles.

There are two problems need to be considered before tracking. The first one is calibration. Vehicles appear small when they are far away from a camera and large when they are close to the camera. Different scales will cause a tracking trajectory that uses coordinates of centroids or bounding boxes, to be curved. This is not the actual trajectory that the vehicles pass through. The second problem is the area of the tracking window. Tracking will produce a lot of information and require a great deal of storage space. Therefore, we need to reduce the size of the tracking window.

The middle 60% or 80% of the images are taken as the testing window for tracking, as shown in figure 5.1




(A) (B)

Figure 5.1 (A) original window (B) tracking window

## 5.1 Occlusion separation

During tracking, occlusion may occur due to shadows or several vehicles partially occluding with each other.

A vehicle can disappear in the scene because of the following reasons.

Firstly, the vehicle leaves the tracking area. The connection between it in the current frame and that in previous frames is lost. Hence it is no longer visible.

Secondly, the colour and intensity of the car window is sometimes close to that of the road. Also, vehicles are shiny metallic objects and the pattern of reflection captured by the camera can change. When segmentation is performed, some foreground pixels will be misclassified as background pixels. In some situations, only a few pixels stay in the window that presents small vehicles. They are ignored by the system in the subsequent operation. Figure 5.2 shows an example of misclassification of the pixels in a tracking area as background pixels. We can see in (B), the top part of the car is totally separated from the car body as a single object and it is ignored in the next the step, due to the size limit by the system. Hence, in (C), the final contour of the vehicle is only part of the car's body. In some worse situations, the vehicle might disappear, even though it is still visible in the field. Sometimes, those two separate regions stay in the field, and they will be detected later as one region. This also causes a 'missing region' problem.

Thirdly, a vehicle is occluded by some parts of the background or other vehicles and during segmentation they are detected as one single region. Figure 5.3 shows an example of partial occlusion. It can be seen that in (A), two vehicles are running so close, during segmentation they are treated as one moving object in (B).

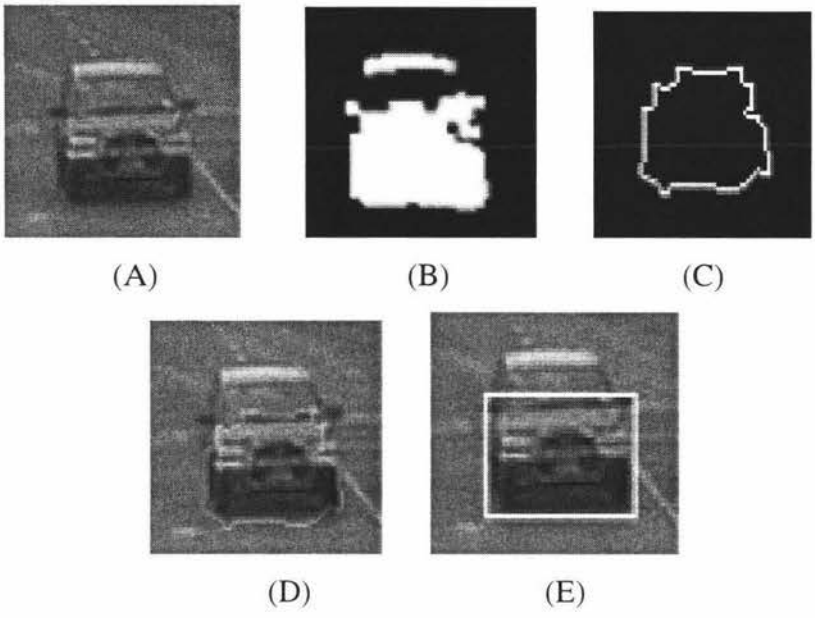



Figure 5.2 an example of misclassification due to the intensity of window of vehicle close to the intensity of road. (A) original image, (B) detected region, (C) final contour, (D) contour on vehicle, (E) refined bounding box

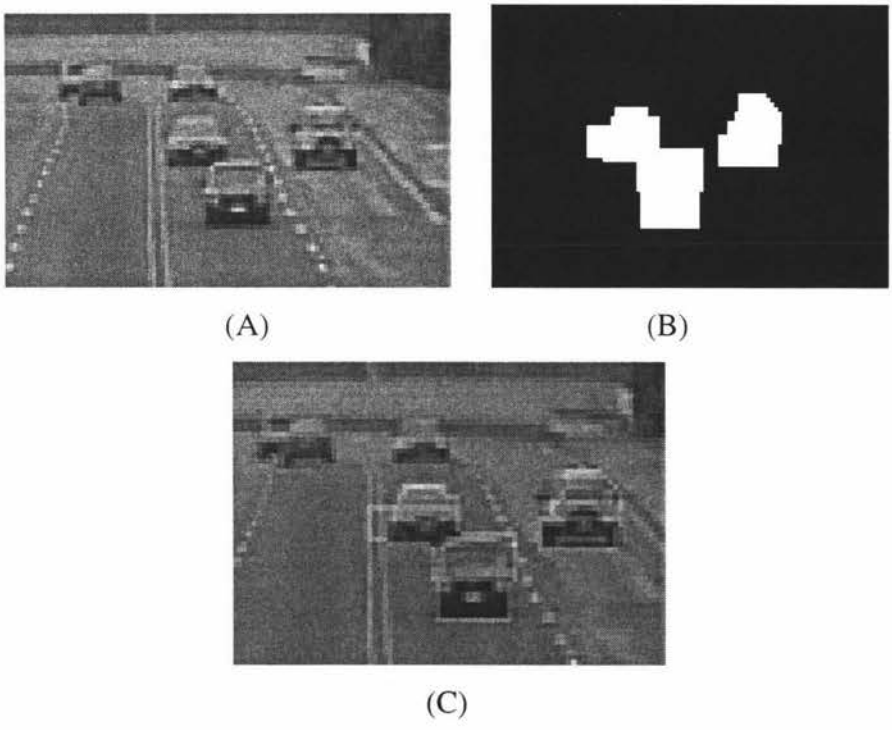


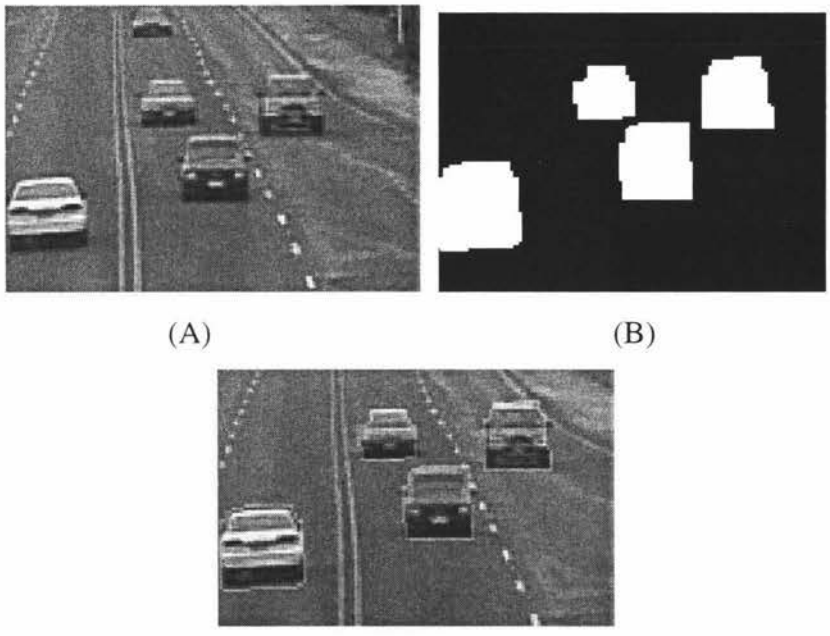

Figure 5.3 a partial occlusion example (A) original image, (B) detected partial occlusion region, (C) final contour on vehicles.

A vehicle can appear in the monitoring area due to the following reasons.

Firstly, a new vehicle enters the tracking area and the system recognises it and tracks it.

Secondly, (for the same reasons as mentioned before) due to the intensity similarity of window areas and reflections of car bodies, some misclassification pixels are detected again or the separated parts are joined again.

Thirdly, occluded vehicles are again separated into two or more individual regions. Figure 5.4 shows one of the later frames of the occluded vehicles in figure 5.3. It can be noted that those two occluded vehicles are separated into two individual vehicles.



(A) (B)

(C)

Figure 5.4 an example of separation of two occluded vehicles after few frames. (A) original image, (B) detected regions, (C) final contours on vehicles.

A tracking algorithm has to be able to solve all types of occlusion problems, otherwise miscount or misclassification problems will occur.

The algorithm we developed considers the traffic flow in two directions, one coming towards the camera, and the other moving away from the camera. This can be done

by comparing the recorded coordinates, e.g., if the $y$ coordinates are increasing in the last three frames, it can be defined as going towards the camera.

During the tracking process, the coordinates of vehicles are recorded. Once vehicles coming towards the camera are near the boundary of the tracking area, and they suddenly disappear, they are either leaving the tracking area or occluding with other vehicles.

Once vehicles are moving close to the camera, which is at the bottom 1/3 part of the field, the areas they occupy are large, and the distance between them will be quite significant. Therefore occlusion occurs infrequently. If one vehicle on the lanes towards the camera is missing, the system will search for all vehicles close to it. If all of them are found, this vehicle can be defined as a left vehicle. In the lanes away from the camera, if a new vehicle appears, it will be treated as a new vehicle entering the field.

If missing vehicles could not be identified, separation will be performed. During tracking, two things are checked: one is the area of all existing vehicles and if there is one object's area greater than the threshold, occlusion may happen; the second one is the trajectories of vehicles. If the centroids and the top left corner of the bounding boxes of two or more vehicles are becoming very close to each other, occlusion may happen. The occlusion area in the last five frames is then examined. If the coordinates are changed significantly in a horizontal way, a horizontal occlusion occurs, otherwise it is a vertical occlusion.

.

We modify Rad and Jamzad's method [40] to separate occluded vehicles. For a horizontal occlusion, we add a bounding box that is 3 pixels away from the detected contour at the top and bottom sides. For a vertical occlusion, we add a bounding box that is 3 pixels away from the detected contour at the left and right sides. For a horizontal occlusion, the vertical distance from the top and bottom sides of the bounding box to the contour is calculated. For a vertical occlusion, the horizontal distance from the left and right sides of the bounding box to the contour is calculated. These distances will be stored in two arrays A (top or left) and B (bottom or right).

Another array, C, is used to store the result as $C(i)=A(i)+B(i)$, here $i = 1, 2, 3\ldots, n$. $n$ is the total number of points on the top and bottom of the contour.

Since occlusion often occurs in the middle of the joint blob, if the maximum value is found near the boundary of the blob, it is ignored. The maximum value of C, in the middle part of the blob, indicates that occlusion may happen at that point. The occlusion is not actually separated. The corresponding coordinates of regions are recorded as the required coordinates in the tracking database. Figure 5.5 shows one example of vertical occlusion. In (D), the vertical occlusion is found. The distance of this occlusion is calculated and the coordinates of splitting points are recorded.

If one vehicle disappears, the occlusion will be checked and the same separation method will be used. In the next frame, if the occlusion still occurs, the same method will be used until no occlusion exists.

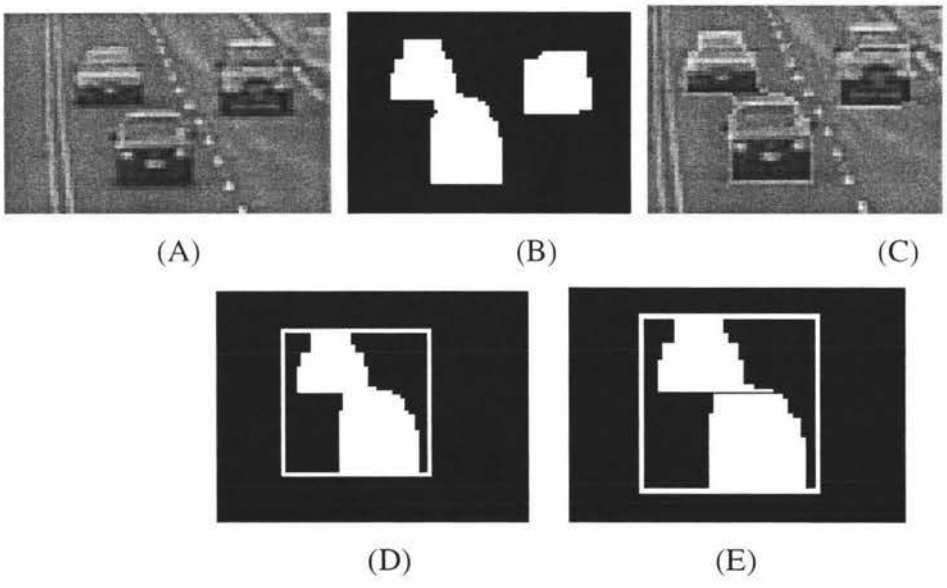



Figure 5.5 an example of vertical occlusion separation (A) original image, (B) detected occluded region, (C) contour on vehicles, (D) 3 pixels more wide bounding box, (E) the red line indicates the separated point

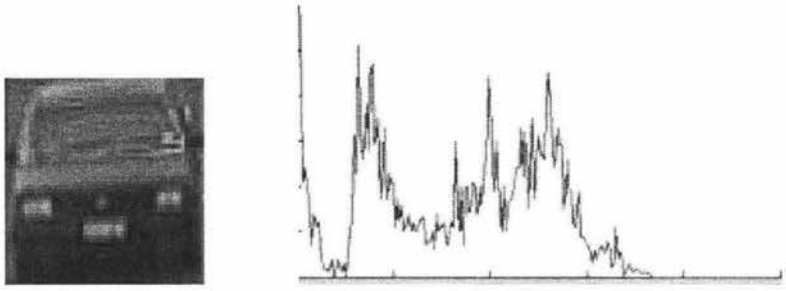
During tracking, information stored in the tracking database will also be frequently checked, in order to find the best matching between different regions. This can provide useful information to identify vehicles. To make separation robust, we add another useful property to this algorithm, which is the pixel distribution of the blobs.

The pixel distribution of the blobs (also called a histogram) in several consecutive frames is used to distinguish several vehicles that are probably occluded. Firstly, the detected vehicles are re-sampled to the size of a 55 x 55 image, which can make the comparison more meaningful. We construct the histograms for foreground objects and find the best matching one for each blob.

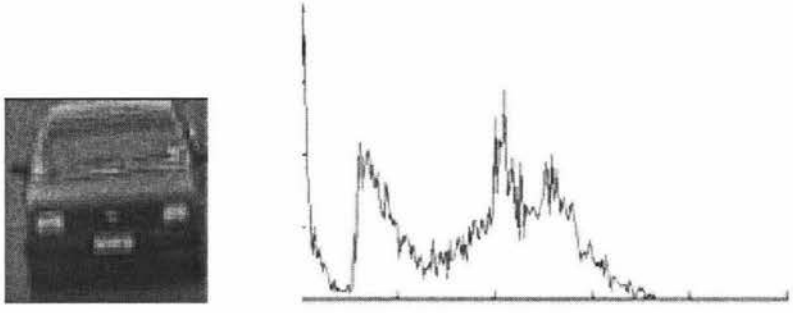
A histogram is the graphical version of a table which shows what proportion of cases fall into each of several or many specified categories. The categories are usually specified as nonoverlapping intervals of some variables and are adjacent. In the histogram, the $x$-axis is the intensity level from 0 to 255, while the $y$-axis is the number of pixels that are found at that level. For example, there are 56 pixels with intensity level 102, then the value for $y$-axis at point 56 on $x$-axis is 102. The algorithm counts the number of pixels on each certain intensity level and store them in an array. Then the histogram is drawn using this array.

Figure 5.6 shows the histogram of one vehicle in frame 152 and 153. Figure 5.6 (C) shows the matching of these two histograms. The green line in 5.6 (C) is the average histogram for 10 consecutive frames. We can see that the histogram for one car is very similar in different frames. Through a matching process, the vehicles with similar histograms can be treated as one vehicle in different frames.

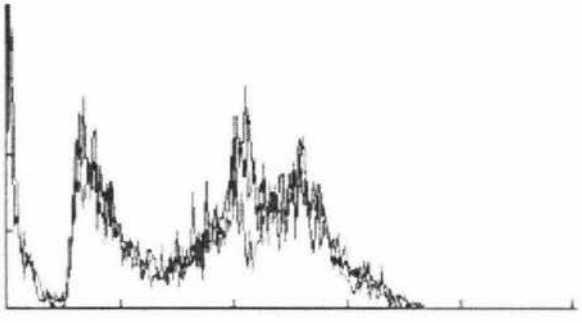
Figure 5.7 shows the histograms for three different vehicles: (A) a white car from frame 025, (B) a grey car from frame 152, and (C) a red car from frame 195. The histograms for these three vehicles are totally different from each other. Therefore, the histogram can be used to distinguish different vehicles



(A)



(B)



(C)

Figure 5.6Histogram diagram and the matching diagram (A) vehicle in frame 152 and its histogram (B) vehicle in frame 153 and its histogram (C) two histogram diagrams compared with the average histogram for 10 consecutive frames.

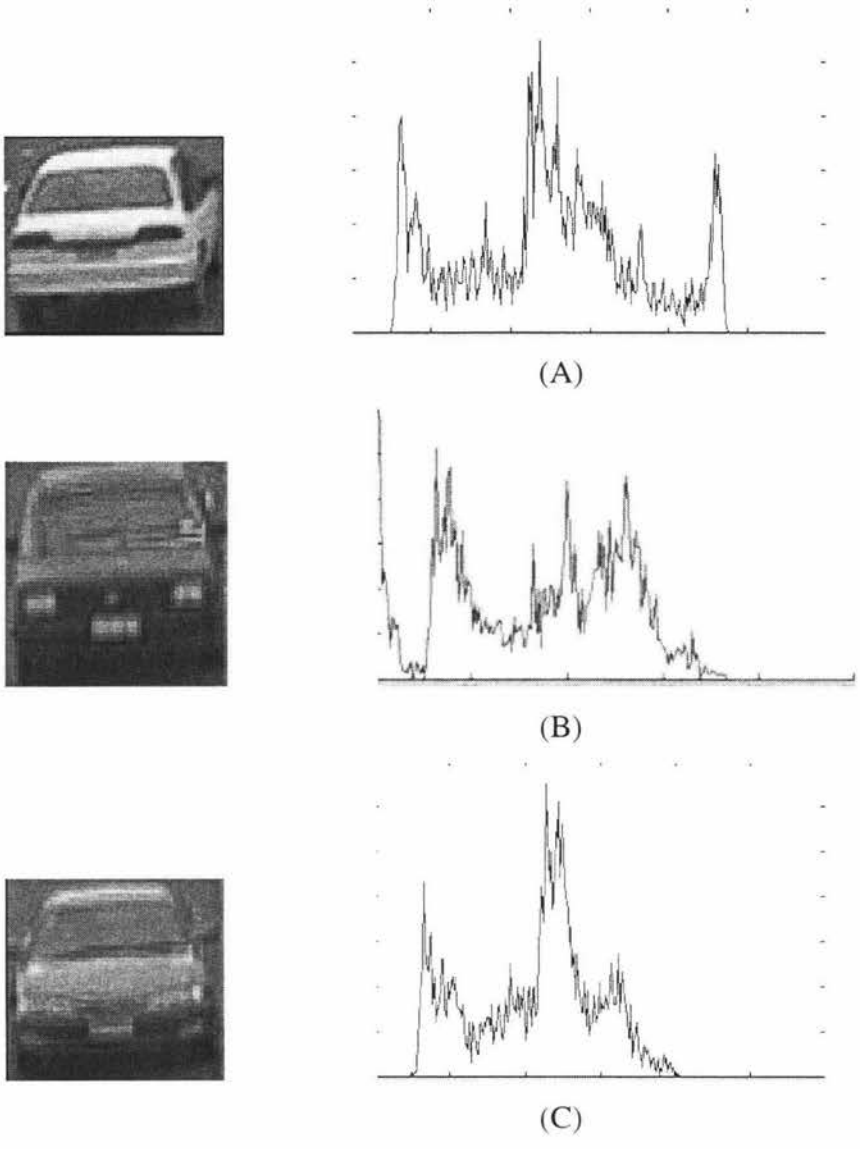


Figure 5.7Histogram diagrams for different vehicles (A) white car from frame 025, (B) gray car from frame 152, (C) red car from frame 195.

## 5.2 Overview of the tracking algorithm

The tracking method we used in our algorithm is the region-based tracking method. The bounding boxes of all vehicles are obtained and the coordinates of four corner points of the bounding boxes are used to track them. The trajectories of centroid points of vehicles are used to record the trajectories of the vehicles. The histograms of the tracking vehicles are used to distinguish the vehicles that are too close to each other. The occlusion separations are used to separate the occluded vehicles.

Two databases and two arrays are used in tracking. One database called tracking database is used to keep the information of vehicles being tracked. The other one named trajectory database is used to record the trajectories of all the vehicles passing through the monitoring area. Temporary array 1 is used to keep the parameters of the vehicles in current frame. Temporary array 2 is used to keep the information of vehicles that have lost connection with the previous record. The following steps are the outline of our tracking algorithm.

1. Tracking starts with two empty databases. The size of the database can be pre-set by the user.

2. For each current image, if the image is the first one, three parameters of each vehicle are extracted and stored into the tracking database. At the same time, the coordinates of those vehicles are recorded in the trajectory database. If it is not the first one, these parameters would be kept in the temporary 1 array.

   1) Bounding box and its coordinates
   2) Centroid and its coordinates
   3) Colour histogram of the vehicles if vehicles are close to each other

3. a. If the tracking database is not empty, three parameters of the last records of the vehicles in the tracking database are extracted:

      1) Bounding box rectangle and its coordinates
      2) Centroid and its coordinates
      3) Colour histogram of above if vehicles are close to each other

   b. Matching the parameters in the database with the parameters obtained in step 2.

   c. The degree of matching for each parameter is recorded. A link is assigned between two vehicles if the matching conditions are satisfied.

d. Repeat several times until all the vehicles in the current image are matched with the recorded vehicles in the database.

f. If vehicle models cannot find a matching model in the database, then the system checks the temporary 2 array, which is used to store vehicles that are in the tracking database, but lose their connections. If it finds a matching one, the connection is linked. On the other hand, the system can also use the previous record of this vehicle as the current record. Else, the system creates a new place for this model.

g. If one vehicle in the database does not have a match in the current image, which means that the vehicles are leaving the testing region, occlusion is occurring or they are stopping. All the information for these vehicles in the database will be copied onto the temporary 2 array. A check sign and the place for coordinates' record in temporary 2 array are put to the corresponding place for that vehicle in the trajectory database.

h. Record the current set of parameters.

4. Repeat steps 2 and 3 for upcoming images. At the same time, check the records in the temporary 2 array.

a. If the coordinates of one vehicle are near the boundary of the tracking window, and the leaving conditions are satisfied, the coordinates of the bounding boxes and centroids will be output as the trajectory of that vehicle. All the records in tracking database are cleared.

b. If a new vehicle appears in the middle of the field, it will be checked as to whether it is separated from the occlusion or whether it is a stopped vehicle that has started again.

c. If a similar matching can be found in the following frames, the complete records of those vehicles will be copied onto the tracking database and

tracking will be continued. At the same time, a copy of all the coordinates of the bounding box and centroids of the old trajectory will be deleted.

5. Repeat step 2 to 4 until the all the image frames have been visited.

Figure 5.8 shows the flow chart of the tracking algorithm.

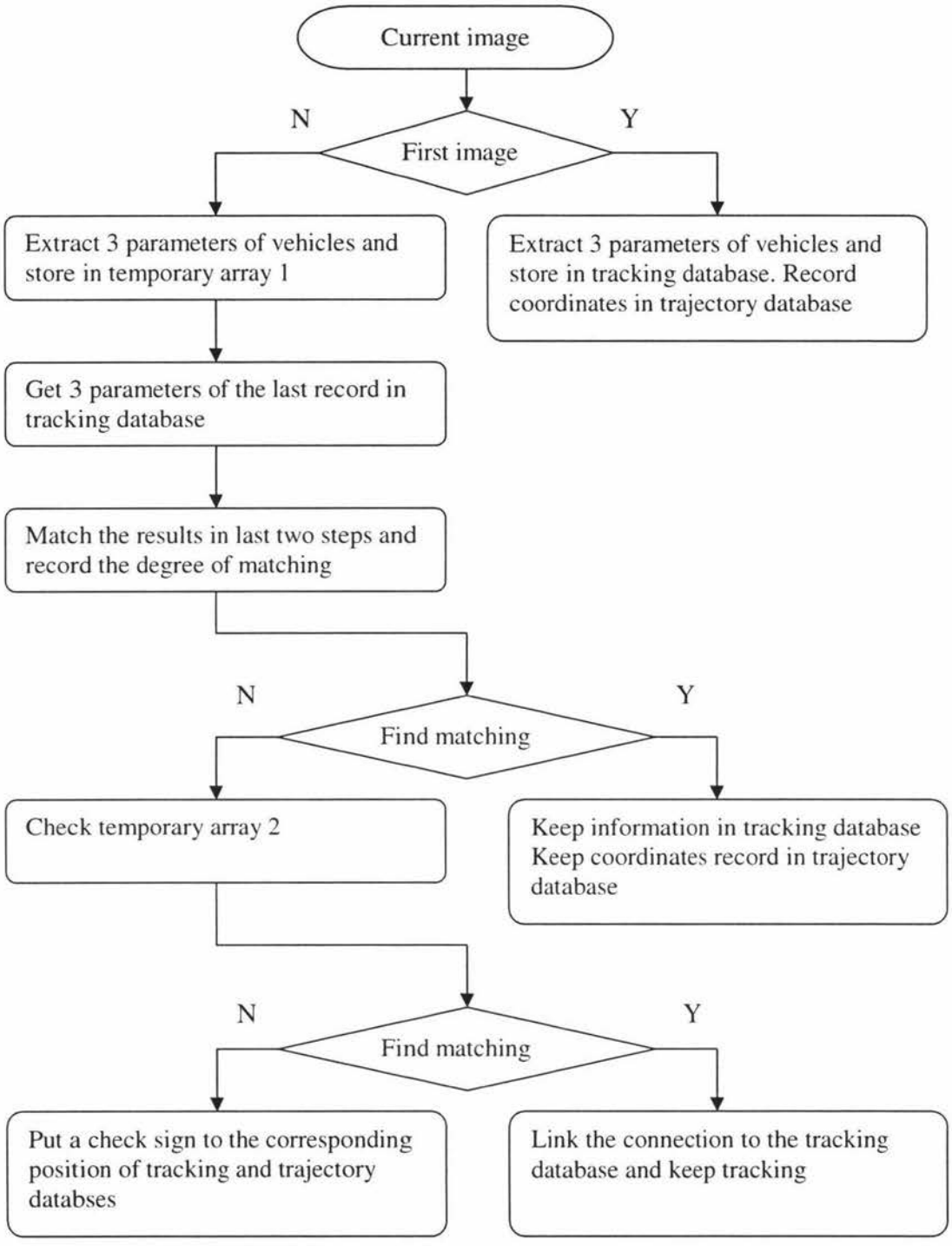
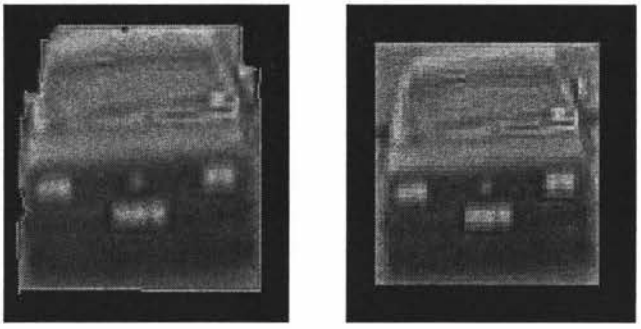



Figure 5.8 Flowchart of tracking algorithm

After obtaining the contour via the snake operation, the final contour is modified as close as to the straight lines as possible so that the vehicles are enclosed. A bounding box is then used to bind the contour. Figure 5.9 shows an example of the detected bounding box.



(A) (B)

Figure 5.9 Bounding box for vehicles (A) Modified contour (B) Bounding box

Bounding boxes and centroids can be used to track vehicles, when they are moving at a relatively low speed compared to the frame rate. For each bounding box, coordinates of the four corner points are recorded into the tracking data base. The centroid of the vehicle is also recorded. Since we are capturing 20 image frames per second from the video, the bounding boxes in two or more consecutive frames will have a considerable overlap.

Firstly, we try to compare only the coordinates of the left top corner of one vehicle with other vehicles and determine the smallest Euclidean distance. If the coordinates of the bounding box of the first vehicle is $(x_1, y_1)$, and the coordinates of the bounding box of the second vehicle is $(x_2, y_2)$, the Euclidean distance is defined as $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$. Euclidean distances for several vehicles can be quite close, due to busy traffic or stopped vehicles. Therefore we record all the close Euclidean distances and compare them later.

The centroids of different vehicles could also be quite close. On the other hand, the distance between each two centroids of the same vehicle is becoming larger when they are moving closer to the camera or smaller when they are moving away from the

camera. Information from the bounding box is used to modify the coordinates in the database, in order to make the centroids more reasonable.

## 5.3 Tracking of slow objects

Pedestrians and cyclists can be found during tracking. Their speeds are much slower when compared to vehicles. If the same track strategy is used to track them, they would be treated as stopped objects and cause false tracking. Pedestrians and cyclists have some properties, such as the likelihood of them appearing in the area of static background. Also, the areas they occupy in a tracking field are smaller and their speeds are slower. These properties can be used to identify and track them using different methods.

In our algorithm, if objects satisfy the following conditions, they will be regarded as pedestrians or cyclists. Some objects are detected in the static background region in one frame, but they disappear in the next few frames, and they are then detected again in one of the following frames. If they are moving very slowly, hardly any difference can be detected in several continuous frames. The aspect ratios of those objects are then calculated. The aspect ratio of one object is defined as its height divided by its width. Pedestrians and cyclists appear relatively tall and thin when they are compared to vehicles. Their aspect ratios are normally greater than 1.5. Hence, we define that, if one object has a small area with an aspect ratio greater than 1.5 and it travels slowly, it is defined as a pedestrian or a cyclist.

Since the travel speed of pedestrians and cyclists is low, we do not need to track them in every frame. Firstly, they are classified into two categories: pedestrians and cyclists. A detailed method can be found in section 5.6. Pedestrians are only tracked every 10 frames and every 4 frames for cyclists. This is because the average speed for a pedestrian is 5 kilometres per hour and the average speed for cyclists is 15 kilometres per hour. The speed limit in urban areas of New Zealand is 50 kilometres per hour. The tracking and occlusion separation strategy is the same as for vehicles.

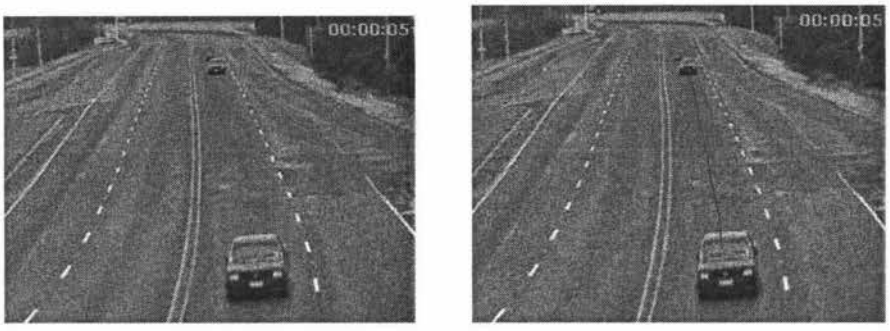
## 5.4 Traffic data extraction

Based on the results of tracking, we can extract some traffic information. The first one is the speed of the vehicles. After counting the number of pixels that one vehicle travelled in one frame, we can get the approximate meters it travelled on the road by using the information in the distance database mentioned in chapter 3. Because 1 frame only takes 0.05 second, which is too short for speed calculation, we take the number of pixels for one vehicle travelled every 20 frames that are equal to 1 second of time. By using the distance information in the distance database, the number of meters travelled by one vehicle in one second can be obtained.

Once the trajectories of all vehicles are obtained, the curvature of each trajectory can be calculated. By comparing the curvature with the curvature information in the curvature database and checking whether the value is greater than the threshold, we can simply determine whether the vehicle is going on the right way. For example, when a vehicle is running on a right way, the curvature of its trajectory is close to the curvature of the lane. When the car is trying to turn right at a corner that is not permitted, the value of curvature will change dramatically. When the value is greater than the predefined threshold, the current frame is written out.

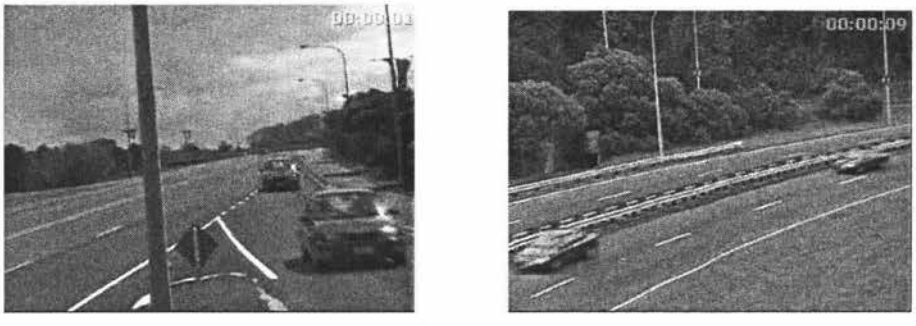
Figure 5.10 shows the trajectory of one car passing through the road. In figures (A) and (B), those two cars are the same car in two different frames. One is in frame number 055, and the other is in frame number 151. In (A), the red points show the centroids that are tracked by the system. In (B), one red line is used to join all tracked centroids together, which shows the trajectory of the car. By comparing this with the curvature of the lane, this car is going on a right way. Figure 5.11 shows the trajectories of the other two cars on two different roads. In (A), the car is turning left into the branch off the main lane. In (B), the car is running on a curve.

Figure 5.12 shows the trajectories of four vehicles on the road. It can be noted that the trajectories of the red car and the grey car are very close.




(A) (B)

Figure 5.10 Trajectory of one tracked car (A) recorded centroid (B) trajectory of the car



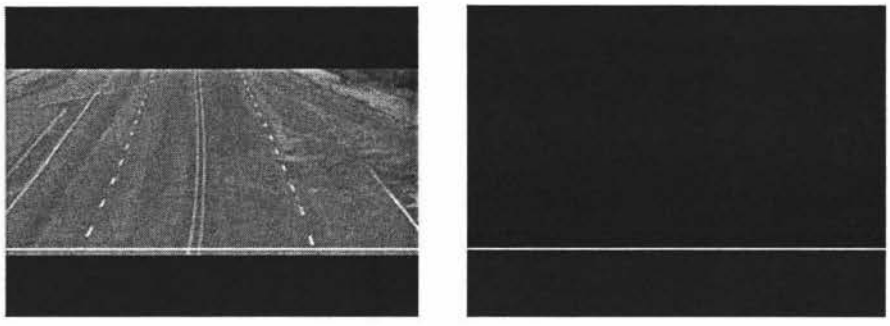

(A) (B)

Figure 5.11 two cars passing on two roads (A) turning left (B) passing on a curve


Figure 5.12 trajectories of four vehicles on road

## 5.5 Vehicle counting

A counting line is set at the bottom of the tracking window. The white line in figure 5.13 (A) is the counting line. Figure 5.13 (B) shows the line in a binary image.

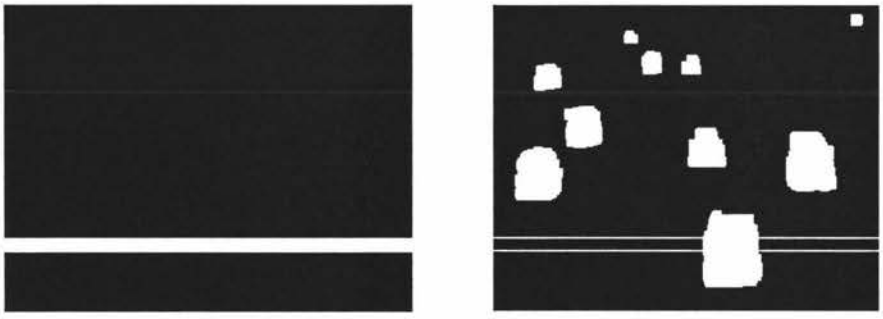


(A) (B)

Figure 5.13 counting line in tracking window (A) in background image (B) in binary image.

Counting is carried out by observing the blob that is in a binary image form passing the counting line. The reason for using a binary blob in counting, instead of an actual vehicle blob, is that sometimes parts of the vehicle body are similar to the background and this causes miscounting. Another reason is that using a binary blob can reduce the computational load.

The actual counting area is formed by using all the pixels that are 5 pixels higher and lower than the counting line. Setting the counting area wider can ensure more pixels of vehicles can be observed. If one vehicle enters the counting area and is going to leave the tracking area, the system begins to check the link between the vehicle in the current position and the corresponding position in the previous frames, through the tracking database. Actually, we count vehicles in two directions, on the left hand side, when one vehicle enters the monitoring area, it will be classified first, and then be counted. On the other hand, if one vehicle enters the counting area, which means it is leaving the tracking area, it will be classified, and the number of vehicle will be increased one. Figure 5.14 shows one example of vehicles passing through the counting area.



(A) (B)

Figure 5.14 one example of vehicle counting (A) actual observing area, (B) one vehicle passing counting area.

## 5.6 Vehicle classification

Vehicle classification is very important in a traffic monitoring system. There are a large number of vehicle classification schemes using different methods. More details can be found in section 2.11. Due to the real-time application requirement, simple vehicle classification schemes that roughly classify vehicles into several categories such as pedestrians, cyclists, motorcycles, cars and trucks, are sufficient. Depending on the application, further classification such as classifying vehicles into detailed models can be performed by off-line analysis. Although we only use a coarse, blob-based classification, which is at the top level of classification, it can still reduce the amount of work to be undertaken at the lower level. Top level classification only categories vehicles into several rough groups, while the low level classification has many categories for vehicle classification that can category vehicles into sufficient large number of classes.

Vehicle classification is performed during counting. When an object enters the counting area, the bounding box information of this object is obtained. This information is used to classify objects into different classes.

We first use the method proposed by Liu and Zhou [74] to classify moving objects into vehicles and non-vehicles. Their approach uses a three-stage binary hierarchical classifier to classify moving objects.

In our approach, after tracking, objects are already classified into fast moving objects (vehicles) and slow moving objects (pedestrians and cyclists). For the slow moving objects, we use the shape-based metrics (compactness and aspect ratio) to separate them into multiple humans and vehicles.

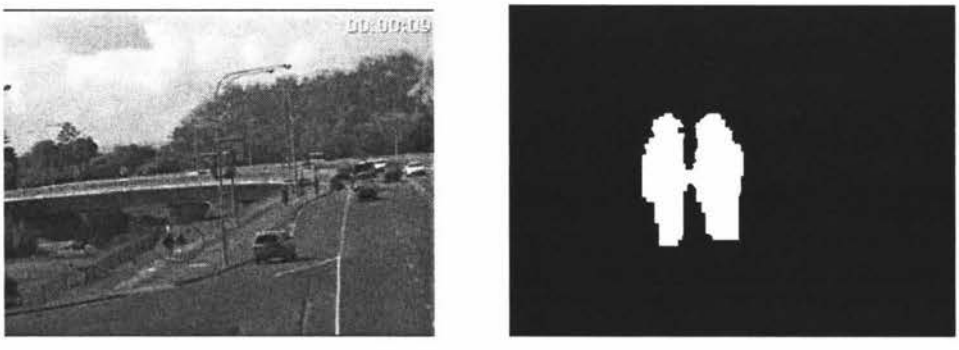
Compactness is defined as the ratio of the area and the square of perimeter:

$$Compactness = \frac{Area}{Perimeter^2} \tag{5.1}$$

Aspect ratio is the ratio of the height and width of the objects:

$$Aspectratio = \frac{Height}{Width} \tag{5.2}$$

When multiple people stand together, the shape and the area they occupy are very similar to one vehicle. Figure 5.15 shows two people walking together. Figure 5.16 shows the comparison of the blob of one vehicle with two walking people. We can see that they look quite similar but the compactness for one car (0.356) and two walking people (0.008483) is significantly different. Therefore, compactness can be used to separate multiple people with slow or stopped vehicles.



(A) (B)

Figure 5.15 detection of multiple humans, (A) two people in the traffic scene (shown within the circle, (B) detected blob.

(A) (B)

Figure 5.16 Comparison of blobs of (A) one car, (B) two close walking people

After comparing the results of our experiment and the result from Liu and Zhou [74], we select 0.045 as the threshold level for single vehicle and multiple people walking closely, if they have similar shapes and areas. If the compactness of one object is less than 0.045, it is defined as multiple people walking together. If it is greater than 0.045, it is defined as a slow moving vehicle.

In [74], Liu and Zhou use the square of aspect ratio and the area of the object to classify humans, cyclists, and motorcycles. In their algorithm, the blobs of human will have larger square of aspect ratio (11.11 in average) while motorcycle has half of that value (5.76 in average). So they can set a clear threshold value (around 7) to classify them. However, in our algorithm, the monitoring area is larger, detected blobs do not have much details, and the detected blobs look very similar in shapes. Hence the difference between a single human blob and a single cyclist blob is not significant. The square aspect ratio still shows that a single pedestrian has a little larger value than that of a motorcycle. However, we could not set a clear threshold in this case. Hence, we classify single human and single cyclists into one category.

For the fast moving objects, motorcycles are classified, first by their aspect ratio and then their area features. When they are compared to cars or trucks, motorcycles only occupy a small area, and their aspect ratio is larger than that of cars. By combining these two features, motorcycles are identified.

In the last step, area and aspect ratio features are used to classify the remainder of moving objects into two rough categories: cars/vans/minibuses and trucks/buses. The actual length and width of the vehicles are computed also. We use a combination of

length and width as a condition for classification due to the camera orientation for observing might be different for different situations. We choose 100 trucks and 100 cars to get the average length and width for the threshold level. Once the features of one detected blob are great than the threshold level, it will be defined as a truck or bus. Otherwise it is a car, van or minibus. However, in some case, some kinds of cars are longer and wider than trucks, and they are classified as trucks or buses. Because we only perform a rough classification of vehicles, these errors are allowed. The classification algorithm we used has provided enough useful information for monitoring use.

Figure 5.17 shows the overview of the vehicle classification algorithm.

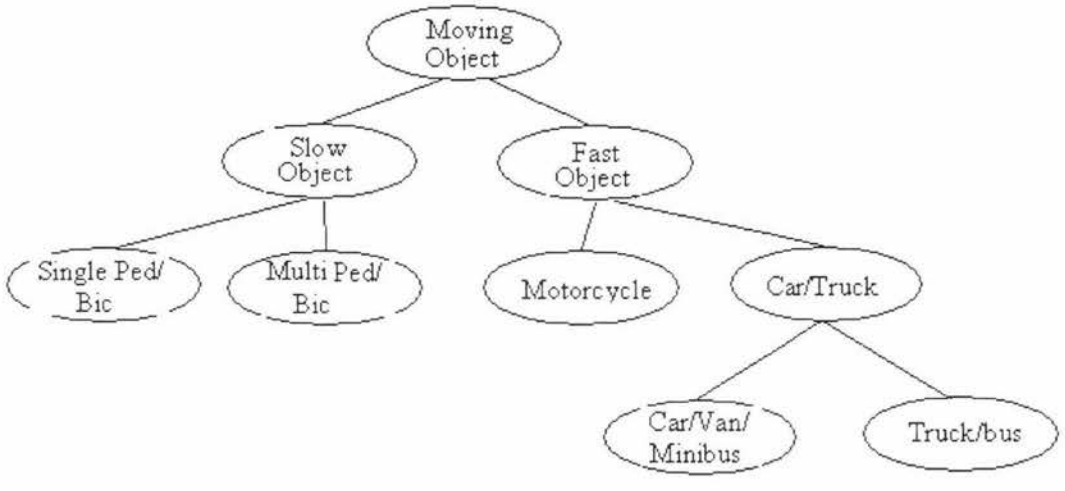



Figure 5.17 Overview of vehicle classification.

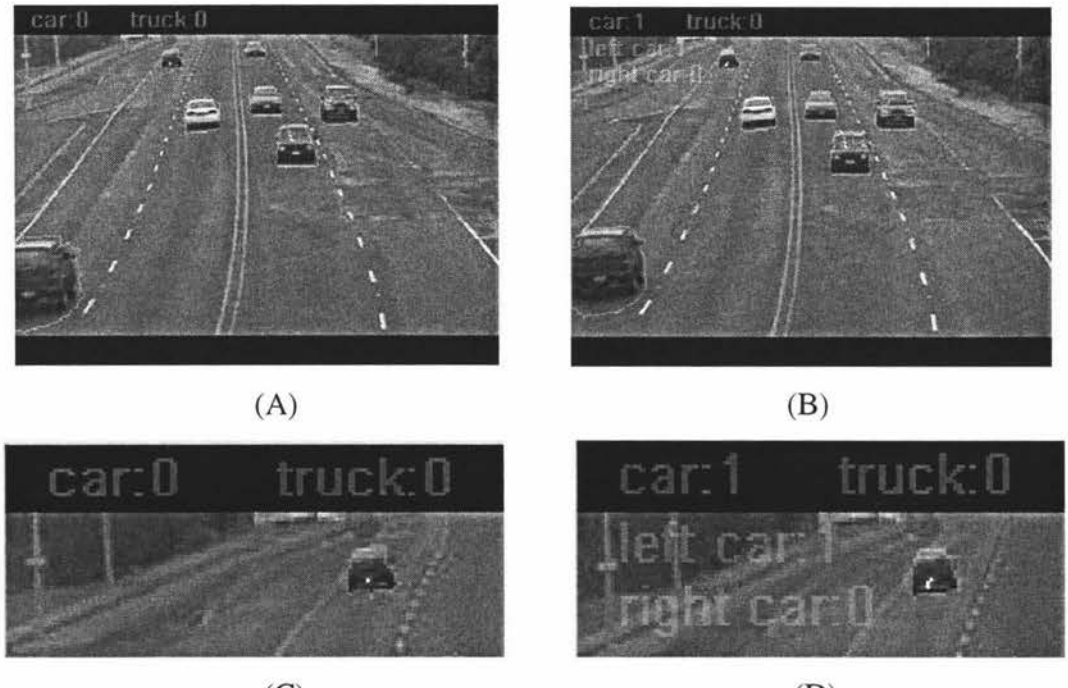
Ped: pedestrian

Bic: bicycler

## 5.7 Result display

The result of tracking, counting and classification is displayed at the left top part of the final monitored image. All the vehicles are tracked by the system once they have been detected. However, vehicles are only counted when they pass the counting line.

When one vehicle passes the counting line, it is classified by the system. If it belongs to the small vehicle group, the number of car will be increased one. If it falls into the big vehicle group, the number of truck will be increased one. On the other hand, the second counting function, which counts the cars or trucks on the left hand side and on the right hand side, will also be performed. Figure 5.18 shows the result of one vehicle passes the left counting line. We can see that once this car passes the counting line, the number of cars is increased by one and the number of cars passes left counting line is also increased. Figure 5.19 shows the result of one vehicle passes the right counting line.




(A) (B)

(C) (D)

Figure 5.18 One example of one vehicle passes left counting line (A) before passing the left counting line, (B) After passing the left counting line (C) , (D) enlarged result display.

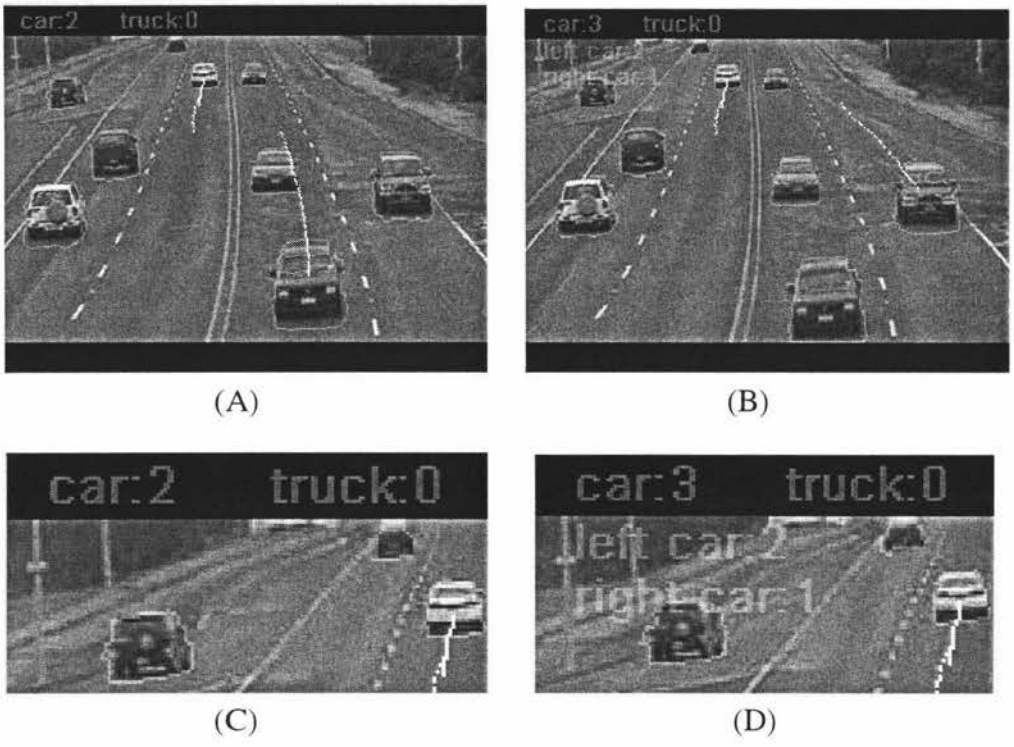



Figure 5.19 One example of one vehicle passes right counting line (A) before passing the right counting line, (B) After passing the right counting line (C) , (D) enlarged result display.

## 5.8 Summary

In this chapter we presented the major parts of traffic monitoring that are vehicle tracking, vehicle counting and vehicle classification.

Occlusion brings tracking errors and miscounting often appears during tracking. A modified method that finds the minimum connecting distance in the middle part of the occlusion area and separates the occlusion is proposed. Histogram is very important information of the objects in colour images that can be used to separate objects moving very close.

A tracking method that combines the benefits of contour-based tracking and region-based tracking is used to perform tracking. Tracking is done by comparing the coordinates of the bounding boxes and histogram information. Through tracking, some traffic data can be extracted. Vehicle counting is done by observing the objects

passing the counting line. A coarse and blob-based vehicle classification is proposed to category objects into several groups.

In the next chapter, we show the results of the testing of the algorithm we developed in this research.

# 6. Experimental results

In the previous chapters, we introduced the background modelling and updating methods, illumination analysis and shadow handlers, and vehicle tracking, counting and classification.

In this chapter, we present the evaluation of the performance and the results of our algorithm discussed in previous chapters. The evaluations are performed based on different functional parts using different methods. Firstly, we present the testing result of background modelling and foreground extraction using quantitative evaluation. Next, we show the testing results of shadow handler, vehicle tracking, counting, and classification.

## 6.1 Background modelling and foreground extraction

Background subtraction has been selected as the foreground extraction method in our algorithm. Hence the results of subsequent processing rely on the output of the background modelling and foreground extraction. We need to ensure that it performs well so that the false detection rate can be low and this will increase the accuracy in the following processes.

Firstly, we check the percentage of vehicles that can be detected in the scene, which includes whole detections and partial detections. A whole detection means all pixels that represent a vehicle are enclosed by the contour. A partial detection means some pixels that represent a vehicle are detected as background pixels. Because we are using three frames together to get the differences between frame 1 and frame 2, and between frame 2 and frame 3, the percentage of vehicles detected is high. Due to the environmental condition and illumination changes, some segments of the vehicles cannot be detected. However, such partially detected vehicles have no effect on vehicle tracking and counting. Obviously, such vehicles will not be classified properly.

Table 6.1 shows the numbers of detected vehicles in 5 different video sequences. Because the changes between two frames are too small, for each video sequence we select one frame in every 10 frames for the test. This test uses the vehicle contours obtained after a snake operation on blobs. The number of vehicles detected is summed up by the computer, and the actual number of vehicles is manually counted by the author. The number is only the sum of the number of vehicles in selected frames. Hence it is not the actual number of vehicles running. Many vehicles are counted repeatedly due to the short time difference between selected frames. This value is only used to show the ability of the system in finding vehicles.

Table 6.1 Results for vehicles detected

| Video Sequence Number | Vehicle Counted | | | Detection Type | | | |
|---|---|---|---|---|---|---|---|
| | By System | By Human | (%) | Whole Detection | (%) | Partial Detection | (%) |
| 1 | 907 | 923 | 98.26 | 863 | 95.15 | 44 | 4.85 |
| 2 | 506 | 515 | 98.25 | 478 | 94.47 | 28 | 5.53 |
| 3 | 1220 | 1238 | 98.55 | 1168 | 95.73 | 52 | 4.27 |
| 4 | 201 | 205 | 98.04 | 193 | 96.01 | 8 | 3.99 |
| 5 | 773 | 782 | 98.84 | 741 | 95.86 | 32 | 4.14 |
| Average | N/A | N/A | 98.39 | N/A | 95.44 | N/A | 4.56 |

We can see that on average 98.39% of the vehicles can be detected by the system, and 95.44% of them are whole detections. Hence on average 93.9% (98.39% x 95.44%) of the vehicles can be wholly detected in the scene.

The above table shows the number of vehicles that are found in the scene. In the next step, we use quantitative evaluation that measures the incorrect pixels to further estimate the performance of background subtraction method. Two types of pixels are checked:

(a) False positive pixels: pixels that are belong to background but have been segmented as foreground.

(b) False negative pixels: pixels that are belong to foreground but have been segmented as background.

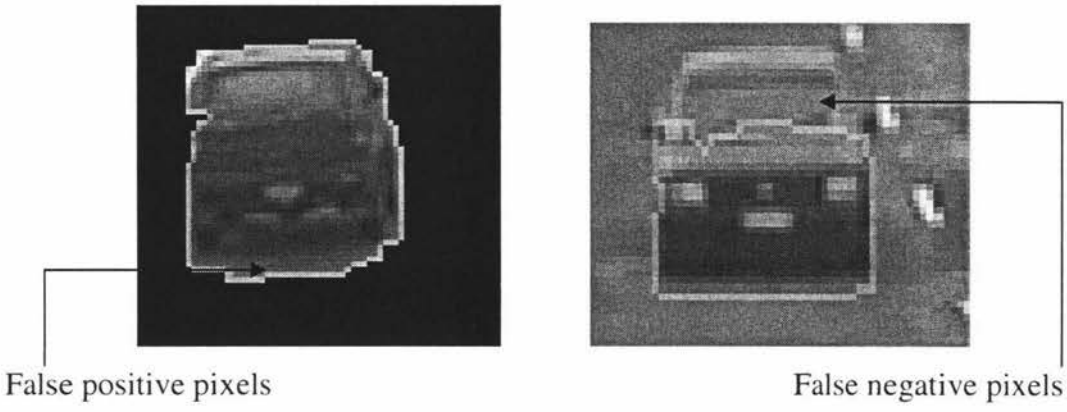



Figure 6.1 False positive and false negative pixels

Following formulas are used to calculate accuracy and pixel ratio for those two mentioned pixels types.

$$\text{Background accuracy} = \frac{N_b}{N_{fp} + N_b} \times 100 \tag{7.1}$$

$$\text{Foreground accuracy} = \frac{N_f}{N_{fp} + N_f} \times 100 \tag{7.2}$$

$$\text{Overall accuracy} = \frac{N_b + N_f - (N_{fp} + N_{fn})}{N_{fp} + N_{fn}} \times 100 \tag{7.3}$$

Where $N_f$ is the number of foreground pixels

$N_b$ is the number of background pixels

$N_{fp}$ is the number of false positive pixels

$N_{fn}$ is the number of false negative pixels

Through this analysis, we can obtain the accuracy of foreground pixels and background pixels using the proposed background modelling and updating methods. Table 6.2 shows the value of accuracy.

Table 6.2 Results of foreground segmentation

| Video Sequence Number | Accuracy (%) | |
|---|---|---|
| | Foreground | Background |
| 1 | 96.35 | 95.21 |
| 2 | 97.65 | 94.26 |
| 3 | 96.77 | 92.63 |
| 4 | 98.32 | 93.64 |
| 5 | 98.42 | 94.67 |
| Average | 97.5 | 94.08 |

The values in table 6.2 are the average values for the video sequences when initial background updating is completed. Actually, at the beginning of each video sequence, the system has to establish a stable background over a stream of frames. Hence the amount of false positive pixels is large during the initial background updating. When the background is updated as close as to the current frames, it reduces to a good level and keeps this level until next update.

## 6.2 Evaluation of shadow handler

The accuracy of tracking also relies on the effectiveness of shadow handler. If the shadow handler could not solve the shadow problems, partial occlusion might occur in later stages. To analyze the robustness and effectiveness of shadow handler, two experiments are performed to check the accuracies of shadow detection and shadow removal, respectively.

When intensity values of the shadows of some vehicles are similar to the background, shadows are treated as background and hence could not be detected.

We use three image sequences that contain shadows to evaluate our algorithm. Table 6.3 shows the results of the accuracy of shadow detection.

Table 6.3 The results of shadows detection (1)

| Video Sequence Number | Number of Vehicles with Shadows (Counted Manually) | Number of Vehicles with Shadows (Detected by Computer) | Accuracy (%) |
|---|---|---|---|
| 1 | 623 | 609 | 97.75 |
| 2 | 961 | 944 | 98.23 |
| 3 | 529 | 513 | 96.97 |
| Average | N/A | N/A | 97.65 |

We can see that 97.65% of the shadows are detected by the system, which is close to the accuracy of vehicle detection (98.39%). This result also shows that our algorithm for background modelling and updating is robust.

It is difficult to remove shadows completely in the image sequences. However, if most of them are removed, vehicles with small parts of shadows can still be tracked correctly. Through experiences, we found that, if more than 70% of the shadow of a vehicle is removed, it would not be affected by the rest part of the shadow in the following stages. Hence we set 70% and 50% as two thresholds to evaluate the performance of shadow handler. Later operations such as occlusion separation can further eliminate shadows. Table 6.4 shows the evaluation results. We can see that, 92.28% of the shadows can be removed by more than 70%. If we consider the total number of vehicles with shadows, 87.91% (95.27% x 92.28%) of the shadows can be removed with good results.

Table 6.4 The result of shadow handler (2)

| Video Sequence Number | Shadows Found | Removed (%) >70% | Removed (%) 50%-70% | Removed (%) <50% |
|---|---|---|---|---|
| 1 | 609 | 92.28 | 3.13 | 4.59 |
| 2 | 944 | 92.58 | 2.76 | 4.66 |
| 3 | 513 | 92.00 | 2.94 | 5.06 |
| Average | N/A | 92.28 | 2.94 | 4.78 |

## 6.3 Evaluation of tracking

We use a quantitative evaluation to estimate the performance of vehicle tracking.

We select 10 video sequences to evaluate the performance of tracking. Tracking starts when vehicles enter the tracking zone, and ends when they leave the field. When a vehicle passes through the tracking area, we can get a set of centroids of it. These centroids will be connected together to form the trajectory of this vehicle. When the trajectory of one vehicle is observed, we can easily determine whether tracking is correct. A correct tracking means a complete trajectory is drawn between the entry and the end of the tracking area for one vehicle. Table 6.5 shows the results of quantitative evaluation of the selected video sequences. The average tracking accuracy is around 87%.

Table 6.5 Results of evaluation for vehicle tracking

| Video Sequence Number | Number of Vehicles $N$ | Number of Vehicles Tracked ($N_f$) | Accuracy ($A_f$) |
|---|---|---|---|
| 1 | 231 | 201 | 87.01 |
| 2 | 359 | 307 | 85.51 |
| 3 | 132 | 117 | 88.63 |
| 4 | 96 | 76 | 79.16 |
| 5 | 326 | 289 | 88.65 |
| 6 | 271 | 233 | 85.97 |
| 7 | 312 | 279 | 89.42 |
| 8 | 93 | 82 | 88.17 |
| 9 | 87 | 81 | 93.11 |
| 10 | 109 | 98 | 89.91 |
| Average | N/A | N/A | 87.02 |

## 6.4 Evaluation of counting

The evaluation of counting is done by comparing the number of vehicles counted by the computer with that counted by the author. We still use those 10 video sequences that are used for vehicle tracking evaluation to analyse vehicle counting. In our algorithm, a vehicle is counted when it passes the counting line that is set at the bottom part of test window. The counting accuracy can reach around 97% for our algorithm.

Table 6.6 Results of vehicle counting

| Video Sequence Number | Number of Vehicles Counted by Computer | Number of Vehicles Counted Manually | Accuracy (%) |
|---|---|---|---|
| 1 | 87 | 90 | 96.67 |
| 2 | 117 | 119 | 98.32 |
| 3 | 56 | 56 | 100 |
| 4 | 27 | 29 | 93.1 |
| 5 | 161 | 167 | 96.4 |
| 6 | 112 | 114 | 98.25 |
| 7 | 93 | 96 | 96.88 |
| 8 | 42 | 43 | 97.67 |
| 9 | 34 | 35 | 97.14 |
| 10 | 41 | 43 | 95.34 |
| Average | N/A | N/A | 96.97 |

## 6.5 Evaluation of classification

The evaluation of classification consists of two parts, which are the slow moving objects classification and the fast moving objects classification.

### 6.5.1 Slow moving objects classification

There are only two categories for slow moving objects, one is the single pedestrian or bicycler and the other is the multiple pedestrians or bicyclers.

The slow moving objects are separated from other detected objects according to their speed, the areas they occupy and the compactness of them. The average accuracy of slow moving objects detection is nearly 98% in the experiments. When two or more pedestrians are walking closely or some parts of their bodies are overlapped with each other, the detected blob is similar to a slow moving or a stopped vehicle. They are wrongly classified as a single pedestrian or a single bicycler instead of multiple pedestrians or bicyclers. However, the classification rate for a single pedestrian or bicycler is 100% unless the detected area is too small so that it is ignored by the system. Since the system is developed for vehicle monitoring, the number of pedestrians and bicyclers and their trajectories are not needed. Therefore it is sufficient for the system to just detect and classify slow moving objects as pedestrians or bicyclers.

We could not find many pedestrians and bicyclers in our video sequences. Hence we checked all of them in total in 10 video sequences that are used in the evaluation of vehicle tracking and counting. There are 121 blobs containing single or multiple pedestrians or bicyclers in all the sequences, and 119 of them are classified correctly, which means the accuracy is 98%.

### 6.5.2 Fast moving objects classification

The algorithm that we used to classify fast moving objects into two categories is very simple and straightforward. Detected objects are categorised by their area, aspect ratio, and length/width information. Hence the accuracy can reach 97%. The errors are due to the features of some special types of vehicles or the camera orientation errors. In some situations, some special types of cars are longer and wider than the threshold level of trucks. On the other hand, some trucks are only a little bit longer and wider than the threshold level of cars due to the camera orientation or calibration errors, which makes them to be misclassified as cars. The error rate is acceptable. Table 6.7 shows the classification results of vehicle.

Table 6.7 Results of vehicle classification

| Video Sequence Number | Long Vehicles | | Short Vehicles | | |
|---|---|---|---|---|---|
| | Counted Manually | Counted by Computer | Counted Manually | Counted by Computer | Accuracy (%) |
| 1 | 17 | 19 | 70 | 68 | 97.7 |
| 2 | 11 | 10 | 106 | 107 | 99.15 |
| 3 | 5 | 5 | 51 | 51 | 100 |
| 4 | 2 | 1 | 25 | 26 | 96.15 |
| 5 | 12 | 15 | 149 | 146 | 97.98 |
| 6 | 21 | 24 | 91 | 88 | 96.7 |
| 7 | 13 | 11 | 80 | 82 | 97.56 |
| 8 | 7 | 8 | 35 | 34 | 97.14 |
| 9 | 3 | 3 | 31 | 31 | 100 |
| 10 | 9 | 7 | 32 | 34 | 94.11 |
| Average | N/A | N/A | N/A | N/A | 97.65 |

## 6.6 Summary

In this research, we have developed a system that can detect and monitor moving objects on the road using the video sequences captured from a camera that is mounted on a bridge over a road. These video sequences are captured on different roads, weather and traffic situations that are common in an urban area. In this chapter, the experimental results show that the system is able to carry out some specific tasks in traffic monitoring such as vehicle tracking, counting, and classification, with a high accuracy. The system is designed for outdoor traffic monitoring purpose.

# 7. Conclusion and future work

In this chapter, a brief conclusion is given to summarise our work. We also propose possible modifications and future developing directions of our research.

## 7.1 Conclusion

In this thesis, we proposed and designed a system that can detect moving objects in the traffic scene, and monitor them via tracking, counting and classification using video sequences that are captured from a stationary camera in the scene. This system is robust for various environmental conditions. The system does not require much of the scene-specific knowledge, which makes it useful for complex outdoor scene application. This system provides some useful facilities such as counting vehicles in both directions, monitoring vehicle trajectories, estimating the speed of vehicles and detecting stopped vehicles. It is also simple to implement, which can fulfil requirements of real-time applications. One of the main advantages of our system is its low computational load and relatively high accuracy. It can be used to gather traffic information in traffic monitoring systems. This system requires no particular information about camera installations. The camera can be installed on roadside or on the bridge. The initial experimental results show that the accuracy of the system is high.

Background subtraction is a very popular method in vehicle detection. Background subtraction is selected in this system for segmenting foreground objects from the frames. The key problem is how to simulate and update the background adaptively. We proposed a method that separates background into static background and dynamic background. Two different methods are used to update those two types of backgrounds. After comparing the results between the new background and the initial modelled background, we found that the proposed method can provide a good presentation of the current background. The experimental results show that the proposed background subtraction method is robust and it can produce the difference image with better quality. Colour information is used to detect foreground objects. By using the maximum differences in three colour channels and a threshold operation, foreground objects are detected effectively. Snake operation, which utilises the

energy minimisation, is used to refine the final contours of detected vehicles. Through the combination of these three methods the vehicle detection rate reaches 93.9% in our tests.

Illumination changes rapidly in outdoor environment due to daytime progresses and weather conditions. Shadow effect is one of the important issues that relates to the illumination condition. Shadow can influence the performance of the system significantly. We use an illumination analysis method to automatically examine whether shadows exist in an image and determine the direction of illumination if shadows are existed. A method is used to find out the shadow seeds in shadows. By using the colour features of shadow seeds and simple normalisation of colour components, moving objects are distinguished from their shadows quickly and accurately. The average removing rate for shadow areas greater than 70% was found to be 92.28%.

Vehicle tracking is the most important part in traffic monitoring since the accuracy of subsequent actions such as counting and classification depends much on the performance of tracking. We use a method that combines region-based tracking and contour-based tracking. The tracking is done by cooperating with occlusion separation and histogram matching. The average tracking accuracy is around 87%. Once vehicles are tracked correctly, miscounts or multiple counts can be reduced. The counting accuracy we obtain is 96.97%. Vehicle classification in this system only groups vehicles into two categories. The classification criterions are simple and straightforward. However, this simple classification can generate enough information for real-time monitoring purpose. Further processing can be done off-line to classify vehicles into more detailed groups according to the user needs. The accuracy for classification is around 98%.

In conclusion, the system we developed can fulfil the requirements for real-time traffic monitoring, and it is robust against environmental changes. The accuracy that the system achieved through experiments is high enough for the monitoring purpose.

## 7.2 Contributions

The proposed traffic monitoring system combines a set of effective and efficient methods, and has the potential to be developed for real-time processing. The key contributions of our research to the traffic monitoring field can be summarized as follows.

Although the idea of separating background into static background and dynamic background is not new, we have proposed an improved method to divide the background into static background and dynamic background for updating separately, which can provide accurate updating of background image with less computational load. The percentage in intensity change (5%) has been suggested in our research as a criterion to group pixels into static and dynamic pixels according to the experiment results. In order to get better updating result, an array called 'dynamic array' has been used to compare and keep the (interim) results until background is separated correctly. The computational load is large for this background separation method , however, this is only required once for the system in the pre-processing stage. In order to reduce the computational load of checking the intensity change, we have proposed a new method that uses four test squares to check the intensity change. Through the combination of brightness compensation method to further reduce the incorrect intensity update, this method can provide fast checking with less computation. In addition, the update frequency is determined by using the knowledge of time and date information. For future developments, we suggest that the weather information can be added to the system.

We have also combined the idea of finding lane dividing lines [38] and 'blob properties' to perform illumination analysis. Once lane dividing lines are found, only detected blobs within two lines are analysed. The properties of blobs such as area and aspect ratios are calculated to determine whether shadows exist. This method uses the road structure information and only focuses on the properties of detected blobs, thus can make a fast decision.

The tracking approach we proposed combines the benefits of region-based tracking and contour-based tracking. Although we have employed in our system some existing

methods such as the use of coordinates, the degree of matching, the colour histogram, we have optimized the implementation of these methods in our system. Through the experiment results, we have demonstarted that our implementation is fast and effective. By using the counting line and rough classification, real-time traffic monitoring system can be further developed based on our implementation.

## 7.3 Future work

At the moment, the system we developed can be used to perform traffic monitoring directly if it is programmed in embedded systems such as FPGA or DSP. The avenues for future work in this area are many. Certain aspects can be considered in further development of our system to make it perform better and develop towards intelligent monitoring.

### 7.3.1 Incorporating accurate vehicle classification

As mentioned before, the classification algorithm we used in our research is the rough classification that only groups vehicles into cars or trucks/buses due to the real-time requirement. However, this rough classification is not enough for traffic monitoring purpose. The traffic monitoring system should have many categories that can cover nearly all types of vehicles such as sedans, liftbacks, couples, wagons, jeeps and so on. Further classification schemes can be involved into the system by doing off-line analysis. Vehicle recognition is a possible way to fulfil the requirement. As many vehicle models as possible can be recorded. These models can be input into a database. Once a vehicle is detected, a matching is performed to find the best match between it and the database to get the right classification result. Neural network approach will be a good choice to perform the matching.

### 7.3.2 Improving the camera resolution

Accurate off-line analysis acquires high resolution images and additional information about the scene that is provided by using one or two more cameras. However, due to

the limitation of computational load and the processing power of the computer, the camera resolution is low in our algorithm, which is only 320 x 240. The vehicles appear in each frame are so small that do not contain much information for further analysis use. When the algorithm is operated in hardware and with the modification of it, the processing can be speeded up so that high resolution images can be obtained and used in the future work.

### 7.3.3 3D vehicle modelling and tracking

With the improvement of computational speed, vehicles can be modelled by using 3D models. Because 3D models provide some useful information such as the length and height of vehicles the recognition and tracking accuracy can be increased. 3D models exploit the local visual cues that are based on local image features. These cues can predict the hidden movement and the acts of the objects, and detect partial occlusions. Hence it improves the tracking result. We can use two or more cameras to monitor the road traffic to facilitate the 3D modelling.

### 7.3.4 Develop towards automatic security and surveillance

The system can be further developed towards automatic security and surveillance that is used at the entrances of buildings or at the corners of roads. By incorporating human body detection and recognition algorithms, human can be tracked as blobs. Further development will involve human action recognition [96 - 97].

### 7.3.5 Capturing traffic video sequence from high altitude

Better quality of video sequences can make image processing much easier with better results. On the other hand, it can also make the life of the system developers easier as they do not have much to worry.

One of the possible ways for us to improve our system is to obtain traffic images with less occlusion problems and shadow effects. In practice, it is very common to install video cameras in the side of the road near the traffic lights or on a bridge over a road. Occlusion problems and shadow effects always appear in the captured sequences.

One of the possible solutions is to install cameras at a high altitude looking downwards the road. This method could significantly reduce occlusion problems and shadows effects. We can also install a spot light around the monitoring area to further eliminate shadow problems. Capturing traffic video sequence from high altitude is a good way to improve the quality of video sequences and our system could be further developed towards this direction.

# Reference


1. J. Gajda, R. Sroka, M. Stencel, A. Wajda, T. Zeglen, *A vehicle classification based on inductive loop detectors* Proceedings on the 18th IEEE Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Volume 1, pp. 460 - 464, 21-23. May 2001.

2. K. Dickinson, C. Wan, *An evaluation of microwave vehicle detection at traffic signal controlled intersections* Third International Conference on Road Traffic Control, 1990. pp.153 – 157, 1-3 May 1990.

3. N. Hoose, *Computer vision as a traffic surveillance tool*, Proceedings on Transportation systems, IFAC, Tianjin, 1994.

4. M.Fathy, M.Y.Siyal, *A window-based image processing technique for quantitative and qualitative analysis of road traffice parameters*, IEEE Transactions on Vehicular Technology 47 (4) 1998.

5. M. Fathy, M.Y. Siyal, C.G. Darkin, *A low-cost approach to real-time morphological edge detection*, Proceedings on IEEE TENCON Conference, Singapore, 1994.

6. D. Beymer, P. McLauchlan, B. Coifman, J. Malik, *A real-time computer vision system for measuring traffic parameters*, Proceedings on the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 495-501 1997.

7. V. Kastrinaki, M. Zervakis, K. Kalaitzakis, *A survey of video processing techniques for traffic applications*, Image and Vision Computing, Volume 21, Issue 4, pp. 359-381, 1 April 2003.

8. W. Enkelmann, *Interpretation of traffic scenes by evaluation of optical flow fields from image sequences*, proceedings on IFAC Control Computers, Communications in Transportation, Paris, France 1989.

9. R. Kories, G.Zimmermann, *Workshop on motion: Representation and Analysis, Kiawah Island Resort*, Charleston/SC, IEEE Computer Society Press, pp. 101-106, May, 1986.

10. A. Giachetti, M. Campani, V. Torre, *The use of optical flow for road navigation*, IEEE Transportation on Robotics and Automation 14 (1) 1998.

11. W. Enkelmann, *Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences*, Computer Vision, Graphics, and Image Processing pp. 150-177, 1998.

12. B.K.E Horn, B. Schunck, *Determining optical flow*. Artificial Intelligence 17, pp. 185-203, 1981.

13. A. Cumani, A. Guiducci, E. Grattoni, *Image description of dynamic scenes*. Pattern Recognition 24 volume 7: 661-673, 1991.

14. Q. Zhang, R. Keltte, *Robust background subtraction and maintenance*, proceedings of the 17th International Conference on Pattern Recognition, (ICPR) 2004, Volume 2, 23-26, pp. 90-93, Aug, 2004.

15. M. Eki, H. Fujiwara, K. Sumi, *A robust background subtraction method for changing background*, Fifth IEEE workshop on applications of Computer Vision, pp. 207-213, 4-6 Dec, 2000

16. S. Beucher, J. M. Blosseville, F. Lenoir, *Traffic spatial measurements using video image processing*, SPIE Vol. 848 Intelligent Robots and Computer Vision, Sixth in a series pp. 648-655, 1987.

17. C.L. Wan, K.W. Dickinson, T.D. Binnie, *A cost-effective image sensor system for transportation applications utilising a miniature CMOS single chip camera*, Proceedings of IFAC transportation systems, Tianjin, , 1994.

18. K. Subramaniam, S. S. Daly, F. C. Rind, *Wavelet transforms for use in motion detection and tracking application*, Seventh International Conference on Image Processing and Its Applications, Volume 2, p.p. 711-715, 13-15 July 1999

19. N. D. Matthews, P. E. An, D. Charnley and C. J. Harris, *Vehicle detection and recognition in greyscale imagery*, Control Engineering Practice, Volume 4, Issue 4, pp. 473-479, April 1996.

20. D. Koller, J. Weber, and J. Malik, *Robust multiple car tracking with occlusion reasoning*, Tech. Rep. UCB/CSD-93-780, EECS Department, University of California, Berkeley, Oct 1993.

21. S. S. Cheung, C. Kamath, *Robust techniques for background subtraction in urban traffic video*, Proceedings of Video Communications and Image Processing, January, 2004

22. R. Cucchiara, C. Grana, M. Piccardi, A. Prati, *Detecting moving objects, ghosts, and shadows in video streams*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 25, Issue 10, pp.1337 – 1342, Oct. 2003.

23. K. Toyama, J. Krumm, B. Brumitt, B. Meyers, *Wallower: Principles and practice of background*, in ICCV (1), pp. 255-261, 1999.

24. A. Elgammal, D. Harwood, L. Davis, *Non-parametric model for background subtraction*, in Proceedings of IEEE ICCV'99 Frame-rate workshop, Sept 1999.

25. N. McFarlane, C. Schofield, *Segmentation and tracking of piglets in images*, Machine Vision and Applications 8(3), pp. 187-193, 1995.

26. J. Heikkila, O. Silven, *A real-time system for monitoring of cyclists and pedestrians* in Second IEEE Workshop on Visual Surveillance, pp. 246-252, Jun 1999.

27. K.-P. Karmann, A. Brandt, V. Cappellini, *Moving object recognition using and adaptive background memory*, in Time-Varying Image Processing and Moving Object Recognition, ed., 2, pp. 289-307, Elsevier Science Publishers B.V.,1990

28. D. Koller, J. Weber, J. Malik, *Robust multiple car tracking with occlusion reasoning*, Rep. UCB/CSD-93-780, EECS Department, University of California, Berkeley, Oct 1993.

29. N. Friedman, S. Russell, *Image segmentation in video sequences: A probabilistic approach*, in Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI{97), pp. 175-181, Morgan Kaufmann Publishers, Inc., (San Francisco, CA), 1997.

30. N. Hoose, IMPACT: *an image analysis tool for motorway analysis and surveillance*, Traffic Engineering Control Journal pp. 140-147, 1992.

31. M. Fathy, M.Y. Siyal, *An image detection technique based on morphological edge detection and background differencing for realtime traffic analysis*, Pattern Recognition Letters 16 pp. 1321–1330, 1995.

32. C. Jiang, O. W. Matthew, *Shadow identification*, Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92, 1992 IEEE Computer Society Conference pp. 606 – 612, 15-18 June 1992.

33. Y. Wu, X. Ye, W. Gu, *A shadow handler in traffic monitoring system*, IEEE 55th Vehicular Technology Conference VTC 2002, Volume 1, pp. 303 – 307, 6-9 May 2002.

34. M. Kilger, *A shadow handler in a video-based real-time traffic monitoring system*, Proceedings of IEEE Workshop on Applications of Computer Vision, pp. 11-18 1992.

35. C. J. Chang, W. F. Hu, J. W. Hsich, Y. S. Chen, *An automatic traffic surveillance system for tracking and classifying vehicles*, Proceedings of 15th IPPR Conference on Computer Vision, Graphics and Image Processing, pp.382-387, 2002

36. J. M. Scanlan, D. M. Chabries, R. W. Christiansen, *A shadow detection and removal algorithm for 2D images*, Proceedings of International Conference on Acoustics, Speech, and Signal Processing, pp.2057-2060,1990.

37. J.M. Wang, Y.C. Chung, C.L. Chang, S.W. Chen, *Shadow detection and removal for traffic images*, IEEE International Conference on Networking, Sensing and Control, 2004 Volume 1, pp. 649-654, 21-23 March 2004.

38. J. Hsieh, S. Yu, Y. Chen, W. Hu, *A shadow elimination method for vehicle analysis*, Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. Volume 4, pp. 372-375, 23-26 Aug, 2004.

39. K. Onoguchi, *Shadow elimination method for moving object detection*, Pattern Recognition, 1998. Proceedings Fourteenth International Conference, Volume 1, pp. 583-587, 16-20 Aug. 1998.

40. R. Rad, M. Jamzad, *Real time classification and tracking of multiple vehicles in highways*, Pattern Recognition Letters, Volume 26, Issue 10, pp. 1597-1607, 15 July 2005.

41. Y. Jung; Y. Ho, *Traffic parameter extraction using video-based vehicle tracking*, Proceedings of 1999 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems, 1999. pp. 764 – 769, 5-8 Oct. 1999

42. L. Heungkyu, K. Hanscok, *Detection of occluded multiple objects using occlusion activity detection and object association*, Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, pp. 100-105, 18-19 Nov, 2004.

43. L. Zhu, J. Song, Q. Huang; M. Zhang; H. Liu, *A novel module of tracking vehicles with occlusion*, Proceedings of IEEE Intelligent Vehicles Symposium, 2005, pp. 894-899, 6-8 June 2005.

44. D. Koller, K. Daniilidis, H.-H. Nagel, *Model-based object tracking in monocular image sequences of road traffic scenes*, International Journal of Computer Vision, vol. 10, no.3 pp. 257-281, 1993.

45. H. Kollnig, H. H. Nagel, *3D pose estimation by fitting image gradients directly to polyhedral models*, Proceedings of Fifth International Conference on Computer Vision, 1995, pp. 569 – 574, 20-23 June 1995

46. H. Kollnig, H. H. Nagel, *3D pose estimation by directly matching polyhedral models to gray value gradients*, International Journal of Computer Vision, vol. 23, no. 3 pp.283-302, 1997.

47. W. Hu, X. Xiao, D. Xie, T. Tan, S. Maybank, *Traffic accident prediction using 3D-model-based vehicle tracking*, IEEE Transactions on Vehicular Technology, Volume 53, Issue 3, pp. 677 – 694, May 2004.

48. M. Haag and H. H. Nagel, *Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences*, International Journal of Computer Vision, vol. 35, no. 3 pp.295-319, 1999.

49. T. N. Tan, G. D. Sullivan, K. D. Baker, *Model-based localization and recognition of road vehicles*, International Journal of Computer Vision, Vol. 27, no.1, pp. 5-25, 1998

50. T. N. Tan, K. D. Baker, *Efficient image gradient based vehicle localization*, IEEE Transaction Image Processing, vol 9, pp.1343-1356, Aug 2000.

51. G. L. Foresti, V. Murino, C. Regazzoni, *Vehicle recognition and tracking from road image sequences*, IEEE Transaction Vehicle Technology, Vol 48, pp.301-318, Jan 1999.

52. B. Gloyer, H. K. Aghajan, S. Kai-Yeung, T. Kailath, *Vehicle detection and tracking for freeway traffic monitoring*, Conference Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, Volume 2, pp. 970-974, 31 Oct.-2 Nov. 1994

53. B. Coifman, D. Beymer, P. McLauchlan, J. Malik, *A real-time computer vision system for vehicle tracking and traffic surveillance*, Transportation Research Part C: Emerging Technologies, Volume 6, Issue 4, pp. 271-288, August 1998.

54. K. Karmann, A. Brandt, *Moving object recognition using an adaptive background memory*, Time-varying Image Processing and Moving Object Recognition, Vol. 2, Elsevier, Amsterdam, 1990.

55. S. Gupte, O. Masoud, R. F. K. Martin, N. P. Papanikolopoulos, *Detection and classification of vehicles*, IEEE Transactions on Intelligent Transportation Systems, Volume 3, Issue 1, pp. 37 – 47, March 2002

56. ——,*Traffic Surveillance and Detection Technology Development*, Sensor Development Final Report, Jet Propulsion Laboratory, Pasadena, CA, Publication 97-10, 1997.

57. J. Malik, S. Russell, J. Weber, T. Huang, D. Koller, *A machine vision based surveillance system for California roads*, PATH project MOU-83 Final Report, University of California, Berkeley, 1995.

58. D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russell, *Towards robust surveillance scene analysis in real-time*, Vol. 1, pp.126-131, ICPR, Israel, 1994.

59. D. Koller, J. Webber, J. Malik, *Robust multiple car tracking with occlusion reasoning*, ECCV, pp. 189-196, Stockholm, Sweden, 1994.

60. M. P. Dubuisson, A. K. Jain, *Contour Extraction of Moving Objects in Complex Outdoor Scenes,* International Journal of Computer Vision, pp. 83 104, 1995.

61. E.C. Pece, *Contour tracking based on marginalized likelihood ratios*, Image and Vision Computing, Volume 24, Issue 3, pp. 301-317, 1 March 2006.

62. B. Schiele, *Model-free tracking of cars and people based on color regions*, in Proceedings of IEEE International Workshop Performance Evaluation of Tracking and Surveillance (PETS '00), pp. 61-71, Grenoble, France, 2000.

63. T. J. Fan, G. Medioni, G. Nevatia, *Recognizing 3-D objects using surface descriptions*, IEEE Trans. Pattern Anal. Machine Intell, pp. 1140–1157.vol. 11, Nov, 1989.

64. P. Tissainayagam and D. Suter, *Object tracking in image sequences using point features*, Pattern Recognition, Volume 38, Issue 1, pp. 105-113, January 2005

65. K.G. Murty, *An algorithm for ranking all the assignments in order of increasing cost*, Oper. Res. 16, Pp. 682-687, 1968.

66. A. Bevilacqua, L. Di Stefano, S. Vaccari, *Using local and global object's information to track vehicles in urban scenes*, Proceedings for IEEE Conference on Advanced Video and Signal Based Surveillance, 2005, pp. 52-57, 15-16 Sept. 2005.

67. S.Kamijo, M.Sakauchi, *Illumination Invariant Segmentation of Spatio-Temporal Images by Spatio-Temporal Markov Random Field Model*, proceedings of ICPR2002, Quebec, Aug. 2002.

68. D. R. Magee , *Tracking multiple vehicles using foreground, background and motion models,* , Image and Vision Computing, Volume 22, Issue 2, pp.143-155, 1 Feb, 2004.

69. J. B. Kim, C. W. Lee, K. M. Lee, T. S. Yun, H. J. Kim, *Wavelet-based vehicle tracking for automatic traffic surveillance* Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, TENCON, Volume 1, pp. 313-316, 19-22 Aug. 2001

70. D. M. Ha, J. -M. Lee and Y. -D. Kim, *Neural-edge-based vehicle detection and traffic parameter extraction*, Image and Vision Computing, Volume 22, Issue 11, pp. 809-907, 20 September, 2004.

71. A. N. Rajagopalan, R. Chellappa, *Vehicle detection and tracking in video*, Proceedings for International Conference on Image Processing, Volume 1, pp.351-354, 10-13 Sept. 2000.

72. J. Soh, B. T. Chun, M. Wang, *Analysis of road image sequences for vehicle counting, Systems, Man and Cybernetics*, IEEE International Conference on Intelligent Systems for the 21st Century., Volume 1, pp. 679 – 683, 22-25 Oct. 1995.

73. M. Perera, K. Harada, *An automatic system for counting and capturing the pictures of moving vehicles in real-time*, Proceedings of IEEE Intelligent Vehicles Symposium, 2003, pp. 85 – 89, 9-11 June 2003.

74. B. Liu, H. Zhou, *Using object classification to improve urban traffic monitoring system*, Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on Volume 2, pp. 1155 – 1159, 14-17 Dec. 2003.

75. D. Noll, M. Werner, W. V. Seelen, *Real-time vehicle tracking and classification*, Proceedings of the Intelligent Vehicles '95 Symposium, pp. 101 – 10625-26 Sept. 1995.

76. J. Kjellgren, S. Gadd, N. U. Jonsson, J. Gustavsson, *Analysis of Doppler measurements of ground vehicles*, 2005 IEEE International Radar Conference, pp. 284-289, 9-12 May 2005.

77. H. M. Abdelbaki, K. Hussain, E. Gelenbe, *A laser intensity image based automatic vehicle classification system*, Proceedings of IEEE Intelligent Transportation Systems, 2001, pp. 460-465, 25-29 Aug. 2001.

78. K. F. Hussain, G. S. Moussa, *Automatic vehicle classification system using range sensor*, International Conference on Information Technology: Coding and Computing, ITCC 2005. Volume 2, pp. 107-112, 4-6 April 2005.

79. A. Y. Nooralahiyan, H. R. Kirby and D. McKeown, *Vehicle classification by acoustic signature*, Mathematical and Computer Modelling, Volume 27, Issues 9-11, pp. 205-214, May-June 1998.

80. J. Hsieh; S. Yu; Y. Chen; W. Hu, *Automatic traffic surveillance system for vehicle tracking and classification*, IEEE Transactions on Intelligent Transportation Systems, Volume 7, Issue 2, pp. 175-187, June 2006.

81. A. K. Jain, and R. K. Dubes, *Algorithms for Clustering Data.* Prentice Hall: Englewood Cliffs, NJ, 1988.

82. R. Ohlander, K. Price, and D. R. Reddy, *Picture segmentation using a recursive region splitting method.* Computer Graphics and Image Processing 8: 313-333, 1978.

83. S. Horowitz, T. Pavlidis, *Picture segmentation by a tree traversal algorithm.* Journal of the ACM 23(2): 368-388, 1976.

84. T. N. Hieu, M. Worring, R. van den Boomgaard, *Watersnakes: energy-driven watershed segmentation*, Pattern Analysis and Machine Intelligence, IEEE Transactions Volume 25, Issue 3, P.P 330-342 March 2003

85. M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, Int. J. Comput. Vision, vol. 1, no. 4, pp. 321-331, 1988.

86. A.A.Amini, S. Tehrani, and T.E.Weymouth, *Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints.* Proceedings of Second International Conference on Computer Vision, P.P 95-99, 1988.

87. A. A. Amini, T. E. Weymouth, R. C. Jain, *Using dynamic programming for solving variational problems in vision.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 12, Issue 9, pp. 855 – 867, Sept. 1990

88. D.J. Williams, M Shah, *A fast algorithm for active contours*, Proceedings of Third International Conference on Computer Vision, 1990. P.P 592 – 595, 1990

89. T.F. Chan, L.A. Vese, *Active contours without edges*, Image Processing, IEEE Transactions Volume 10, Issue 2, P.P 266-277, 2001

90. A. Gouze, C. De Roover, B. Macq, A. Herbulofi, E. Debreuve, M. Barlaud, *Watershed-driven active contours for moving object segmentation*, Image Processing, ICIP 2005. IEEE International Conference on, Volume 2, pp. II - 818-2111-14 Sept. 2005.

91. P. Salembier and L. Garrido, *Binary partition tree as an efficient representation for image processing,segmentation, and information retrieval*, IEEE Transaction on Image Processing, vol. 9, no. 4, pp. April, 2000.

92. P. Kumar, K. Sengupta, A. Lee, *A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system*, Proceedings for the IEEE 5th International Intelligent Transportation Systems Conference 2002 pp. 100 – 105.

93. M. Huang; S. Yen, *A real-time and color-based computer vision for traffic monitoring system*, IEEE International Multimedia and Expo Conference ICME '04 Volume 3, pp. 2119 – 2122, 27-30 June 2004.

94. Z. He, J. Liu, P. Li, *New method of background update for video-based vehicle detection*, Proceedings in the 7th International IEEE Intelligent Transportation Systems Conference, pp. 580-584, 3-6 Oct 2004.

95. X. Ji, Z. Wei, Y. Feng, *Effective vehicle detection technique for traffic surveillance systems*, Journal of Visual Communication and Image Representation, Volume 17, Issue 3, pp.647-658, June 2006.

96. S. J. McKenna, S. Jabri, Z. Duric, H. Wechsler, *Tracking interacting people*, Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 348-353, 2000.

97. C. R. Wren, A. Azebayejani, T. Darrel, A. Pentland, Pfinder: *Real-time tracking of the human body*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 780-785, 1997.

# Appendix I User Guide

An automatic traffic monitoring simulation system was developed to allow the purpose of traffic monitoring. This system can load the pre-processed traffic images into the system, and then perform subsequent steps to analyse traffic images and output the required avi format video showing the monitoring results.

This system contains the following 7 major functions:

**1. Initial background modelling:**

The initial background is modelled using averaging technique. The first 200 images are read into the system, and then their format is converted from uint8 to double precision. After that, those double numbers are added together. The average number is obtained by dividing the sum by 200. The final step is to convert the double format result to uint8 format.

**2. Background separation and updating:**

Due to the limitation of memory, the comparison of pixels between current frames and the background frame is performed every 30 frames. The result of each comparison is written out first (data files namely ISPBK to ISPBK7). Then these data files are loaded into the system to continue the comparison for next 30 frames.

Function background update is used to update the static and dynamic backgrounds.

**3. Snake function:**

The snake function block includes 7 sub functional blocks, namely: find_coordinates, dist, curvat, Big_cont, gradient_mag, Assign_value, Energy_min.

When blobs that represent moving object in the scene are obtained from the background, they are labelled. The system processes them one by one to ensure

necessary iteration times of energy minimization for each blob are given. Snake operation includes the subsequent steps.

1. find_coordinates: Once the contour of the blob is obtained, the coordinates of each point on the contour are recorded. The top left point of the contour is defined as the first point of the contour. At the mean time, the direction of next point from the current point is also recorded.

2. dist: The first term can be calculated as $|V_i - V_{i-1}|^2$.

3. curvat: The curvature of each point is calculated using $|V_{i-1} - 2V_i + V_{i+1}|^2$, which is the second term of energy minimization.

4. Big_cont: The 8 neighbour point of each point on the contour are enclosed in a big contour in order to obtain the minimum energy point from those 9 points.

5. gradient_mag: The third term $E_{image}$ is the image force that is the gradient magnitude (mag). Sobel approximation is used to determine the image gradient. (min-mag) / (max-min) is used for the normalized edge strength term.

6. Assign_value: The quantity being minimized can be written as $E = \int(\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds$ . The values of the three mentioned terms are assigned to each point.

7. The minimum energy point in every nine points is found, and the original point will move to that point if the condition is satisfied.

**4. Shadow handler:**

1. Lane line finding:

   A function called lane line finding is used to find out the lane lines of the road. Two sub-functions are involved: find_nextpoint, which is used to find out the key points of the lane line segments. Connect_points, which is used to connect those found lane line points to draw the trajectories of those vehicles.

2. illumination_analysis:

   A function is used to analyse the direction of the illumination. Through the analysis, the existence of shadows can be confirmed. The elimination of shadows is only conducted on the corresponding side of illumination.

3. shadow_handler:

   A function is used to eliminate the shadows within the detected blobs of vehicles based on the result of the illumination analysis.

**5. Vehicle tracking:**

1. find_BC:

   After getting the contours of blobs in the current image, the co-ordinates of bounding boxes and centroids of those blobs are obtained and recorded.

2. assign_TBC:

   The co-ordinates of bounding boxes and centroids of found blobs are put into the tracking arrays.

3. output_trt:

   Check any outputs after the last tracking and display the results if output happens. After outputting results, the size of tracking arrays will be resized.

4. draw_ahead_trj

   Draw the trajectory of the leading vehicle in both directions.

5. TrackCar:

   Check any new entry cars and put their co-ordinates into the tracking arrays, and then do the tracking.

**6. Vehicle classification:**

Vehicle_classification:

Classify vehicles according to their properties.

**7. Vehicle counting:**

Vehicle_counting:

Count vehicles based on the result of classification.

# Appendix II Pseudo codes and sample codes of major functional parts

## A2.1 Pseudo code of snake function:

```
Get the contours of all vehicles in the middle frame from three continuous frames
and background
Label all found contours
Count the number of total contours (num)
Create a new snake map
Get one single blob

run_num = 1; (record the running times)

while run_num<=num
        Get one single contour
        Record the coordinates (x ,y) of every pixel on the contour
        Calculate the first term of snake function
        Calculate the curvature of each point
        Get 8 neighbor points of each point
        Calculate the image force (gradient magnitude) of each point
        Assign three calculated values to each point
        Find the point that contains minimum energy in every nine points
        Move the contour point to that point if the conditions are satisfied
        Clear the processed contour
        run_num = run_num+1;
    Record the modified contour to the snake map
end;
```

## A2.2 A sample code of snake function:

```
[blobs,cont]=get_contour(I1, I2, I3, BG);

[L,num] = bwlabel(cont,8);
cars = regionprops(L,'all');
carsArea = [cars.Area];

[BL,num] = bwlabel(blobs,8);
cars_BL = regionprops(BL,'all');
cars_BLArea = [cars_BL.Area];

I=rgb2gray(I2);

snake_map=zeros(240,320);

run_num=1;

while(run_num<=num)
   c1=find(cars_BLArea==max(cars_BLArea));
   S=ismember(L,c1);
   S=bwmorph(S,'spur',5);
   [dire, xco, yco]=find_coordinate(S);
   [dist_n]=dist(xco,yco);
   [curva]=curvat(xco,yco);
   [new_cont]=Big_cont(S,I);
   [gra_mag]=gradient_mag(new_cont);
   [dist_big, curvat_big]=Assign_value(xco, yco, dist_n, curva, new_cont);
   [E1]=Energy_min(dist_big,curvat_big,gra_mag,xco,yco,dire);


   indices=find(cars_BLArea==max(cars_BLArea));
   cars_BLArea(indices)=0;
   run_num=run_num+1;
end;

figure,imshow(snake_map);
```

## A2.3 Pseudo code of shadow handler:

```
Find land line
Read images
Get blobs
Label blobs

While (run_num<num_of_blobs)
        Find one blob;
        Illumination analysis;
        if no shadow
                break;
        else
                shadow handler;
        run_num=run_num+1;
end;

Combine the resulting blobs
```

## A2.4 A sample code of shadow handler:

```
BG=imread('BG200.jpg');
I1=imread('031.jpg');

lane_finding;

[rd3]=get_blobs(I1,BG);

[L,num] = bwlabel(rd3,8);
cars_BL = regionprops(L,'all');
cars_BLArea = [cars_BL.Area];

run_num=1;

while(run_num<=num)
   c1=find(cars_BLArea==max(cars_BLArea));
   S=ismember(L,c1);
   [LA, RA, direction]=illumination_analysis(S);

   if direction==0
      break;
   end;

   [S2]=shadow_handler(LA,RA,direction,S,I1,BG);

   for i=1:240
      for j=1:320
         for k=1:3
            if S2(i,j,k)~=0
               BG(i,j,k)=S2(i,j,k);
            end;
         end;
      end;
   end;

   indices=find(cars_BLArea==max(cars_BLArea));
   cars_BLArea(indices)=0;
   run_num=run_num+1;
end;

BG=uint8(BG);
figure,imshow(BG);
```

## A2.5 Pseudo code of tracking:

```
Create an avi file
Read the first three traffic images and background image
Get the contours of all vehicles in the middle frame from three continuous frames
and background
Get the coordinates of bounding boxes and centroids of all vehicles (BoBox1,
Centd1)
Assign the coordinates as the first result of tracking box and centroid box
Add text to the first blobs image
Add the first frame to the avi file

for run_num = 1:limit

        Read next three traffic images
        Get the contours of all vehicles in the middle frame from three continuous
        frames and background
        Get the coordinates of bounding boxes and centroids of all vehicles
        (BoBox2, Centd2)
        Add text to the first blobs image
        Add current frame to the avi file
        Check any outputs from the current frame when comparing to last frame

        if output = true
                add text to trajectory image
                output trajectory image
                add text to counted vehicle image
                output counted vehicle image
        end;

        Assign the coordinates of the last record of tracking box to BoBox1
        Assign the coordinates of the last record of centroid box to Centd1
        Do the tracking
        Assign the result of tracking to tracking box and centroid box
end;

Close avi file
Read the avi file
Play the created avi file
```

## A2.6 A sample code of tracking:

```
lane_finding;

aviobj=avifile('traffic2','fps',1);
aviobj.quality = 100;

Inum1=1;
Inum2=2;
Inum3=3;

Imnum1=int2str(Inum1);
Imnum2=int2str(Inum2);
Imnum3=int2str(Inum3);

I1=imread([Imnum1 '.jpg']);
I2=imread([Imnum2 '.jpg']);
I3=imread([Imnum3 '.jpg']);
BG=imread('BG.jpg');

TBBox=zeros(20,40,8);
TCentd=zeros(20,40,2);
car_num=zeros(1,5);

[output,blobs1,cont1]=get_contour(I1, I2, I3, BG, I2a);

[BoBox1, Centd1]=find_BC(blobs1,cont1);

fr_num=1;

[TBBox, TCentd]=assign_TBC(BoBox1, Centd1, TBBox, TCentd, fr_num);

[output]=add_text(output, car_num);

aviobj=addframe(aviobj,output);

for run_num=1:39

   Inum1=Inum1+3;
   Inum2=Inum2+3;
   Inum3=Inum3+3;

   Imnum1=int2str(Inum1);
   Imnum2=int2str(Inum2);
   Imnum3=int2str(Inum3);

   I4=imread([Imnum1 '.jpg']);
   I5=imread([Imnum2 '.jpg']);
   I6=imread([Imnum3 '.jpg']);
   I5a=I5;
```

```
    [output, blobs2,cont2]=get_contour(I4, I5, I6, BG, I5a);

    [BoBox2, Centd2]=find_BC(blobs2,cont2);

        [out_trtB,out_trtC,TBBox,TCentd, car_num, Traj,A, ...
        op]=output_Trt(TBBox, TCentd, BoBox2, Centd2, fr_num, ...
        I5a, car_num, BGa);

    [output]=draw_ahead_trj(TBBox, TCentd, output, fr_num);

    [output]=add_text(output, car_num);

    aviobj = addframe(aviobj,output);

    [BoBox1, Centd1]=assign_BC(TBBox, TCentd, fr_num);

    fr_num=fr_num+1;

        [TrackBBox, TrackCentd, car_num,B, cc]=TrackCar(BoBox1, ...
        Centd1, BoBox2, Centd2, car_num, I5a, BGa);

        [TBBox, TCentd]=assign_TBC(TrackBBox, TrackCentd, TBBox, ...
        TCentd, fr_num);

end;

aviobj=close(aviobj);

mov = aviread('traffic2');

movie(mov,3,1)
```

# Appendix III Program Disc