Automatic Vehicle Detection and Tracking in Aerial Video

by

Xiyan Chen

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of

Doctor of Philosophy of Loughborough University

30th September 2015

Abstract

This thesis is concerned with the challenging tasks of automatic and real-time vehicle detection and tracking from aerial video. The aim of this thesis is to build an automatic system that can accurately localise any vehicles that appear in aerial video frames and track the target vehicles with trackers.

Vehicle detection and tracking have many applications and this has been an active area of research during recent years; however, it is still a challenge to deal with certain realistic environments. This thesis develops vehicle detection and tracking algorithms which enhance the robustness of detection and tracking beyond the existing approaches. The basis of the vehicle detection system proposed in this thesis has different object categorisation approaches, with colour and texture features in both point and area template forms. The thesis also proposes a novel Self-Learning Tracking and Detection approach, which is an extension to the existing Tracking Learning Detection (TLD) algorithm. There are a number of challenges in vehicle detection and tracking. The most difficult challenge of detection is distinguishing and clustering the target vehicle from the background objects and noises. Under certain conditions, the images captured from Unmanned Aerial Vehicles (UAVs) are also blurred; for example, turbulence may make the vehicle shake during flight. This thesis tackles these challenges by applying integrated multiple feature descriptors for real-time processing.

In this thesis, three vehicle detection approaches are proposed: the HSV-GLCM feature approach, the ISM-SIFT feature approach and the FAST-HoG approach. The general vehicle detection approaches used have highly flexible implicit shape representations. They are based on training samples in both positive and negative sets and use updated classifiers to distinguish the targets. It has been found that the detection results attained by using HSV-GLCM texture features can be affected by blurring problems; the proposed detection algorithms can further segment the edges of the vehicles from the background. Using the point descriptor feature can solve the blurring problem, however, the large amount of information contained in point descriptors can lead to processing times that are too long for real-time applications. So the FAST-HoG approach combining the point feature and the shape feature is proposed. This new approach is able to speed up the process that attains the real-time performance. Finally, a detection approach using HoG with the FAST feature is also proposed. The HoG approach is widely used in object recognition, as it has a strong ability to represent the shape vector of the object. However, the original HoG feature is sensitive to the orientation of the target; this method improves the algorithm by inserting the direction vectors of the targets.

For the tracking process, a novel tracking approach was proposed, an extension of the TLD algorithm, in order to track multiple targets. The extended approach upgrades the original system, which can only track a single target, which must be selected before the detection and tracking process. The greatest challenge to vehicle tracking is long-term tracking. The target object can change its appearance during the process and illumination and scale changes can also occur. The original TLD feature assumed that tracking can make errors during the tracking process, and the accumulation of these errors could cause tracking failure, so the original TLD proposed using a learning approach in between the tracking and the detection by adding a pair of inspectors (positive and negative) to constantly estimate errors. This thesis extends the TLD approach with a new detection method in order to achieve multiple-target tracking. A Forward and Backward Tracking approach has been proposed to eliminate tracking errors and other problems such as occlusion. The main purpose of the proposed tracking system is to learn the features of the targets during tracking and re-train the detection classifier for further processes.

This thesis puts particular emphasis on vehicle detection and tracking in different extreme scenarios such as crowed highway vehicle detection, blurred images and changes in the appearance of the targets. Compared with currently existing detection and tracking approaches, the proposed approaches demonstrate a robust increase in accuracy in each scenario.

Acknowledgments

I would like to acknowledge my appreciation firstly to my supervisor Dr. Qinggang Meng. His help and suggestions have assisted me a lot. He was always there when I needed any help with programme problems, the direction of the research or my writing up. He has been a very responsible supervisor.

Many people have also contributed in various ways to this PhD thesis and I would like to take the opportunity to show my sincere gratitude to these people, who have supported me throughout my PhD studies.

I would like to thank my family for supporting me all these years, without their support I would never have had the opportunity to study in the UK.

I would like to thank the staff at the Computer Science as they always been very helpful when I need any help.

Contents

List of Figures

List of Tables

List of Abbreviations

2D:	two-dimensional
3D:	three-dimensional
AIVeRT:	Active Learning-based Vehicle-Recognition and Tracking
AP:	Affinity Propagation algorithm
ASIFT:	Affine-SIFT
bLPS-HoG:	boosting Light and Pyramid Sampling Histogram of Oriented Gradients
CoGD:	Co-trained Generative-Discriminative tracking
DARPA:	Defence Advanced Research Projects Agency
EKF:	Extended Kalman Filter
EM:	Expectation-Maximisation
FAST:	Features from Accelerated Segment Test
FBT:	Forward and Backward Tracking
GLCM:	Gray Level Co-occurrence Matrix
HCP:	Harris Corner Points
HF:	Haar-like Features
HoG:	Histogram of Oriented Gradient
HSV:	Hue, Saturation, Value
ISM:	Implicit Shape Model
ITS:	Intelligent Transportation System
IVT:	Iterative Visual Tracking

KF:	Kalman Filter
KNN:	K-Nearest Neighbours
LBF:	Local Binary Feature
LESH:	Local Energy based Shape Histogram
MIL:	Multiple Instance Learning
MSER:	Maximally Stable Extremal Regions
OB:	Online Boosting
ODB:	Online Discriminative Features
PCA-SIFT:	Principal Components Analysis SIFT
PF:	Particle Filter
RF:	Random Forests
SCD:	SUSAM Corner Detector
SF:	Shape-let Feature
SIFT:	Scale-Invariant Feature Transform
SLTD:	Self-Learning Tracking and Detection
SMC:	Sequential Monte Carlo
STARS:	Spatio-Temporal Appearance-Related Similarity
SURF:	Speeded Up Robust Feature
SVM:	Support Vector Machines
TLD:	Tracking Learning Detection
TVD:	Tracked Vehicles Database
UAV:	Unmanned Aerial Vehicles

Chapter 1

Introduction

Unmanned Aerial Vehicles, also known as UAVs, have become the new generation of the worldwide aviation industry. A UAV is an aircraft without an actual human pilot; instead, it has an autonomous system which can fly autonomously or be flown remotely from ground stations (Figure 1.1). The very first UAV product created was a balloon that carried explosives during the American Civil War [1]. At that moment, people realised that UAVs were a new key element in war. After some time had gone by, the first pilotless aircraft was created by the US in 1916, and during the WWII, more and more types of UAV were made in the technology rush [2]. During the cold war, the US military and the Soviet Union developed a number of types of UAV for spying and nuclear tests. Most applications of UAVs are for the beginning stages of military missions, such as reconnaissance. Nowadays, UAVs even have the ability to take part in attack missions. UAVs have the advantage of zero casualties occurring during battle, which is the major consideration in modern warfare.

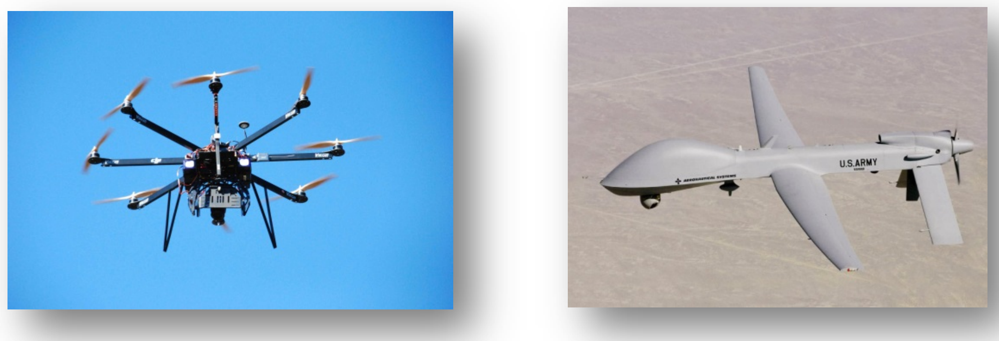


Figure 1.1: The different appearances of some modern UAVs.

In recent years, UAVs have also been involved in civilian operations due to their potential abilities: high mobility, fast deployment and wide surveillance scope, as well as being able to be deployed in extreme environments and weather. UAVs can be equipped with different types of imaging camera depending on the mission. They also have GPS equipped on board, along with automatic positioning and stabilization systems. UAVs are light-weight, inexpensive components with low power consumption, which can easily complete extreme tasks. In May 2008, a magnitude 9.0M_s earthquake hit Wenchuan city and caused massive damage. All transportation and communication cut out, so the rescue teams could not get any information about the damages, casualties etc. Because of the complexity of the geographical conditions (the city is located 4,000 metres above sea-level), the flight conditions for planes or helicopters were terrible and it was too dangerous to send aircraft into the area, so the rescue teams had no idea what was happening out there. In situations like this, multiple low-cost UAVs can be sent to survey the area in order to get damage information. In 2010, the Yushu earthquake occurred, registering a magnitude of 7.1M_s. The Chinese military had learned a lesson form the Wenchuan earthquake, and sent multiple UAVs into the centre of the disaster area immediately after the earthquake happened. They received first-hand images of the destruction (Figure 1.2) in the city, so the rescue teams knew the precise places where help was needed and what tools they needed to take. These two incidents show how crucial UAVs can be in real life.

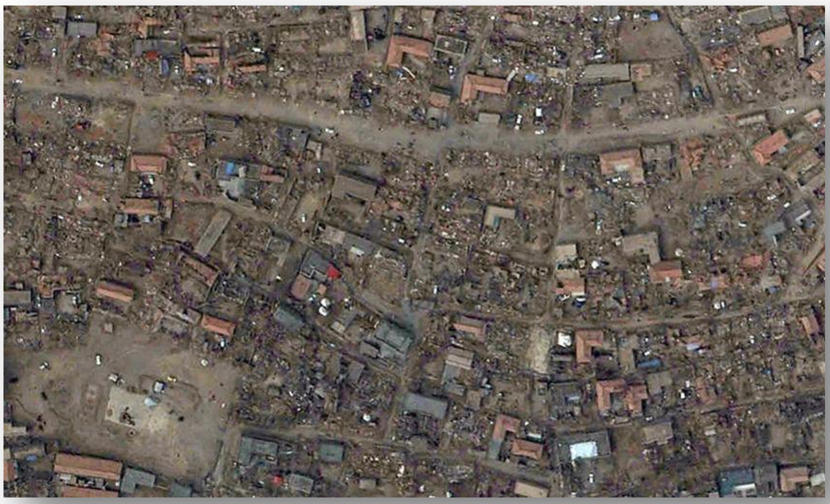


Figure 1.2: An image captured from Yushu by a UAV after the earthquake [3].

Normally, UAVs are controlled by an operator with a terminal device that receives the aerial images taken from the UAVs. These operators will examine the images/videos themselves and then make decisions regarding how to control the UAVs, therefore it is basically a remote control. On the other hand, autonomous UAV control is where UAVs make their own decisions based on sensory information. In recent years, there has been increased research into vehicle detection and tracking in surveillance traffic monitoring and in military applications such as border control, etc. However, moving object detection and tracking from a UAV platform remains very challenging under certain circumstances. Most of these challenges include complex backgrounds [4] and fast movement of UAVs [5] resulting in blurred images [6], and limited computational resources, etc.

This thesis proposes various methods of detecting and tracking vehicles using the images and videos captured by UAVs. In particular, long-term video tracking and the detection of specific targets is considered, and the solutions to problems such as blurred images, target occlusions and other invariant requirements will be discussed in the following chapters.

1.1 Vehicle Detection and Tracking

Vehicle detection has received a great deal of attention in computer vision literature. Historically, many vehicle detection and tracking approaches have been developed in the fields of video surveillance by UAVs and fixed CCTV cameras. Recently, contributions have also focused on object model recognition for other categorisations. Recognition models usually need higher resolution images or videos for the detection method, and using these models, impressive results have been presented for the object classes of, for instance, vehicles and faces detection.

This thesis proposes several successful methods and algorithms for the vehicle detection and tracking processes. The goal is to create a high-accuracy, long-term vehicle detection and tracking approach. The proposed system has been divided into three processes: vehicle detection, vehicle tracking and self-learning. The three methods used in the proposed detection system are: 1) a HSV-GLCM algorithm with blurring removal; 2) an ISM-SIFT algorithm and 3) a FAST-HoG algorithm. For the tracking process, a Forward and Backward Tracking (FBT) algorithm is proposed. The FBT can monitor detection performance by using existing tracking results. For the learning process, two inspectors (positive and negative) have been developed for making error estimations for tracking performance. The proposed mechanism uses a generative object model in the tracking system, which can update the classification model throughout whole process. The process is carried out under the assumption that the tracking target model is not in the training categories, which means the system can learn the appearance of new categories simultaneously. At the same time, the algorithm can reliably distinguish the distinctive features of vehicles from background structures.

1.1.1 Applications

Vehicle detection and tracking have a large range of applications from pure observation to interactive tasks, and are used with both fixed cameras and moving cameras. These applications always require real-time detection or tracking during the process.

In the field of image processing, face detection is a mature technique that is commonly used in real life. Meanwhile, the image process of detection and tracking on people and vehicles from UAVs has raised a significant attention. In the previous researches, these are motivated by the military applications, such as aerial surveillance and automatic target recognition. However, in recent years there has been an increase in the civil and commercial applications from the UAVs. These applications are used in traffic monitoring [7], aerial surveillance [8], and security related tasks such as border control [9], search and rescue missions [10].

Nowadays, as the significant increasing of the vehicles, the road surveillance and transport control system are required. In these applications, large amount of videos or images are need to be captured and analysed, also most of the applications are require process in real-time. Thus, the automatic vehicle detection and tracking is becoming a key research topic. The researchers are focused on different applications, such as in the traffic monitoring, and the detection of the traffic flow, these applications also become the key element for the development of the future intelligent transportation system. In the past, the traffic monitoring is applied in a fixed location on the crossing, which has several disadvantages such as the high maintenance cost and limited observation area, etc. As a result, traffic monitoring by using the UAVs has been increasingly invested. The advantage of using the UAVs, which can provide more flexible and accurate images and videos in the traffic incidents, also it has high mobility to detect and track vehicles.

1.1.2 Goals and Challenges

The task of this thesis is to create methods that can automatically detect and track multiple vehicles in both static and moving status from the video captured from the UAVs in complex environments. The aerial video is captured from a low altitude UAV that has a fixed, downwards-facing camera capable of capturing low-resolution video. The framework was developed under the assumptions that the UAV has only camera information without any other sensors such as GPS.

The challenges of vehicle detection and tracking can be summarised as follow:

Challenge 1: Different UAV angles, altitudes and positions cause vehicles to have different shapes, sizes and orientations in images. There are many challenges in the vehicle detection and tracking, such as the UAV's shooting angle, flight altitude and position can lead different detection results; also the size, shape and orientation of vehicles can change considerably in the images [11] (An example of the challenge is shown in Figure 1.3).

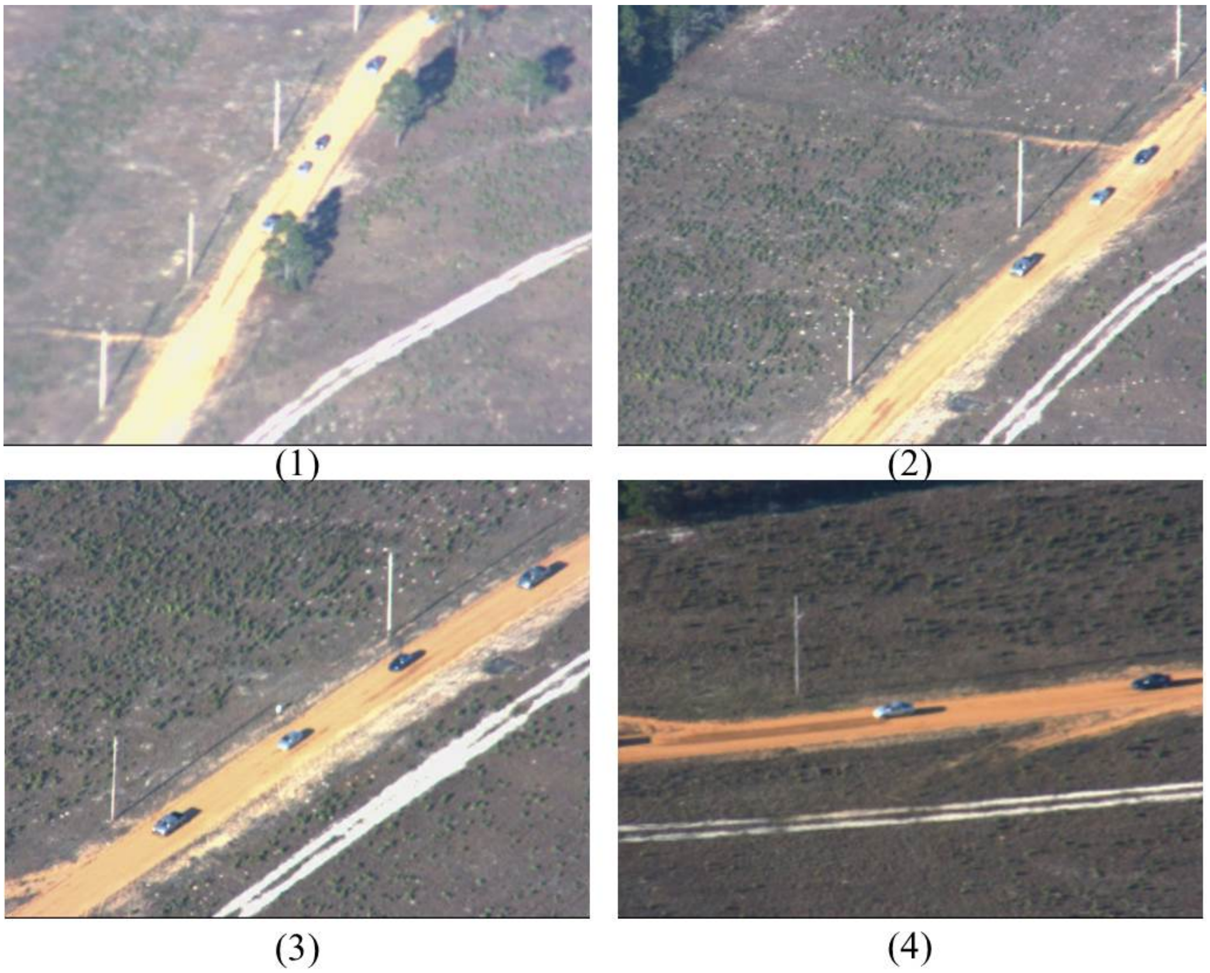


Figure 1.3: The appearance of the target vehicles can be changed during filming or between the capturing of images. The sequence frames from (1-4) indicate that the vehicles are changing their appearances in the images. The video was filmed from UAV by VIVID [4].

Challenge 2: The low resolution of the on-board camera can cause the shapes of vehicles to have simple textures. Another challenge is that, due to the low resolution of the captured video and the small size of the target in the images, the details of the vehicles are not clear enough, which makes them difficult to distinguish from similarly shaped objects [12]. In addition, the UAV is very sensitive to the atmospheric turbulence, which leads the blur images captured from UAV [13]. Figure 1.4 indicates the blurred images captured by a low resolution camera in UAV.

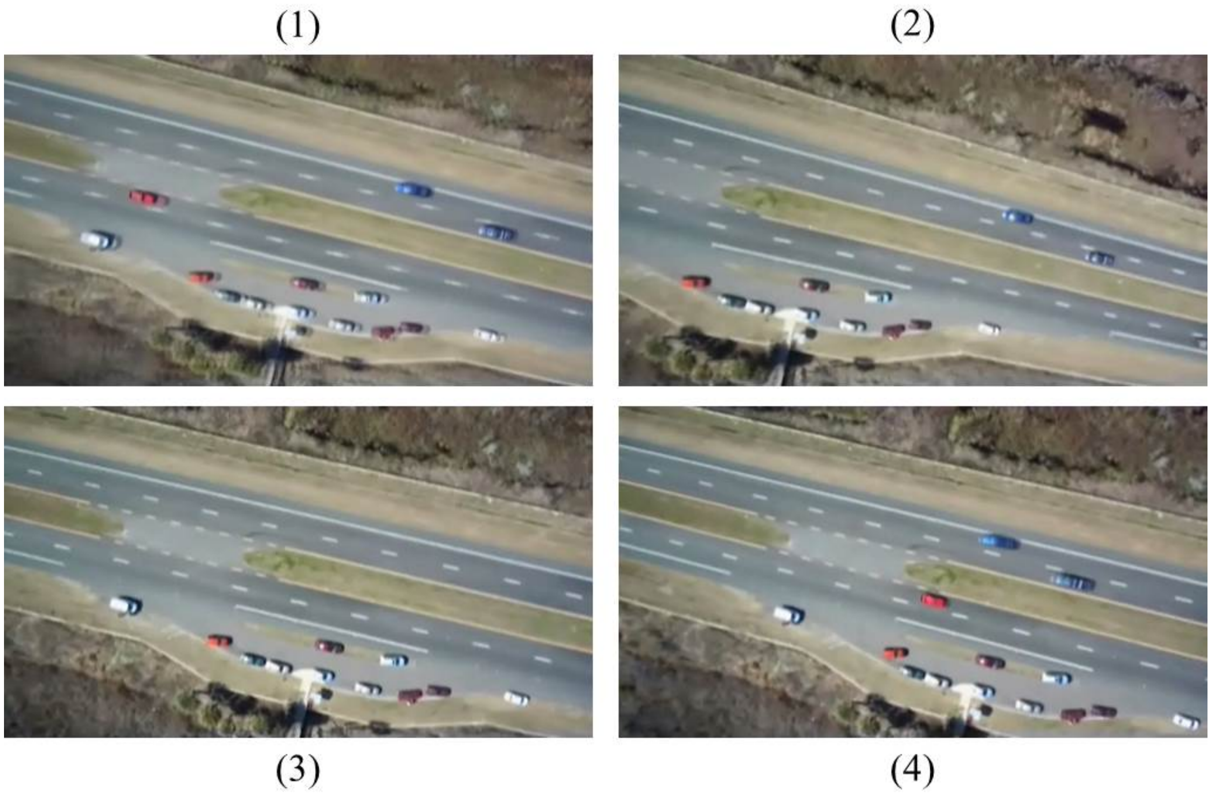


Figure 1.4: The camera can have low image resolution and the relatively small size of the vehicles in the image means many of the details of the vehicles are not visible. The images can also be blurred due to atmospheric turbulence. The video was captured from our experiment in the urban road.

Challenge 3: The background of the images can also affect the detection accuracy. Because the captured images are in low contrast, so the vehicles and the background are possible to have similar colours and features. Also, the lighting conditions in the video are changing, which require the detection to be invariant to the colour and illumination [4].

Furthermore, the artificial objects such as road-light, windows and buildings may have similar feature to the vehicle [5]. Figure 1.5 shows the example of complex background challenge situation.

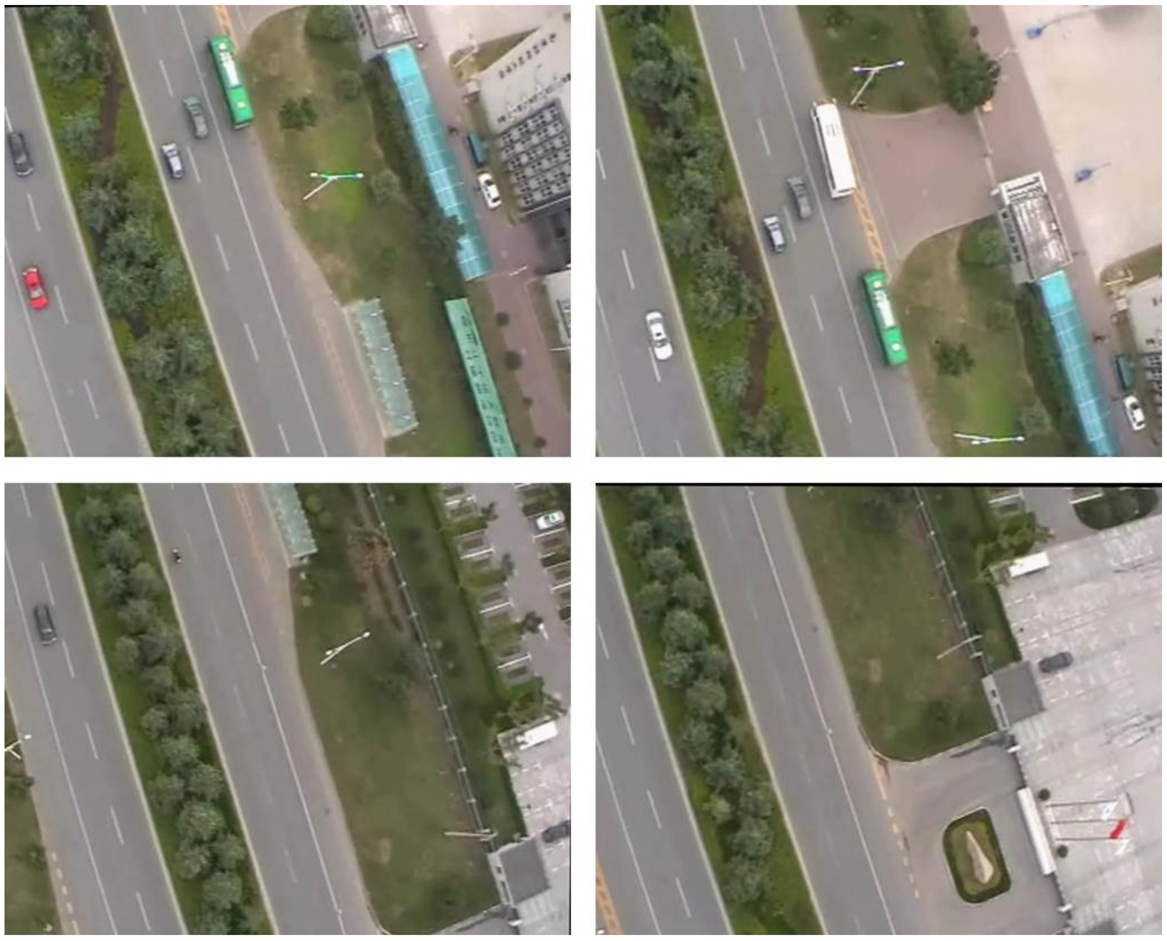


Figure 1.5: The background can be very complex and make it difficult to identify the vehicles. The video was captured in a unban road from UAV in our experiment.

Challenge 4: Tracking process involves the recognition of an individual vehicle in a set of sequence frames.

The difficulties are raised for the multiple targets tracking purposes [14, 15]. Firstly, vehicles are moving fast at variable speed and different directions. With different shooting angles and the speed of the UAVs may change. Thus, the vehicle motion between the frames is affected by both movement of the target and the UAV. Furthermore, the tracking method employed must also be capable of tracking vehicles in potentially clustered and complicated scenes, where there might be multiple vehicles with similar appearances moving in close proximity to each other and a potential for complete vehicle occlusions [16]. Figure 1.6 indicates an occlusion problem occurred during the video capture.

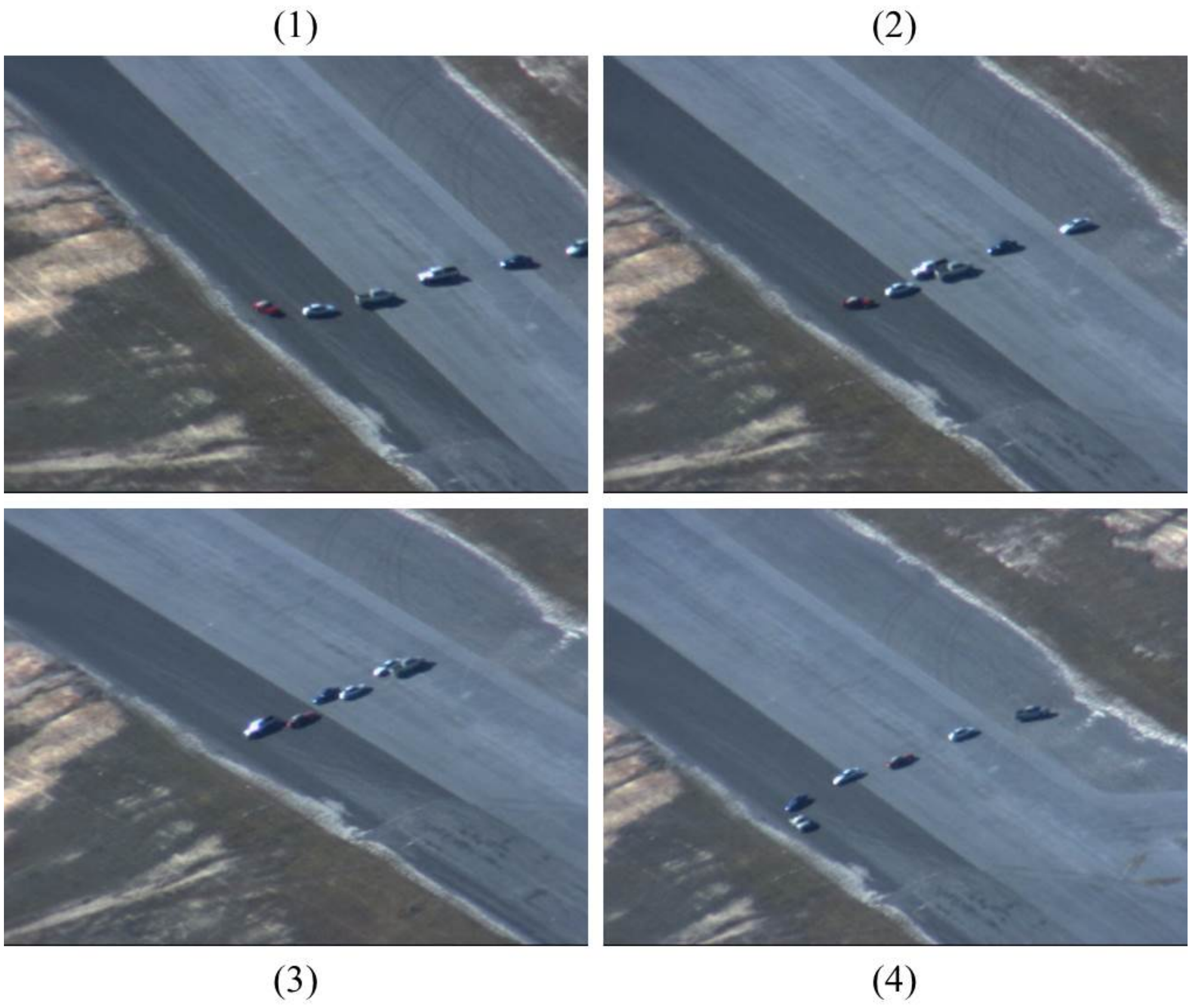


Figure 1.6: Occlusions can be occurred when vehicles are close to each other. The video was filmed from UAV by VIVID [17].

1.2 Key Contributions

This thesis builds on successful detection and tracking frameworks, which have already been proposed in the field of vehicle detection and tracking. A number of improved algorithms and algorithmic extensions in various areas of the original detection and tracking framework are proposed, which can successfully tackle the existing challenges. The system uses shape-encoding local features and machine learning techniques to learn a model of the appearance of vehicles. The proposed modification algorithms increase detection and tracking accuracy under certain circumstances related the challenging tasks in vehicle detection and tracking from aerial video. The following paragraphs will briefly outline the main contributions of this thesis.

Firstly, a HSV-GLCM algorithm with blur remove is proposed, which is a modified vehicle detection method. This detection method uses texture identification combined with colour recognition to distinguish complex backgrounds to target vehicles. This proposed method can successfully tackle the Blur and low resolution challenges. Experimentation has shown that significant improvements on the original GLCM algorithm can be achieved by using the new method. Additionally, a Blind Deconvolution algorithm has been integrated into the HSV-GLCM algorithm to solve the detection errors caused by the blurring problem.

Secondly, in order to improve detection performance and tackle the different angle, shape and orientations of the target challenge, a SIFT-ISM method of vehicle detection is proposed, which operates by detecting the SIFT points for predicting the centre of the vehicle. This allows the system to detect the appearance of particular shapes rather than textures, which can reduce the detection errors caused by the similarity between the target vehicles and the noise of the background. At the same time, the ISM contains a codebook system which can narrow the detection area, thus the process can become more efficient. The results of testing show that this method has a more stable detection performance and has more target information to provide for the tracking systems.

Furthermore, a new detection method using the FAST-HoG feature is proposed. With this method, a more efficient performance can be achieved in order to tackle the complex background challenge. The proposed method uses the assumption of the rectangle shape of the vehicle, which narrows the detection area by searching for the corners first. A modified HoG algorithm is proposed to detect vehicles, which tackles the orientation invariant problem of the original HoG algorithm. Additionally, a smarter training scheme is proposed which improves the way training data is exploited in order to generate better classification models, which has more reliability to separate target and background objects.

Finally, a framework of Self-Learning Tracking Detection (SLTD) is designed for vehicles (Figure 1.7). The proposed approach upgraded the existing Tracking Learning Detection (TLD) [18] approach; the proposed system can track multiple targets, rather than just a single target as tracked by the original TLD, in real-time. The tracking process involves a FBT tracking process with two learning inspectors, which can not only monitor detection and tracking performance, but also learn the existing environment by updating the detection classification model. In the FBT, a Tracked Vehicle Database (TVD) has been developed which contains all the information about the vehicles tracked previously. The TVD has been developed for templet-matching purposes in later tracking. The proposed tracking framework can tackle the challenges of long-term tracking (because it assumes that the targets can suddenly disappear and reappear in the captured videos), the problem of significant appearance changes, and the problem of short-term occlusions during tracking.

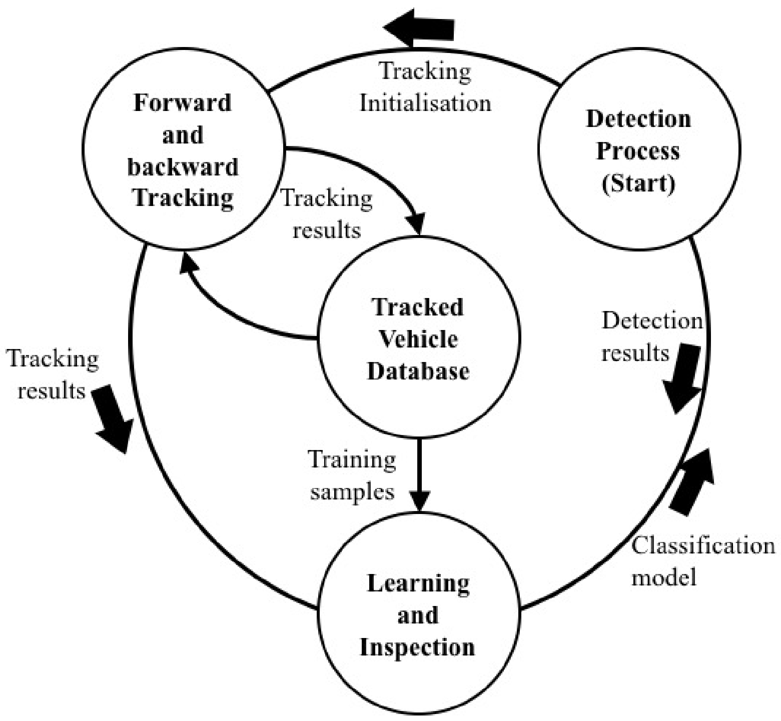


Figure 1.7: The framework of the SLTD approach.

1.3 Thesis Outline

This section provides an overview of the structure of this thesis. This thesis is organised into ten chapters and the summaries of each chapter are as follows:

Chapter 2: Related Work

The second chapter begins by reviewing a broad range of publications relevant to vehicle detection and tracking. This chapter briefly address the related methods and algorithms developed with the thesis, as well as the advantages of the method compared with others.

Chapter 3: HSV-GLCM and Blur Remove

This chapter introduces the HSV-Grey-Level Co-Occurrence Matrix (HSV-GLCM) detection algorithm. The chapter includes an explanation of the basic algorithm and its implementations. This chapter indicate how the parameters were set step by step. It also explains the blur remove system used in the detection algorithms. Examples of the detection results are shown in the end of the chapter.

Chapter 4: The Proposed ISM-SIFT Algorithm

The Implicit Shape Model with the Scale-Invariant Feature Transform (ISM-SIFT) algorithm will be described in this chapter. The chapter introduces the general concept and formulation of the algorithm. The differences and the improvements from the original SIFT and ISM algorithms will also be addressed, as well as the detection results. This chapter conduct a comparison of the feature descriptors and interest point detectors in the Implicit Shape Model framework and show that significant performance improvements can be achieved by choosing the right descriptor.

Chapter 5: The Proposed FAST-HoG Algorithm

In Chapter 6, a Feature from Accelerated Segment Test and Histogram Oriented Gradient (FAST-HoG) approach to vehicle detection is proposed. This chapter describe the FAST-HoG algorithm, and evaluate the strengths and weaknesses of all the detection approaches have addressed. The vehicle detection methods employ both region-based and point-based features. This chapter also will make a comparison between these two approaches and explain the reasons for the decisions.

Chapter 6: The Proposed Self-Learning Tracking and Detection Algorithm with Forward and Backward Tracking

The tracking system will be addressed in this chapter. The tracking system is formed of a Self-Learning Tracking and Detection (SLTD) system and a Forward and Backward Tracking (FBT) system. It is assumed that the processes of both detection and tracking can include errors, so that both processes should monitor each other and also keep up-to-date the detection and tracking results. A Forward and Backward Tracking (FBT) mechanism between detection and tracking is proposed; the main purpose of using FBT is to check whether there are any errors in the results of the detection and/or tracking process. The FBT has a Tracked Vehicle Database (TVD) which stores SIFT information about previously-tracked vehicles. The FBT results indicate whether all matching results between the trackers in the frame sequences are from the same vehicle.

Chapter 7: Conclusions and Future Improvement

This chapter will close the report and provides a conclusion on the outcomes of the project, followed by suggestions of future work to improve the system.

Chapter 2

Related Works

On-road vehicle detection and tracking has been a topic of great interest to researchers over the past few decades. Several sensing modalities have been used in detection; these include radar, LIDAR and computer vision. Image processing technology has progressed immensely in recent years. The very first computer image process of object recognition can be tracked back to the 1960s’ and 1970s’. Since then, several detection methods have been proposed. Recently these methods have improved significantly for certain applications. Thus, the image process of object recognition can handle different tasks by using particular methods and algorithms. This chapter briefly discusses the literature related works on vehicle detection and tracking.

Firstly, the processes involved in any approach to vehicle recognition are classified. Basically, vehicle recognition can be divided into four sections:

- Identification of the object’s appearance
- Classification of the object into one of a number of different categories
- Detection of the target object/s
- Tracking of the target object/s

Object identification addresses the unique features of a known object which can be identified by the computer as distinct from objects from other categories, and from background clutter in certain images and videos. Object classification tries to find similarities with a whole object category which can distinguish the object from others; a classification model can then be created for further detection. The object detection method is the next step of the overall process and can localise the position and other features of the target in the image. Object tracking is based on the detection results; a tracker is assigned to the targets which can re-detect the target in the following frame of the sequence.

This thesis focused on vehicle detection and tracking algorithms, in particular vehicle detection from aerial videos. The primary target is to develop a system which can automatically detect and track multiple vehicles in a video captured by a UAV. The system should give the user the position and appearance of any static and moving vehicles that appear in the images.

Vehicle detection and tracking is one of the key object detection research areas, as many approaches are applicable in the field. These approaches have to deal with similar issues, such as complex background clutter, image blurring problems, etc. This thesis builds upon several generic object detection approaches which perform vehicle detection and tracking. The literature review is split into three parts: vehicle detection, vehicle tracking and machine learning.

2.1 Vehicle Detection

Vehicle detection refers to computer vision-object recognition, which is the scientific study of how machines, rather than human eyes, see. The basic task of vehicle detection is to let the system localise one or multiple vehicles in an input image. Vehicle detection is based on two methods: local features method [19] and the sliding window method [20]. Local features-based method approaches always follow the principles of finding the feature of one object or a class of objects first, and then categorising these features into classification models, which leads to the last step, in which the system identifies which group the testing object should belong to, and decisions are made based on the classification models. The main strength of this method is that features and geometry of the target objects are known before the detection of the target's features. However, this strength can be the limitation of the method as it means the system can only detect targets from pre-learned object classes. A sliding window-based system works by scanning the input image in a certain number of windows of various sizes; it then decides whether the target appears in each window or not. It is noteworthy that, in a normal testing image, there are approximately 50,000 patches that need to be evaluated in every frame. So in order to achieve real-time detection performance, a sliding window-based method adopts a cascaded architecture method [21] which assumes that the sliding windows are more likely to process the background rather than the target area;

a classifier is therefore applied at a number of stages for the rejection of background patches, which can reduce the overall number of stages that have to be evaluated.

For both methods, pattern recognition is an essential element of detection. Pattern recognition requires local descriptors to correspond to the object detection models. Local descriptor models have become popular in the pattern recognition field, and represent the object or a class of a kind of object. Generally, descriptors are grouped and clustered to represent a vocabulary or codebook of a type of object’s features, and a new test image is used to compare these features and match them with the vocabulary or codebook to distinguish the targets from the background clutter. Pattern recognition involves two kinds of classification: Supervised Classification and Unsupervised Classification. Generally speaking, supervised classification needs a lot of samples to support it, and the kind of measurement which is used to classify the target object may change. It is worth noting that, normally, both the local features-based method and the sliding window method are used with Supervised Classification, which means that a large number of training examples are needed to train the detector. These large numbers of training samples and intensive computations are used to accurately separate the decision boundary between the target and the background. In vehicle detection approaches, the features of the vehicle are used to create a generic model, constructed by representing the shape of the vehicle with a 2D/3D model e.g. a wire-frame representation. Detection is performed by extracting the features such as edges and grouping them together to create structures that can be compared to the model. These methods have proven very robust. However, because of the small size of vehicles in aerial imagery, models must be simplistic and are therefore in danger of matching many regions in the image. The inherent weaknesses of this approach are that the models generated by the classifiers are dependent on the training data and often struggle to generalise well as it cannot be assured that the training data captures changes in illumination, pose and other possible influences on vehicles’ appearance from neighbouring objects and the relative position of UAVs, which refers to the low resolution and blur; different size, orientation and illumination challenges.

X. Wang et al [22] proposed a method of detecting objects in UAV footage by using edge detection (Figure 2.1). The proposed algorithm gives us an example of using edge detection to recognise artificial objects. This method uses an edge detection algorithm to detect the straight lines on a vehicle. The method removes background noise by using a higher threshold in its edge detection. Not only detecting vehicles by their edge texture features, J. Sokalski et al [23] also proposed a method that uses edge detection and colour features to distinguish natural and artificial objects. The difference is that in this paper, the authors extract nine features from the different colour channels of the original image and use these features to identify edges and changes of edge to distinguish natural and artificial objects (Figure 2.2). These two papers gave us the idea of summarising local descriptors into colour features and texture features. The process is faster when using the colour feature to detect objects, but the accuracy is challenged, especially in scenarios that have complex backgrounds. The objects in images are more accurately represented by the texture feature, because texture holds more information about an object. However, both papers use detection with Unsupervised Classification, which means no training occurs before detection, so neither proposed method can detect any specific objects.

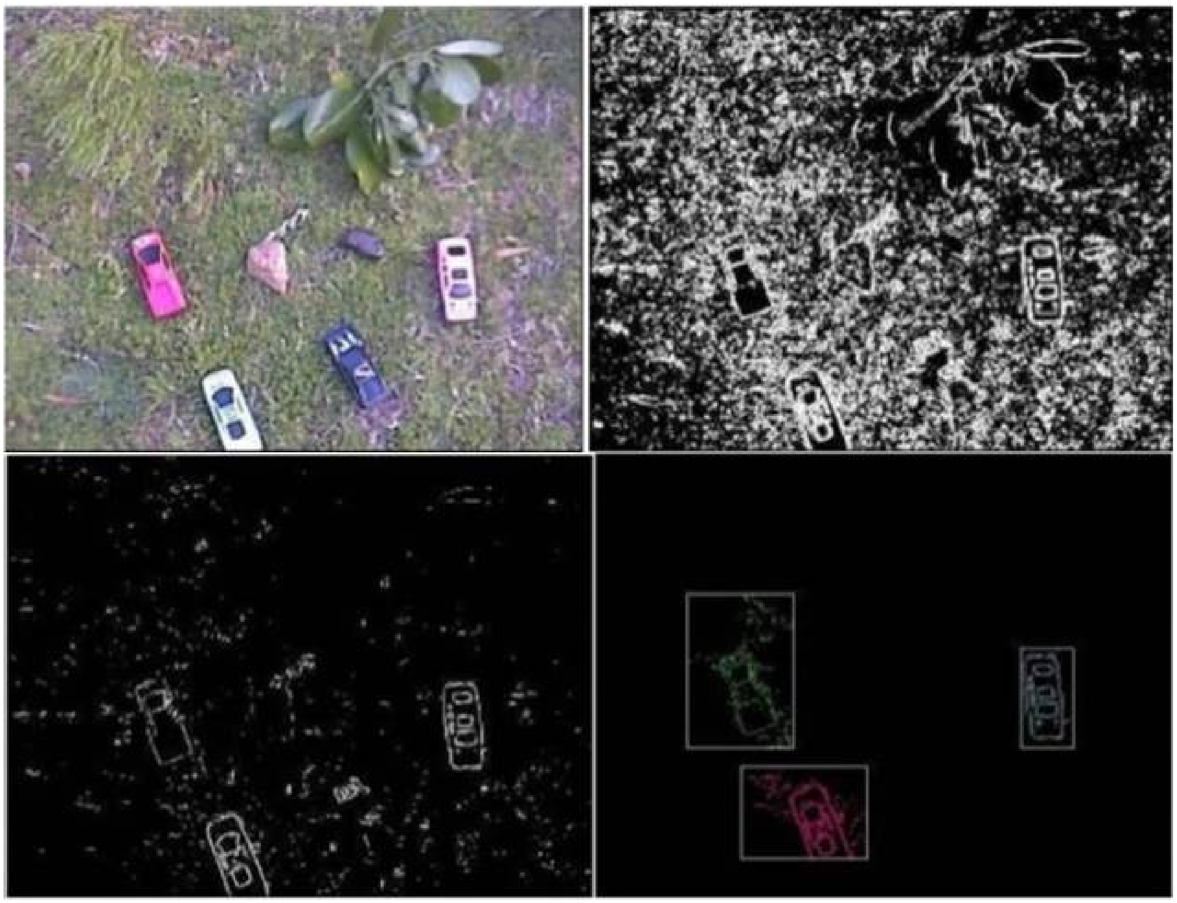


Figure 2.1: Examples of vehicle detection using edge features at different thresholds. The background noise can be removed in higher threshold detection. [22]

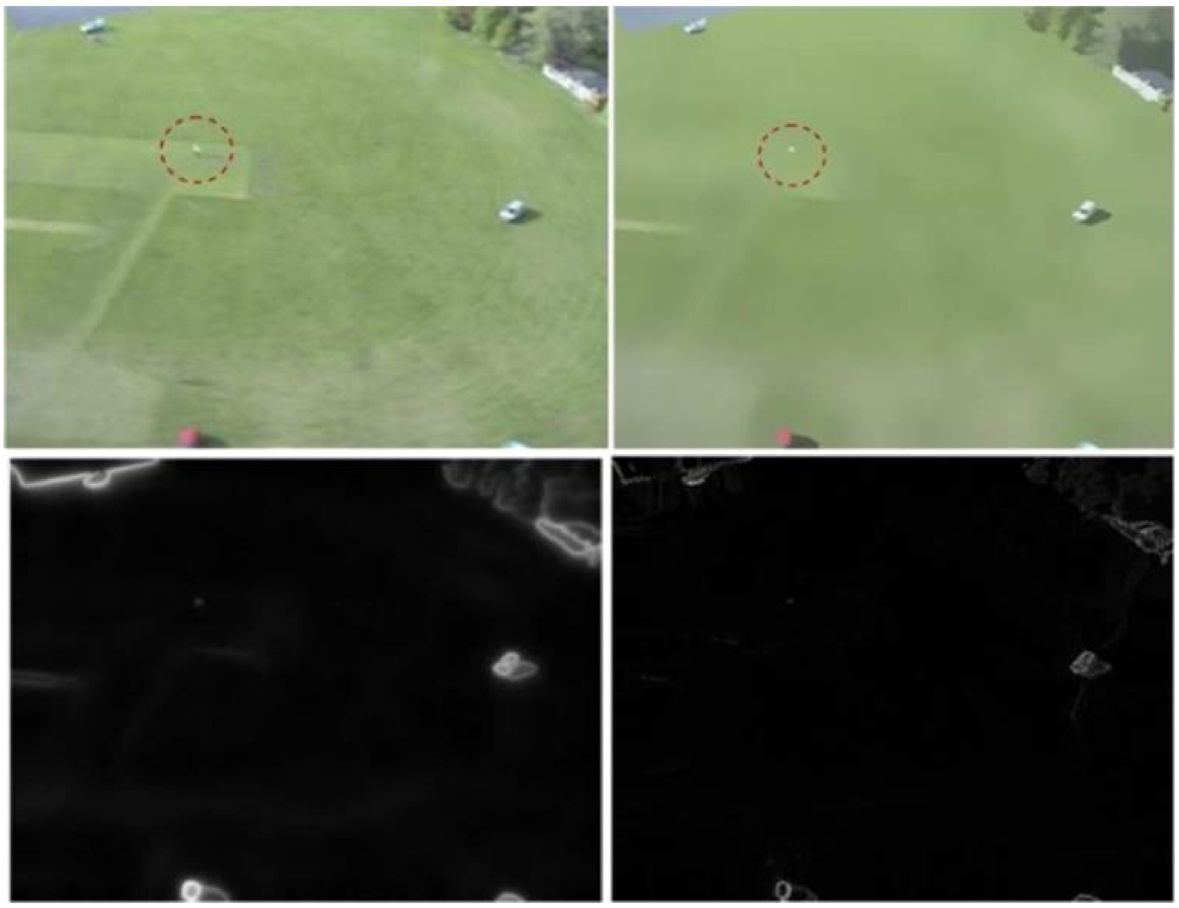


Figure 2.2: Object detection using edge and colour features. This method uses mean-shift segmentation and edge features to detect artificial objects. [23]

The two papers above use texture and colour features for object detection, which is typical for object detection. Using colour features to detect objects is the simplest method. N. Baha et al [24] presented empirical landmark recognition experiments of using colour and texture features. Once landmarks are recognised from a class of geo-referenced images, they can be used to estimate the UAV’s position and help autonomous navigation. This paper uses one set of aerial geo-referenced images and another set of aerial images of the same region, collected at a different time, which are not geo-referenced. This work presents a landmark recognition system based on the extraction of colour and texture features. The proposed method uses these features to provide information about surface orientation, shape and colour. This approach is being studied for application in a PITER (Real-Time Image Processing) research project that is being carried out at the Institute for Advanced Studies (IEAv – Instituto de Estudos Avancados), and applied in autonomous UAV navigation based on images. In this project, 126 samples are used for supervised learning, which is a particular type of machine learning algorithm that allows the prediction of the class of a previously unknown instance based on the knowledge of the class of a training sample. A HSV colour and Gray Level Co-occurrence Matrix (GLCM) feature is used to train the classifier the detection. The method uses a sliding-window approach in detection. In this paper shows clearly that the combination of colour and texture features can improve the accuracy of detection; the usage of the Supervised Classification method can also improve detection performance.

Saman G et.al [25] proposed a vehicle detection approach by the density measurement. The gradient vectors have been calculated in the edge map of the aerial images. It is proposed that, the directions of the gradient vectors are changing significantly in the boundary of the target and by calculating the standard deviation of the gradient vectors. So by predefine the threshold, the vehicles can be detected within the value. The evaluation used the aerial images taken from road in Turkey and achieved 86% accuracy in F-measure. However, such detection methods without the training have a common drawback, which the target objects are difficult to distinguish from the complex background situations.

Peng W et.al [26] proposed a vehicle detection method that improves background extractor. This approach used pavement segmentation with 8-neighborhood filling to removing the road markers in order to filtering the complex background. The result indicates that the proposed method can avoid the detection errors caused by the misleading of the road markers. However, this detection method is based on fixed camera detection, which is contrary to the purpose of this thesis, but the background extractor mechanism can be adopted.

J. Susaki et al. [27] proposed a two-stage approach to the automatic detection of vehicles within aerial imagery. Their approach is based on the use of multiple cascaded Haar classifiers [28] for vehicle classification and a secondary verification stage that attempts to eliminate non-vehicle candidates based on UAV altitude and vehicle size constraints. The cascaded Haar classifier provides a reliable detector that is invariant to vehicle colour, type and configuration. To achieve vehicle orientation invariance, they use four separate cascaded Haar classifiers trained in sample vehicle images categorised into four positional orientations. The four classifiers are then evaluated using a query image at multiple scales and positions using a sliding window approach and multiple classifier detection, and detection overlaps are resolved using a spatial merging technique (Figure 2.3). J. Berni et al. [29] later extended this work with the use of additional thermal imagery for thermal signature confirmation, which improved performance considerably.

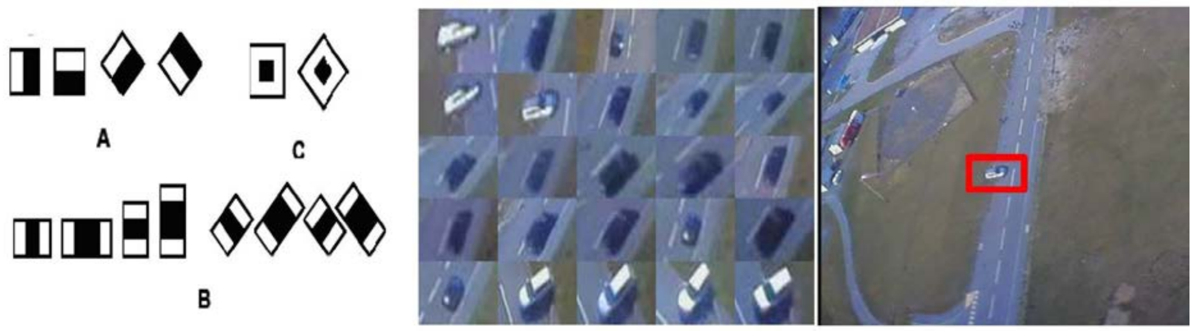


Figure 2.3: In the first image, A shows the Haar-like features; B shows the line features and C shows the centre-surround features. The second image shows the training samples for Haar-like features and the last image shows the detection results of the feature. [27]

Gleason et al. [30] focused on automatic vehicle detection in rural environments. Their approach consists of a cascade detection algorithm, which is comprised of two stages. In the first stage, a Harris corner detector is used to identify features of interest in the images; the authors argue that “vehicles in particular have a large number of edges and corners compared to natural objects”. Next, an efficient sliding window approach is used to determine regions with a feature density higher than a predetermined threshold. Overlapping regions are grouped together and further refined based on the colour characteristics of background areas. The regions selected in this stage are then put through image classification techniques in the second stage, which determine the presence of a vehicle. These authors’ research compared the performance of two image patch descriptors, a modified Histogram of Oriented Gradients (HoG) feature and Histogram of Gabor Coefficients features. They also investigated the performance of three statistical classification techniques; K-Nearest Neighbours (k-NN), Random Forests (RF) and Support Vector Machines (SVM). From the first stage of their algorithm, they obtained an average detection rate of 85% and found the top performing classifier to be Random Forests using Histogram of Gabor Coefficients features; this was capable of classifying 98.9% of vehicles and 61.9% of background images correctly. Sebastien R et.al [31] proposed a vehicle detection method in unconstrained environments with state-of-the-art object detection approaches, which is using the sliding window classification method with the SVM classifiers.

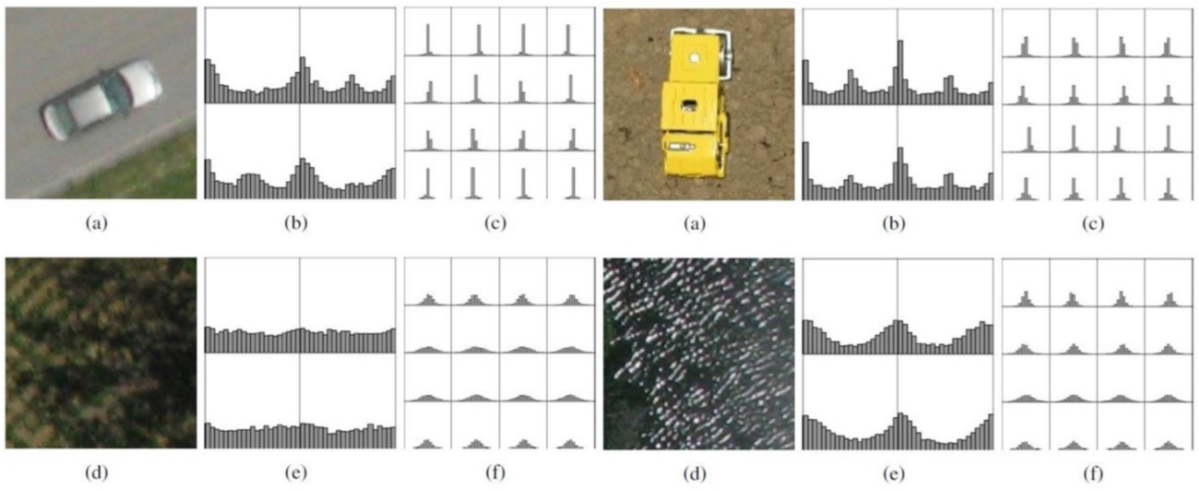


Figure 2.4: Two examples of HoG feature vectors: vehicle image (a); HoG vector (b); Gabor histogram (c); background image (d); corresponding HoG feature (e) and Gabor histogram (f). [30]

Nguyen et al. [32] proposed a boosting based framework for vehicle detection in aerial images. They used Haar features, Histogram of Oriented Gradient (HoG) features and Local Binary Patterns (LBP) with an online version of the AdaBoost training algorithm to select the most informative features. In this approach, each feature resembles a single weak classifier and boosting is used to select an informative subset from these features to create a strong classifier. To create a weak classifier from the features they model the probability distribution of each feature in vehicle and background images, using a Bayesian learning algorithm for the Haar features and a nearest neighbour learning algorithm for the two histogram-based features. Detection is performed by using a sliding window approach and threshold the output confidence values. Overlapping detections are grouped together by applying mean shift clustering to the probability density distribution of the classifier outputs.

Histograms of Gradients (HoG) have become very popular and widely used in object detection in recent years. HoG has proved that it is robust in various lighting conditions and generalise well for object recognition. Dalal and Triggs [33] presented an evaluation of the use of HoG for object detection and compared it with other approaches, such as Haar wavelets; the HoG has an advantage in object detection. One of HoG’s main aims is to locally normalise image gradients so low contrast areas can be compensated for certain images. The authors proposed a method that divides the detection window into small, overlapping rectangular or circular blocks of various sizes. Each block is subsequently divided again into a number of cells; the original concept divides each block into 2×2 cells, but it is noting that the block size and the cells can vary. The HoG orientation of each cell in the block is computed. The HoG interpolates both spatially, between neighbourhood cells, and with respect to the gradient orientation angle. This can avoid discretisation during processing. Thus, each block can concatenate a block feature vector and the concatenated normalisation of these block features vectors are the final descriptors of the HoG. Dalal and Triggs trained an SVM classifier model based on two sets of training samples: positive and negative images of object in the target class and background objects. Their evaluation shows that they achieved low false positive rates by using their algorithm.

Xianbin Cao [34] created a new feature a boosting Light and Pyramid Sampling Histogram of Oriented Gradients (bLPS-HoG). This method was proposed for use together with a linear support vector machine (SVM) for vehicle detection. The output of vehicle detection is denoted by regions which may potentially contain vehicles. These regions are refined for the improvement of vehicle detection performance and the computation of the trajectories of vehicles for measuring traffic information (Figure 2.5). A Spatio-Temporal Appearance-Related Similarity (STARS) measure is proposed for analysing the motion of detected vehicles. The STARS measure can help to effectively correlate vehicles from different frames and obtain the vehicles' trajectories for analysis. This paper offers two major contributions: first, for the speeding up of feature extraction and the retention of additional global features at different scales for higher classification accuracy, a bLPS-HoG feature extraction method is proposed; second, to efficiently correlate vehicles across different frames for vehicle motion trajectory computation, a STARS measure is proposed. Compared to other representative existing methods, the experimental results showed that this proposed method is able to achieve a better performance, with a higher detection rate, a lower false positive rate, and faster detection.

Kang et.al [35] proposed a Fast-Multiclass Vehicle detection approach by using the HoG feature extraction. The method used two stages approach for the detection including a binary sliding-window detector and a multi-direction detection. The method sets a fixed 35 × 70 pixels scale detector for the sliding-window approach by using the HoG to classify the target with AdaBoost classifier in a soft-cascade structure. The HoG factors were set in 4 × 4 cell size and 1 × 1 block size, which was the best performance the authors got. The detection result has reached 98.2% in their testing images. They also suggested that the detection window can be increase in a higher resolution testing images.

Sahli et al. [36] proposed a local feature-based approach for automatic vehicle detection in low-resolution aerial imagery. Their approach was developed with the aim of being free from the constraints related to detection methods based on a vehicle's visual appearance, i.e. a vehicle's rectangular shape and the presence of frontal and/or rear windows. Their approach is based on the extraction of Scale-Invariant Feature Transform (SIFT) features from vehicle and background images. These features are used to train a Support Vector Machine (SVM) classifier to define a model that can be used to classify SIFT features extracted from the cars and background in a query image. The collection of SIFT features that are predicted to belong to a vehicle are then clustered in the 2D image space into subsets associated with individual cars. The authors' clustering method is based on a modified Affinity Propagation (AP) algorithm that is bound by the spatial constraints related to the geometry of vehicles at the given resolution. They obtained a classification accuracy of 95.2% in aerial imagery of a parking lot containing 105 cars, with no false-positive detections.

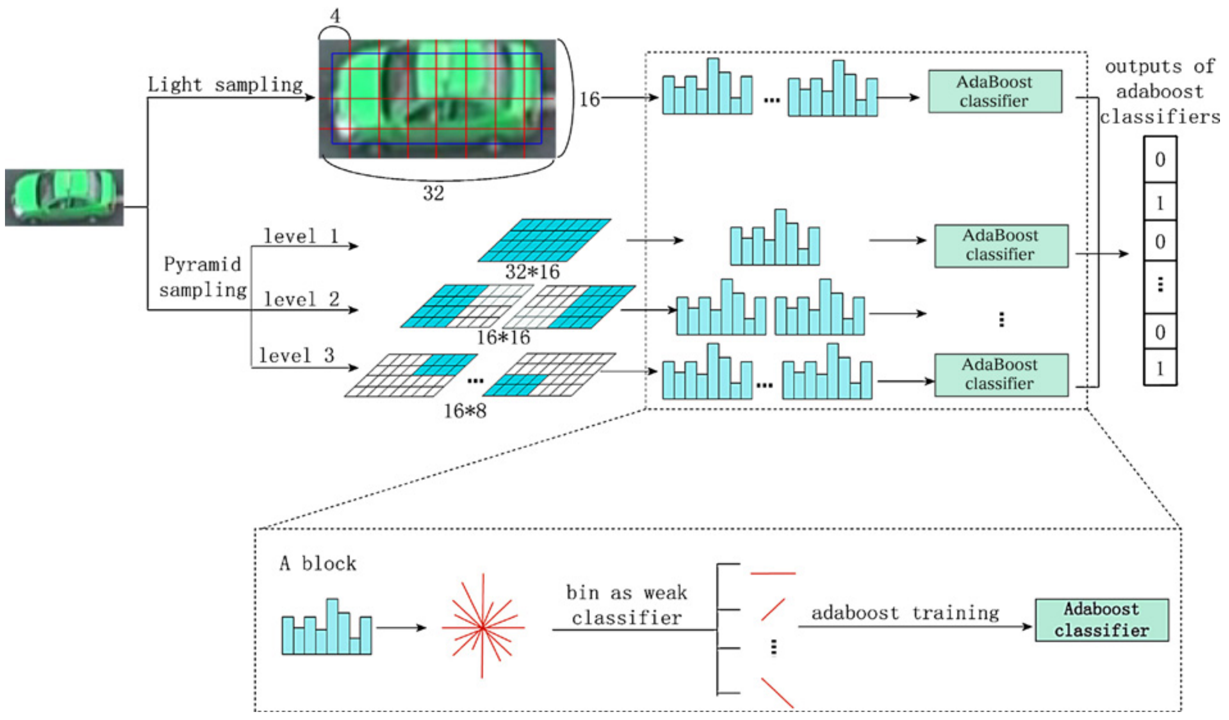


Figure 2.5: The workflow for the extraction of bLPS-HoG features. [34]

Abdulla et.al [37] developed a context-driven framework that uses scene context to improve the detection of moving vehicles in urban environments. Their approach is comprised of three stages; motion detection, vehicle detection and an online method of road network estimation for detection filtering. To detect moving objects, they use an image stabilisation technique, through which Speeded Up Robust Features (SURF) features are registered across successive frames and fitted to an affine model for image warping. With stabilised frames, background subtraction is used to detect regions that do not move according to the estimated homography. A cascade of Support Vector Machine (SVM) classifiers that use shape and size and Histogram and Oriented Gradient (HoG) features are then used to detect vehicles in the candidate regions. In the final stage, multi-object tracking is used to track the trajectories of the detected objects in order to estimate the road network and use this information to discard detected vehicles that are not on the roads. These authors evaluated their system using the CLIF data set and obtained positive and negative classification rates of 0.843 and 0.797 respectively for their cascade classifier.

Jinhe Lan et al. [38] proposed a new framework of multi-motion layer analysis to detect and track moving vehicles from airborne platforms. Moving vehicles are first detected by registration and temporal differencing to establish motion layers. After the motion layers are constructed, they are maintained over time for tracking vehicles. All vehicles are tracked by maintaining their corresponding motion layers. The proposed framework is based on features that are tracked from frame to frame. Firstly, background features are separated by region division and motion consistence to form the background motion layer. By using background layer features, an affine transformation can be computed to register a frame to its previous frame. After registration, temporal differencing is applied to detect moving objects. Features that belong to vehicles are clustered in a coarse-to-fine mode to form different vehicle motion layers, i.e. individual feature groups. Thus vehicles are tracked by maintaining the related vehicle motion layers.

Motion detection methods were also used by R. Wu et al. and D. He et al. [39, 40]. Both of these authors used motion features to detect moving vehicles. The approach proposed performs object detection by using the background subtraction technique. Background subtraction considers the static part of an image as background, and the difference between an image and its corresponding background model is considered the foreground. One famous background subtraction method is the background mixture model, based on Gaussian mixture [41, 42].

There lots of other features of object detection those have been used in other field. In paper [43], a feature called Maximally Stable Extremal Regions (MSER) [44] is used to describe images' features. MSER is affine invariant regions with irregular shapes, which are highly robust to changes in 3D viewpoint, altitude and illumination. Compared to some other affine invariant region detectors, such as Harris-affine [45] and Hession-affine [46], the MSER detector is better in terms of anti-interference ability, robustness and speed. Therefore it is widely used in wide baseline stereo matching, image indexing and so on. In paper [47], a feature Local Energy based Shape Histogram (LESH) is used for the recognition of road traffic signs. The Local Energy Model was first introduced by M. Concetta et al. [48] proving that features can be extracted at those points in an image where local frequency components represent maximum uniformity. In the paper [49] the LESH was used for the vehicle make and model recognition.

In drawing a conclusion about global approaches to vehicle detection, this can state that the methods use two kinds of local descriptor: area local descriptors and point local descriptors. Area local descriptors are descriptors that are extracted from regions in an image which may represent certain objects. On the other hand, point local descriptors the points extracted from an image to represent the information about a single point in the image.

Area descriptor methods	**Point descriptor methods**
Grey Level Co-occurrence Matrix (GLCM)	Scale Invariant Feature Transform (SIFT)
Maximally Stable Extremal Region (MSER)	Principal Components Analysis SIFT (PCA-SIFT)
Local Energy based Shape Histogram (LESH)	Affine-SIFT (ASIFT)
Local Binary Feature (LBF)	Speed Up Robust Feature (SURF)
Shape-let Feature (SF)	Harris Corner Points (HCP)
Implicit Shape Model (ISM)	Features from Accelerated Segment Test (FAST)
Haar-like Features (HF)	SUSAM Corner Detector (SCD)
	Scale Invariant Feature Transform (SIFT)

2.2 Vehicle Tracking

Beyond recognising and identifying vehicles from the current frame, vehicle tracking is also the task of the prediction of predicting vehicles' positions and motion characteristics in upcoming frames in a video. This tracking is based on a vehicle's position in the previous frame according to the vehicle detection results. It is also worth noting that tracking typically assumes that the targets are visible throughout the process. Various approaches to object tracking are used in practice, for example, tracking by points articulated models, shapes and optical flow.

P.Sand et al. [50] proposed a new approach to motion estimation in a video sequence. The authors represented video motion by using a set of particles. Each particle represented a single point sample in the input image, with a long-duration trajectory and other properties. They matched points along the particle trajectories with the distortion between the particles. This approach is useful for various applications which cannot be directly achieved by using existing methods such as optical flow or feature tracking. It also increased the accuracy of motion estimation in challenging real-world videos, including complex scene geometry, multiple types of occlusion, regions with low texture, and non-rigid deformations.

J. Ding et al. [51] proposed an automatic tracking system. The system has two stages: firstly, it builds a model of the appearances of people in a video and then it tracks those people based on detecting those models in each frame (“tracking by model-building and detection”). They developed two algorithms to form the model: a bottom-up approach, which groups together candidate body parts found throughout a sequence and a top-down approach that automatically builds people-models by detecting convenient key poses within a sequence. The tracking results of this method have shown that building a discriminative model of appearance is quite helpful since it exploits structure in a background.

V. Diaz-Ramirez et al. [52] derived a probabilistic framework for robust, real-time, visual tracking of multiple previously unseen objects from a moving camera. The algorithm uses objects’ poses and shapes to observe the image. The poses are formed by a group of transformations and the shapes are represented by an implicit contour level set. These methods also demonstrate how motion models can be incorporated within the same probabilistic framework and show how this enables the system to track complete occlusions. This method has proven very effective in a variety of challenging video sequences.

Vehicle tracking using the approach of object tracking can be split into two kinds of vision approach: **partial detailed monocular approaches** and **stereo vision approaches**. In both kinds of tracking, estimation and filtering methods are applied to tracking algorithms.

2.2.1 Monocular Vision Tracking Approach

In monocular vision, vehicles are detected and tracked in an image sequence. Tracking vehicles by using the monocular method has two main aims: firstly, the estimation of the vehicles' motion and prediction of their positions, and secondly, enforcing temporal coherence, which can maintain awareness of the detected vehicles from a previous frame that disappear in the following frames and filter out the false positives during the tracking process. Oron et al. [53] proposed a learning algorithm for vehicle tracking. This method used a Harr-like feature matching process with a weighted correlation. The object tracker searches features that have already been calculated for detection, thus it is possible to reuse features for detection and tracking. The weight values of features of the vehicles are optimised for the tracking system. Additionally, the kalman filter is applied.

Template tracking is a unique vision-based tracking method. The object is described by a template, which is normally an image patch or a colour histogram. Template tracking can also be divided into static template tracking, where the template of the target does not change during the tracking process, and adaptive template tracking, which means the template has extracted from previous frames. W. Liu et al. [54] proposed monocular vision-based rear vehicle detection and tracking system. In their tracking system, they used template tracking method; the templates are dynamically created online based on the detection results, and they estimated the motion to adaptively adjust the tracking window. In the other words, their tracking approach is based on the similarity in appearance of the vehicles from frame to frame. This can define this method as appearance-based tracking, that is, based on cross correlation scores during the tracking process.

Working from the appearance-based tracking approach, vision-based tracking is taken one step further by using feature-based tracking. In W. Zhao et al.'s [55] approach, the target vehicles' locations are predicted using a kalman Filter in the following image. A similarity feature score takes place in the local search of the input image during the tracking process, which gives a prediction of the target even if detection has failed in the input frame.

Optical flow also has been proposed for the tracking process; this can predict the target's position by measuring the displacement of interest points. M. Li et al. [56] proposed a detection framework using appearance, scene geometry and vehicle motion. They applied an optical flow motion detector and it was shown that this approach greatly improves the robustness and reliability of a detection system.

Bayesian filter techniques are widely used in vehicle tracking literature. The filter techniques can be divided into two methods: the particle filtering technique [57] and kalman filtering technique [58].

Particle filter, also known as Sequential Monte Carlo (SMC), is a method that uses online posterior density estimation algorithms to predict posterior density. The particle filter takes a sampling approach, using a set of particles to represent posterior density. Each sample has a set of particles, each of which has weight assigned to it. The weight represents the probability of these sample particles from the probability density function.

Y. Liu et al [59] proposed an automatic vision-based system to detect and track vehicles on highways under various lighting and weather conditions. The authors generated the distribution of probabilities on the vehicles as part of a particle filter framework. A modified particle filter was proposed, which can detect multiple targets with a single particle filter through a high-level cluster tracking strategy. The particle filter vector typically consists of the pixel coordinates in a rectangular area in the image and the pixels' moving velocities.

C. Idler et al [60] proposed a multi-instance-based multi-target-tracking method using particle filters. The authors developed a robust system that can process the real-time tracking of vehicles using early-vision image features. In the object localisation and tracking processes the probabilistic approach utilising a particle filter is taken. The system calculate the particle weighting vectors for multiple sets of particles to track different vehicles simultaneously, rather than a single target, as in the original particle filter approach.

S. Sivaraman et al [61] proposed a general active-learning framework for robust vehicle detection and tracking on-road. The framework involves a novel active learning approach to vehicle detection and tracking systems. The proposed approach used a classical supervised learning system in detection and an active learning-based model was created. In their tracking system, the authors integrated a particle filter into the proposed model to build an Active learning-based Vehicle-Recognition and Tracking (AlVeRT) system. Toledo-Moreo et al [62] and Lee et al [63] also used particle filters in their tracking systems.

Kalman filter, also known as linear quadratic estimation, is an algorithm that uses a measurement series o which estimates variables over certain of time. It keeps track of the estimated state of the targets, as well as the uncertainty of the targets. The kalman filter works in two parts; a prediction step and an average weighting step. Once estimations of the current state have been calculated, these estimations are updated using the weighted average.

Fontanelli et al. [64] proposed a real-time vision-based side vehicle detection system that employs a parts-based boosting algorithm. Window-based tracking is employed in their system. Kalman filtering is also used to predict the position of each part of each vehicle in the input image. The proposed approach can relocate tracking windows effectively during the tracking process, and the system improves efficiency and accuracy under different lighting conditions, changing vehicle poses, and partial occlusions.

J. Arrospide et al [65] used a kalman filter in their tracking system to smooth the trajectories of the predicted targets. They used a central outlier rejection stage to strengthen the tracker based on probabilistic techniques. B. Aytekin et al [66] also used a kalman filter to estimate the position and size of vehicles’ shadows in the following frames.

2.2.2 Stereo Vision Approach

Vehicle tracking using stereo vision measures and estimates the position and velocity of the target vehicle. The state vectors often include the lateral and longitudinal positon, the width and the height of the vehicle, as well as their velocity. Like in monocular vision tracking, a kalman filter is widely used in prediction; linear motion and Gaussian noise are considered optimal for prediction [67]. Kalman filter is widely used for stereo vision vehicle tracking [68]. Noise in stereo matching is generally modelled as Gaussian noise [69, 70], and filter over time can make a cleaner disparity maps in the tracking system [71].

Having noticed that vehicle motion is nonlinear in real life; A. Barth et al [72] proposed an Extended Kalman filter (EKF) to predict nonlinear parameters by linearising the motion equations for estimation. The filter combines the movement information of points on the background with a dynamic model of the vehicle. The EKF is widely used in stereo vision vehicle tracking, especially in the cases of nonlinearity in motion and observation models. The EKF was used to predict turning behaviour in [73, 74].

M. Grinberg et al [75] proposed that the EKF is significantly affected by the position of the camera in vehicle tacking. The authors used a side-mounted stereo camera to observe the nonlinear motion of the target vehicles. Z. Duan et al. [76] used EKF to model the nonlinearity of the 3D position of vehicle into a stereo image. B. Kitt et al. [77] used EKF combined with Longuet-Higgins-equations and optical flow to estimate the motion.

Particle filter has been also fairly commonly used for vehicle tracking. To be specific, particle filtering can be defined as an alternative method to EKF in the estimation of nonlinear parameters. A particle filter has multiple hypotheses which are weighted by a likelihood function. G. Catalin et al. [78] proposed a vehicle tracking particle filter system based on a grey histogram with sparse optical flow detection in stereo images. The proposed approach used a 2D tracked feature to compute 3D correspondence, which can improve particle filter tracking performance. This method integrates vision based particle filter tracking with stereovision of optical flow, which creates a robust object tracking algorithm.

C. Hermes et al. [79] proposed a hierarchical tracking system which includes optical flow prediction and kinematic prediction with a particle filter-based motion pattern method based on the trajectories of learned objects. This method achieves higher accuracy than the standard kinematic prediction method.

R. Danescu et al. [80] proposed a novel occupancy grid tracking method that uses particles for tracking the dynamic driving environment. The particles in the system have hypotheses which are also used to build model blocks with the parameters of position, speed and the motion. The system has a weighted resampling mechanism which can create and destroy specific particles, which is the same as the original particle filter algorithm.

This thesis focuses on the frame-to-frame tracking method, in which vehicles are represented by their geometric shapes and their motions between consecutive frames. As mentioned above, template tracking is the most straightforward method for a tracking system, and it includes static and adaptive tracking. These two template methods can also be combined in a tracking system. P. Singh et al. [81] and H.Lim et al. [82] proposed a tracking approach that has both offline and online tracking methods. This can make it relatively easy for the system to build trackers, and different trackers can be used in different situations, which improves tracking efficiency. However, template tracking has limited modelling capabilities because each tracker can only represent a single appearance of the target features. In order to build more appearance models, the generative models are been proposed.

Generative models can be built offline or online during the tracking process. M.J et al [83] proposed a matching system using an offline generative model which can particularly improve situations involving occlusion, background clusters, noise etc. X. Cheng et al. [84] and Z. Wang et al. [85] proposed tracking approaches that incrementally learn low-dimensional subspace representations and efficiently adapt the appearance of the target online. The system updates the model based on the sample mean and an error factor, which means it can expend less modelling power. Both features improve overall tracking performance. Generative trackers only model the appearance of the target object, so tracking can be fail on complex, cluttered backgrounds. To tackle this issue, more recent trackers also model the environments in which the targets are located. Two approaches to environment modelling have been proposed. In the first, the environment is selected for supporting the target motion which is correlated in the region of interest. G. Zhu et al. [86] and Y. Liu et al. [87] proposed this method, in which online supporter learning is takes place in order to determine the most likely predicted position of the targets in the scene. These supporters can help the tracking process when the targets disappear from the image or undergo complex transformations. The second approach is to consider the environments as negative class, as opposed to positive class, which discriminates the target from the background. The most common approach is to build a binary classifier, which represents the decision boundary between the object and the background that can discriminate the different trackers during the process. Y. Wu [88] proposed a static discriminative tracker that trains the object classifier before tracking takes place. This method has the limitation that the applications must know an object’s class, which means the tracker cannot track unknown objects. Thus, an adaptive discriminative tracker has been proposed. Q. Liu et al. [89], J. Xiao et al. [90], A. Karami et al. [91] and S. Li et al. [92] all proposed adaptive discriminative trackers in their tracking systems, in which the classifier is built during tracking. The most essential characteristic of adaptive discriminative trackers is the processing of updates. The tracker extracts the nearest neighbourhood features of the current target location and uses them as positive training samples, while the distant and surrounding features of the target are used as negative training samples. This process is applied in every frame to update the classifier. It has been proven that this updating process can handle unpredicted appearance changes in the target, short-term occlusions and complex backgrounds. However, this method can also cause errors when the object disappears from the image for longer than expected. Sh. Pang et al. [93] proposed an auxiliary classifier, which is trained in the first frame and can tackle the disappearance problem. Z. Wang et al. [94] and Q. Yu et al. [95] also proposed that training two independent classifiers for trackers can solve the problem.

In conclusion, the tracking systems can be summarised as follows:

Monocular Vision	Stereo Vision
Optical flow	Kalman filtering
Geometric constraints	Extended Kalman filtering
Template matching	Particle filtering
Feature-based tracking	
Particle filtering	

2.3 Learning

As discussed above, classification is an important component in pattern recognition. Classification can be either supervised or unsupervised. Object features are learned under the assumption that all the training samples are labelled before detection or tracking take place. I. Misra [96] and Ororbia II et al [97] proposed a semi-supervised learning approach that can be used under the assumption that the labels of entire training samples are too strong in certain training cases, and the detector is only required to train from a single labelled sample. A number of learning algorithms based on similar assumptions have been proposed, which include Expectation-Maximisation (EM), self-learning and co-training.

Expectation-Maximisation (EM) is a generic method for predicting model parameters by using unlabelled data. EM is an iterative binary classification process which alternates estimates over unlabelled data and trains a classifier. K. Nigam et al [98] and J. Garriga et al [99] used EM for object classification and object category learning, and their systems show reasonable performance. In the semi-supervised learning method, the EM algorithm is based on the low-density separation assumption proposed by O. Chapelle [100] which means that the separation of different classes can be performed well.

Self-learning is a process that trains an initial classifier from a set of labelled training samples. Then the classification model is evaluated using a set of unlabelled samples. The most confident data as decided by the classifier model will be added to the training set for the retention of the classifier, which means this is an iterative process. J. Kuen et al [101] used self-learning in human eye detection. However, their results showed that detection improved more if the unlabelled data was selected by independent measurement rather than selected by the classifier. Their conclusion suggests that the low-density separation method is not suitable for object detection.

Co-training is a learning method which uses independent classifiers that can mutually train each other. Co-training assumes that independent feature space is available when creating these independent classifiers. Learning starts with the training of two separate classifiers using labelled training data, and both classifiers are then evaluated using unlabelled training samples. The two classifiers then iteratively augment each other's training sets. Co-training can achieve a better performance for problems with independent modalities, such as text classification or biometric recognition systems, which include appearance and finger print recognition. F. Li et al. [102] and B. Zhong et al. [103] applied co-training to vehicle detection by surveillance cameras and moving-object recognition methods. However, co-training is limited for object detection or tracking because the samples are extracted from a single environment, which means the results can be dependent and rely on the training samples.

2.4 Conclusions and Inspirations

Many implicit modelling approaches have been proposed for the tasks of automatic vehicle detection and tracking. The general approach is to decompose the problem into two stages; a detection stage followed by a filtering and verification stage. In most cases, detection is achieved using a sliding window strategy, in which an exhaustive search is performed across position and scale. To accelerate detection, some approaches make use of integral images and histograms for the real-time computation of features. Features that encode shape information are the most commonly used for vehicle representation and appear to perform well. The classification approaches used vary; however, most choose to create discriminative learning models as opposed to probabilistic models, with the most prominent classifier being the Support Vector Machine.

The inspiration for this thesis comes from vehicle detection and tracking methods. Previous literature has used various methods to approach vehicle detection. Firstly, the detection process adopts the vehicle descriptor feature from colour and texture. A combination of GLCM and HSV is one of the standard approaches to detection, but this method is limited by blur problems and the improvement upon this approach was proposed.

Further inspirations for this thesis come from various descriptor feature approaches, which use both area descriptors and point descriptors. The Implicit Shape Model (ISM) has been used mostly in pedestrian detection rather than vehicle detection. The approach integrates point descriptors and area descriptors together to increase the detail of the target features, which could make a better classification model for detection. The SIFT point descriptor has been used in object matching. The segmentation abilities of ISM with the scale-invariance of the SIFT feature were used to create a detection approach.

HoG features are also widely used in object detection, and several improved HoG features have been proposed. Most HoG features are used with a sliding window approach, which is very inefficient. This is counterproductive to the aim of real-time processing. However, the detection could use edges and corners which are the most characteristic features of vehicles. There are various approaches to finding edges and corners; the FAST corner approach is most appropriate method for us to use because of its fast processing performance. An integrated FAST and HoG feature was used to make a novel vehicle detection process.

Furthermore, the Tracking Learning Detection (TLD) algorithm has been a source of motivation, and the learning approach in this algorithm was used in the proposed tracking system, which built a self-learning tracking process with the proposed detection system to make a system which can detect and track multiple vehicles, and beyond detection and tracking the system can learn the detected vehicle by itself to improve the classification model for later tracking and detection. This method is aimed for a long-term self-learning vehicle tracking and detection approach.

Chapter 3

HSV-GLCM Feature with Blur-removal

In this chapter, a vehicle detection method is proposed that uses HSV-GLCM (Hue, Saturation, Value and Grey Level Co-occurrence Matrix). The proposed feature combines the GLCM texture recognition method with the HSV colour system to detect vehicles and avoid the false detection of non-vehicles; this method also integrated blur-removal into the detection feature which solved the problem of blurred detection errors.

3.1 Basic Concept of GLCM

Definition: The GLCM (Grey Level Co-occurrence Matrix) is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image. In summary, there are five necessary steps in generating a GLCM matrix [104]:

- Create a framework matrix
- Decide on the spatial relation between the reference and neighbour pixel
- Count the occurrences and fill in the framework matrix
- Add the matrix to its transpose to make it symmetrical
- Normalise the matrix to turn it into probabilities.

An example will be introduced to explain the definition and how to generate a GLCM matrix in detail. Here is a test image:

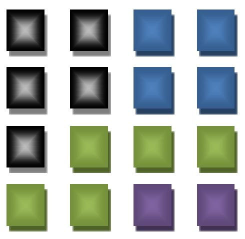


Figure 3.1: A GLCM testing image.

As the Figure 3.1 shows, the image has four different values of grey. Each pixel value in the image is shown in Table 3.1.

The value form is as shown below:

	X-axis			
Y-axis	**0**	0	1	1
	0	0	1	1
	0	2	2	2
	2	2	3	3

Table 3.1: Values of the GLCM testing image. (the top-left corner is the origin in [0,0])

The GLCM is a "second order" texture calculation. It only considers the relationship between two pixels; one is the reference pixel and the other is its neighbour (the distance may not be equal to 1). First order and higher order calculations are easy to understand. First order means the data that used to analyse the texture feather is directly from the original image, such as the variance, which does not consider the pixel-neighbour relationships in the image. The higher order is more complex than GLCM, because it needs to take account of the relationship of three or more pixels.

Before discussing how the GLCM matrix can be generated, the measurements used to extract values from the original image to build the matrix should be chose. First, “the spatial relationship between two pixels”; in brief, this just means the orientation between a reference pixel and a neighbour pixel. In a GLCM matrix, values are extracted from four different orientations: 0°, 45°, 90° and 135°. For example, in the given image, pixel (0, 1) and pixel (0, 2) have the relationship, and the distance is 1. This type of relationship is called as “(1, 0) relation”, because it consists of one pixel in the x direction and 0 pixels in the y direction. It may found that this combination happens again between pixel (1, 1) and pixel (1, 2). Therefore, the frequency of this combination is 2. Moreover, this kind of combination is based on a 0°orientation.

In fact, the given image window may be a small part of the original picture, and the reason for using this method to analyse images is that it reduces the amount of necessary calculation. Each pixel within the window becomes a reference pixel in turn, starting in the upper left-corner and proceeding to the lower-right. Pixels along the right edge have no right hand neighbour, so they are not used for this count.

Secondly, there is “separation between two pixels” meaning the distance between a reference pixel and a neighbour pixel. In the same example, pixel (0, 1) and pixel (0, 2) have a relationship, and it is called “1 pixel offset”, which refers to the reference pixel and its immediate neighbour. Actually, if the window is large enough, using a larger offset is possible. This chapter will discuss the value of offset that should be chosen, and also use statistical data to prove it. As mentioned above the offset distance may not equal to 1, this is because if the only the neighbour pixels are considered, the system will require a large amount of calculations. Thus, by using the larger offset distance, the system can reduce the numbers of pixels that needs to calculate, which can increase the speed of the process.

The final step before building the GLCM matrix is based on the grey level to get a combination form. In the given image window, only four grey levels are available: 0, 1, 2 and 4. So 4 × 4 data cells can be obtained:

Reference pixel value	Neighbour pixel value			
	0	**1**	**2**	**3**
0	**0,0**	**0,1**	**0,2**	**0,3**
1	**1,0**	**1,1**	**1,2**	**1,3**
2	**2,0**	**2,1**	**2,2**	**2,3**
3	**3,0**	**3,1**	**3,2**	**3,3**

Table 3.2: GLCM combination form.

It is assumed that the orientation is east (left to right), and the distance between two pixels is 1. Then the GLCM matrix is obtained as:

2	**2**	**1**	**0**
0	2	0	0
0	0	3	1
0	0	0	1

Table 3.3: GLCM matrix for the testing image.

The first value is 2, which means the combination of the reference pixel being 0 and its eastern neighbour also being 0 occurs twice. A value of 3 means the reference pixel is 2 and its neighbour is also 2, etc., three times.

Texture calculations require a symmetrical matrix. The next step is therefore to change the GLCM into this form. It should be noted that if the west orientation matrix was calculated, a matrix would be as shown below:

2	0	0	0
2	2	0	0
1	0	3	0
0	0	1	1

Table 3.4: GLCM matrix of the testing image's west orientation.

The form shows that east matrix and the west matrix have a diagonal relationship. East (i, j) = West (j, i). To reflect this attribute, a new matrix was used, which called a symmetrical matrix. The matrix is shown below:

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Table 3.5: Symmetrical GLCM matrix of the testing image.

As the figure shows, symmetry will be achieved if each pixel pair is counted twice: once "forwards" and once "backwards" (interchanging the reference and neighbour pixels for the second count).

After making the GLCM symmetrical, there is still one step to take before texture measures can be calculated. The measures require that each GLCM cell contain not a count, but rather a probability. Usually, this operation is referred to as normalization. The following equation can be used to realise normalisation of the symmetrical matrix:

$$\mathrm{P}(i,j) = \frac{\mathrm{Vi,j}}{\sum_{\mathrm{i,j=0}}^{\mathrm{N-1}} \mathrm{Vi,j}} \qquad (3.1)\ [104]$$

where 'i' is the row number and j is the column number. After calculation, the normalised matrix is shown as:

4/24	2/24	1/24	0
2/24	4/24	0	0
1/24	0	6/24	1/24
0	0	1/24	2/24

Table 3.6: Normalised GLCM matrix of the testing image.

After the normalisation matrix has been obtained, this matrix can be used to calculate the texture measures. In brief, most texture calculations are weighted averages of the normalised GLCM cell contents. For example, CON = GLCM × W, where $\mathrm{W} = (i-j)^2$.

The texture measures can be classified into three groups:

Contrast group: Measures related to contrast use weights related to the distance from the GLCM diagonal. For example, Contrast, Dissimilarity and Homogeneity.

Orderliness group: Measures related to orderliness show how regular (orderly) the pixel values are within the window. For example, as ASM, MAX Probability and Entropy.

Group using descriptive statistics of GLCM texture measures: these measures consist of statistics derived from the GLC matrix. For example, GLCM Mean, GLCM Variance, and GLCM Correlation.

A note on the contrast group: As per the definition, the results of this group's measures are mainly affected by the value is separated from the diagonal. The Contrast (CON), and Homogeneity (HOM) will be discussed in the following sections and the reasons of not using other features to measure the images in this method will also be discussed later.

3.1.1 Contrast (CON)

Contrast (CON) is also called "sum of squares variance". It can be expressed as:

$$\sum_{\mathrm{i,j}=0}^{\mathrm{N}-1} \mathrm{P(i,j)} \times (i-j)^2 \tag{3.2}$$ [104]

The equation $\sum_{\mathrm{i,j}=0}^{\mathrm{N}-1} \mathrm{P(i,j)}$ expresses all the elements of the normalised symmetrical matrix of GLCM. The equation $(i-j)^2$ is the value of the weight, element closer to the diagonal in value will be nearer to 0. On the contrary, the value will increase. As learned, an element around the diagonal indicates that the value of the reference pixel is similar to its neighbour. Therefore, a small value of CON means the differences of values between the image's pixels are small.

In summary, contrast also reflects the sharpness and depth of texture furrows. The deeper texture furrows will be easier to recognise and have greater CON values. Conversely, shallow-textured furrows will result in lower CON values.

3.1.2 Homogeneity (HOM)

Homogeneity is also called the “inverse difference moment”. It can be expressed as:

$$\sum_{i,j=0}^{N-1} P(i,j) \times \frac{1}{1+(i-j)^2} \qquad (3.3)\ [104]$$

Unlike CON, as the elements get further away from the diagonal, their weight will decrease. That is to say, the value of CON indicates the difference in values in the image. If all the pixels are similar to one certain value, the CON value will be close to 1.

In summary, the inverse difference moment reflects the homogeneity of the image and measures local changes in the image. A high value of HOM means the texture between regions does not vary.

There are also measures related to orderliness, i.e. how regular (orderly) the pixel values are within the window. These include ASM, MAX probability and entropy. This group is mainly discussed ASM and entropy (ENT), which are the key measures in texture feature analysis. However, there is redundancy between these two measures, so the trade-offs between them in this approach will also be discussed.

The following example of two images will demonstrate what orderliness is and the difference between that and contrast group measures.

Image 1				Image 2			
1	2	3	4	3	4	3	2
1	2	3	4	1	2	3	4
1	2	3	4	2	3	4	5
1	2	3	4	4	5	6	7

Table 3.7: Two testing images features.

Assume that only 0° (horizontal orientation) is being taken into consideration, and the offset is 1. From the perspective of the contrast feature the two images are the same, because the difference between each pair of reference pixel and neighbour pixel is 1. So depend on CON or HOM alone to distinguish between these two images is impossible. Orderliness measures also use a weighted average of the GLCM values, but the difference is that this weight is affected by the how many times a given pair occurs:

- A weight that increases with commonness will yield a texture measure that increases with orderliness.
- A weight that decreases with commonness yields a texture measure that increases with disorder

In comparison, although image 1 has few combinations, the number of times each combination occurs is more than that of image 2. For instance, pair (1, 2) occurs four times in image 1 but only once in image 2. So the value of the orderliness will be different, which this attribute can be used to distinguish between these two images.

3.1.3 Angular Second Moment (ASM) and Energy

Angular Second Moment (ASM) and energy - also called uniformity

ASM and Energy use each P_{ij} as a weight for itself. High values of ASM or Energy occur when the window is very orderly. This is expressed as:

$$\sum_{\mathrm{i,j=0}}^{\mathrm{N-1}} {P_{ij}}^2 \qquad (3.4)\ [104]$$

The square root of the ASM is sometimes used as a texture measure, and is then called energy. Energy reflects the uniformity and thickness of the texture. As the equation above shows, if each element in the GLCM has the same value, the outcome will be a small ASM value. In contrast, if some elements have greater values than others, a large ASM value will be calculated. If the elements of GLCM are concentrated within the matrix, the ASM value will be large. In summary, a large ASM indicates a more uniform and regular change in texture pattern.

3.1.4 Entropy (ENT)

This is expressed as:

$$\sum_{i,j=0}^{N-1} P_{i,j}\left(-\ln P_{i,j}\right) \qquad (3.5)\ [104]$$

Entropy means the amount of information of the image contains. Texture also belongs to image information, and it is a random measure. The maximum value of ENT is 1, and this maximum is reached when the elements of GLCM have maximum randomness, scattered distribution and all probabilities are equal. In other word, ENT indicates the non-uniformity or complexity of the image.

The other group uses descriptive statistics of GLCM texture measures; these measures consist of statistics derived from the GLC matrix. Members of this group include GLCM mean, GLCM variance, and GLCM. Compared to the contrast and orderliness groups, this group is more considerate of the GLCM matrix itself. The pixel value is weighted not by the frequency of its occurrence by itself (as in a "regular" or familiar mean equation) but by the frequency of its occurrence in combination with a certain neighbour pixel value.

The GLCM correlation texture measures the linear dependency of grey levels on those of neighbouring pixels. The equation shown below:

$$\sum_{i,j=0}^{N-1} P_{i,j}\left[\frac{(i-\mu_i)\left((i-\mu_i)\right)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right] \qquad (3.6)\ [104]$$

where

$$\mu_i = \sum_{i,j=0}^{N-1} i\left(P_{i,j}\right), \mu_j = \sum_{i,j=0}^{N-1} j\left(P_{i,j}\right), \sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j}(i-\mu_i)^2, \sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j}\left(j-\mu_j\right)^2 \qquad (3.7)\ [104]$$

GLCM correlation is quite a different calculation from the other texture measures described above. As a result, it is independent of them (i.e. it gives different information) and can often be used profitably in combination with another texture measure. It also has a more intuitive relationship with the actual calculated values: 0 is uncorrelated, 1 is perfectly correlated. However, if an image has only one grey value, the variance of the GLCM will be 0. Moreover, the result of the correlation will be NaN (not a number), because the result is a certain number divided by 0.

The equation also indicates that correlation measure the similarity of row or column elements in GLCM matrix, which means, the correlation value reflects the correlation of certain parts of the image. If the matrix element values are equal, the correlation value is large. Conversely, if the values of the elements differ, the correlation value will be small. Furthermore, if the image mainly has a horizontal texture, then the COR value in this direction will be larger than the other orientation’s COR value.

3.2 GLCM Parameter Decisions

Once the basic conception of GLCM and its related parameters were explained, the GLCM descriptors can be used in artificial object detection. However, the texture measurements need to be set first, which are:

- Window size
- Number of grey levels
- Direction of offset
- Offset distance
- Which channel to run
- Which measure to use

3.2.1 Window Size Decision

The test images are selected from the training video, which are listed in the Appendix. The video was captured by UAVs on a motorway near an urban area. The size of the images is in 720 × 576 pixels. The Figure 3.2 shows an example of an image from the video.

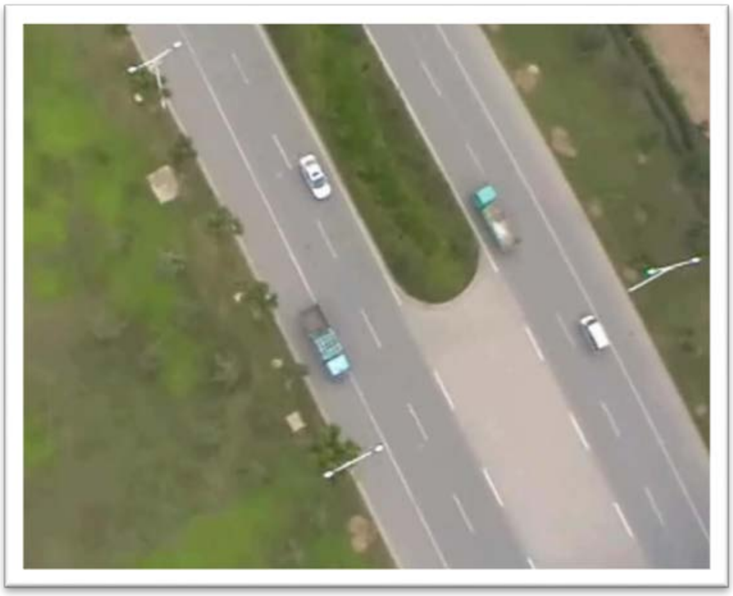


Figure 3.2: Sample image (720 × 576 pixels).

The detection window should be a square and include all the texture detail of the vehicles, a size of 70 ×70 pixel window was set in the proposed detection approach. This window size value can be changed based on the size of the vehicles appeared in the images as the different altitude of the UAVs. The figure below shows an example of window samples.

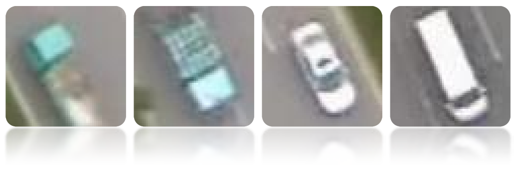


Figure 3.3: Target window samples.

3.2.2 Number of grey levels

As is well known, an 8-bits grey image has 256 grey levels, so there will be 256 x 256 (65,536) combinations in the GLCM matrix; analysing it will require huge computing power and waste lots of time. Therefore, to save time and computing power, this could reduce the number grey levels. Usually, it has the choice of 16, 32, or 64. The greater the value, the better the effect, but also the longer the time required. In this approach, 32 grey levels was chose. The reason for this choice is that it is easy to get a “NaN” value of COR values by using a small grey level. This is because the denominator of the COR equation will be 0 in a uniform image. Such phenomenon more often occurs in small windows and at low grey levels. So in the proposed approach, taking into account both the operation time of the programme and the error occurrence rate, thus 32 grey levels has been used.

3.2.3 Direction of Offset

The GLCM matrix is calculated from four different orientations: 0°, 45°, 90°, and 135°. This is because GLCM is not direction invariant. Vehicles are coming from different directions in the video, so four main directions were defined to detect vehicles, which can solve this invariant problem.

3.2.4 Offset Distance

Usually, a value of 1 is chosen for the distance between the two pixels. F. Zhou et al. [105] proposed that idea that there are relationships between distance and the calculated values (CON, COR, HOM, etc). In the perspective of the authors of this paper, using a Markov Random Field (MRF) could prove that a calculation is correct only when the distance is greater than the value of a GLCM feature. Conversely, when the distance is small, the results of GLCM calculations are random or change. A. Chaddad et al. [106] also proposed the same idea; in their opinion, when the distance is small, or the two chosen pixels are close together, the result of GLCM calculations rapidly change with any increase of distance. But when the distance becomes large, the result will be more stable. The conclusions of these two papers are same. As a result, it is essential to find a suitable value for the distance.

As it is well known, the most used features in GLCM texture analysis are contrast (CON), correlation (COR), homogeneity (HOM), energy (ASM) and entropy (ENT). In this proposed detection method the GLCM has therefore been divided into five groups, each group focusing on one specific feature. The unit of x-axis is the distance, and the unit of y-axis is the value of one of the features. Moreover, in each diagram, four different curves were found which indicates the values of four vehicles in the sample image (Figure 3.4).

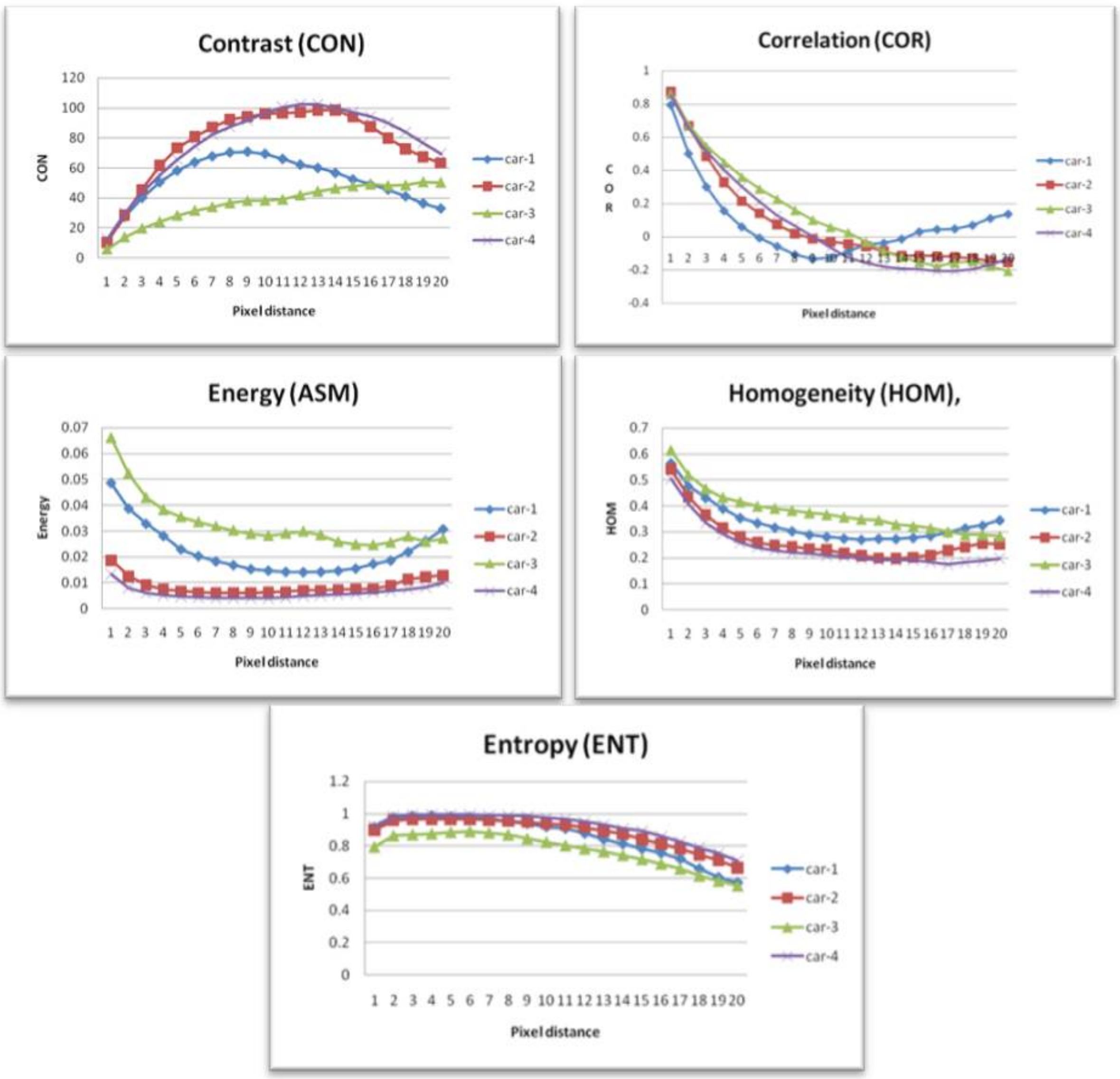


Figure 3.4: Graphs of GLCM texture values for four vehicles.

The simulation's results confirmed the conclusions of F. Zhou et al. and A. Chaddad et al. [105, 106]. The GLCM calculation results become stable past a certain value. It can also be noted that changes of CON, COR, ASM and HOM are affected by pixel distance, but ENT is less affected. With increases in distance, the values of ASM, COR, and HOM decrease. Furthermore, the decrease slows down when the distance has reached 2. When the distance is between 3 and 5 the change is significantly smaller. On the other hand, with an increase in pixel distance, CON also increases, but the increase rate is reduced when reaches 2; once the distance is larger than 3, the value of CON is stable. In summary, when the distance is between 3 and 5, the curve becomes stable; especially after 10, when the curve becomes flat. Therefore, when the distance equals to 3, the texture features will be better for measuring the vehicles.

3.2.5 GLCM Channel

It has been suggested the GLCM features could be calculated on several different channels. For example, a coloured image may consist by three channels: R, G and B. So, the image can be measured by three CON values rather than only one CON value, which will increase the accuracy of the feature descriptor. However, more computational resources are required for all three channels, so a grey level image has been used as the input image. ($\text{The grey level image} = 0.212671\ \times \text{R} + 0.715160\ \times \text{G} + 0.072169\ \times \text{G}$). After the shift to grey level, the image looks like this:

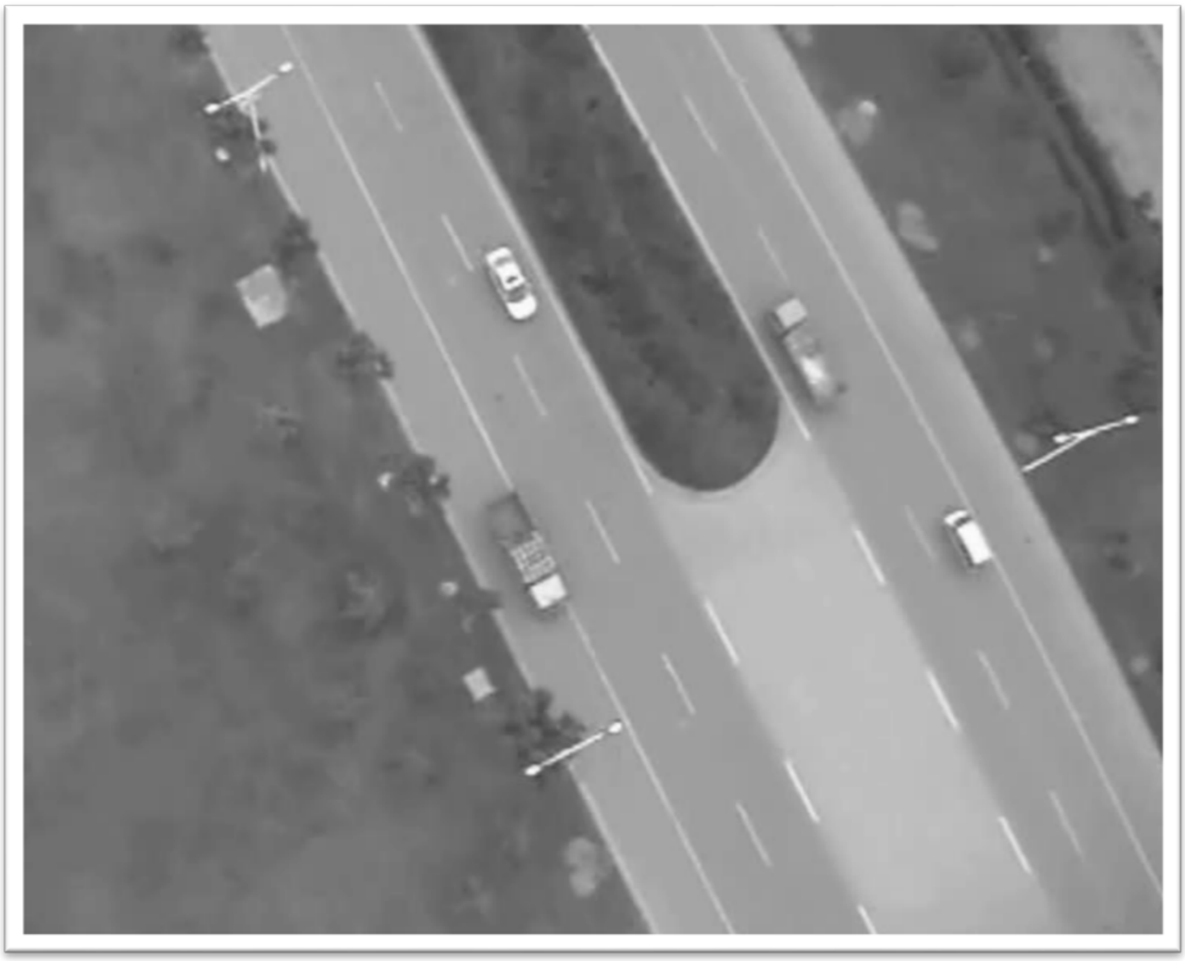


Figure 3.5: Grey level testing image.

3.2.6 GLCM Measurement

P.Mohanaiah et al. [107] suggest that GLCM has 14 features with which to measure an image, but researchers often only use six of them: CON, COR, ASM, HOM, ENT and DIS. X. Qu et al. [108] shows that there are lots of redundancies among the 14 features, and even amongst the six often-used features, there are still some redundancies.

In the discussion of the three GLCM measure group (the contrast groups, measures related to orderliness and the group that uses descriptive statistics of GLCM texture measures), it was noticed that all the groups shared some common features.

In the contrast group, the result of the measure is mainly affected by the position of the element in the GLCM matrix, or the difference between the two chosen pixels. In the second group, the result of the measure is mainly affected by the value of the element in the GLCM matrix, or the element itself. In the third group, the result is affected by the entire GLCM matrix.

B. Hua et al. [109] uses the same idea as this approach. After comparing the expressions of the six features, they state that ΔCON > ΔDIS > ΔHOM, ΔENT > ΔASM. Therefore, the authors suggested that CON, ENT and COR should be used as texture feature measure. Y. Bin et al. [110] provide a more detailed table which discloses the correlation of 11 texture features. These forms were summarised, and picked out the relationships of the five most commonly used features. The table is shown below:

	Asm	Con	Cor	Ent	Hom
Asm	**1.00**	- 0.69	0.32	**- 0.94**	**0.83**
Con	- 0.69	1.00	-0.59	**0.87**	**- 0.80**
Cor	0.32	- 0.59	1.00	- 0.37	0.75
Ent	**- 0.94**	**0.87**	- 0.37	1.00	**- 0.83**
Hom	**0.83**	**- 0.80**	0.75	**- 0.83**	1.00

Table 3.8: Correlation of GLCM features.

The table disclosed the correlation of the features, the absolute value closest to 1, the two most relevant, and the redundancy. Values that is larger than 0.8 have been marked in the table. To solve the redundancy problem, two features were removed from the table. One had to be ASM or ENT, the other CON or HOM. Because ΔENT > ΔASM, which kept ENT as a measure feature. However, though ΔCON > ΔHOM, the HOM was still chose as the second texture measure. This is because the value of HOM is between 0 and 1, but the value of CON is not. That is to say, the absolute values of COR, ENT and CON are all already normalised, because they are all in the interval of 0 to 1. For the purposes of study, it is very easy to allocate a weight to them. As a result, a new correlation form was generated as below.

	COR	ENT	HOM
COR	1.00	- 0.37	0.75
ENT	- 0.37	1.00	- 0.83
HOM	0.75	- 0.83	1.00

Table 3.9: The correlation of the three selected GLCM features.

3.2.7 Parameters

Parameter	Value
Window size	70 × 70 pixels
Number of grey levels	32
Direction of offset	four directions (0°, 45°, 90°, 135°)
Offset distance	three pixels
Channel	one channel (grey level)
Measurements	correlation (COR), entropy(ENT) and homogeneity (HOM)

Table 3.10: The parameter of selected GLCM features.

3.3 GLCM Feature Training

After these three texture features for GLCM detection were selected, the SVM classifier could be trained. First of all, a total of 850 vehicle images (Figure 3.6) from the training video were selected. For each sample, the mean and variance of correlation, entropy and homogeneity were calculated (Figure 3.7). These features were set as the positive samples represented by 1 in the classifier. Then 260 images of objects in the environment, such as grass, trees, road markings and buildings, were selected as the negative samples (Figure 3.8). The mean and variance of correlation, entropy and homogeneity (Figure 3.9) were also calculated. These features were set as the negative samples, represented by 0 in the classifier.

The classifier gave us a classification model (Figure 3.10) for making decisions on the identification of vehicles.

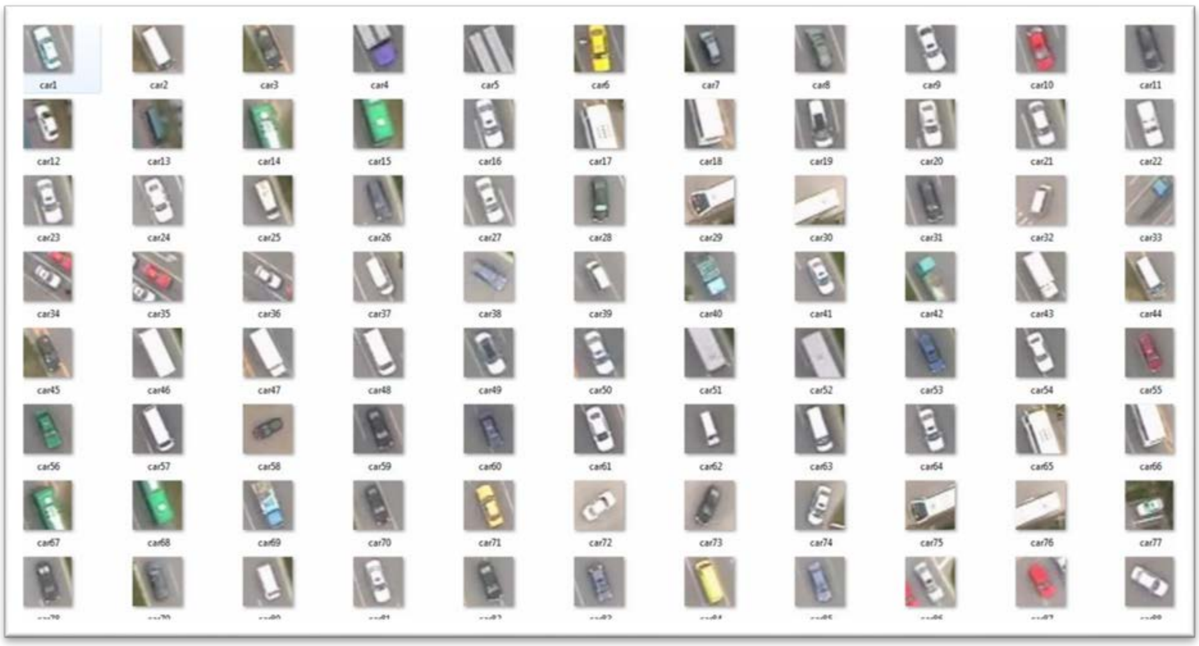
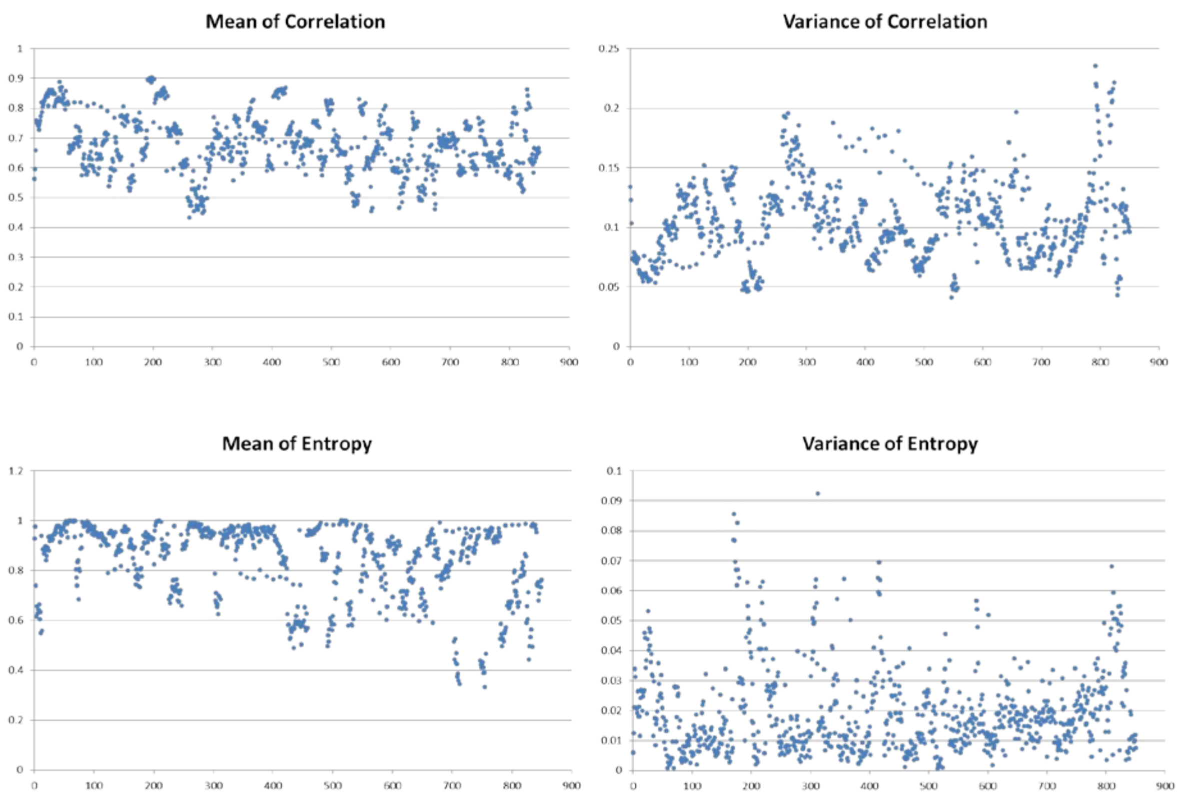


Figure 3.6: Examples of positive training samples.



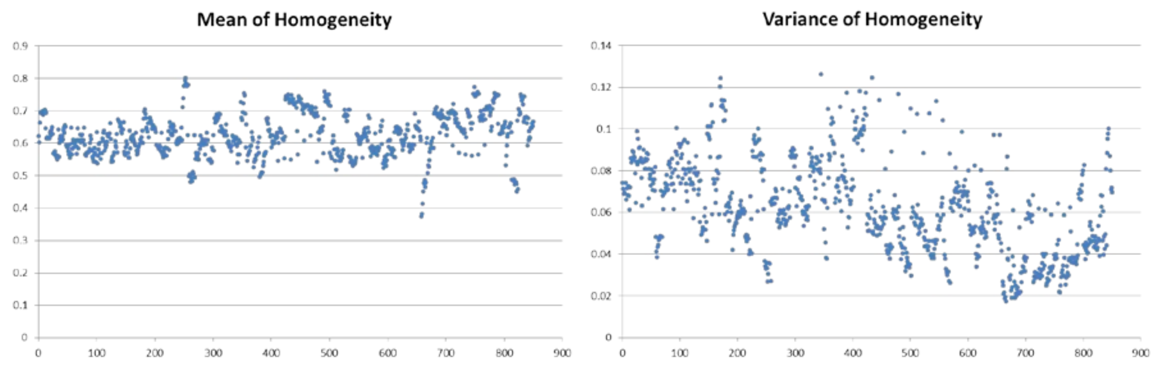


Figure 3.7: GLCM features for positive samples.

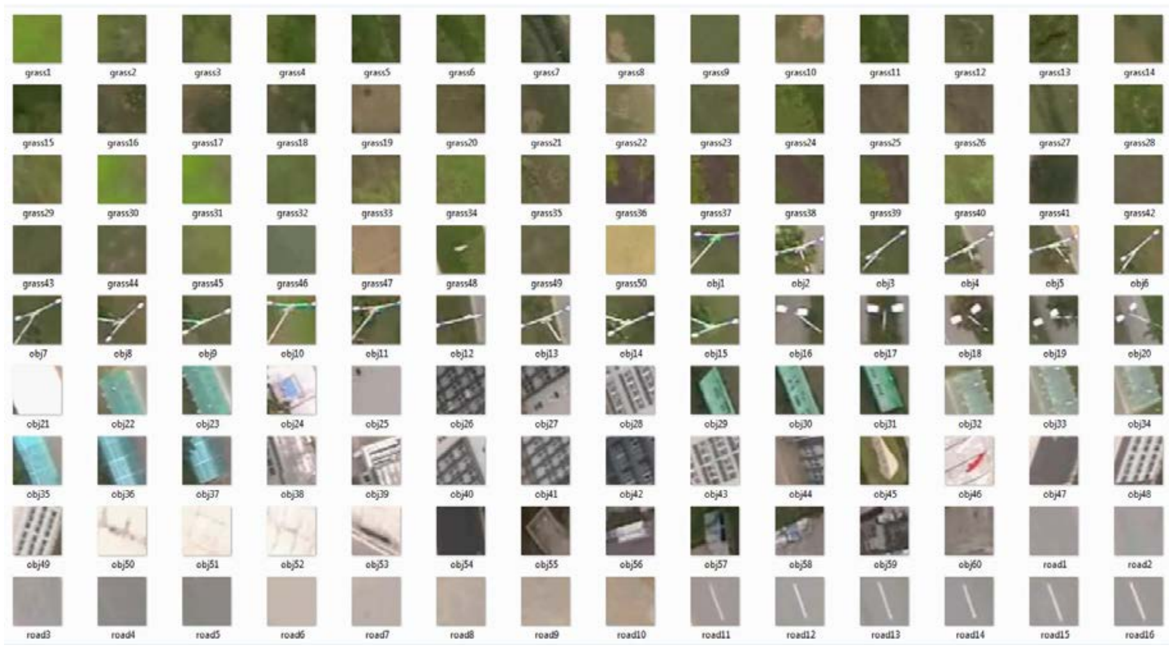
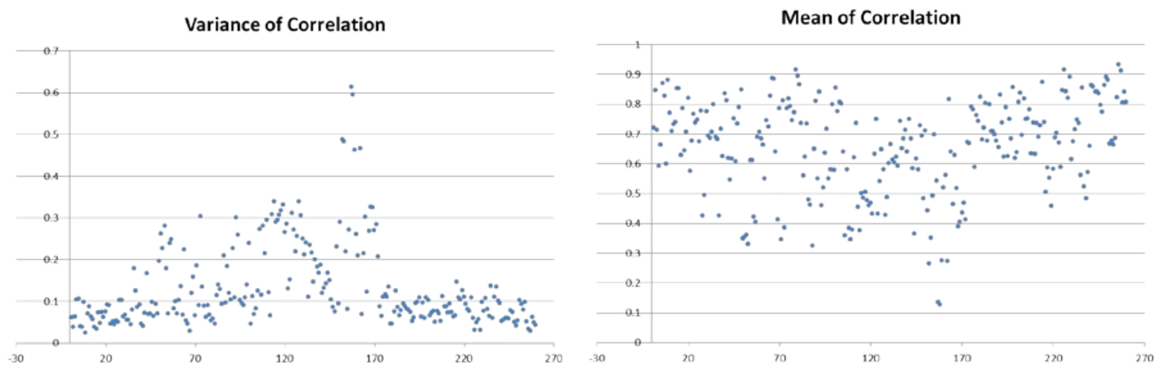

grass1 grass2 grass3 grass4 grass5 grass6 grass7 grass8 grass9 grass10 grass11 grass12 grass13 grass14
grass15 grass16 grass17 grass18 grass19 grass20 grass21 grass22 grass23 grass24 grass25 grass26 grass27 grass28
grass29 grass30 grass31 grass32 grass33 grass34 grass35 grass36 grass37 grass38 grass39 grass40 grass41 grass42
grass43 grass44 grass45 grass46 grass47 grass48 grass49 grass50 obj1 obj2 obj3 obj4 obj5 obj6
obj7 obj8 obj9 obj10 obj11 obj12 obj13 obj14 obj15 obj16 obj17 obj18 obj19 obj20
obj21 obj22 obj23 obj24 obj25 obj26 obj27 obj28 obj29 obj30 obj31 obj32 obj33 obj34
obj35 obj36 obj37 obj38 obj39 obj40 obj41 obj42 obj43 obj44 obj45 obj46 obj47 obj48
obj49 obj50 obj51 obj52 obj53 obj54 obj55 obj56 obj57 obj58 obj59 obj60 road1 road2
road3 road4 road5 road6 road7 road8 road9 road10 road11 road12 road13 road14 road15 road16

Figure 3.8: Examples of negative training samples.



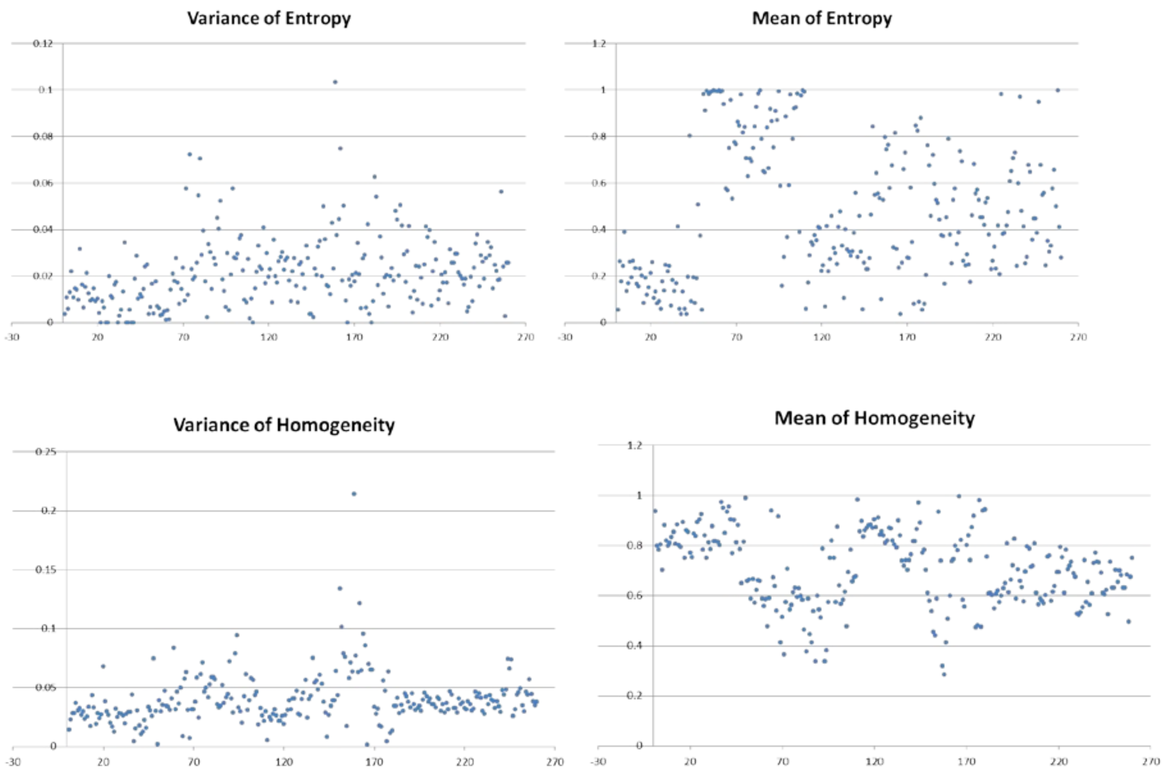


Figure 3.9: GLCM features for negative samples.

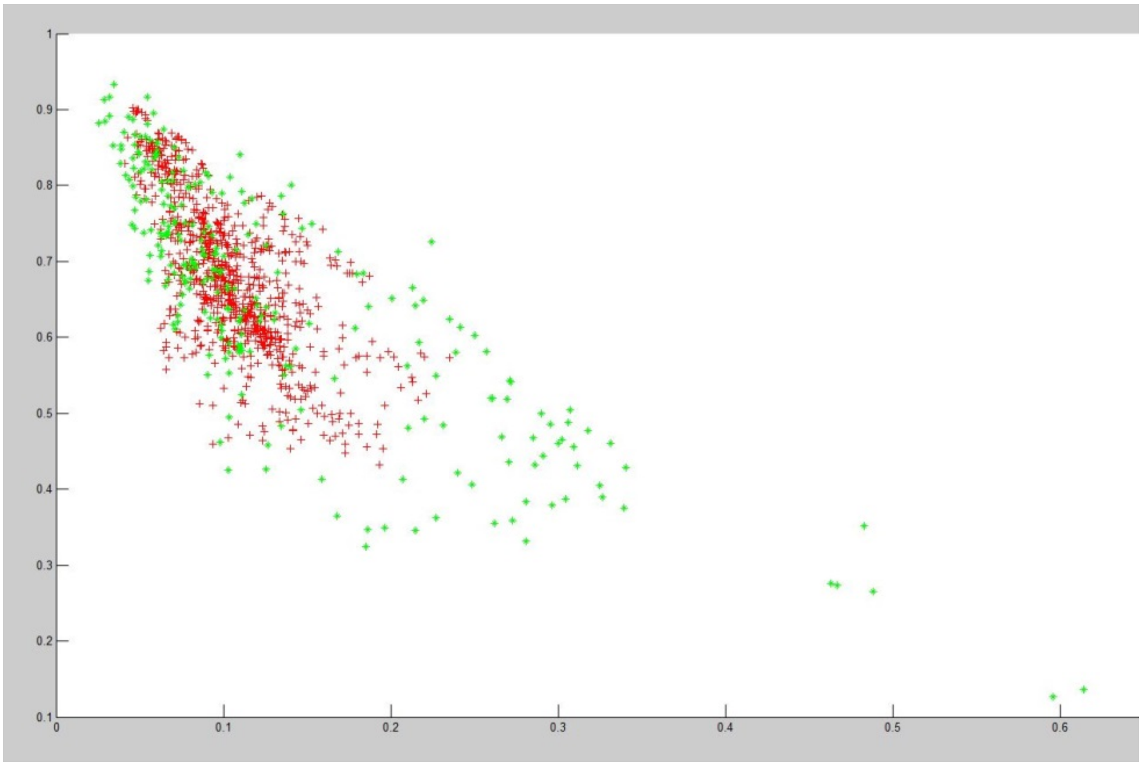


Figure 3.10: The SVM classification.

The diagram shows the distribution of the values of the texture measures. The diagram shows that the values of the six texture features are within certain ranges, especially that of entropy. The results are as follows:

Positive samples						
	stdCorp	mCorp	stdEntp	mEntp	stdHomp	mHomp
MIN	0.0409	0.4324	0.0005	0.3321	0.0174	0.3731
AVG	0.1049	0.6797	0.0192	0.8445	0.0625	0.6248
MAX	0.2352	0.9022	0.0924	0.9995	0.1262	0.8012
Negative samples						
	stdCorn	**mCorn**	**stdEntn**	**mEntn**	**stdHomn**	**mHomn**
MIN	0.0258	0.1270	0.0000	0.0369	0.0013	0.2850
AVG	0.1327	0.6569	0.0212	0.4759	0.0397	0.7065
MAX	0.6141	0.9329	0.1034	0.9990	0.2140	0.9950

Table 3.11: The range of GLCM features.

Table 3.11 shows the minimum, average and maximum values of the GLCM features for both positive and negative samples. Now knowing the range of each classifier and the training model, the approach was applied to the testing video 1. The frame images sizes are in 720 × 576 pixels and the window size was 70 × 70 pixel, which were divided in every 35 pixels. So in the test images, there were 17 × 22 (374) windows have been tested. For each test window, six GLCM texture features were calculated then those features were inserted into the SVM classification model. The SVM model makes the decisions that whether the test window was a vehicle or not by giving the prediction labels 1 or 0. Windows with the predict label 1 were considered as the vehicles, and will be marked by red rectangles (Figure 3.11).

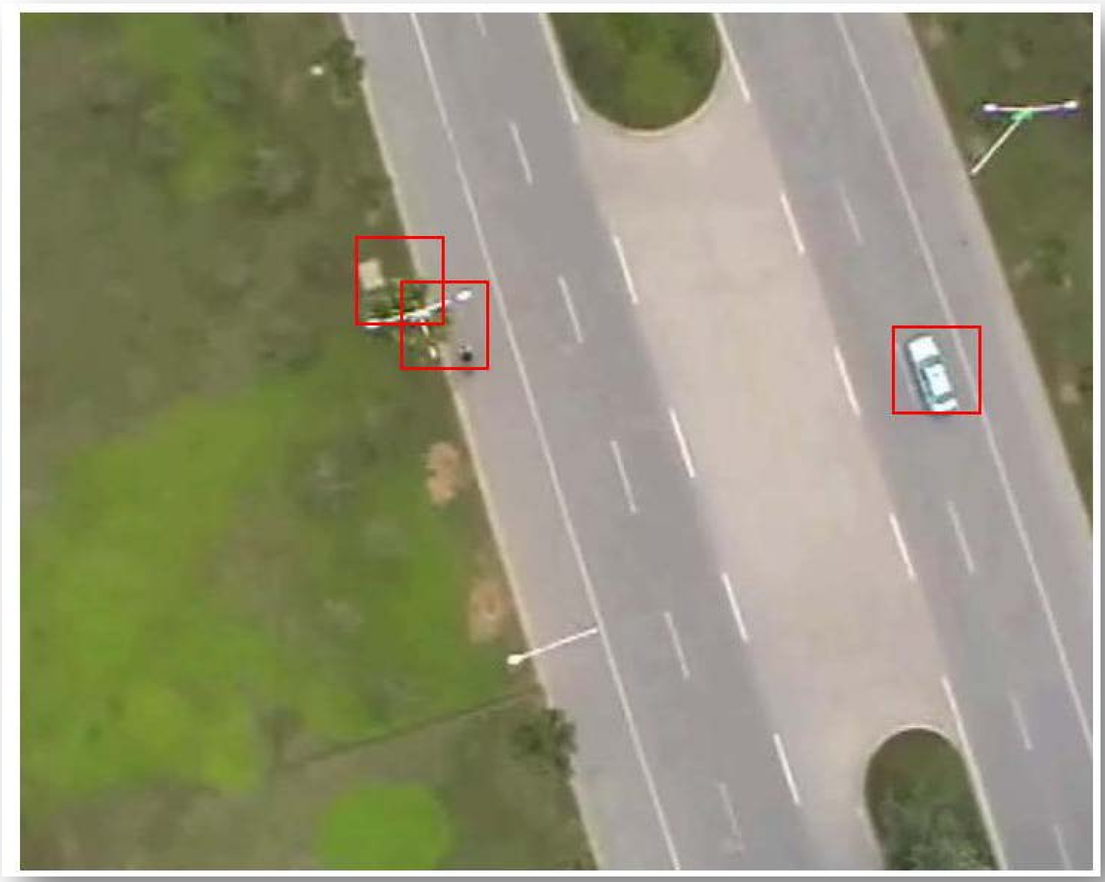


Figure 3.11: An example of the detection results by using the GLCM approach.

After the testing, a problem has occurred as Figure 3.11 shows, although the vehicle was marked correctly, the street light was also marked as vehicle, which is a false positive detection. Vehicles are typical artificial objects, but other common artificial objects that may appear in the environment include houses, bridges, road markings and street lights, which could have similar GLCM features. The differences between vehicles and the environments are their sizes, shapes and colours. 50 positive and negative samples from the training data were selected and the GLCM features were calculated for each sample.

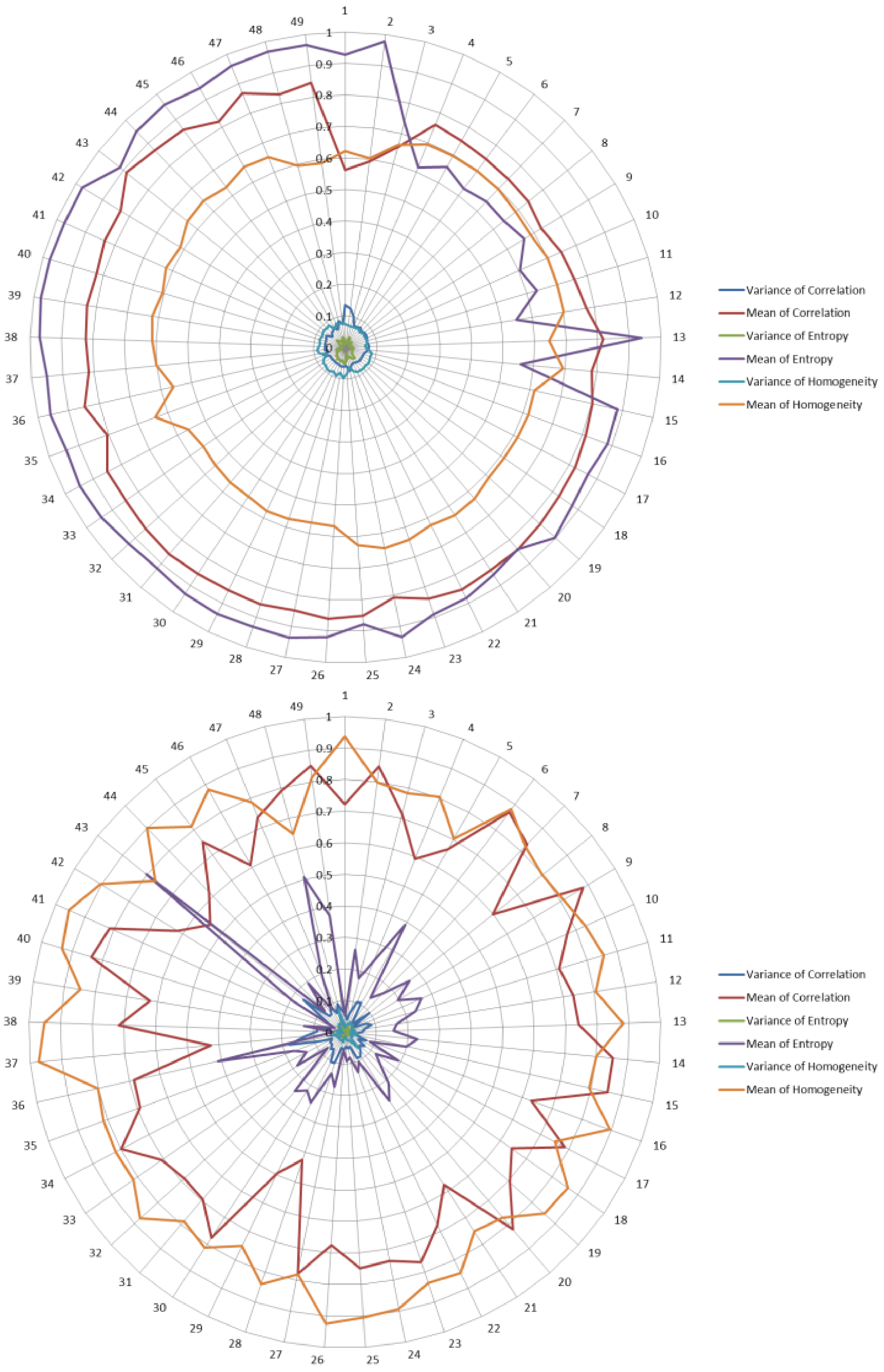


Figure 3.12: Vehicle GLCM features (top circle); environment GLCM features (bottom circle).

The diagram in Figure 3.12 show the feature distribution of the two kinds of artificial objects, vehicles (top circle) and other objects in the environment (bottom circle). It is noticed that the general feature distribution of vehicles and other objects is similar, but the mean, of the entropy (purple line) are different from each other. Thus, the entropy factor in the SVM training model was adjusted. The new entropy model:

$$\sum_{i,j=0}^{N-1} P_{i,j}\left(-\ln P_{i,j}\right) ENT \in [0.85, 1.00] \qquad (3.8)\ [104]$$

where $P_{i,j}$ is the weight, the N is the pixel number and the ENT is the entropy. After applying the new entropy model to the SVM classifier, the approach was applied into the same testing video again and one example of the detection results in shown in Figure 3.13 indicated that the vehicle is now marked by three markers rather than only one (Figure 3.11), this is because by using the new entropy model, more features around the vehicle can be detected. However, the road light is still marked as a vehicle. It is found that such error occurs at the junction of two textures, the road and grass, which has similar GLCM textures as vehicles. Texture detection has its advantages, such as being less affected by colour and light. However, only using texture to distinguish artificial objects from natural objects is far from enough. Having encountered the problem that some junctures will cause error windows, the colour feature was combined into the GLCM texture based approach.

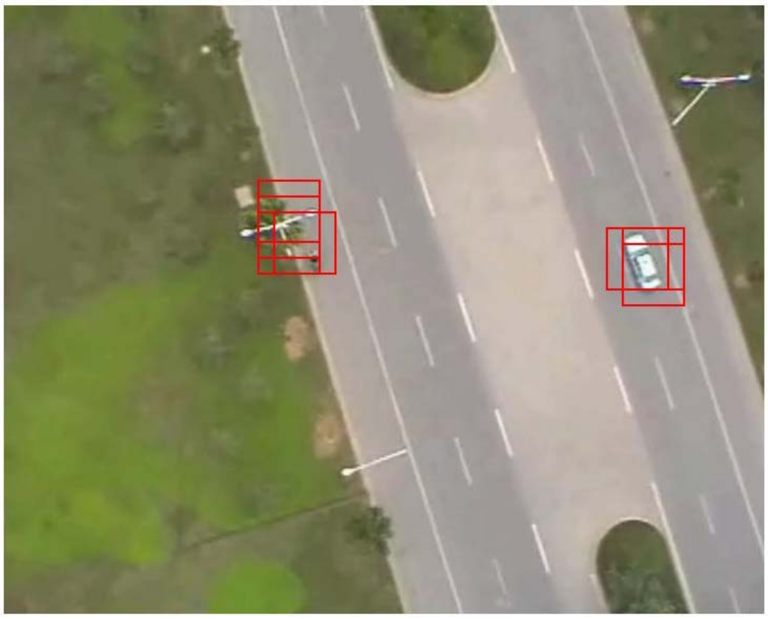


Figure 3.13: The new example of the detection result by using the new ENT model.

3.4 Basic Conception of HSV Space

HSV space [111] is a system used to describe colour. Like RGB space, it has three channels: hue, saturation, and value. The advantage of HSV space is this system is similar to the way humans recognise colour. For instance, the colours always describe as their base colour and its brightness. HSV colour space can be shown as below (Figure 3.14).

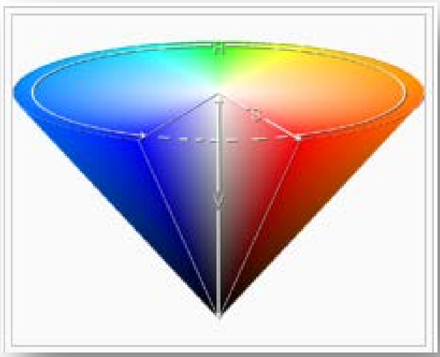


Figure 3.14: An example figure of HSV space.

As the figure shows, the bottom of the cylinder's central axis is black, the top is white and the middle is grey. The angle around the axis corresponds to hue, and the distance to this axis corresponds to saturation, moreover, the distance along this axis corresponds to value. The hue is the most essential parameter, because it decides the colour category, while other parameters just affect how bright the colour looks. For example, grass can be described as green-based, which is represented by the hue value. The saturation indicates what kind of green it is and the value indicates the brightness of the colour.

The Algorithm of HSV:

The R, G and B values are divided by 255 to change the range from 0 – 255 to 0 – 1:

$$R' = {}^{R}/_{255}\ ;\ G' = {}^{G}/_{255}\ ;\ B' = {}^{B}/_{255}$$

$$\begin{gathered} C_{max} = Max\ (R', G', B') \\ C_{min} = Min\ (R', G', B') \\ \Delta = (C_{max} - C_{min}) \end{gathered} \qquad (3.9)\ [111]$$

Hue calculation:

$$H = \begin{cases} 60° \times \left(\frac{G' - B'}{\Delta} mod6\right), C_{max} = R' \\ 60° \times \left(\frac{B' - R'}{\Delta} + 2\right), C_{max} = G' \\ 60° \times \left(\frac{R' - G'}{\Delta} + 4\right), C_{max} = B' \end{cases} \tag{3.10}$$ [111]

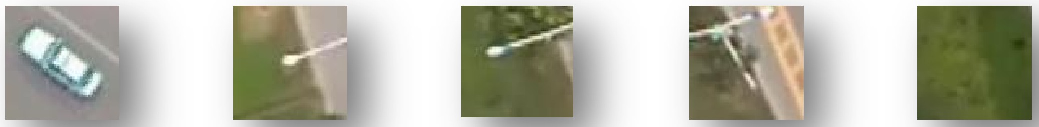
Saturation calculation:

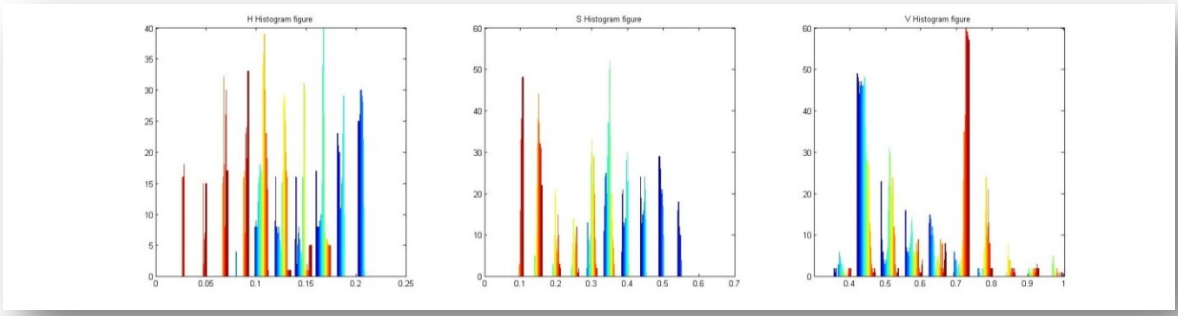
$$S = \begin{cases} 0 \ , \quad C_{max} = 0 \\ \frac{\Delta}{C_{max}}, C_{max} \neq 0 \end{cases} \tag{3.11}$$ [111]

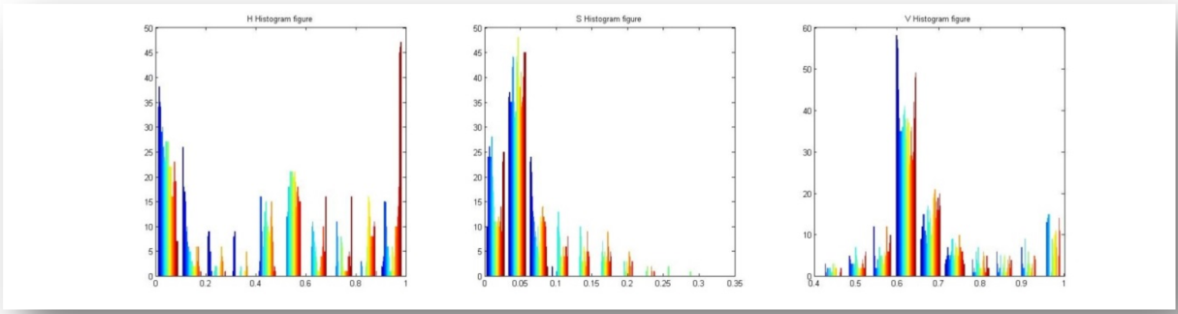
Value calculation:

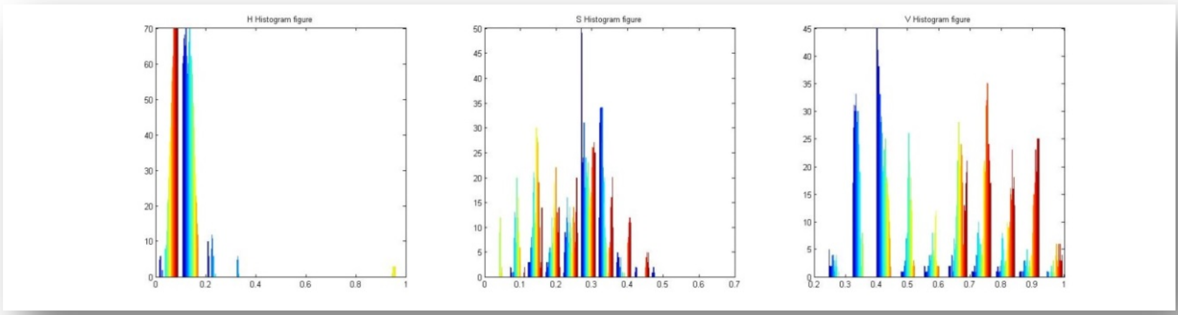
$$V = C_{max} \tag{3.12}$$ [111]

The figure below (Figure 3.15) shows examples of the HSV colour feature of environment samples and one vehicle sample. The colour is shown in histograms of hue, saturation and value. In order to show the difference of the HSV colour feature between the vehicles and environment objects, one vehicle sample and four different environment samples were selected to compare the patterns of the HSV colour feature. Firstly, the hue histogram figures indicate that different base colour histograms have different distributions. Furthermore, when the colour distribution of the sample is similar, the histogram has a conspicuous peak value. Conversely, the histogram has a dispersed distribution. This pattern also applies to the “saturation” and “value” feature. As a result, this method can use the mean and variance for each feature of HSV to classify the environment.




H Histogram figure
S Histogram figure
V Histogram figure


H Histogram figure
S Histogram figure
V Histogram figure


H Histogram figure
S Histogram figure
V Histogram figure

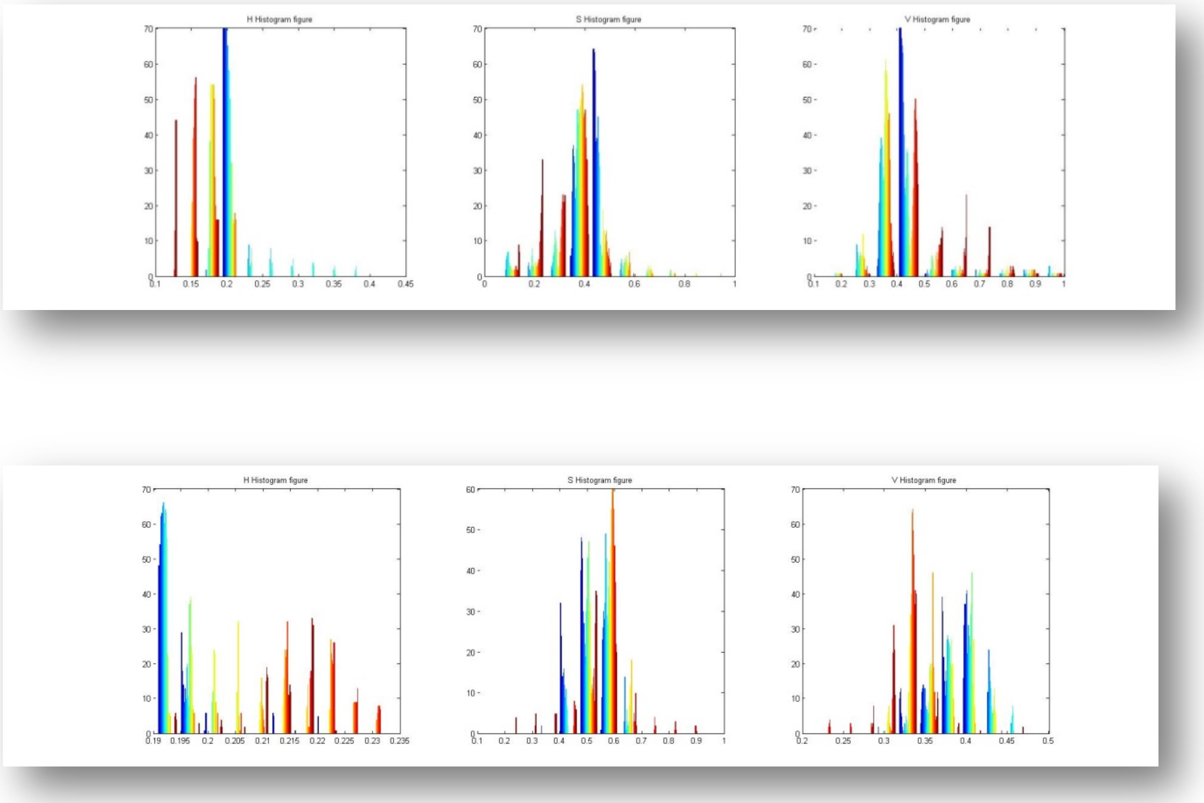
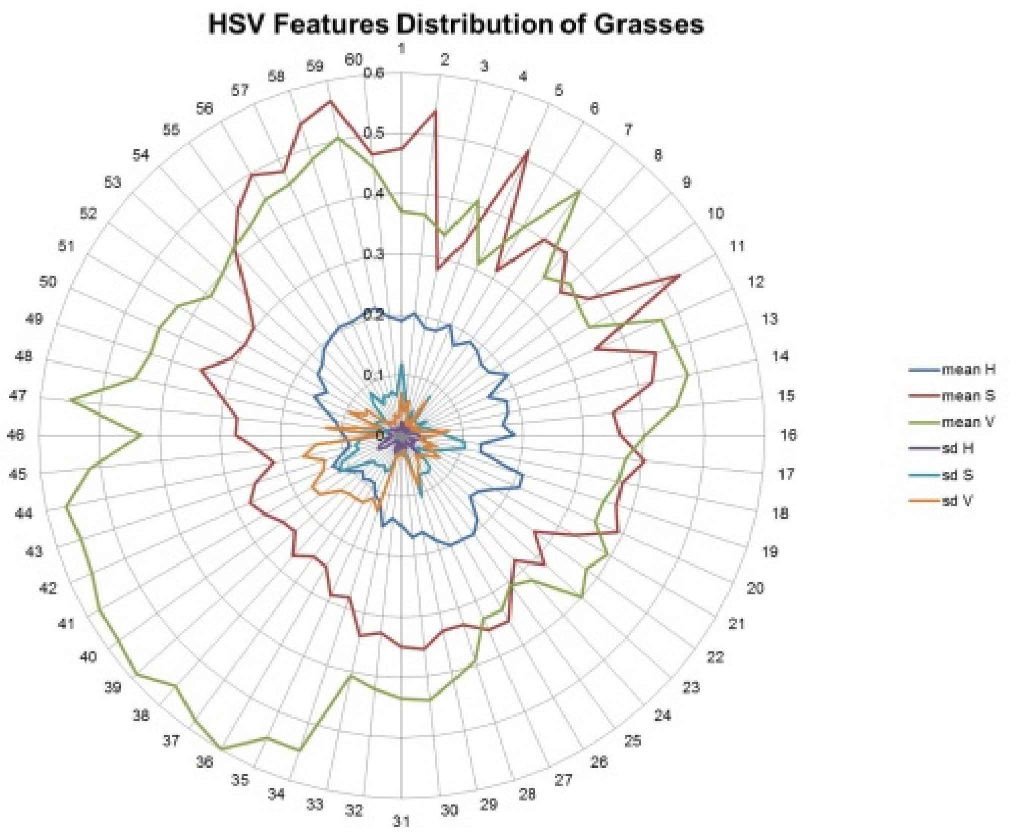
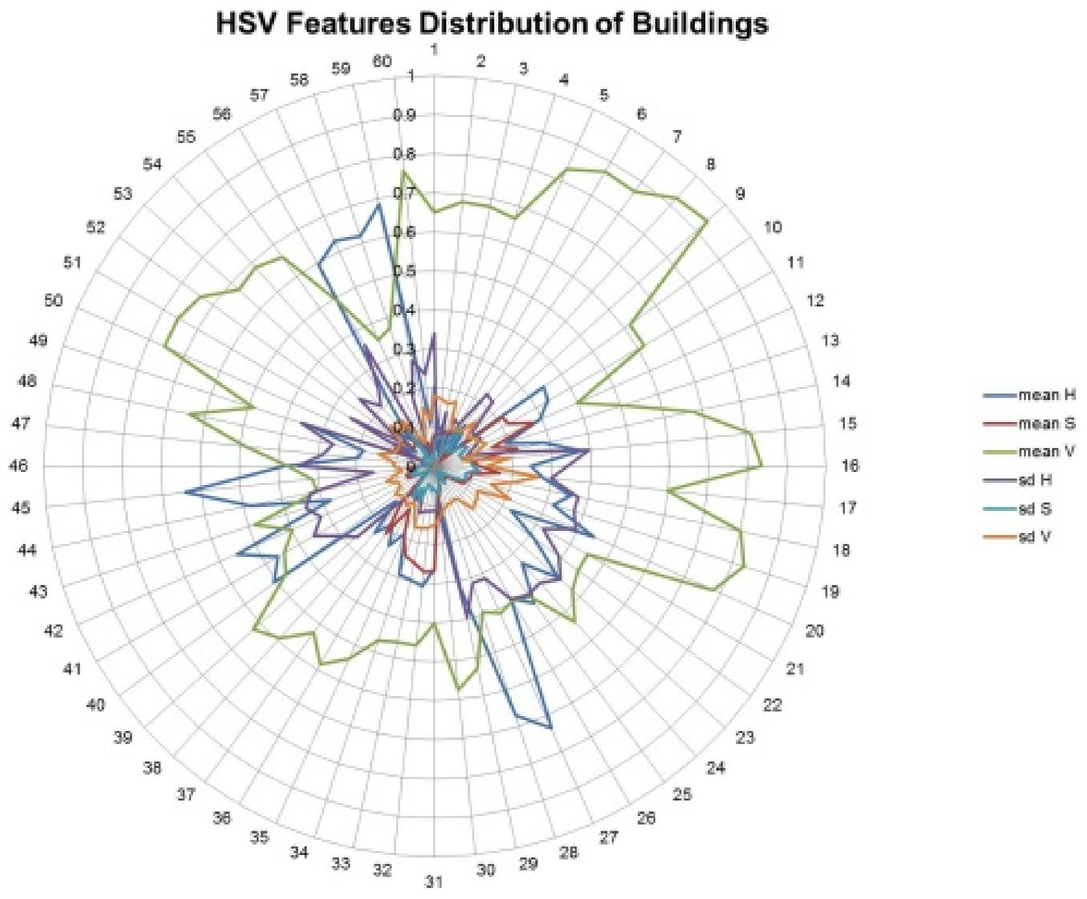
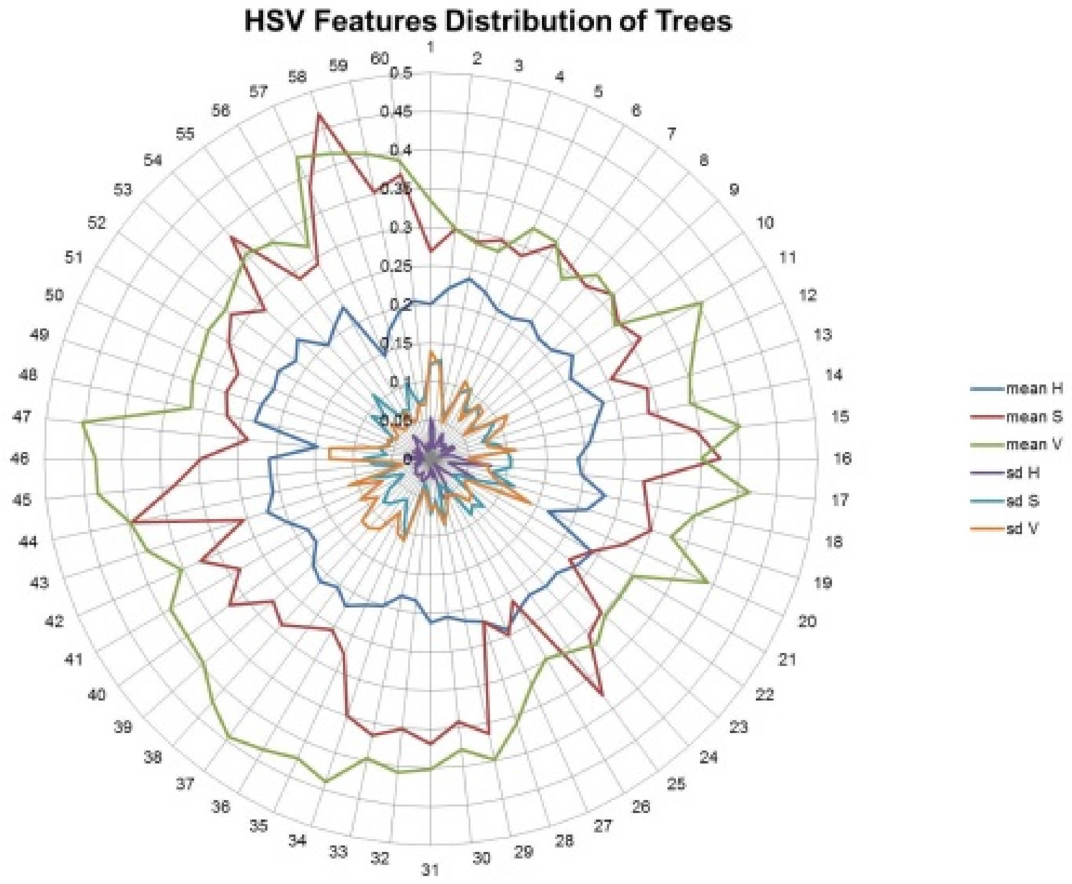


Figure 3.15: HSV histograms for the examples (In top-down orders).

As known, the HSV colour feature is very sensitive to the base colour of the samples, so the environment training samples were divided into different groups: trees, grasses, buildings, road markings, street lights, etc. 60 random samples from each group were selected and the HSV features were calculated. The HSV feature distributions are shown in the figure below (Figure 3.16), which shows that in each group, there is a clear pattern for the HSV feature, so HSV features were added to the classification descriptors used for detection.


HSV Features Distribution of Grasses
mean H
mean S
mean V
sd H
sd S
sd V


HSV Features Distribution of Buildings
mean H
mean S
mean V
sd H
sd S
sd V


HSV Features Distribution of Trees
mean H
mean S
mean V
sd H
sd S
sd V

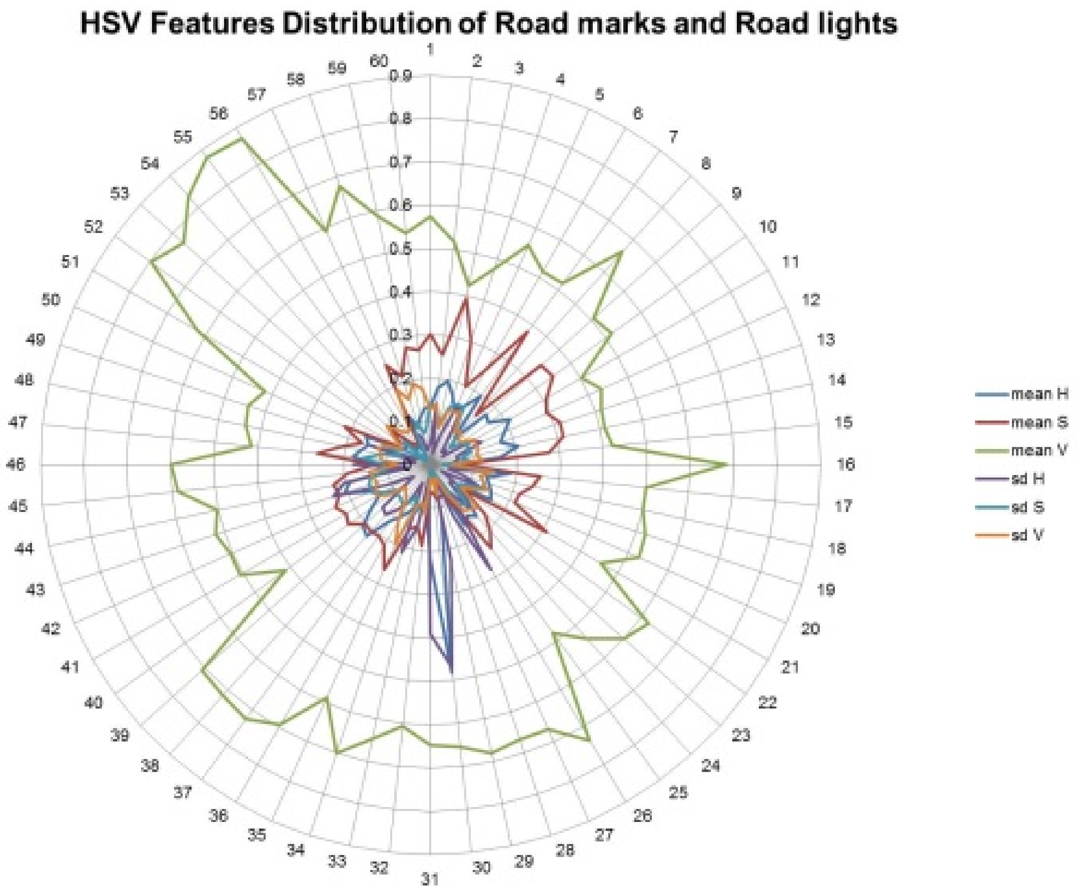


Figure 3.16: HSV distributions for the environments samples.

The HSV model was applied to the detection algorithm. The main propose of using HSV is to identify objects in the environment other than vehicles. After detection using the GLCM feature, the results were checked again with the HSV model to detect whether the result windows contained environmental object colour. One of the testing examples is shown in Figure 3.17. After implementing the HSV feature, the environment windows that contained grass and street lights were marked in green marker. The statistical results of the comparisons of applying the HSV feature into GLCM with the pure GLCM feature will be shown in the section 3.6.

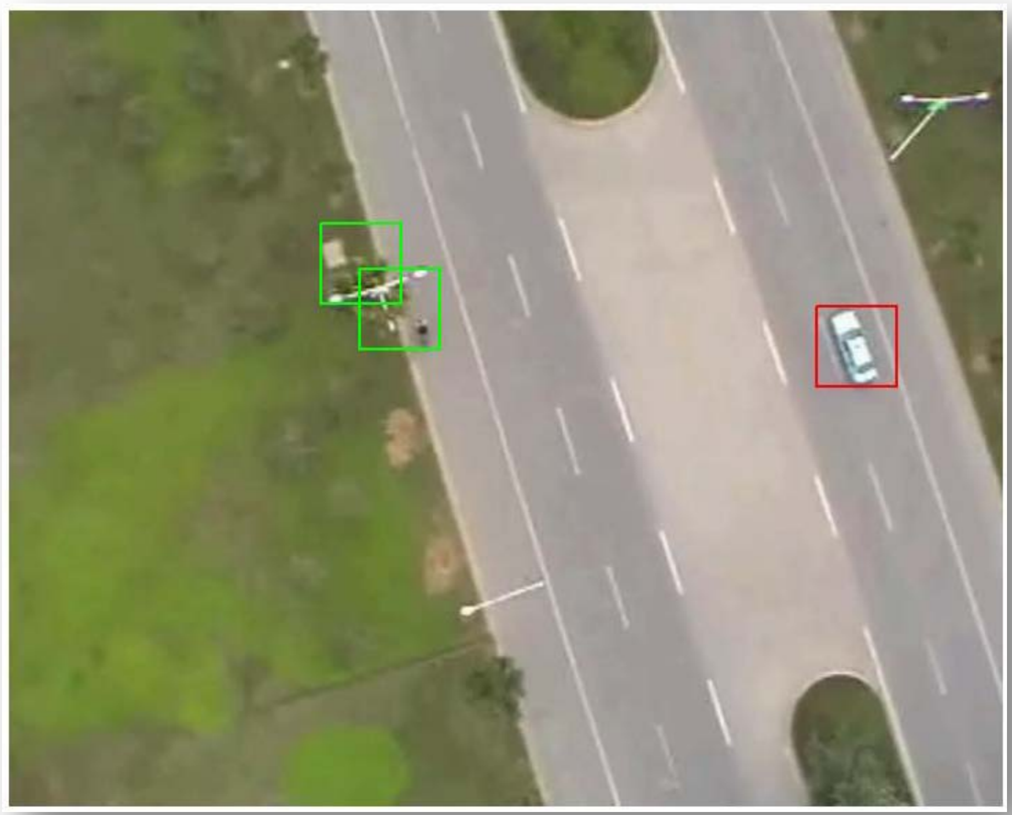


Figure 3.17: The result of testing with the HSV feature added to the GLCM feature.

This detection method is a combination of the GLCM texture feature with the HSV colour feature. In the detection process, objects can be divided into different categories by their texture features. However, the texture of objects from different groups can be similar to each other, which can cause false detections. Thus, the HSV colour feature is used in the proposed detection method to reduce false detection based on the GLCM texture. The unique colours of the background (grass, trees and building) were selected from training images to set the boundaries of the environment objects in order to remove the fault detections caused by similar textures.

3.5 Blur Removal

During testing, the detection result was affected badly by blurred images. Figure 3.18 shows an example of detection in a blurred image using the proposed detection method, which shows that most of the vehicles in the images were miss detected.

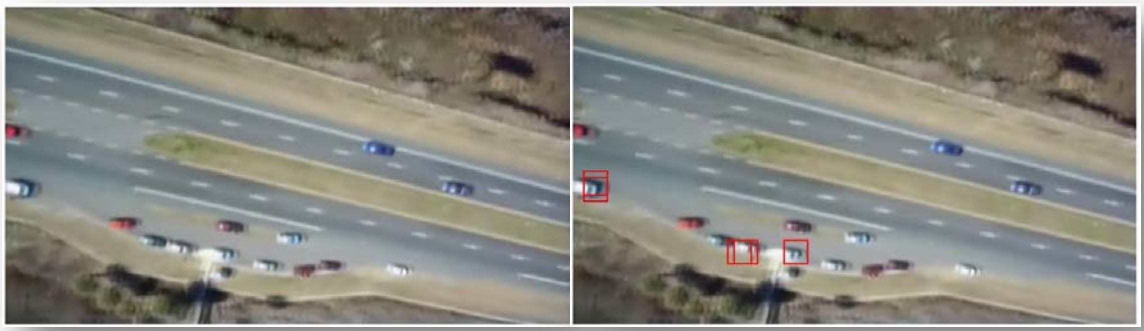


Figure 3.18: A blurred image and the detection result.

Several different methods have been developed in literature for image de-blurring [112,113,114]. In the de-blur process, a variable of distortion is set which gives the noise function which has caused the blur. However, in this situation, the blurring noise is unknown so the system has to simulate a noise level by itself. In this proposed approach, the Blind Deconvolution Algorithm (BDA) [115] de-blur mechanism was used to estimate the noise. The BDA is shown below; H is the estimated blur function of the input image U and its degraded image V according to the image U in M number of blocks.

$$\log|H| = \frac{1}{M}\sum_{k=1}^{M}[\log|V_k| - \log|U_k|] \qquad (3.13)\ [115]$$

After the calculation of the blur function, a Point Spread Function (PSF) was applied, which describes the diffraction pattern of light emitted from a point and transmitted to the image plane. In the process of the PSF, the estimated blur function H can be used as the size of the PSF, which is the most important variable in the de-blur process. In order to improve de-blurring, a weighted array was created to filter out the high-contrast areas, which could reduce contrast-related ringing in the outcome. The de-blurring process has multiple iterations and processes which help to reduce the blurring of pixels and re-join similar elements in blurred images. Figure 3.19 shows the de-blur results and the detection results after the de-blurring process.

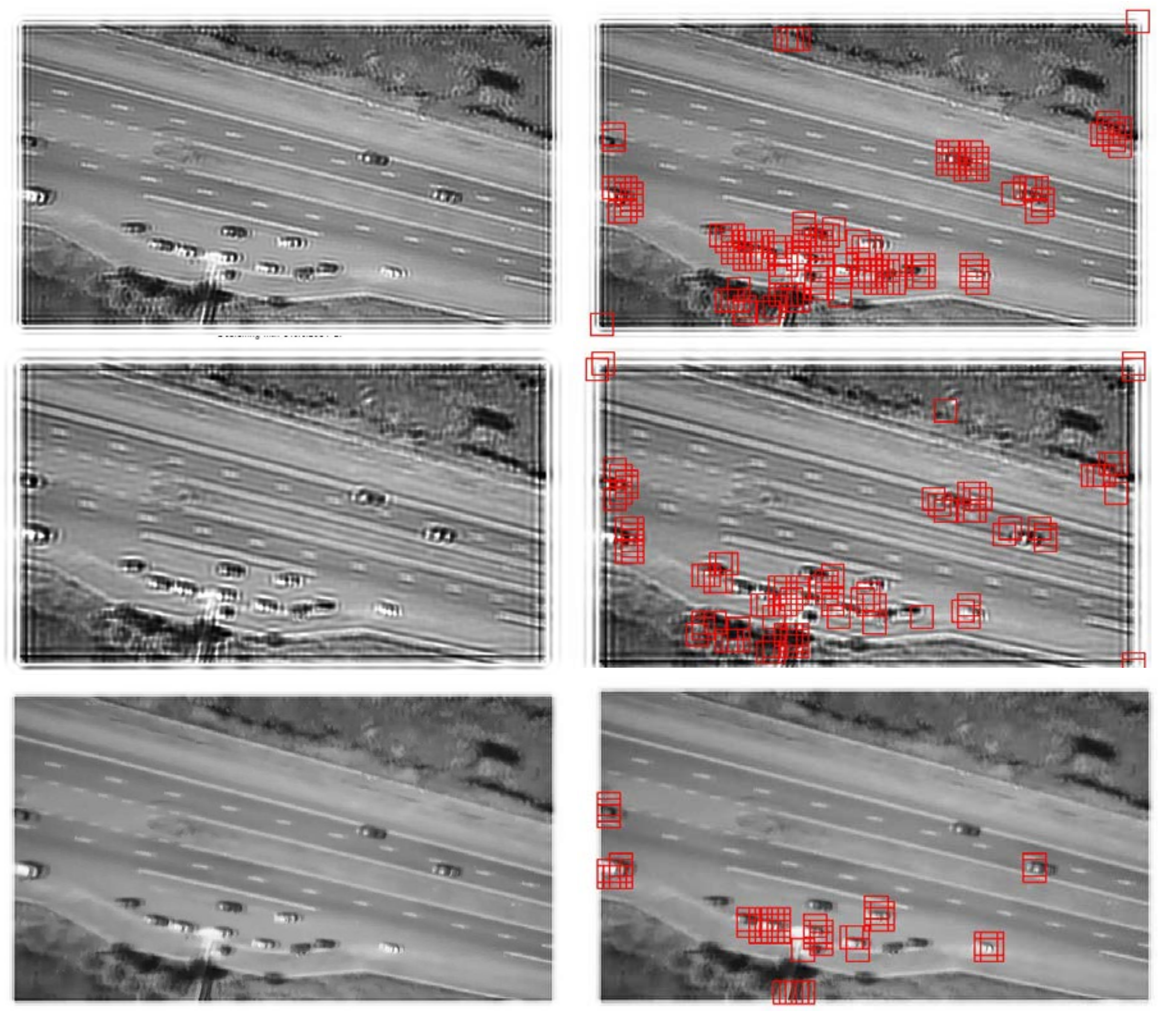


Figure 3.19: Detection results using the new de-blurring approach. (Top row: five iterations; middle row: three iterations; bottom row: one iteration)

3.6 Evaluation

3.6.1 Evaluation Criteria

For the evaluation of the detection and tracking, vehicles are located by bounding boxes, which outline the positions in the image sequences. The most common detection accuracy evaluation methods used by the researchers are Receiver-Operating-Characteristic (ROC) [116], Recall-Precision-Curve (RPC) [117] and the F-measure [118].

First of all, in each frame, correctly detected vehicles are referred to as True Positives (TP), background regions that are incorrectly classified as vehicles are considered False

Positives (FP), and vehicles that were missed are recorded as False Negatives (FN). The detection and tracking hypotheses are shown in Table 3.12.

	Predicted positive	**Predicted negative**
Positive Ground truth (PG)	True Positive (TP)	False Negative (FN)
Negative Ground truth (NG)	False Positive (FP)	True Negative (TN)

Table 3.12: The detection hypothesis matrix.

The Receiver-Operating-Characteristic (ROC) has been developed to measure the binary classifier performance by calculate the true positive rate and false positive rate, which are computed as:

$$\text{true positive rate} = \frac{TP}{TP + FN} \qquad (3.14)\ [116]$$

$$\text{false positive rate} = \frac{FP}{FP + TN} \qquad (3.15)\ [116]$$

The true positive and false positive rate can be plotted in a 2D coordinate system. In the literature [116], ROC has been used as the evaluation criteria. However, in order to apply the ROC in object detection, a sliding window search has to done over the test image to estimate the possible locations. In this case, ROC plot is sensitive to the number of windows tested, which the results could be changed by the parameter of the sliding window approach. Therefore, the ROC curves are not reliable.

The Recall Precision Curve (RPC) indicates the relationship between the detection recall to the detection precision, which the recall measure the number of positive examples successfully detected by the system and the precision measures the hypotheses percentage. The values are computed as:

$$\text{recall} = \frac{TP}{TP + FN} \qquad (3.16)\ [117]$$

$$\text{precision} = \frac{TP}{TP + FP} = \frac{TP}{hypotheses} \qquad (3.17)\ [117]$$

In the literature [117], RPC has been used as the evaluation criteria. In addition, both recall and precision value depends on the true negative value, which is not necessarily needed during the detection. Thurs, it is believed that the RPC is more reliable for the object detection evaluation.

The F-measure metric can provide a more informative measurement in the evaluation by using the recall and precision value. The F-measure is calculated as:

$$\mathrm{F} = 2 \times \frac{precision\ \times recall}{precision + recall} \qquad (3.18)\ [118]$$

where the F-measure can provide a value between 0 and 1, which larger value indicates higher detection rate.

According to the literature, it is believed that F-measure is the best measure to evaluate the detection performance in the realistic scenario. In addition, the detection rate was evaluated, i.e. its ability to correctly identify the regions in the testing videos that contain vehicles. To assess the detection rate across all the video sequences it is assumed that for each frame the number of positive detections t is indicted by $\mathrm{p_t}$ and the number of ground-truth vehicles is indicted by $\mathrm{N_g^{(t)}}$; the detection accuracy can be computed as:

$$\text{Detection Accuracy} = \frac{\sum_{\mathrm{t=1}}^{\mathrm{N_{frames}}} \mathrm{p_t}}{\sum_{\mathrm{t=1}}^{\mathrm{N_{frames}}} \mathrm{N_g^{(t)}}} \qquad (3.19)$$

3.6.2 Evaluation results

The testing process was carried out using the five testing videos as explained in the appendix with different challenges. Testing was split into different steps. First of all, detection using only the GLCM was applied. Secondly, the HSV feature was added to the detection in order to see the difference the improved method can provide. Finally, the de-blur feature was added to the detection process. The detection performance was evaluated by the detection hypotheses matrix and the F-measure metric.

Detection was run on each frame and all of the detections were accumulated. Table 3.13 shows the detection results for each video using the GLCM feature only. The average detection results floated around 88% in accuracy and the F-measure, except Video 4, which achieved a very low detection rate of 61.85%. Video 4 is the blurred image challenge, so it can be concluded that GLCM feature detection is very sensitive to noise in an image. In Video 5, the detection rate is very high because the video contains only a single vehicle against a simple background, which is very easy to detect.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4579	531	745	86.00%	87.77%
Video 2	5511	4847	452	664	87.95%	89.68%
Video 3	5134	4603	387	531	89.66%	90.93%
Video 4	1848	1143	204	705	61.85%	71.55%
Video 5	1918	1876	63	42	97.81%	97.28%

Table 3.13: The results of detection using the GLCM feature only. (Vehicles means the sum of total vehicles appeared in each frame)

Then the HSV colour feature was added to the GLCM feature. The performance resulting from using the HSV-GLCM feature is shown in the Table 3.14. With the HSV-GLCM feature, the results increased for every testing video. This is because the false positive errors were reduced by adding the colour feature; the compression between Table 3.13 with Table 3.14 shows that the FN figure for each video has dropped. HSV successfully distinguished environment objects with similar feature; however, the detection results were still very low in the Video 4, which means the HSV colour feature cannot solve the blur problem.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4938	326	386	92.74%	93.27%
Video 2	5511	5213	265	298	94.59%	94.87%
Video 3	5134	4936	157	198	96.14%	96.52%
Video 4	1848	1267	198	581	68.56%	76.49%
Video 5	1918	1883	55	35	98.17%	97.66%

Table 3.14: The results of detection using the GLCM and HSV feature. (Vehicles means the sum of total vehicles appeared in each frame)

Next, the de-blur mechanism was integrated into the detection process; the results of detection using HSV-GLCM with de-blurring are shown in Table 3.15. The detection results generally remained the same, but the results for Video 4 increased to 90.47% and 87.60%, which means the detection performance is no longer affected by blur problems.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4942	328	382	92.82%	93.29%
Video 2	5511	5231	266	280	94.92%	95.04%
Video 3	5134	4938	166	196	96.18%	96.46%
Video 4	1848	1643	152	205	88.91%	90.20%
Video 5	1918	1884	56	34	98.22%	97.67%

Table 3.15: The results of detection by using HSV-GLCM feature with de-blurring.

The chart in Figure 3.20 shows the results of using all three detection methods on all five testing videos. The y-axis indicates the percentages of detection accuracy and F-measure. The chart shows that the detection rate when using HSV-GLCM and de-blurring was the best, above 90%; it also solved problem of the large amount of false negative errors caused by the blur problem in Testing Video 4. Figure 3.21 shows some example outcomes of detection using this proposed detection method.

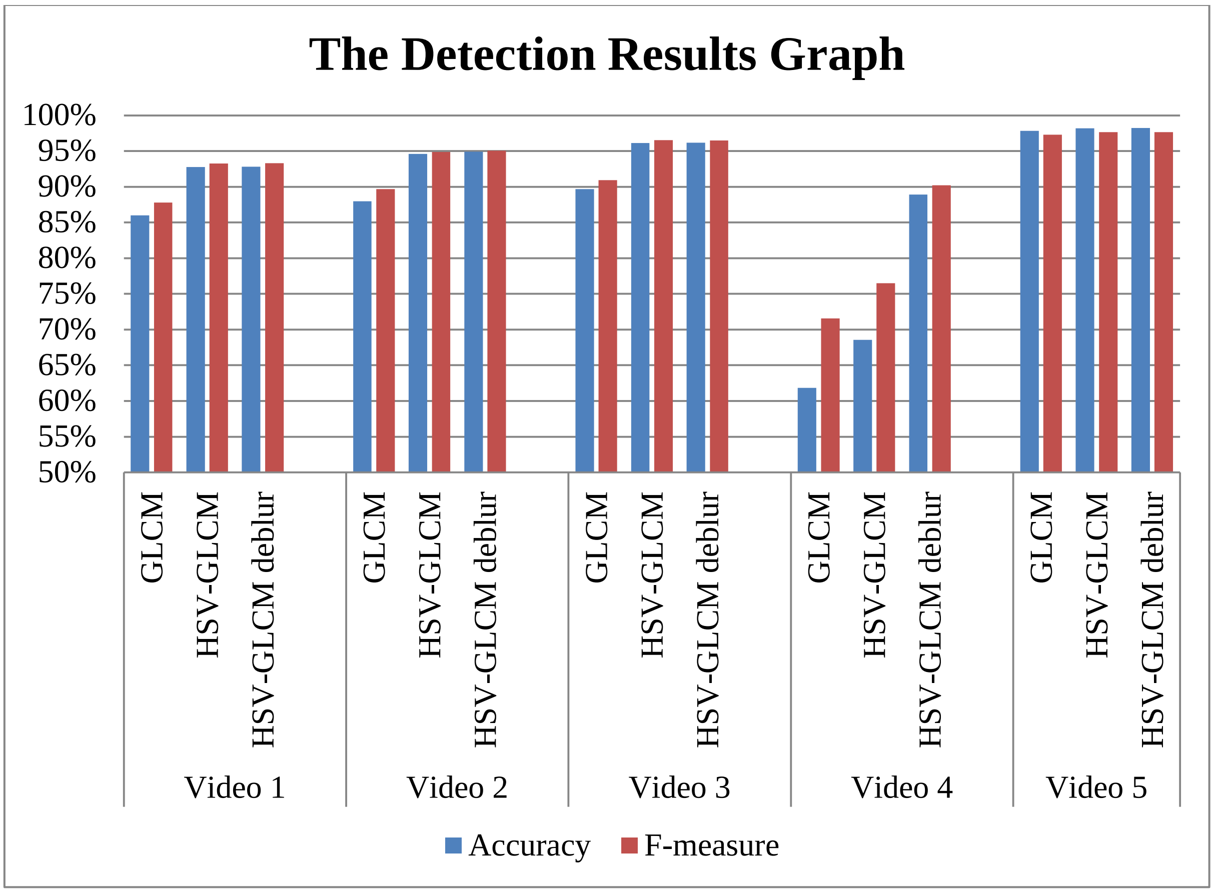
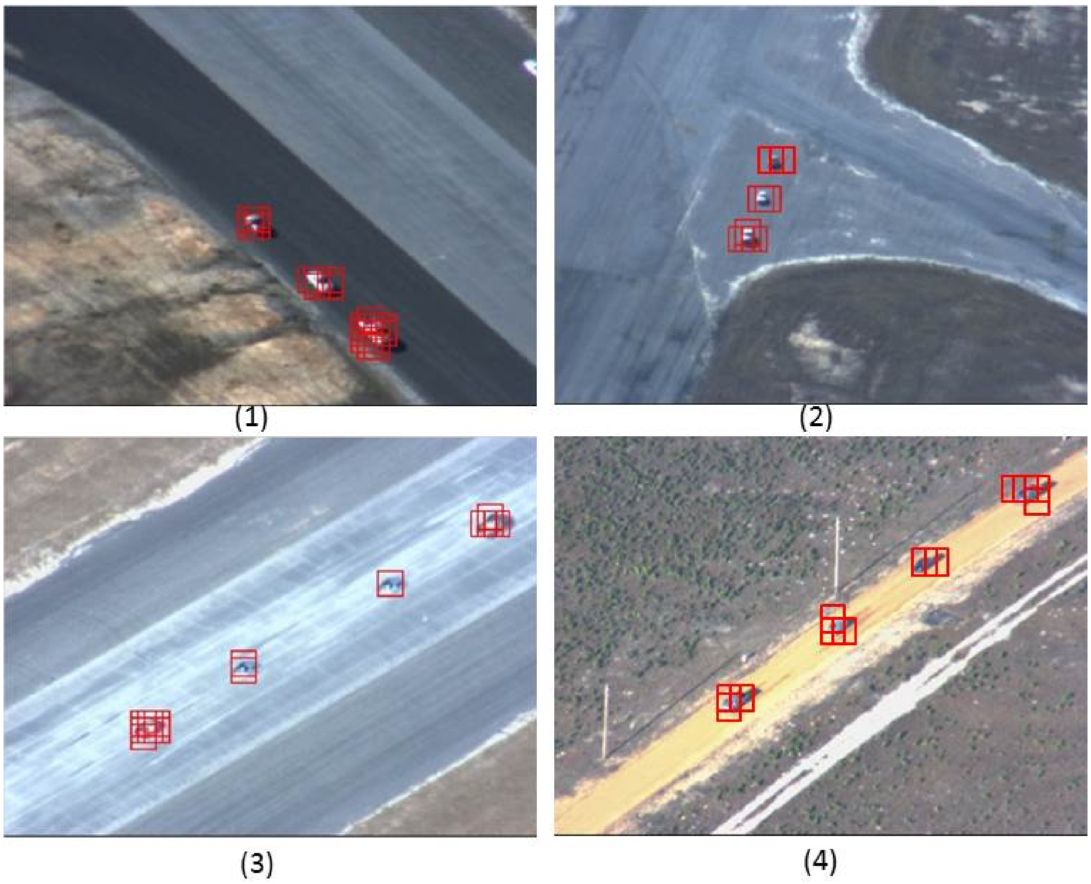
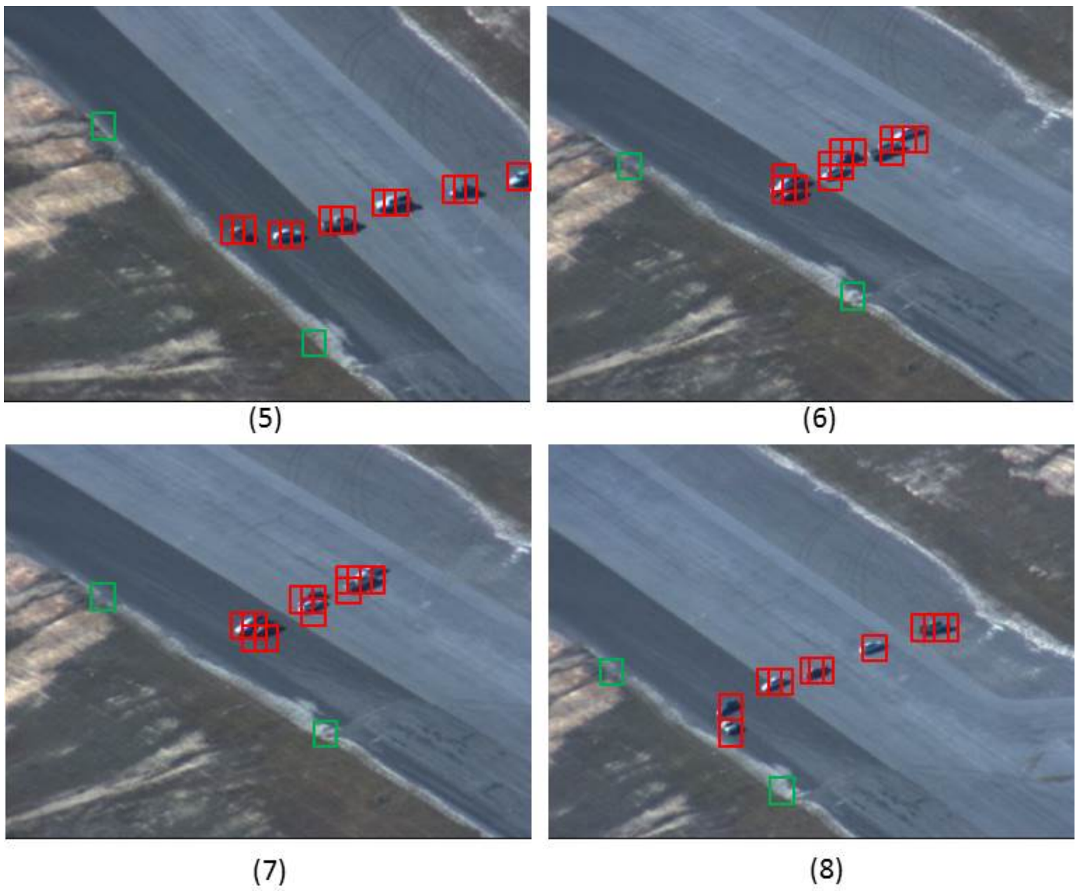


Figure 3.20: Chart of the results of using all three methods in all testing videos.


(1)
(2)
(3)
(4)


(5)
(6)
(7)
(8)

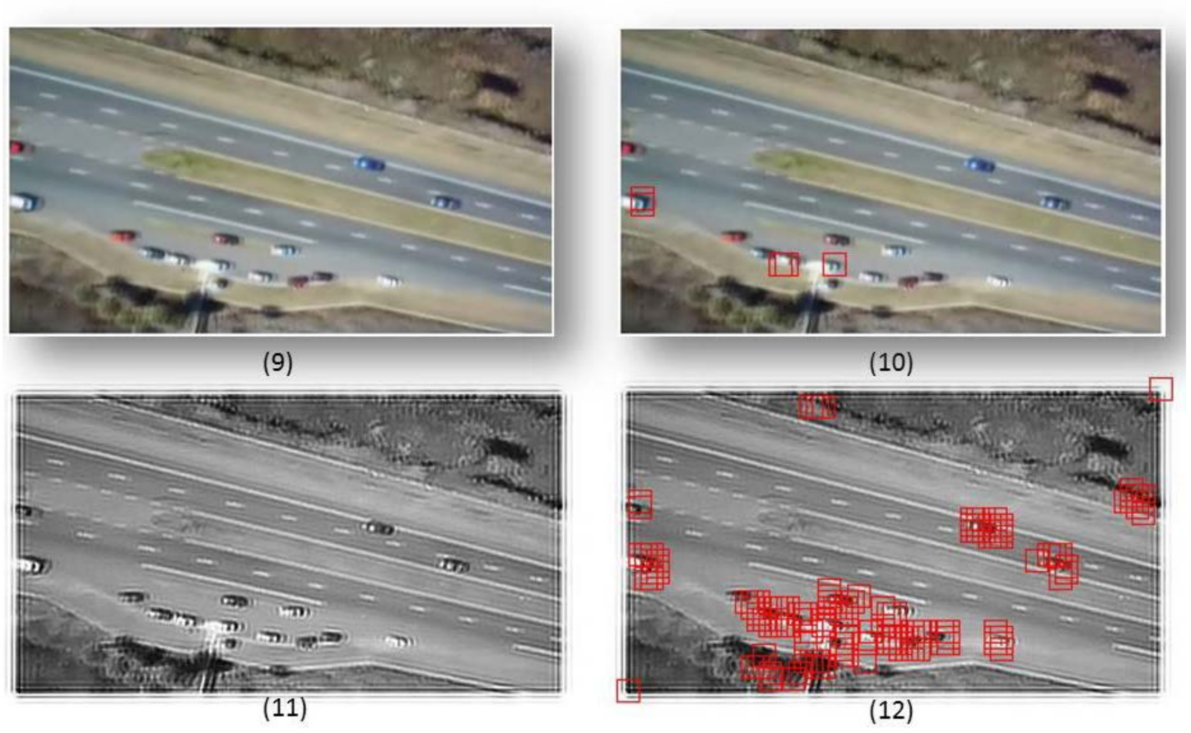


Figure 3.21: Detection outcome examples (Images 1 to 4 indicate the detection result examples in 4 different frame images from 1 UAV video; images 5 to 8 indicate the detection results in a continuous frame sequence, Red boxes are the predicted vehicles and green boxes are the predicted environment; the image 9 to 12 indicate the detection results before and after the de-blur process.).

3.7 Conclusions

In this chapter, a novel combination detection method has been proposed. The detection method combines the GLCM texture feature extraction and the HSV colour description to achieve the vehicle detection task. Additional, a de-blur algorithm has added in the detection scheme in order to tackle the blurring challenge. In the introduction chapter, 4 different challenges of vehicle detection and tracking have been proposed. The main challenge in the vehicle detection process is to distinguish between the target vehicles and the environmental objects. In the object detection process, the objects in a testing image can be classified into different groups according to their texture values; in this case, the GLCM texture. However, the texture of objects from different groups can be similar, which causes the false detections. In order to tackle this problem, the HSV colour feature was added into the classification process, which can provide more accurate descriptor of the testing object. Moreover, a de-blur mechanism by using BDA with PSF has been added into the detection process to solve the blurring challenges. Finally, the different size and orientation challenge can be solve during the parameter setting in the GLCM feature descriptor. According to the detection results, the proposed detection method can successfully solve the challenges. For example, the detection result in video 4 indicates that by adding the HSV colour descriptor, the accuracy and F-measure increased from 61.58%, 71.55% to 68.56%, 76.49% and by adding the de-blur mechanism the accuracy and F-measure increased to 88.91%, 90.20%. However, the results can provide the conclusion of the HSV-GLCM with de-blur method is very sensitive to the resolution of the image and easy to affected by the blur problem. To enhance the detection accuracy under these situations, a novel detection method of ISM-SIFT will be introduced in the following chapter. Also, the comparison of the detection accuracy to other methods will be present in the following chapter, which to indicate whether the proposed method can improve the detection accuracy than the existing approaches.

Chapter 4

The Proposed Implicit Shape model with SIFT Detection Algorithm

In this chapter, an ISM-SIFT vehicle detection method is proposed. This method includes the features of Scale Invariant Feature Transform (SIFT) and the Implicit Shape Model (ISM). The main contribution of this method is the integration of the SIFT process and the ISM in vehicle detection system which increases the detection accuracy compared with the results of the detection using each process separately.

In 2004, David G. Lowe first proposed the Scale Invariant Features Transform (SIFT) feature to extract key points and compute their descriptors [119] and then made further improvements [120]. Lowe not only presented SIFT point descriptors but also discussed key point matching [121]; the SIFT descriptors are scale invariant, rotate invariant and affine distortion invariant. SIFT descriptors have more specific information for each key point than the FAST corner point; each SIFT point has a unique descriptor of 128 bytes. So the same point can be found in different frames, even if its status has changed. However, because a SIFT descriptor carries so much information, processing can take a long time. So the Implicit Shape Model (ISM) was applied to the detection with SIFT so the system can narrow down the detection region, which speeds up the detection process.

4.1 Scale-Invariant Feature Transform (SIFT)

During flight, UAVs can be affected by turbulence, which can cause unstable images. The size of the target vehicle can change according to the flight status of the UAV; for instance, the altitude of the UAV can change the size of the vehicle, a sudden change of the speed and vibration may blur the image, and different capture angles can change the shape of the target during continual flight. SIFT descriptors are the extreme points of Gaussian scale space differences. In a Gaussian image, each key point is the results of either the maximum or minimum values from the comparison of its 26 neighbourhood pixels with the current, upper and lower scales. The SIFT algorithm calculates the unstable extreme point and the accurate position of the pixel by using the Taylor expansion and the Hessian matrix. In a Gaussian image, the gradient value and direction of each pixel in the neighbourhood near the key point is also calculated to find the key point independence scale and the direction. SIFT key points can be invariant to scale, rotation, and translation. These features can solve the problem of unstable images during flight. There are four main steps involved in SIFT feature extractions which are:

Insert image and detect extreme points
Determine and filter the keypoints
Orientate each key point
Generate the SIFT vector for each key point

4.1.1 Detecting Extreme Points in Scale Space

Firstly, scale-invariance is one of the characteristics of the SIFT point feature. Koenderink and Lindeberg [122, 123] proved that a Gaussian convolution kernel is only for linear nuclear scale transformations. So an input image was defined as $\mathrm{I}(\mathrm{x},\ \mathrm{y})$. The keypoints are the stable points across the image, which can help identify the possible scale-invariant locations. The scale space of an image can be defined as $\mathrm{L}(\mathrm{x},\ \mathrm{y},\ \sigma)$ with the variable scale Gaussian function G (x, y, σ), and σ is the scale factor.

$$\mathrm{L}(\mathrm{x},\mathrm{y},\sigma) = \mathrm{G}(\mathrm{x},\mathrm{y},\sigma) \times \mathrm{I}(\mathrm{x},\mathrm{y}) \qquad (4.1)\ [119]$$

The Difference of Gaussian (DoG) was used in the detection of the stable key points, which can find the scale-space extrema in DoG.

$$\begin{aligned}\mathrm{D}(\mathrm{x},\mathrm{y},\sigma) &= \big(\mathrm{G}(\mathrm{x},\mathrm{y},\mathrm{k}\sigma) - \mathrm{G}(\mathrm{x},\mathrm{y},\sigma)\big) \times \mathrm{I}(\mathrm{x},\mathrm{y}) \\ &= L(x,y,k\sigma) - L(x,y,\sigma)\end{aligned} \qquad (4.2)\ [119]$$

where k is a constant factor which separates two adjacent scales from the original image and G D(x, y, σ) is the Difference of Gaussian(DoG).

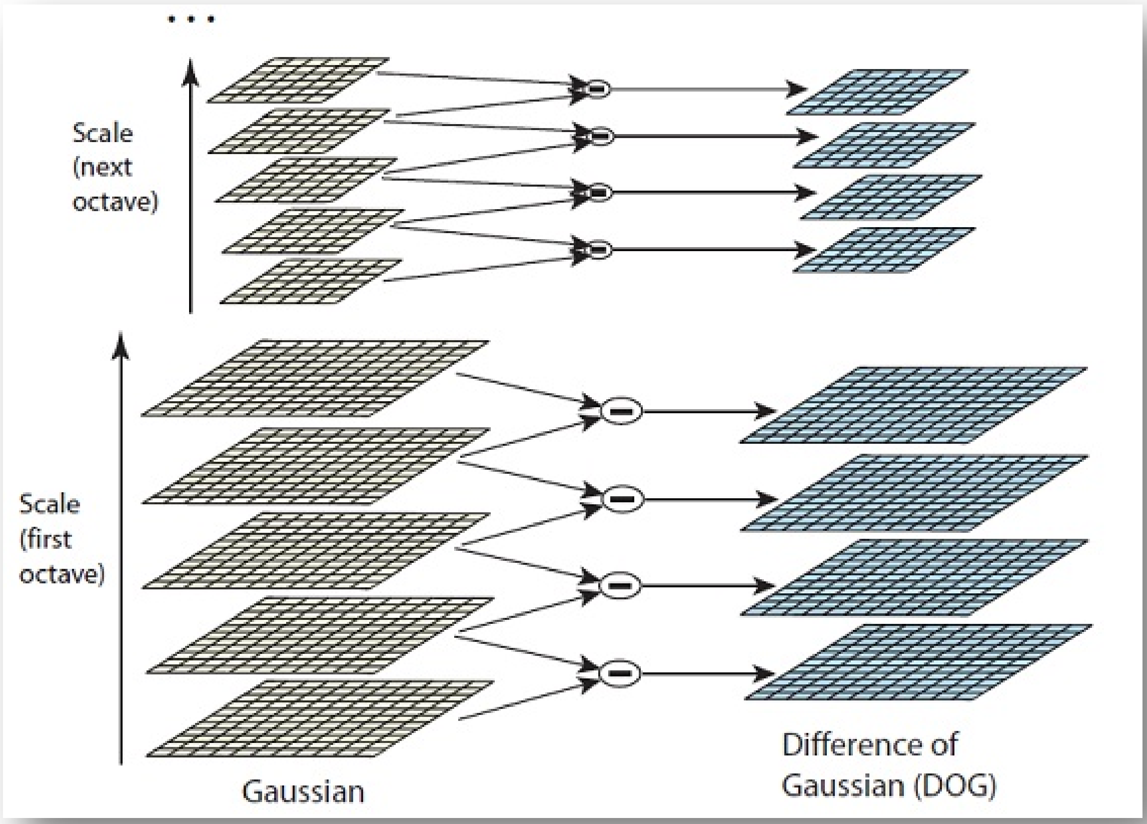


Figure 4.1: Gaussian and Difference of Gaussian.

To find the key points that are invariant to scale, it is necessary to find the difference of the adjacent scale images. A pyramid was created to calculate the DoG (Figure 4.1). After generating the DoG, the system can compare each pixel in the DoG with its neighbours. Each keypoint has to be compared with eight neighbours in the same scale and another 2×9 = 18 neighbours in the upper and lower scales, which means there are 8+2×9 = 26 neighbours in total to compare. If a point is the maximum or minimum in all 26 neighbours, it will be classified as a key point in the image scale (Figure 4.2).

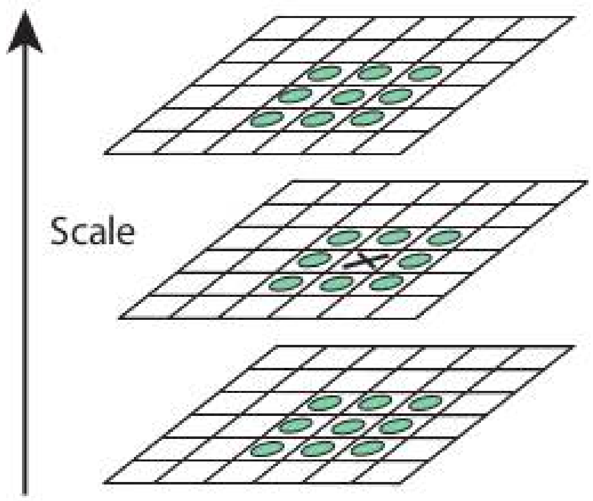


Figure 4.2: Maxima and minima of the Different of Gaussian images.

4.1.2 Determining and Filtering the Key Points

The DoG method is sensitive to the noise and edges in an image, so this method has to set a filter to reject low contrast points and poor points along the edges. Using the fitting 3D quadratic function, the location and the scale of the key points can be more accurately identified.

4.1.3 Orientate each point

For each keypoint, the direction of the point to the maximum of the gradient direction in the histogram is generated. The subsequent descriptor structure takes this direction as a reference. For each image L(x, y), the gradient magnitude m(x, y) and the orientation θ(x, y) are calculated as:

$$\mathrm{m(x,y)} = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2} \qquad (4.3)\ [119]$$

$$\theta(x,y) = \tan^{-1}\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \qquad (4.4)\ [119]$$

The sampling area of a keypoint is in the centre of its neighbours, which adapts histogram statistics to the gradient direction of the neighbour pixel. The range of the gradient histogram is 0 to 360 degrees (36 in total; 10 degrees each). The main key point direction of its neighbours' gradient is represented by the peak of the histogram and if there is other energy that reaches 80% of the peak value, this will be represented as the secondary direction of the key point. As a result, each keypoint has a total numbers of eight directions.

4.1.4 Generating the SIFT Vector

In the last step, each keypoint is assigned a descriptor, which has gradients to achieve further invariance. In the construction of these descriptors, the direction of each descriptor is rotated to the main direction of the image, which satisfies rotational invariance. After that, an 8×8 window of the keypoint is created. As shown in Figure 4.3, the centre point in the left image, in which every cell represents a pixel of a key point's neighbourhood, uses the formula to get the gradient and gradient direction of each pixel. The arrow direction represents the gradient direction of the pixel and the length represents the gradient modulus. The blue circle represents the ranged Gaussian weight. Further calculations give the direction of the gradients of the eight direction histograms for each 4×4 block, which accumulates the value of each gradient direction. Finally, the total of the neighbourhood's directional information can reduce the noise, which can provide a better fault tolerance for detections.

Lowe proposed $4 \times 4 = 16$ seeds to describe each keypoint in the calculation process in order to increase the matching rate. So $4 \times 4 \times 8 = 128$ bytes of data are formed for each key point. This is then normalised to unit lengths, which can reduce the defects caused by illumination changes. Any change in contrast in any pixel is cancelled by vector normalisation.

Now the SIFT feature has removed all the invariance effects, including changes in scale, rotation and illumination. Two vectors were found which are: the coordination position of the key points with the orientation and scale value, and a 128 bytes descriptor. Ultimately, the key point can be located from the coordination and classify the key point using the descriptor.

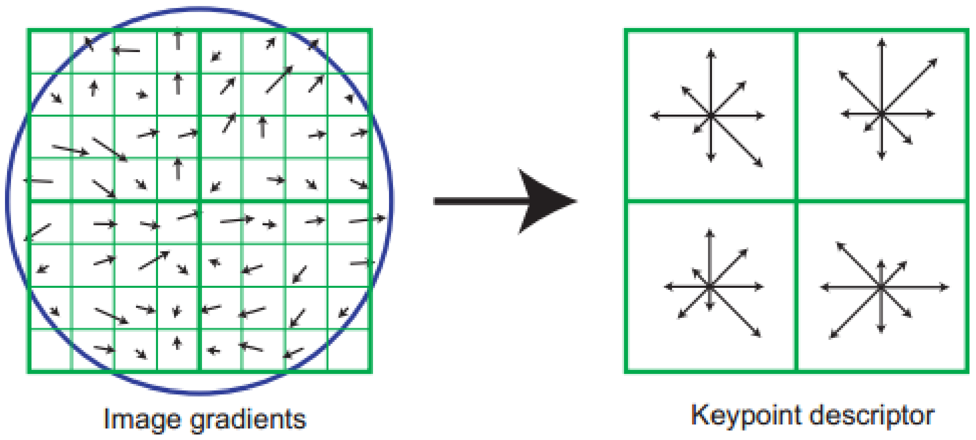


Figure 4.3: Key point gradients and descriptor.

4.1.5 SIFT Parameters

4.1.5.1 Scale Space Parameters

The SIFT detector and descriptor are generated from the Gaussian scale space of the input image I. The formula for the Gaussian scale space is:

$$\mathrm{G}(x;\sigma) \triangleq (g_\sigma \times I)(x) \qquad (4.5)\ [119]$$

where $g_\sigma$is an isotropic Gaussian kernel of variance σ^2 I, x is the spatial coordinate and σ is the scale coordinate. This algorithm can also apply to the Difference of Gaussian (DoG), which is the scale derivative of the Gaussian scale space. The scale space $\mathrm{G}(x;\sigma)$ represents the image I at different levels of scale which can reduce the redundancy. The domain of the variable σ is discretised in logarithmic steps into O octaves, and each octave is further divided into S sub-levels. At each successive octave, the data is spatially reduced by half, so the distinction between the octaves and sub-levels is significant. The octave index o and the sub-level index s are represented by the corresponding scaleσ :

$$\sigma(o,s) = \sigma_{0}{}_{2^{o+s/s}}\ , \mathrm{o}\ \in\ o_{min} +\ [0, \ldots, O-1], s\ \in\ [0, \ldots, S-1] \qquad (4.6)\ [119]$$

where σ_0 is the base scale level. The Gaussian scale space parameters include:

- Number of octaves
- Index of the first octave
- Number of sub-levels
- Base smoothing

4.1.5.2 Detector Parameters

SIFT frames (x, σ) are the local extreme points selected in the DoG scale space. The selection is based on the parameters of:

- Local extrema threshold
- Local extrema localisation threshold
- Boundary point removal

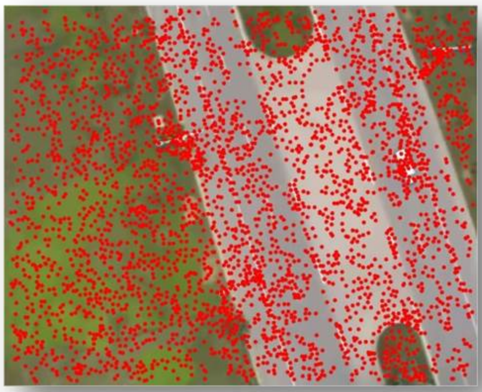
4.1.5.3 Descriptor parameters

The SIFT descriptor is a weighted and interpolated histogram of the gradient orientations and locations in a patch surrounding the keypoint. The parameters for the descriptors are:

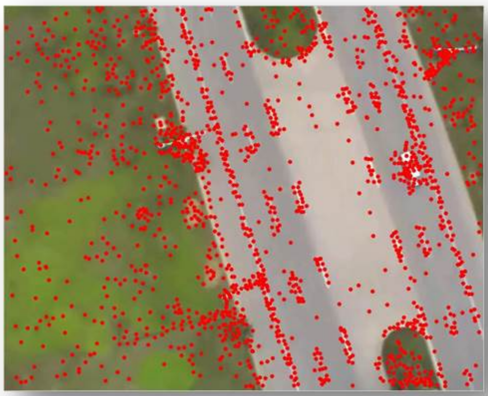
- Magnification factor
- Number of spatial bins
- Number of orientation bins

4.1.5.4 SIFT Parameter Settings

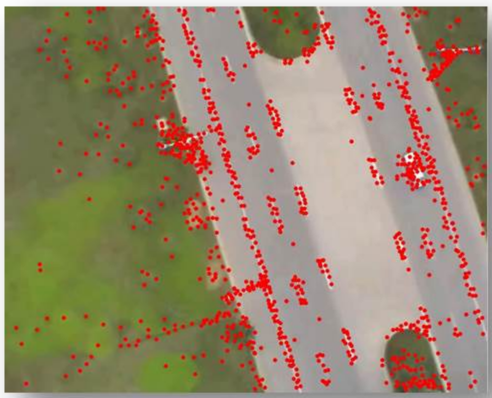
The SIFT descriptor can detect different numbers of the extreme points in an image with different threshold settings. Also, the descriptors need to set a suitable threshold for detection in order to increase the efficiency of the programme. In the testing, seven different SIFT descriptor thresholds were applied to one image to select the best threshold (Figure 4.4).



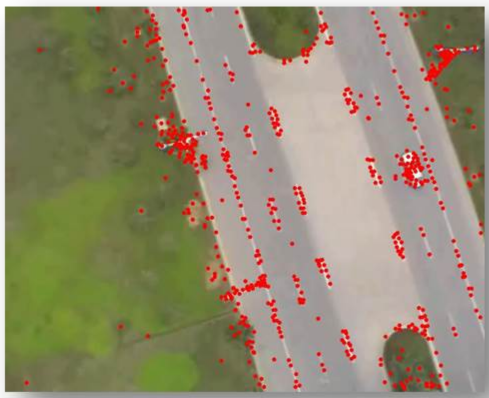
Threshold = 0



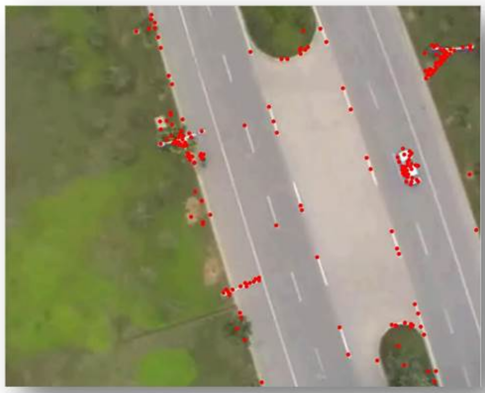
Threshold = 0.5



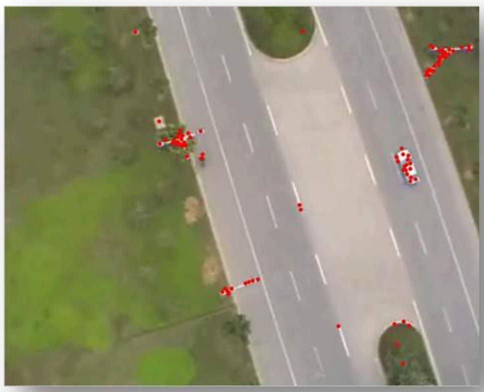
Threshold = 1



Threshold = 2



Threshold = 5



Threshold = 7

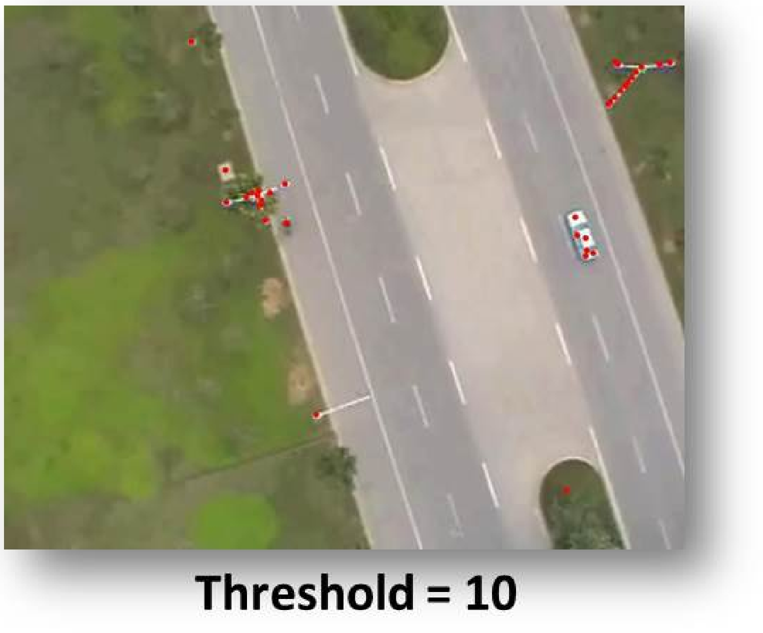


Figure 4.4: SIFT points at different thresholds.

Testing results:

Threshold Value	0	0.5	1	2	5	7	10
Number of SIFT points	3925	1910	1314	771	235	116	52
Processing Time(s)	1.857	1.760	1.748	1.739	**1.713**	**1.717**	**1.715**

Table 4.1: SIFT point results at different thresholds.

Table 4.1 and Figure 4.4 shows the SIFT point detection in one image at different thresholds. According to the results, it can be conclude that a higher threshold can filter un-useful SIFT points, which cost more computational recourses. Table 4.1 shows that the processing time for SIFT points is around 1.7 seconds. The processing time for thresholds 5, 7 and 10 are similar which means the processing time will not be further reduced when the threshold increases. Also, according to the result images at threshold 10, all important objects were detected by the SIFT points and with the lowest noise points. As a result, the SIFT threshold was set as 10. The table below shows all the setting of the SIFT detection according to the experiments.

Parameters	Values
σ_N (SigmaN)	0.5000
σ_O (SigmaO)	2.0158
O (Number of Octaves)	7.0000
O_1 (First Octave)	-1.0000
S_{min} (Smin)	-1.0000
S_{max} (Smax)	3.0000
EdgeThreshold	10.0000

Table 4.2: SIFT parameters.

4.1.6 Classification

The interest points extracted by the SIFT feature from an image are sent to a classifier model by SVM. First of all, the SIFT feature is scale-and-rotational invariant, so it is unnecessary to organise the training samples into different directions and scales. However, the positive and negative points have to be distinguished for the generation of the classification model. The background was removed from the sample images and collect the SIFT point descriptors of the vehicles in the images, which can remove the SIFT point descriptor from the background in order to set as the positive sample (Figure 4.5).

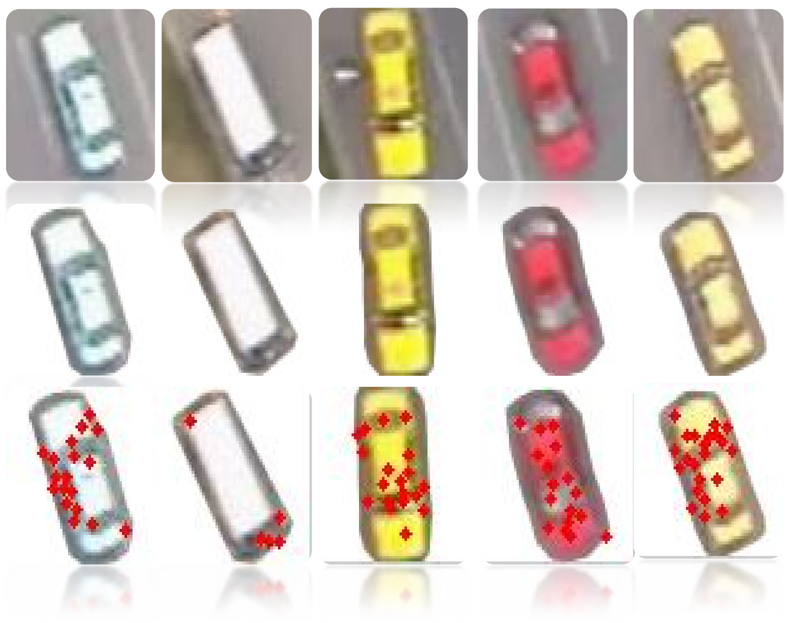


Figure 4.5: Examples of positive sample selections of SIFT points (First row: original sample images; second row: the background removal; third row: distributions of SIFT point's).

4.1.7 Bag-of -Words

After extracted the SIFT points, Bag-of-Words feature [124] was applied into classification. The goal of using this approach is to distinguish the SIFT points as being part of particular vehicles or other objects in the environment. Having extracted SIFT features from the training data, both positive and negative samples, the recognition algorithm determines which the testing area belongs to by selecting the most often occurring SIFT combinations matching the classification, which is the “Bag” feature.

The first step of the algorithm is to collect SIFT features from the training samples and cluster them into a visual vocabulary, called “visual words”. After all of these features were extracted from the training data, these can be clustered using k-means clustering, which the cluster centres can be defined as the “visual words” (Figure 4.6).

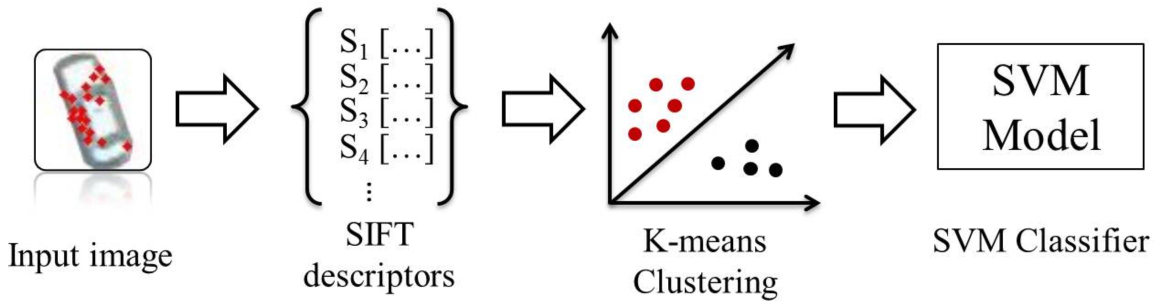


Figure 4.6: Bag-of-Words process.

4.1.8 SVM Classifier Model

850 positive samples were selected, which 42,617 SIFT points were extracted, and 260 negative samples, which 16,518 SIFT points were extracted. Each SIFT point has a 128 bytes descriptor. All 59,135 SIFT point descriptors were inserted into the SVM classifier to create the training model. The SVM can find hyper plans in a kernel-induced feature space, which can divide the training samples into two separate groups: a positive sample group, labelled as 1 and a negative sample group, labelled as 0. After the SVM modelling, the system can now make decisions on whether the input point is related to a vehicle or others object based on the point descriptors.

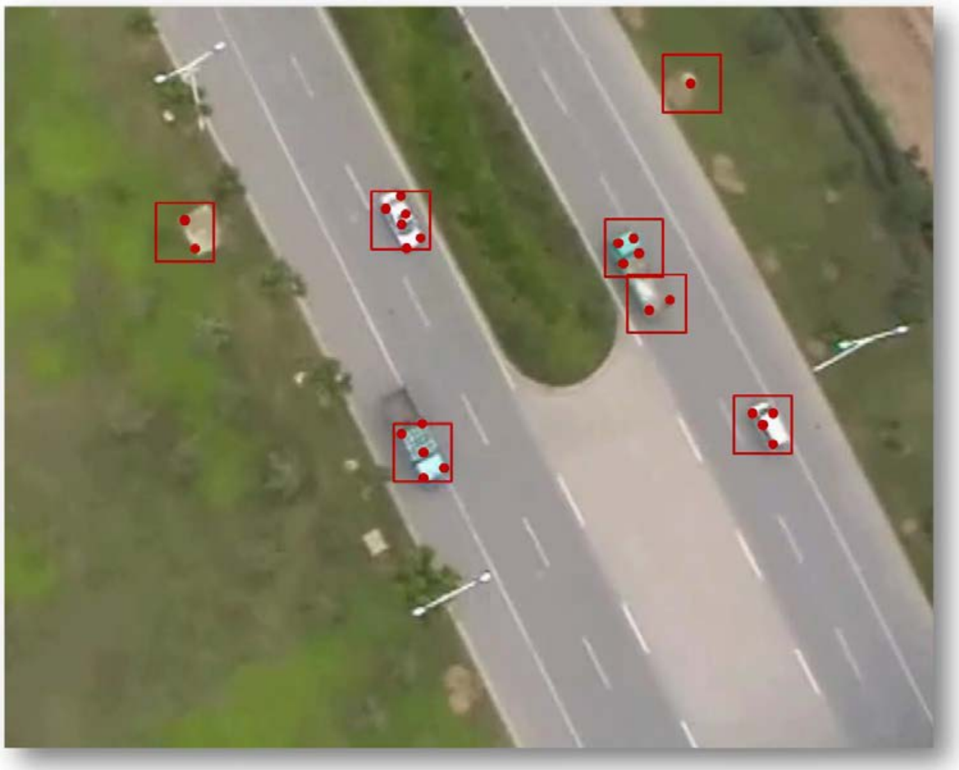


Figure 4.7: Examples of detection results using the SIFT feature approach.

The system was now able to carry out vehicle detection by itself; however, the detection results were not good enough. In the Figure 4.7, which is an example of the detection results, two environment objects were detected as the vehicle and one of the vehicles was detected as two vehicles separately. To improve detection performance, a dimension for final decision making was applied, which is the Implicit Shape Model. The ISM algorithm was intergraded with the SIFT model, which created a new detection process and increased the detection accuracy.

The classification accuracy was evaluated by using different SIFT thresholds. The thresholds of the SIFT can directly affect the numbers of SIFT points that can be detected; with a larger threshold applied, less SIFT points can be detected. Furthermore, the AdaBoost feature was also applied into the SVM classifier training process, which increased the classification accuracy. Figure 4.8 shows the classification accuracy of different SIFT vectors, with and without the AdaBoost. As the Figure 4.8 shows, when the threshold of the SIFT is decreased, more points will be extracted, so there will be more information on the features, which increases the accuracy of classification. Below a threshold value of 10, the accuracy has reached its saturation, so there is no need to use a lower threshold; that would give us too many points, which could reduce the efficiency of the process.

In addition, the results using AdaBoost are clearly higher than those attained by using just the original SVM model. The AdaBoost can perform a constrained gradient descent to optimise the margin in order to minimise the errors. In the literatures [125, 126, 127, 128], the detection performance can be boosted by adding the AdaBoost mechanism into the classification process. Thus, the AdaBoost was used in the classification training process.

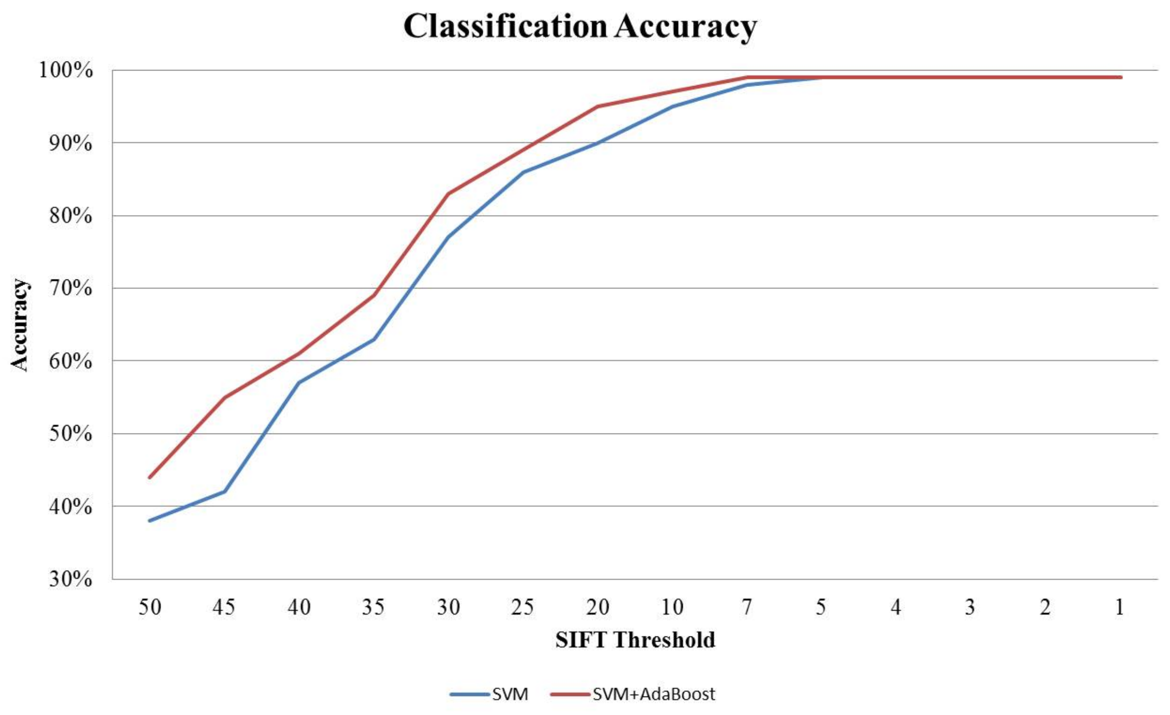


Figure 4.8: Classification results using SVM and SVM with AdaBoost.

4.2 Implicit Shape Model (ISM)

The Implicit Shape Model was originally proposed by Leibe and Schiele [129]. It offers reasonably good detection results for various object categories. The ISM separates object detection and figure-ground segmentation, which are the most distinct features of the object detection process, into two interactional processes. The ISM approach is based on a model of the object and the shape descriptors of the object. The centre of the object is predicted by locating the local features around it. With different descriptors, totally different results from each object category group can be created, so the local image descriptors for the training samples need to be carefully considered before generating the overall object category. In the proposed approach, edge feature descriptors were used to determine the centres of vehicles. The Implicit Shape Model framework can be divided into five steps:

- Codebook generation
- Learning of spatial occurrence distribution
- Hypothesis generation
- Figure-ground Segmentation
- Minimum-description-length based verification

4.2.1 Codebook Representation

The first step of ISM is to create a vocabulary codebook based on the training samples (Figure 4.9). The key reason for this is to let the ISM automatically learn a large group of samples and generate local appearance prototypes. This process can divide the modelling into sub-sectors which can achieve better representative structures.

The codebook process is linked with the SIFT point detection method that is an earlier step of the programme. The reason for use this method was to identify the informative and distinctive locations of vehicles. In this process, the edge features of the local area features were used around the SIFT point, which are extracted and represented in a feature description. The edge feature descriptors are then grouped with an unsupervised clustering scheme to model different types of similar features. The method used agglomerative clustering for the process because it is found that agglomerative clustering offers better object detection performance. The agglomerative method sets every feature as a separate cluster and computes the similarity between all clusters. It is also noticed that this feature similarity can be aggregated in different forms as the descriptors contain several features. The forms of measurements are: single-linkage, complete-linkage, group average, minimum variance etc. Similar clusters are then merged repeatedly based on these measurements until a single cluster has been computed. The final cluster results are represented in hierarchy format. However, Leibe points out that the agglomerative algorithm method requires higher computational resources. The agglomerative algorithm has to compute the similarity between the object matrixes, which requires more computer memory, which often limit the number of object features that can be processed. The programme used an algorithm based on reciprocal nearest-neighbour pair construction, which means that if a point p is assumed the nearest neighbour of q, then point p and q will be considered a pair. However, this approach is only applicable to the clustering distance measure; in this case, Euclidean distance was used for shape context descriptors and Chamfer distance was used for edge patches. In the agglomerative clustering, when the distance between all pair clusters is above the threshold have been set, the clustering process is stopped. The clustering continued until the size reaches a certain value, to make the different feature representations more comparable. It was found that agglomerative clustering generates not only large clusters, but also many small clusters, which means some areas of the sample images are described in detail and others are more general. The experimental results show that this can create a better classification model for the recognition process compared with pure K-means clustering, which generates more balanced clusters.

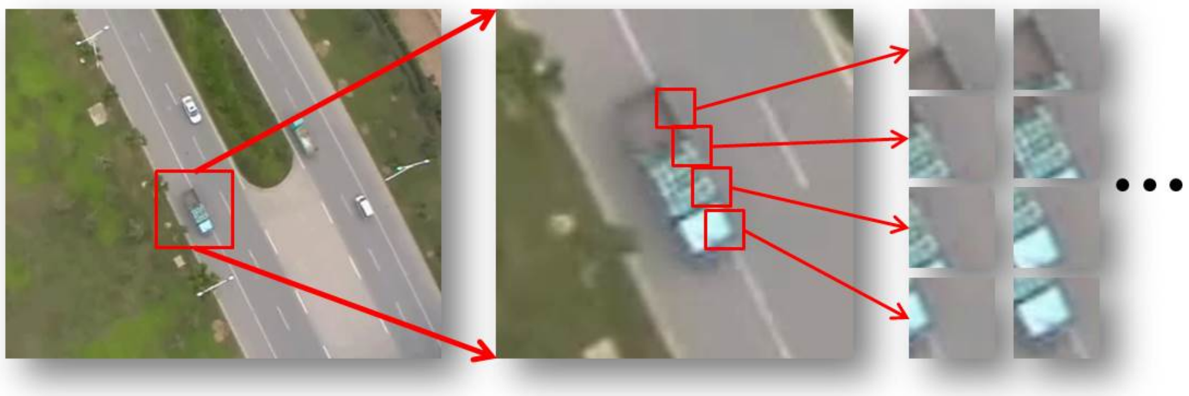


Figure 4.9: An example of a vocabulary codebook.

4.2.2 The Spatial Occurrence Distribution

After the generation and clustering of the codebook, the implicit shapes of vehicles were defined with different appearances that have consistent relevance to each other. The main purpose for using this method is that the codebook is flexible and similar testing images can be categorised in detection. After the first step, which is the SIFT point detection, lots of superimposed image patches were obtained for further detection. It is assumed that this would take longer time if all of these image patches have to processed, so the advantage of using implicit shapes is that the system can learn and classify from a relatively low number of samples, which somewhat reduces the processing time. Furthermore, the detection can made sure that if the candidate image patch contains any edges or corners of vehicles, the system is able to combine this information and self-predict the centre of the vehicle.

The model for clustering has fully covered all the possible shapes of vehicles; it also has a probability distribution for the shapes in the codebook which specifies the predicted positions of the centre of the vehicle. It is assumed that the distributions of the codebooks could be related to each other, the codebook objects were combined during detection. In order to predict the right direction of the centre of the vehicles, this method performs a further matching detection process on the descriptors around the interest points, where the descriptors will not only contrast with their nearest neighbours, but also with the entire codebooks' images. The matching results were weighted by the probability of the alignment and the position relative to the vehicle's centre. (Figure 4.10)

The SIFT and ISM detection process is carried out as by a loop. In the first step, the SIFT point detection was applied to the input images which gives the descriptors of the interest points. Note that, before the process begins, a SVM model of the SIFT points from the training samples has been created, ready for the next step of the process. In the next step, all the SIFT point descriptors extracted from the input image were put into the SVM model. The model will classify the SIFT points into positive and negative results. In this case, it only needs the positive results that are related to vehicles. The regions around the positive SIFT points are extracted for matching with the codebook in the following step; also note that the codebook for vehicles must be created beforehand, the same as the model. In this step, two processes that must be carried out; the first is the decision of the SVM model on the SIFT points and the second is how the regions around these positive SIFT points relate to the codebook database. In each of these two processes, the threshold plays a crucial role that will directly affect the detection results. A low thresholds of the SIFT matching process was used according to the testing, which need as many interest points as possible to avoid missing any important features in the beginning of detection. However, a slightly higher threshold was used for the codebook matching process that can achieve a better matching result. At the same time, it can also allow the algorithm to avoid matching errors, as some local features are very similar to each other, so stricter thresholds are needed. Once the matching processes are completed, the algorithm gives us the rough location of the centre of the vehicles in the input images. Once the hypothetical centres of the vehicles in the input image are evaluated, their distributions were also analysed. The Figure 5.10 shows the process diagram of this proposed detection approach.

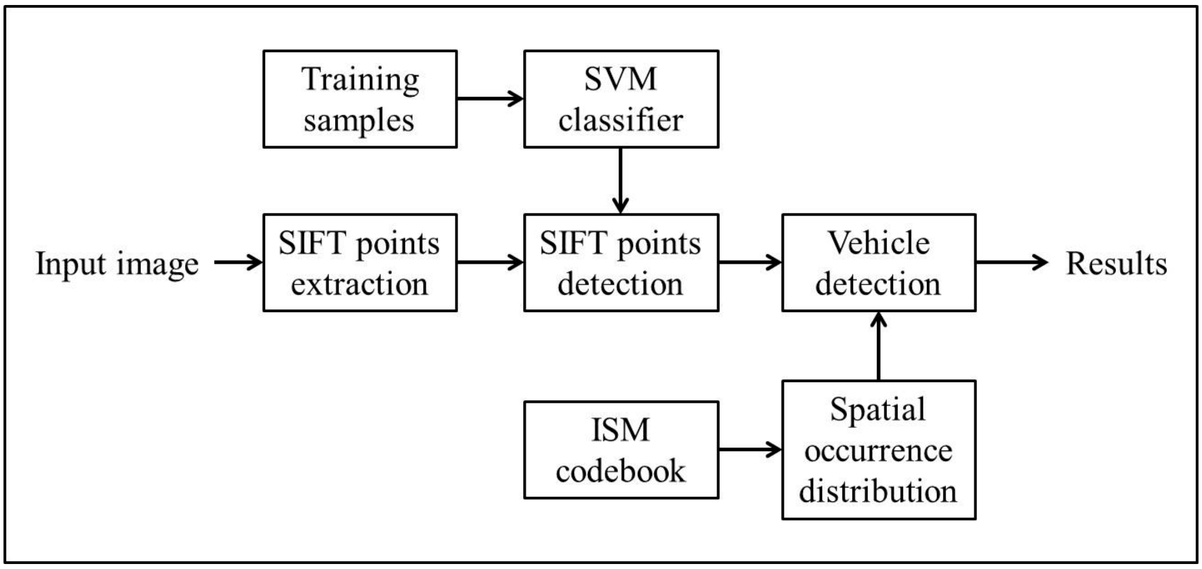


Figure 4.10: The diagram of the ISM-SIFT detection process.

4.3 Evaluation

As same as in the last chapter, the five testing videos containing different challenges were used in the testing of this detection approach (Figure 4.11). Testing was divided into separate steps. Firstly, detection using only SIFT was applied. Then the ISM and SIFT method was applied to compare the results. Detection performance was evaluated by the detection hypotheses matrix and the F-measure metric.

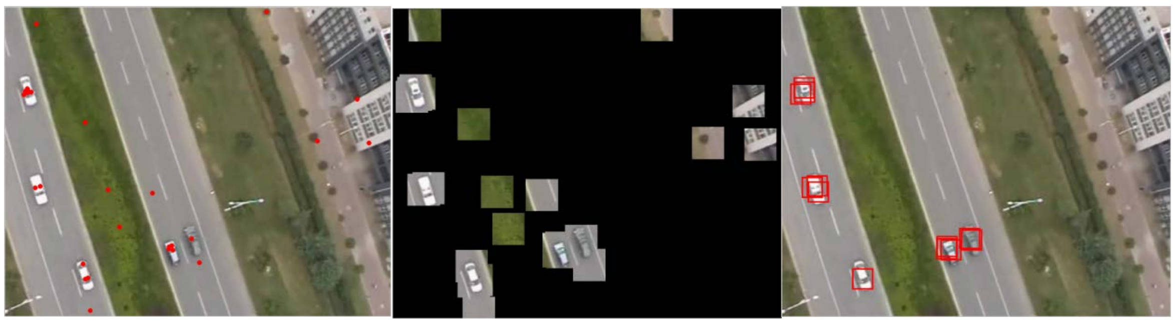


Figure 4.11: An example of the ISM-SIFT detection process. (Left: the SIFT point detection, Middle: codebook selections, Right: final detection)

Using this approach, vehicles can be detected based on interest point features and shape features. The proposed detection approach is invariant to image translation, scaling and rotation, and partially invariant even to illumination changes, which means it can tackle most vehicle detection challenges (Figure 4.12).

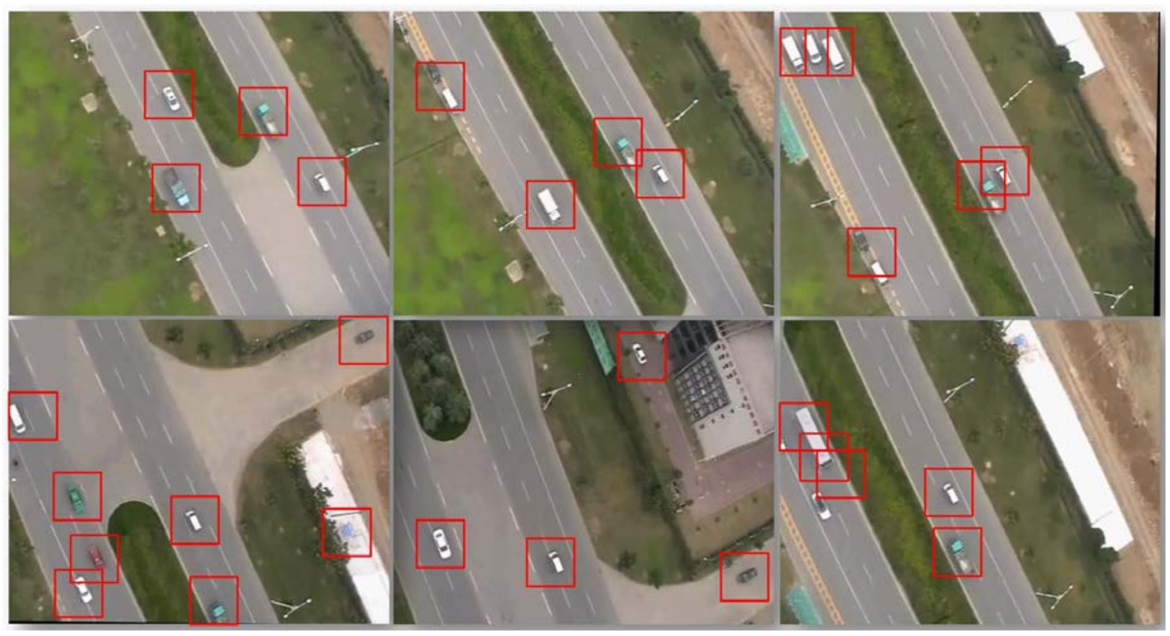


Figure 4.12: Examples of detection results using the ISM-SIFT detection approach.

The detection process was applied to each frame of each of the videos and the overall detection results were calculated. Table 4.3 shows the results of using the SIFT feature only. As the table shows, the detection achieved around 90% accuracy and F-measure, except Video 5, which has a higher accuracy. This is because, in Video 5, only one vehicle appears in each frame, so there is less noise to affect the detection process.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4863	802	461	91.34%	88.51%
Video 2	5511	4938	513	573	89.60%	90.09%
Video 3	5134	4722	489	412	91.98%	91.29%
Video 4	1848	1589	305	259	85.98%	84.93%
Video 5	1918	1886	55	32	98.33%	97.75%

Table 4.3: The results of detection using SIFT feature only.

Then the ISM and SIFT detection method was used on each testing video. Table 4.4 displays the results of using this method. The detection results are improved over those in the last table. The false positive values have been decreased by this method because of the ISM feature. The detection results have achieved above 90% in both accuracy and F-measure.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	5142	411	182	96.58%	94.55%
Video 2	5511	5212	336	299	94.57%	94.26%
Video 3	5134	4837	316	297	94.22%	94.04%
Video 4	1848	1699	223	149	91.94%	90.13%
Video 5	1918	1893	47	25	98.70%	98.13%

Table 4.4: Results of detection using ISM+SIFT feature.

The bar chart in Figure 4.13 shows a general view of the detection results of using both methods.

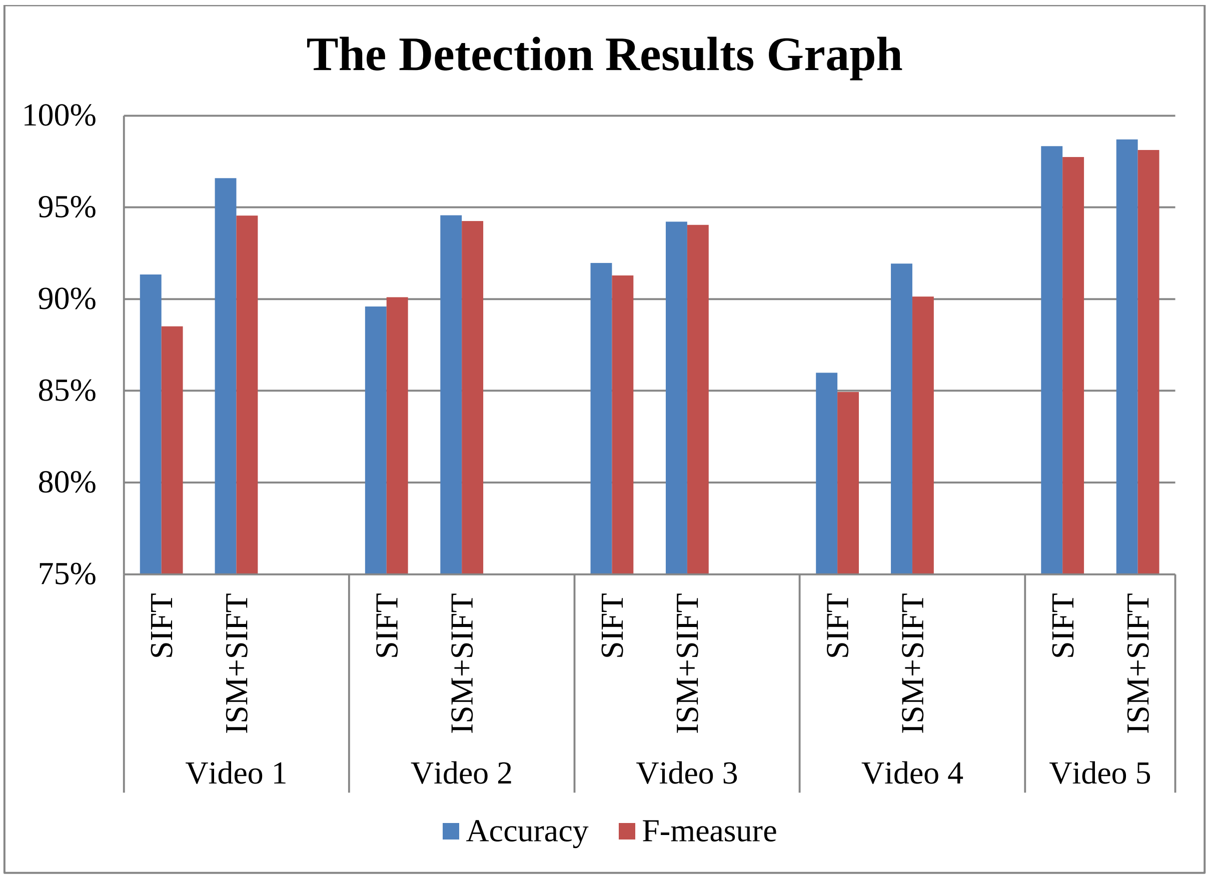


Figure 4.13: A bar chart of all detection results.

4.4 Conclusions

This chapter has proposed a novel detection approach of ISM-SIFT. The proposed approach has been explained in details in this chapter, which it used combinations of point with shape features to detection the vehicles. The main contribution of the proposed approach is to solve the challenges of different size, shape and orientation of the target vehicles during the detection. The point feature detection has the benefits of scale invariant, rotate invariant and affine distortion invariant then the region texture features and it also has the benefit of insensitive to the blur problem. This can be concluded in the detection accuracy result comparison with the HSV-GLCM approach in the previous chapter. The detection results in in the video 4, which is the blurring problem taking place, are 88.91% vs 91.94% in the final approach, and also in the video 5, which is the changing appearance and size challenge, the detection result is higher than others in 98.22% and 98.70%. The average detection accuracy in video 1, 2 and 3 is around 94% in both ISM-SIFT and HSV-GLCM approach. This can be conclude that the proposed method could handle the complex background, occlusion and blocked vehicle challenges.

However, there are few points that still have to be discussed. First of all, the false positive values given by the SIFT method were very high, which is because the SIFT values of the background are similar to those of vehicles. Adding the ISM feature meant there was one more filter which could detect the shape of the object and predict the centre, so the false positives were decreased significantly. Moreover, both the ISM and SIFT method require high computational resources leading to an inefficient process. In order to increase the performance speed and efficiency, a novel FAST and HoG detection approach will be discussed in the following chapter.

Chapter 5

The Proposed FAST and HoG Detection System

In this chapter, the FAST and HoG features for vehicle detection systems will be described. The proposed detection system that uses these features comprises three stages: region of interest selection, classification and detection. The vehicle detection system is initialised with the region of interest selection process, which attempts to identify the regions of the image that are more likely to have vehicles. This process uses the FAST corner detector and a density estimation technique to identify the regions with high-density corner points. Then the classification stage categorises the regions of interest into vehicle and environment through the extraction of the Histogram of Oriented Gradients (HoG) feature. The classification stage uses Support Vector Machines (SVM) to create a classification model using training samples. Finally, the detection process attempts to locate the vehicles in the image from the overlapping detected windows.

5.1 Region of Interest Selection (ROI)

Performing an entire search over all image sections on multiple scales is very computationally expensive and often excessive, as vehicles are more likely to occupy a small area in the image. Therefore, selecting regions of interest prior to classification is an important stage in the development of a real-time system. The selection of regions of interest depends on the observation of man-made objects, specifically vehicles, which have a large number of edges and corners in a certain area compared to other background objects such as roads, trees and buildings. With this observation, the algorithm uses an efficient sliding window technique to select regions in the images with a high density of corner features, and consequently selects the regions that are most likely to contain vehicles. To detect the corner features, this detection method proposes the use of the Feature from Accelerated Segment Test (FAST) corner detector. The reason for use FAST is that it has a similar corner response to other features, such as the Harris corner detector [130], but it processes faster.

5.1.1 Feature from Accelerated Segment Test (FAST)

The Feature from Accelerated Segment Test (FAST) detector was developed by Rosten and Drummond [131] and is based in principle on the SUSAM corner detector [132]. The FAST detector classifies a pixel ***p*** as a corner by performing simple brightness tests on a discretised circle of 16 pixels around ***p***. The basic method is shown in Figure 5.1. If there are 12 contiguous pixels around pixel ***p*** with intensities that are all brighter or darker than the centre pixel ***p*** by a threshold ***t***, pixel ***p*** is detected as corner. For this test condition to be satisfied, three of the four pixels at circle positions 1, 5, 9 and 13 must have intensity above or below the intensity of ***p*** by threshold ***t***. This allows the test process to be optimised by testing these four pixels first before examining all pixels in the circle.

In contrast to the Harris corner detector, the FAST detector does not compute a corner response function. Therefore, to perform non-maximal suppression the following score function must be evaluated for each candidate corner:

$$\text{Score}\ (p) = \text{MAX}\left(\sum_{q\in S_{Bright}} \left|I_q - I_p\right| - t,\ \sum_{q\in S_{dark}} \left|I_p - I_q\right| - t\right) \qquad (5.1)\ [131]$$

where $\boldsymbol{S_{bright}}$ is the subset of pixels in the circle that are brighter than ***p*** by threshold ***t*** and $\boldsymbol{S_{dark}}$ is the subset of pixels that are darker than ***p*** by ***t***.

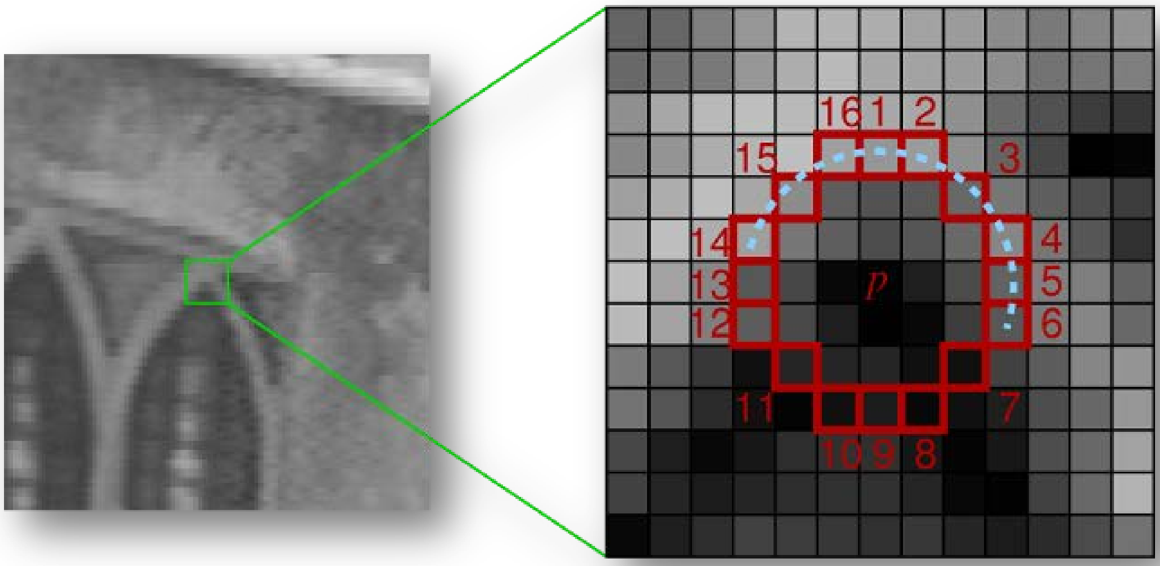


Figure 5.1: The FAST corner detector; a discretised circle of 16 pixels around pixel ***p***.

With different thresholds ***t*** applied to the FAST detection, the detector gives us different corner detection results. If the threshold ***t*** is high, fewer corners are detected and less computational process time is needed; otherwise, more corners are detected and more computational power are required (Figure 5.2). During detection, there is no need to get too much corner information. Thus, in this approach, the threshold for the FAST corner detector is set to 30 (Figure 5.3) which detects most corner features are needed. The threshold was tested on the training videos and it satisfied the requirement.

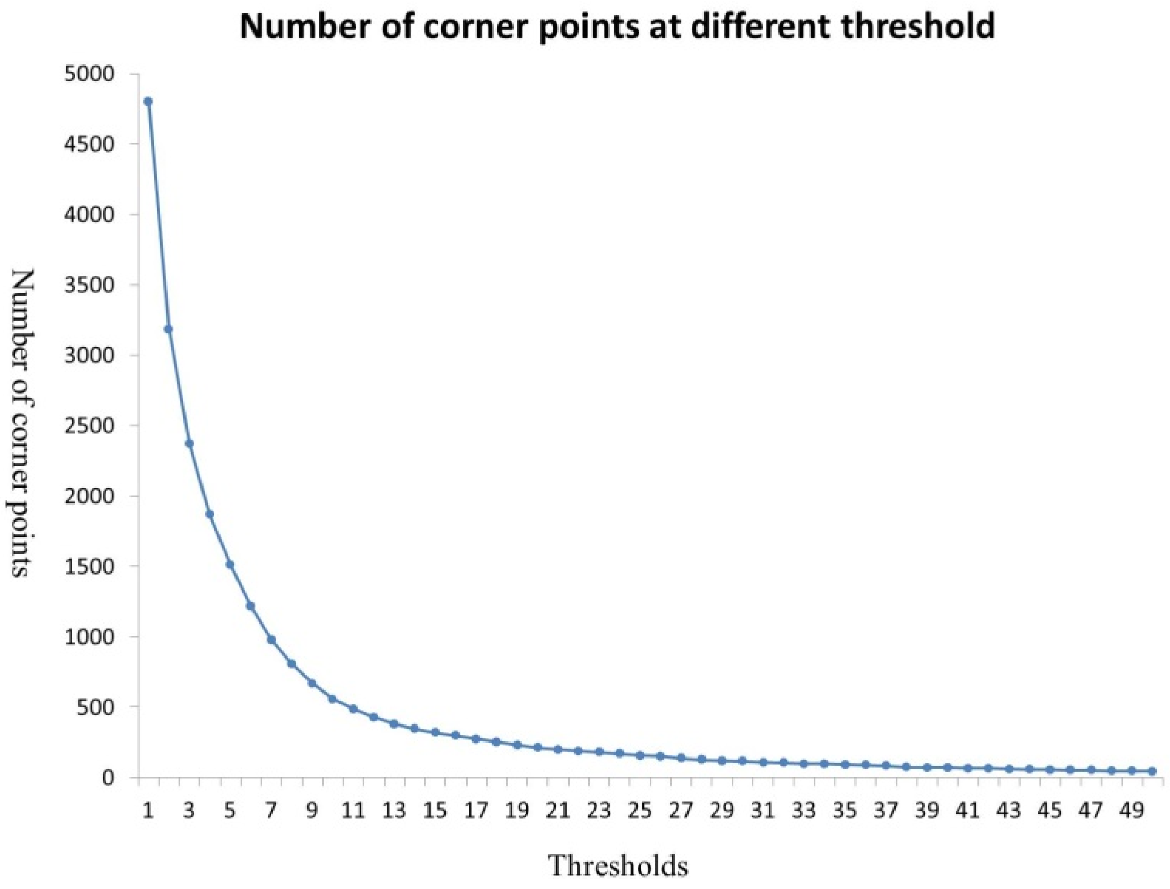


Figure 5.2: The number of corner points at different FAST thresholds.

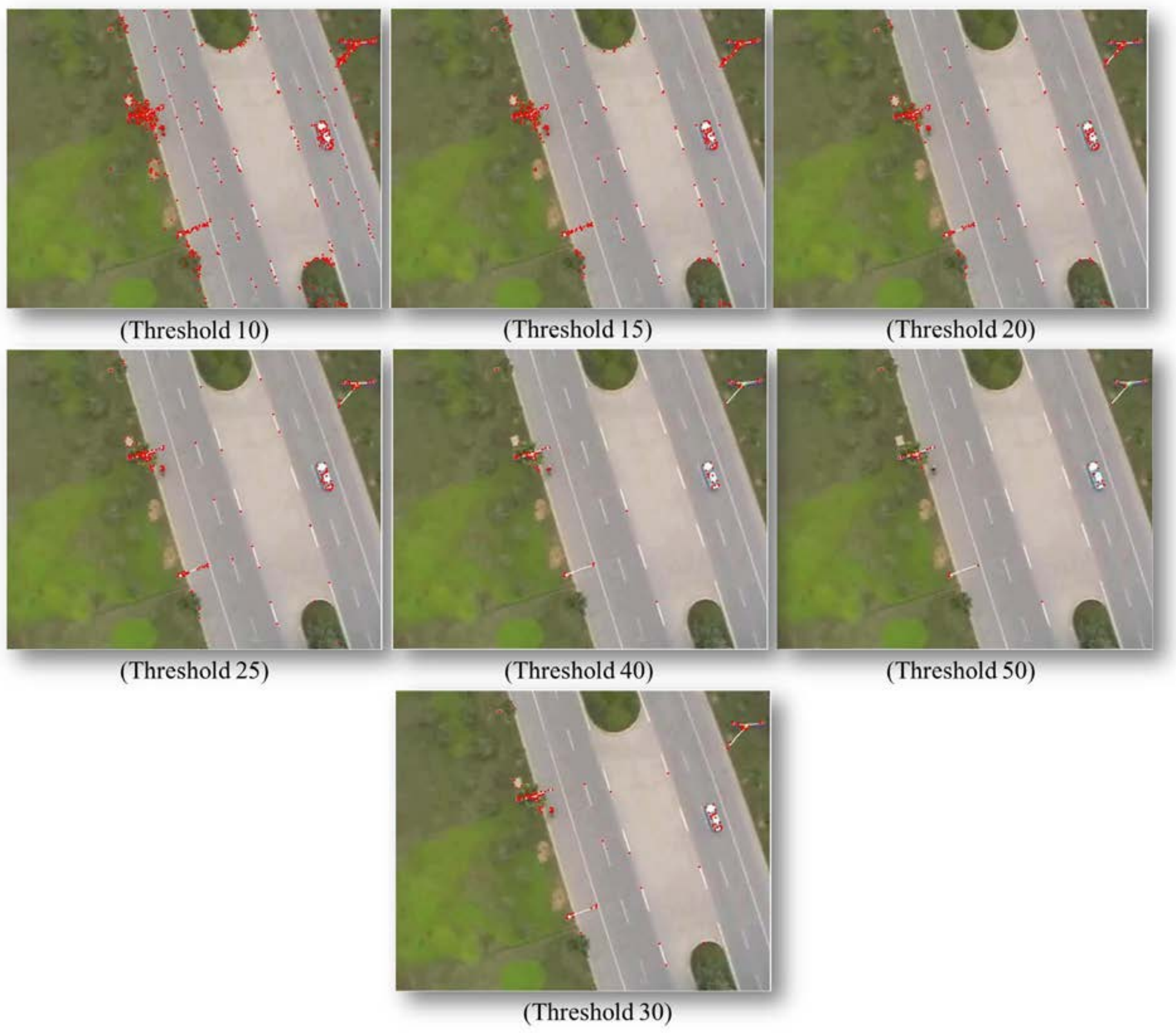


Figure 5.3: FAST corner detection with different thresholds t.

5.1.2 Feature Density Estimation

Having detected the FAST corner features, the next stage of the algorithm is to identify regions within the image that have a high concentration of these features. This is achieved by sliding a window over every location in the test image and selecting the windows that have a feature density greater than a certain threshold. The density maps are shown in Figure 5.4 which gives us a general idea of the corner point density distributions.

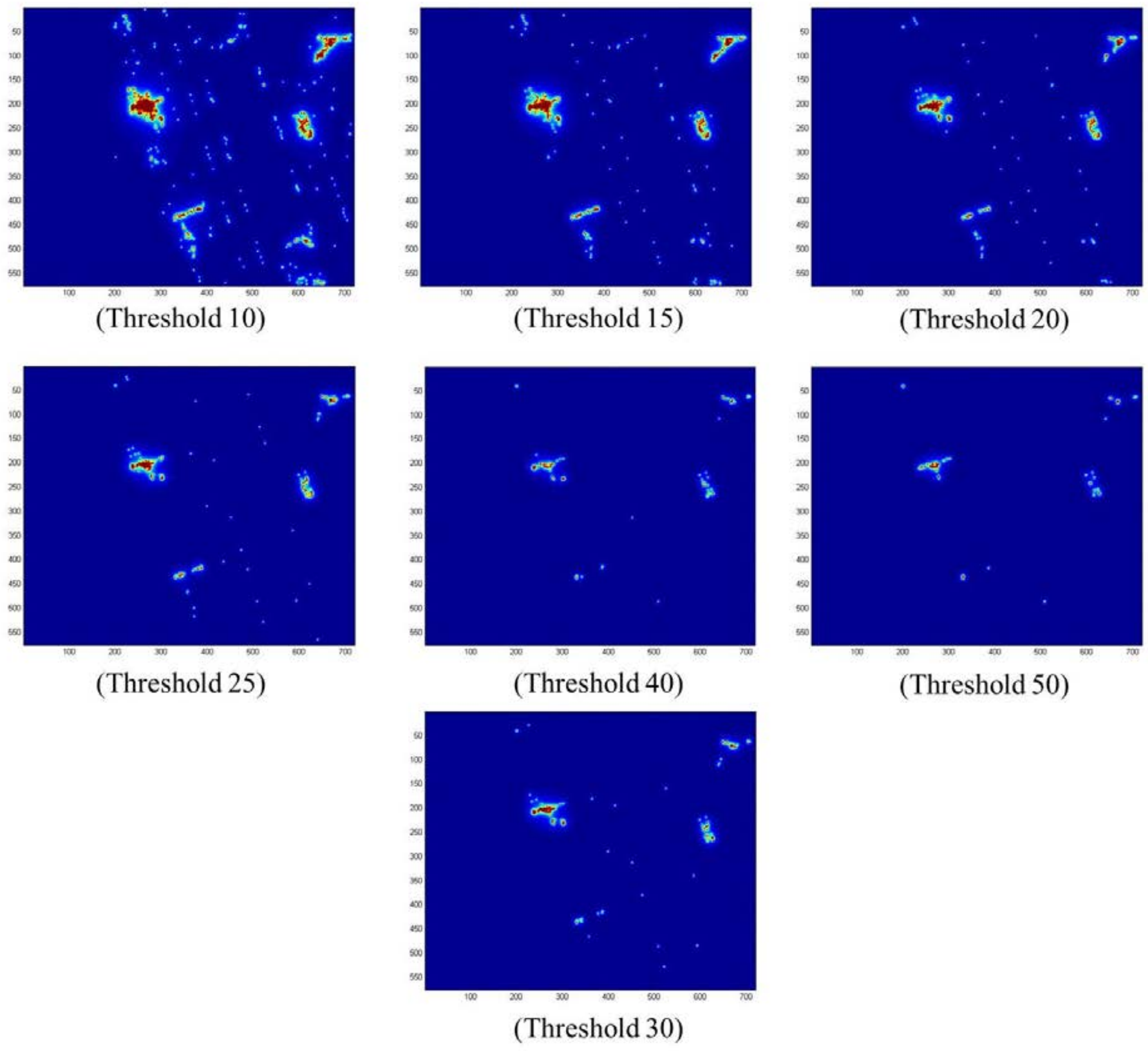


Figure 5.4: FAST corner density at different thresholds $\boldsymbol{t}$.

Given a window with whose top left corner is $(\boldsymbol{x}, \boldsymbol{y})$ with width $\boldsymbol{w}$ and height $\boldsymbol{h}$, the feature density for this window is calculated by the score function:

$$\text{Score}\ (x, y, w, h) = \frac{S_{x,y,w,h}}{w \times h} \tag{5.2}$$

where $S_{x,y,w,h}$ is the number of features detected within the window. By sliding the window over the image at multiple scales and normalising $S_{x,y,w,h}$ by the area of the window, this allows to build a scale-invariant detector with the ability to detect vehicles at multiple scales in the image.

The efficient computation of the number of features in a window is determined using integral images. Given a set of FAST corners for a query image, a corner image $\mathrm{i}(u,v)$ can be created where the value at any point is 1 or 0, indicating the presence or absence of a corner respectively. For this corner image, the corresponding integral image $\mathrm{I}(i,j)$ is the image, where each point (i,j) is computed as the sum of all the corners above and to the left of (i,j) inclusive:

$$\mathrm{I}(i,j) = \sum_{u\le i, v\le j} i(u,v) \tag{5.3}$$

This image can be computed efficiently in a single pass over the corner image. Once computed this image allows for the evaluation of any window in constant time with only four lookups, where by the number of features in the window is computed as:

$$S_{x,y,w,h} = I(x+w, y+h) + I(x,y) - I(x+w,y) - I(x,y+h) \tag{5.4}$$

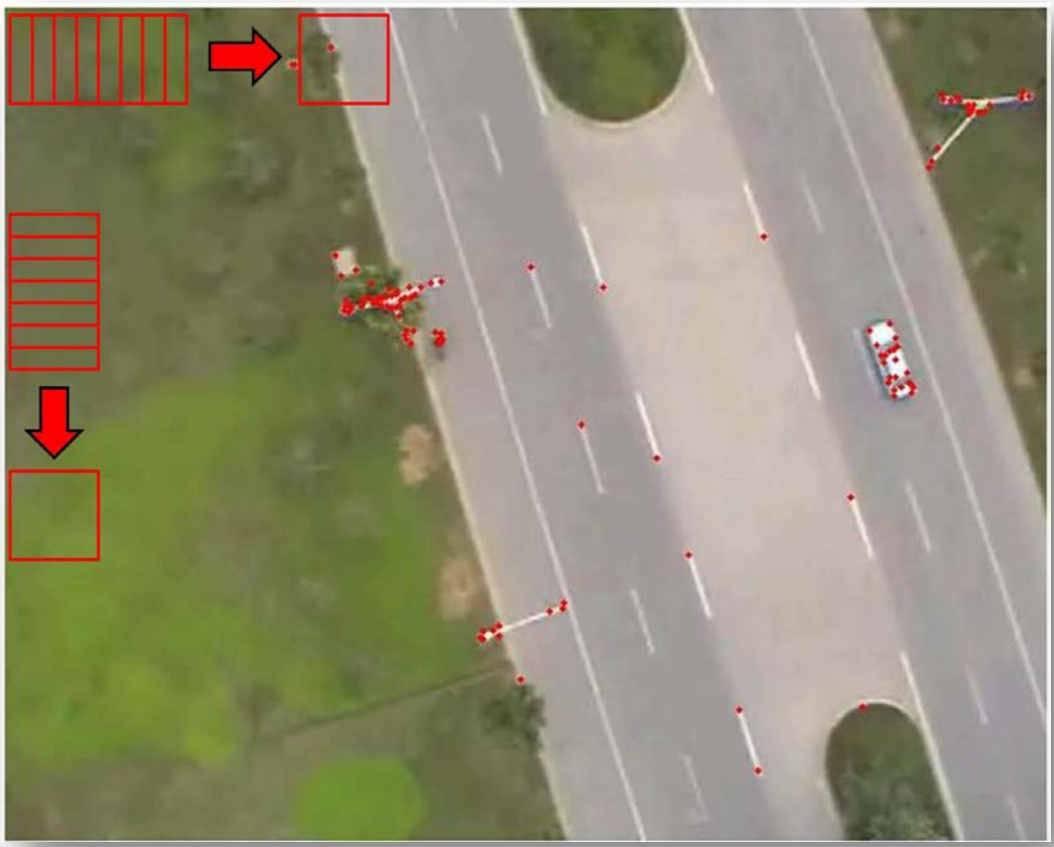


Figure 5.5: Feature density estimation sliding windows.

5.2 Feature Extraction

Feature extraction is a special form of dimensionality reduction; it is the process of transforming an image into a reduced representation that describes the most informative features in the image. In the proposed system, feature extraction is used to transform the image patches selected by region of interest detection into a representation that can be presented to a classifier. The representation must be invariant to changes in scale and rotation to enable vehicles to be recognised irrespective of their size and orientation in the image. In addition, this representation must also be invariant to illumination changes, colour and motion blur, as well as intra-class variations and partial occlusions. These properties are necessary to ensure that the classifier generalises well and is capable of identifying a diverse variety of vehicles under a wide range of conditions. Furthermore, the representation also needs to be sufficiently distinctive in order to maximise the classification accuracy, and due to the real-time requirement of the system it must also be computationally fast.

Humans are very good at recognising vehicles in aerial imagery, for this reason Zhao et al. [133] carried out a series of psychological tests to find out what features of vehicles humans use to allow us to make the decision about the presence of a vehicle. They found that the two most significant features were the rectangular shape and the presence and position of the frontal and rear windshields. Shape is recognised by the presence and position of edges, for a vehicle these edges consist of four primary edges that identify the outline of the vehicle and a set of secondary edges for the windshields. Therefore, to capture this information a feature descriptor is required, which is capable of encoding this edge information.

The idea behind the Histogram of Oriented Gradients (HoG) descriptor is that the shape of objects can often be well described by the distribution of edge directions, even without precise information on the edges themselves. This makes it a good choice for vehicle detection, because edges on vehicles can generally be grouped in two major edge directions; the direction of the sides and the direction of the back, front and windshield edges. Furthermore, these edge orientations are largely perpendicular and therefore this gives a common distribution of edge directions among vehicles. In addition, the HoG descriptor is advantageous as it is relatively invariant to the geometric and photometric changes described above. A weakness of the HoG descriptor is that it is not rotationally invariant; however, this functionality is provided by the classification approach. The remainder of this section describes the theoretical aspects and the problem specific formulation of the HoG feature vector.

5.2.1 Histogram of Oriented Gradients (HoG)

The Histogram and Oriented Gradients (HoG) feature was developed by Dalal and Triggs [33] and was originally proposed for the task of human detection. The main principle of HoG descriptors is the identification of the appearance and shape of objects in an image by using the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small regions called cells; for each cell, a histogram of gradient directions or edge orientations for the pixels within that cell is compiled. The descriptor represents the combination of these histograms. To improve the performance of the HoG descriptor, the local histogram can be contrast-normalised by calculating the intensity across the region of the image (called a block), and then using this value to normalise all cells within the block. This normalisation results in better invariance to changes in illumination or shadowing. The extraction of a HoG feature vector from a detection window is composed of five steps:

- Normalise gamma and colour
- Compute gradients
- Weighted vote into spatial and orientation cells
- Contrast normalise overlapping spatial blocks
- Collect HoG's over detection window

After colour and gamma normalisation, edges are detected by convolving the image patch with the simple 1D $[-1,0,1]$ mask both horizontally and vertically. Specifically, this method requires filtering the greyscale image with the filter kernels:

$$D_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \; and \; D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \qquad (5.5)\ [33]$$

In the second step the image patch is subdivided into rectangular regions called cells. The gradient for each pixel within each cell were computed. In colour images the gradient is computed separately for each channel and the largest gradient is chosen for the gradient for that pixel. So, being given an image I, the x and y derivatives were obtained by using a convolution operation:

$$I_x = I \times D_x \ and \ I_y = I \times D_y$$ (5.6) [33]

The magnitude of the gradient is:

$$|G| = \sqrt{I_x^2 + I_y^2}$$ (5.7) [33]

The orientation of the gradient is given by:

$$\theta = \arctan \frac{I_y}{I_x}$$ (5.8) [33]

In the next step, each pixel within the cell then computes a weighted vote for the orientation of the cell, where the vote is weighted by the gradient magnitude (i.e. the L2 norm). These votes are accumulated into orientation bins; a vote is cast into the closest bin in the range 0 to 180 degrees or 0 to 360 degrees, depending on whether it is unsigned or signed gradient. In this algorithm, the gradient is unsigned, so the range is 0 to 180 degrees. These gradients are stored in a histogram. Dalal and Triggs found that using a conjunction with nine channels in the histogram for unsigned gradients can lead to better performance by the algorithm.

In the penultimate step, local contrast normalisation is used to suppress the effects of changes in illumination and contrast with the background on the gradient magnitude. This stage was found to be essential for good performance and is achieved by grouping cells into large blocks and normalising within these blocks, ensuring that low-contrast regions are stretched. In addition, to ensure consistency across the image patch but still keep local variations, overlapping blocks can be used. The HoG descriptor is then the vector of the components of the normalised cell histograms from all of the block regions.

There are different methods for block normalisation. Let v be the non-normalised vector containing all histograms in a given block, $||v_k||$be its k-norm for k = 1, 2 and e be the constant. The normalisation factor L2-nor:

$$f = \frac{v}{\sqrt{||v||_2^2 + e^2}}$$ (5.9) [33]

Finally, the normalised orientation histograms for each cell are collected together and result in a b $\times$ $c_x \times c_y$–dimensional feature vector where the number of orientation bins is, $c_x \times c_y$is the number of image cells.

The HoG approach provides a reasonably flexible descriptor that can be tuned for different applications. Naturally this leads to a range of choices that have to be made, such as cell size, block size and number of orientation bins, which will be discuss in the next section, on classification.

5.3 Classification

The HoG feature vectors extracted from the Regions of Interests (RoI) are set to a binary classifier which determines the presence of a vehicle in the image patch. As mentioned, the HoG features need to set a range for cell size, block size and number of orientation bins. It is also noticed that the HoG features are not rotationally invariant in the previous section. As a result, the parameters for the HoG feature were evaluated in order to achieve better detection performance.

First of all, the HoG features are not rotationally invariant, therefore to facilitate the detection of vehicles at all orientations, this functionality must be provided at the classification stage. The conventional method to achieve this is to train one classifier on images of vehicles at a single “norm” orientation and then evaluate each region of interest at multiple orientations by rotating the detection window. However, the drawback of this method is that the HoG features have to be recomputed for each rotation, which is computationally expensive. Instead, four separate Support Vector Machine (SVM) models are trained on sample vehicles images that are categorised into one of four angular offsets to horizontal, which are 0°, 45°, 90° and 135° (Figure 5.6). These four SVM models are then combined to construct a single classifier that evaluates a rotationally invariant response for a single HoG feature vector.

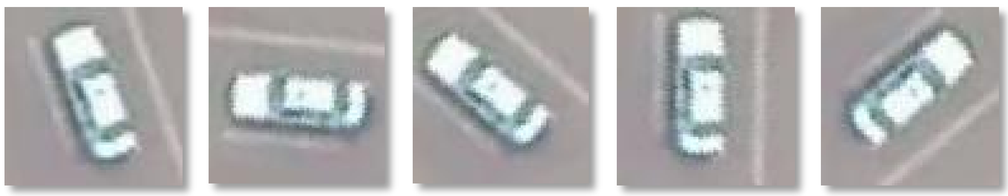


Figure 5.6: Original sample, 0° orientation, 45° orientation, 90° orientation, 135° orientation.

The parameters of HoG include the size of HoG cell, the number of cells in a block, the number of overlapping cells between adjacent blocks and the number of orientation histogram bins.

Cell size is specified in pixels as a two-element vector. For large-scale object feature extraction a larger cell size is needed, but when the cell size is increased, small-scale detail might be lost. Block size is the number of cells in one block. A small block size helps to capture the significance of local pixels; it can also help suppress illumination changes to HoG features. The number of orientation histogram bins is specified as a positive scalar. To encode finer orientation details, the number of bins should be increased. Increasing this value increases the size of the feature vector, which requires more time to process.

The training and detection window of this algorithm is 70×70 pixels. So the best combination of cell size and block size will be divisible by 70. The testing was categorised into six groups by cell size, and for each group different block size were extracted (Table 5.1). In each combination of cell size and block size, the number of orientation histogram bins was set to nine, which provided a reasonably low dimensional feature vector that delivered good descriptive power and resulted in better classification accuracy.

Cell size (pixel)	**Block Size (Cell)**				
35 × 35	1 × 1	2 × 2			
14 × 14	1 × 1	2 × 2	3 × 3	4 × 4	5 × 5
10 × 10	1 × 1	2 × 2	3 × 3	5 × 5	7 × 7
7 × 7	1 × 1	2 × 2	4 × 4	7 × 7	10 × 10
5 × 5	1 × 1	2 × 2	3 × 3	4 × 4	14 × 14
2 × 2	1 × 1	2 × 2	7 × 7	14 × 14	35 × 35

Table 5.1: Cell size and block size table.

The HoG features were extracted from the training samples with different parameters to get HoG descriptors. The training samples were grouped into positive samples and negative samples (Figure 5.7). The positive samples selected were basic 70×70 pixel windows containing a single vehicle. Additionally, 10 samples were selected from different frames in the video for each vehicle so it could obtain training sample features in different conditions. The negative samples were selected from the environment objects in the videos, such as road markings, street lights, buildings and plants. Because the HoG feature is a texture-based detection method, the samples should be selected from featured objects rather than coloured objects, unlike GLCM feature training samples. The sizes of the HoG feature descriptors vary depending on the cell size and block size. These descriptors of both positive and negative samples were applied to the SVM classifier to create the classification model, and then the classification performances of each model were evaluated. The most accurate model was used as the HoG parameter setting for the detection system.

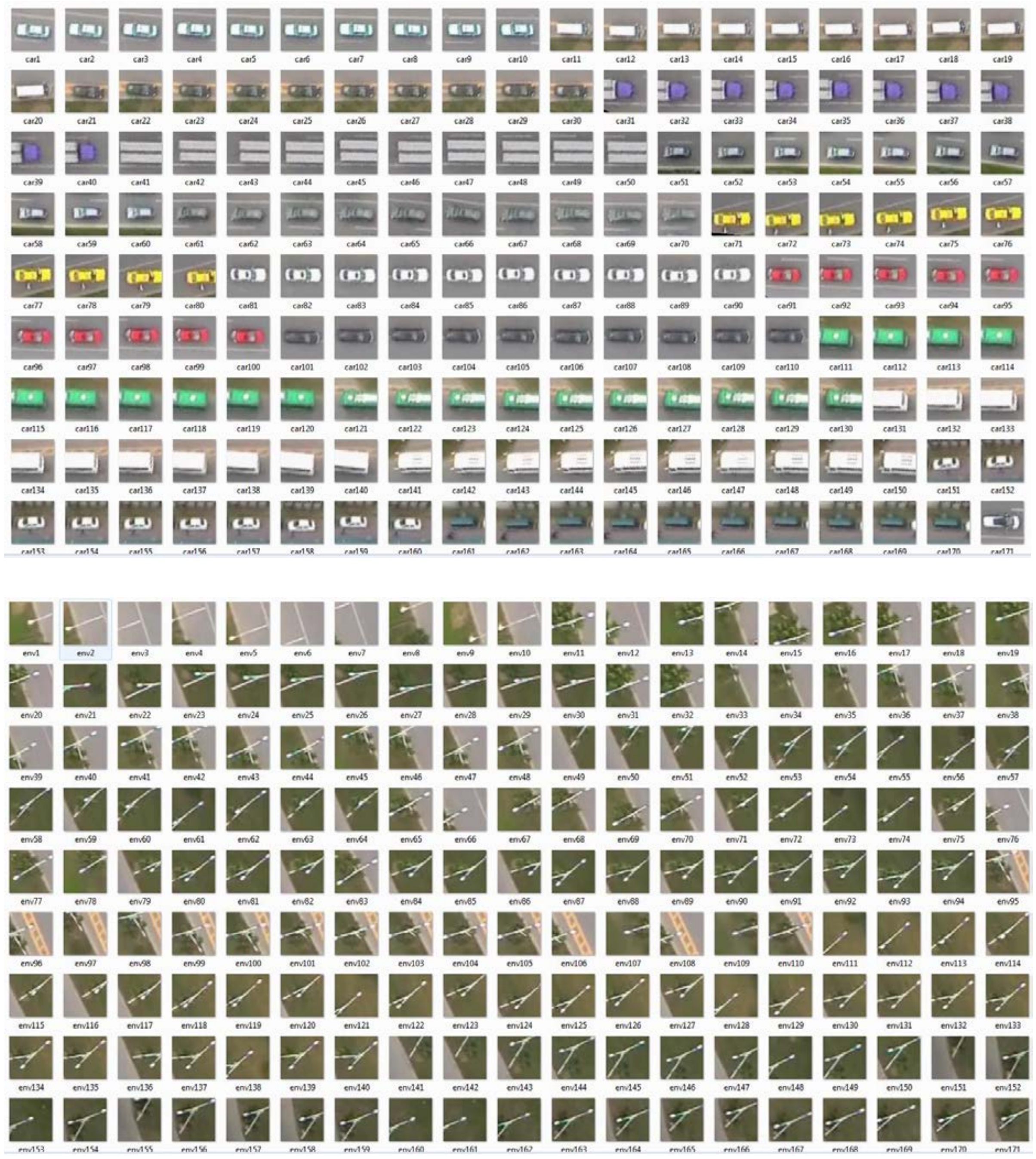


Figure 5.7: Examples of positive samples (top) and negative samples (bottom).

There were 2,100 positive samples (labelled as 1) and 1,810 negative samples (labelled as 0) selected. The classification models calculated by the SVM are based on these samples. To evaluate the models' performance, all the training data were applied to the SVM model, which gives a prediction label for each training sample.

The following figures (5.8 to 5.13) show HoG descriptor visualisations and the histogram charts for a single training sample (Figure 5.7) which gives a general idea of how the HoG feature applied during the detection. The tables (5.2 to 5.7) display the descriptor size and the evaluation results of the classification model with different parameters (cell size and block size).

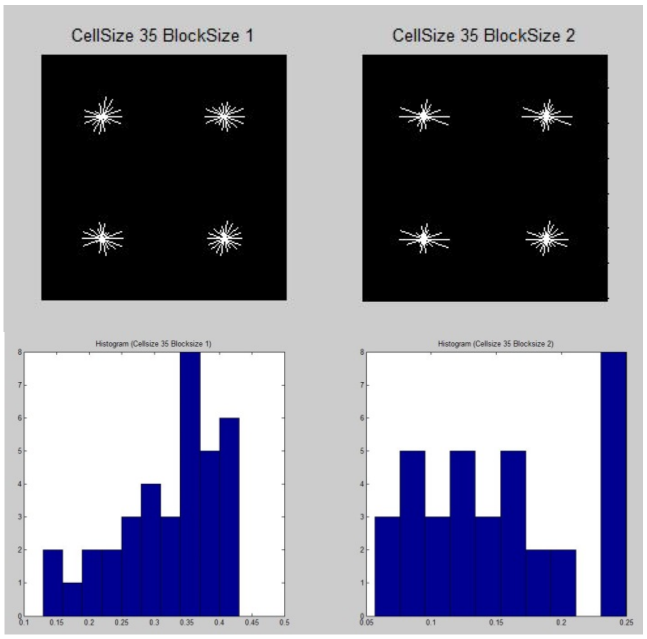


Figure 5.8: HoG descriptors and histogram values of cell size group 35 × 35.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
35 × 35	1 × 1	36	89.57	11.77	88.23	10.43
35 × 35	2 × 2	36	93.48	18.84	81.16	6.52

Table 5.2: Classification accuracy for cell size group 35 × 35 pixels.

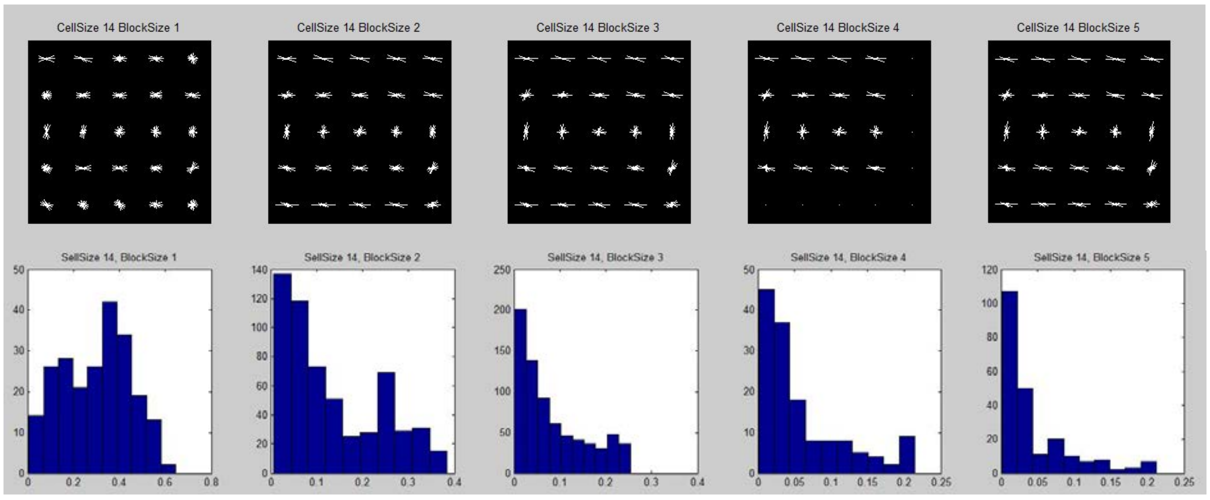


Figure 5.9: HoG descriptors and histogram values of cell size group 14 × 14 pixels.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
14 × 14	1 × 1	225	97.05	5.64	94.36	2.95
14 × 14	2 × 2	576	99.19	2.10	97.90	0.81
14 × 14	3 × 3	729	98.95	2.76	97.24	1.05
14 × 14	4 × 4	144	94.90	8.51	91.49	5.10
14 × 14	5 × 5	225	96.00	7.46	92.54	4.00

Table 5.3: Classification accuracy for cell size group 14 × 14 pixels.

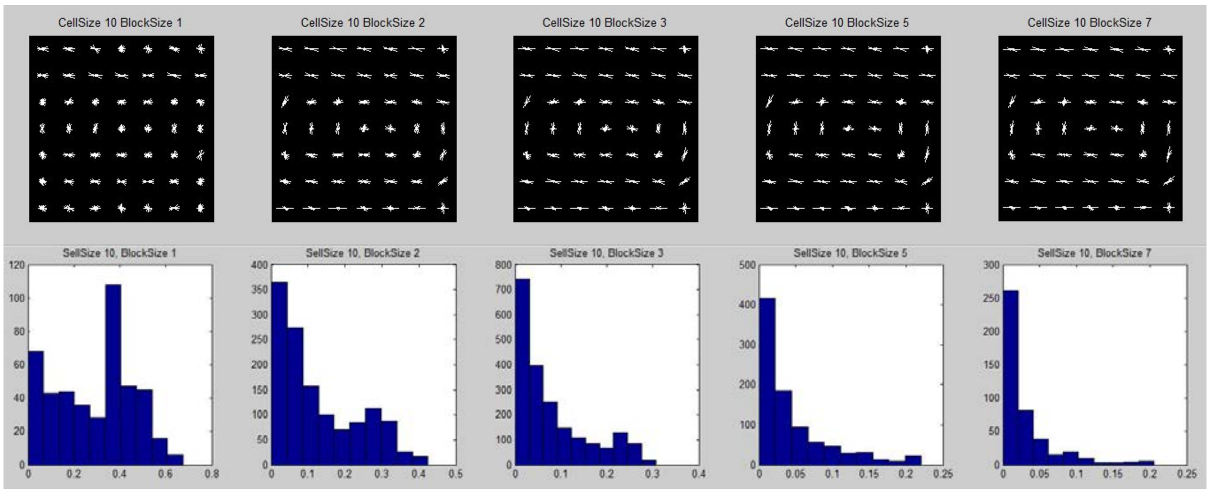


Figure 5.10: HoG descriptors and histogram values of cell size group 10 × 10 pixels.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
10 × 10	1 × 1	441	98.95	2.98	97.02	1.05
10 × 10	2 × 2	1296	99.90	0.55	99.45	0.10
10 × 10	3 × 3	2025	99.90	0.61	99.39	0.10
10 × 10	5 × 5	900	97.43	3.70	96.30	2.57
10 × 10	7 × 7	441	96.19	5.52	94.48	3.81

Table 5.4: Classification accuracy for cell size group 10 × 10 pixels.

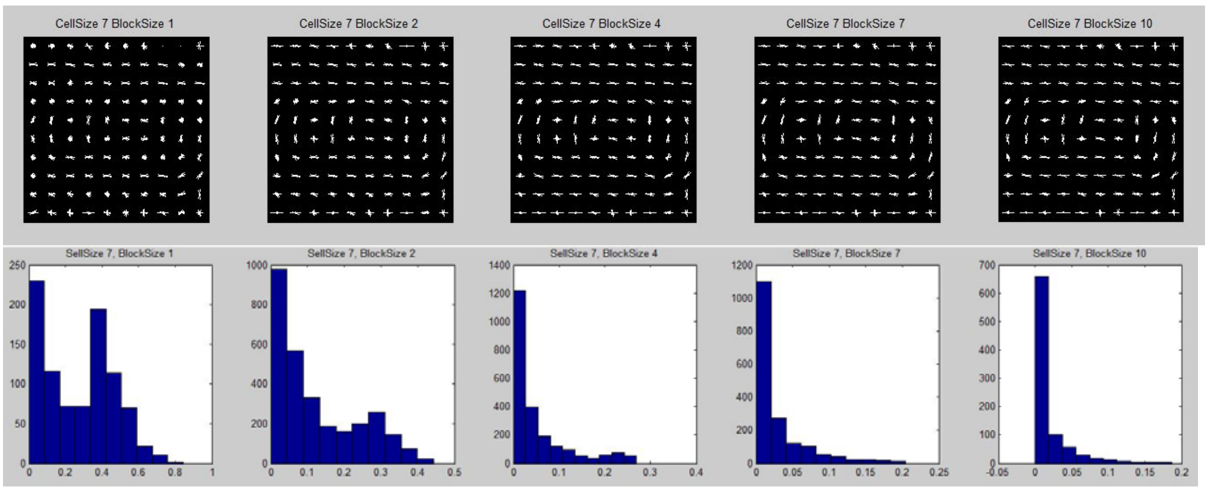


Figure 5.11: HoG descriptors and histogram values of cell size group 7 × 7 pixels.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
7 × 7	1 × 1	900	100	0.17	99.83	0
7 × 7	2 × 2	2916	100	0	100	0
7 × 7	4 × 4	2304	100	0.39	99.61	0
7 × 7	7 × 7	1769	98.43	2.65	97.35	1.57
7 × 7	10 × 10	900	97.05	4.03	95.97	2.95

Table 5.5: Classification accuracy for cell size group 7 × 7 pixels.

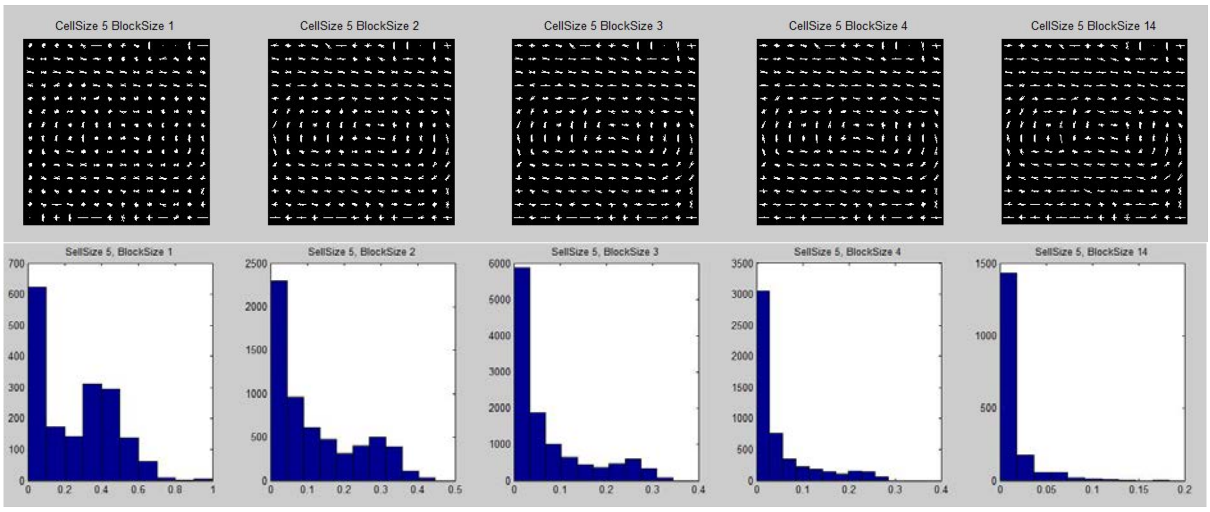


Figure 5.12: HoG descriptors and histogram values of cell size group 5 × 5 pixels.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
5 × 5	1 × 1	1764	100	0	100	0
5 × 5	2 × 2	6084	100	0	100	0
5 × 5	3 × 3	11664	100	0	100	0
5 × 5	4 × 4	5184	100	0	100	0
5 × 5	14 × 14	1764	97.62	2.87	97.13	2.38

Table 5.6: Classification accuracy for cell size group 5 × 5 pixels.

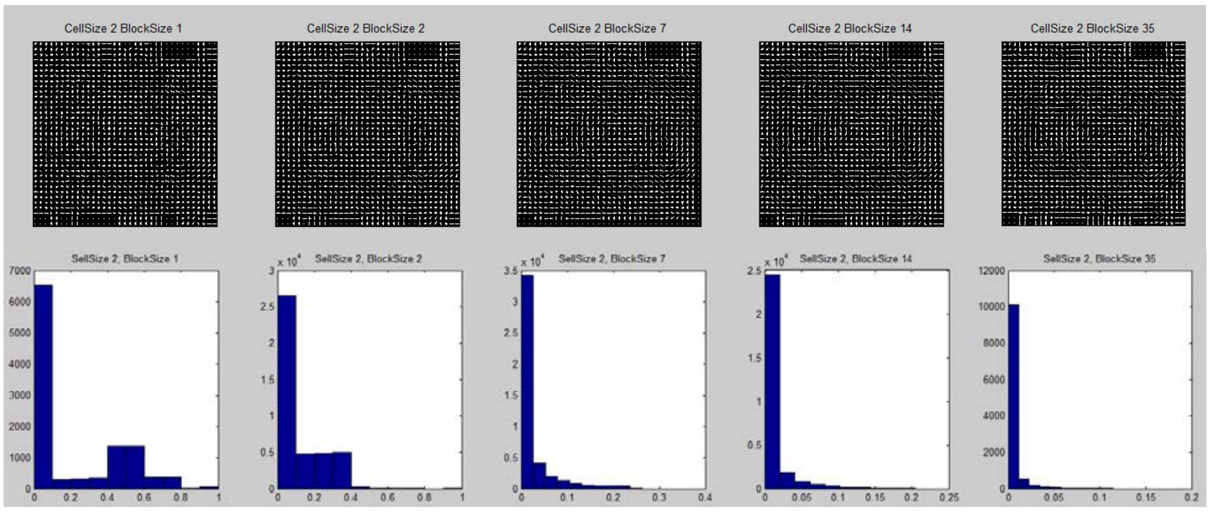


Figure 5.13: HoG descriptors and histogram values of cell size group 2 × 2 pixels.

Cell Size (Pixel)	Block Size (Cell)	Number of descriptors	TPR (%)	FPR (%)	TNR (%)	FNR (%)
2 × 2	1 × 1	11025	100	0	100	0
2 × 2	2 × 2	41616	100	0	100	0
2 × 2	7 × 7	44100	100	0	100	0
2 × 2	14 × 14	28224	100	0	100	0
2 × 2	35 × 35	11025	100	0	100	0

Table 5.7: Classification accuracy for cell size group 2 × 2 pixels.

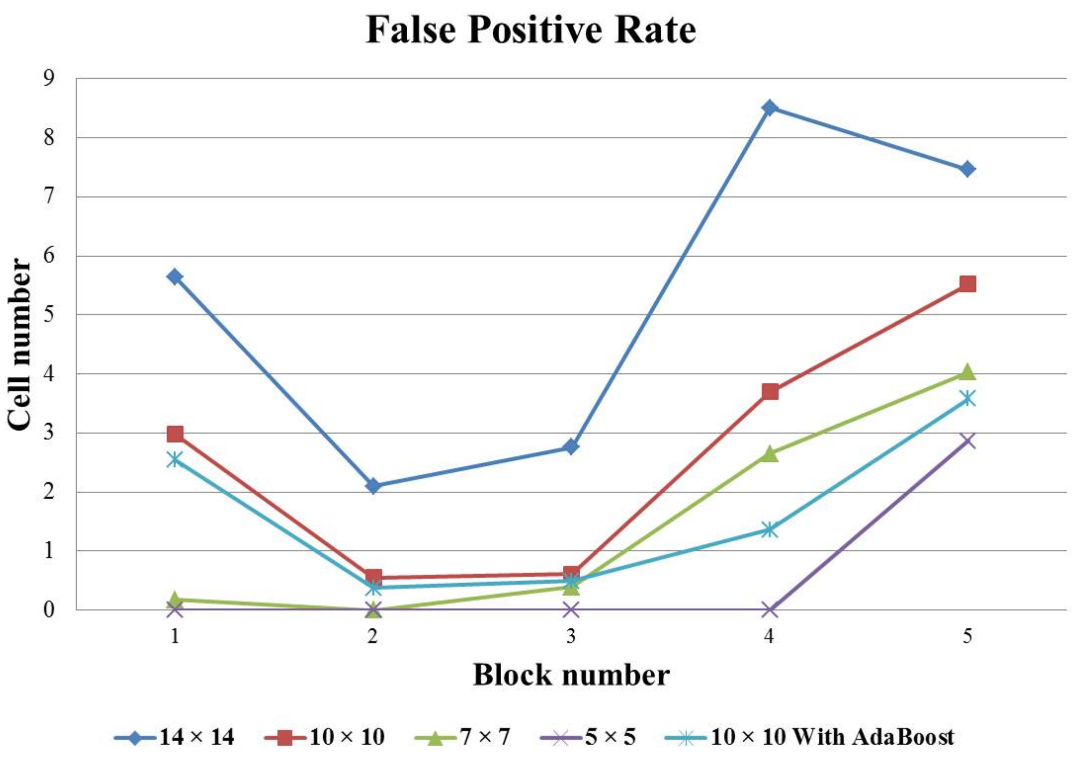


Figure 5.14: False positive rate for different cell size groups.

According to the tables (5.2 to 5.7) and figures (5.7 to 5.14) above, it can conclude that a small cell size gives more HoG feature information, which can achieve greater classification model accuracy. It is also noticed that, as the number of HoG descriptors increases, more computational resources are needed, which entails a longer processing time. The False Positive Rate (FPR) figure for all of the groups to compare has shown which combination is the best for detection (Figure 5.14). According to all the results, cell size groups 35 × 35 and 2 × 2 should be rejected, because cell size group 35 × 35 offers less much lower classification accuracy and cell size groups 2 × 2 and 5 × 5 have many more descriptors, which are not necessary for such a large amount of data. Two combinations have been selected as the candidates for the final parameters: cellsize 10 × 10 with blocksize 2 × 2, and cellsize 7 × 7 with blocksize 1 × 1. Both sets have a nearly 100% true positive rate, and a less than 1% false positive rate. The difference between them is that the 10 × 10 group has 1,296 descriptors and 7 × 7 group has 900, which can reduce the computational resources needed. So both descriptors were used in the classifications which will be discussed later in this chapter.

As mentioned, the classification model was trained from four angular offsets to horizontal to solve the rotational invariance problem. Figure 5.15 below gives the HoG features for all offset groups from a training sample. Each offset group created one classification model and all of these were integrated into one main classification model.

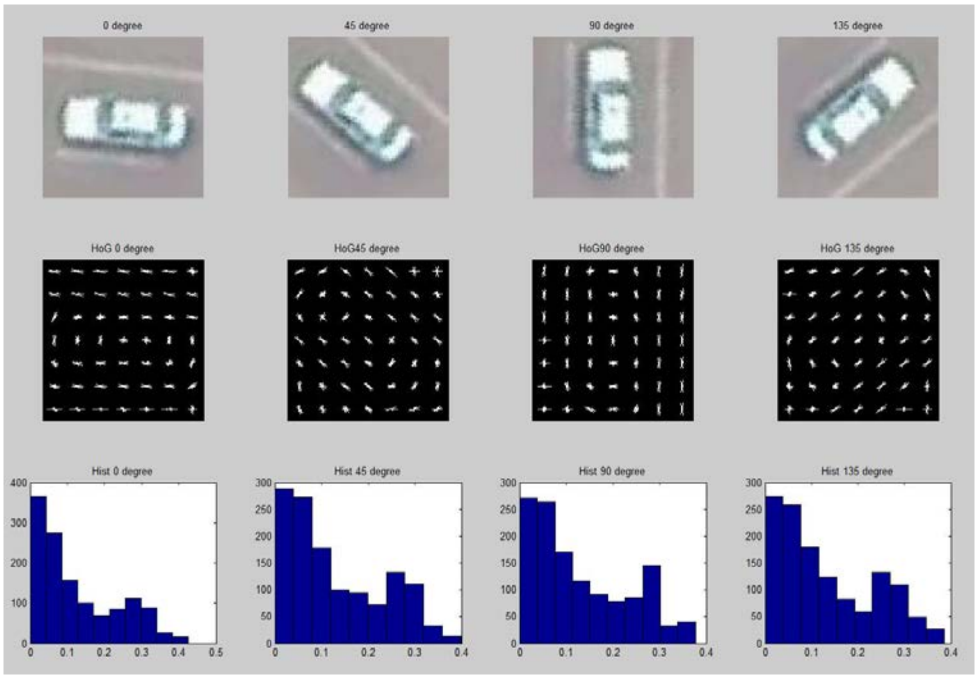


Figure 5.15: Four orientations offset group for training.

The testing used two size combinations: cell size 10 × 10 pixels and block size 2 × 2 cells, and cell size 7 × 7 pixels and block size 1 × 1, which were the candidates for HoG parameters, to train the models separately. The images below show the SVM training model graph of both cell size groups in each orientation of all training samples.

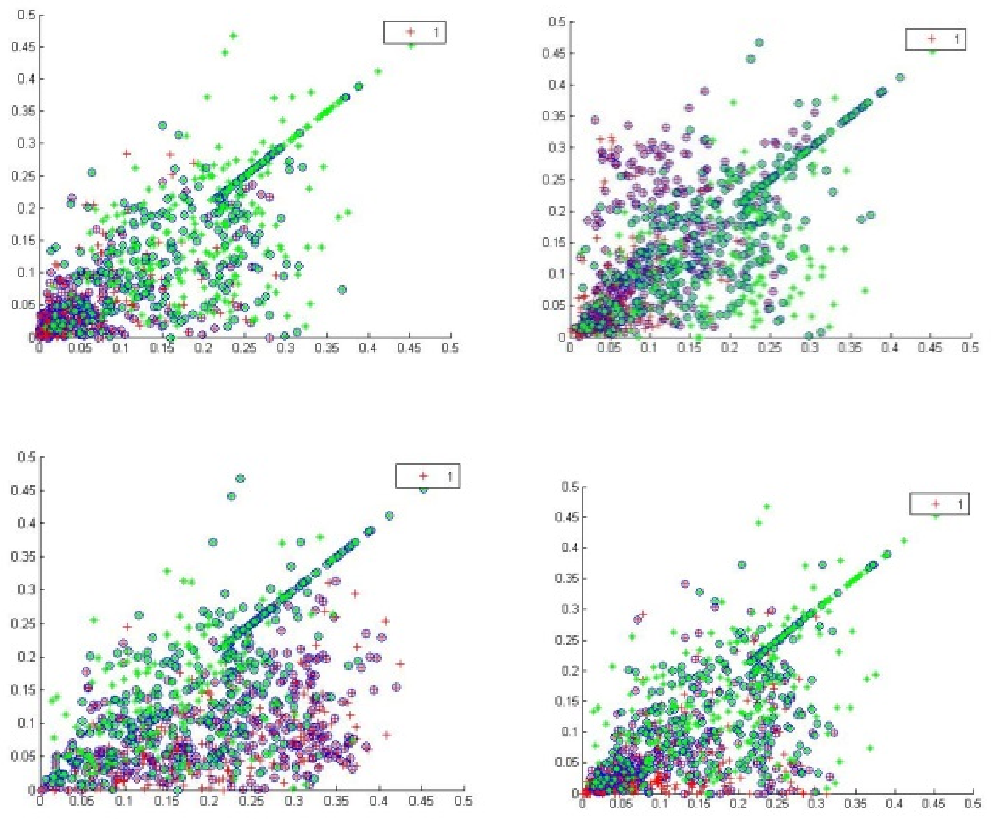
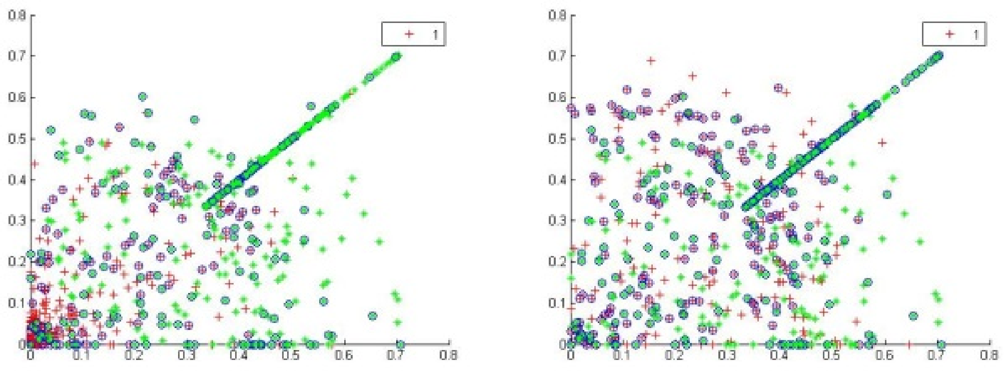


Figure 5.16: SVM training model for cell size 10 × 10. (Top-left: 0°, top-right: 45°, bottom-left: 90°, bottom-right: 135°)



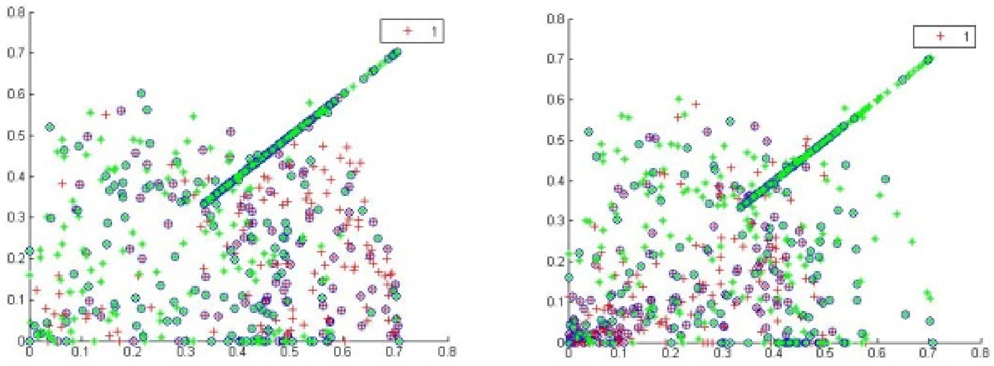


Figure 5.17: SVM training model for cell size 7 × 7. (Top-left: 0°, top-right: 45°, bottom-left: 90°, bottom-right: 135°)

According to the graphs (Figures 5.16 and 5.17), models of cell size 10 have more centralised distributions and models of cell size 7 have more scattered distributions, which means the features of each group are more independent to each other in terms of vehicle detection. Both classification models were used in the programme to test the accuracy. The results (Figure 5.18) show clearly that cellsize 10 × 10 with blocksize 2 × 2 parameters achieve the best detection results. As a result, cell size 10 × 10 and block size 2 × 2 were used as the HoG parameters. Furthermore, in order to boost the classification accuracy, AdaBoost has been applied to the SVM classification which boosted the classifier model.

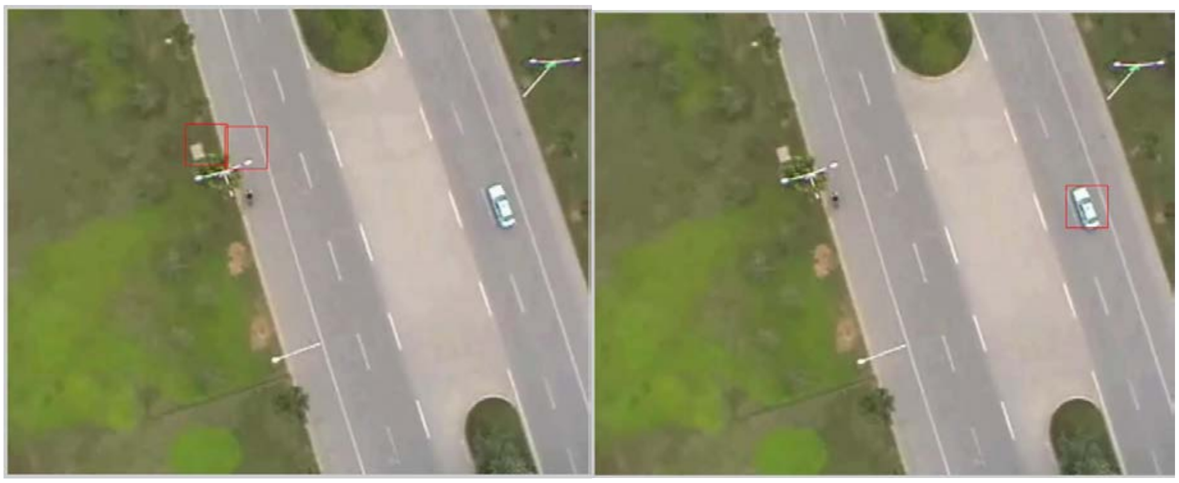


Figure 5.18: Testing results for 7×7 cellsize (left) and 10×10 cellsize (right).

5.4 Evaluation

Again, the testing used five videos of different scenarios. This time the AdaBoost mechanism was used in classification to improve accuracy. Table 5.8 and 5.9 shows the detection accuracy results for each testing video by using FAST-HoG and FAST-HoG with the AdaBoost. Figure 5.19 shows some examples of detection results.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4763	412	561	89.46%	90.73%
Video 2	5511	4893	431	618	88.79%	90.32%
Video 3	5134	4679	387	455	91.14%	91.75%
Video 4	1848	1643	209	205	88.91%	88.81%
Video 5	1918	1821	58	97	94.94%	95.92%

Table 5.8: Detection results when using the FAST-HoG method.

Data set	Vehicles	TP	FP	FN	Accuracy (%)	F-measure (%)
Video 1	5324	4972	351	352	93.39%	93.40%
Video 2	5511	5213	387	298	94.59%	93.83%
Video 3	5134	4814	296	320	93.77%	93.99%
Video 4	1848	1723	153	125	93.24%	92.53%
Video 5	1918	1875	48	43	97.76%	97.63%

Table 5.9: Detection results when using FAST-HoG with AdaBoost.

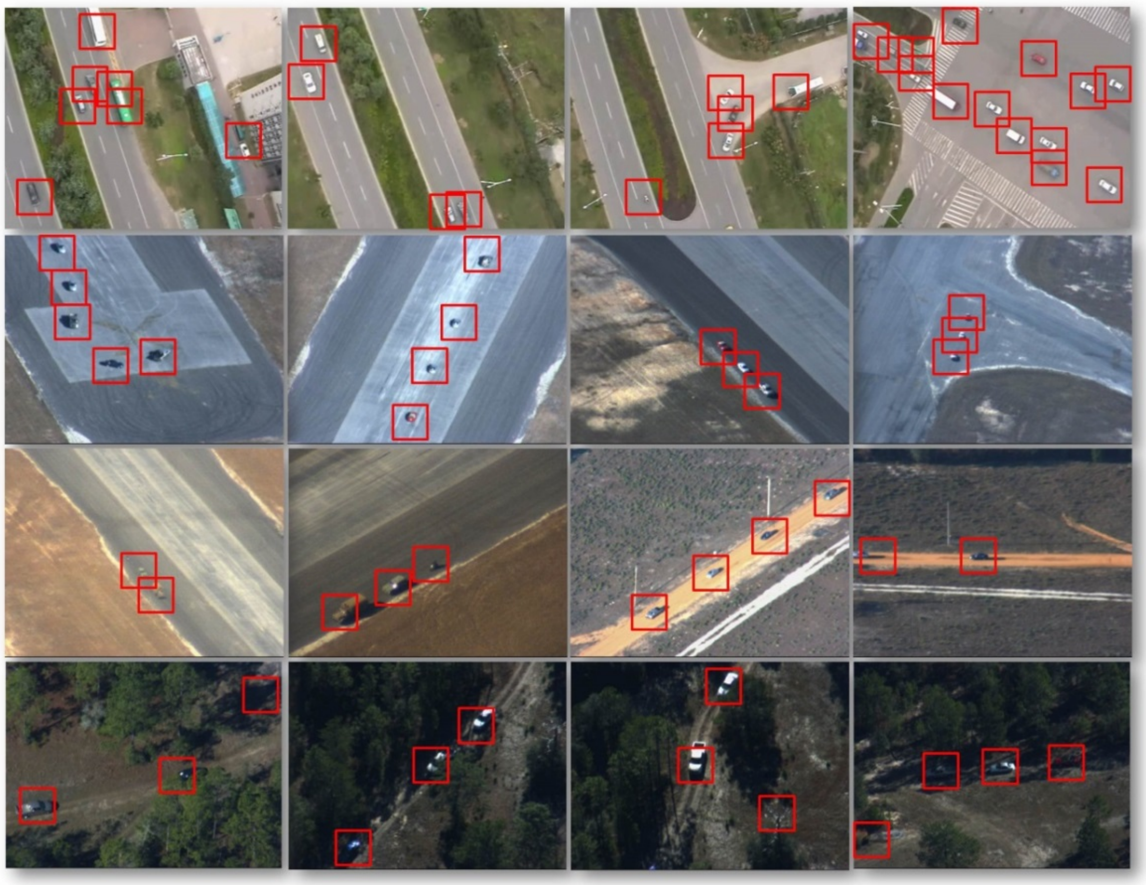


Figure 5.19: Examples of the detection results of using the FAST-HoG approach (with AdaBoost).

5.5 Comparison and Evaluations

After introduced all three vehicle detection approaches, this section discusses the compressions of the proposed detection methods performance to each other and other vehicle detection approaches. This could clearly carry out the improvement of the proposed methods to others, also indicates the better approach in different scenarios.

5.5.1 Detection performance

The tables below (Table 5.10 to Table 5.14) summaries the detection accuracy and F-measure values obtained by using the three proposed methods (HSV-GLCM with de-blurring feature, ISM-SIFT feature and FAST-HoG feature) on each testing video. These tables can give us a general idea of which detection method performed better in each different scenario.

Detection Results in Video 1 (2461 frames)						
Method	Vehicles	TP	FP	FN	Accuracy	F-measure
HSV-GLCM	5324	4942	328	382	92.82%	93.29%
ISM-SIFT	5324	5142	411	182	**96.58%**	**94.55%**
FAST-HoG	5324	4972	351	352	93.39%	93.40%

Table 5.10: Detection results for the complex background video (Video 1).

Detection Results in Video 2 (1301 frames)						
Method	Vehicles	TP	FP	FN	Accuracy	F-measure
HSV-GLCM	5511	5231	266	280	**94.92%**	**95.04%**
ISM-SIFT	5511	5212	336	299	94.57%	94.26%
FAST-HoG	5511	5213	387	298	94.59%	93.83%

Table 5.11: Detection results for the occlusion video (Video 2).

Detection Results in Video 3 (1833 frames)						
Method	Vehicles	TP	FP	FN	Accuracy	F-measure
HSV-GLCM	5134	4938	166	196	**96.18%**	**96.46%**
ISM-SIFT	5134	4837	316	297	94.22%	94.04%
FAST-HoG	5134	4814	296	320	93.77%	93.99%

Table 5.12: Detection results for the blocked vehicle video (Video 3).

Detection Results in Video 4 (168 frames)						
Method	Vehicles	TP	FP	FN	Accuracy	F-measure
HSV-GLCM	1848	1643	152	205	88.91%	90.20%
ISM-SIFT	1848	1699	223	149	91.94%	90.13%
FAST-HoG	1848	1723	153	125	**93.24%**	**92.53%**

Table 5.13: The detection results in blurred images video (Video 4).

Detection Results in Video 5 (1918 frames)						
Method	Vehicles	TP	FP	FN	Accuracy	F-measure
HSV-GLCM	1918	1884	56	34	98.22%	97.67%
ISM-SIFT	1918	1893	47	25	**98.70%**	**98.13%**
FAST-HoG	1918	1875	48	43	97.76%	97.63%

Table 5.14: Detection results for the changed vehicle size video (Video 5).

For general viewing, the ISM-SIFT approach has the highest average detection rate (95.20%, 94.22%), and is better than HSV-GLCM (94.21%, 94.53%) and FAST-HOG (94.55%, 92.28%). However, in certain situations, such as the occlusion challenge in Video 2 and the blocked vehicle challenge in Video 3, HSV-GLCM achieved higher accuracy than the others. In the blurring challenges of Video 4, FAST-HoG method offered the best detection results. Therefore, each detection approach achieves differently under different difficulties and challenges presented by the testing videos. The proposed tracking system with a learning process to constantly update the classifier during the tracking process was able to improve detection performance.

5.5.2 Impact of the training samples

Each detection approach requires training samples for the classification to be used during the detection process. The testing used different numbers of training samples to evaluate the effects on classification performance and the sensitivity to the use of different numbers of training samples.

The numbers of training samples were used from 300 to 1,200, which contained positive samples and negative sample. The classifier was tested after adding each one more sample, and the classification accuracy against the number of training samples was calculated.

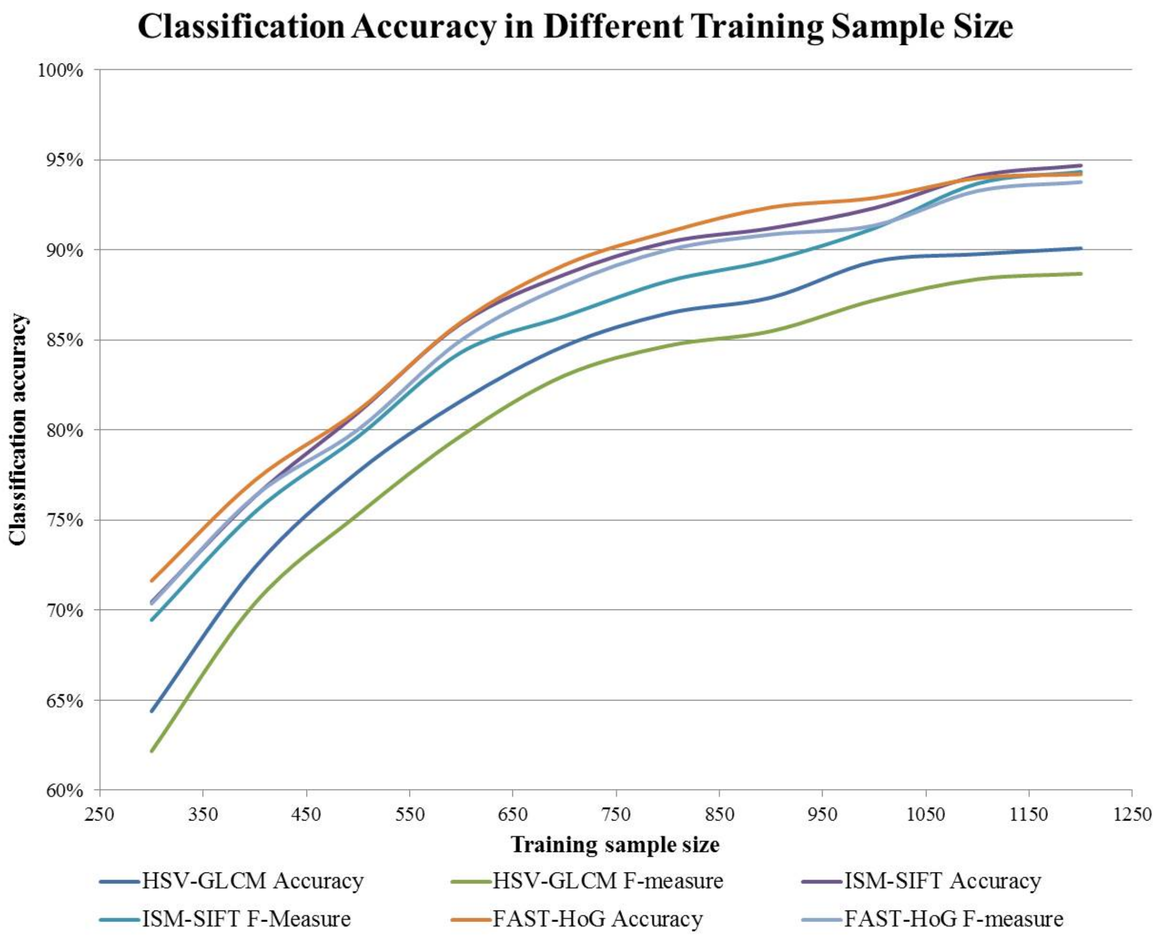


Figure 5.20: The trend lines of detection accuracy based on different training sample sizes.

The Figure 5.20 shows that the classification accuracy of each method is very sensitive to the training samples numbers, which increased when larger training sample size was used. However, each curve eventually level off after 1,000 training samples were used, which means it became saturated in the end so there is unnecessary to use too much training samples in the classifications.

5.5.3 Comparison of Detection Performance

This experiment evaluated the detection components of the detection approaches. For each video sequence the proposed approaches were compared with the approaches using:

- Vehicle detection by using bLPS-HOG feature [134]
- Vehicle detection by using SIFT point feature [135]

- Vehicle detection by using PLS Hough transform [136]
- Vehicle detection by using HSV-GLCM approach
- Vehicle detection by using ISM-SIFT approach
- Vehicle detection by using FAST-HOG approach

Table 5.15 shows the results achieved. The accuracy and F-measure are shown for each approach. In the Video 1, which contained significant background clutter and objects similar to the vehicles, all of the detection results are above 90% as opposed to the compared approaches, which offer results lower than ours, especially the PLS Hough, the accuracy of which was around 70%. The detection rates in the occlusion situation are lower for all approaches, because the detection process is only based on the current image’s features, and when the vehicles’ features are joined together detection will fail. However, this problem can be solved during the tracking process with additional previous detection and tracking information. This phenomenon also applies to the blocked vehicle situation, in which vehicles are blocked by the environment. The SIFT and ISM-SIFT approaches are very strong in the detection of blurred images and changing size and appearance situations, with above 90% detection accuracy. This is due to the nature and characteristics of the SIFT feature, which include strong descriptor information.

Data set	bLPS-HoG	SIFT	PLS Hough	HSV-GLCM	ISM-SIFT	FAST-HOG
Video 1	89.90%/87.13%	70.14%/69.12%	70.52%/68.14%	92.82%/93.29%	**96.58%/94.55%**	93.39%/93.40%
Video 2	80.17%/79.16%	70.96%/68.83%	74.21%/72.15%	**94.92%/95.04%**	94.57%/94.26%	94.59%/93.83%
Video 3	73.43%/70.96%	74.18%/72.48%	67.15%/65.19%	**96.18%/96.46%**	94.22%/94.04%	93.77%/93.99%
Video 4	82.21%/80.33%	93.01%/90.16%	75.33%/73.48%	88.91%/90.20%	91.94%/90.13%	**93.24%/92.53%**
Video 5	75.09%/73.66%	96.05%/92.17%	97.91%/95.16%	98.22%/97.67%	**98.70%/98.13%**	97.76%/97.63%

Table 5.15: Comparisons of detection performance using different detection methods.

5.6 Conclusions

This chapter has proposed a FAST-HOG approach to vehicle detection. The approach used a modified HoG approach with FAST corner point extraction. This method can significantly tackle the complex background and the direction invariant challenges, this method also can improve the performance of the process by narrow down the testing regions using corner point detection. The AdaBoost mechanism was also added into the classification process. According to the experimental results, detection performance is improved by using AdaBoost. Also note that for Testing Video 4, the detection results were around 90%, thus the blur problem did not affect detection performance. However, because this detection method was based on the region texture feature, which means the detection performance is relied on the training classifier. Thus, the detection will become unreliable if a brand new testing video with environment is used. This challenge can be solved by using adaptive model tracking process which can update the detection classifier during the process, which will discussed in the following chapter.

In additional, this chapter compared the three proposed vehicle detection approaches (HSV-GLCM, ISM-SIFT and FAST-HoG) with other existing detection methods (bLPS-HoG, SIFT and PLS Hough). According to the comparison results, it can be concluded that the new proposed approaches have improved detection performance under various conditions and have better detection accuracy than others. In the next chapter, a new tracking mechanism will be discussed.

Chapter 6

The Proposed Self-Learning Tracking Detection Approach

In this chapter, a Self-Learning Tracking and Detection (SLTD) approach is proposed. The proposed approach can achieve long-term tracking of multiple vehicles in aerial videos. The idea of this approach was inspired by the Tracking-Learning-Detection (TLD) approach [18], in which the detection and tracking systems are linked by a learning component. The learning process can estimate missed detections and false alarms in the tracking process. Inspired by this idea, this chapter propose a new self-learning method, which can estimate the detector errors and update the detection classification model using the results from trackers. It is also assumed that the trackers may make errors during the tracking process, so a Forward and Backward Tracking (FBT) approach is proposed in order to monitor tracking performance. This approach estimates the discrepancies between the tracked results in time sequence order so that unusual appearances in a tracker compared with the previous trackers will be considered errors.

6.1 Introduction

The main challenge of vehicle tracking is caused by the fact that the target vehicle might change its appearance or disappear and reappear during the tracking process, which can cause errors. The tracking process should be able to handle various problems. First of all, the tracker and detector should be scale-invariant to the targets, which can solve the problem of the target potentially changing scale in the image when the UAV changes altitude during flight. Secondly, the UAV’s flight direction changes rapidly and unpredictably, which can cause apparent changes in the detection of the target. Thus, a rotationally invariant system is needed for the process. Furthermore, the illumination of the target may vary depending on the UAV’s flight direction and the shooting angle relative to the target which means illumination invariance also is a requirement. When the UAV changes its flight direction, the images captured by the camera may become blurred and warped, so the images get transformed. When such a problem occurs, transformation invariance is needed. It might also come across other issues, such as background confusion and targets occlusion.

The Tracking-Learning-Detection (TLD) algorithm [18] was discussed in the literature chapter. The TLD algorithm is a novel tracking process that uses a P-N learning process between the tracking and detection steps of the process, which can estimate errors. The TLD approach begins with the selection of a region that contains a target; that region is defined as a detector, which stores the feature of the target. In the following frame, a tracker is addressed to the target, which will predict the trajectories of the target between the consecutive frames. The detector treats every frame as independent and carries out a full scan of the image to localise the targets that have been learned in the previous frames. The learning observes the performances of both tracker and detector and basically estimates the errors made by the detectors (either false positive errors or false negative errors). The learning process generates training samples, which can help avoid errors in future tracking. The classical tracking approaches have a common problem, which is the lack of communication between the detectors and the trackers. The TLD approach tackled this problem by creating a novel algorithm for a P-N learning system between the detection and tracking processes. However, it is found that the TLD has the following drawbacks and difficulties:

- The TLD approach can only track a single target during the process. This limits tracking performance in real-life applications which usually have several targets.

- The TLD approach requires the manual selection of the target at the beginning of the tracking process; this can be seen as an advantage in that no pre-training is needed for the tracking process. However, this defeats the point of this research, which is to automatically detect and track the target.
- In TLD tracking, the tracker and the detector will monitor each other’s performance horizontally trough the learning component. It is noted that, in the scenario of the P-N Learning component making an error, there is no process that can monitor its performance which could lead to detectors errors

To tackle these problems a method of Self-Learning Tracking Detection (SLTD) approach was proposed for detection and tracking, the main process of which is shown in Figure 1.7 and a detailed chart in Figure 6.1. The process makes the assumption that both the detection and tracking process may make errors. The P-N learning was applied to the detection and tracking processes so they can monitor each other. The difference between the proposed approach and the TLD approach is that, during detection process, vehicles are detected automatically by the detection approach while TLD needs to manually select the target at the beginning. Furthermore, the proposed approach can track multiple vehicles rather than just a single target, as in the TLD approach. This method also revised the P-N learning approach so that it not only estimates the errors between tracking and detection, but also updates the classification model for detections and saves the estimated positive and negative samples from the current process into the classifier database. This has the advantage that, when using the application in an unknown environment, the system can learn and adopt the features in the current scene. In the original TLD P-N learning is a semi-supervised learning, which means the training data has been used as the supervisory information in order to classify the unlabelled data. There are two types of experts in the P-N learning: P-experts and N-experts which can identify the classified samples by certain criterions. The classified samples used in the P-N learning were from the selection of the first frame, which has limited ability to identify the objects apart from the selected target. In this proposed method, the learning components are selected from detectors and tracker in every frame, which means the learning, is taking place during the process. The trajectory estimation of tracker is also considered in the tracking and learning process.

Like in the tracking and detection components, it is also assumed that the learning component could make errors, so a Forward and Backward Tracking (FBT) mechanism was proposed. The main purpose of using the FBT is to check whether there are any errors in the tracking result sequences. Basically, this method saved all the tracked vehicles' features in to the Tracked Vehicle Database (TVD), in which each vehicle has an individual array. The FBT will detect suspicious results in the trackers based on the information in the TVD. For example, it is assumed that if the tracker is tracking a specific vehicle in a sequence of frames, the features of the current tracking result in the tracker should be very similar to the previous results. So when there is a significant change between the current and previous tracker in term of the feature matching, the FBT will determine the tracking result is an error. When such an error occurs, the tracker will stop the tracking process and go into a standby mode. In the meantime, there are two decisions that the tracker can be made, first decision is address a new tracker to this error if this error result is the features of another vehicle; second decision is considered the result as real error, which is the features of the environment object. The main advantage of using the FBT is that, when the learning component identifies a fault, it responds to the detector, which will lead to the tracker making an unnoticed error; at this point, the FBT can estimate such errors and correct from them, and the samples in the learning component will also update the training samples for classification in future detection. This method applied the SIFT matching method to the FBT process, because SIFT offers a considerable matching performance and its processing resources requirements are acceptable.

The FBT has a tracked vehicle database (TVD) which stores the SIFT information about previously tracked vehicles. The FBT will compare the SIFT feature in current tracker with the tracker in the previous frame sequences, if the matching result is above the threshold value, which means the current tracker is tracking a same vehicle as previous and the SIFT information of this target will be saved in the TVD. When the current tracker is considered as tracking the same target, it will be compared with the detector of this target in the next coming frame. This new matching process will monitor whether the detector is correct or not, which the detection result will be used for updating the detection classifier as positive or negative sample. On the other hand, if the matching result is below the threshold when compare with the previous frame sequences the FBT will consider the current tracker has tracked another vehicle or environment objects. When the FBT is considered the current tracker is a new tracked vehicle, a new tracker will be addressed to this new target, which also will be saved into the TVD. All these results in the FBT will be considered positive or negative samples for further classification.

6.2 Self-Learning Tracking and Detection Algorithm

This section investigates the SLTD framework. The purpose of this approach is to improve vehicle detection and tracking performance in real-time video stream. The SLTD approach is designed for the long-term tracking of multiple vehicles after the detection process is complete. The diagram of the SLTD process is shown in Figure 1.7. The framework contains three components: **Trackers** (Forward and Backward Tracking) estimate the position of the targets in the frame sequences, under the assumption that the target is visible in the following frame; **Detectors** (Detection) perform full scans of every frame and get the position of the targets, (note that the detectors are independently to the trackers). As mentioned above, under the assumption that both detection and tracking can make mistakes during the process, and that there are two types of errors that can occur: false positives and false negatives. Thus, a **Self-learning** (Learning) model was created to monitor the performance of both the detectors and the trackers and to evaluate any errors made by them in order to update the classification model continuously throughout the process and avoid these errors in future detection and tracking. Through this learning process, the classification model can absorb new vehicle appearances for later detection and tracking. In every frame of the video, the aim is to be able to evaluate the current detector and tracker to identify whether there are any errors and update the classification model in order to avoid these errors occurring again. The key idea of SLTD is that the errors between detectors and trackers can be identified (these errors come in two types: false negatives and false positives). Figure 6.1 shows a detailed process diagram of the entire process. In this process, the FAST and HoG detection method is used for the detection process.

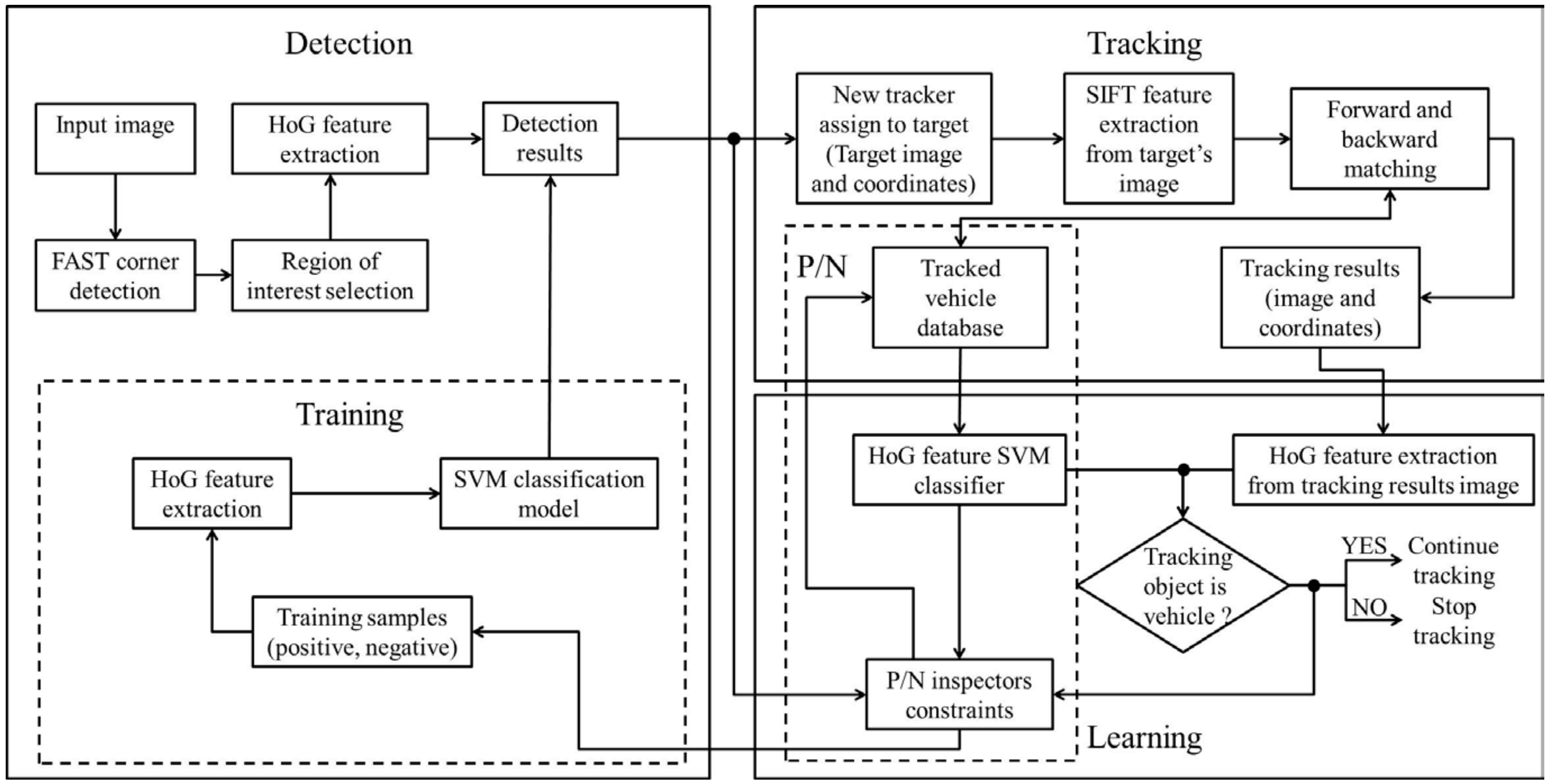

Detection
Input image
HoG feature extraction
Detection results
FAST corner detection
Region of interest selection
Training
HoG feature extraction
SVM classification model
Training samples (positive, negative)
Tracking
New tracker assign to target (Target image and coordinates)
SIFT feature extraction from target's image
Forward and backward matching
P/N
Tracked vehicle database
Tracking results (image and coordinates)
HoG feature SVM classifier
HoG feature extraction from tracking results image
Tracking object is vehicle ?
YES
Continue tracking
NO
Stop tracking
P/N inspectors constraints
Learning

Figure 6.1: The detailed process diagram of the self-learning detection and tracking process.

6.2.1 Forward and Backward Tracking

After the detection, the Forward and Backward Tracking (FBT) process is followed. This method utilises a TLD tracking algorithm based on optical flow and extends it to track multiple targets. The FBT method has been proposed to monitor the vehicle tracking and detection results. FBT runs in parallel with detection and monitor the tracking results by setting the detection results as ground truth. It can also run self-check based on prior and later information.

Algorithm 6.1 : Forward and Backward Tracking

```
Input: target coordinates $C_f^n$ , target image $I_f^n$, threshold $\alpha$, $framenumber$
Output: tracking results $T_f^n$
 $R_{f+1}^n$ ← generate RoI ($C_{f+1}^n$, $F_{f+1}$)  ($C_{f+1}^n$ refer to algorithm 7.2)
 $R_{f-1}^n$ ← generate RoI ($C_{f-1}^n$, $F_{f-1}$)  ($R_t^n$ = Region of Interest)
for $f = 1 : framenumber$
  $S_f^n$ ← calculate SIFT features of $I_f^n$
  $S_{f+1}^n$ ← calculate SIFT features of $R_{f+1}^n$
  $S_{f-1}^n$ ← calculate SIFT features of $R_{f-1}^n$
  ( $S_f^n$ = SIFT descriptors )
  Matching score $m_{n+1}$ ← SIFT matching ($S_f^n$, $S_{f+1}^n$)
  Matching score $m_{n-1}$ ← SIFT matching ($S_f^n$, $S_{f-1}^n$)
 If $m_{n+1} > \alpha$ and $m_{n-1} > \alpha$
  Continue tracking and $T_f^n \in M_p$ ($M$ = tracked vehicle database )
 else
  Stop tracking and $T_f^n \in M_n$
End for
```

An algorithmic description of FBT is given in Algorithm 6.1. After the detection process, all detected vehicles are labelled by the centre coordinates $C_n^f(x, y)$ and the image window I_n^f, where n is the number of the target and f is the current frame number, the I_n^f is from the detector results.

Set F_f as the current image at frame f, the region of interest (RoI) in the next frame (R_{f+1}^n) and previous frame (R_{f-1}^n) of the target is calculated by using the current target coordinate C_n^f and the images from next and previous frames (F_{f+1}, F_{f-1}). The RoI is selected around the coordinate of the target in the next frame, it is assumed that the target is impossible to appear to a place in next frame where is far away from the location in the current frame, unless the target is blocked and will be appeared again in further frame. If the blocking issue occurs, the forward and backward matching process with tracked vehicle database can solve this problem. The RoI can be selected as:

$$RoI = area\ (\ (C_n^f(x) - {}^{w}/_{2}), C({}_n^f(y) - {}^{h}/_{2})\ , w, h\ ) \tag{6.1}$$

where the x and y is the centre coordinates of target, w and h is the width and height of the region.

The forward and backward matching process is conducted with the image patches of the targets I_f^n, the region of interest R_{f+1}^n and R_{f-1}^n, which gives their SIFT descriptors S_f^n, S_{f+1}^n and S_{f-1}^n. Then the SIFT matching process, which refers to the detection method in chapter 5, is taking place, which generates the matching score m_{n+1} and m_{n-1}. A threshold α was set for classify the weather the RoI contains the target or not by compare the similarity of SIFT feature between the target image patch with the RoI in both next and previous frames. If both matching score are higher than the threshold, which means the current tracking result is correct.

Algorithmically, the tracker T_f^n in the tracking system computes the similarity of SIFT features between the detected vehicles' image patches and the RoI. The detected vehicles' area is based on the detection in the previous frame of the video; this area is considered a sample of the vehicle and is used to find matching features in the regions of the following frames. Each frame of tracking will give a positive or negative result from the matching processes. However, these results might be inaccurate if there are any other vehicles similar to the sample vehicle. For instance, it may find matching results just above the threshold, or when the target exits the image, there may still be a vehicle similar to the target. The proposed FBT method solves this kind of issue. It is assumed that, if the tracker is tracking the same vehicle in the video, the features of that vehicle will be highly similar throughout. Thus, this thesis compares each tracking result to the next frame (forward) and to the previous frame (backward). If the similarity of the feature is lower than the threshold, it is considered a lost target or a target that has left the image. Tracked vehicle databases M_P, M_N are created to store the positive results and the negative results, which will be of use in further tracking and detection processes. The main purpose of creating Tracked Vehicle Database (TVD) M is to let the system continuously update the database for the detection classifier. For each frame, the regions of interest undergo a matching process with each vehicle in positive memory $\mathrm{M_P}$. $m_{n+1} > \alpha$ and $m_{n-1} > \alpha$

$$TVD = \begin{cases} M_p \in T_f^n \ (m_{n+1} > \alpha \text{ and } m_{n-1} > \alpha) \\ M_n \in T_f^n \ (m_{n+1} > \alpha \text{ or } m_{n-1} > \alpha) \end{cases} \tag{6.2}$$

6.2.1.1 Forward Trajectory Estimation

Algorithm 6.2 shows the process of the trajectory estimation. It is assumed that the same single object appearing in several locations in a single frame is unlikely to happen and each object should appear at one position in each frame in the sequence and can build up a trajectory path, so let the system predict a trajectory for the target by analysing the locations of the tracker in prior frames. In other words, a trajectory prediction can identify incorrect detectors in the following frames. The resulting trajectory is measured by the appearance of the SIFT points of the target in the forward and backward frames. Set the $F_f^n = (C_f^n, C_{f+1}^n \dots C_{f+k}^n)$ where f stands for the frame number and k indicates the length. The SIFT point set of a target is S_f^n and the S_{f-1}^n, S_{f+1}^n are the SIFT points of this target in the previous and next frame.

The process creates a temporal structure in the video and assumes that the object moves along a trajectory. The tracking result has the location information of the object in the previous frame and predicts the object's location in the future frame using a tracker. The tracker not only contains the texture features of the target but also has the location coordinates of the target. If the detector has labelled the current location as negative, i.e. if the detection system has made a false negative error, the PNSL generates a positive training sample. It also assumes the target can only appear at a single location in the frame image patch, so it selects the most likely patch in the testing frame. Note that there are multiple image features that are similar to the trackers in the testing image, unlike the detector. The patches that are not overlapping with the most likely detector are labelled as negative samples and the most likely patch can relocate the tracker in the following testing frame. Figure 6.2 shows an example of how this system works. The vehicle is detected by the detection process and a tracker is assigned to the detected vehicle. The tracker represents the positive outputs which will be sent to the positive training sample database.

Algorithm 6.2 : Forward Trajectory Estimation

```
Input: target coordinates $C_f^n$ , target image $I_f^n$ , region of interest $R_{f+1}^n, framenumber$
Output: target coordinates $C_{f+1}^n$
for $f = 1: framenumber$
    $S_f^n$ ← calculate SIFT features of $I_f^n$
    $S_{f+1}^n$ ← calculate SIFT features of $R_{f+1}^n$
    $S_{f-1}^n$ ← calculate SIFT features of $R_{f-1}^n$ ( $S_f^n$ = SIFT descriptors )
    SIFT difference ← $\left( \frac{\left|(S_{f+1}^n - S_f^n)\right|}{S_f^n} + \frac{\left|S_f^n - S_{f-1}^n\right|}{S_{f-1}^n} \right)$
    If SIFT difference $< 0.2$
        $C_{f+1}^n \leftarrow Centre\ coodinates\ S_{f+1}^n$
    else
        $update\ S_{f+1}^n$
End for
```

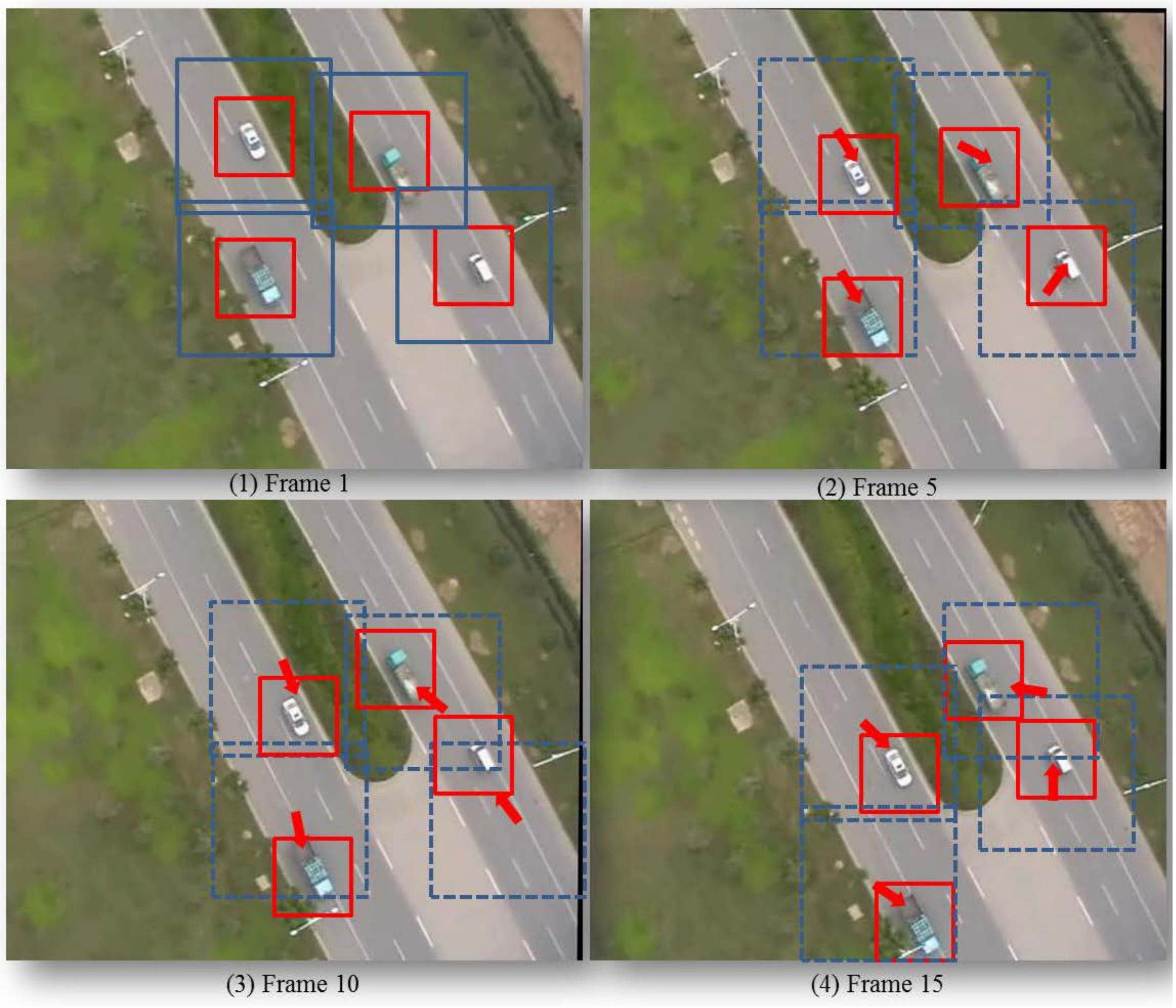


Figure 6.2: The detector process in a sequence of frames. The red boxes are the detectors that detected vehicles in the current frame. The blue boxes are the prediction areas in the following frame and the arrows are the moving factors of the detectors.

Figure 6.3 shows an example of the FBT process. The figure contains the tracking process of three trackers. In the first row, the first tracker was assigned by the detector in the first frame. Then in the second frame, the target was tracked by the tracker and the tracker run a backward matching process with the tracker in the first frame, the matching result was 98.7%. Furthermore, the tracker in the second frame also made a forward matching process with the next frame tracker when it became available. The forward and backward matching results was 98.7% and 99.2%, which were high enough to conclude that the tracer was tracking the same target, which a green face was assigned. In the meanwhile, if the system is tracking a same target, the feature of the tracker can set as a reference substance for monitoring the detector. As same as the first row, the tracker in the second row was tracking the same vehicle in the first five frames, however, because there were two vehicles very close to each other so the tracker has tracked the other vehicle, which lead a low backward matching result and forward matching result in the previous tracker. In this case, the tracker was compared with the detector in the same frame to check if this is a vehicle or not, if the detector response a positive feedback, which means what the tracker tracked is a vehicle. Then the tracker was searched in the TVD to find weather this vehicle has already been detected or not, if yes, the tracker of this vehicle will be replaced to the current tracker, if not, a new tracker was assigned and the feature of the tracker was stored into the TVD. Finally, the third row shows the situation that the tracker was tracked a non-vehicle feature, which has low backward matching result and cannot match with the detector, so a false positive error was made and the feature was stored as negative training sample.

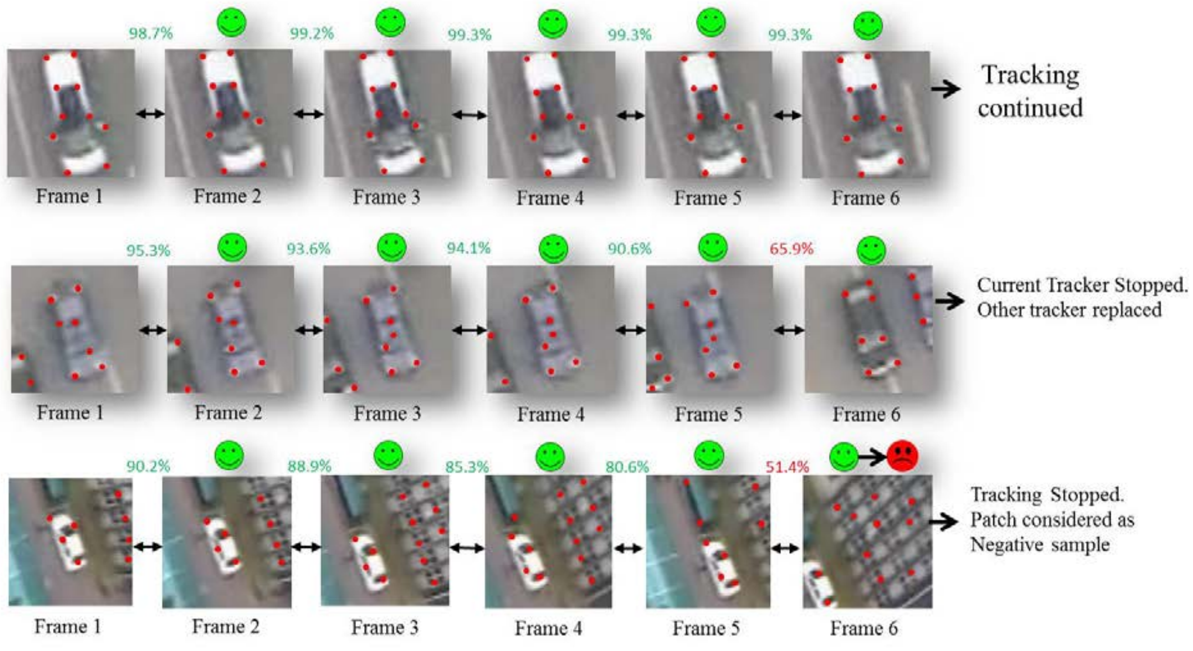


Figure 6.3: An example of the FBT process.

6.2.2 Positive and Negative Self-Learning (PNSL)

This section introduces the self-learning process. The purpose of self-learning is to improve the performance of vehicle detection by using the tracking results, and vice visa, by using the detection result to ensure the tracking accuracy. In order to evaluate the detector(s) in each frame, two different self-learning inspectors are included in the proposed approach: positive inspectors (P) and negative inspectors (N). Positive inspectors are used to identify whether a tracker labelled as positive by the classifier has been recognised as negative by the detector. Negative inspectors are used to identity whether those trackers labelled as negative by the classifier have been recognised as positive by the detector.

The self-learning process has four steps:

- Generation of a HoG classification model from the images in TVD
- Collection of labelled data from the tracking results in TVD
- Supervised training based on labelled data from detection and tracking results
- Generation of positive and negative training samples to update the SVM detection classification model

As shown in Figure 6.1, the learning component is linked with the tracking results and the training classification model in the detection process. In the tracking process, the tracking results from tracker T_f^n are stored in the TVD and each result contains the target image I_f^n. The training process is initialised by generate a SVM classification model using the HoG feature descriptors from the targets' images in the TVD. Followed by calculate the HoG feature descriptors from the current tracking image are calculated. By using these two values the SVM gives the decision of whether the tracker is the vehicle or not, which generates a label L_f^n of this tracker ($L \in \{0, 1\}$), which is considered as labelled data S_l. The L_f^n is assigned to the training classifier from the training samples, then it passed to the supervised learning to trains the SVM classifier θ. Then the iterative bootstrapping is applied after the classifier is generated.

As mentioned, the process proceeds iteratively, and in iterationk, the classifier trained in $f-1$ assigns labels to the training samples formed from the tracking results; $L_f^n = Classifier\left(I_f^n \middle| \theta^{f-1} \right)$, where the L_f^n is the tracking label of the image I_f^n of the tracking result T_f^n vehicle at f frame. Note that, the classifier can operate on multiple trackers at the same time. Then the self-learning system is used to check whether the labels assigned by the classifier are correct or not. The sample labels that are incorrect are corrected and added to the training samples. The iteration is ends with the retraining of the classifier with the updated tracking results.

6.2.2.1 P/N inspector constraint

The input to the P/N inspector system is a labelled data S_l and an unlabelled data S_u where $l \ll u$. This is because the number of ground truth data has to be larger than the unpredicted data, which the classification can be accurate. The task of the P/N process is to estimate the detection result error based on the labelled tracking results S_l and generate the positive and negative training samples to update the detection classifier.

The process is started by inputting the labelled data set into the training set. Then the detection classification model is trained using the training set data, in this process, the SVM classification model was used. After the SVM classifier has been created, the P/N process then proceeds with an iterative bootstrapping. In this process, the classifiers which have been trained on the labelled detector data classify the unlabelled tracker data as shown in Figure 6.4. Then the P/N inspector analyses the classification and estimates which data have been tracked and detected incorrectly. These data are then re-inserted into the training data with modified labels. The process continuously iterates until the tracking is finished.

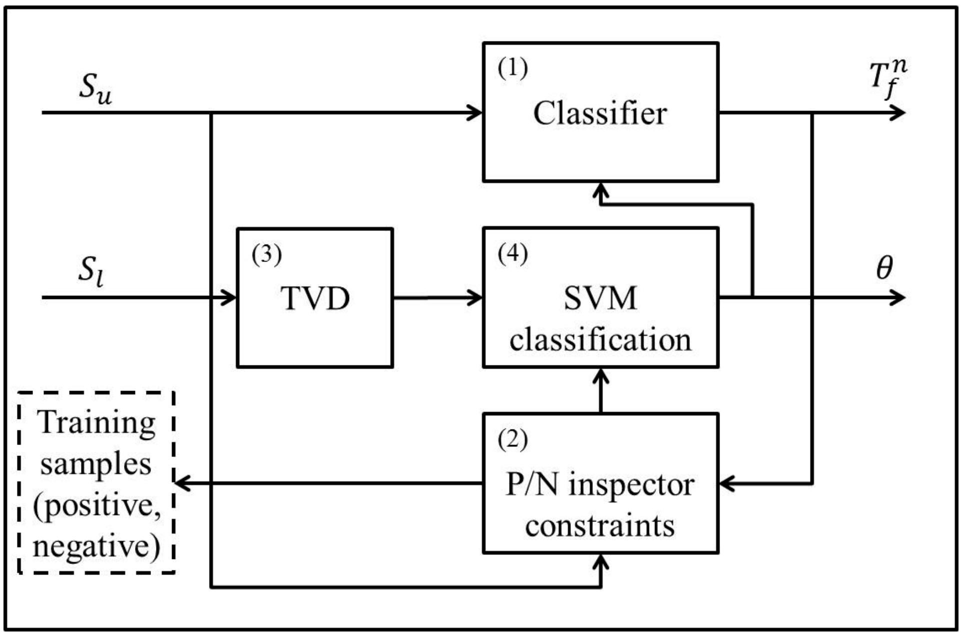


Figure 6.4: The P/N process diagram (the number indicates the process order).

The most important element of PNSL is the estimation of the errors between the detectors and trackers. The key purpose is to differentiate the false positives from the false negatives. As a result, the unlabelled data were separated into two groups; the positive label group and the negative label group. The PNSL process separately analyses the positive and negative label data, and define these processes were defined as P-inspector and N-inspector. The P-inspector analyses the data that is classified as negative, identifying false negatives and assigning positive labels to the testing data. The N-inspector analyses the data that is classified as positive, identifying the false positives. Set $n^{+}(k)$ as the P-inspector and $n^{-}(k)$ as the N-inspector in the iteration k. The P-inspector and N-inspector can increase the generality and the discriminability of the classification model through the learning process. The framework of the approach is shown in Figure 6.4. As shown in Figure 6.4 an unlabelled tracker has been input into the system and assigned a label by the current existing classifier. Then the tracker is send to the P/N inspectors, in this process, the tracker will be classified by both inspectors by their own classification model. The P-inspector requires a high threshold, which in this case was set to 95%. In the other hands, the N-inspector requires lower threshold. Thus, the training samples are updated according to the inspector constraints.

To supervise the bootstrapping of the learning classifier, the tracker data were put into the PNSL under the assumption that the labels of the data are known. By comparing the labelled data and the unlabelled data it can directly recognise the mislabelled data and add them to the training sample set with the correct labels. This method is commonly referred to as supervised bootstrapping [137]. Bootstrapping is normally focused on decision making in the classification process, and it often processes randomly training samples. This method used the same idea of focusing on the decision boundary in the PNSL; the difference is that the labels of input data in this method are unknown, which means the process can be defined as standard bootstrapping in the unlabelled data scenario and the labels are decided by the P/N inspectors.

Based on this assumption, the inspectors of the PNSL were analysed as follow. In the classifier f, the errors will be characterised by false positives α (f) and false negatives β (f). Let $n_c^{+}(f)$ be the number of training samples for which the label was correctly changed in the TVD and $n_i^{-}(f)$ the number of samplesfor which the labels that was incorrectly changed in the TVD. The error of the classifier will be:

$$\alpha(\mathrm{f}+1) = \alpha(f) - n_c^{-}(f) + n_i^{+}(f) \qquad (6.3)\ [18]$$

$$\beta(\mathrm{f}+1) = \beta(f) - n_c^{+}(f) + n_i^{-}(f) \qquad (6.4)\ [18]$$

The quality of the checking process is characterised by four measures. **P-true** is the number of correct positive samples divided by the total number of results. **P-false** is the number of correct positive samples divided by the number of false negatives; **N-true** is the number of correct negative samples divided by the number of results. Finally, **N-false** is the number of correct negative samples divided by the total number of false positives. It is assumed that self-learning is characterised by fixed measures throughout the training. The number of correct and incorrect results is then expressed as follows:

$$n_c^+(f) = R^+\,\beta(f),\; n_i^+\,\frac{(1-P^+)}{P^+}\,R^+\beta(f) \tag{6.5}$$ [18]

$$n_c^-(f) = R^-\,\alpha(f),\; n_i^-\,\frac{(1-P^-)}{P^-}\,R^-\alpha(f) \tag{6.6}$$ [18]

These equations were combined together to get:

$$\alpha(f+1) = (1-R^-)\alpha(f) + \frac{(1-P^+)}{P^+}\,R^+\beta(f) \tag{6.7}$$

$$\beta(f+1) = \frac{(1-P^-)}{P^-}\,R^-\alpha(f) + (1-R^+)\beta(f) \tag{6.8}$$

As set vector $\vec{x}(f) = [\alpha(f)\ \beta(f)]^t$ and $\mathrm{L} = \begin{bmatrix} 1-R^- & \frac{(1-P^+)}{P^+} \\ \frac{(1-P^-)}{P^-} & (1-P^+) \end{bmatrix}$, which can get the final equation:

$$\vec{x}(f+1) = L\vec{x}(f) \tag{6.9}$$ [18]

This equation shows the errors from the classifier during the process. The dynamic system was used to understand the conditions that could cause more errors. Notice that matrix L is the quality measurement and the state vector $\vec{x}$, which converges to zero if both eigenvalues $\theta_1\theta_2$ of matrix L are smaller than 1. Therefore, if the quality measurements are known, this allows us to obtain the status of the learning process.

The approach was tested using the testing videos captured from UAVs. The purpose was to analyse the performance of the PNSL system. The process is started by assigning an initial detector, which is done by the detection element of the system. Then the system evaluates the current detector which can estimate the errors of the detector and update the labels from the detector; then the results are put into the training database.

6.3 Implementation of the system

First, during the tracking process, target vehicles are represented by their trackers, which is a bounding box that estimates each detected vehicles between the sequence frames, also the feature of the first tracker is assigned by the detector in the first frame, and each tracker has two possibilities: indicated by either a bounding box or an empty state (when the vehicle is out of the image). The bounding box is given a fixed aspect ratio by the initial detection results and the box will be updated based on the tracking results. The rotation of the bounding box is not considered.

The tracker estimates the position of a number on points of the target. This method used SIFTS point matching approach to measure the similarity between the tracker and detector. In the original TLD, it is assumed that the targets will always be visible in the videos, so in a real-life scenario, if the target moves out of the camera’s range, the tracker’s search will cause an error. This problem can be tackled using two methods. The first method is a simple approach of setting an edge boundary in the image; if the tracker moves to the edge of the image, which means the target is about to exit the image, the tracker will terminate the searching process and when the detector redetect a new target the tracking will be restart the searching. The second method assumes that the target might be blocked by something in the environment, such as trees, buildings etc. In this case, the target might be disappearing in the middle of the image during tracking, which cannot be processed by using the first approach. In this case, the FBT can solve such problem, which the tracker can be re-assigned to the disappeared target if it reappears and detected by the detector, because the target has been saved in the TVD and the matching process between the detector and the TVD can easily find the feature of this target. This method can also identify failures caused by sudden motion or fast occlusion of the target. When a failure is detected, the tracker will terminate its current tracking and initialise for further tracking.

The detectors and trackers combine their bounding boxes together in the main process and if both tracker and detector are fail to return a bounding box; the system will decide there are no visible targets. Otherwise, the system will measure the similarity between the tracker and the detector. Because detection is applied first, the detectors have the location information of the target first and the tracker located by using the detector information, so the tracker can put the new tracked data into the training database for detections and also the TVD.

Figure 6.5 shows the tracking process when an occlusion problem occurs. The red and green boxes are indicated two different detected targets. During the occlusion, the red vehicle target was blocked by the green vehicle target so in these frames where the occlusion was take place, the red tracker was tracked the other vehicle which belongs to the green tracker, so the red tracked stopped tracking. However, the detection process was not stopped in the following frame, so the blocked target was re-detected by the detector and the tracker belongs to this target was re-assigned to it after the occlusion.

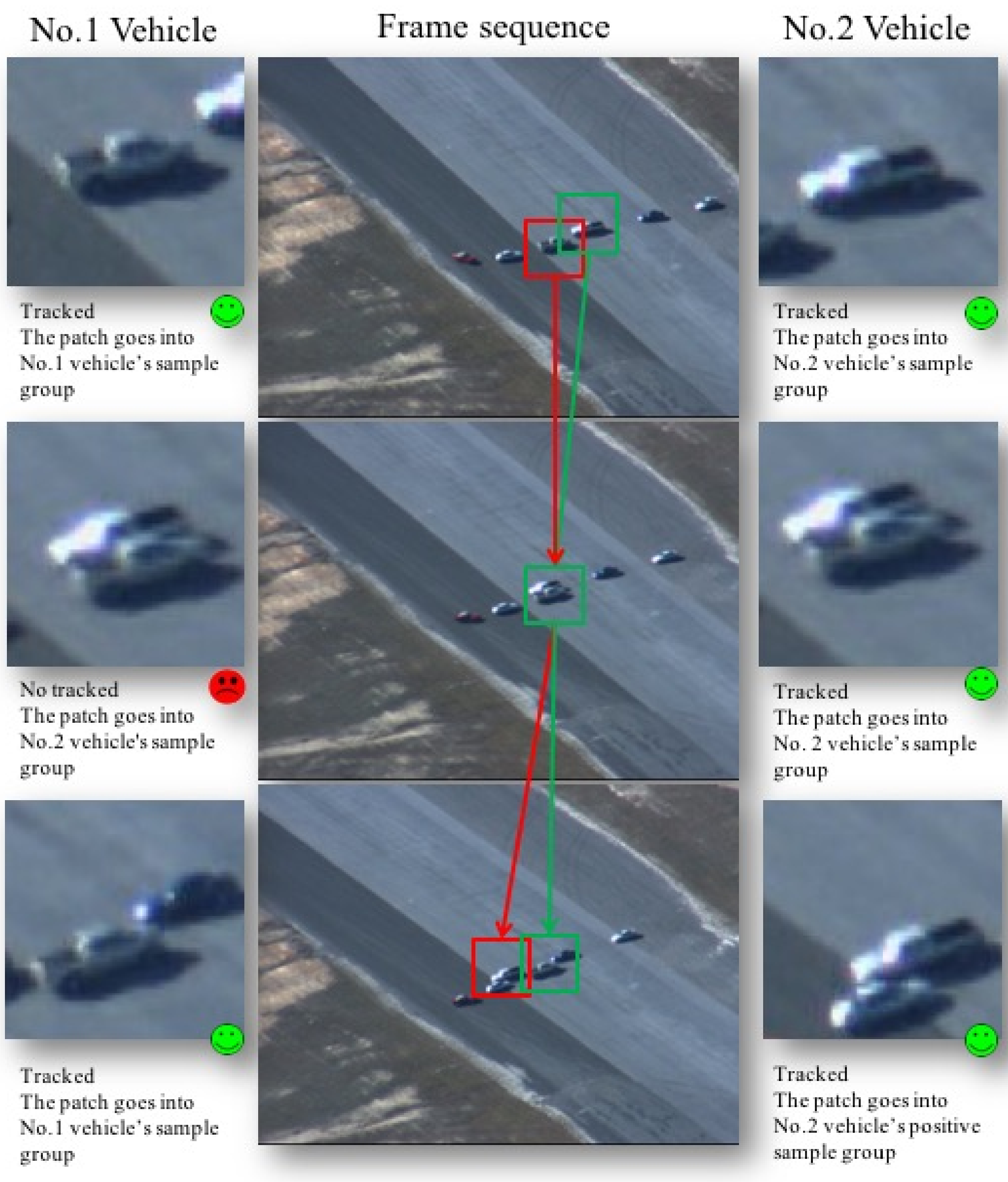

No.1 Vehicle
Frame sequence
No.2 Vehicle
Tracked
The patch goes into No.1 vehicle's sample group
Tracked
The patch goes into No.2 vehicle's sample group
No tracked
The patch goes into No.2 vehicle's sample group
Tracked
The patch goes into No. 2 vehicle's sample group
Tracked
The patch goes into No.1 vehicle's sample group
Tracked
The patch goes into No.2 vehicle's positive sample group

Figure 6.5: An illustration of the example of the tracking process.

According to the evaluation, the main advantage using the FBT is that it can prevent tracking failure in cases where moving targets are moving faster than expected or when the target is temporarily blocked by the environment. Such challenges can be tackled by the region of interest estimations and the feature matching processes of the trackers in FBT, for example, Algorithm 6.1 indicates the process of FBT, each current tracker is compared with the previous and afterwards trackers for the feature similarities so the tracking process can be monitored by the sequences of the frames. The FBT mechanism provides a compression of the tracking results in the tracking sequence and also with the detector by SIFT matching process. Also, with this method, multiple targets can be tracked in the same video.

6.4 Evaluation of the SLTD Approach

In this experiment, several goals were set to test the performance of this approach. The testing was split into:

- Analyse classification performance with the learning component
- Compare tracking performance with and without the learning component
- Analyse the performance of Forward and Backward Tracking
- Test tracking performance in different circumstances

In order to prove that the learning process of SLTD could increase the performance of both tracking and detection, the detector in SLTD was replaced by all three proposed detection methods (HSV-GLCM, ISM-SIFT and FAST-HoG) and tested on each testing videos. In this tracking experiment, the Multiple Object Tracking Accuracy (MOTA) metric [138] was used to measure performance. This provides an accuracy score that takes into account the number of missed detections, the false positive rate and mismatches between the trackers and actual vehicles. Assuming that for frame t the number of missed detections is indicated by $\mathrm{fp_t}$, the number of false positives is indicated by $\mathrm{Tr_t}$ and the number of tracker mismatches using the previous vehicle detection results in frame $(\mathrm{t}-1)$ is indicated by $\mathrm{Tr_t}$, the MOTA can be computed as:

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t) + c_s(Tr_t))}{\sum_{t=1}^{N_{fames}} N_G^{(t)}} \qquad (\ 6.10\ )\ [138]$$

The accuracy and the MOTA value were calculated throughout the tracking process. Two assumptions were made before testing. The first assumption is that, in the detection process, classification accuracy should be increased, because the training data has been updated by the learning process so the classifier model should adapt to the current situation, which will allow accurate detection. The second assumption is that the tracking performance should be better because of the improvement of the detection results.

6.4.1 Classification Performance

Testing was carried on all five testing videos in the data set, which were captured from UAVs above vehicles. A different video was used from the data set to train the detector's classification model; therefore the training sample and the testing video are totally unrelated. During the tracking process, the classification model is updated based on the tracking results in every frame. After every update, the classification model is evaluated based on the testing video sequences to measure its performance by using the F-measure. The performance of the classification and the detector is shown in the Figure 6.6; at the beginning of the process, the classification accuracy was lower than what are expected because the process was applied in a brand new environment, so the vehicles' features are different than in the classification model, but as the process continued, the accuracy of classification increased, showing that the model had adopted the features of the new environment.

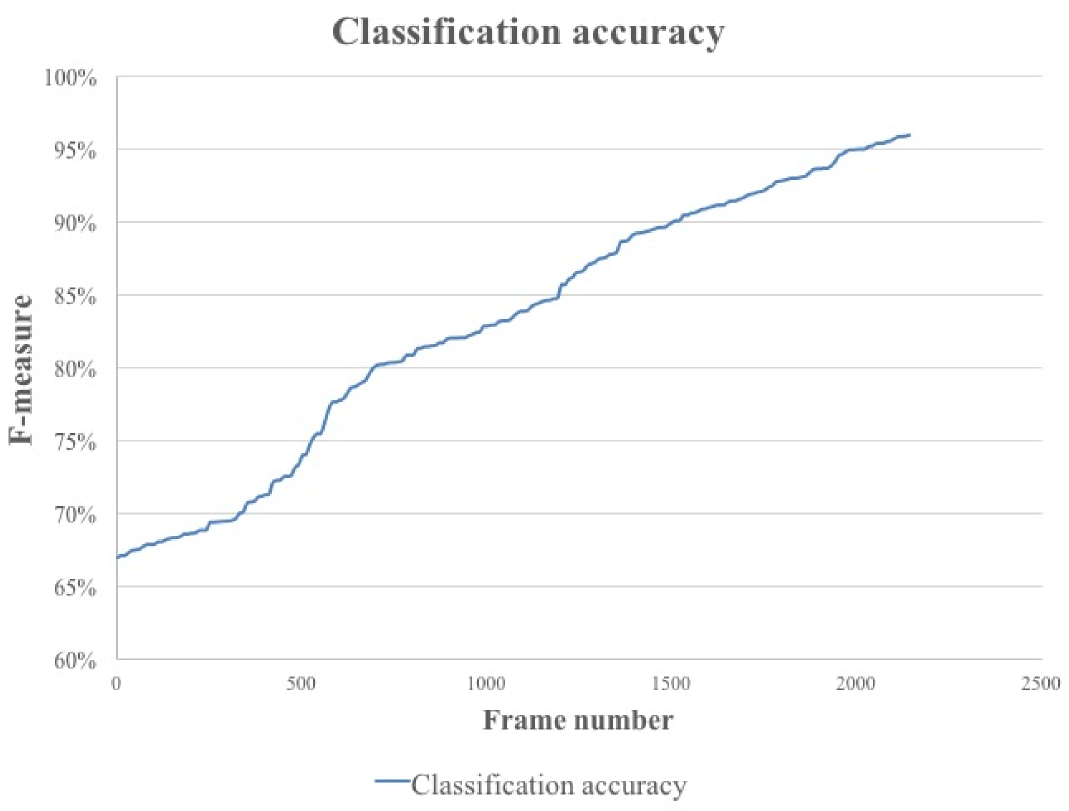


Figure 6.6: A classification accuracy chart for the testing videos. The F-measure value is the average results from all five testing videos.

6.4.2 The Tracking Performance with and without PNSL

In order to test the proposed PNSL approach's performance, the results of the tracking process were compared using the PNSL approach with its performance without the PNSL. In testing without PNSL system, the learning and detector components were removed from PNSL and the system simply predicted the trajectories of the vehicles using the trackers. All five testing videos were used from the data set. To evaluate the overall performance of the system, an experiment was performed to assess the system's ability to accurately detect and track vehicles. In this experiment, accuracy measure for the system was used by the MOTA metric. This is because the MOTA is more suitable for evaluate the tracking performance and the accuracy measurement used in the classification is more suitable for evaluate detection performance [139]. The results are shown in the following figure (Figure 6.7). During testing, 10 vehicles were chosen randomly and tracked in each testing video to get the average MOTA values.

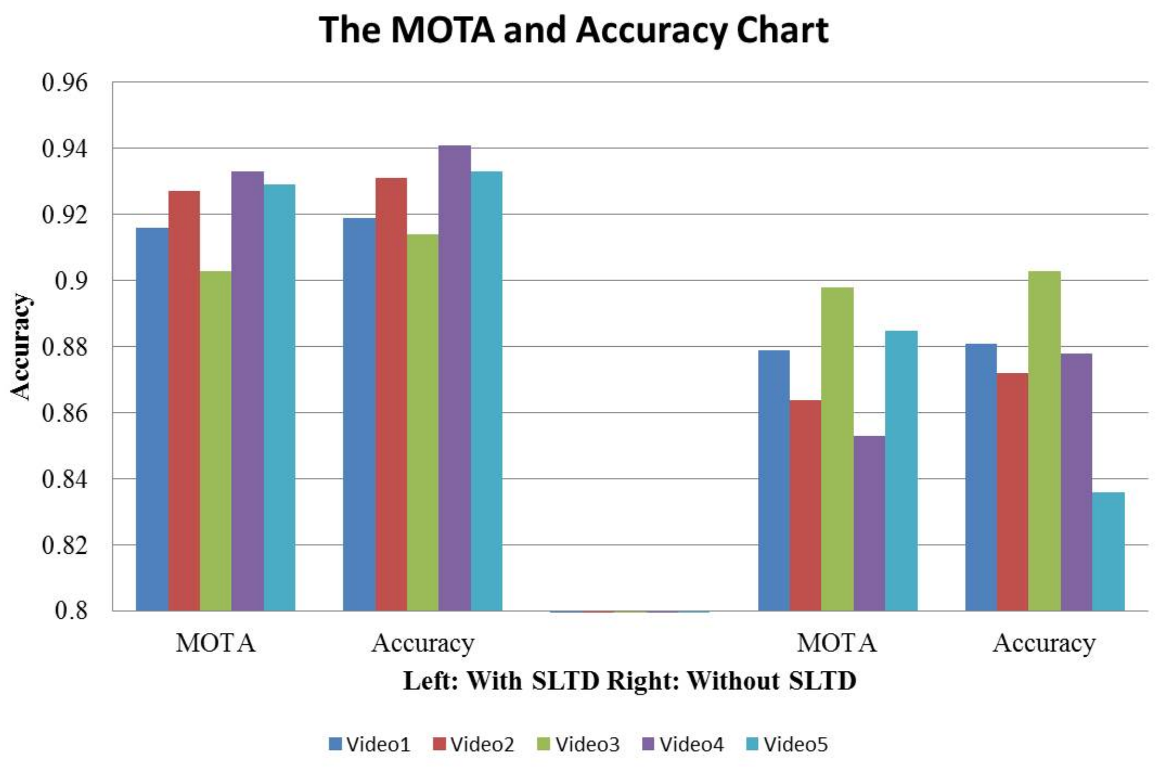


Figure 6.7: Average result of tracking with PNSL and without PNSL.

Using the MOTA metric, it is found that an accuracy of **0.922** was obtained overall for the tracking process that used the PNSL, and an accuracy of **0.876** was obtained through the process without PNSL. According to the results, tracking accuracy was improved by using the PNSL, which means the proposed PNSL can improve tracking performance by adding the learning component.

6.4.3 Forward and Backward Tracking Performance

The performance testing of FBT used Test Video 1, which was captured above a motorway in a suburban area. The video contains a very complex background, full of objects which might have similar features as vehicles, so the detector was likely to get confused between them. In cases like this, the FBT can avoid such problems. Figure 7.4 shows an example of the FBT process. The first tracker continually tracked on vehicle; all matching results were above the threshold. The second tracker, in the middle, mistakenly tracked the neighbouring vehicle towards the end, which was detected by the FBT process; however, the detector considered the patch positive, and another tracker was already tracked this vehicle, so this patch was saved into the neighbour vehicle's database. Finally, the third tracker lost the target in the end; the matching result was very low. In the meantime, the tracker did not find any features similar to those of the last patch, but the detector sent a positive result message to the tracker, which was clearly a mistake. So the detector was corrected from positive to negative by the FBT process, and the patch has been saved as a negative training samples.

Figure 6.8 indicates the average tracking accuracy and MOTA value in all three testing processes. The SLTD+FBT approach had the highest detection accuracy, with more than 90% for every testing video, which means FBT can also improve tracking performance by reducing the detector errors. This process also compared the false positive and false negative rates and found that both errors were reduced dramatically by using the FBT approach (Figure 6.9).

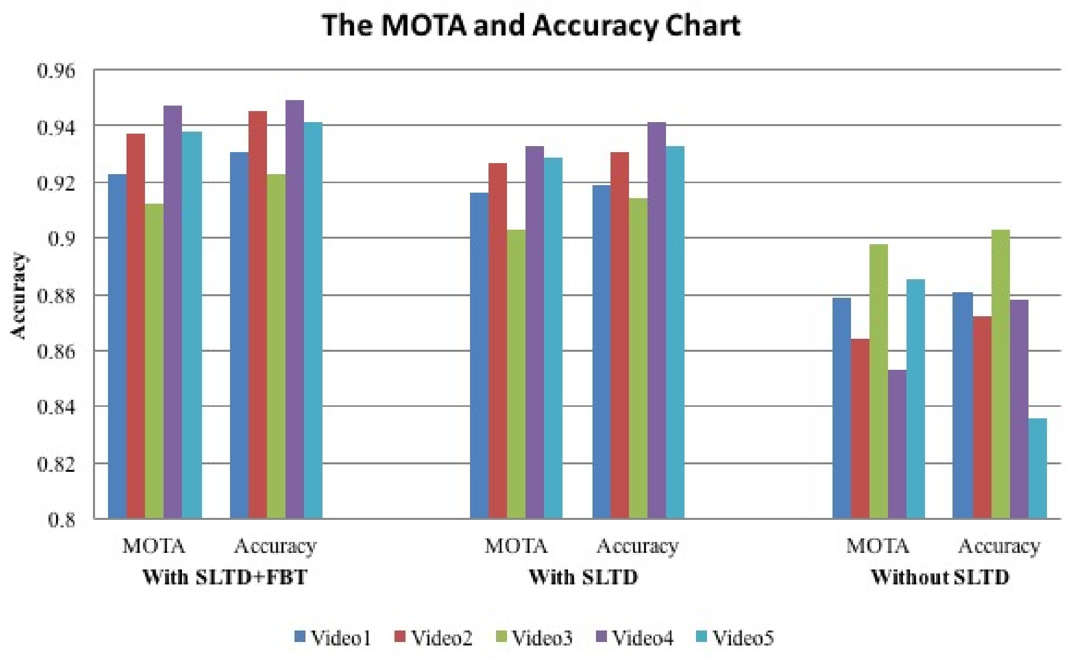


Figure 6.8: A comparison of the tracking results of using different methods.

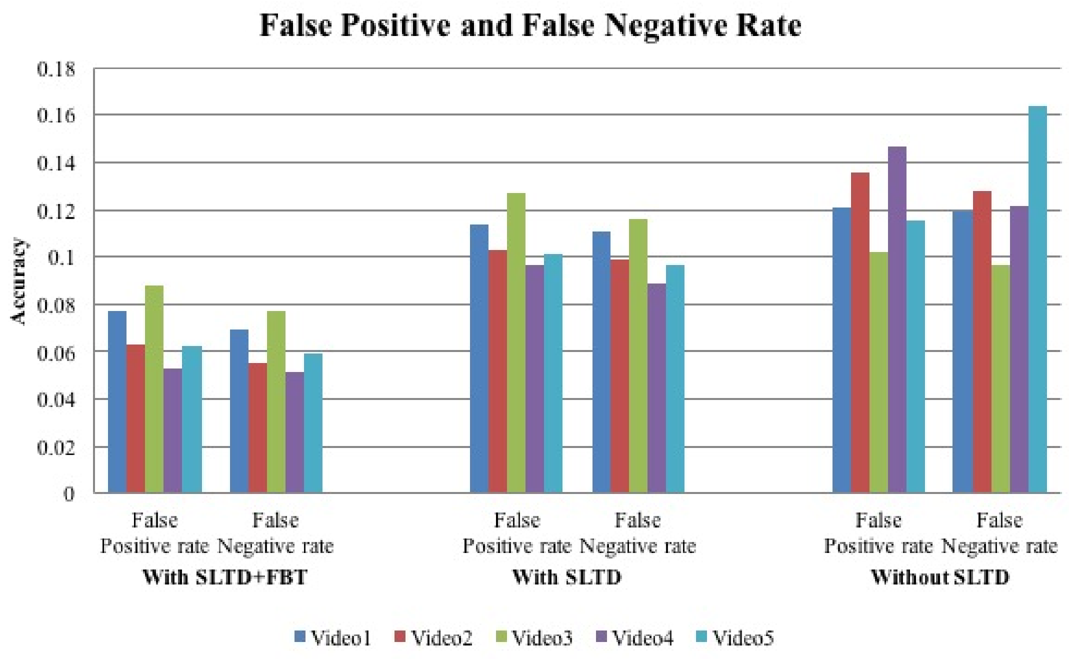


Figure 6.9: The false positive and false negative rates.

6.4.4 Testing Tracking Performance in Different Circumstances

In this test, three videos were selected (video 2, video 3 and video 5) from the data set which contained the challenges of occlusion, changing appearance and blocked vehicles. The reason to use these three videos is that they are the classic challenges in the object tracking process, so it would be helpful to see whether the approach could successfully tackle these challenges.

Figures 6.10 to 6.12 show some example results from this testing, the statistics results can be found in Figure 7.8. Each figure illustrates the moment when the challenge occurred and the results of the tracking process. As shown, most of the challenges were dealt with and the tracking was successful.

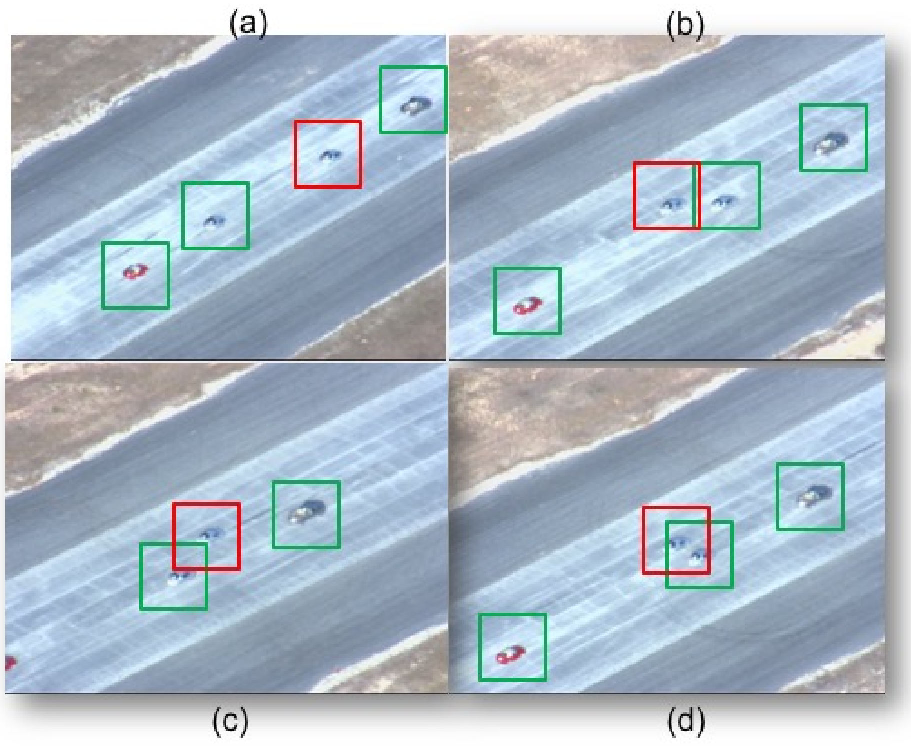


Figure 6.10: Tracking results from a situation in which two similarly-featured vehicles are moving closed to each other. The red boxes are the target tracker, and other trackers are shown as green boxes.

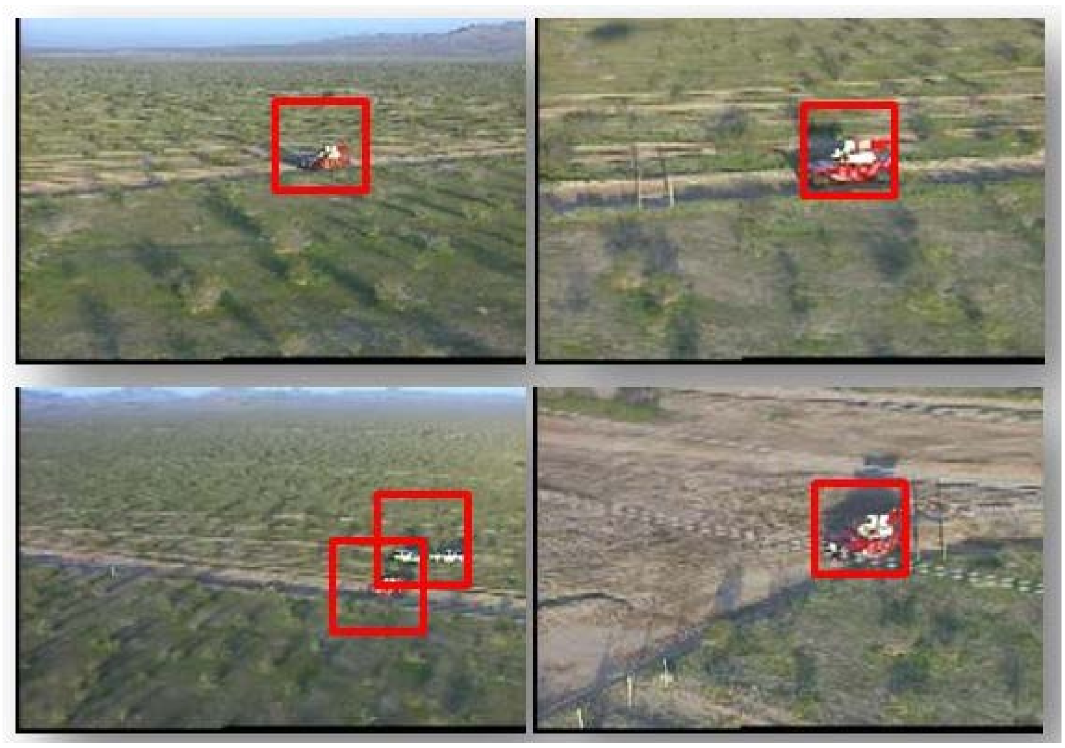


Figure 6.11: Tracking results from a situation in which the target changes appearance and size during tracking.

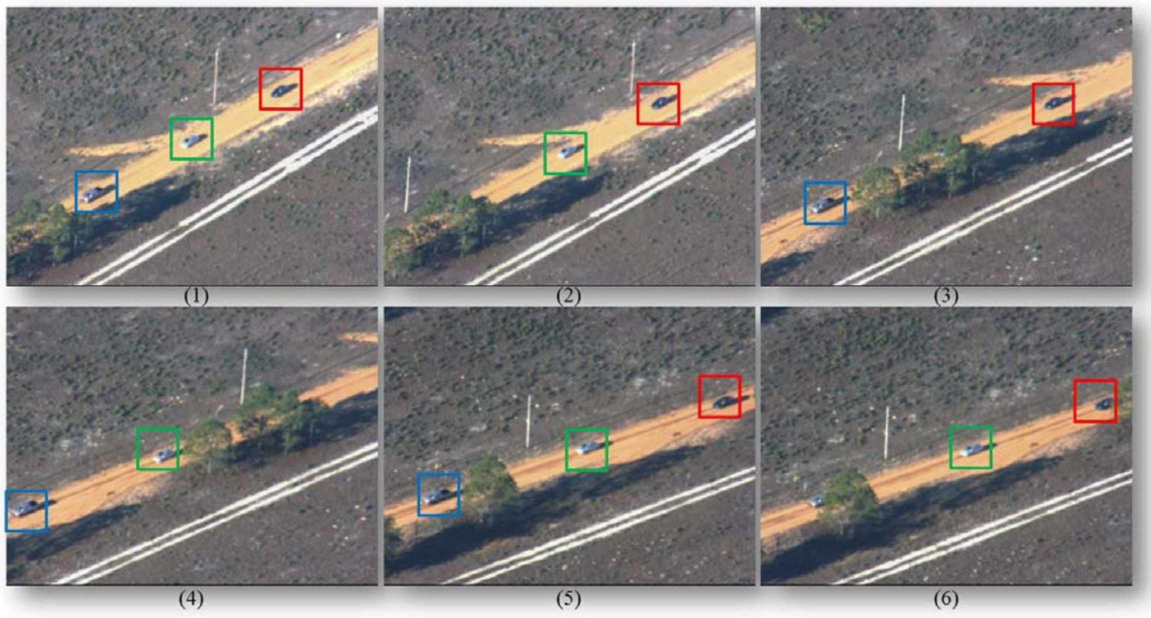


Figure 6.12: Tracking results in a situation in which the target disappears for a short period. Each differently coloured tracker was issued to an individual target.

6.5 Comparison of Tracking Performance

Comparisons were conducted of the tracking results of the proposed method and six other tracking algorithms, including:

- Online Boosting (OB) [140]
- Iterative Visual Tracking (IVT) [141]
- Online Discriminative Features (ODB) [142]
- Multiple Instance Learning (MIL) [143]
- Co-trained Generative-Discriminative tracking (CoGD) [144]
- Tracking Learning Detection (TLD)[18]

First of all, because some of the tracking approaches can only track single targets, the tracking comparison process only focused on a single target vehicle. The testing process was based on the number of frames in which the target vehicle was successfully tracked compared to the total number of frames in which it actually appeared. Table 6.1 shows a comparison of the tracking results. It shows that the proposed SLTD achieved the best score for each video and matched the performance of the original TLD and CoGD. In certain situations, such as occlusions, SLTD performed better than others.

Video	Frames	OB	IVT	ODB	MIL	CoGD	TLD	SLTD
1	367	109	131	76	297	**367**	**367**	**367**
2	255	87	121	93	175	240	240	**248**
3	307	198	226	204	279	281	**290**	**290**
4	108	85	83	95	90	**108**	**108**	**108**
5	207	123	136	153	188	204	205	**207**

Table 6.1: A table of the numbers of successfully tracked frames.

Moreover, MOTA and MOTP (Multiple Object Tracking Precision) metrics have been used to compare the tracking performance in each testing video with different challenges. According to the literature [138], MOTA and MOTP could provide the quality and the main characteristics for evaluating the tracking systems. The MOTP can be extracted as:

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}} \left[\frac{\left| G_i^{(t)} \cap D_i^{(t)} \right|}{\left| G_i^{(t)} \cup D_i^{(t)} \right|} \right]}{\sum_{j=1}^{N_{fames}} N_{mapped}^{j}} \qquad (6.11)\ [138]$$

where $\mathrm{N}_{\mathrm{mapped}}$ refers to the tracked objects during the video in t frames. However, as mentioned before, the TLD approach can only track single target at a time, so it has been excluded in the MOTA and MOTP process. Table 6.2 indicates the tracking results for all 5 testing videos with different challenges, which showing that the SLTD can perform a better tracking result in all challenges.

Video	OB	IVT	ODB	MIL	CoGD	SLTD
	MOTA and MOTP values					
1	0.297/0.276	0.357/0.308	0.631/0.605	0.829/0.803	0.903/0.887	**0.927/0.903**
2	0.348/0.320	0.479/0.468	0.482/0.457	0.697/0.672	0.939/**0.928**	**0.941/0.928**
3	0.644/0.631	0.736/0.711	0.614/0.598	0.903/0.875	0.901/0.892	**0.915/0.881**
4	0.787/0.771	0.769/0.752	0.790/0.773	0.882/0.865	0.945/0.910	**0.947/0.918**
5	0.594/0.562	0.658/0.621	0.726/0.709	0.903/0.891	**0.937/0.925**	**0.937/0.925**

Table 6.2: A table of the MOTA and MOTP results.

6.6 Conclusions

This chapter has proposed a Self-Learning Tracking Detection approach, which modified and expanded the Tracking-Learning-Detection (TLD) approach and upgraded a multiple vehicle tracking system, which the original TLD can only track a single target. P-N learning was applied in the learning component to monitor the tracking and detection processes. In addition, the classification model of the detector was linked to the learning process so the model could be updated based on the tracking results. A Forward and Backward Tracking (FBT) mechanism was also proposed to monitor the learning process, so the tracking process and detection process is monitored by each other. The test results showed that tracking accuracy was improved by both FBT and SLTD. The classification model of the detector can also be adjusted by adding more training samples when the system is applied in a new environment.

The detection process takes place at the beginning, the corners in input image are detected by the FAST algorithm and the area around each corner point has selected as the region of interest. The regions have extracted by the HoG features. Before the detection, a SVM classifier is created by the HoG extraction from the training samples. Then the SVM model will decide whether the region of interest is a vehicle or not. Once the SVM gives a positive result, which means a vehicle has detected, then the detection results is send to the tracking process. In the tracking process, the current tracker is compare with the trackers from previous and afterward frames, the matching process can decide if the current tracker is tracking a same vehicle, a different vehicle or not a vehicle. The tracked vehicles are sand to the Tracked Vehicle Database. Finally, the tracking results are sent to the learning component, consider the current tracking result is an unlabeled data and it has been assigned a label by the classifier of the tracking. Then this label will be check by the P/N inspectors and the results will be stored as either positive samples or negative samples which are used as the training samples to update the classification model for the detection.

Chapter 7

Conclusions and Future Work

This thesis investigated vehicle detection and tracking from UAVs. Unlike many existing approaches, the detection and tracking processes was linked together and let them monitor each other. This thesis has pointed out the challenges and showed that the proposed approaches can solve them.

This main work includes three detection approaches and one tracking approach. Several existing detection and tracking methods were adapted, and a number of modifications, extensions, and verification stages to these methods have been proposed. The proposed algorithms increase both detection and tracking performance. This method particularly focused on the mutual monitoring between the tracking and detection processes, which let the system learn the appearances of targets and check errors during the process.

Significant progress was made during work on this thesis in the development of automatic vehicle detection and tracking which offers reliable results. Additional steps are necessary to further increase the performance of detection; the classification model could be improved by using more featured positive and negative training samples. Understanding more about the background and geometric shape of objects in the environment could also improve performance, because the target and background could be differentiated better.

7.1 Discussion of Contributions

The main contribution of this thesis is that it offers a way to make the vehicle detection and tracking process more robust in different situations.

Firstly, this thesis proposed a HSV-GLCM vehicle detection approach. This method is based on the texture and colour features of vehicles. It adjusted the parameters of GLCM for vehicle features and modified the original GLCM feature, which is a second order texture calculation. Multiple order calculation was applied to feature extraction, so the processed texture pixel not only interacts with neighbour pixels, but also consider a further level of pixels, which can put more feature information in the descriptors. It also added the colour feature to the descriptors, which can identify background features as distinct from vehicles. This is because the colour features of background objects are sometimes very similar to those of vehicles. However, this method was able to distinguish objects such as trees, buildings and road lights etc. by their colour features. The proposed method could increase the classification accuracy of the detection, which tackles the complex background challenge in object dejection. Furthermore, this method integrated a de-blur component to boost the detection accuracy for the low resolution and blur challenge, the results indicate the accuracy have increased by apply the de-blur component. This approach achieved detection results of 90.09% of accuracy and 88.68% of F-measure value in different detection situations. Also, the approach was compared with similar detection methods from the literature which used the PLS Hough transform approach and the results showed that the performance was higher than that of the PLS Hough transform, the accuracy of which was 77.02% and a 74.82% F-measure value.

This thesis also proposed an ISM-SIFT detection approach. The traditional ISM approach is mostly used in people detection because it has strong shape descriptors. ISM directly uses grey level images for the process, which are strongly influenced by illumination changes and noise, which is a disadvantage in vehicle detection using UAV footage. The SIFT point descriptor has the advantages of scale, illumination, transformation and rotation invariance and finds more specific information for each point. Thus this approach can tackle the different size and orientation; complex background; low resolution and blurred challenges. The SIFT algorithm was used in matching process and showed a great performance. Unlike the matching process, which is the comparison between two similar images, vehicle detection needs more unique features from vehicles in order to classify the target as distinct from the background, otherwise points from the background that have similar descriptors as vehicles will be misdirected. Bag-of-Words approach was used to narrow down the detection candidate patches. However, the detection results were still unreliable. So the SIFT and ISM features were combined together in order to detection vehicles on complex backgrounds. The proposed approach used the vehicle’s unique rectangular shape feature to extract information from the input images. In addition, based on the SIFT point, the centres of vehicles can be predicted in order to detect the vehicles correctly. The approach achieved 94.69% detection accuracy and a 94.34% in F-measure value, which is higher than the SIFT approach, with 80.87% detection accuracy and a 78.55% F-measure value.

The third detection approach is the FAST-HoG approach. FAST is a corner detection algorithm which is very high efficient at in the detection. HoG is a classical detection descriptor which is mostly used in human detection. The HoG feature has a strong edge extraction performance, which is a great advantage in vehicle detection because of the shape of vehicles. However, the HoG feature is not rotation-invariant, so its detection is very sensitive to the direction of the target. This problem was solved by adding an orientation feature into the training samples for classification models. The proposed approach also improved the efficiency of the detection process compared with the classical HoG approach because we expedite detection by identifying the image regions that are most likely to contain vehicles. This is done using the FAST corner process, and proposed a density estimation technique to narrow down the candidate patches. In the classification stage, the system uses shape encoding local features and robust machine learning techniques to perform efficient vehicle detection. As same as the ISM-SIFT approach, this approach can tackle most of the research challenges. To evaluate the proposed system we performed a number of experiments to test both the overall system's performance and the comprising stages individually. To evaluate the region of interest selection stage, the detection rate was tested across the entire video data set; this stage performed very well and we achieved a detection rate of 98.3%. To evaluate the classification stage, it assessed its ability to correctly classify the regions of interest selected in the previous stage; to do this the classification accuracy was evaluated over a subset of the data set. This stage performed well and achieved an F-measure value of 78%, which indicates a fairly high classification rate. Finally, this approach achieved higher detection accuracy than the bLPS-HOG approach (94.20% and 93.77 against 80.16% and 78.25%).

Finally, a Self-Learning Tracking Detection (SLTD) approach was proposed. This approach solved a tracking and detection problem, and training samples were taken from other videos rather than the actual testing video. The appearance of a vehicle is affected by the altitude, speed and image resolution of the UAV filming it. This system can learn vehicles' features and create a unique detection model for each testing video during the tracking process. A Forward and Backward Tracking (FBT) approach was also proposed, which can check the errors made by the tracking and detection process and allow the system to avoid them in subsequent tracking processes. The proposed system demonstrates a reasonably high accuracy and is capable of successfully detecting and tracking a variety of differing vehicle types under varying rotation, sheering and blurring conditions, especially it can tackle the occlusion challenge in the object tracking research. The Multiple Object Tracking Accuracy (MOTA) measure was used on the system and other tracking approaches. Note that this approach was inspired by the Tracking-Learning-Detection (TLD) approach; the compared tracking results have shown that this approach can achieve higher tracking accuracy than the TLD approach under certain complex circumstances, such as occlusions and blocked vehicles on a complex background. Also the proposed SLTD can track multiple vehicles rather than only one achieved by TLD. Intensive experiments have been conducted to compare our proposed approaches in vehicle detection and tracking with other approaches in the literature and demonstrated the better performance of our approaches

7.2 Perspectives and Improvements

The proposed algorithms have achieved good detection and tracking results in different situations. However, there are still lots of modifications and extensions that can be applied to improve the robustness of their detection and tracking.

In the HSV-GLCM approach, it is found that errors always occurred on certain unique artificial objects, such as buildings, car park markers etc. This is because these objects have similar texture and colour features to vehicles. This issue could be solved by generating a better training classification by adding negative samples similar to these objects. In addition, although this method can tackle most challenges, the different size challenges can be a weakness. To improve this weakness, detect window can be more flexible depends on the altitude of the UAV in which the target size in the image is changeable. This requires the system extract the basic flying status from the UAV.

The real-time process capabilities of the ISM-SIFT system could be improved. Each SIFT point has a 128-bytes descriptor, which requires a lot of processing resources that can directly affect the real-time process performance. This could reduce the SIFT descriptor from 128-bytes to a 64-bytes descriptor or lower; lowering the descriptor dimension without losing the feature abilities is an interesting research area.

We could also investigate the performance of additional shape encoding features such as the Gradient Location and Orientation Histogram (GLOH) and the Local Energy based

Shape Histogram (LESH) with the FAST-HOG approaches. The Region of Interest selection process can be further optimised by parallelising the evaluation of the Support Vector Machines that comprise the classifier. In addition, a density-based clustering method could be used to reduce the total number of regions presented to the classifier. Training the system with a larger, more diverse data set that has a greater variety of vehicles and background appearances, would likely result in the improved generalisation of the classifier and better performance from both the detection and the tracking aspect of the system.

In the SLTD approach, the learning component could be improved by adding more different experts to upgrade the checking system between the detector and tracker. The initial tracker and detector are directed based on the previous detection results, so the system can monitor any errors made by the tracker or the detector in the following process. However, the number of failed tracking results between the first tracker and the frame in which the system realises the error has occurred can be reduced. It is found that sometimes the tracking system takes a long time to find an error. It can be conclude that using more, stronger inspectors in the checking process could reduce this effect. Finally, the detection and tracking approach should have the real-time processing ability, which requires low computational resources.

Finally, as the detection and tracking technology has been developed rapidly, new feature extractions will be proposed which can be more accurately to describe the artificial objects. Such as the extension of the SIFT features ASIFT, PCA-SIFT, etc. These new feature descriptions can be applied to generate more accurate detection and tracking approaches.

Appendix

List of Publications

1. Xiyan Chen, and Qinggang Meng. "Vehicle Detection from UAVs by Using SIFT with Implicit Shape Model." In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pp. 3139-3144. IEEE, 2013.

2. Xiyan Chen and Qinggang Meng. "Self-Learning Vehicle Detection and Tracking from UAVs." In International Conference on Robotics and Computer Vision (ICRCV 2015), International Journal of Mechanical Engineering and Robotics Research, Vol. 5, No. 2, pp. 149-155, April 2016. DOI: 10.18178/ijmerr.5.2.149-155

   Excellent oral presentation Awarded

3. Xiyan Chen and Qinggang Meng, "Robust Vehicle Tracking and Detection from UAVs." In *IEEE International Conference on Soft Computing and Pattern Recognition (SoCPaR Japan 2015)* pp. 241-246. IEEE, 2015

Testing Videos

Most vehicle detection and tracking systems are evaluated using the videos captured from the front or back of the other vehicles. In this thesis, all testing videos were captured from UAVs above vehicles. In order to evaluate the performance of the proposed method, testing videos were also captured in different circumstances that involved certain challenging scenarios.

Training and Testing Video Data Sets

In order to evaluate the performance of the various vehicle detection and tracking algorithms proposed in this thesis, five testing scenarios were considered:

- Scenario 1: Complex background
- Scenario 2: Occlusion of vehicles
- Scenario 3: Vehicles blocked by objects in the environment
- Scenario 4: Blurred images
- Scenario 5: Changes in vehicle size and appearance

Based on these scenarios, five video sets were used for each challenge, and one video set was used for training purposes. All video sets in these scenarios contained natural images recorded in realistic environments by UAVs above the vehicles. These video sets were used in the performance evaluation of every proposed detection and tracking method in the following chapters.

Training Video Set

The training video set was captured above a motorway in a suburban area by a UAV (Figure A1). The vehicles were captured clearly from above and the UAV flew constantly along the motorway in one direction. The resolution of the video was 720×576 pixels. For training, the video was transformed into frame images, of which there were 1,609 in total; 97 vehicles appeared in these images.

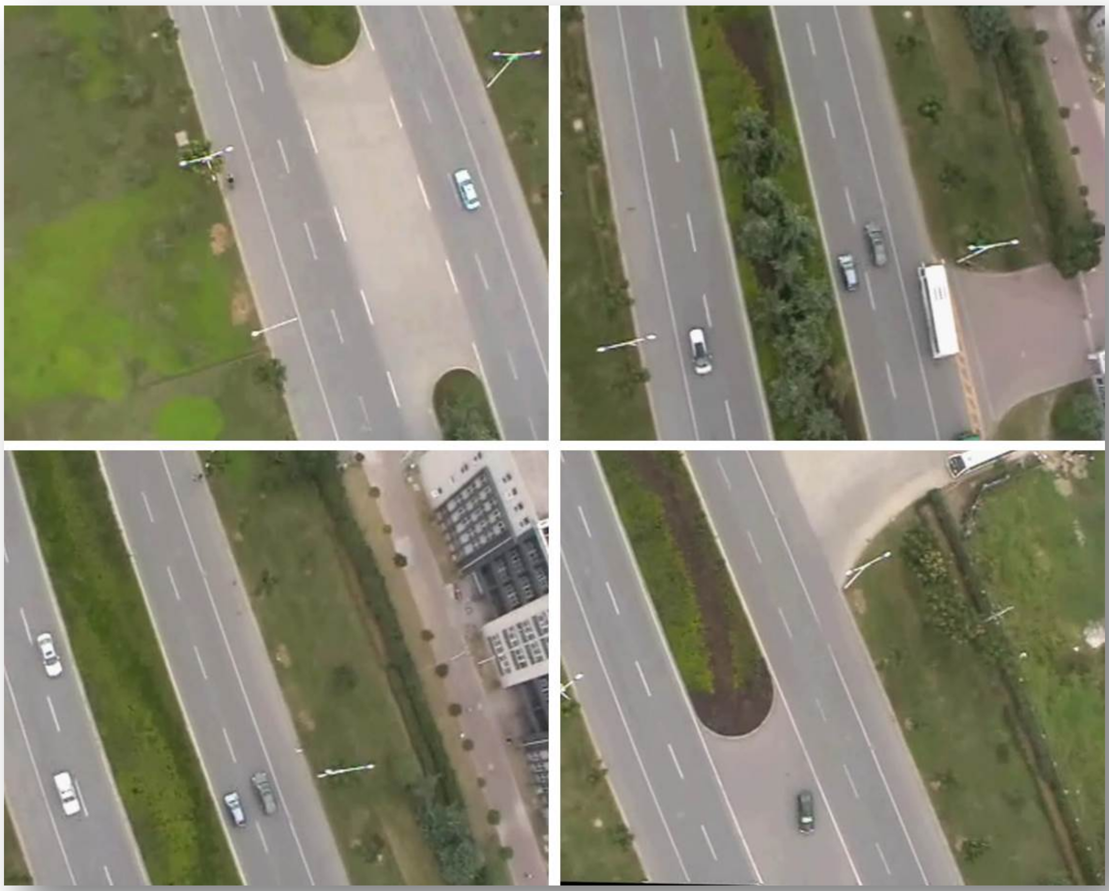


Figure A1: Examples from the training set images.

For the positive training set, each vehicle was cropped into a 70×70 pixel image patch from the video frames. The negative training set was selected from objects in the environment such as road markings, trees, buildings, etc. Figure A2 shows examples of both positive and negative samples.

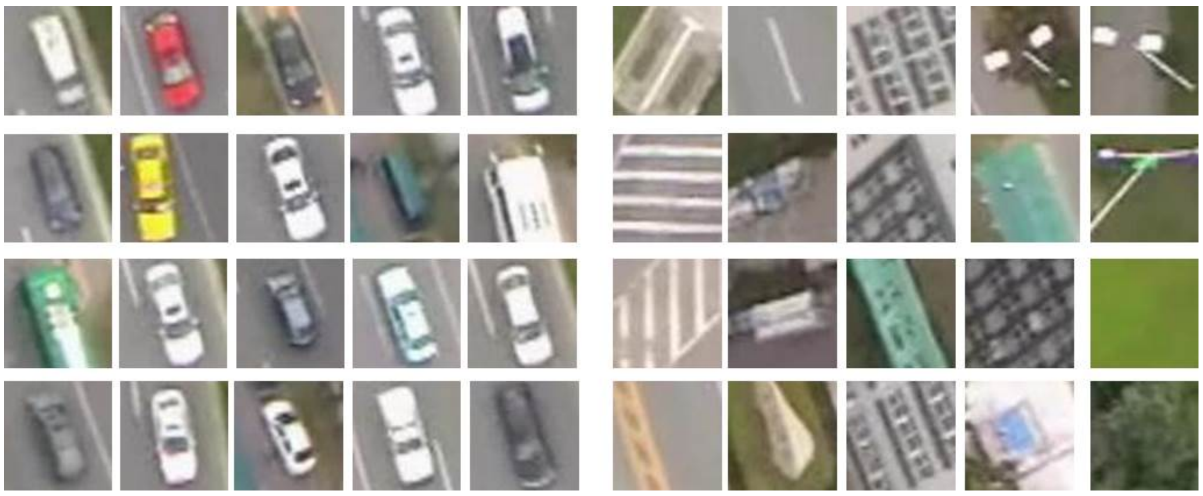


Figure A2: Examples of training samples (left: positive; right: negative).

Testing Video Set 1

Testing video set 1 is very similar to the training video, as it comes from the same video sequence. The first half of the video was used for training purposes and the rest of the video was used for testing. This video was captured directly above a public motorway in a suburban area that contains lots of artificial objects such as street lights, bus stops and buildings. It also contains complex backgrounds, trees, grasses and lots of road markings. As a result, this video was considered a good test for complex background scenarios. This testing video is called “video 1”. Figure A3 shows some examples of video 1. The resolution of this video is 720×576 pixels and it consists of 2,461 frames in total.

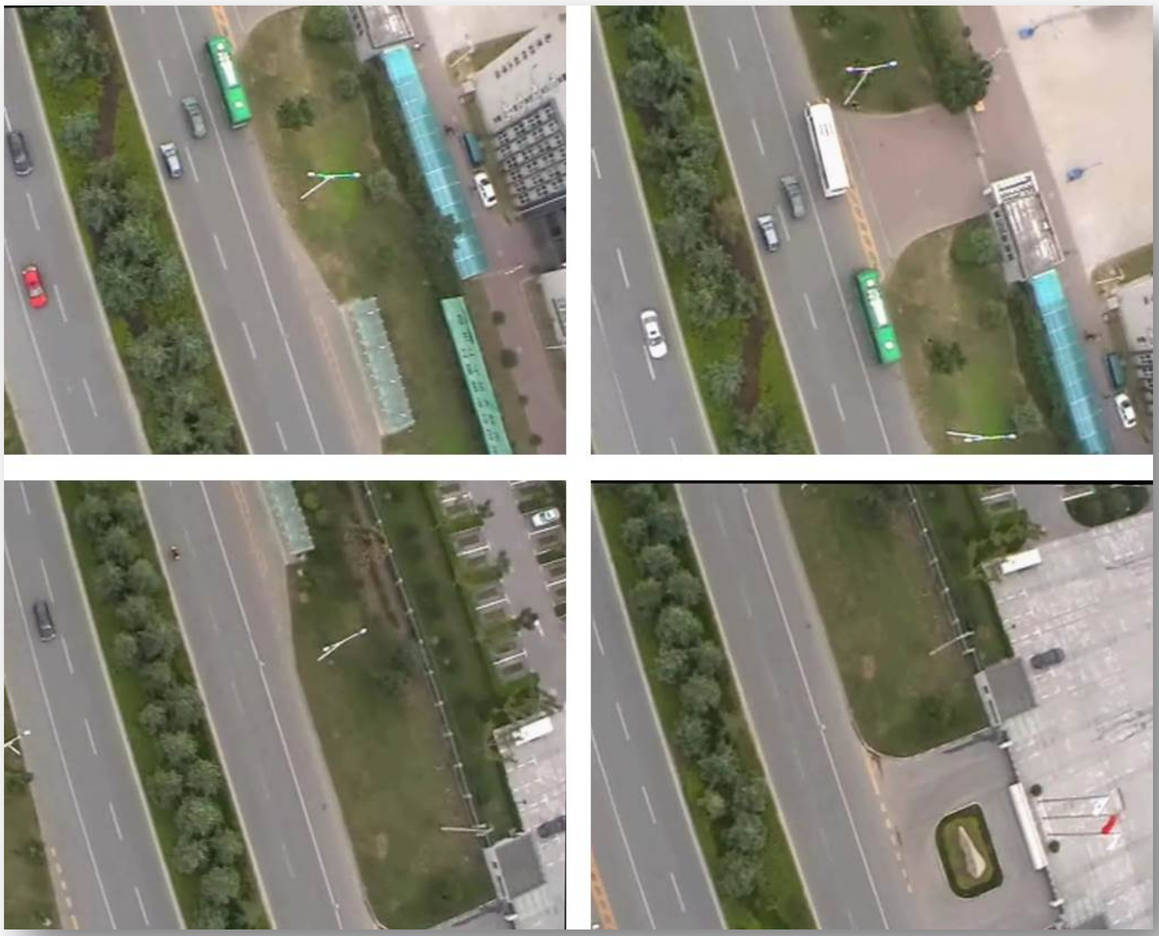


Figure A3: Some examples of video 1 (complex background)

In the top-left image of Figure A3, the vehicles not only appear on the road but are also located near the building. In these circumstances, vehicles are difficult to detect because they are much closer to the environment, so the detecting windows will inevitably contain negative texture. This issue could directly cause false negative errors in the detection process. In the bottom-left image, there are several parking space markers in the top right, which could easily cause false positive errors because of their similarities to the actual vehicles.

Testing Video Set 2

Test video set 2 was downloaded from the Defense Advanced Research Projects Agency (DARPA) website [145]. The video was captured from above vehicles on a wide country road. This video sets the challenge of the occlusion problem (Figure A4). In Figure A4, there are six vehicles, lined up horizontally in two groups, moving towards to each other. This scenario can bring difficulties to both the detection and the tracking process, because the gap between the vehicles is very small, so in the view of the camera, these vehicles have joined together. Also, the vehicles on the far side are blocked by the vehicles in the front, which can be considered another occlusion challenge. The resolution of this video is 640×480 pixels and it contains 1,301 frames. Six vehicles appear in the video.

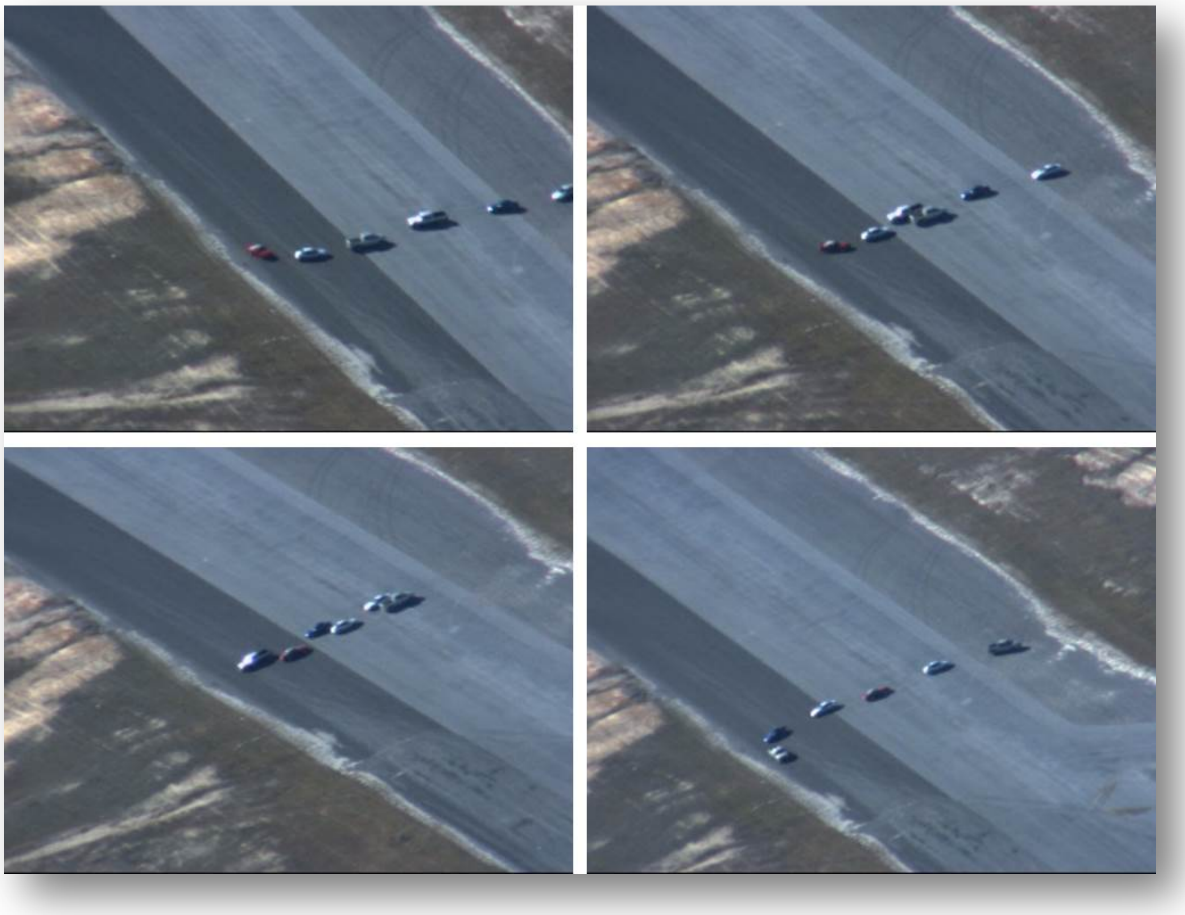


Figure A4: Some examples of testing video 2(occlusion problem).

Testing Video Set 3

Testing video set 3 was also downloaded from the DARPA [145]. This video was selected to represent the scenario of blocked vehicles. Figure A5 shows some example frames from this video. Three vehicles appear in the video and are then blocked by the trees at the side of the road. Each vehicle disappears from the video for a certain period. This video can test tracking performance when the target is lost in the image during tracking, i.e. whether the tracker can re-track the same target when it reappears. This video has a resolution of 640×480 pixels and contains 1,833 frames in total.

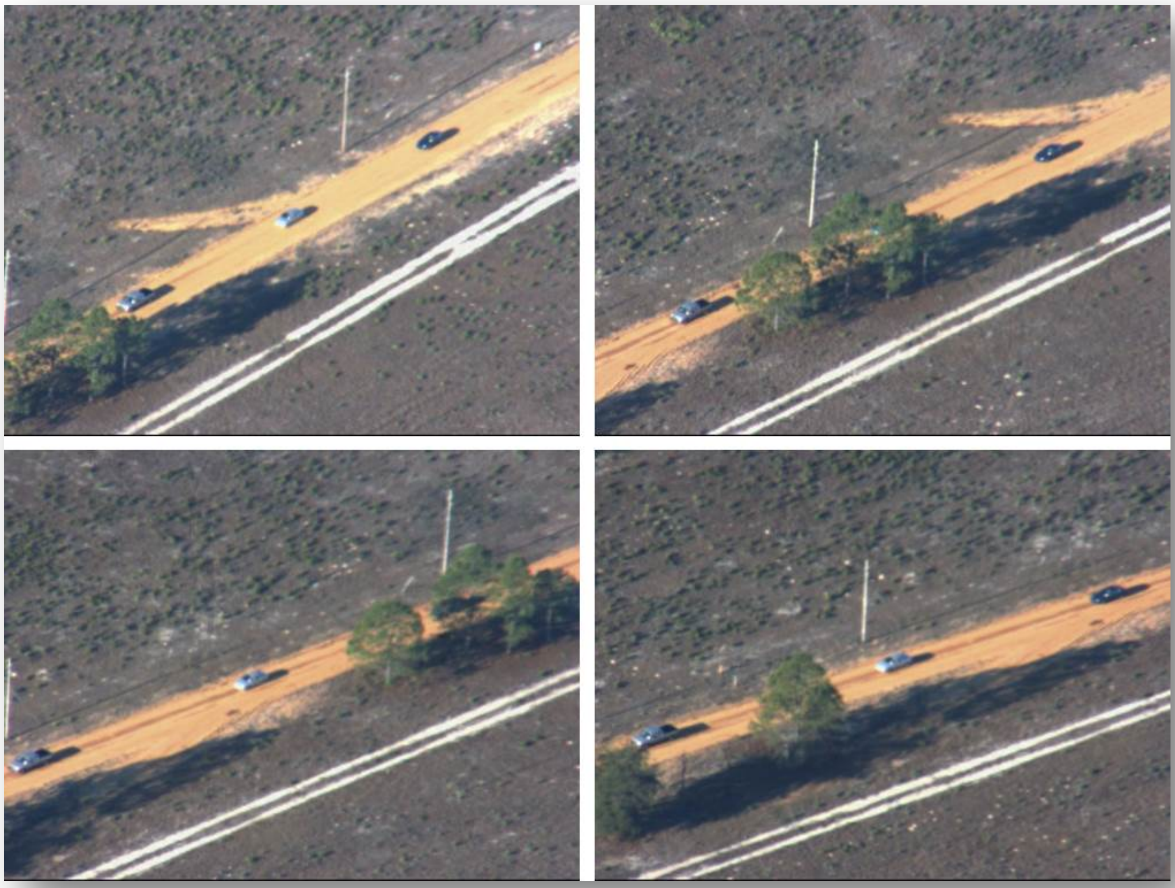


Figure A5: Some examples of testing video 3 (blocked vehicles).

Testing Video Set 4

Testing video set 4 was captured by a UAV above a highway. The UAV hovered overhead at a very high altitude while capturing the video. The images are blurred because the UAV was shaking badly. Figure A6 shows that most of the images are blurred, which could change the texture of the vehicles. The challenge of this scenario is to test detection and tracking performance when the testing images have been transformed. This video was designated "video 4". It has 168 frames in total and a resolution of 720×406 pixels.

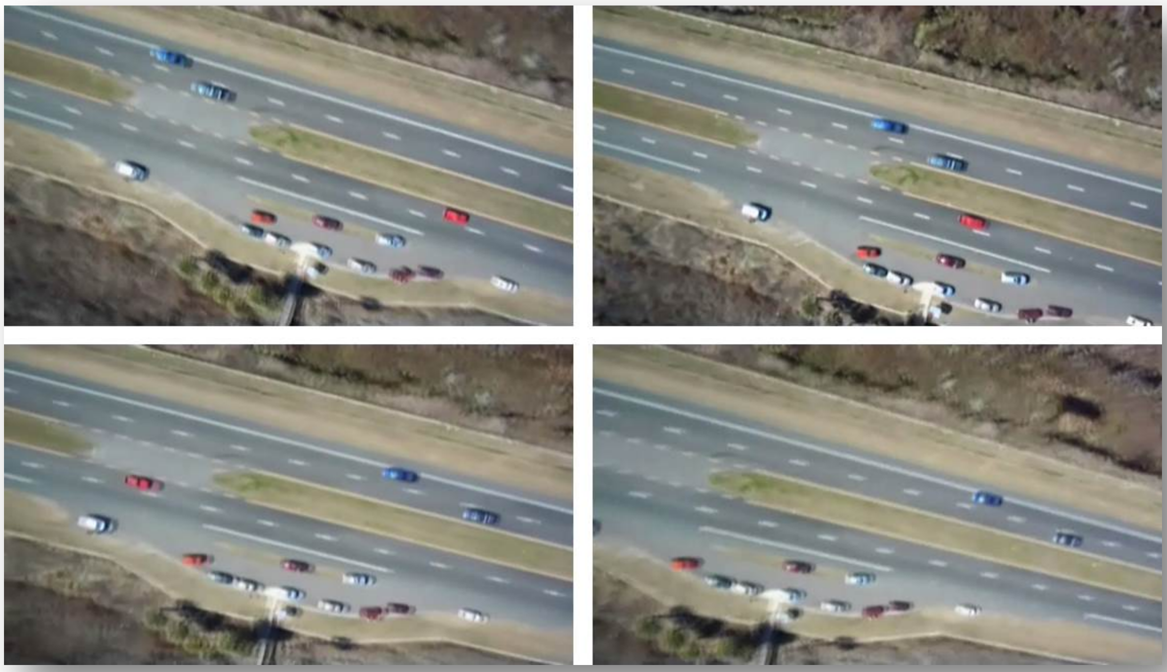


Figure A6: Some examples of testing video 4 (blurred image).

Testing Video Set 5

The final testing video was also downloaded from the DARPA [145]. Only one vehicle appears in this video, however, the UAV is chasing the vehicle from different angles and at different altitudes, so the appearance and size of the vehicle continuously change during the video. This is excellent data for testing tracking performance when the target keeps changing status; the tracking system has to adapt and learn the target's appearance. Figure A7 shows some examples of this video. The resolution of this video is 352×240 pixels and only one vehicle appears in the entire video of 1,918 frames.

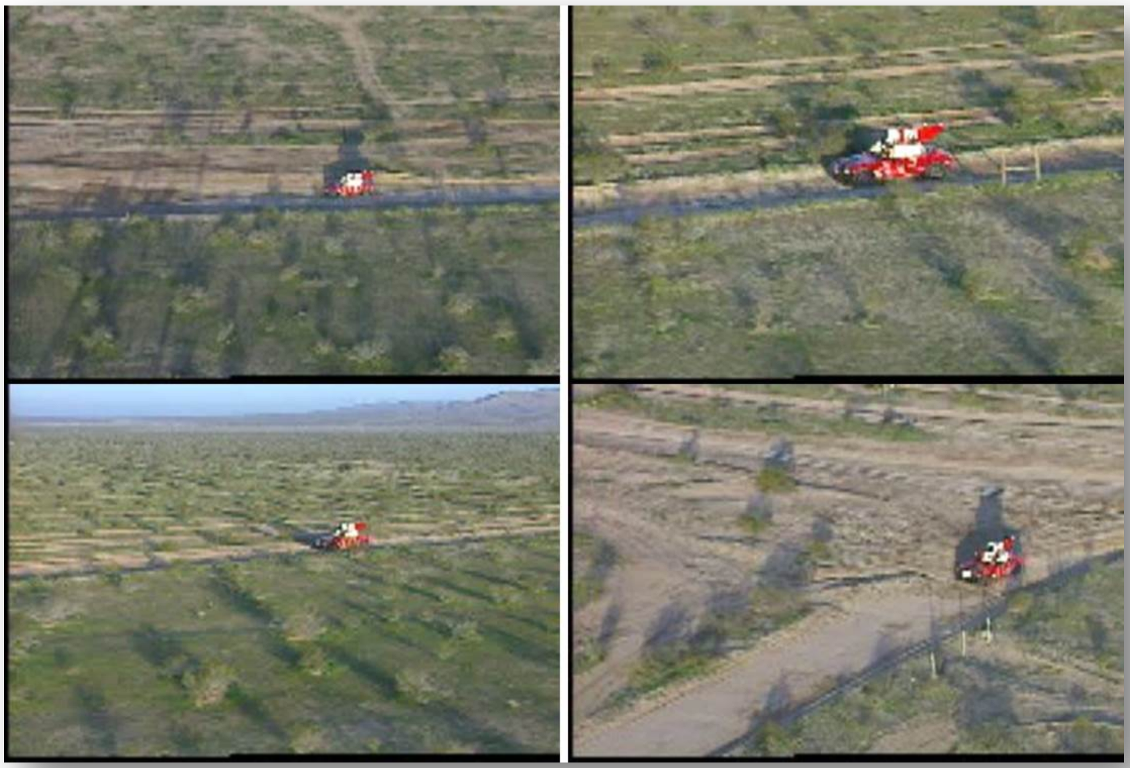


Figure A7: Examples of testing video 5 (changing appearances)

Reference

[1]. T. Scheve, 'A Brief History of UAVs', *How Stuff Works*, 2015. [Online]. Available: http://science.howstuffworks.com/reaper1.htm. [Accessed: 25- Sep- 2013].

[2]. J. Taylor and K. Munson, *Jane's pocket book of remotely piloted vehicles*. New York: Collier Books, 1977.

[3]. Boston.com, 'Earthquake in Yushu, China', 2015. [Online]. Available: http://www.boston.com/bigpicture/2010/04/earthquake_in_yushu_china.html. [Accessed: 09- Feb- 2014].

[4]. Chen, Ziyi, Cheng Wang, Chenglu Wen, Xiuhua Teng, Yiping Chen, Haiyan Guan, Huan Luo, Liujuan Cao, and Jonathan Li. "Vehicle Detection in High-Resolution Aerial Images via Sparse Representation and Superpixels."*Geoscience and Remote Sensing, IEEE Transactions on 54, no. 1* (2016): 103-116.

[5]. Ghaffarian, Saman, and Ilgın Gökaşar. "Automatic vehicle detection based on automatic histogram-based fuzzy C-means algorithm and perceptual grouping using very high-resolution aerial imagery and road vector data." *Journal of Applied Remote Sensing 10, no. 1* (2016): 015011-015011.

[6]. Lu, Qingbo, Wengang Zhou, Lu Fang, and Houqiang Li. "Robust Blur Kernel Estimation for License Plate Images from Fast Moving Vehicles." (2016).

[7]. Sincha, Dmitry, Mikhail Chervonenkis, and Pavel Skribtsov. "Vehicle Detection in Aerial Traffic Monitoring." *American Journal of Applied Sciences13, no. 1* (2016): 46.

[8]. Ramon, P., Begoña C. Arrue, J. J. Acevedo, and A. Ollero. "Visual Surveillance System with Multi-UAVs Under Communication Constrains." In *Robot 2015: Second Iberian Robotics Conference*, pp. 705-713. Springer International Publishing, 2016.

[9]. Bein, Doina, Wolfgang Bein, Ashish Karki, and Bharat B. Madan. "Optimizing Border Patrol Operations Using Unmanned Aerial Vehicles." In *Information Technology-New Generations (ITNG), 2015 12th International Conference* on, pp. 479-484. IEEE, 2015.

[10]. Kurdi, Heba, and Jonathon How. "Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions." In *AIAA Guidance, Navigation, and Control Conference*, p. 1377. 2016.

[11]. Ugliano, M., L. Bianchi, A. Bottino, and W. Allasia. "Automatically detecting changes and anomalies in unmanned aerial vehicle images." In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, 2015 IEEE 1st International Forum on, pp. 484-489. IEEE, 2015.

[12]. Hammoud, Riad. "METHOD AND SYSTEM FOR DISMOUNT DETECTION IN LOW-RESOLUTION UAV IMAGERY." U.S. Patent 20,160,078,272, issued March 17, 2016.

[13]. Tang, Swee Ho, Takaaki Kojima, Toru Namerikawa, Che Fai Yeong, and Eileen Lee Ming Su. "Unscented Kalman filter for position estimation of UAV by using image information." In *Society of Instrument and Control Engineers of Japan (SICE), 2015 54th Annual Conference of the*, pp. 695-700. IEEE, 2015.

[14]. Quintero, Steven AP, Michael Ludkovski, and Joao P. Hespanha. "Stochastic Optimal Coordination of Small UAVs for Target Tracking using Regression-based Dynamic Programming." *Journal of Intelligent & Robotic Systems* (2015): 1-28.

[15]. Ingersoll, Kyle, Peter C. Niedfeldt, and Randal W. Beard. "Multiple target tracking and stationary object detection in video with Recursive-RANSAC and tracker-sensor feedback." In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference* on, pp. 1320-1329. IEEE, 2015.

[16]. Ramon, P., Begoña C. Arrue, J. J. Acevedo, and A. Ollero. "Visual Surveillance System with Multi-UAVs Under Communication Constrains." In *Robot 2015: Second Iberian Robotics Conference*, pp. 705-713. Springer International Publishing, 2016.

[17]. Vision.cse.psu.edu, 'VIVID Tracking Evaluation Web Site', 2015. [Online]. Available: http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html. [Accessed: 16- Mar- 2011].

[18]. Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34, no. 7 (2012): 1409-1422.

[19]. X. Yu and Z. Shi, 'Vehicle detection in remote sensing imagery based on salient information and local shape feature', *Optik - International Journal for Light and Electron Optics*, 2015.

[20]. S. Noh, D. Shim and M. Jeon, 'Adaptive Sliding-Window Strategy for Vehicle Detection in Highway Environments', *IEEE IEEE Transactions on Intell. Transport. Syst.*, pp. 1-13, 2015.

[21]. A. Su, X. Sun, H. Liu, X. Zhang and Q. Yu, 'Online cascaded boosting with histogram of orient gradient features for car detection from unmanned aerial vehicle images', *Journal of Applied Remote Sensing*, vol. 9, no. 1, p. 096063, 2015.

[22]. X. Wang, H. Zhu, D. Zhang, D. Zhou and X. Wang, 'Vision-based Detection and Tracking of a Mobile Ground Target Using a Fixed-wing UAV', *International Journal of Advanced Robotic Systems*, p. 1, 2014.

[23]. Sokalski, Jan, Toby P. Breckon, and Ian Cowling. "Automatic salient object detection in uav imagery." *Proc. 25th International Unmanned Air Vehicle Systems* (2010): 11-1.

[24]. N. Baha, 'Real-Time Obstacle Detection Approach using Stereoscopic Images', *IJIEEB*, vol. 6, no. 1, pp. 42-48, 2014.

[25]. Kanistras, Konstantinos, Goncalo Martins, Matthew J. Rutherford, and Kimon P. Valavanis. "Survey of Unmanned Aerial Vehicles (UAVs) for Traffic Monitoring." In *Handbook of Unmanned Aerial Vehicles*, pp. 2643-2666. Springer Netherlands, 2015.

[26]. Wei, Peng, Xiaobo Lu, Tao Tang, Cong Li, and Jiaji Song. "A highway vehicle detection method based on the improved visual background extractor." In *Fuzzy*

Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on, pp. 1519-1524. IEEE, 2015.

[27]. J. Susaki, 'Region-based automatic mapping of tsunami-damaged buildings using multi-temporal aerial images', *Nat Hazards*, vol. 76, no. 1, pp. 397-420, 2014.

[28]. Viola, Paul, Michael J. Jones, and Daniel Snow. "Detecting pedestrians using patterns of motion and appearance." In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 734-741. IEEE, 2003.

[29]. J. Berni, P. Zarco-Tejada, L. Suarez and E. Fereres, 'Thermal and Narrowband Multispectral Remote Sensing for Vegetation Monitoring From an Unmanned Aerial Vehicle', *IEEE Trans. Geosci. Remote Sensing*, vol. 47, no. 3, pp. 722-738, 2009.

[30]. Gleason, Joshua, Ara V. Nefian, Xavier Bouyssounousse, Terry Fong, and George Bebis. "Vehicle detection from aerial imagery." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2065-2070. IEEE, 2011.

[31]. Razakarivony, Sébastien, and Frédéric Jurie. "Vehicle Detection in Aerial Imagery: A small target detection benchmark." *Journal of Visual Communication and Image Representation* 34 (2016): 187-203.

[32]. Betke, Margrit, and Huan Nguyen. "Highway scene analysis from a moving vehicle under reduced visibility conditions." In *IEEE International Conference on Intelligent Vehicles, Stuttgart, Germany*, pp. 131-136. 1998.

[33]. Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886-893. IEEE, 2005.

[34]. Xu, Yanwu, Xianbin Cao, and Hong Qiao. "An efficient tree classifier ensemble-based approach for pedestrian detection." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 41, no. 1 (2011): 107-117.

[35]. Liu, Kang, and Gellert Mattyus. "Fast multiclass vehicle detection on aerial images." *Geoscience and Remote Sensing Letters*, IEEE 12, no. 9 (2015): 1938-1942.

[36]. Sahli, Samir, Yueh Ouyang, Yunlong Sheng, and Daniel A. Lavigne. "Robust vehicle detection in low-resolution aerial imagery." In *SPIE Defense, Security, and Sensing*, pp. 76680G-76680G. International Society for Optics and Photonics, 2010.

[37]. Al-Kaff, Abdulla, Arturo de la Escalera, and José María Armingol. "SIFT and SURF Performance Evaluation and the Effect of FREAK Descriptor in the Context of Visual Odometry for Unmanned Aerial Vehicles." In *Computer Aided Systems Theory–EUROCAST 2015,* pp. 739-747. Springer International Publishing, 2015.

[38]. Cao, Xianbin, Changxia Wu, Jinhe Lan, Pingkun Yan, and Xuelong Li. "Vehicle detection and motion analysis in low-altitude airborne video under urban environment." *Circuits and Systems for Video Technology, IEEE Transactions on* 21, no. 10 (2011): 1522-1533.

[39]. Wu, Ryan, Bingwei Liu, Yu Chen, Erik Blasch, Haibin Ling, and Genshe Chen. "Pseudo-real-time Wide Area Motion Imagery (WAMI) processing for dynamic feature detection." In *Information Fusion (Fusion), 2015 18th International Conference on*, pp. 1962-1969. IEEE, 2015.

[40]. He, Dongmei, Congyan Lang, Songhe Feng, Xuetao Du, and Chen Zhang. "Vehicle detection and classification based on convolutional neural network." In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service,* p. 3. ACM, 2015.

[41]. KaewTraKulPong, Pakorn, and Richard Bowden. "An improved adaptive background mixture model for real-time tracking with shadow detection." In *Video-based surveillance systems*, pp. 135-144. Springer US, 2002.

[42]. Wang, Kejun, Ying Liang, Xianglei Xing, and Rongyi Zhang. "Target Detection Algorithm Based on Gaussian Mixture Background Subtraction Model." In *Proceedings of the 2015 Chinese Intelligent Automation Conference*, pp. 439-447. Springer Berlin Heidelberg, 2015.

[43]. X. Fan, W. Lin, J. Cao, B. Li and Y. Si, 'A Description Method for MSER with SIFT Descriptor', *AMM*, vol. 127, pp. 115-120, 2011.

[44]. J. Matas, O. Chum, M. Urban and T. Pajdla, 'Robust wide-baseline stereo from maximally stable extremal regions', *Image and Vision Computing*, vol. 22, no. 10, pp. 761-767, 2004.

[45]. R. Lin, H. Huang, R. Sun and L. Sun, 'An invariant interest point detector under image affine transformation', *Journal of Central South University*, vol. 22, no. 3, pp. 914-921, 2015.

[46]. K. Mikolajczyk and K. Mikolajczyk, 'Scale & Affine Invariant Interest Point Detectors', *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63-86, 2004.

[47]. Zakir, Usman, Iffat Zafar, and A. E. Edirisinghe. "Road Sign Detection and Recognition by using Local Energy Based Shape Histogram (LESH)." *International Journal of Image Processing* 4, no. 6 (2011): 566-582.

[48]. Morrone, M. Concetta, and Robyn A. Owens. "Feature detection from local energy." *Pattern recognition letters* 6, no. 5 (1987): 303-313.

[49]. Sarfraz, M. Saquib, Ahmed Saeed, M. Haris Khan, and Zahid Riaz. "Bayesian prior models for vehicle make and model recognition." In *Proceedings of the 7th International Conference on Frontiers of Information Technology*, p. 35. ACM, 2009.

[50]. P. Sand and S. Teller, 'Particle Video: Long-Range Motion Estimation Using Point Trajectories', *International Journal of Computer Vision*, vol. 80, no. 1, pp. 72-91, 2008.

[51]. J. Ding, Y. Tang, H. Tian, W. Liu and Y. Huang, 'Robust tracking with adaptive appearance learning and occlusion detection', *Multimedia Systems*, 2015.

[52]. V. Diaz-Ramirez, V. Contreras, V. Kober and K. Picos, 'Real-time tracking of multiple objects using adaptive correlation filters with complex constraints', *Optics Communications*, vol. 309, pp. 265-278, 2013.

[53]. Oron, Shaul, Aharon Bar-Hillel, and Shai Avidan. "Real-time tracking-with-detection for coping with viewpoint change." *Machine Vision and Applications* 26, no. 4 (2015): 507-518.

[54]. Liu, Wei, XueZhi Wen, Bobo Duan, Huai Yuan, and Nan Wang. "Rear vehicle detection and tracking for lane change assist." In *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 252-257. IEEE, 2007.

[55]. Zhao, Wei, Jun Tan, Xiangjing An, and Peidong Wang. "Person localization and tracking for a leader following vehicle by wireless sensors." In *Mechatronics and Automation (ICMA), 2015 IEEE International Conference on*, pp. 2615-2620. IEEE, 2015.

[56]. Li, Mengxin, Xiangqian Tian, Ying Zhang, Ke Xu, and Dai Zheng. "A Review of Vision-Based Vehicle Detection and Tracking Techniques for Intelligent Vehicle." In *2015 International Conference on Intelligent Systems Research and Mechatronics Engineering*. Atlantis Press, 2015.

[57]. Del Moral, Pierre. "Measure-valued processes and interacting particle systems. Application to nonlinear filtering problems." *Annals of Applied Probability* (1998): 438-495.

[58]. Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." *Journal of Fluids Engineering* 82, no. 1 (1960): 35-45.

[59]. Liu, Yisha, Nan Jiang, Jian Wang, and Yiwen Zhao. "Vision-based moving target detection and tracking using a quadrotor UAV." In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pp. 2358-2363. IEEE, 2014.

[60]. Lin, Qing, Pei Cao, Dingan Liao, Yongzhao Zhan, and Yaping Yang. "Improved Multi-target Tracking Algorithm Based on Gaussian Mixture Particle PHD Filter." *International Journal of Multimedia and Ubiquitous Engineering* 10, no. 2 (2015): 227-236.

[61]. Sivaraman, Sayanan, and Mohan Manubhai Trivedi. "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis." *Intelligent Transportation Systems, IEEE Transactions on* 14, no. 4 (2013): 1773-1795.

[62]. Toledo-Moreo, R., Carlos Colodro-Conde, and F. Javier Toledo-Moreo. "A Multiple-Model Particle Filter Based Method for Slide Detection and Compensation in Road

Vehicles." In *Bioinspired Computation in Artificial Systems*, pp. 156-165. Springer International Publishing, 2015.

[63]. Lee, Minchae, Chulhoon Jang, and Myoungho Sunwoo. "Probabilistic lane detection and lane tracking for autonomous vehicles using a cascade particle filter." *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* (2015): 0954407014567719.

[64]. Fontanelli, Daniele, David Macii, and Tizar Rizano. "A fast and low-cost vision-based line tracking measurement system for robotic vehicles." *ACTA IMEKO* 4, no. 2 (2015): 90-99.

[65]. Arróspide, Jon, Luis Salgado, Marcos Nieto, and Fernando Jaureguizar. "On-board robust vehicle detection and tracking using adaptive quality evaluation." In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, 2008.

[66]. Aytekin, Bureu, and Erdinç Altuğ. "Increasing driving safety with a multiple vehicle detection and tracking system using ongoing vehicle shadow information." In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 3650-3656. IEEE, 2010.

[67]. Franke, Uwe, Clemens Rabe, Hernán Badino, and Stefan Gehrig. "6d-vision: Fusion of stereo and motion for robust environment perception." In *Pattern Recognition*, pp. 216-223. Springer Berlin Heidelberg, 2005.

[68]. Lim, Young-Chul, Chung-Hee Lee, Soon Kwon, and Jong-hun Lee. "A fusion method of data association and virtual detection for minimizing track loss and false track." In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 301-306. IEEE, 2010.

[69]. Rabe, Clemens, Uwe Franke, and Stefan Gehrig. "Fast detection of moving objects in complex scenarios." In *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 398-403. IEEE, 2007.

[70]. Estatico, Claudio, Alessandro Fedeli, Matteo Pastorino, and Andrea Randazzo. "Buried object detection by means of a Lp Banach-space inversion procedure." *Radio Science* 50, no. 1 (2015): 41-51.

[71]. Broggi, Alberto, Stefano Cattani, Elena Cardarelli, Brad Kriel, Michael S. McDaniel, and Hong Chang. "Disparity space image's features analysis for error prediction of a stereo obstacle detector for heavy duty vehicles." In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 80-86. IEEE, 2011.

[72]. Shantaiya, Sanjivani, Kesari Verma, and Kamal Mehta. "Multiple Object Tracking using Kalman Filter and Optical Flow." *European Journal of Advances in Engineering and Technology* 2, no. 2 (2015): 34-39.

[73]. Modalavalasa, Nagamani, G. Sasi Bhushana Rao, K. Satya Prasad, L. Ganesh, and M. N. V. S. S. Kumar. "A new method of target tracking by EKF using bearing and elevation measurements for underwater environment." *Robotics and Autonomous Systems* (2015).

[74]. Vatavu, Andrei, Radu Danescu, and Sergiu Nedevschi. "Stereovision-Based Multiple Object Tracking in Traffic Scenarios Using Free-Form Obstacle Delimiters and Particle Filters." *Intelligent Transportation Systems, IEEE Transactions on* 16, no. 1 (2015): 498-511.

[75]. Grinberg, Michael, Florian Ohr, and Jürgen Beyerer. "Feature-based probabilistic data association (FBPDA) for visual multi-target detection and tracking under occlusions and split and merge effects." In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pp. 1-8. IEEE, 2009.

[76]. Duan, Zhansheng, and Xiaoyun Li. "A new nonlinear state estimator using the fusion of multiple extended Kalman filters." In *Information Fusion (Fusion), 2015 18th International Conference on*, pp. 90-97. IEEE, 2015.

[77]. Tribou, Michael J., Adam Harmat, David WL Wang, Inna Sharf, and Steven L. Waslander. "Multi-camera parallel tracking and mapping with non-overlapping fields of view." *The International Journal of Robotics Research* (2015): 0278364915571429.

[78]. Catalin, Golben, and Sergiu Nedevschi. "Object tracking from stereo sequences using particle filter." In *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on*, pp. 279-282. IEEE, 2008.

[79]. Hermes, Christoph, Julian Einhaus, Markus Hahn, Christian Wöhler, and Franz Kummert. "Vehicle tracking and motion prediction in complex urban scenarios." In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 26-33. IEEE, 2010.

[80]. Danescu, Radu, Florin Oniga, and Sergiu Nedevschi. "Modeling and tracking the driving environment with a particle-based occupancy grid." *Intelligent Transportation Systems, IEEE Transactions on* 12, no. 4 (2011): 1331-1342.

[81]. Singh, Pritpal, Tanjot Sethi, Bunil Kumar Balabantaray, and Bibhuti Bhushan Biswal. "Advanced vehicle security system." In *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, pp. 1-6. IEEE, 2015

[82]. Lim, Hyon, Sudipta N. Sinha, Michael F. Cohen, Matt Uyttendaele, and H. Jin Kim. "Real-time monocular image-based 6-DoF localization." *The International Journal of Robotics Research* 34, no. 4-5 (2015): 476-492.

[83]. Spulak, David, Richard Otrebski, and Wilfried Kubinger. "Object Tracking by Combining Supervised and Adaptive Online Learning through Sensor Fusion of Multiple Stereo Camera Systems." *ARW 2015*: 5.

[84]. Cheng, Xu, Nijun Li, Tongchi Zhou, Lin Zhou, and Zhenyang Wu. "Object tracking via collaborative multi-task learning and appearance model updating." *Applied Soft Computing* 31 (2015): 81-90.

[85]. Wang, Zelun, Jinjun Wang, Shun Zhang, and Yihong Gong. "Visual tracking based on online sparse feature learning." *Image and Vision Computing* 38 (2015): 24-32.

[86]. Zhu, G., J. Wang, C. Zhao, and H. Lu. "Weighted Part Context Learning for Visual Tracking." *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society* (2015).

[87]. Liu, Yangbiao, Bo Ma, Hongwei Hu, and Yin Han. "Boosting-Based Visual Tracking Using Structural Local Sparse Descriptors." In *Computer Vision--ACCV 2014*, pp. 522-533. Springer International Publishing, 2015.

[88]. Wu, Yuwei, Mingtao Pei, Min Yang, Yang He, and Yunde Jia. "Landmark-based inductive model for robust discriminative tracking." In *Computer Vision--ACCV 2014,* pp. 320-335. Springer International Publishing, 2015.

[89]. Liu, Qingshan, Jing Yang, Kaihua Zhang, and Yi Wu. "Adaptive Compressive Tracking via Online Vector Boosting Feature Selection." *arXiv preprint arXiv:1504.05451* (2015).

[90]. Xiao, Jingjing, Rustam Stolkin, and Aleš Leonardis. "Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4978-4987. 2015.

[91]. Karami, Amir Hossein, Maryam Hasanzadeh, and Shohreh Kasaei. "Online adaptive motion model-based target tracking using local search algorithm." *Engineering Applications of Artificial Intelligence* 37 (2015): 307-318.

[92]. Li, Shaomei, Chao Gao, and Yawen Wang. "Adaptive object tracking via both positive and negative models matching." In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014),* pp. 94430R-94430R. International Society for Optics and Photonics, 2015.

[93]. Pang, Sh Ch, Anan Du, and Zh Zh Yu. "Robust multi-object tracking using deep learning framework." *Journal of Optical Technology* 82, no. 8 (2015): 516-527.

[94]. Wang, Zelun, Jinjun Wang, Shun Zhang, and Yihong Gong. "Visual tracking based on online sparse feature learning." *Image and Vision Computing* 38 (2015): 24-32.

[95]. Zhou, Tianfei, and Yao Lu. "Online Visual Tracking using Multiple Instance Learning with Instance Significance Estimation." *Journal of Computer Vision and Pattern Recognition arXiv preprint arXiv:1501.04378* (2015).

[96]. Misra, Ishan, Abhinav Shrivastava, and Martial Hebert. "Watch and learn: Semi-supervised learning of object detectors from videos." *arXiv preprint arXiv:1505.05769* (2015).

[97]. Ororbia II, Alexander G., David Reitter, Jian Wu, and C. Lee Giles. "Online Learning of Deep Hybrid Architectures for Semi-supervised Categorization." In *Machine*

Learning and Knowledge Discovery in Databases, pp. 516-532. Springer International Publishing, 2015.

[98]. Metaxas, Dimitris, and Atul Kanaujia. "System and method for detecting and tracking features in images." U.S. Patent 9,014,465, issued April 21, 2015.

[99]. Garriga, Joan, John R. Palmer, Aitana Oltra, and Frederic Bartumeus. "Expectation-Maximization Binary Clustering for Behavioural Annotation." *arXiv preprint arXiv:1503.04059* (2015).

[100]. Chapelle, Olivier, and Bernhard Schölkopf. "A discussion of semi-supervised learning and transduction." MIT Press (2006): 473-478.

[101]. Kuen, Jason, Kian Ming Lim, and Chin Poo Lee. "Self-taught learning of a deep invariant representation for visual tracking via temporal slowness principle." *Pattern Recognition* (2015).

[102]. Li, Fei, Congqing WANG, Xin ZHOU, and Dake ZHOU. "Co-Training Object Tracking with Online Multiple Instance Learning." *Journal of Jilin University (Information Science Edition)* 2 (2015): 014.

[103]. Zhong, Bineng, Yingju Shen, Yan Chen, Weibo Xie, Zhen Cui, Hongbo Zhang, Duansheng Chen et al. "Online learning 3D context for robust visual tracking." *Neurocomputing* 151 (2015): 710-718.

[104]. Haralick, Robert M., Karthikeyan Shanmugam, and Its' Hak Dinstein. "Textural features for image classification." *Systems, Man and Cybernetics, IEEE Transactions on* 6 (1973): 610-621.

[105]. Zhuo, Fu Qing, Pei Qun Lin, and Yu Mu Gu. "Vision-based Vehicle Detection in Real Traffic Environment Using Fast Wavelet Transform and Kalman Filter." In *Advanced Materials Research*, vol. 998, pp. 717-722. 2014.

[106]. Chaddad, A., F. Ahmad, M. G. Amin, P. Sévigny, and D. DiFilippo. "Textural feature selection for enhanced detection of stationary humans in through-the-wall radar imagery." In *SPIE Defense+ Security*, pp. 90770F-90770F. International Society for Optics and Photonics, 2014.

[107]. Mohanaiah, P., P. Sathyanarayana, and L. GuruKumar. "Image texture feature extraction using GLCM approach." *International Journal of Scientific and Research Publications* 3, no. 5 (2013)

[108]. Ou, Xiang, Wei Pan, and Perry Xiao. "In vivo skin capacitive imaging analysis by using grey level co-occurrence matrix (GLCM)." *International journal of pharmaceutics* 460, no. 1 (2014): 28-32.

[109]. Hua, B. O., Ma Fu-Long, and Jiao Li-Cheng. "Research on Computation of GLCM of Image Texture [J]." *Acta Electronica Sinica* 1, no. 1 (2006): 155-158.

[110]. Bin, Yu Haipeng Liu Yixing Zhang, and Li Yongfeng. "Application of Spatial Gray Level Cooccurrence Matrix in Wood Surface Texture Quantitative Analysis [J]." *Scientia Silvae Sinicae* 6 (2004): 020.

[111]. Milligan, Peter R., Michael P. Morse, and Shanti Rajagopalan. "Pixel map preparation using the HSV colour model." *Exploration Geophysics 23, no. 1/2* (1992): 219-224.

[112]. Couzinie-Devy, Florent, Jian Sun, Karteek Alahari, and Jean Ponce. "Learning to estimate and remove non-uniform image blur." In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 1075-1082. IEEE, 2013.

[113]. Elad, Michael, and Yacov Hel-Or. "A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur." *Image Processing, IEEE Transactions on* 10, no. 8 (2001): 1187-1193.

[114]. Truscott, Roger John. "Presbyopia. Emerging from a blur towards an understanding of the molecular basis for this most common eye condition." *Experimental eye research* 88, no. 2 (2009): 241-247.

[115]. Ayers, G. R., and J. Christopher Dainty. "Iterative blind deconvolution method and its applications." *Optics letters* 13, no. 7 (1988): 547-549.

[116]. Hanley, James A., and Barbara J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." Radiology 143, no. 1 (1982): 29-36.

[117]. K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *Pattern Analysis and Machine Intelligence,* vol. 27, no. 10, pp. 1615–1630, 2005.

[118]. Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).

[119]. Lowe, David G. "Object recognition from local scale-invariant features." In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150-1157. Ieee, 1999.

[120]. Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60, no. 2 (2004): 91-110.

[121]. Turcot, Panu, and David G. Lowe. "Better matching with fewer features: The selection of useful features in large database recognition problems." In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 2109-2116. IEEE, 2009.

[122]. Florack, Luc, Bart Ter Haar Romeny, Max Viergever, and Jan Koenderink. "The Gaussian scale-space paradigm and the multiscale local jet." *International Journal of Computer Vision* 18, no. 1 (1996): 61-75.

[123]. Lindeberg, Tony, and Jonas Gårding. "Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure." *Image and vision computing* 15, no. 6 (1997): 415-434.

[124]. Wallach, Hanna M. "Topic modeling: beyond bag-of-words." In *Proceedings of the 23rd international conference on Machine learning*, pp. 977-984. ACM, 2006.

[125]. Frejlichowski, Dariusz, Katarzyna Gościewska, Paweł Forczmański, Adam Nowosielski, and Radosław Hofman. "Applying Image Features and AdaBoost Classification for Vehicle Detection in the 'SM4Public'System." In *Image Processing and Communications Challenges* 7, pp. 81-88. Springer International Publishing, 2016.

[126]. Gu, Qin, Jianyu Yang, Yuqiang Zhai, and Lingjiang Kong. "Vision-based multi-scaled vehicle detection and distance relevant mix tracking for driver assistance system." *Optical Review* 22, no. 2 (2015): 197-209.

[127]. Sivaraman, Sayanan, and Mohan M. Trivedi. "Active learning for on-road vehicle detection: A comparative study." *Machine vision and applications* 25, no. 3 (2014): 599-611.

[128]. Khammari, Ayoub, Fawzi Nashashibi, Yotam Abramson, and Claude Laurgeau. "Vehicle detection combining gradient analysis and AdaBoost classification." In *Intelligent Transportation Systems*, 2005. Proceedings. 2005 IEEE, pp. 66-71. IEEE, 2005.

[129]. Leibe, Bastian, Ales Leonardis, and Bernt Schiele. "Combined object categorization and segmentation with an implicit shape model." In *Workshop on statistical learning in computer vision, ECCV*, vol. 2, no. 5, p. 7. 2004.

[130]. Harris, Chris, and Mike Stephens. "A combined corner and edge detector." In *Alvey vision conference*, vol. 15, p. 50. 1988.

[131]. Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." In *Computer Vision–ECCV 2006*, pp. 430-443. Springer Berlin Heidelberg, 2006.

[132]. Smith, Stephen M., and J. Michael Brady. "SUSAN—A new approach to low level image processing." *International journal of computer vision* 23, no. 1 (1997): 45-78.

[133]. Zhao, Tao, and Ram Nevatia. "Tracking multiple humans in complex situations." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26, no. 9 (2004): 1208-1221.

[134]. Cao, Xianbin, Changxia Wu, Pingkun Yan, and Xuelong Li. "Linear SVM classification using boosting HOG features for vehicle detection in low-altitude airborne videos." In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 2421-2424. IEEE, 2011.

[135]. Kumar, R. Sharan, P. G. Student, and T. MANI. "Vehicle Detection and Classification from Satellite Images based on Gaussian Mixture Model." *Digital Image Processing* 7, no. 4 (2015): 99-104.

[136]. Takeuchi, Remma, Kunihito Kato, David Harwood, and Larry S. Davis. "Vehicle detection using PLS Hough transform." In *Frontiers of Computer Vision (FCV), 2015 21st Korea-Japan Joint Workshop on*, pp. 1-6. IEEE, 2015.

[137]. Hillis, David M., and James J. Bull. "An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis." *Systematic biology* 42, no. 2 (1993): 182-192.

[138]. Bernardin, Keni, and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the CLEAR MOT metrics." *Journal on Image and Video Processing* 2008 (2008): 1.

[139]. Bernardin, Keni, and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the CLEAR MOT metrics." *Journal on Image and Video Processing* 2008 (2008): 1.

[140]. Grabner, Helmut, Christian Leistner, and Horst Bischof. "Semi-supervised on-line boosting for robust tracking." In *Computer Vision–ECCV 2008*, pp. 234-247. Springer Berlin Heidelberg, 2008.

[141]. Fan, Zhenhua, Hongbing Ji, and Yongquan Zhang. "Iterative particle filter for visual tracking." *Signal Processing: Image Communication* 36 (2015): 140-153.

[142]. Collins, Robert T., Yanxi Liu, and Marius Leordeanu. "Online selection of discriminative tracking features." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27, no. 10 (2005): 1631-1643.

[143]. Maron, Oded, and Tomás Lozano-Pérez. "A framework for multiple-instance learning." *Advances in neural information processing systems* (1998): 570-576.

[144]. Yu, Qian, Thang Ba Dinh, and Gérard Medioni. "Online tracking and reacquisition using co-trained generative and discriminative trackers." In *Computer Vision–ECCV 2008*, pp. 678-691. Springer Berlin Heidelberg, 2008.

[145]. Vision.cse.psu.edu, 'VIVID Tracking Evaluation Web Site', 2015. [Online]. Available: http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html. [Accessed: 16- Mar- 2011].