A Robust Vehicle Make and Model Recognition System for ITS Applications

by

Abdul Jabbar Siddiqui

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

A real-time Vehicle Make and Model Recognition (VMMR) system is a significant component of security applications in Intelligent Transportation Systems (ITS). A highly accurate and real-time VMMR system significantly reduces the overhead cost of resources otherwise required. In this thesis, we present a VMMR system that provides very high classification rates and is robust to challenges like low illumination, occlusions, partial and non-frontal views. These challenges are encountered in realistic environments and high security areas like parking lots and public spaces (e.g., malls, stadiums, and airports). The VMMR problem is a multi-class classification problem with a peculiar set of issues and challenges like multiplicity, inter- and intra-make ambiguity among various vehicles makes and models, which need to be solved in an efficient and reliable manner to achieve a highly robust VMMR system. To reliably overcome the ambiguity challenges, a global features representation approach based on the Bag-of-Features paradigm is proposed. We extract key features from different make-model classes in an optimized dictionary, through two different dictionary building strategies. We represent different samples from each class with respect to the learned dictionary. We also present two classification schemes based on multi-class Support Vector Machines (SVMs): (1) Single multi-class SVM and (2) Attribute Bagging-based Ensemble of multi-class SVMs. These classification schemes allow simultaneous learning of the differences between global representations of different classes and the similarities between different shapes or generations within a same make-model class, to further overcome the multiplicity challenges for real-time application. Extensive experiments conducted using our approaches yield superior results for images that were occluded, under low illumination, partial camera views, or even non-frontal views, available in a recently published real-world VMMR dataset. The approaches presented herewith provide a highly accurate VMMR system for real-time applications in realistic environments.

Acknowledgements

All thanks and praises are due to the Almighty Lord for blessing me with the capability, talent, zeal and vigour to pursue passionate research through my thesis.

I will be ever grateful to a number of sincere people who have guided and motivated me throughout my thesis work. I express my deepest gratitude to my supervisor Prof. Azzedine Boukerche and my mentor Dr. Abdelhamid Mammeri for their selfless efforts in guiding me throughout my research and providing me invaluable feedback at every step of my work. I sincerely thank Prof. Boukerche and the NSERC-DIVA Strategic Research Network for providing me financial support as a Research Assistant to pursue my research interests. I owe a debt of gratitude to my beloved father, Mr. M. A. B. Siddiqui, for being my source of inspiration always and supporting me throughout my studies; and to my dear father-in-law, Mr. Abdullah Ahmed, for the fruitful discussions in analyzing problems and guidance in prioritizing tasks. In addition, I would also like to acknowledge the help and assistance from my colleagues at PARADISE Lab of UOttawa.

Finally, I thank my dear mother, siblings and wife for their encouragement and love that kept my morale high throughout the ups and downs during my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Over the recent years, a plethora of innovative technologies and solutions are bringing Intelligent Transportation Systems (ITS) closer to reality. ITS is defined as "the application of advanced and emerging technologies (computers, sensors, control, communications, and electronic devices) in transportation to save lives, time, money, energy and the environment", by the ITS Society of Canada [1]. In an ITS, vehicles, infrastructure, drivers and other users dynamically interact with each other to achieve these goals. The governments, industry and academia are making joint efforts [1–5] towards adopting and implementing advanced ITS to enhance transportation safety and security. The development of digital image sensors and computer vision techniques offer a great deal of advantages in enabling many important ITS applications and components such as Advanced Driver Assistance Systems (ADAS), Automated Vehicular Surveillance (AVS), traffic and activity monitoring, traffic behaviour analysis, traffic management, etc. Identification and classification of vehicles is of great interest in these applications, owing to heightened security concerns in ITS [3].

AVS broadly includes Vehicle Detection, Identification, Classification, and Vehicle Tracking. Over the years, significant research has been done to solve challenges in vehicle identification, detection and tracking. However, to classify vehicles into fine categories such as makes and models, has gained attention only recently, and many challenges remain yet to be addressed [6–

16]. The focus of this thesis is on developing novel approaches to address the challenges in real-time and automated Vehicle Make and Model Recognition (VMMR), utilizing state-of-the-art vision-based techniques.

Security is a great concern in highly vulnerable areas such as parking lots of public spaces (e.g., malls, stadiums, airports, etc.). In these critical scenarios, an AVS system running over surveillance cameras' images can greatly assist the security personnel in identifying vehicles belonging to certain colors, types, makes, or models. Moreover, in cases where the police are searching for a target vehicle of a specific make or model, AVS systems would save considerable amount of time, resources and manpower. The mobile police vehicles equipped with video/images sensors could share the video/images [17, 18] of target vehicles with other police vehicles and with roadside infrastructure, through various video dissemination techniques over Vehicular Ad-Hoc Networks (VANETs) [19–26]. In addition, for applications such as electronic toll collection, where different charges are applied to different types of vehicles, vision-based AVS systems could serve as a complementary tool in improving efficiency of existing systems.

In VANETs, vehicles can share information amongst themselves and/or with roadside infrastructure to provide critical services such as target tracking, monitoring or surveillance. The vehicle classification systems such as ours enable surveillant vehicles to recognize important information about targets. These targets can be even localized though various localization protocols [27–33]. In order to meet the speed and fault-tolerance requirements of critical surveillance and monitoring applications, to enable reliable dissemination of information in a fast manner, protocols such as [34–39] need to be utilized. In a dense surveillance scenario such as a large area with numerous surveillance cameras, or a large city with a number of surveillant vehicles, efficient synchronization between different nodes is required [40–42]. To ensure connectivity of mobile surveillant vehicles with each other and the infrastructure, mobility management protocols such as [43] can help reduce packet losses and latency. Since security of VANET-enabled surveillance systems is an essential requirement, owing to privacy concerns when it comes to sharing targets' information over the open wireless channels, trust-based security systems such as [44–48] could be used.

Figure 1.1: (Left-Right) Some cases where detected license plates are ambiguous, forged, damaged, or duplicated, making the license plate recognition based VMMR Systems to fail.

The traditional vehicle identification systems recognize makes and models of vehicles relying on manual human observations or automated license plate recognition (ALPR) systems, hardly meeting the real-time constraints. Both approaches are failure-prone and have several limitations. Firstly, it is practically difficult for human observers to remember and efficiently distinguish between the wide variety of vehicle makes and models. Secondly, it becomes a laborious and time-consuming task for a human observer to monitor and observe the multitude of screens and record the incoming or outgoing makes and models, or to even spot the make and model being looked for. On the other hand, the VMMR systems that rely on license-plates suffer from the following disadvantages. License plates are easy to be forged, damaged, modified, or occluded, as depicted in Figure 1.1. Also, there are some license-plates that can be ambiguous (e.g., between "0" and "O"), as shown at the leftmost of Figure 1.1. Moreover, in some areas, it may not be required to bear the license-plate at the front or rear. If the ALPR system is not equipped to check for license-plates at both (front and rear) views of the vehicle, it could fail. Consequently, when ALPR systems fail to correctly read the detected license-plates due to the above issues, the wrong make-model info could be retrieved from the license-plates registry or database.

To overcome the above shortcomings in traditional vehicle identification and classification systems, automated VMMR techniques have recently gained attention, but hardly meeting real-time processing speed requirements. The make and model of the vehicle recognized by the VMMR system can be cross-checked with the license-plate registry to check for any fraud. In this way, vision-based automated VMMR techniques, such as the ones proposed in this work, augment traditional ALPR-based vehicle identification and classification systems to further enhance security.

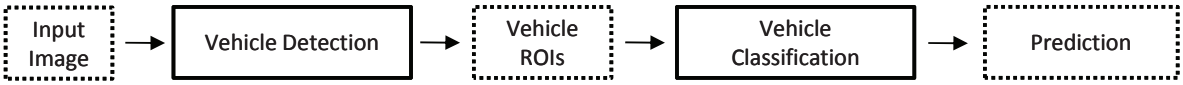


Figure 1.2: General Architecture of Vehicle Classification Systems

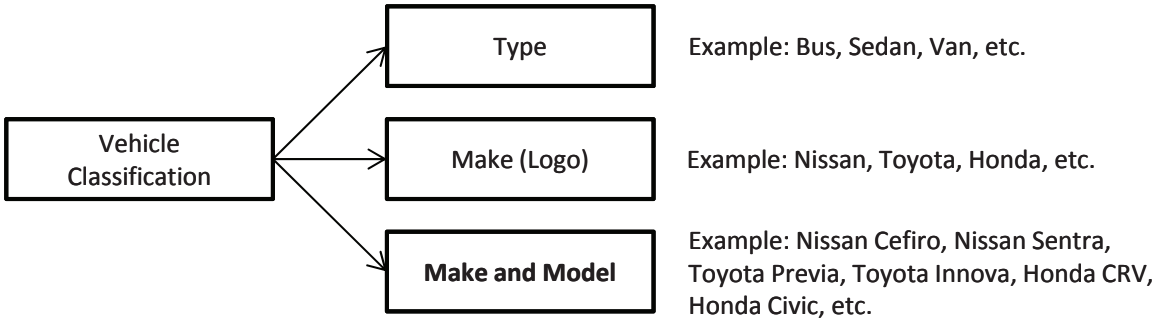


Figure 1.3: A taxonomy of Vehicle Classification works: Vehicle Type Recognition (VTR), Vehicle Make (Logo) Recognition (VMR or VLR), and Vehicle Make and Model Recognition (VMMR)

1.2 General Architecture of Vehicle Classification Systems

The problem of automated vehicle classification into makes and models is an important task for AVS and other ITS applications. We provide the general architecture of vehicle classification systems in Figure 1.2. Most works first adopt a Vehicle Detection step which produces Regions of Interest (ROIs) containing the vehicles' faces (front or rear), segmented from the background. The Vehicle Classification systems then work on these ROIs. Depending on the granularity of classification, vehicle classification systems could be classified into three categories: Type, Make (Logo), or Make-and-Model recognition, as depicted in Figure 1.3. The focus of our work is on automated Vehicle Make and Model Recognition (VMMR), which basically comprises of three steps: (1) Features Extraction, (2) Global Features Representation, and (3) Classification, as shown in Figure 1.4. To specify, this thesis proposes and investigates unexplored Global Features Representation and Classification approaches for VMMR, to effectively tackle the issues and challenges therewith.

Figure 1.4: A flowchart of most Vehicle Make and Model Recognition (VMMR) approaches

1.3 Challenges and Issues

The problem of vision-based automated VMMR can be considered as a challenging multi-class image classification problem, in which a "class" is a particular vehicle make and model. However, VMMR presents a more diverse and challenging set of issues than in other image classification problems. There are two broad categories of challenges in VMMR: (1) *Multiplicity*, and (2) *Ambiguity* [10]. The *multiplicity* problem occurs when a vehicle model (of the same make) has different shapes and/or appearances. Figure 1.5 shows some examples of the multiplicity problem in NTOU-MMR Dataset [10]. We further classify the *ambiguity* problem into two kinds: (a) *Inter-Make Ambiguity*, and (b) *Intra-Make Ambiguity*. The former ambiguity refers to the issue of vehicles (models) of different companies (makes) having visually similar shape or appearance, i.e., two different make-model classes have similar front or rear views. For example, "Toyota Camry 2005" and "Nissan Cefiro 1999" have visually similar appearance (See Figure 1.6). The latter kind of ambiguity results when different vehicles (models) of same company (make) have similar shape or appearance. For example, the "Altis" and "Camry" models of the "Toyota" make have similar front faces (See Figure 1.7).

1.4 Overview of Proposed Solutions

To address the above-mentioned challenges and issues, we propose and investigate unexplored approaches for VMMR in which the key features from different make-model classes are learned in an optimised dictionary to tackle the ambiguity issues, based on the Bag of Features (BoF) paradigm. We represent different samples from each class with respect to this learned dictionary. In order to simultaneously learn the differences between global representations of different classes and the similarities between different shapes or generations within a same make-model class, efficient classification schemes are designed: two based on multi-class Support Vector Ma-

(a) T Wish 2010 (b) T Wish 2009 (c) T Wish 2005

(d) T Camry 2008 (e) T Camry 2010

(f) F Mondeo 2 (g) F Mondeo 3

(h) H CRV 2003 (i) H CRV 2005 (j) H CRV 2009

Figure 1.5: Multiplicity Problems with Toyota Wish (1.5a)-(1.5c), Toyota Camry (1.5d)-(1.5e), Ford Mondeo (1.5f)-(1.5g), Honda CRV (1.5h)-(1.5j) in NTOU-MMR Dataset [10].

(a) T Camry 2005 (b) N Cefiro 1999 (c) T Tercel 2005

(d) N Sentra 2003 (e) T Camry 2006 (f) F Mondeo 0

Figure 1.6: Inter-Make Ambiguity Problems between (1.6a)-(1.6b), (1.6c)-(1.6d), and (1.7e)-(1.6f) in NTOU-MMR Dataset of [10]. "T", "N", and "F" stand for "Toyota", "Nissan", and "Ford", respectively.

(a) N Sentra 2003 (b) N Cefiro 1997 (c) T Altis 2008

(d) T Camry 2008 (e) T Camry 2006 (f) T Altis 2006

Figure 1.7: Intra-Make Ambiguity Problems between (1.7a)-(1.7b), (1.7c)-(1.7d), (1.7c)-(1.7e) and (1.7e)-(1.7f) in NTOU-MMR Dataset of [10]. "N" and "T" stand for "Nissan" and "Toyota", respectively.

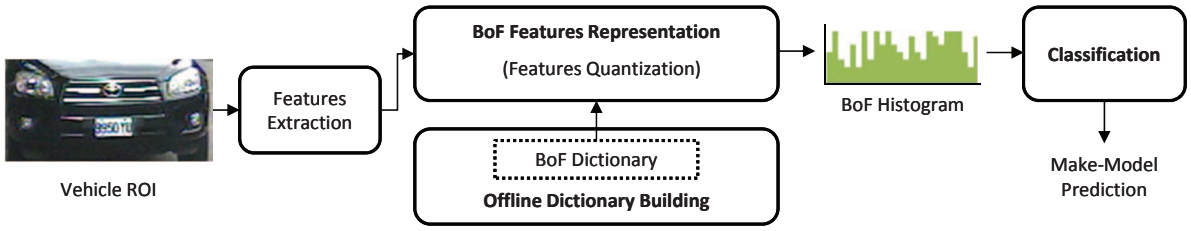


Figure 1.8: An overview of the proposed BoF-based VMMR approaches

chines and one based on Random Forest. Therefore, we tackle the multiplicity and ambiguity issues in VMMR through optimized global representations and efficient classification schemes.

1.5 Contributions

The major contributions of this work on vehicle make and model recognition are summarised as follows.

- We propose and evaluate unexplored global representation and classification approaches for VMMR, based on the BoF paradigm, and prove their effectiveness in realistic scenarios.
- To learn the key characteristics and features from all classes of makes and models in an optimised dictionary, two schemes of Dictionary Building are studied, optimized and evaluated, to address the multiplicity and ambiguity problems of VMMR: (1) the *Single-Dictionary* scheme, and (2) the *Modular-Dictionary* scheme.
- For classification, we present three classification schemes: two based on multi-class SVM (Single multi-class SVMs and Attribute Bagging based Ensemble of single multi-class SVMs), and one based on a tree-based classifier (Random Forest).
- The effectiveness (in terms of speed and accuracy) of our VMMR approaches, and their superiority over related works, are validated on N_D random training-testing dataset partitions of a recent publicly available real-world dataset.

The work accomplished through this thesis has resulted in the following publications:

- A. J. Siddiqui, A. Mammeri, A. Boukerche, "*Towards Efficient Vehicle Classification in Intelligent Transportation Systems,*" to be presented at the Fifth ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications (ACM DIVANet '15).

- A. J. Siddiqui, A. Mammeri, A. Boukerche, "*Real-time Vehicle Make and Model Recognition based on Bag of SURF Features,*" submitted to the IEEE Transactions on Intelligent Transportation Systems (TITS).

1.6 Thesis Outline

This thesis is organised as follows:

- Chapter 2 provides a comprehensive literature review showing how representative works in vehicle identification and classification have evolved over the years.

- Chapter 3 deals with detailed descriptions and discussions of feature extraction and global representation approaches in notable VMMR works and presents the proposed BoF-based approaches for VMMR.

- Chapter 4 focusses on the classification techniques. We first review and discuss the classification schemes designed by representative VMMR works, and then describe the three classifications schemes proposed by us for BoF-VMMR.

- Chapter 5 describes the target environment, presents the performance metrics, and discusses the optimal parameters for the different modules in our BoF-VMMR.

- Chapter 6 evaluates the performance and efficiency of the proposed VMMR approaches.

- Chapter 7 concludes by summarizing the work done in this thesis, and presents future directions of research for VMMR works.

Chapter 2

Related Works

In this section, we provide an overview of works done in different types of vehicle classification works, to show how research works are evolving in this challenging area. The vehicle classification works can be categorized into Vehicle Type Recognition, Vehicle Make (Logo) Recognition, or Vehicle Make and Model Recognition, as depicted in Figure 1.3. Since this work focusses on developing and investigating improved approaches for the Global Features Representation and Classification steps of VMMR, we provide a comprehensive discussion of related works in the context of these stages.

2.1 Vehicle Type Recognition

The objective of works on Vehicle Type Recognition (VTR) is to classify the vehicles into high-level categories such as van, mini-van, truck, sedan, bus, taxi, etc. In such works, the exact make and model is not recognized. Having an automated VTR system helps in applications such as electronic toll collection, traffic studies and analyses, etc. With the development of computer vision techniques and traffic surveillance cameras, vision-based VTR systems have gained a lot of attention over the years.

In Table 2.1, we summarize some of the representative VTR works in terms of features extraction and classification techniques. Ma and Grimson [49] employed implicit and explicit

shape models based on edges and modified SIFT for describing vehicle types, and used a 2-class Bayesian Decision Rule for classification. Zhang et al. [50] applied a texture descriptor known as Multi-Block Local Binary Patterns (MB-LBP) and an AdaBoost classifier based on multi-branch regression trees. Buch et al. [51] proposed 3D-HOG features and 3D models to describe vehicle types, and performed model-based matching for classification. Chen et al. [52,53] used various geometry- and shape-based features with several classifiers such as SVM, Random Forest, and Model-based matching. Wang et al. [54] also used geometry-based features such as length and height of vehicle silhouettes, but applied simple euclidean distance-based matching for classification. Dong et al. [55, 56] proposed learning useful local and global features through a two-stage unsupervised or semi-supervised convolution neural network, and applied softmax regression for classification. Most of these works assume a static background and rely on background subtraction, which makes these approaches fail in cases of occlusions.

As noted in Table 2.1, the VTR works can only discriminate between the broad categories of vehicles such as sedans, vans, mini-vans, buses, microbuses, SUVs, trucks, bikes , etc. However, in many security related applications, we require finer classification of vehicles into their makes and models, and not their types alone.

Table 2.1: Representative Works on Vehicle Type Recognition

Work	**Features**	**Classification**	**Types**
Ma and Grimson (2005) [49]	Edges- and Modified-SIFT-based Implicit Shape Model, Explicit Shape Model, Constellation Model	2-Class Bayesian Decision Rule	Sedans, mini-vans, taxis
Zhang et al. (2007) [50]	Multi Block Local Binary Patterns (MB-LBP)	AdaBoost (Multi-branch Regression Trees)	Cars, Vans, Trucks, Bikes
Buch et al. (2009) [51]	3D-HOG and 3D-Model	Matching with Models	Bus, Sedan, Van, Bike
Chen et al. (2011) [52]	Size & Shape Features, Boundig Box Perimeter, Ellipticity, Filled Area, Convex area, etc.	SVM, Random Forest, Model-based Matching	Car, Van, Bus, Bike (Bicycle or Motorcycle)
			Contd. on next page

Table 2.1 – contd. from prev. page

Work	Features	Classification	Types
Chen et al. (2012) [53]	Measurement based Features (MBF), Intensity Pyramid HOG (IPHOG)	SVM	Car, Van, Bus, Bike (Bicycle or Motorcycle)
Wang et al. (2014) [54]	Length Height of vehicle Silhouettes	Euclidean Distance based Matching	Bikes, Cars, Minibuses
Dong et al. (2014) [55]	Local and Global Features generated by a 2 Stage Unsupervised Convolution Neural Network (with Layer Skipping)	Softmax Regression	Bus, MicroBus, Minivan, SUV, Sedan, Truck
Dong et al. (2015) [56]	Local and Global Features generated by a 2-Stage Semi supervised Convolution Neural Network	Softmax Regression	Bus, MicroBus, Minivan, SUV, Sedan, Truck

2.2 Vehicle Make (Logo) Recognition

A deeper level of classifying vehicles is to recognize their make (manufacturer) rather than their type. Most works achieve this through vehicle logo recognition (VLR) [57–63]. Various local feature extraction techniques have been used in the literature to build discriminating representations of logo images. While works like [57,58,60,64] employ SIFT-based representations, others use Histogram of Oriented Gradients (HOG) [59], Sharpness Histogram Features [61], Hierarchical Feature Maps selected by a Convolution Neural Network [62], or Statistical Random Sparse Distribution (SRSD) [63].

The different classification schemes proposed in the above-mentioned works include Nearest Neighbour-based matching [57,58,63,64], Support Vector Machines-based classifiers or ensembles [59–61], or Neural Network-based approaches [62]. In Table 2.2, we provide a brief summary of the features extraction, global representation and classification approaches employed in these recent and representative VLR works, along with the number of classes and images used to validate their works.

Table 2.2: Overview of recent works on Vehicle Make (Logo) Recognition

Work	Features	Classification	#Classes	#Images
Psyllos et al. (2010) [57]	SIFT	Enhanced Matching Scheme + Geometric Validation	10	1200
Yu et al. (2013) [58]	Dense-SIFT in Bag of Words	kNN, SVM, SIFT-Matching	14	840
Llorca et al. (2014) [59]	Histogram of Oriented Gradients (HOG)	SVM	27	3579
Ou et al. (2014) [60]	Dense-SIFT, weighted Locality-Constrained Linear Codes (LLC) in a Spatial Pyramid framework	SVM	15	1791
Badura and Skotnicka (2014) [64]	SIFT Patterns	Pattern Matching-based rules	30	1225
Xiao et al. (2015) [61]	Sharpness Histogram Features	Weighted Multi-class SVM Ensemble Model	8	800
Huang et al. (2015) [62]	Multi-layer Convolution Neural Network (CNN) based Hierarchical Feature Maps	Back Propagation Neural Network (last layer of CNN)	10	11500
Peng et al. (2015) [63]	Statistical Random Sparse Distribution (SRSD)	Multi-scale Scanning and Nearest Neighbor	56	3370

A major limitation in VMR or VLR works is the reliance on localizing logo regions in input images from video streams. Most approaches are inapplicable in real-time applications due to time-expensive features representation techniques, or slow logo detection schemes. In contrast to these prior works, our work aims to achieve vehicle make as well as model recognition, not relying on logo detection, but using the vehicles' front or rear faces themselves. In this way, we remove the need for a logo detection and localization module in vehicle classification systems, thereby increasing processing speed. In the next section, we shall review the state-of-the-art in vehicle make and model recognition.

2.3 Vehicle Make and Model Recognition

In this section, we provide a comprehensive review of state-of-the-art works in each of the three steps involved in most Vehicle Make and Model Recognition (VMMR) systems, as depicted in Figure 1.4: Features Extraction, Global Features Representation (Section 2.3.1), and Classification (Section 2.3.2).

2.3.1 Features Extraction & Global Features Representation

To describe the vehicle makes and models, various local features are extracted from the vehicle ROIs (Features Extraction), with or without embedding them into Global Features Representations. Table 2.3 summarises some of the features extraction and global representation techniques used in the VMMR works. The works like that of [65] use raw image features like Scale Invariant Feature Transform (SIFT [66]) to describe make-model instances. In fact, SIFT has been used by many VMMR works such as [9, 10, 16, 65, 67]. Due to high dimensionality and relatively slow computational speed of SIFT, some works have adopted the Speeded Up Robust Features (SURF [68]) (e.g., [10, 67, 69]) and the Histogram of Oriented Gradients (HOG [70]) (e.g., [7, 8, 10, 11]). Other features explored for VMMR include those based on edges, gradients or corners (e.g., by [6, 12, 15, 71–73]), as well as MPEG-7 descriptors such as Edge Histograms [74] [75] (e.g., by [67]).

In most approaches, the raw features are embedded into global representations of vehicle makes and models ([7, 9, 10, 67, 72]). As summarised in Table 2.3, there are different techniques proposed to integrate the raw image features into holistic global representations. The quality of a global features representation technique is assessed by its processing speed, computational complexity in forming the holistic representations, and the VMMR accuracy which reflects its discriminative capacity in representing the different makes and models while generalizing over the multiplicity issues within a make-model class.

Edge images of vehicles' faces have been considered by Munroe and Madden [72], as numerical feature vectors. Pearce and Pears [15] concatenate the Square-Mapped Gradients (SMG)

or Locally Normalised Harris Strengths (LNHS) as global feature vectors for the images. Varjas and Tanacs [12] also used concatenated SMG. The SMG based techniques require well-aligned ROIs with strictly frontal views, or planar projection of skewed views onto frontal-like views. However, as we show later in the thesis, our approaches are greatly successful in achieving an extremely accurate VMMR system even under a broad range of viewpoints (or vehicle orientations) without requiring projection onto perfectly frontal views.

A grid-based global representation of features is proposed by Hsieh et al. [10], in which the local features (SIFT, SURF, or HOG) extracted from frontal vehicle faces are grouped in a grid-wise fashion. Chen et al. [7, 13] proposed grid-based concatenation of HOG features from the vehicle images into a global ensemble representation. Using their dataset, we prove that our approaches perform significantly better. The grid-based schemes assume a fixed camera and are prone to failures in cases where the camera height, pitch or yaw may change, resulting in vehicle views which the system might not be trained for. Some works like Llorca et al.'s [8] use the positions and sizes of car emblems and HOG features of emblem regions to classify vehicle models, assuming the make is known. The car emblems such as model symbol and trim level were considered. However, it is unclear if their approach can achieve both make and model recognition.

The local features like SURF have been used by [67, 69] to build a dictionary which is then used to represent vehicle images by sparse vectors of occurrence counts of the dictionary words. Differently from their works, we propose and investigate optimized dictionaries in the context of VMMR challenges, through two schemes of dictionary building. Amongst the most recent works on VMMR is that of Fraz et al. [9]. They form a lexicon comprising of all the training images' features as words. The words of the lexicon are computed based on a Fisher Encoded Mid-Level-Representation (MLR) of image features like SIFT. Their MLR construction is computationally expensive, reported to consume about 0.4s per image, and hence not suitable for real-time VMMR. Unlike [9], we learn a dictionary by retaining only the key features of training images as codewords, and not all the features.

Table 2.3: Summary of Features Extraction and Global Features Representation approaches in VMMR works

Work	Local Features	Global Representations
Petrovic & Cootes (2004) [71]	Raw pixels, Sobel Edge Responses, Edge Orientations, Harris Corner Response, Normalised Gradients, Square Mapped Gradients (SMG)	Simple Concatenation of Local Features, Spectrum Phase
Dlagnekov (2005) [65]	SIFT	-
Munroe and Madden (2005) [72]	Canny Edges	Concatenation of edge image pixels
Lee (2006) [76]	-	Texture Descriptors based on Gray-Level Co-occurrence Matrix (GLCM)
Zafar et al. (2007) [77]	-	Fisher Feature Matrices (based on Linear Discriminant Analysis)
Clady et al. (2008) [73]	Oriented Contour Points from Sobel Edges	Simple Concatenation
Zafar et al. (2009) [78]	Localised Features based on Contourlet Transform	Contourlet Feature Maps (with localized properties)
Psyllos et al. (2011) [16]	Phase Congruency (Make) + SIFT (Model)	
Pearce and Pears (2011) [15]	Canny Edges, Harris Corners, SMG	LNHS and Concatenated-SMG
Jang and Turk (2011) [69]	SURF	Bag of Words
AbdelMaseeh et al. (2012) [14]	Local Shape Descriptor (LSD), Local Appearance Descriptor (DAISY over FAST)	Global Shape Context Descriptor (GSCD)
Baran et al. (2013) [67]	SIFT; SURF; Edge Histogram	Dictionary based Sparse Vector of Occurrence Counts
Varjas and Tanacs (2013) [12]	SMG	Concatenated-SMG

Contd. on next page

Table 2.3 – contd. from prev. page

Work	Local Features	Global Representations
Zhang (2013) [11]	Wavelets, Oriented Gradients	Gabor Wavelet Transform (GWT), Pyramid-HOG (PHOG)
Hsieh et al. (2014) [10]	SURF, SIFT, HOG	Grid-based Representation
Llorca et al. (2014) [8]	Emblem Position & Size, HOG of Emblem Regions	-
Fraz et al. (2014)[9]	SIFT	Fisher Encoding based MLR
Chen et al. (2015) [7]	HOG	Grid-based Concatenated HOGs, Sparse Representation (SR)
He et al. (2015) [6]	Edges- and Gradients-based	Fixed-length Numerical vectors, after normalisations and textural filtering

2.3.2 Classification Approaches

In the literature, there have been various classification approaches proposed for VMMR, based on the local features and/or global features representations of the make-model classes. A summary of the most representative of such works is given in Table 2.4 (The feature extraction and global representation approaches used in these works have been previously given in Table 2.3). For example, many works like [6, 14, 15, 65, 71–73, 77, 78] explored Nearest Neighbors (NN) classifiers based on simple brute-force matching scheme using local features or their global representations to match query images to the gallery images. A k-NN-based classification scheme was also used by Varjas and Tanacs [12], but with a correlation based distance metric. The brute-force pattern matching approach is very time consuming and hence not suitable for real-time VMMR. In addition to NN, Munroe and Madden [72] also used machine learning algorithms like C4.5 Decision Trees and Feed-forward Neural Networks as classifiers for VMMR. Other works who used Neural Networks include [16, 76]. He et al. [6] built an ensemble of neural networks for classification. However, such approaches based on edges from images severely suffer in cases of occlusions, and hence not applicable to real-life scenarios.

In [15], Naive Bayes classifiers were tested with a variety of features. In their approach, the accuracy degrades when ROIs are even slightly different than ground truth ROIs. The classification scheme adopted by Fraz et al. [9] includes matching a probe image's words with the gallery of lexicons in a brute-force manner, assigning the best matching lexicon's class as the predicted class. Such an exhaustive matching scheme makes their approach inapplicable to real-time VMMR systems. A 2-stage cascade classifier ensemble was proposed in [11]. In the first ensemble, classifiers such as k-NN, Multi-Layer Perceptron (MLP), Random Forest (RF), and SVM were employed. The second ensemble employed a Rotation Forest of MLPs. However, incorporating so many classifiers greatly decreases the processing speed, making the methods inapplicable to real-time VMMR systems.

Baran et al. [67] utilised a simple multi-class SVM trained over the sparse occurrence vectors. However, they did not investigate optimizing the dictionaries for VMMR. Their dictionaries are 50% larger than ours, yet with lower accuracies. On the other hand, [69] applied an image retrieval approach using the Lucene Search Engine library to retrieve matching vehicle images

followed by Structural Verification to narrow down the list of matched images. Unlike them, we propose optimized dictionaries and two SVM based classification schemes that are designed to solve VMMR issues. Moreover, the superiority of our approaches is proven by using a more challenging dataset. Hsieh et al. [10] employ a grid-wise ensemble of SVM classifiers, each of which is trained over SURF features from a specific grid-block over frontal vehicle faces. Other works using SVM based classifiers include [6, 78]. In [8], Bayesian Inference Voting was used based on linear SVM. AdaBoost was also tested in the context of VMMR in [6]. On the other hand, Chen et. al. [7, 13] propose a classification approach for VMMR, based on Sparse Representation and Hamming Distance.

In spite of the several works that have been published on the theme of VMMR, there are several challenges and issues yet to be addressed. Most of the prior works are based on datasets that do not represent multiplicity and ambiguity issues fairly. A majority of the works hardly meet real-time constraints. The multiplicity and ambiguity problems need to be tackled, perhaps through more representative and discriminative global features representations and enhanced classification techniques. The limitations of VMMR works reviewed in this section are summarized in the third column of Table 2.4. To overcome the shortcomings of prior works, this thesis proposes and investigates unexplored approaches for global representation and classification steps involved in VMMR. The superiority of the proposed approaches (in terms of processing speed and accuracy) is demonstrated using a recently published real-world dataset.

Table 2.4: Summary of Classification approaches used in some of the most representative VMMR works

Work	Classifier	Comments
Petrovic & Cootes (2004) [71]	Nearest Neighbors	Slow; Dependence on edges leads to failures under occlusions
Dlagnekov (2005) [65]	Brute-force Matching	Exhaustive matching strategy is very time-consuming
Munroe and Madden (2005) [72]	k-Nearest Neighbors, Neural Network, C4.5 Decision Trees	Fails under occlusions, due to edges-based techniques
Lee (2006) [76]	Neural Network	Prone to failure under occlusions due to dependency on pixel-based texture descriptors; No report on processing speed
Zafar et al. (2007) [77]	Euclidean Distance based matching	Low accuracies on a simplistic dataset; Viewpoint variations not considered; Slow (due to matching)
Clady et al. (2008) [73]	Nearest Neighbors	Occlusions degrade performance; Slow (due to exhaustive matching with training models)
Zafar et al. (2009) [78]	1-vs-all SVM, and k-NN	Prone to failure by occlusions (due to reliance on whole vehicle ROIs to build feature maps); Varying viewpoints not considered; No report of features representation and classification speed
Psyllos et al. (2011) [16]	Probabilistic Neural Network (PNN)	Tested on simple and non-occluded scenes; Viewpoint and illumination variations not considered; Very low accuracy even on simplistic dataset; Very slow processing speed (due to dependence on a logo recognition module)

Contd. on next page

Table 2.4 – contd. from prev. page

Work	Classifier	Comments
Pearce and Pears (2011) [15]	k-NN and Naive Bayes	Prone to failures under occlusions (due to considering whole vehicle ROI for global representation); Small and simplistic dataset used; Best performing scheme is very slow (>1s)
Jang and Turk (2011) [69]	Matching using Lucene Search Engine Library + Structural Verification	Toy car images used, not tested on real-world images; Matching strategy makes the approach too time-consuming; Best dictionary size is five orders of magnitude (way larger than our optimized dictionaries); Dictionary is not optimized for VMMR
AbdelMaseeh et al. (2012) [14]	Nearest Neighbors	Only sedan type vehicles; Very poor accuracy despite an unrealistic dataset
Baran et al. (2013) [67]	Multi-class SVM	Small dataset; Dictionary is 50% larger than ours, yet with lower accuracies; Not tested under occlusions
Varjas and Tanacs (2013) [12]	Nearest Neighbors	Exhaustive matching strategy is very time-expensive (1-25s); Depends on minimal perspective distortion of license plates
Zhang (2013) [11]	2-Stage Cascade Classifier Ensemble	Ignores Intra-Make Ambiguity (combines visually similar models of a make into one make-model class); Fails under occlusions or partial views of vehicles; No mention of processing speed

Contd. on next page

Table 2.4 – contd. from prev. page

Work	Classifier	Comments
Hsieh et al. (2014) [10]	Grid-based Ensemble of SVM	Requires aligned ROIs; Assumes fixed ROI aspect ratios for all makes and models; Change in camera pitch angle would affect ROIs
Llorca et al. (2014) [8]	Linear SVM based Bayesian Inference Voting	Make+Model recognition accuracies are unclear, results indicate either make or model recognition rates; Processing speed not indicated
Fraz et al. (2014) [9]	Brute-force Matching	High-dimensional features and exhaustive matching make it very time-expensive; Not applicable for real-time applications; No tests in occlusions;
Chen et al. (2015) [7]	Sparse Representation-based and Hamming Distance-based	Best performing scheme has an extremely low speed (0.46 fps); Discriminative capability of local features not considered
He et al. (2015) [6]	Ensemble of Neural Networks; SVM; kNN; AdaBoost	Not applicable in real-time applications (obtained speed is around 1 fps); Occlusions are not considered

2.4 Conclusion

In this chapter, we have provided a comprehensive review of the most representative works done in VTR, VMR and VMMR. Realizing the need to have a fine-grained classification of vehicles into makes and models, rather than types and makes alone, we focussed more elaborately on the approaches proposed in the literature for the different steps in VMMR, namely: Features Extraction, Global Features Representation, and Classification. Keeping in mind the multiplicity and ambiguity challenges in VMMR, we realize the need for developing improved methods to represent and classify vehicle makes and models, by enhancing both discriminative capacity and generalization capability of the approaches. The focus of this thesis is on the latter two components of VMMR systems, i.e., to enhance the state-of-the-art in global features representation and classification approaches, in the context of real-time automated VMMR.

Chapter 3

Features Extraction and Global Features Representation

To describe the vehicle makes and models, various local features are extracted from the vehicle ROIs, with or without embedding them into Global Features Representations (GFRs). In Table 2.3, we mentioned the features extraction and global representation techniques used in the VMMR works. In this chapter, we first describe and discuss the most representative existing *Feature Extraction* and *Global Features Representation* techniques used in VMMR (Sections 3.1 and 3.2, respectively). Then, we present and describe the approaches we propose and investigate for VMMR (Section 3.3).

3.1 Existing Feature Extraction Techniques

3.1.1 Keypoint-based

Under the keypoint-based techniques, we include those feature extractors and descriptors that consider only the keypoint patches for computing the descriptors, and not the entire image or ROI. The two most popular keypoint-based features are SIFT and SURF.

3.1.1.1 Scale-Invariant Feature Transform

The Scale-Invariant Feature Transform (SIFT) is a bundled feature detector and descriptor. The detector extracts keypoints and patches from an image such that they are consistent across variations in scale, illumination, rotation, and viewpoints. The descriptor then generates a compact and robust signature for each of these patches, which identifies its appearance. The method was initially proposed by Lowe in [66] and later enhanced in [79]. Since then, SIFT has been popularly used in object detection and recognition, image retrieval, and image classification. It has also been used in attempts to solve VMMR problems, such as [9, 10, 16, 65, 67]. There are four steps involved in extracting SIFT features, described as follows:

1. *Scale-space Extrema Detection*: Across various image scales and locations, scale- and orientation-invariant keypoint candidates are identified using Difference of Gaussian (DoG) images. The scale-space of a given image comprises of several octaves, as shown in Figure 3.1. For each successive octave, the image size is downsampled by a factor (e.g., of $^1/_4$). Within each octave, the image is progressively blurred through a Gaussian Blur operator with increasing scale. From every consecutive pair of Gaussian blurred images, the Difference-of-Gaussian (DoG) images are formed. Finally, using maxima or minima technique of [66], the keypoint candidates are detected from these DoG images.

2. *Keypoint Localization*: In this step, unstable keypoint candidates are filtered out. The unstable keypoints are those which have low contrast or do not represent corners (i.e., those that represent edges or flat regions).

3. *Orientation Assignment*: To provide for rotation-invariance, each keypoint location is assigned one or more dominant orientations based on its local region's gradients, and the respective keypoint descriptors are represented with respect to these dominant orientations.

4. *Keypoint Description*: In the standard SIFT technique, a 16×16 neighbourhood of points (pixels or sub-pixels) is selected around each keypoint, divided into 16 windows of 4×4 each. The keypoint's dominant orientation (computed in the previous step) is subtracted from each neighbourhood point's gradient orientation. Then, the gradient magnitudes and

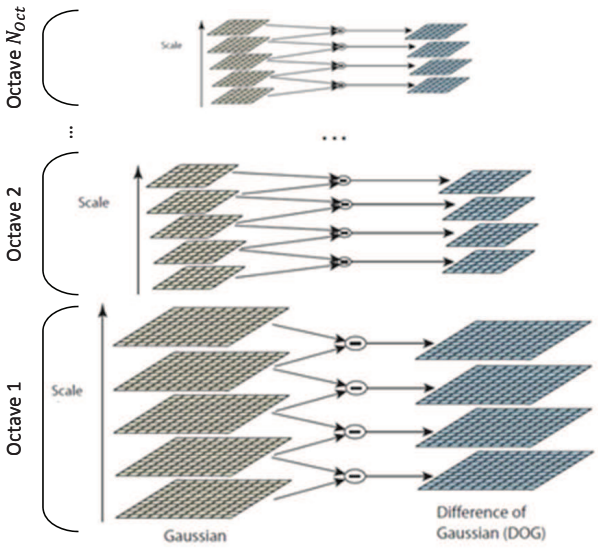


Figure 3.1: The Scale-space generation step to obtain DoG images for an input image. (adapted from [79])

directions at each point within a 4×4 window are computed and collected in an 8-bin histogram (see Figure 3.2). For example, a point with gradient direction of say 15° will be added to the 0°-44° bin of the 8-bin histogram. The quantity added to the respective bin of the histogram is based on the point's gradient magnitude multiplied with a Gaussian-weight computed based on its distance from the keypoint. The farther the point is from the keypoint, lesser would be its weight. The Gaussian weighting window is a 2D Gaussian function with σ in both dimensions equal to half the width of the keypoint neighborhood (i.e., $\frac{16}{2} = 8$, in standard SIFT). The Gaussian-weighting window is applied to prevent small changes in window positions from causing sudden changes in descriptor. It also reduces the influence of points farther away from the descriptor center (as they cause most mis-registration errors, according to [79]). Finally, a descriptor of $4 \cdot 4 \cdot 8 = 128$ values is obtained for every keypoint.

The scale-invariance of SIFT features is mainly due to the keypoints being extracted over varying sizes of the original image and with varying scales of the Gaussian blur. The rotation invariance is provided by considering the relative gradient orientations (with respect to a keypoint's dominant orientation) while computing the keypoint descriptor. Illuminance invariance is achieved through vector normalization.

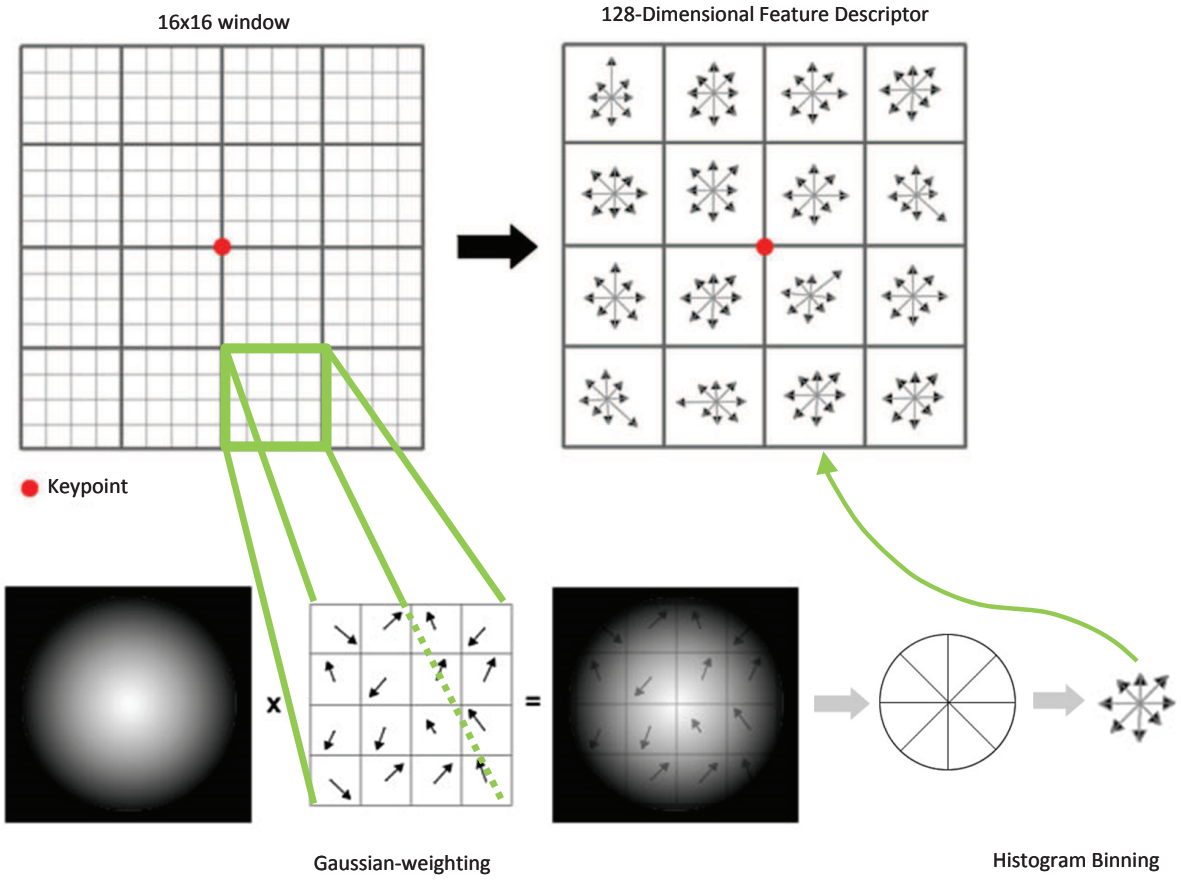


Figure 3.2: Computing the SIFT descriptor around a keypoint. (Top-Left) The 16x16 neighborhood of the red keypoint. (Bottom-Left) Applying Gaussian-weighting to the gradient orientations computed at each point in a window. (Bottom-Right) The circle demarcating the distribution of orientations into 8 bins (i.e., the Histogram binning step). (Top-Right) The overall collection of sixteen 8-bin orientation histograms from the 4x4 windows. The lengths of arrows represent gradient magnitudes while orientation represents gradient direction. (adapted from [80])

3.1.1.2 Speeded Up Robust Features

The Speeded Up Robust Features (SURF) are another type of scale-invariant and rotation-invariant feature detectors and descriptors. It was proposed by Bay et al. in [68], who proved its efficiency (in terms of processing speed and accuracy) over other feature descriptors such as SIFT. SURF has shown encouraging performance in various applications such as ADAS [81] and VMMR [10, 67, 69]. In what follows, we describe the procedure to build the SURF descriptors.

- *Interest Points:* The interest points are located using a Hessian matrix approximation on an integral image. The second-order partial derivatives of an image are derived from the Hessian matrix to describe its local curvatures. The fast processing speed of SURF is due to the usage of integral images in approximating the second-order Gaussian derivatives with blob-like feature detectors.

- *Descriptor Computation:* Around each interest point obtained in the previous step, a neighbourhood of 4×4 blocks is extracted. Each block has 5×5 regularly spaced sample points. For each block b (where $b = 1, ..., 16$), a 4-D descriptor vector f is formed by summing up the Haar wavelet responses (d_x and d_y) of each sample point within the block.

$$f_b = (\Sigma dx, \Sigma|dx|, \Sigma dy, \Sigma|dy|) \tag{3.1}$$

  Figure 3.3 illustrates the SURF descriptor computation method for one block in the square region centred around the keypoint. The SURF descriptor for the square region around an interest point is formed by concatenating the 4-D vectors of each block, as represented by:

$$f = [f_1, f_2, ..., f_{16}] \tag{3.2}$$

- *Orientation Assignment*: To provide for rotation invariance, every interest point and descriptor is assigned a dominant orientation. First, Gaussian weighted x and y Haar-wavelet responses for each interest region are calculated. Then, using a rotating orientation window, orientation of the largest sum of responses is estimated as the dominant orientation of the interest point and descriptor. This step can be skipped in cases where rotation invariance is not needed, to yield a faster SURF, called Upright SURF (U-SURF).

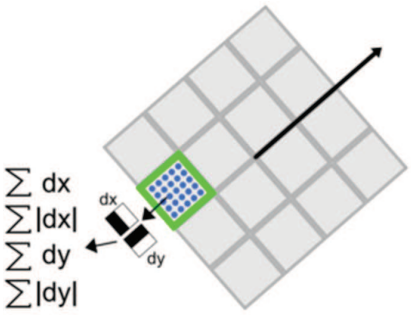


Figure 3.3: Computing the SURF descriptor around an interest point [82].

3.1.2 Window-based

Under the window-based techniques, we include feature extractors and descriptors that do not work on keypoints. Instead, the entire image or ROI is considered to compute the descriptors. We provide a brief description of some of the popular window-based feature descriptors applied in image classification problems, such as Histogram of Oriented Gradients and Multi-scale Block Local Binary Patterns. The former has been used in VMMR works such as [7, 8, 10], while the latter in VTR works such as [50].

3.1.2.1 Histograms of Oriented Gradients

The Histograms of Oriented Gradients (HOG) were introduced by Dalal and Triggs [70] for robust human detection, and have been widely used in many works on object recognition and ADAS [83–85]. Figure 3.4 illustrates the HOG representations and descriptor vectors for vehicle front faces. To compute the HOG descriptor of an image or ROI, a sliding window approach is used. A given window is divided into a grid of n overlapping blocks, each block with p cells. Within each cell, weighted pixel-wise gradients are accumulated into an orientation histogram of h bins, in a similar fashion as SIFT. The histograms of cells within a block are concatenated to give the block-level histogram vector.

To provide strong illumination invariance, a robust normalization process is run on each block. Then, the normalized histograms of all overlapping blocks within a window are concatenated to give the window-level histogram vector. Figure 3.5 shows the flowchart of building

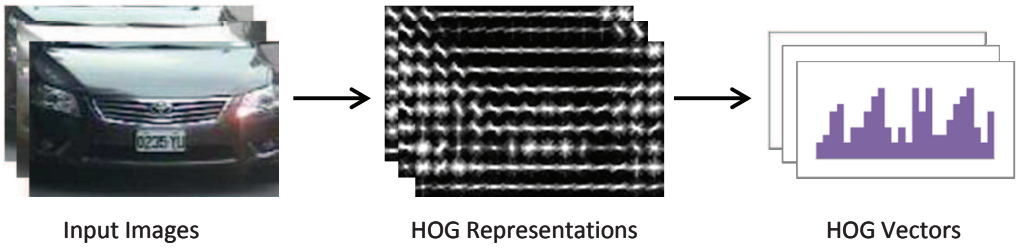


Figure 3.4: Illustrative samples for HOG representations and descriptor vectors, obtained after sliding window-based HOG feature extraction over vehicle front faces.

HOG descriptor for one of the sliding windows. Spatial and angular linear interpolation, and in some cases Gaussian windowing over the blocks, are used to reduce aliasing during voting. The blocks overlap spatially so that each cell appears several times with different normalizations, as this was shown to improve performance in [70]. The histograms from all sliding windows over the input image or ROI are then concatenated together to give its overall HOG descriptor.

3.1.2.2 Multi-scale Block Local Binary Patterns

The Multi-scale Block Local Binary Patterns (MB-LBP) were proposed by [86] and applied to the problem of face recognition. In the original LBP operator proposed by [87], the image pixels are labelled based on comparing each pixel with its 3×3 neighbourhood, and representing the comparisons as a binary string or decimal number. Figure 3.6a shows a toy example where the center pixel is compared to its 3×3 neighbourhood in a clockwise manner. For every pixel-wise comparison, if the center pixel is of greater value, a "0" is inserted in the respective position of the binary string; otherwise a "1" is inserted. The overall texture descriptor is then a histogram of all such pixel labels. The enhancement proposed in MB-LBP by [86] basically considers sub-regions in comparisons, instead of the pixels. Around each pixel, a square neighbourhood of $s \times s$ pixels is considered. The neighbourhood is divided into non-overlapping blocks (sub-regions) of dimensions ${}^{s}/_{3} \times {}^{s}/_{3}$. The average gray values of $(s-1)$ sub-regions are compared with that of the central sub-region. Figure 3.6b shows a toy example where a 9×9 MB-LBP operator ($s = 9$) is illustrated.

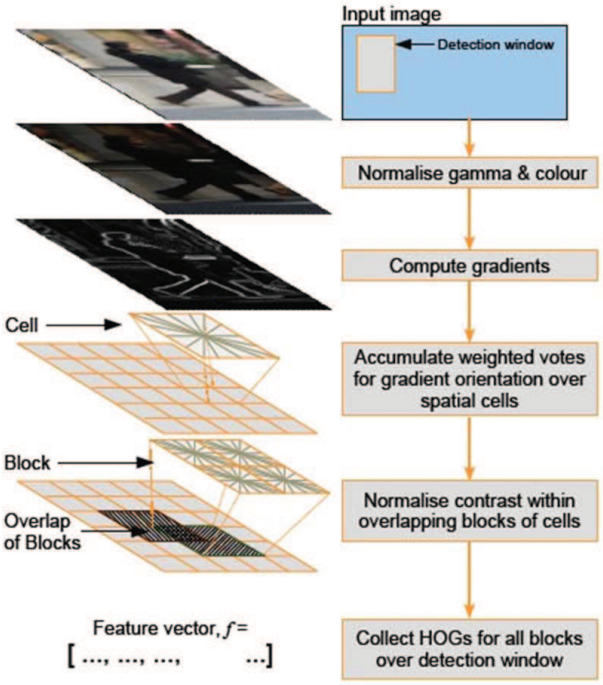
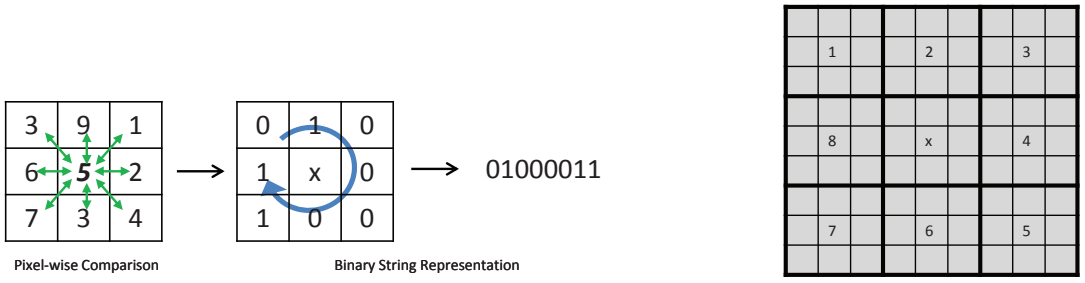


Figure 3.5: Computing the HOG descriptor for one of the sliding windows over an input image or ROI. (reprinted from [70])



(a) (b)

Figure 3.6: (a) Computing the LBP binary string (label) for a pixel, considering its 3x3 neighbourhood: comparisons are done between pixels; (b) A 9×9 MB-LBP operator with 8 subregions around a central region: comparisons are done between average gray values of subregions.

3.2 Existing Global Representation Techniques

The local features extracted from the images are often used to build a global representation of the object of interest in the image, e.g., a vehicle of some make-model (MM) class. Different global features representation techniques used in VMMR works were mentioned in Table 2.3. In this section, we provide a comprehensive description and discussion of most representative of these global representation schemes: (1) Grid-based Vehicle MM Representation, (2) Lexicon based Mid Level Representation, and (3) Local Features Concatenation. By discussing the limitations in these schemes, we build the motivation for our proposed solution to describe vehicle makes and models.

3.2.1 Grid-based Representation

The grid-based representation for vehicle MMs, proposed by [10] and [7], utilizes the local features (such as HOG, SIFT or SURF) of a vehicle ROI to build a global representation while capturing their spatial location information to some extent.

3.2.1.1 Description

The vehicle ROI (R_v), obtained from the Vehicle Detection module, is divided into a grid of equally sized non-overlapping blocks through two division schemes: (1) *Direct Division*, and (2) *Inside Division*. While the former division scheme divides R_v into a grid of $m \times n$ blocks including the hood region, the latter divides R_v into a grid of $m \times n$ blocks excluding the hood region. The hood region was found to be less informative and less discriminative across various MM classes. Figure 3.7a and Figure 3.7b illustrate the two division schemes for a vehicle.

The Direct Division scheme can be implemented by two methods of determining the upper R_v boundary: (a) based on the hood boundary, or (b) based on the width of R_v (using a learned width-height ratio). The former is named Wide Direct Division (WDD), and the latter is referred to as Narrow Direct Division (NDD). Figures 3.8a and 3.8b illustrate the two methods of Direct Division. In [10], it was shown that WDD and NDD methods perform inferior in comparison to

(a) (b)

Figure 3.7: Grid-based Representation of vehicle makes and models: (a) Direct Division scheme; (b) Indirect Division scheme. (reprinted from [10])

Inside Division-based methods. The WDD method results in an inferior performance probably due to the inclusion of hood region, which has negligible or no discriminative information. On the other hand, since the NDD depends on a fixed width-height ratio applied to all makes and models, there may be some MMs for which this ratio is not applicable. Hence, informative face regions (parts of the air grille or headlights, etc.) could get excluded, thereby degrading MMR performance.

The Inside Division scheme is implemented in two ways: (a) *Symmetrical Inside Division (SID)*, and (b) *Full Inside Division (FID)*. In the former, only half of the grid blocks are selected (e.g., left half). In the latter, all grid blocks are considered. Figures 3.9a and 3.9b illustrated the SID and FID methods.

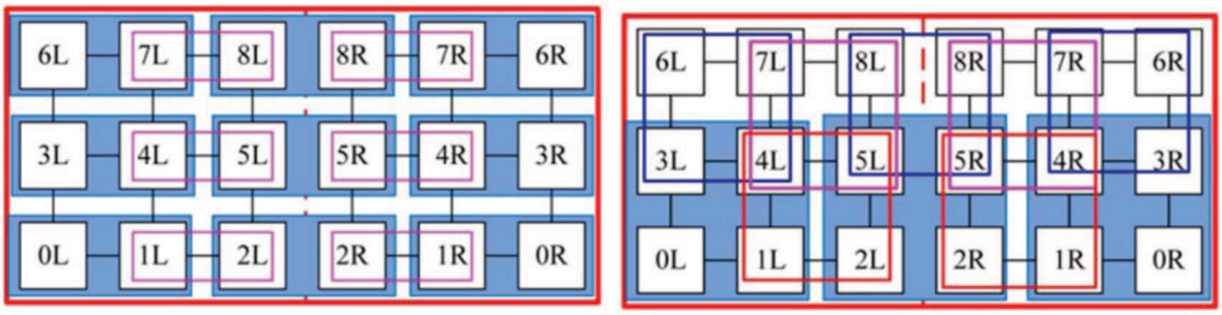
The global representation is built using two strategies: (1) *Block-wise Concatenation*, and (2) *Block-wise Grouping*, depending on the kind of local features used. For window-based local features like HOG, *Block-wise Concatenation* is used, in which the block descriptors are concatenated according to several combination patterns. Figures 3.10a and 3.10b illustrate two examples of block combination patterns. For a grid with 3×6 blocks ($m = 3, n = 6$), concatenating every one (1×1), two (1×2), four (2×2), six (2×3), nine (3×3), and eighteen (18×1) blocks result in a total number of 51 block combination patterns. The size of the global representation would then

(a) (b)

Figure 3.8: Two methods of Direct Division scheme: (a) Wide Direct Division (WDD); (b) Narrow Direct Division (NDD). (reprinted from [10])

(a) (b)

Figure 3.9: Two methods of Inside Division scheme: (a) Symmetrical Inside Division (SID); (b) Full Inside Division (FID). (reprinted from [10])



(a) (b)

Figure 3.10: Different block combination patterns for: (a) Combining every two (1×2) adjacent blocks; (b) Combining every four (2×2) adjacent blocks. (reprinted from [10])

be $51 \cdot N_{bins}$, where N_{bins} is the number of HOG bins in a block descriptor.

On the other hand, for keypoint-based local feature like SURF and SIFT, *Block-wise Grouping* is used, in which descriptors are simply grouped together block-wise. So, from every image, $m \cdot n$ groups of local descriptors are collected, to be subsequently used in training a classifier corresponding to each block (i.e., an ensemble of $m \cdot n$ classifiers is achieved).

3.2.1.2 Limitations

There are several limitations with the Grid-based representations of vehicle MMs proposed by [10] and [7], summarised as follows:

- The WDD method considers the hood region, in which there is little or no discriminative information for VMMR.
- The NDD method relies on a fixed width-height ratio of learned from ground truth ROIs, to determine the upper boundary of a R_v. This fixed ratio may not apply to all MMs, as some MMs may have ROIs of different aspect ratios.
- The SID method considers only left (or right) half of any R_v, based on which the other half is estimated. If the selected half is under occlusion, the estimated half will also represent occluding object's features, and hence, the MMR system will fail.

- The FID method is also highly prone to failures in cases of occlusions. The local features from a block that is over the occluding object could confuse the respective block's classifier.

- If the pitch angle of the camera capturing the incoming vehicles is different from that in training phase, the resulting face-views, R_v's and $m \times n$ grid-based representations would be different and could degrade the MMR accuracy.

3.2.2 Mid-Level Representations

In the Mid-Level Representation (MLR) techniques, the densely extracted local features of an image or ROI are encoded into a single vector representation. Examples of such encoding techniques for MLRs include: Locality Sensitive Linear Coding (LLC) [88], Vectors of Locally Aggregated Descriptors (VLAD) [89,90], Super Vector Coding (SVC) [91], and Fisher Vector (FV) [92,93]. Amongst these, the FV-based MLR has been shown to be successful for VMMR by Fraz et al. [9]. We describe here their MLR method and discuss several drawbacks and limitations in their approach.

3.2.2.1 Description

The FV-based representations for images are built using the approach devised by Perronnin et al. [93] for image classification problems. From a certain number of training images, local descriptors are collected and a Gaussian Mixture Model (GMM) is built using a pre-specified number of Gaussian distributions, N_G.

For a given image, scale- and orientation-invariant keypoints are extracted based on the Difference of Gaussians (DoG) detector [79]. Then, from a patch around each keypoint, densely sampled feature descriptors (such as SIFT or PCA-SIFT [94]) are extracted. To build a keypoint-based patch's FV representation, the average first- and second-order differences between its densely sampled descriptors and GMM centers are computed.

Let $f_1, ..., f_K$ be a set of K densely sampled local feature descriptors extracted from a keypoint-based patch in an image. The first- and second-order differences between the feature descriptors

and GMM centers are then given by:

$$\alpha_g = \frac{1}{K\sqrt{w_g}} \sum_{j=1}^{K} s_{gj} \Sigma_g^{-1/2} (f_j - \mu_g) \tag{3.3}$$

$$\beta_g = \frac{1}{K\sqrt{2w_g}} \sum_{j=1}^{K} s_{gj} \left[(f_j - \mu_g) \Sigma_g^{-1} (f_j - \mu_g) - 1 \right] \tag{3.4}$$

where s_{gj} is the soft assignment weight of j^{th} feature descriptor ($j = 1, ..., K$) to the g^{th} Gaussian ($g = 1, ..., N_G$), and Σ_g's are the diagonal covariance matrices of the GMM. All the computed α's and β's are then concatenated to build the Fisher Vector F for the keypoint-based patch:

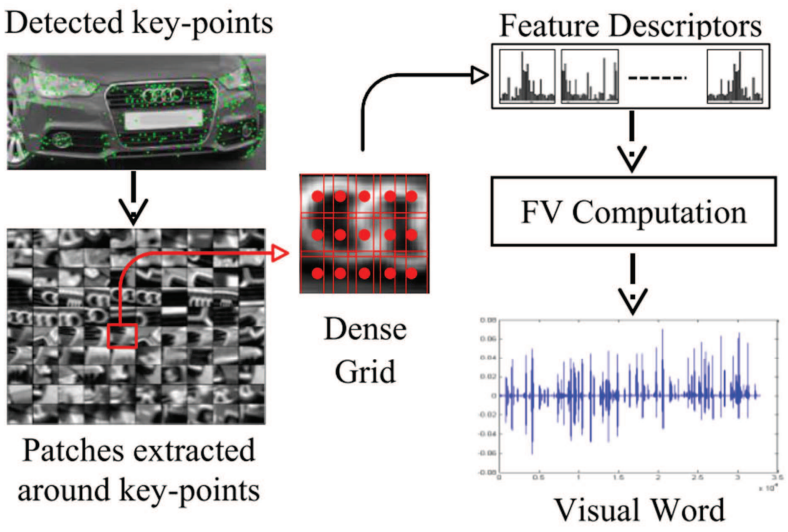
$$F = [\alpha_1, \beta_1, ..., \alpha_G, \beta_G] \tag{3.5}$$

In this way, every keypoint-based patch yields a FV descriptor referred to as a *visual word* of size $2 \cdot N_G \cdot d$, where d is the dimension of an f_j. Figure 3.11a illustrates the flowchart of generating a visual word for a keypoint-based patch, through FV descriptor computation. The global representation of an image is then simply the set of all FV-based visual words from the image. If there are Np_i keypoint-based patches in an image i, then the size of its global representation is $2 \cdot N_G \cdot d \cdot Np_i$. In [9], upto 300 keypoint-based patches were obtained from a 128×256 sized image. This means that the global representation of this image, using a GMM of $N_G = 256$, and PCA-SIFT of $d = 64$, would be of size $2 \cdot 256 \cdot 64 \cdot 300 = 9830400$ (very high dimensional).

The visual words from all training images of a class c where ($c = 1, ..., Nc$) are pooled to form a sub-lexicon S_c. Each S_c is of dimensions $(\sum_i^{M_c} Np_i) \times M_c$, where M_c is the number of training images of class c, and $\sum_i Np_i$ is the total number of keypoint-based patches in these M_c images. The sub-lexicons from all Nc classes are collected together in a comprehensive lexicon L, which is an over-complete description space of all Nc classes.

$$L = [S_1, S_2, ..., S_{Nc}] \tag{3.6}$$

In the testing phase, the visual words are computed for the keypoint-based patches from the vehicle ROI in the probe image. The similarity score of a probe image with each class' sub-lexicon is computed based on the visual-words matching. A Euclidean distance-based matching



(a)

Figure 3.11: The Fisher Vector descriptor computation pipeline for a keypoint-based patch. (reprinted from [9])

is performed between each visual word of the probe image and the visual words of each sub-lexicon S_c. The similarity score of a probe visual word and the closest S_c's visual word is taken as the probe visual word's similarity score with S_c. The sum of all probe visual words' similarity scores with respect to S_c is taken as the similarity score of the probe image with class c. The probe image is assigned the class that wins the highest similarity score.

3.2.2.2 Limitations

The major limitations in the FV-based MLR technique proposed by [9] are as follows:

- The densely sampled descriptor computation within every patch is very laborious that decreases the processing speed.
- The visual words are high dimensional, owing to dense sampling, which in turn leads to very high dimensional global representations. Consequently, computational complexity and memory requirements increase and make it difficult to train machine learning based classifiers such as SVM.

- Using a brute-force matching scheme for classification is also very time consuming and makes the technique inapplicable to real-time scenarios.
- In forming the sub-lexicons, all keypoints-based patches are considered, irrespective of their discriminating and representative capacities, which could cause classifier confusion.
- Occlusions could degrade the performance, as the keypoint-based patches from an occluding object will also be considered in matching.

3.2.3 Local Features Concatenation

Many works utilize a simple local features concatenation method to build global representations of images. These mostly include works that use edges- or gradients–based local features extraction [6, 12, 15, 71–73]. A pixel-wise concatenation is made from the vehicle ROI, to build its global representations.

For example, Figure 3.12 illustrates the different image features and global representations used in [71] to describe vehicle makes and models. Starting from the top row, seeing left-right, the first image is the input vehicle ROI, followed by Harris corner responses, spectrum phase, vertical Sobel edge responses, and finally the horizontal Sobel edge responses. From the bottom-left, Figure 3.12 shows a sample weight vector for horizontal Square-Mapped gradient responses, vertical and horizontal components of Locally Normalized gradients, and vertical and horizontal components of Square Mapped gradients. These are considered as fixed-length numerical feature vectors based on simple pixel-wise concatenation.

In Figure 3.13, [6]'s features extraction and global representations are illustrated. From left column onwards, the images correspond to raw vehicle ROIs, illuminance normalised ROIs, Multi-Scale Retinex (MSR) results for the ROIs, and finally the normalized features of the ROIs. The normalized features of the whole ROI are simply concatenated into a single vector.

Figure 3.12: Petrovic and Cootes' features extraction and global representations of vehicle makes and models. (reprinted from[71])

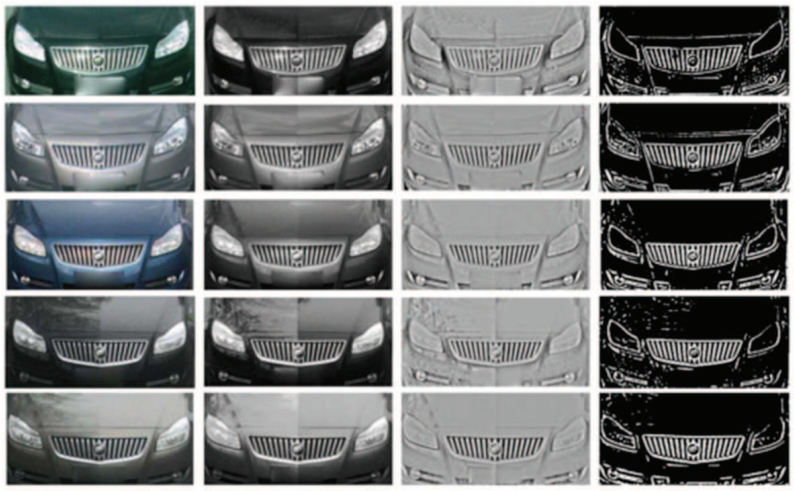


Figure 3.13: He et al.'s features extraction and global representations of vehicle makes and models. (reprinted from [6])

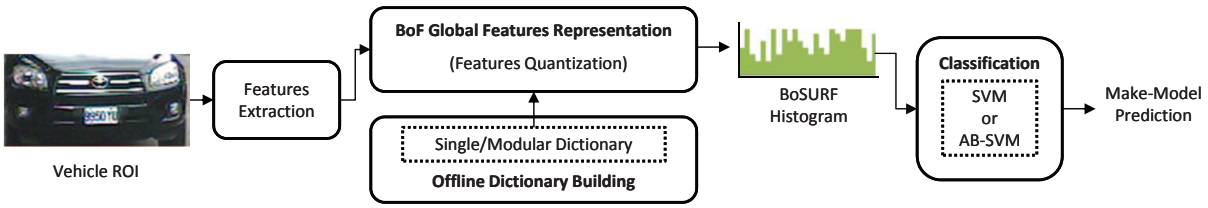


Figure 3.14: Overview of the proposed BoF-based approaches for VMMR.

3.3 Proposed Approaches

Although many attempts have been made to build more informative and discriminating global features representations of vehicle makes and models, only a handful have been proven to be successful across high degrees of multiplicity and ambiguity, yet with many limitations such as slow processing speeds and failures under occlusions. The objective of this work is to propose a highly robust yet efficient global features representation approach for VMMR, that accounts for intra-class differences and inter-class similarities, thereby solving the multiplicity and ambiguity issues in VMMR.

We propose to capture the dominant characteristic features from different make-model classes in an optimised dictionary (to tackle the ambiguity issues) (See Section 3.3.3). Different samples from each class are represented with respect to this learned dictionary, to describe the make-model class (See Section 3.3.4). In Section 4.2, we propose efficient multi-class Support Vector Machines-based classifiers which are trained to simultaneously learn the differences between global representations of different make-model classes and the similarities between different generations of the same make-model class (thereby solving the multiplicity problem).

To describe objects of interest using their raw image features embedded into global representations, the Bag-of-Features (BoF) framework [95, 96] has been very successful and widely adopted in the works on object recognition [97], scene classification [98], action recognition [99], image classification [100], and image retrieval [101, 102]. However, to the best of our knowledge, BoF has not been extensively studied in the context of VMMR. Inspired by the success of BoF in the aforementioned works, we propose and investigate BoF-based approaches with optimized dictionaries and classifiers, to solve the issues in real-time VMMR.

3.3.1 Overview

The overview of proposed BoF-VMMR approaches is illustrated in Figure 3.14. Apart from Features Extraction, there are three main steps involved in the proposed BoF approaches for VMMR, as shown in Figure 3.14: (1) Offline Dictionary Building (Section 3.3.3), (2) Global Features Representation (Section 3.3.4) and (3) Classification (will be presented in Section 4.2).

In this work, we investigate two dictionary building schemes in the context of real-time VMMR: (1) *Single Dictionary (SD)* (See Section 3.3.3.2), and (2) *Modular Dictionary (MD)* (Section 3.3.3.2). The SD is based on the standard method of dictionary building in the BoF framework, in which dictionary codewords are learned from the collective pool of training data (i.e., of local image features) from all combined classes. The MD, on the other hand, is composed of many individual dictionaries, each corresponding to a make-model class. The codewords of each such sub-dictionary are learned from the training data of the respective make-model class. To build the BoF representations from local features (e.g., SURF) of vehicle ROIs, SD or MD are used.

3.3.2 Features Extraction

For Features Extraction, SURF has gained wide popularity in many computer vision applications. It has been shown to have higher accuracy and speed in comparison to other feature descriptors in the context of object recognition, image classification, etc. [68]. Both the *Offline Dictionary Building* and the *Global Features Representation* steps rely on local image features such as SURF [68]. In fact, SURF can be easily replaced with any other good feature descriptor in our BoF-VMMR approaches. For the purpose of completeness, we provided a brief description of how SURF are extracted and represented, in Section 3.1.1.2. Encouraged by the success of SURF in some VMMR works such as [10, 13], we employ SURF as the building blocks of our global features representations.

3.3.3 Offline Dictionary Building

The training images of all classes are used to extract their local features (e.g., SURF). The key features (*codewords*) are then retained in a "bag" or *dictionary*. We capture and describe the overall appearance of the front or rear face for each vehicle make and model using the built dictionary. The dictionary can be considered as a compact representation of the key features from training images of all classes. Just as a text document comprises of textual dictionary words, an image can be regarded to be comprised of codewords from a visual dictionary. The vehicles' images are represented as BoF features, which are histograms of occurrences of the dictionary codewords. Building the dictionary is usually done offline and only when needed, so that it may be used in the training and testing phases. The Single- and Modular-Dictionary schemes proposed and investigated in this work are described in this section. An overview of the SD and MD schemes is depicted in Figures 3.15a and 3.15b respectively. The pseudocode for the Offline Dictionary Building step is given in Algorithm 1.

Let $\mathbf{I}$ represent the set of training images for Nc number of classes, as shown in Equation 3.7, where $\mathbf{I_i}$ represents the set of training images of class i in the dataset being used.

$$\mathbf{I} = \{\mathbf{I_1}, \mathbf{I_2}, ..., \mathbf{I_{Nc}}\} \tag{3.7}$$

From each j-th image in $\mathbf{I_i}$, we extract its set of local features, F_j, as represented in Equation 3.8, where f_p is the p-th local feature in image-j and Np_j is the number of local features extracted from the j-th image.

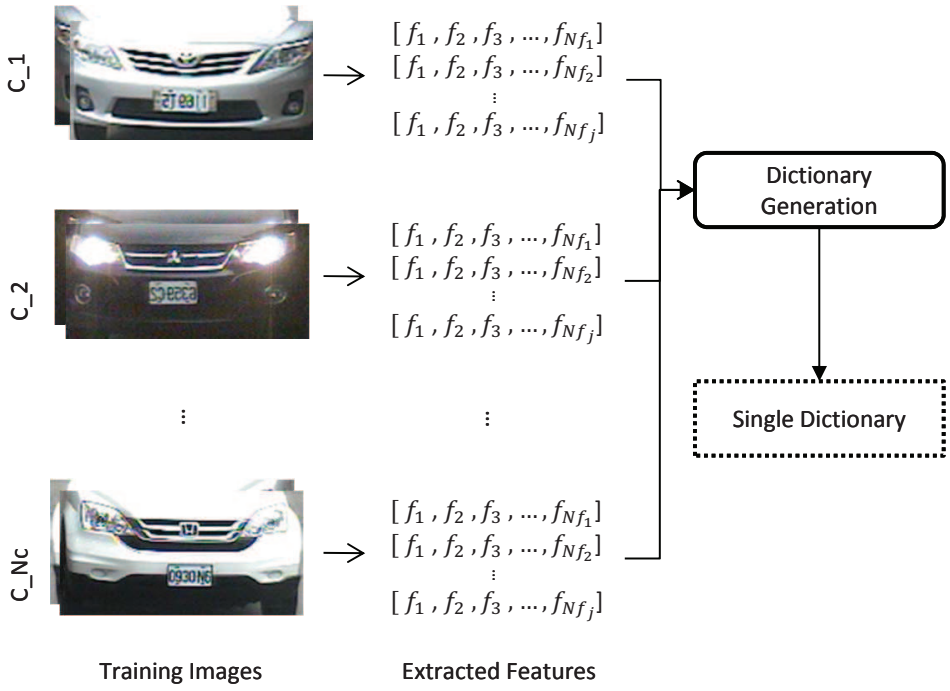
$$F_j = \{f_p | p = 1, ..., Np_j\} \tag{3.8}$$

The set containing all the features from N_i images of a class i is then

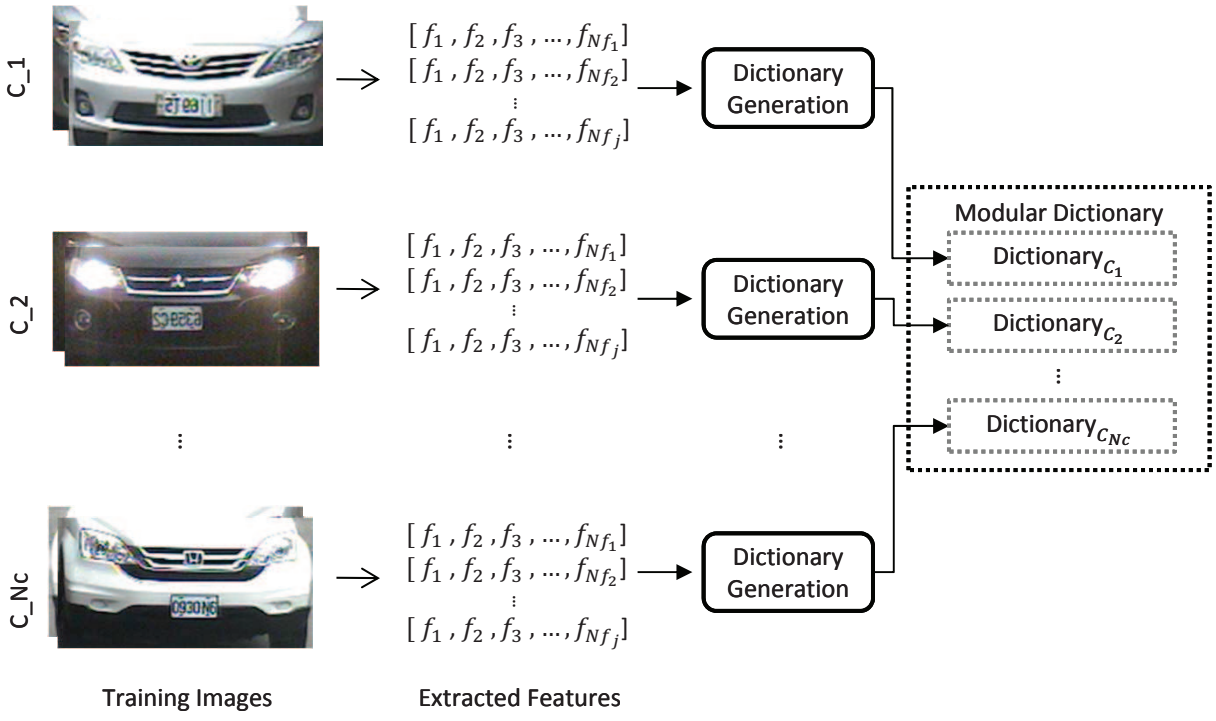
$$\mathbf{F_i} = \{F_j | j = 1, 2, ..., N_i\} \tag{3.9}$$

The pool of features from images of all classes is represented by $\mathbf{F}$ in Equation 3.10:

$$\mathbf{F} = \{\mathbf{F_i} | i = 1, 2, ..., Nc\} \tag{3.10}$$



(a)



(b)

Figure 3.15: Offline Dictionary Building in our BoF-VMMR approaches: (a) Single-Dictionary Building Scheme, (b) Modular Dictionary Building Scheme.

3.3.3.1 Dictionary Generation

Each feature descriptor f_x can be regarded as a point in the feature-space (e.g., for SURF, it would be a 64-dimensional space). The process of *Dictionary Generation* involves grouping all such training data points in high-dimensional space into a specified number of clusters. To initialize the clusters, we adopt the k-means++ technique [103] and select the initial set of cluster centres. Thereafter, the k-means clustering technique is applied to perform the clustering. Firstly, for each cluster centre, its 100 nearest neighbours are found from the data points. If a data point was close to more than one cluster centre, it gets assigned to the closest centre only. Thereafter, each cluster centre is updated to be the average of itself and its cluster members (nearest neighbours). Each data point is then re-grouped to the new cluster centre closest to it. This process of updating cluster centres followed by re-grouping of data points based on the updated centres is repeated a number, say 1000, times to ensure stability of the clusters. The cluster centres are stored as *codewords* of the respective dictionary.

3.3.3.2 Single Dictionary (SD)

To build the SD (denoted by **D**), key features are selected through Dictionary Generation using the combined set of training features from all classes, i.e., **F** of Equation 3.10. The SD is represented in Equation 3.11, where cw_k are its *visual words* or *codewords*. The number of clusters (or codewords) determines the overall Dictionary Size, S_D. See Figure 3.15a for an overview of the SD scheme. We refer to the BoF approach based on the SD scheme as BoF-SD.

$$\mathbf{D} = \{cw_k | k = 1, ..., S_D\} \tag{3.11}$$

The motivation to build a dictionary using the combined set of features from all classes is to obtain the key features across multiplicity and ambiguity of vehicle makes and models. This helps to build more discriminative global representations

3.3.3.3 Modular Dictionary (MD)

In this second scheme of dictionary building, we build the main dictionary (denoted by $\mathbf{D_M}$) by combining individual dictionaries of each class, motivated by the results of [104]. The intuition

behind this scheme is that, in the Single Dictionary scheme, several distinct features could be clustered under the same codeword due to their closeness. More importantly, having a modular dictionary greatly reduces the time consumed in dictionary building and also provides flexibility. If classes need to be added (removed), their respective dictionaries can be flexibly appended (deleted) to (from) the main dictionary without requiring a reconstruction the entire dictionary, thus saving a considerable amount of time. The *Modular Dictionary* ($\mathbf{D_M}$) is formed as:

$$\mathbf{D_M} = \{\mathbf{D_i} | i = 1, 2, ..., Nc\} \tag{3.12}$$

$$\mathbf{D_M} = \{cw_k | k = 1, ..., (S_{ID} \cdot Nc)\} \tag{3.13}$$

where each $\mathbf{D_i}$ is the individual dictionary of class i, built by retaining S_{ID} key features (codewords) out of $\mathbf{F_i}$ by the similar clustering procedure as mentioned in Section 3.3.3.1. The size of the overall dictionary $\mathbf{D_M}$, and hence the number of cw_k's, is then $S_D = S_{ID} \cdot Nc$. See Figure 3.15b for an overview of the MD scheme. The BoF-VMMR approaches based on the MD scheme are referred to as BoF-MD.

3.3.3.4 Size of the Dictionary

The dictionary size (S_D) is an important parameter that affects processing speed, *discriminative capacity* and *generalizability* of the built dictionary, and hence affects the overall performance of the BoF-VMMR approaches. A small dictionary could suffer due to reduced discriminatory capacity. In small dictionaries, more than one feature could get assigned to the same cluster, despite being different. On the other hand, a large dictionary loses capacity for generalization, adds higher penalties to noises, and increases processing overhead [105]. As a contribution of this work, we study the effect of various dictionary sizes (for both SD and MD schemes) on overall VMMR speed and accuracy (further described in Section 5.3).

3.3.4 BoF Global Features Representation

The second step of the proposed BoF-VMMR approaches (as shown in Figure 3.14) uses the dictionary $\mathbf{D}$ or $\mathbf{D_M}$ (constructed in *Offline Dictionary Building* step) to embed given images'

local features into global BoF representations through *Features Quantization*. For a given image I_j of class i, its BoF representation is a normalized histogram H_{ij}, of votes (by its local features) to the dictionary codewords. The histogram H_{ij} can be represented by:

$$H_{ij} = [h_1, h_2, ..., h_{S_D}] \tag{3.14}$$

where the bins (h_k's) hold the number of votes to the respective codewords (cw_k's), respectively ($k = 1, ..., S_D$). Every match of a feature to the closest dictionary codeword adds a unit vote.

To build the BoF features representation (histogram) of image I_j, each local feature $f_p \in F_{I_j}$ from I_j is matched to its nearest codeword cw_k of the dictionary ($\mathbf{D}$ or $\mathbf{D_M}$), and the corresponding histogram bin h_k's vote-count is incremented. This step is also referred to as *Features Quantization*. After all features in F_{I_j} are quantized, the histogram undergoes a normalization procedure. In this manner, we obtain the final histogram after matching all features of a given image, and we call it a BoF histogram or feature. Algorithm 2 outlines the procedure to generate BoF-based global representation histogram for a given image. The BoF histogram for the set of local features F_j of a given image I_j of class i is computed as follows:

$$H_{ij}(k) = \frac{1}{Nf_j} \sum_{p=1}^{Np_j} \begin{cases} 1 & \text{if } k = \underset{t \in [1, S_D]}{\operatorname{argmin}} \, dist(cw_t, f_p) \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where $dist(a, b)$ is the euclidean distance between features a and b, and $H_{ij}(k) = h_k$; f_p is the p^{th} local feature and Nf_j is the total number of local features extracted from the image I_j.

Algorithm 1 Algorithm for *BuildDictionary*

Input: The sets of images from all classes, $\mathbf{M_I} = \{\mathbf{I_1}, \mathbf{I_2}, ..., \mathbf{I_{Nc}}\}$, Dictionary sizes for SD (S_{SD}), MD (S_{MD}) and its sub-dictionaries (S_{ID})
Output: $\mathbf{D}$ or $\mathbf{D_M}$
Initialize: $\mathbf{M_F}$
Step 1: Collecting Local Features
for each class $i \in [1, Nc]$: **do**
    Initialize $\mathbf{F_i} = \{\}$
    for each image $j \in [1, N_i]$: **do**
        $F_{ij} \leftarrow FeatureExtraction(I_{ij})$ ▷ Equation 3.8
        $\mathbf{F_i} = \mathbf{F_i} \cup \{F_{ij}\}$ ▷ F_{ij} is set of local features in j
    end for
$\mathbf{M_F} = \mathbf{M_F} \cup \mathbf{F_i}$
end for

Step 2: Dictionary Building
Initialise:
Single-Dictionary $\mathbf{D}$,
Modular-Dictionary $\mathbf{D_M}$
Sub-dictionaries ($\mathbf{D_i}$'s) of $\mathbf{D_M}$
Build SD:
$\mathbf{D} \leftarrow Cluster(\mathbf{M_F}, S_{SD})$

Build MD:
for each class i: **do**
    $\mathbf{D_i} \leftarrow Cluster(\mathbf{F_i}, S_{ID})$
    $\mathbf{D_M} \leftarrow \mathbf{D_M} \cup \mathbf{D_i}$
end for

Algorithm 2 Algorithm for *GetGlobalRepresentation*

Input: An input image I, The dictionaries ($\mathbf{D}$ and $\mathbf{D_M}$), Mode (= SD or MD), Dictionary sizes for SD (S_{SD}), MD (S_{MD})
Output: The computed BoF Histogram, H_I
Initialize:
$H_I \leftarrow [0, ..., 0]$ where $|H_I| = S_{SD}$ or S_{MD}
Step 1: Features Extraction
$F_I \leftarrow FeatureExtraction(I)$
Step 2: Features Quantization ▷ See Algorithm 3
if Mode = SD **then**
    $H_I \leftarrow FeatureQuatization(F_I, \mathbf{D})$
elseif Mode =MD
    $H_I \leftarrow FeatureQuatization(F_I, \mathbf{D_M})$
end if
Step 2: Histogram Normalization
for each $b \in [1, ..., |H_I|]$ **do**
    $H_I(b) \leftarrow H_I(b)/Nf_I$ ▷ where $Nf_I = |F_I|$
end for
return H_I

Algorithm 3 Algorithm for *FeaturesQuantization* Method

Input: The set $F_I = \{f_p | p = 1, ..., Nf_I\}$ of local feature descriptors of an image I, the respective dictionary ($\mathbf{D}$ or $\mathbf{D_M}$)
Output: The computed BoF Histogram, H_I
Initialize: $H_I \leftarrow [0, ..., 0]$ where $|H_I| = S_{SD}$ or S_{MD}
for each $f_p \in F_I$ **do**
    $k = \text{argmin}_{t \in [1, S_D]}\, dist(cw_t, f_p)$
    $H_I(k) \leftarrow H_I(k) + 1$
end for
return H_I

Chapter 4

Classification: Learning to Identify

The goal of classification step in VMMR is to identify the make and model of a given vehicle image not seen before. The classifiers learn the intra-class similarities and inter-class differences, given sufficient training samples, in order to differentiate between and identify objects of different classes in unseen testing samples. The techniques used could range from simple brute-force matching of the test sample with all or selected training samples, to more sophisticated machine learning based classifiers. Depending on the extracted features and training datasets, i.e., number of classes, training samples per class, feature vector dimensions, etc., a suitable classifier needs to be chosen. In this chapter, we first describe and discuss the most popular classification techniques used in VMMR literature. Then, we present our proposed classification approaches for real-time VMMR.

4.1 Existing Classification Techniques

In this section, we provide a brief description of classification techniques used in state-of-the-art works on VMMR, discussing their pros and cons. Here, we present Support Vector Machines, Tree-based classifiers such as Random Forests, and Sparse Representation-based matching.

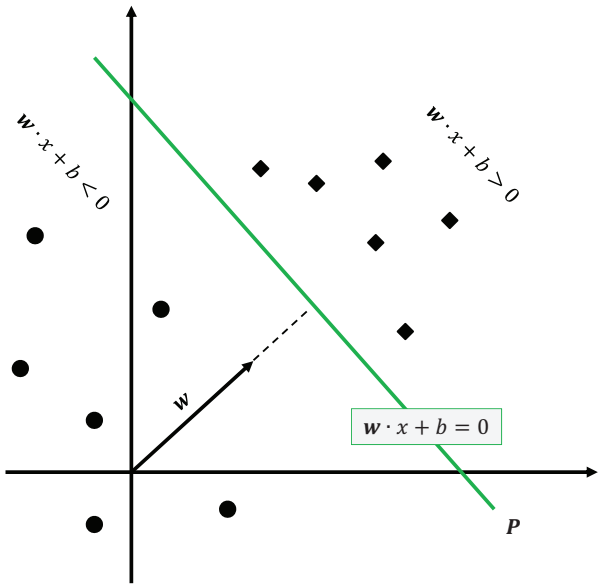


Figure 4.1: Illustrating the hyperplane P separating 2D data points of two classes. The circles correspond to data points of class label -1, whereas the diamonds correspond to those of label $+1$.

4.1.1 Support Vector Machines

The Support Vector Machine [106][107] is a very effective binary classifier in which the support vectors (SVs) are a subset of the training data samples that represent the best separation between two classes. A test data sample is then classified based on its distance from these support vectors. An ensemble of many such binary classifiers are used to build a multi-class SVM classifier.

4.1.1.1 Binary SVM Classifiers

Consider a training dataset of N linearly separable, d-dimensional samples $(\mathbf{x_i}, \mathbf{y_i})$ where $\mathbf{x_i} \in \mathbb{R}^d$ and the labels $y_i \in \{-1, +1\}$. A hyperplane P that can separate the data points of the two classes is defined as $\mathbf{w} \cdot \mathbf{x} + b = 0$, where the vector $\mathbf{w}$ is normal to P, b is bias, and the perpendicular distance from origin to P is $\frac{|b|}{\|\mathbf{w}\|}$. See Figure 4.1 for an illustration of this data separation by P.

All data points of labels $+1$ and -1 that lie at shortest distances d_+ and d_- to P respectively, are called the *Support Vectors* (SVs). The margin surrounding P is then of width $d_+ + d_-$. Under the assumption of linear separability of the considered training data, it can be assumed that all

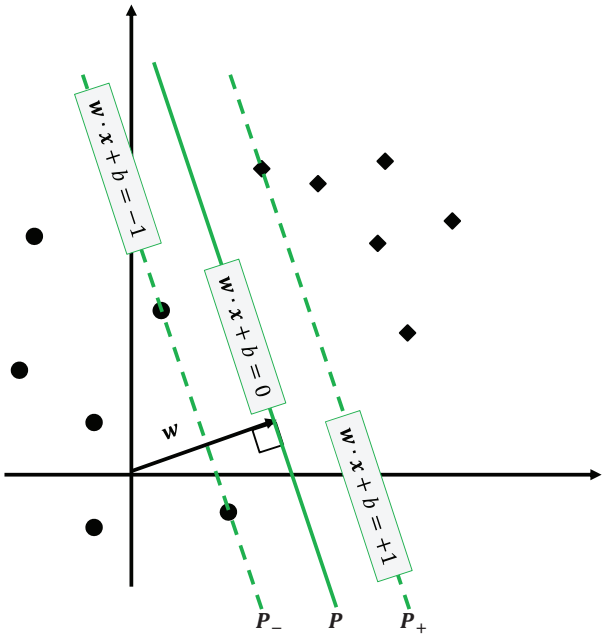


Figure 4.2: Illustrating the hyperplanes P_-, P, and P_+.

the data points (x_i) satisfy the two following inequalities:

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1, \quad y_i = -1 \tag{4.1}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq +1, \quad y_i = +1 \tag{4.2}$$

Combining the inqualities of Equations 4.1 and 4.2, we get:

$$y_i \cdot (\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geq 0, \quad 1 \leq i \leq N \tag{4.3}$$

The hyperplanes P_{+1} and P_{-1} contain the SVs of labels $+1$ and -1 respectively, with $\mathbf{w}\cdot\mathbf{x}+b = \pm 1$ (See Figure 4.2). The perpendicular distances from the origin to P_{+1} and P_{-1} are then $\frac{|-1-b|}{\|\mathbf{w}\|}$ and $\frac{|1-b|}{\|\mathbf{w}\|}$ respectively, which implies that the margin between P_{+1} and P_{-1} is $\frac{2}{\|\mathbf{w}\|}$

The Lagrangian Formulation

To optimize the classification performance, the hyperplanes P_{+1} and P_{-1} are chosen to maximize the margin width $\frac{2}{\|\mathbf{w}\|}$, which is equivalent to minimizing $\frac{\|\mathbf{w}\|^2}{2}$, achieved through quadratic programming.

The minimization problem can be defined as a Lagrangian formulation L:

$$L = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1] \tag{4.4}$$

where $\alpha_i \geq 0$. Such formulation replaces the inequality in Equation 4.3 by the easier to handle Lagrangian multipliers α_i.

L is then minimized with respect to $\mathbf{w}$ and b, with the following Karush-Kuhn-Tucker (KKT) conditions, to find the optimal solution.

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i} = 0, \quad \forall i \tag{4.5}$$

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \forall i \tag{4.6}$$

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geq 0, \quad \forall i \tag{4.7}$$

$$\alpha_i \geq 0, \quad \forall i \tag{4.8}$$

$$\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1) \geq 0, \quad \forall i \tag{4.9}$$

Substituting the Equations 4.5 and 4.6 into Equation 4.4, we get the dual formulation L_D, which is to be maximized with respect to α_i.

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \tag{4.10}$$

Having found the optimal α_i's from solving Equation 4.10, the optimal $\mathbf{w}$ and b can be then found using:

$$\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x_i} \tag{4.11}$$

$$b = y_k - \mathbf{w} \cdot \mathbf{x_k}, \quad \alpha_k > 0 \tag{4.12}$$

where $\alpha_k > 0$ for the support vectors $\mathbf{x_k}$. The classification function $f(\mathbf{x})$ for a new test sample $\mathbf{x_t}$ is defined as:

$$f(\mathbf{x_t}) = sign(\mathbf{w} \cdot \mathbf{x_t} + b) = sign\left(\sum_{k=1}^{m} y_k \alpha_k (\mathbf{x_k} \cdot \mathbf{x_t}) + b\right) \tag{4.13}$$

In this way, to determine the label of an unknown data sample vector $\mathbf{x_t}$, we need only the support vectors, $\mathbf{x_1}, ..., \mathbf{x_m}$

Linearly Inseparable Data

The training data may not be linearly separable due to many reasons such as white noise, in accurate measurements or erroneous labelling. So, the constraints are relaxed using a slack variable $\xi \geq 0, \ 1 \leq i \leq N$, to tolerate the samples within the separating margin. The constraints in Equation 4.3 are relaxed to:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 \geq -\xi_i \tag{4.14}$$

which translates into:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i, \quad y_i = +1 \tag{4.15}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i, \quad y_i = -1 \tag{4.16}$$

A data point $\mathbf{x_i}$ on the wrong side of P is considered an error, when ξ_i exceeds 1. The upper bound for the training errors is $\sum_{\forall i} \xi_i$. Figure 4.3 illustrates the use of slack variables.

The primal (and dual) optimization problems are then modified to accommodate the noisy data points, in two different ways, yielding two different SVM formulations: (1) C-Support Vector Classification [108, 109], and (2) ν-Support Vector Classification [110], described in Sections 4.1.1.2 and 4.1.1.3 respectively.

As a contribution of this work, we compare performance of BoF-VMMR with C-SVC and ν-SVC in Section 6.1, to find the most optimal multi-class SVM for VMMR purposes.

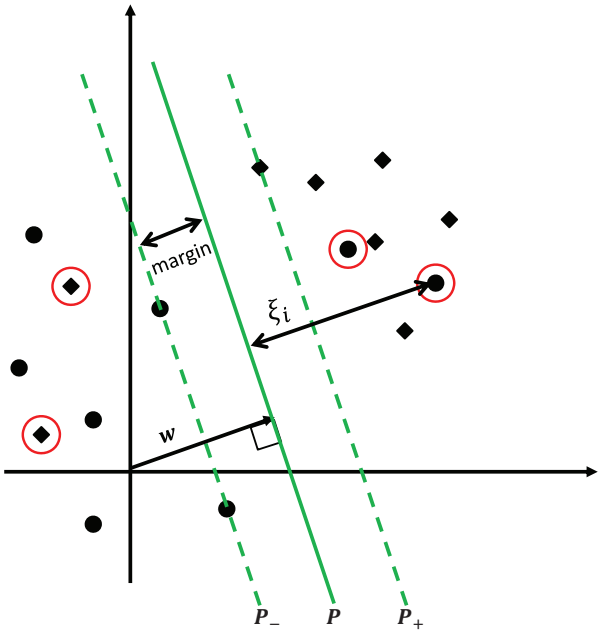


Figure 4.3: Illustrating the use of slack variables, in case of linearly inseparable data

Nonlinear Support vector Machines

The linear data classification can be generalized to find a separator for linearly inseparable data. This is done by mapping the data to a different Hilbert space $\mathscr{H}$ using a nonlinear function Φ. The function Φ basically maps the training data points onto a higher dimensional feature space, in which the separating hyperplane is then learned. The linear classification of the data points in $\mathscr{H}$ is equivalent to their non-linear classification in $\mathbb{R}^d$. Figure 4.4 gives an example of linearly inseparable data in $\mathbb{R}^2$, mapped onto $\mathbb{R}^3$ space where it becomes linearly separable.

The dot product of the data points $\mathbf{x_i} \cdot \mathbf{x_j}$ used in training is replaced by $\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j})$. However, this makes it difficult to handle due to the high dimension of $\mathscr{H}$. To avoid the explicit computation of Φ, the adjusted dot product is replaced by a kernel function K.

$$K(\mathbf{x_i}, \mathbf{x_j}) = \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}) \tag{4.17}$$

The kernel function K is symmetric:

$$K(\mathbf{x_i}, \mathbf{x_j}) = K(\mathbf{x_j}, \mathbf{x_i}) \tag{4.18}$$

and satisfies the Mercer criteria

$$\int K(\mathbf{x_i}, \mathbf{x_j}) g(\mathbf{x_i}) g(\mathbf{x_j}) d\mathbf{x_i} d\mathbf{x_j} \geq 0 \tag{4.19}$$

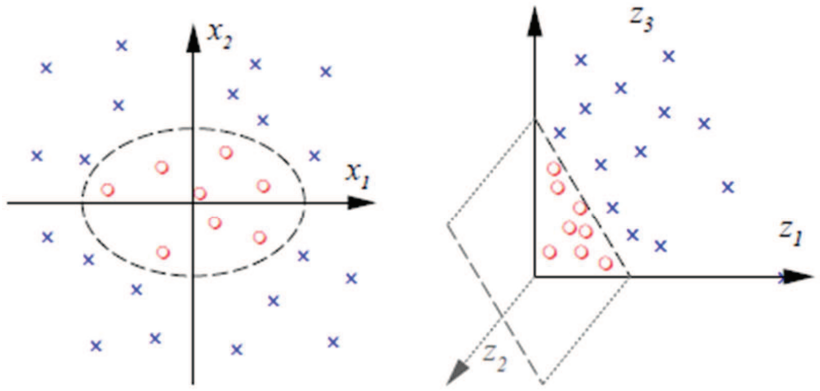


Figure 4.4: The linearly inseparable data points in $\mathbb{R}^2$ (Left), mapped onto $\mathbb{R}^3$ (Right), in which the separating hyperplane can be learned. Here, $(x_1, x_2) \mapsto (z_1, z_2, z_3)$, where $z_1 = x_1^2$, $z_2 = \sqrt{2}x_1x_2$, $z_3 = x_2^2$. (reprinted from [111])

for any function g, having a finite value of $\int g(\mathbf{x})^2 d\mathbf{x}$. This criteria ensures that the kernel matrix $K(i, j)$ is positive semi-definite. For more details on the characteristics of kernel functions and their construction, readers are referred to [112]. Here, we give a few examples of frequently used kernels:

- **Linear Kernel**

$$K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j} \tag{4.20}$$

- **Quadratic Kernel**

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i} \cdot \mathbf{x_j} + r)^2, \ \gamma \in \mathbb{R}^+, r \in \mathbb{R} \tag{4.21}$$

- **Polynomial Kernel**

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i} \cdot \mathbf{x_j} + r)^d, \ \gamma \in \mathbb{R}^+, r \in \mathbb{R}, d \in \mathbb{N} \tag{4.22}$$

- **Radial Basis Kernel**

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma ||\mathbf{x_i} - \mathbf{x_j}||^2} \tag{4.23}$$

For classification, the function defined in Equation 4.13 now becomes:

$$
\begin{aligned}
f(\mathbf{x}) &= sign(\mathbf{w} \cdot \mathbf{x} + b) \\
&= sign\left(\sum_{i=1}^{m} y_i \alpha_i (\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x})) + b\right) \\
&= sign\left(\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + b\right)
\end{aligned}
\tag{4.24}
$$

where $\mathbf{x_i}$ ($i = 1, ...m$) are the SVs.

4.1.1.2 C-Support Vector Classification

In the C-Support Vector Classification (C-SVC) formulation of SVM [108, 109], the primal optimization problem to be solved is:

$$
\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i \tag{4.25}
$$

with the following constraints:

$$
y_i(\mathbf{w}^T\Phi(\mathbf{x_i}) + b) \geq 1 - \xi_i, \forall i \tag{4.26}
$$

$$
\xi_i \geq 0, \forall i \tag{4.27}
$$

where $C > 0$ is the regularization parameter. In practise, to avoid the high dimensionality of the vector variable $\mathbf{w}$, the following dual problem is solved instead.

$$
\min_{\alpha} \quad \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \tag{4.28}
$$

with the following constraints:

$$
\mathbf{y}^T\boldsymbol{\alpha} = 0 \Rightarrow \sum_{i}^{m} \alpha_i \cdot y_i = 0, \forall i \tag{4.29}
$$

$$
0 \leq \alpha_i \leq C, \forall i \tag{4.30}
$$

where Q is an $N \times N$ positive semi-definite matrix, $Q_{ij} = y_i y_j K(\mathbf{x_i}, \mathbf{x_j})$, $\mathbf{e} = [1, \ldots, 1]^T$ is a vector of all ones, and $K(\mathbf{x_i}, \mathbf{x_j}) = \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$ is the kernel function

By solving the problem in Equation 4.28, subject to the constraints in Equations 4.29 and 4.30, and using the primal-dual relationship, the optimal $\mathbf{w}$ satisfies:

$$\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \Phi(\mathbf{x_i}) \tag{4.31}$$

Then, for classification, the decision function is similar to Equation 4.24:

$$\begin{aligned} f(\mathbf{x}) &= sign(\mathbf{w}^T \Phi(\mathbf{x}) + b) \\ &= sign\left(\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + b \right) \end{aligned} \tag{4.32}$$

where $\mathbf{x_i}$ $(i = 1, ...m)$ are the SVs.

4.1.1.3 ν-Support Vector Classification

In the ν-Support Vector Classification (ν-SVC) formulation of SVM, [110] introduced the parameter $\nu \in (0, 1]$ in the optimization problem, to control the number of SVs. The ν gives a lower bound to the fraction of samples becoming SVs, and an upper bound on the fraction of training errors (i.e., the training samples lying on wrong side of the hyperplane).

The primal optimization problem of 4.25 is modified as:

$$\min_{\mathbf{w}, b, \xi, \rho} \quad \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i \tag{4.33}$$

with the following constraints:

$$y_i(\mathbf{w}^T \Phi(\mathbf{x_i}) + b) \geq \rho - \xi_i, i = (1, \ldots, m) \tag{4.34}$$

$$\xi_i \geq 0, \rho \geq 0 \tag{4.35}$$

The dual problem of which is:

$$\min_{\alpha} \quad \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \tag{4.36}$$

with the following constraints:

$$\mathbf{e}^T \boldsymbol{\alpha} \geq \nu \Rightarrow \sum_{i}^{m} \alpha_i \geq \nu \tag{4.37}$$

$$\mathbf{y}^T \boldsymbol{\alpha} = 0 \Rightarrow \sum_{i=1}^{m} y_i \alpha_i = 0, \tag{4.38}$$

$$0 \leq \alpha_i \leq \frac{1}{m}, \forall i \tag{4.39}$$

where Q and $\mathbf{e}$ are as described previously.

4.1.1.4 Multi-class SVM

A collection of many binary SVM classifiers are used to build a single multi-class SVM classifier. Two popular ways of building the individual binary classifiers are 1-vs-1 or 1-vs-All. In the former approach, a binary SVM learns to differentiate between every two classes. So, there would be $\frac{Nc(Nc-1)}{2}$ binary SVMs in the multi-class SVM, where Nc is the number of classes in the dataset. In the latter approach on the other hand, a binary SVM is learned to differentiate each class against all other $Nc - 1$ classes. So, there would be Nc binary SVMs in total.

The classification outputs of all binary SVMs in the multi-class SVM are combined using different combination rules, such as:

- **Majority Voting**: Each binary SVM adds a vote to a class. The class winning the maximum votes across all the binary SVMs, is the final classification output.
- **Least Square Error (LSE)-weighted Outputs**: Extends the majority voting scheme by weighting the outputs of binary SVMs.
- **Double Layer Hierarchical Combination**: A second-layer classifier evaluates the outputs of first layer classifiers.

4.1.2 Random Forests

The Random Forests classifier is an ensemble of weak random decision trees. Breiman introduced the algorithm for inducing Random Forests in [113]. It is a combination of Breiman's idea

of creating random subsets of training data (known as "bagging") and the application of Random Subspace Method to randomly subsample the features to build the decision trees, as introduced by Ho in [114].

The Random Forests have been proven to be successful in multi-class image classification problems. For example, Bosch et al. [115] showed that Random Forests perform superior to SVM on the Caltech-101 [116] and Caltech-256 datasets [117]. Also, Zaklouta and Stanciulescu [118] applied Random Forest to the problem of traffic signs recognition, and showed that it outperforms SVM. Moreover, Zhang [11] has shown applicability of Random Forest in vehicle type recognition systems. Some advantages of using Random Forests include feasibility to be implemented in a parallel or distributed computing environment, and can be used for online learning. To the best of our knowledge, Random Forests have not been explored in the context of VMMR on a real-world dataset. In Section 6.4, we explore Random Forest for VMMR problems, to answer the question if Random Forests work better than SVM in the context of VMMR, using a real-world dataset.

4.1.2.1 Random Forest Construction

A Random Forest R is composed of many Random Trees $T_t \in \{T_1, T_2, ..., T_T\}$. Each random tree T_t is built according to the following procedure:

- From the set of all training data X containing N_{tr} number of samples, a subset X_i is randomly chosen (with replacement). Using this subset, the tree is grown further (without pruning)
- Within a tree, at each node, the algorithm randomly chooses a subset F of features. To split the X_i into X_i^l and X_i^r, the feature $f_j^* \in F$ with the maximum information gain $\triangle$ and a threshold $t \in [\min(f_j^*), \max(f_j^*)]$ are used.

$$X_i^r = \{j \in X_i | f_{jk} > t\}, \quad 1 \leq k \leq |F| \tag{4.40}$$

$$X_i^l = X_i \setminus X_i^r \tag{4.41}$$

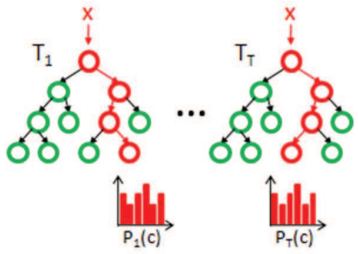


Figure 4.5: Illustrating the classification procedure for a text sample **x** with a Random Forest classifier. The final decision is based on combining posteriors ($P_t(c)$'s) from all T_t's. (reprinted from [118])

The information gain is computed by the following equation, where entropy E is computed based on the class frequencies in X_i:

$$\triangle = -\frac{|X_i^r|}{|X_i|}E(X_i^r) - \frac{|X_i^l|}{|X_i|}E(X_i^l) \tag{4.42}$$

Then, the optimal f_j to split on is found based on the maximum information gain:

$$f_j^* = \max_j \triangle \tag{4.43}$$

The number of features ($|F|$) to select depends on the dataset being used. Having a large F decreases the training speed and could lead to over-fitting issues. On the other hand, a small F increases the training speed and randomization strength. However, if F is too small, it could lead to under-fitting issues. In Section 6.4, we empirically find the optimal size of F in the context of VMMR, based on a realistic dataset.

4.1.2.2 Classification

Given a test sample **x**, each of the random trees T_t of the Random Forest R is traversed. In the lead node reached in T_t, the posterior probability that **x** belongs to the class c (where $c = 1, ..., Nc$) is given by $P_t(c|\mathbf{x})$. The final classification output c^* is based on combining the decisions of all T_t's. Figure 4.5 illustrates the classification procedure.

$$P(c|x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|x) \tag{4.44}$$

$$c^* = \underset{c}{\operatorname{argmax}}\, P(c|x) \tag{4.45}$$

4.1.3 Sparse Representation-based Classification

Sparse Representation (SR) has been a popular technique driving a plethora of works in statistical signal processing community, in which sparse linear representations of signals are computed using an over-complete dictionary of base elements or signal atoms. The main objective of these works has been to represent and compress high-dimensional signals, and not to classify. In the recent years, theories and algorithms for SR have been extended for new problem domains such as image classification or retrieval.

Wright et al. [119] have shown that SRs of images could encode their content or semantic information, and be applied to solve computer vision (CV) and image classification problems. This sparked great interest to the CV community and led to various image classification techniques and frameworks based on SR. The various image classification problems in which SR has been successful include: face recognition [120–122], multi-modal biometrics [123], object recognition [124], action recognition [125], event analysis [126], expression recognition [127], iris recognition [128], etc. Chen et al. [7] have used SR-based classification for VMMR with encouraging accuracies, but with slow processing speeds.

In CV and image classification domain, SR is used as an approach to represent objects of interest using an over-complete dictionary, which is composed of training images or their global feature representations.

4.1.3.1 Dictionary and Sparse Coefficients Learning

Given the training data $\mathbf{x} = [\mathbf{x_1}, \ldots, \mathbf{x_N}] \in \mathbb{R}^{m \times N}$, the goal is to find an optimal dictionary $\mathbf{D} = [\mathbf{d_1}, \ldots, \mathbf{d_K}] \in \mathbb{R}^{m \times K}$, such that an image's global feature representation vector $\mathbf{x_i}$ can be represented as a sparse approximation over it. The $\mathbf{x_i}$ could be approximated as a linear combination of few columns (atoms) from $\mathbf{D}$, i.e., $\mathbf{x_i} = \mathbf{D}\boldsymbol{\alpha}_n$, or, generalizing over the training data:

$$\mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha} \tag{4.46}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N] \in \mathbb{R}^{K \times N}$ is the set of combination coefficients in the sparse decomposition. K-SVD [129] algorithm is used to learn the optimal $\mathbf{D}$ and $\boldsymbol{\alpha}$. This learning process can be formulated as the following joint optimization problem:

$$\underset{\mathbf{D},\boldsymbol{\alpha}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \;\text{ subject to }\; \|\boldsymbol{\alpha}_i\|_0 \leq T, \forall i \tag{4.47}$$

where $\|\boldsymbol{\alpha}_i\|_0$ is the ℓ_0-norm, i.e., a count of non-zero values in $\boldsymbol{\alpha}_i$, T is the maximum number of non-zero coefficients desired.

The sparse solution $\boldsymbol{\alpha}$ can be obtained by solving the optimization problem in Equation 4.48, through a greedy iterative method called Orthogonal Matching Pursuit (OMP), which finds the optimal $\boldsymbol{\alpha}_i$ for each $\mathbf{x_i}$, while fixing $\mathbf{D}$:

$$\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\mathbf{x_i} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \;\text{ subject to }\; \|\boldsymbol{\alpha}_i\|_0 \leq T \tag{4.48}$$

The K-SVD algorithm has two stages: (1) sparse coding, and (2) dictionary update, iterated through a number of times until convergence, to optimise $\mathbf{D}$ and $\boldsymbol{\alpha}$. Within each iteration, each $\boldsymbol{\alpha}_i$ in $\boldsymbol{\alpha}$ is optimized by solving Equation 4.48 in the sparse coding stage while fixing $\mathbf{D}$, and in the dictionary update stage, the column vectors (atoms) in $\mathbf{D}$ are sequentially updated so that their coefficients improve the representation of $\mathbf{x}$.

4.1.3.2 SR-based Classification

Given the optimal $\mathbf{D}$ obtained through K-SVD, the SR-based classification (SRC) scheme of an input vector $\mathbf{x_t} \in \mathbb{R}^m$ can be described as in Algorithm 4. The $\mathbf{x_t}$ is approximated as a linear combination of atoms of $\mathbf{D}$:

$$\mathbf{x_t} = \mathbf{D}\boldsymbol{\alpha}_t = \alpha_1 \mathbf{d_1} + \cdots + \cdots + \alpha_k \mathbf{d_K} \tag{4.49}$$

where $\alpha_k \in \boldsymbol{\alpha}_t$ and $\mathbf{d_k} \in \mathbf{D}$. The sparse solution $\boldsymbol{\alpha}_t$ is efficiently found by solving the following optimization problem through the second-order cone programming [29]:

$$\underset{\boldsymbol{\alpha}_t}{\operatorname{argmin}} \|\boldsymbol{\alpha}_t\|_1 \;\text{ subject to }\; \|\mathbf{x_t} - \mathbf{D}\boldsymbol{\alpha}_t\|_2^2 \leq \varepsilon \tag{4.50}$$

where ε is the reconstruction error tolerance. Its worth mentioning here that a major drawback with the SRC scheme is the time-consuming optimization process to find $\boldsymbol{\alpha}_t$.

The dictionary $\mathbf{D}$ is separated into Nc classes: $\mathbf{D} = [\mathbf{D_1}, \ldots, \mathbf{D_{Nc}}]$. An input sample $\mathbf{x_t}$ is approximated as $\mathbf{D}\delta_i(\boldsymbol{\alpha}_t)$, where the function $\delta_i : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is to select the set of coefficients corresponding to class i. Then, the residual of x_t with respect to class i, i.e., $r_i(\mathbf{x_t})$ is computed as:

$$r_i(\mathbf{x_t}) = \|\mathbf{x_t} - \mathbf{D}\delta_i(\boldsymbol{\alpha}_t)\|_2 \tag{4.51}$$

The classification function is then expressed as below, where the output is the class to which $\mathbf{x_t}$ has the minimal residual:

$$f(\mathbf{x_t}) = \underset{i}{\operatorname{argmin}}\, r_i(\mathbf{x_t}) \tag{4.52}$$

Algorithm 4 Algorithm for *Sparse Representation-based Classification (SRC)* Scheme

1: **Input:** The set optimal $\mathbf{D} \in \mathbb{R}^{m \times K}$, a test feature vector $\mathbf{x_t} \in \mathbb{R}^m$
2: **Output:** The predicted class label of $\mathbf{x_t}$
3: Finding the sparse solution $\boldsymbol{\alpha}_t$, by solving Equation 4.50
4: Computing the residuals (Equation 4.51)
5: **return** $f(\mathbf{x_t}) = \operatorname{argmin}_i r_i(\mathbf{x_t})$ (Equation 4.52)

4.1.3.3 Hamming Distance-based Classification

Chen et al. [7] proposed a Hamming Distance-based Classification scheme for VMMR, based on the sparse representation basis $\mathbf{D}$, to circumvent the slow processing speed of SRC scheme. The underlying assumption is that if a test sample $\mathbf{x_t}$ belongs to class i, then the linear combination coefficients will have most of the non-zero values for entries associated to class i. Greater the sparsity of this recovered $\boldsymbol{\alpha}_t$, higher the feasibility to accurately classify $\mathbf{x_t}$.

Given $\mathbf{D} = [\mathbf{D_1}, \ldots, \mathbf{D_{Nc}}]$, the maximum pair-wise distance between samples in each $\mathbf{D_i}$ is found using a Euclidean distance metric. If $\mathbf{d_{ij}}$ is the j^{th} column vector in $\mathbf{D_i}$ of class i, then η_i denotes the maximum distance between any pair of samples in $\mathbf{D_i}$:

$$\eta_i = \max_{1 \leq j,k \leq N_i} dist(\mathbf{d_{i,j}}, \mathbf{d_{i,k}}) \tag{4.53}$$

where $dist(\mathbf{a}, \mathbf{b})$ is the Euclidean distance between vectors $\mathbf{a}$ and $\mathbf{b}$.

The $\mathbf{x_t}$ is encoded into a binary code $B(\mathbf{x_t}) = [B_1(\mathbf{x_t}), \ldots, B_{Nc}(\mathbf{x_t})]$ with respect to the elements in $\mathbf{D}$, where $B_i(\mathbf{x_t}) = [b_{i,1}, \ldots, b_{i,N_i}]$, determined by:

$$b_{i,j}(\mathbf{x_t}) = \begin{cases} 1 & \text{if } dist(\mathbf{x_t}, \mathbf{d_{i,j}}) \leq \eta_i \\ 0 & \text{otherwise} \end{cases} \tag{4.54}$$

Let B_i^o be a binary code representation for elements of class i in $\mathbf{D}$. Each $b_{i,j}^o$ corresponds to the element $\mathbf{d_{i,j}} \in \mathbf{D_i}$. The entries of B_i^o are non-zero (ones) only for $b_{i,j}^o$'s corresponding to $\mathbf{d_{i,j}}$'s, and zero otherwise: $B_i^o = [0_{1,1}, 0_{1,2}, \ldots, 0_{1,N_1}, \ldots 0_{i-1,N_{i-1}}, 1_{i,1}, 1_{i,2}, \ldots, 1_{i,N_i}, 0_{i+1,1}, \ldots, 0_{Nc,N}]$, where $N = \sum_i^{Nc} N_i$ is the total number of elements in $\mathbf{D}$ for all Nc classes.

The classification function for $\mathbf{x_t}$ is:

$$f(\mathbf{x_t}) = \underset{1 \leq i \leq Nc}{\operatorname{argmin}} \xi_i(\mathbf{x_t}) \tag{4.55}$$

where $\xi_i(\mathbf{x_t}) = HamDist(B(\mathbf{x_t}), B_i^o)$, which is the Hamming distance between binary vectors $B(\mathbf{x_t})$ and B_i^o, i.e., the distance between $\mathbf{x_t}$ from the i^{th} class.

To account for unknown classes, a threshold based rejection option such as that in [130] is employed. If the $\xi_i(\mathbf{x_t}) \geq \theta_i$, it is rejected (not assigned to any known class). Here, θ_i is the maximum allowable distance to class i, for any sample to belong to class i, determined by $\theta_i = \mu_i + \tau\sigma_i$, where σ_i is the variance of $\xi_i(\mathbf{x})$ and τ sets its confidence interval. The decision function in Equation 4.55 now becomes:

$$f(\mathbf{x_t}) = \underset{1 \leq i \leq Nc}{\operatorname{argmin}} \xi_i(\mathbf{x_t}) \quad \text{subject to} \quad \xi_i(\mathbf{x_t}) \leq \theta_i \tag{4.56}$$

The major bottleneck in SRC scheme is the slow speed of generating the sparse representation of an input vector (a processing speed of 0.31 fps was reported in [7]). This high time-cost makes SRC inapplicable to real-time scenarios. On the other hand, in HDC, comparison with all training samples to generate the binary codes of a test sample is very time consuming, which could degrade the VMMR system's processing speed. As the number of classes or training images increases, the processing speed of HDC would decrease.

4.1.3.4 SRC+HDC Scheme

To improve on the processing speed of SRC, Chen et al. [7] proposed to integrate the HDC scheme into SRC scheme. It was shown that SRC+HDC scheme yielded the best accuracies

and was slightly faster (at 0.46 fps compared to 0.31 fps in SRC). In SRC+HDC scheme, the classification function in Equation 4.52 is modified to:

$$f(\mathbf{x_t}) = \underset{i}{\operatorname{argmin}}\, r_i(\mathbf{x_t}) + \frac{\xi_i(\mathbf{x_t})}{N} \tag{4.57}$$

4.2 Proposed Classification Approaches

Classification constitutes the third step in the proposed BoF-VMMR approaches (See the BoF-VMMR flowchart in Figure 3.14). It includes training a classifier over the BoF representations of all training images, to be used subsequently in VMMR testing. In this work, we propose two multi-class Support Vector Machine (SVM)-based classification schemes for BoF-VMMR. Since the datasets are usually unbalanced, extensive experiments must be conducted to find the optimal SVM parameters for the dataset at hand. The two approaches for multi-class classifier training and testing introduced here for BoF-VMMR are: (A) Single Multi-Class SVM Classifier, and (B) Ensemble of Multi-class SVM Classifiers based on Attribute Bagging.

4.2.1 Single Multi-class SVM Classifier

For each training image of given classes in the training phase, local features such as SURF are extracted and embedded into the BoF histograms using the Single Dictionary (or the Modular Dictionary), as described in Section 3.3.4. These BoF histograms from all training images are collected and used to train the multi-class SVM classifier described in Section 4.1.1. For testing, the BoF histogram of the given test image is generated using the same dictionary used in training. Based on this histogram, each of the binary classifiers that make up the multi-class SVM adds a vote to its predicted class. The class with the highest votes is assigned as the predicted make-model class of the test image. Algorithm 5 and 6 show the pseudocodes of training and testing phases respectively. This classification scheme will be referred to by SSVM.

Algorithm 5 Algorithm for Training Phase of BoF-VMMR with SSVM

Input: The collection of sets of training images from all classes $\mathbf{M_I} = \{\mathbf{I_1}, \mathbf{I_2}, ..., \mathbf{I_{Nc}}\}$, The Single- or Modular-Dictionary ($\mathbf{D}$ or $\mathbf{M_D}$), Mode $\in$ [*'SD', 'MD'*] indicating which dictionary to use
Output: The trained Single-SVM classifier, $\mathbf{C}$
Initialize: The pool of all training BoF histograms $\mathbf{H}$
Step 1: Generating BoF Global Representations of Images ▷ Using Algorithm 2
for each class $i \in [1, \ldots, N_c]$ **do**
    Initialize $\mathbf{H_i} = \{\}$ ▷ $\mathbf{H_i}$ is set of class i's training BoF histograms
    for each image $j \in [1, \ldots, N_i]$ **do**
        $F_{ij} \leftarrow FeaturesExtraction(I_{ij})$
        if Mode == *'SD'* **then**
            $H_{ij} \leftarrow GetGlobalRepresentation(F_{ij}, \text{'SD'})$
        elseif Mode == *'MD'*
            $H_{ij} \leftarrow GetGlobalRepresentation(F_{ij}, \text{'MD'})$
        end if
        $\mathbf{H_i} \leftarrow \mathbf{H_i} \cup (H_{ij}, y_i)$ ▷ y_i is the label of class i
    end for
    $\mathbf{H} \leftarrow \mathbf{H} \cup \mathbf{H_i}$
end for

Step 2: Training SVM Classifier
$C \leftarrow SVM.Train(\mathbf{H})$

Algorithm 6 Algorithm for Testing Phase of BoF-VMMR with SSVM

Input: A test image I_t, its ground truth class-label y_t^o, Mode $\in$ [*'SD', 'MD'*] indicating which dictionary to use
Output: y_t^p (the predicted class label of I_t)
Initialize: $\mathbf{H_t}$, the BoF histogram of I_t
Step 1: Generating BoF Global Representation ▷ Using Algorithm 2
if Mode == *'SD'* **then**
    $H_t \leftarrow GetGlobalRepresentation(F_t, \text{'SD'})$
elseif Mode == *'MD'*
    $H_t \leftarrow GetGlobalRepresentation(F_t, \text{'MD'})$
end if
Step 2: Prediction
$y_t^p \leftarrow C.Predict(H_t)$

4.2.2 Ensemble of Multi-class SVM Classifiers based on Attribute Bagging

In this section, we describe the ensemble of multi-class SVM classifiers approach we propose for BoF-VMMR. Instead of training a single classifier over the entire set of feature vectors, we explore the idea of building an ensemble of individual multi-class classifiers that are trained over different random feature subspaces (i.e., random feature subsets). This concept is referred to by different names in the literature: *Attribute Bagging (AB)*, *Multiple Feature Subsets*, and *Random Subspace Method*. The Random Subspace Method is a more generic term which could refer to: (1) applying Random Subsampling over the training data samples to create bootstrap subsets of the training dataset, or, (2) applying Random Subsampling over the feature-space to create random subsets of feature-vectors (used in this work). We prefer to use the term *Attribute Bagging*, as it best describes the technique with which feature subsets are created. In testing, the predictions from each of the classifiers in the ensemble are combined using a certain combination rule to produce the final prediction.

4.2.2.1 Motivation to use Attribute Bagging

The motivation to adopt Attribute Bagging (AB) for training the individual classifiers of the ensemble arises from the following observation. In the MMR dataset used in this work, the training samples per class are too few in number when compared to our feature vector dimensions, which could lead to over-fitting problems for classifiers such as SVM. For example, while the average number of training samples per class is 182 (See Table 5.1) in the 80-20 Dataset versions we make from NTOU-MMR Dataset, the best performing feature vector length is 2000. Remember that the feature vector length is equal to the size of the dictionary used to generate the feature, and that each attribute of the feature vector corresponds to the votes assigned to the respective dictionary codeword. Employing an ensemble of classifiers built using AB helps to avoid over-fitting problems.

To avoid the over-fitting issue by reducing the difference between the number of training data samples and the feature vector dimensions, random subsets of feature vectors are created in a similar fashion to [131–133]. Unlike them, we do not create random subsets of the training

dataset, but we create random subsets out of training feature vectors. Moreover, several works in the literature (e.g., [97]) have shown that the AB-based ensemble of classifiers could perform better than the stand-alone individual classifiers (i.e. classifiers trained over whole feature vectors). Motivated by their findings, we investigate if the AB-based ensemble of multi-class SVM classifiers, hereby referred to as AB-SVM, could improve the performance of VMMR in comparison to the single classifier scheme of Section 4.2.1. Table 4.1 summarizes the symbols used in regards to AB-SVM.

Table 4.1: Nomenclature (for symbols used in Section 4.2.2)

Symbol	Meaning
N_{ss}	# of Random Feature Subspaces, # of Classifiers in Ensemble
S_{ss}	Size of (#Attributes in) a Random Feature Subspace
N_i	# of training images of class-i
N_{tr}	$= \sum_{i=1}^{N_c} N_i$ (i.e., total number of training images)
S_D	Size of SD or MD, #Codewords in dictionary
A_g	Set of randomly chosen attribute indices for g^{th} subspace
H_{ij}	BoF feature vector of image-j of class-i
H_{ij}^g	g^{th} feature subset of H_{ij}, extracted based on A_g
$\mathbf{H_i^g}$	Training set of class-i from g^{th} feature subspace (containing all H_{ij}^g's)
$\mathbf{H^g}$	Training set of all classes from g^{th} feature subspace (containing all $\mathbf{H_i^g}$'s)
C^g	a multi-class SVM classifier trained over $\mathbf{H^g}$
C	the ensemble of C^g's

4.2.2.2 Creating Random Feature Subsets by AB

We will illustrate the AB method of creating feature subspaces through a simple example. Assume we have BoF feature vectors such as $F = [a_1, \ldots, a_{10}]$, where a_x is the value of the x^{th} attribute. Let the number of random feature subspaces to create be $N_{ss} = 4$, each comprising of $S_{ss} = 5$ attributes (or dimensions). Let A_g denote the set of randomly chosen attribute indices for

the g^{th} feature subspace, as described by Equation 4.58, where $|A| = S_{ss}$, and $g = 1, ..., N_{ss}$:

$$A_g = \{x | x \in [1, S_D]\} \tag{4.58}$$

For example, consider $A_1 = \{9, 6, 2, 5, 8\}$, $A_2 = \{1, 7, 9, 3, 4\}$, $A_3 = \{8, 1, 6, 10, 3\}$, and $A_4 = \{5, 1, 2, 6, 9\}$. So, out of each original feature vector F, we would extract $N_{ss} = 4$ random feature subsets based on A_1,A_2,A_3 and A_4 respectively, resulting in the following random feature subsets out of F: $F_1 = [a_9, a_6, a_2, a_5, a_8]$, $F_2 = [a_1, a_7, a_9, a_3, a_4]$, $F_3 = [a_8, a_1, a_6, a_{10}, a_3]$, and $F_4 = [a_5, a_1, a_2, a_6, a_9]$. We would then build an ensemble of $N_{ss} = 4$ classifiers, each trained over the respective feature subspace.

Now, let us generalize the application of AB over our training dataset. Let H_{ij} be the BoF feature vector for a given image j of class i, comprised of S_D attributes:

$$H_{ij} = [h_{j,1}, \ldots, h_{j,K}] \quad where\ K = S_D \tag{4.59}$$

To create each feature subset H^g_{ij}, we must randomly select $S_{ss} < S_D$ different attributes (without replacement) from H_{ij}, based on the attribute indices in A_g. Sampling without replacement ensures that within a subset, each attribute is selected only once. However, an attribute could be chosen in more than one subset. All such feature subsets of class i (i.e. H^g_{ij}'s), are collected in $\mathbf{H^g_i}$ as shown in Equation 4.60 where the dimensions of $\mathbf{H^g_i}$ are $N_i \times S_{ss}$.

$$\mathbf{H^g_i} = [H^g_{i1}, H^g_{i2}, ..., H^g_{iN_i}]^T \tag{4.60}$$

The classwise pools of feature subsets ($\mathbf{H^g_i}$'s) for all classes $i = 1, ..., Nc$ are then collected in the respective overall training set for the g^{th} feature subspace ($\mathbf{H^g}$) as shown in Equation 4.61:

$$\mathbf{H^g} = [\mathbf{H^g_1}, \mathbf{H^g_2}, ..., \mathbf{H^g_{Nc}}]^T \tag{4.61}$$

where each $\mathbf{H^g}$ is of dimensionality $N_{tr} \times S_{ss}$.

4.2.2.3 Classification

The feature subsets in $\mathbf{H^g}$ are used to train the corresponding multi-class classifier C^g. In this way, we achieve an ensemble of classifiers C (composed of C^g's), to predict the vehicle's make

and model in given test images. A greater value for N_{ss} will yield a larger number of feature subspaces, which would increase the chances of having qualitatively different C^g's, as discussed by [131].

In the testing phase, the input image's BoF feature vector H_t is sub-sampled into subsets H_t^g based on the respective set of attribute indices A_g. To create each subset, the same sequence of S_{ss} attributes indices (given by A_g) that were selected in creating $\mathbf{H^g}$ are used. The classifier C^g is then used to predict the label of H_t^g, adding a vote to the winning class. The class that wins majority of the votes amongst all C^g's ($g = 1, \ldots, N_{ss}$), is produced as the predicted make and model of the test image. Algorithms 7 and 8 provide the pseudocode for training and testing phases of BoF-VMMR with ABSVM.

Algorithm 7 Algorithm for Training Phase of BoF-VMMR with ABSVM

Input: The collection of sets of training images from all classes $\mathbf{M_I} = \{\mathbf{I_1}, \mathbf{I_2}, ..., \mathbf{I_{Nc}}\}$, The Single- or Modular-Dictionary ($\mathbf{D}$ or $\mathbf{M_D}$), Mode $\in$ [*'SD','MD'*] indicating which dictionary to use, The collection $\mathbf{A} = \{A_g | g = 1, \ldots, N_{ss}\}$, where A_g is the set of selected attribute indices for g^{th} feature subspace.
Output: The trained ABSVM classifier, $\mathbf{C} = \{C^g | g = 1, ..., N_{ss}\}$
Initialize:
$\mathbf{H_i^g}, \forall g \in [1, \ldots, N_{ss}], \forall i \in [1, \ldots, Nc]$
$\mathbf{H^g}, \forall g \in [1, \ldots, N_{ss}]$
Step 1: Generating BoF Global Representations of Images ▷ Using Algorithm 2
for each class $i \in [1, \ldots, N_c]$ **do**
    for each image $j \in [1, \ldots, N_i]$ **do**
        Initialise: H_{ij}, and $H_{ij}^1, \ldots, H_{ij}^{N_{ss}}$
        $F_{ij} \leftarrow FeaturesExtraction(I_{ij})$
        if Mode == *'SD'* **then**
            $H_{ij} \leftarrow GetGlobalRepresentation(F_{ij}, 'SD')$
        elseif Mode == *'MD'*
            $H_{ij} \leftarrow GetGlobalRepresentation(F_{ij}, 'MD')$
        end if
        for $g = 1, \ldots, N_{ss}$ **do**
            $H_{ij}^g \leftarrow SubSample(H_{ij}, A^g)$ ▷ select $S_{ss} < S_D$ attributes from H_{ij}, based on A_g
            $\mathbf{H_i^g} \leftarrow \mathbf{H_i^g} \cup (H_{ij}^g, y_i)$ ▷ y_i is the label of class i
        end for
    end for
end for

Step 2: Collecting Training Data
for $g = 1, \ldots, N_{ss}$ **do**
    for $i = 1, \ldots, N_c$ **do**
        $\mathbf{H^g} \leftarrow \mathbf{H^g} \cup \mathbf{H_i^g}$
    end for
end for

Step 3: Training the Ensemble of Classifiers
for $g = 1, \ldots, N_{ss}$ **do**
    $C^g \leftarrow SVM.Train(\mathbf{H^g})$
end for

Algorithm 8 Algorithm for Testing Phase of BoF-VMMR with ABSVM

Input: A test image I_t, its ground truth class-label y_t^o, Mode $\in$ ['*SD*', '*MD*'] indicating which dictionary to use, Mode $\in$ ['*SD*', '*MD*'] indicating which dictionary to use, The collection $\mathbf{A} = \{A_g | g = 1, \ldots, N_{ss}\}$, where A_g is the set of selected attribute indices for g^{th} feature subspace.
Output: y_t^p (the predicted class label of I_t)
Step 1: Generating BoF Global Representations of Images ▷ Using Algorithm 2
Initialise: H_t, and $H_t^1, \ldots, H_t^{N_{ss}}$
$F_t \leftarrow FeaturesExtraction(I_t)$
if Mode == *'SD'* **then**
    $H_t \leftarrow GetGlobalRepresentation(F_t, 'SD')$
elseif Mode == *'MD'*
    $H_t \leftarrow GetGlobalRepresentation(F_t, 'MD')$
end if
for $g = 1, \ldots, N_{ss}$ **do**
    $H_t^g \leftarrow SubSample(H_t, A^g)$
end for

Step 3: Prediction
$P_t^g = \{\}$ ▷ The list of prediction outputs of individual classifiers
for $g = 1, \ldots, N_{ss}$ **do**
    $P_t^g \leftarrow P_t^g \cup C^g.Predict(H_t^g)$
end for
$y_t^p = MajorityVoting(P_t^g)$ ▷ Select the class with maximum votes

Chapter 5

Experimental Setup and Implementation

This thesis proposes and investigates unexplored approaches for real-time automated Vehicle Make and Model Recognition (VMMR) based on BoF and multi-class SVMs. The three main steps in the proposed BoF-VMMR approaches, as depicted in Figure 3.14, are: (1) *Offline Dictionary Building* (to be used for Features Quantization), (2) *BoF Features Representation* and (3) *Classification*, which involves classifier training and testing. In this chapter, we first describe and discuss the target environment and dataset (Section 5.1). In Section 5.2, the performance metrics used to analyse processing speed and accuracy of VMMR approaches are presented. The parameters for the optimized dictionaries and classifiers we propose for BoF-VMMR are discussed in Section 5.3. Finally, in Sections 5.4 and 5.5 respectively, we present how vehicle ROIs are extracted in VMMR works and describe the hardware and software platform we use.

5.1 Target Environment and Dataset Description

To demonstrate the effectiveness of the proposed BoF-based approaches for VMMR, we target public areas that are highly vulnerable to security threats. Such scenarios include entrances or exits of parking facilities at public places such as malls, airports, stadiums, etc. (See Figure 5.1). The video/images sensor (camera) is fixed on the entrance/exit of a given parking facility or cross-border checkpoint. Vehicles may be occluded by pedestrians or other objects. The pro-

posed approaches for VMMR can be easily applied to other scenarios in which the camera is not fixed, e.g., an on-board camera on a mobile surveillance vehicle, etc.

We find that the above characteristics of the target environment are best represented by the NTOU-MMR dataset [10]. It is a very recent and publicly available dataset for vehicle makes and models, with published results of several VMMR works. Hence, it serves as a good benchmark dataset to compare performances of our approach with other works. In what follows, we further describe the dataset and note a few problems therein.

Published in the recent related work of Hsieh et al. [10], the NTOU-MMR dataset was collected under the Vision-based Intelligent Environment (VBIE) project [134] and can be accessed at [135]. Speeds of up to 65 km/h were allowed for the oncoming vehicles. The original dataset is divided into a training and a testing set. There are $2,846$ images for training, and $3,793$ images for testing. The total number of classes is 29.

The motivation to use this dataset in our work stems from the following characteristics of the dataset. The images have vehicles in different viewing angle pans ranging from -20° to 20°, which sufficiently represent real-life scenarios. Moreover, the dataset's images were taken throughout the daytime and night-time, and under weather conditions varying between sunny, cloudy and rainy. In addition, there are also images with vehicles occluded by irrelevant objects (such as pedestrians). As we shall present in Section 6.5, the effectiveness of our approaches is proven even in such challenging scenarios (See Figure 6.6).

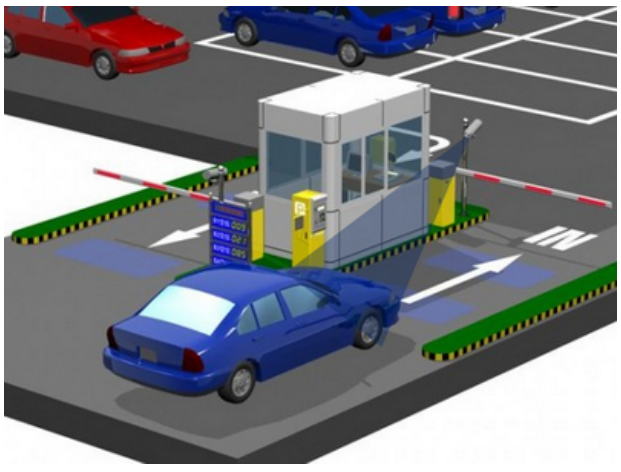
However, we note some problems with the NTOU-MMR dataset (downloaded from [135]):

- *Wrongly placed images:* some class directories have images belonging to other classes.
- *Duplicated images:* many classes have duplicate images (with different names).
- *Biased partitioning of data:* it is unclear which strategy is employed to partition data into training and testing for each class.

The manner in which data is partitioned into training and testing greatly impacts performance results. A biased partitioning can give misleading results of a system's accuracy, as we demonstrate in Figure 6.1 (Section 6.8). Unlike previous works based on the dataset, we repeatedly

(a)

(b)



(c)

Figure 5.1: Examples of the targeted environment where VMMR is needed: (a)-(b) Gates of a cross-border checkpoint, and (c) Entrance or exit of parking spaces in airports or malls, etc. The cameras capture the front faces of vehicles, to be used for VMMR. (Courtesy: Google Images)

randomly partition the original NTOU-MMR dataset to form N_D number of different training and testing splits for each class. For each split, 80% of images are randomly chosen for training, and the remaining 20% for testing. We refer to these as 80-20 NTOU-MMR Datasets, or simply 80-20 Datasets. Table 5.1 outlines the number of training (#Tr) and testing (#Te) images in each of the N_D datasets. The mean accuracies and processing speeds of our approaches are determined by averaging the results over the N_D datasets. The 80-20 ratio for the training and testing split is one of the standard dataset partitioning schemes employed by many works in object recognition and image classification.

5.2 Performance Metrics

5.2.1 Speed

In order to be used in real-life scenarios, a good VMMR system needs to meet real-time processing speed requirements, apart from being accurate. For the processing speed of a VMMR approach, we take the inverse of average time taken per image (in seconds) in extracting features, building the global features representation, and classifying it to predict the MM class label. We report the processing speed of the VMMR approaches in frames-per-second (fps).

5.2.2 Accuracy

We have $k = 1, 2, ..., N_D$ different datasets, where each dataset is a random 80-20 training-testing split of the original NTOU-MMR Dataset [10]. Let $L = \{l_i | i = 1, 2, ..., Nc\}$ be the set of labels for all Nc number of classes in a dataset. The accuracies of the VMMR approaches can be represented by the following metrics:

- **Average Correct Classification Rates** ($ACCR_{l_i}$): the classwise accuracies, based on the ratio of the number of correctly classified images of l_i to the total number of test images for l_i, averaged over N_D dataset splits.

$$ACCR_{l_i} = \frac{1}{N_D} \sum_{k=1}^{N_D} \frac{\text{\#Correctly Classified Images of } l_i \text{ in dataset } k}{\text{Total \# Images of } l_i \text{ in dataset } k} \tag{5.1}$$

Table 5.1: Vehicle Make-Model Classes and #Images in each 80-20 random partition of the NTOU-MMR Dataset [135]

Label	Make	Model	#Train	#Test
1	Toyota	Altis	972	240
2		Camry	556	136
3		Vios	547	136
4		Wish	240	58
5		Yaris	244	60
6		Previa	50	12
7		Innova	36	9
8		Surf	67	16
9		Tercel	145	36
10		RAV4	127	31
11	Honda	CRV	496	123
12		Civic	265	66
13		FIT	84	20
14	Nissan	March	157	38
15		Livna	203	50
16		Teana	76	19
17		Sentra	77	18
18		Cefiro	123	30
19		Xtrail	111	27
20		Tiida	208	51
21	Mitsubishi	Zinger	26	6
22		Outlander	36	8
23		Savrin	43	10
24		Lancer	64	16
25	Suzuki	Solio	101	25
26	Ford	Liata	16	3
27		Escape	90	22
28		Mondeo	91	20
29		Tierra	52	13
Overall			5303	1299

- **Mean Average Correct Classification Rate** (*mACCR*): the overall VMMR accuracy, a metric similar to [136], which is the ratio of the total number of correctly classified images (of all classes) to the total number of test images in the dataset, averaged over N_D dataset splits.

$$mACCR = \frac{1}{Nc} \cdot \frac{1}{N_D} \sum_{i=1}^{Nc} \left(\sum_{k=1}^{N_D} \#\text{Correctly Classified Images of } l_i \text{ in dataset } k \right) \tag{5.2}$$

5.2.3 Discriminative Capability

To visualize the discriminative capabilities of VMMR approaches, the confusion matrix serves as a good tool. While the row indices of the matrix correspond to Ground Truth class labels, the column indices correspond to Predicted class labels.

The value at r^{th} row and c^{th} column, i.e. at (r, c), where $r = 1, ..., Nc$ and $c = 1, ..., Nc$, represents the percentage of class-r's images that are predicted to be of class-c by the VMMR approach. The main diagonal values represent the $ACCR_{l_r}$. In other words, at each (r, r), the value is the $ACCR_{l_r}$ for class-r. The confusion matrix helps us identify the classes which could be apparently similar (in the feature space) and could be leading to inaccurate predictions.

5.3 Optimal Parameters Selection

We obtain the optimal parameters for each step of our BoF-VMMR approaches by cross-validation using the N_D different 80-20 Datasets First, we find optimal dictionary parameters for both SD and MD schemes. Then, we find the optimal classifier parameters for both single classifier and ensemble classifier schemes. The best trade-off between processing speed and accuracy is considered in each case, to achieve the requirements of a real-time VMMR system. In Table 5.2, we summarize the optimal set of parameters used in this work.

Table 5.2: Optimal Parameters for BoF-VMMR

Method	Dictionary		Classifier	
	Type	Size (S_D)	SSVM	ABSVM
BoF-SD	SD	2000	$C = 50, \gamma = 5$	$N_{ss} = 15, S_{ss} = 500$
BoF-MD	MD	$29 \cdot 100$	$C = 50, \gamma = 5$	$N_{ss} = 15, S_{ss} = 1500$

5.3.1 Optimized Dictionaries

To obtain the optimal SD and MD, we run extensive experiments on the BoF-VMMR approach by employing SURF in local Features Extraction step and Single Multiclass SVM in Classification step. The significant parameter affecting the processing speed and accuracy of the overall BoF-VMMR system is the Dictionary Size, S_D. In the Offline Dictionary Building step, varying the S_D in the SD scheme from 100 to 4000, we found that $S_D = 2000$ yielded the best trade-off between speed and accuracy (as shown in Figure 5.2). With $S_D = 2000$, we obtain an accuracy of 95.54% at a speed of 7.4fps. For $S_D > 2000$, one can observe that accuracy only improves very slightly, while the speed falls rapidly. For example, $S_D = 4000$ results in a higher accuracy (96%), but the speed is reduced to 6.7fps. So, we choose $S_D = 2000$ in all our experiments based on BoF-VMMR with SD (BoF-SD), unless otherwise stated.

As for the MD scheme, we conducted similar experiments by varying the size of individual dictionaries (S_{ID}) that make up the main MD, and found that the $S_{ID} = 100$ (which makes the overall MD of size $S_D = S_{ID} \cdot Nc = 100 \cdot 29$), yielded the best trade-off between speed and accuracy (See Figure 5.3). Increasing the S_{ID} beyond 100 gradually decreases both speed and accuracy. Hence, in our experiments based on BoF-VMMR with MD (BoF-MD), we adopt $S_{ID} = 100$.

5.3.2 Optimised Classifiers

Based on the obtained optimized dictionaries, we then find the optimal classifiers for BoF-VMMR. To build the two classification schemes proposed in this work for BoF-VMMR, we utilize the multi-class SVM library of OpenCV [137] which is based on LibSVM [106, 138].

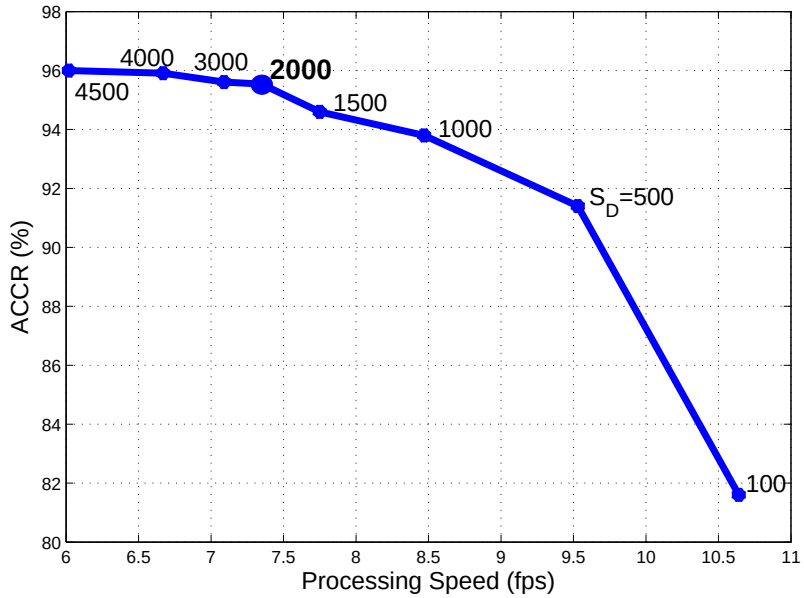


Figure 5.2: Effect of varying S_D (from 100 to 4000) on Accuracy and Speed of BoF-SD based VMMR. $S_D = 2000$ yields the best trade-off between speed and accuracy.

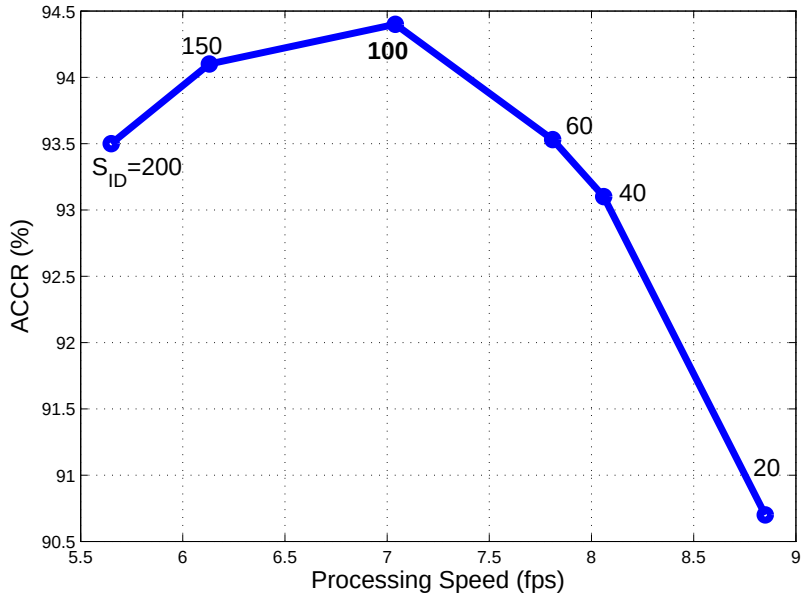


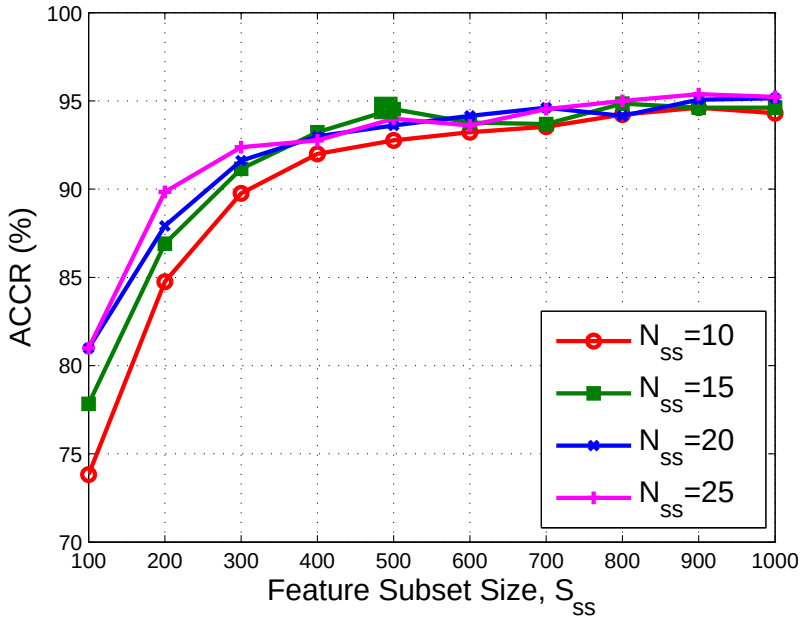
Figure 5.3: Effect of varying size of individual dictionaries (S_{ID}) from 20 to 200, on accuracy and processing speed of BoF-MD based VMMR. At $S_{ID} = 100$ (and thus the overall MD's size $S_D = S_{ID} \cdot N_c = 100 \cdot 29$), we obtain the best trade-off between speed and accuracy.

The values $C = 50$, $\gamma = 5$ for the individual C-SVC SVMs were empirically determined to yield the best results. So, we use these values in building the SSVM and ABSVM classifiers presented respectively in Sections 4.2.1 and 4.2.2.

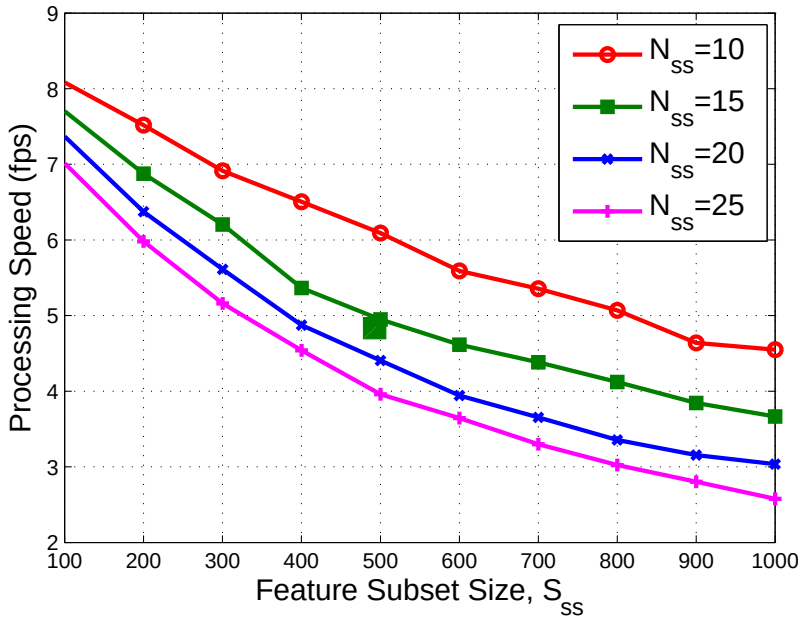
Given that the dimensionality of the BoF feature vectors is quite high, if the number of feature subspaces N_{ss}, and hence the number of classifiers, is not sufficiently large enough, there could be cases where some attributes may never be chosen. In the illustrative example given at the beginning of Section 4.2.2.2, if $N_{ss} = 2$, and say we choose the two feature subspaces as per A_1 and A_2 only, then we see that the 10^{th} attribute is left out. Similarly, if A_1 and A_3 are chosen, 4^{th} and 7^{th} attributes would be left out in the resulting feature subsets. It may occur that the omitted attributes had high discriminative capacity or significance. We decide on the optimal values for N_{ss} and S_{ss} based on experimental evaluations, as discussed below.

In this work, we choose the optimal values for feature subspace sizes (S_{ss}) and the number of feature subspaces (N_{ss}) by studying their effect on the processing speed and accuracy of BoF-SD with AB-SVM (See Figures 5.4a and 5.4b). We varied S_{ss} from 100 to 1000 (in steps of 100), for each test with $N_{ss} = 10, 15, 20$, and 25. One can observe from Figures 5.4a and 5.4b that while accuracy tends to increase with the increase in S_{ss} and N_{ss}, speed tends to decrease. This is obvious because a higher S_{ss} indicates greater dimensionality of the feature vectors, and a higher N_{ss} represents a greater number of classifiers in the ensemble, both of which lead to an increase in processing time consumption.

To find the optimal values of N_{ss} and S_{ss} for BoF-SD, we observe the accuracy vs. speed plot as shown in Figure 5.5. In this figure, we see that for accuracies above 94%, the speed tends to fall drastically while accuracy improves only slightly towards 95%. Our objective is to achieve a processing speed of at least 5 fps to meet real-time requirements while having a minimum accuracy of around 95%. We observe from these figures that $S_{ss} = 500$ and $N_{ss} = 15$ gives an accuracy of around 95% and speed of around 5 fps (represented by the highlighted green square datapoint in Figures 5.4a, 5.4b, and 5.5). Similar experiments were conducted for AB-SVM based BoF-MD and $N_{ss} = 15$, $S_{ss} = 1500$ yielded the best speed-accuracy trade-off. So, in our experiments, we adopt the values ($N_{ss} = 15$, $S_{ss} = 500$) and ($N_{ss} = 15$, $S_{ss} = 1500$) for the AB-SVM based BoF-SD and BoF-MD approaches respectively.



(a)



(b)

Figure 5.4: Effect of Feature Subset Sizes (S_{ss}) and Number of sub-samples (N_{ss}) on (a) Average Correct Classification Rate, and (b) Processing Speed of BoF-SD with AB-SVM

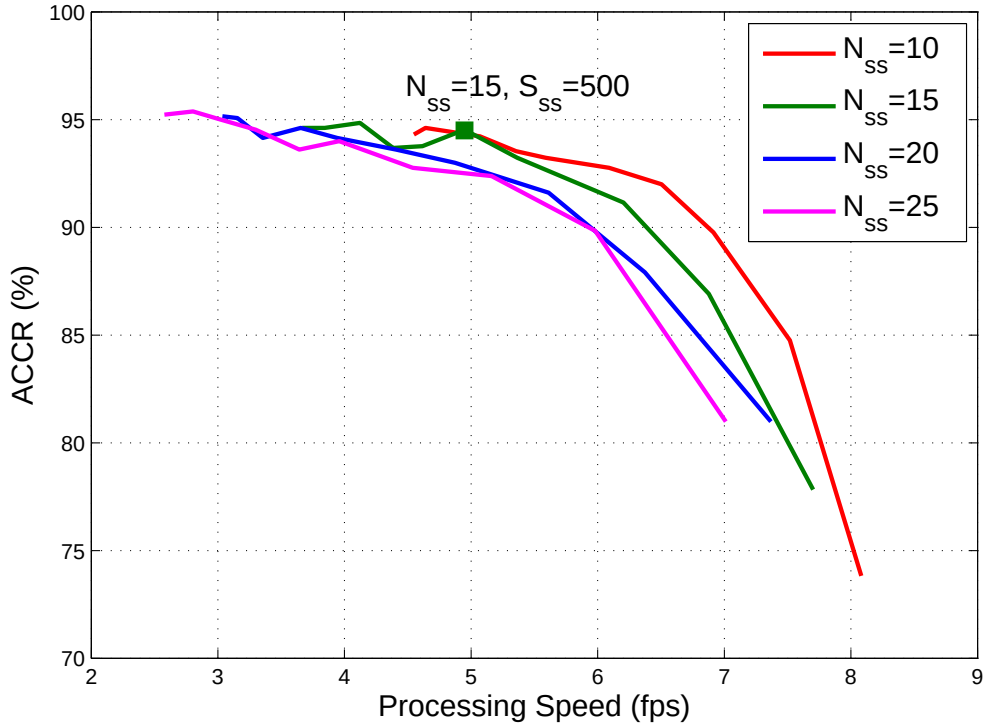


Figure 5.5: *ACCR* vs. Processing Speed of BoF-SD with AB-SVM for different Number of sub-samples (N_{ss}) and Feature Subset Sizes (S_{ss}). The best speed-accuracy trade-off is at $N_{ss} = 15$ and $S_{ss} = 500$

5.4 Vehicle Region of Interest

In most VMMR works, instead of using the whole input image for features extraction, global representation and classification, a narrow region of interest (ROI) over the vehicle is considered such that it covers the front (or rear) face. This eliminates background regions of input images, which could otherwise degrade VMMR performance. The works that focus on extracting this vehicle ROI from input images are classified under *Vehicle Detection* (VD) systems, which have been a very extensively investigated topic over the years. A comprehensive survey of VD techniques and approaches can be found in [139]. Most VMMR works use a pre-built VD module. Since the focus of our work is on improving methods to represent and classify vehicle makes and models, vehicle detection is out of this thesis' scope. Nonetheless, for the purpose of completeness, we provide here a brief overview of VD techniques that have been straight-forwardly utilized in VMMR works.

To detect and localize vehicles in a given image, cues such as bumper shadow [10, 140], air-grille [16], license-plate [6, 9], etc. have been popular in VMMR works due to fast processing speed and lower computational complexity than other advanced VD techniques. Since the objective of VMMR works is to augment traditional ALPR-based vehicle identification and classification systems, it is more efficient to use the detected license-plates as a cue based on which the vehicle ROIs can be defined around them. Many real-time and robust license-plate detection techniques such as our prior work of [141] have been proposed in the literature. Although license-plate recognition systems are highly failure-prone, the license-plate detection techniques have been proven to be highly robust to different lighting conditions and have the advantages of higher processing speed, lower computational complexity and minimal failure cases [142–145]. Given the license-plate coordinates and dimensions, the ROI with corresponding dimensions and aspect ratio is delineated. In fact, any real-time and robust VD technique can be coupled to our VMMR approaches.

5.5 Hardware and Software Platform

In all our experiments, the computing platform used is an Intel(R) Core(TM) i5 3475S CPU (2.9 GHz), with 16 GB RAM, Intel(R) HD Graphics 4000 card. The BoF-VMMR approaches were implemented using C++ and OpenCV 2.4.8 libraries through Microsoft Visual Studio 12 Professional, on a 64-bit Microsoft Windows 7 Operating System.

Chapter 6

Results and Discussions

In this chapter, we analyse the performance of proposed BoF-VMMR approaches. In Section 6.1, we evaluate the accuracy and efficiency (processing speed) of SSVM-based BoF-VMMR. In Section 6.2 we provide a comparative analysis of BoF-SD and BoF-MD, discussing pros and cons of each. We present the results of BoF-VMMR with ABSVM and Random Forest classification schemes, in Sections 6.3 and 6.4, respectively. Section 6.7 summarizes the results of the proposed BoF-VMMR approaches. The effectiveness of our approaches is established even in extremely challenging environmental and occlusion conditions, as shown in Section 6.5. In Section 6.8, we prove the superiority of our approaches over the representative VMMR works.

6.1 Performance of SSVM-based BoF-VMMR

In this section, we evaluate the performance of BoF-VMMR under both SD and MD schemes for Global Features Representation (of Sections 3.3.3.2 and 3.3.3.3 respectively), and SSVM (of Section 4.2.1) for Classification.

Table 6.1: Performance of BoF-SD with SURF and SSVM (*C*-SVC)

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	99	97.35	97.57	96.38	98	90.83	95.56	90.63
Avg #Cor	237.6	132.4	132.7	55.9	58.8	10.9	8.6	14.5
Make	Toyota		Suzuki	Nissan				
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	89.72	99.35	94.40	92.94	93.42	96.00	89.47	65.56
Avg #Cor	32.3	30.8	23.6	47.4	35.5	48	17	11.8
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	75.67	92.96	80.00	75.00	82.00	98.13	99.43	96.67
Avg #Cor	22.7	25.1	4.8	6	8.2	15.7	122.3	63.8
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	90.50	66.67	79.09	88.00	80.77	94.84		
Avg #Cor	18.1	2	17.4	17.6	10.5	1232		

6.1.1 Accuracy

6.1.1.1 C-SVC based SSVM

Utilizing the *C*-SVC formulation for SSVM (described in Section 4.1.1.2), we evaluate the BoF approaches based on SD and MD. The $ACCR_{l_i}$'s and the average number of correctly classified images for BoF-SD and BoF-MD are given in Tables 6.1 and 6.2, respectively. The accuracies are averaged over $N_D = 10$ random 80-20 training-testing dataset splits. The dictionary sizes used are $S_D = 2000$ (for SD) and $S_D = S_{ID} \cdot N_c = 100 \cdot 29$ (for MD), as described in Section 5.3. The mean average correct classification rates (*mACCR*) of our BoF-SD and BoF-MD approaches are 94.84% (95% CI [93.7%, 96.0%]) and 93.7% (95% CI [93.3%, 94.1%]) respectively.

The classwise $ACCR_{l_i}$'s with 95% confidence intervals, of our SURF- and SSVM-based BoF-SD and BoF-MD approaches, over the $N_D = 10$ different 80-20 training-testing dataset versions, are shown in Figure 6.1 Each of the ten dataset versions is populated by randomly choosing 80% of the total images for training and the rest for testing. While most of the classes have high

Table 6.2: Performance of BoF-MD with SURF and SSVM (C-SVC)

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	98.67	97.65	97.21	95.00	97.50	90.83	95.56	90.63
Avg #Cor	236.8	132.8	132.2	55.1	58.5	10.9	8.6	14.5
Make	Toyota		Suzuki	Nissan	Nissan			
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	88.61	98.06	93.20	88.82	93.42	96.00	89.47	65.56
Avg #Cor	31.9	30.4	23.3	45.3	35.5	48	17	11.8
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	75.67	92.96	83.33	80.00	75.00	99.38	99.27	94.24
Avg #Cor	22.7	25.1	5.0	6.4	7.5	15.9	122.1	62.2
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	89.00	70.00	80.00	84.00	77.69	93.70		
Avg #Cor	17.8	2.1	17.6	16.8	10.1	1217.2		

ACCR's, some classes consistently performed badly across the 10 different dataset versions, such as: classes 18 (Nissan Cefiro), 17 (Nissan Sentra), 18 (Nissan Cefiro), 22 (Mitsubishi Outlander), 23 (Mitsubishi Savrin), and 27 (Ford Escape).

It is noteworthy to mention that for some classes with low accuracy, e.g: classes 21 (Mitsubishi Zinger), 26 (Ford Liata), 28 (Ford Mondeo), and 29 (Ford Tierra), BoF-SD had *ACCR*'s of 95 – 100% with at least one of the dataset splits (see Figure 6.1). Similarly, for Class 27 (Ford Escape), although the accuracy (averaged over all N_D datasets) turned out to be 79.09%, for at least one of the 80-20 dataset splits, an $ACCR_{l_i}$ of more than 90% was achieved. This clearly indicates that accuracy greatly depends on how images were distributed into training and testing. Whereas we assign images to training-testing sets randomly, the assignment of images into training and testing sets in the original NTOU-MMR Dataset [10] is not clear and seems to be biased (described in Section 5.1).

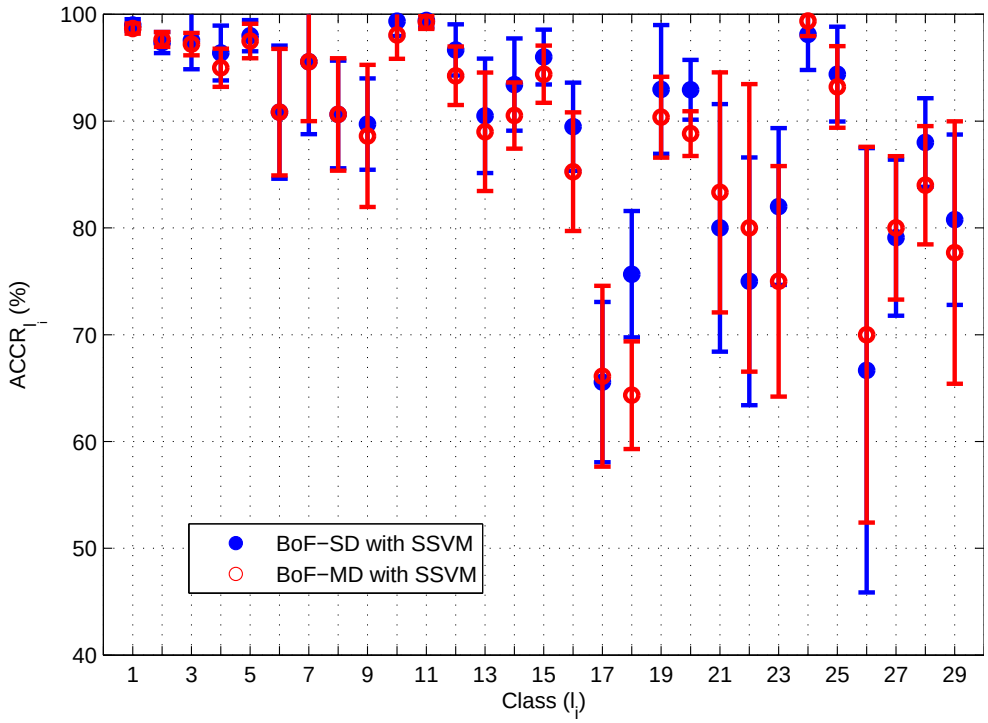


Figure 6.1: Classwise $ACCR_{l_i}$'s with 95% Confidence Intervals for BoF-SD and BoF-MD with SSVM (over the $N_D = 10$ different 80-20 Dataset splits).

6.1.1.2 ν-SVC based SSVM

We investigated another formulation of SSVM classifiers: the ν-SVC (described in Section 4.1.1.3). Since BoF-SD outperformed BoF-MD, we evaluate ν-SVC with BoF-SD. Tables 6.1 and 6.3 shows the results of BoF-SD with SSVM based on C-SVC and ν-SVC, respectively. These results were obtained using the empirically determined optimal values of $\nu = 0.02$ and $\gamma = 5$. Interestingly, with these optimal values, ν-SVC performed similarly to C-SVC in terms of overall accuracy $mACCR$. However, its average processing speed was about 7.4 fps, while that of BoF-SD with C-SVC was around 7.5 fps.

6.1.2 Speed

The average processing speeds of the SD- and MD-based BoF-VMMR approaches with SSVM are 7.5 fps and 6.99 fps respectively, which proves the suitability of BoF for real-time VMMR applications. Higher speeds can be obtained with a slight compromise in accuracy by decreasing

Table 6.3: Performance of BoF-SD with SURF and SSVM (ν-SVC)

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	99.13	97.50	97.65	96.72	98.00	90.83	95.56	91.25
Avg #Cor	237.9	132.6	132.8	56.1	58.8	10.9	8.6	14.6
Make	Toyota		Suzuki	Nissan				
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	89.17	100	94.0	91.96	92.11	95.0	89.47	65.0
Avg #Cor	32.1	31	23.5	46.9	35	47.5	17	11.7
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	75.67	92.96	85.0	73.75	84.0	97.5	99.67	96.52
Avg #Cor	22.7	25.1	5.1	5.9	8.4	15.6	122.6	63.7
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	89.0	66.67	80.0	88.0	80.0	94.80		
Avg #Cor	17.8	2	17.6	17.6	10.4	1231.5		

the S_D (as previously shown in Figures 5.2 and 5.3). Depending on the requirements of the specific application, the BoF parameters could be easily adapted to meet high processing speeds with a slight compromise in accuracy, or vice-versa. Common surveillance cameras have a frame rate of 25-30 fps [146, 147]. However, to run the VMMR on each and every incoming frame would waste computational resources. Instead, every 5^{th} incoming frame can be processed seamlessly for VMMR purposes, which effectively requires only 5-6 frames per second to be processed. In this manner, both of our BoF approaches are highly suitable for real-time VMMR applications.

6.2 Comparison of BoF-SD and BoF-MD

6.2.1 Accuracy and Speed

Observing the accuracies of BoF-SD and BoF-MD in Tables 6.1 and 6.2 respectively, one can see that the overall performance of BoF-SD is superior to that of BoF-MD. Contrary to our

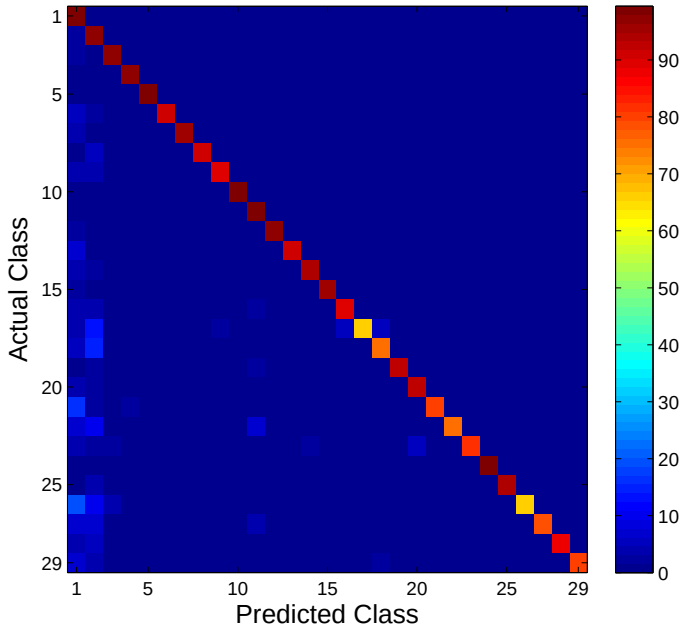
expectation that BoF-MD would perform better than BoF-SD, the results indicate otherwise. One reason could be the fixed size of individual sub-dictionaries of all classes, which could lead to many less discriminative and noisy features being selected as codewords in the overall dictionary. In future, we shall investigate dictionary pruning methods to build a more robust Modular Dictionary. However, we note that, for some classes, BoF-MD had better *ACCR*'s than BoF-SD. See for example: Toyota Camry, Nissan Sentra, Mitsubishi Zinger, Outlander, Lancer, Ford Liata, and Escape.

We show the confusion matrix for the 29 make-model classes using BoF-SD and BoF-MD approaches (with SURF and SSVM) in Figures 6.2a and 6.2b, respectively. One can observe that most of the inaccurate classifications are towards Classes 1 (Toyota Altis) and 2 (Toyota Camry). One of the major reasons for this effect could be a considerably greater amount of training data available for these classes (as we can see from Table 5.1) which may have lead to a biased classifier. Under such imbalanced training data, even the dictionary could have become biased by retaining more codewords from classes 1 and 2 than from other classes.

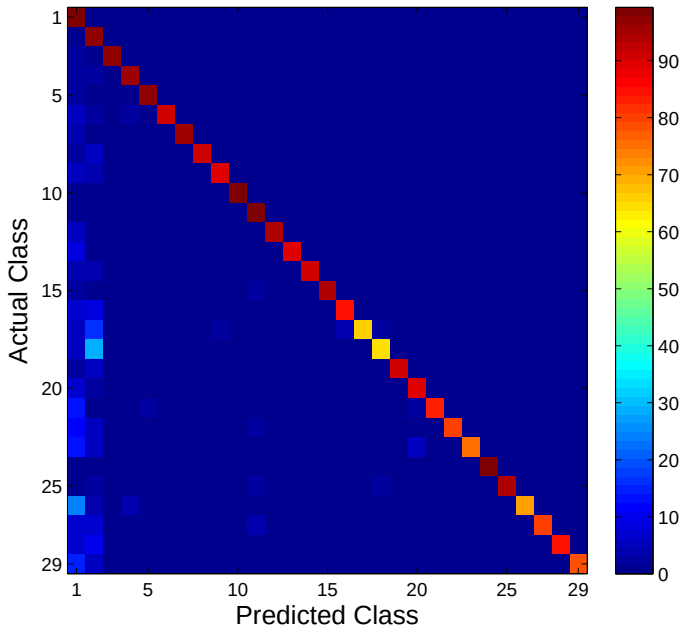
The average processing speeds of the BoF-SD and BoF-MD with SSVM are 7.5 fps and 6.99 fps, respectively. The BoF-MD works slightly slower than BoF-SD due to its greater size of the overall dictionary. Whereas the overall SD has size $S_D = 2000$, the overall MD has size $S_D = S_{ID} \cdot 29 = 2900$, i.e. size of MD is around 45% greater than that of SD.

6.2.2 Dictionary Training Time

Although the accuracy and speed of BoF-MD-VMMR is slightly less than compared to BoF-SD, the time required to build or re-build the MD, i.e. the *Dictionary Training Time (T_{DTr})*, is drastically less than compared to SD's T_{DTr}. One can observe in Figure 6.3 that there is a huge difference in T_{DTr} for MD and SD. Unlike SD, the increase in dictionary size does not cause the T_{DTr} of MD to increase rapidly. The cost in time for training and re-training of MD is therefore significantly less than that of SD. In real-life scenarios, security personnel may be looking for different subsets of vehicle makes and models at different times. Therefore, a VMMR system should recognize only those makes and models, rather than all that are passing by. In such applications where re-building of dictionaries due to addition or removal of desired or undesired



(a)



(b)

Figure 6.2: Confusion Matrices for (a) BoF-SD and (b) BoF-MD, both using SURF and SSVM; values averaged over the ten 80-20 Datasets.

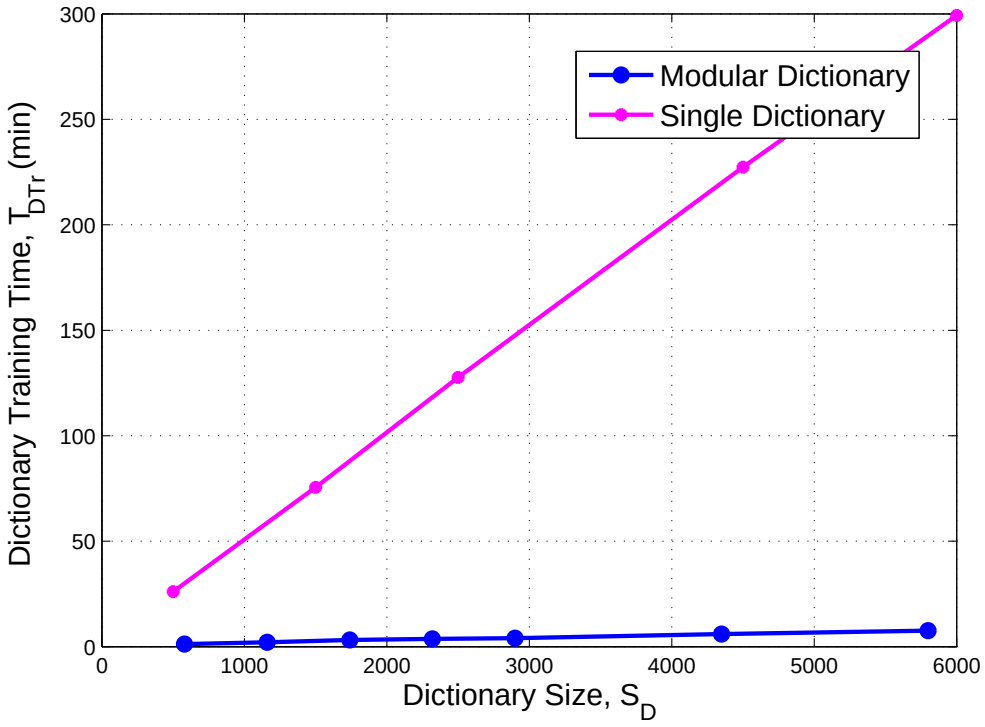


Figure 6.3: Effect of Dictionary Size (S_D) on the Dictionary Training Time (T_{DTr}), for the Single and Modular Dictionaries.

make-model classes is needed, BoF-MD would be a more efficient choice. However, in applications where the reconstruction of the dictionary is unnecessary, then BoF-SD stands as a better choice.

6.3 Performance of AB-SVM based BoF-VMMR

Motivated by the success of using AB to build an ensemble of classifiers in several works such as [97], we investigate whether the AB-based ensemble of multi-class SVM classifiers (AB-SVM), trained over BoF representations of different makes and models, could improve the performance of VMMR in comparison to the SSVM classifier of Section 4.2.1.

The classwise $ACCR_{l_i}$'s and average number of correctly classified images for AB-SVM based BoF-SD and BoF-MD approaches are shown in Tables 6.4 and 6.5, respectively. Figure 6.4 shows the $ACCR_{l_i}$'s with their 95% confidence intervals. These results are obtained using (N_{ss} = 15, S_{ss} = 500) for BoF-SD, and (N_{ss} = 15, S_{ss} = 1500) for BoF-MD, as obtained in Sec-

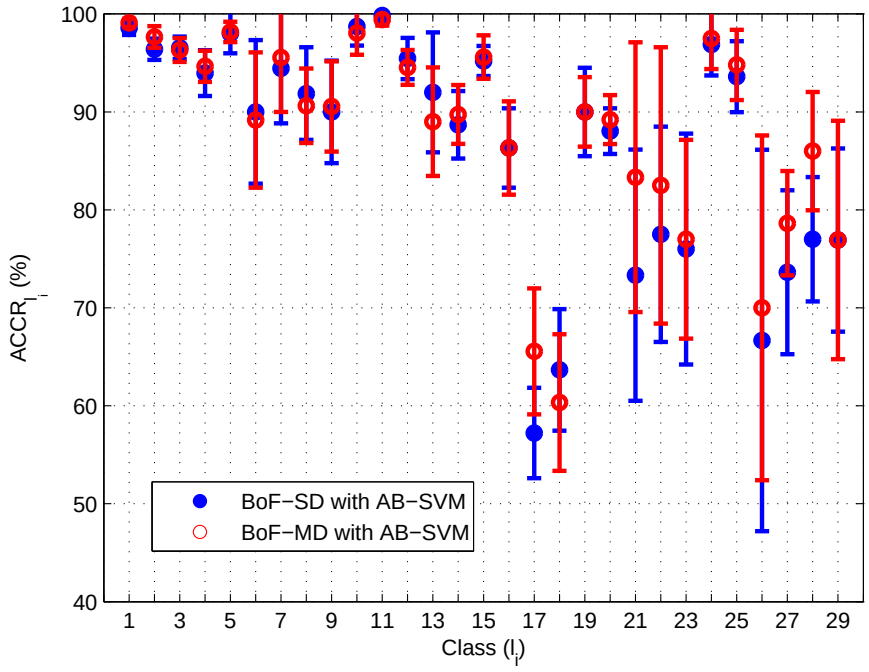


Figure 6.4: Classwise $ACCR_{l_i}$'s with 95% Confidence Intervals for BoF-SD and BoF-MD with AB-SVM (over the $N_D = 10$ different 80-20 Dataset splits).

tion 5.3. The average processing speed with these configurations was around 5 fps with SD, and around 3 fps with MD. The average accuracies with SD and MD turned out to be around 93.02% (95% CI [92.5%, 93.6%]) and 93.7% (95% CI [93.4%, 94%]), respectively. It can be observed that BoF-MD with AB-SVM had very similar accuracy as BoF-MD with SSVM, although processing speed was compromised. However, the performance of BoF-SD with AB-SVM (in terms of accuracy and speed) is slightly reduced, compared to SSVM-based BoF-SD and BoF-MD. The reduced performance of AB-SVM based BoF-SD could be attributed to the random selection of dictionary codewords to form feature subsets, without considering the importance (or discriminative capacity) of the selected codewords. This could have led to the selection of noisy or non-discriminative codewords. By incorporating the discriminative capacity of the codewords, or by increasing N_{ss} (as discussed in Section 5.3), AB-SVM could perform better with BoF for VMMR.

We also observe that with AB-SVM, BoF-MD yields higher $ACCR_{l_i}$'s than BoF-SD, for the following classes: Toyota Altis, Toyota Camry, Toyota Wish, Toyota Yaris, Toyota Innova, Toy-

Table 6.4: Performance of BoF-SD with AB-SVM

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	98.58	96.40	96.54	93.97	98.00	90.00	94.44	91.88
Avg #Cor	236.6	131.1	131.3	54.5	58.8	10.8	8.5	14.7
Make	Toyota		Suzuki	Nissan				
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	90.00	98.71	93.6	88.04	88.68	95.20	86.32	57.22
Avg #Cor	32.4	30.6	23.4	44.9	33.7	47.6	16.4	10.3
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	63.67	90.00	73.33	77.50	76.00	96.88	99.84	95.45
Avg #Cor	19.1	24.3	4.4	6.2	7.6	15.5	122.8	63.0
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	92.00	66.7	73.64	77.00	76.92	93.02		
Avg #Cor	18.4	2.0	16.2	15.4	10.0	1210.5		

ota Tercel, Suzuki Solio, Nissan Tiida, Nissan March, Nissan Livna, Nissan Sentra, Mitsubishi Zinger, Mitsubishi Outlander, Mitsubishi Savrin, Mitsubishi Lancer, Ford Liata, Ford Escape, and Ford Mondeo. That is, for around 18 out of 29 classes, BoF-MD actually outperforms BoF-SD. BoF-MD yields lower accuracies only for 8 of the 29 classes: Toyota Vios, Toyota Previa, Toyota Surf, Toyota RAV4, Nissan Cefiro, Honda CRV, Honda Civic, Honda FIT. The decreases in accuracies of BoF-MD for these 8 classes leads to the decrease in overall BoF-MD's accuracy when compared to BoF-SD's.

6.4 Performance of Random Forest-based BoF-VMMR

We also investigated the use of Random Forest, a tree-based classifier, with BoF-VMMR. In these experiments, we use the best performing BoF-VMMR approach, i.e., BoF-SD with SSVM. Table 6.6 summarizes the results obtained. The optimal Random Forest parameters were empirically determined for these experiments. One can observe that tree-based classifier such as Random Forest didn't work well with BoF-based representations for VMMR. Although Random

Table 6.5: Performance of BoF-MD with AB-SVM

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	99.08	97.65	96.32	94.66	98.17	89.17	95.56	90.63
Avg #Cor	237.8	132.8	131.0	54.9	58.9	10.7	8.6	14.5
Make	Toyota		Suzuki	Nissan				
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	90.56	98.06	94.80	89.22	89.74	95.60	86.32	65.56
Avg #Cor	32.6	30.4	23.7	45.5	34.1	47.8	16.4	11.8
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	60.33	90.00	83.33	82.50	77.00	97.50	99.43	94.55
Avg #Cor	18.1	24.3	5.0	6.6	7.7	15.6	122.3	62.4
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	89.00	70.00	78.64	86.00	76.92	93.7		
Avg #Cor	17.8	2.1	17.3	17.2	10.0	1217.9		

Forest may be a good classifier for VTR (as in [11]), its performance is very limited in the context of VMMR. This clearly shows that VMMR is a much more challenging problem than VTR. By observing the confusion matrix in Figure 6.5, one can see that the confusions are scattered between classes, indicating the poor discriminating capacity of Random Forest classifier for the BoF-SD-based global representations of vehicle makes and models. The low performance of Random Forest could be attributed to the high-dimensionality of BoF-VMMR's global features representations, and possible dependencies among their attributes. So, we recommend multi-class SVM-based classifiers for BoF-VMMR.

6.5 Performance in Challenging Conditions

The BoF-based approaches for VMMR perform well even in challenging scenarios such as vehicles under occlusion, non-frontal views, and low lighting, as depicted in Figure 6.6. The invariance to such challenging conditions could be attributed to the sparse nature of BoF-based global

Table 6.6: Performance of BoF-SD with Random Forest classifier

Make	Toyota							
Model	Altis	Camry	Vios	Wish	Yaris	Previa	Innova	Surf
ACCR(%)	71.67	69.85	86.03	96.55	96.67	91.67	77.78	81.25
#Cor	172	95	117	56	58	11	7	13
Make	Toyota		Suzuki	Nissan				
Model	Tercel	RAV4	Solio	Tiida	March	Livna	Teana	Sentra
ACCR(%)	86.11	100	84.00	82.35	55.26	98.00	84.21	33.33
#Cor	31	31	21	42	21	49	16	6
Make	Nissan		Mitsubishi				Honda	
Model	Cefiro	Xtrail	Zinger	Outlander	Savrin	Lancer	CRV	Civic
ACCR(%)	66.67	81.48	83.33	62.50	70.00	93.75	94.31	96.97
#Cor	20	22	5	5	7	15	116	64
Make	Honda	Ford				Overall		
Model	FIT	Liata	Escape	Mondeo	Tierra			
ACCR(%)	95.00	66.67	72.73	50.00	69.23	81.29		
#Cor	19	2	16	10	9	1056		

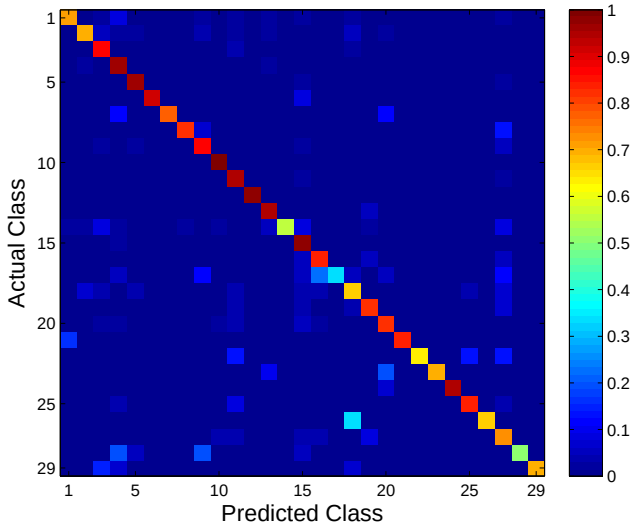


Figure 6.5: Confusion Matrix of Random Forest based BoF-SD, using a 80-20 NTOU-MMR dataset.

representations, in which the non-zero values are the aggregated votes of similar keypoint-based patches.

When the vehicle face is under occlusion, there can be two cases: (a) The occluding object has a relatively texture-less surface (e.g., a uni-color umbrella), or (b) the occluding object has a highly varying texture or appearance (e.g., a person). In the former case, there are little or no keypoint-based patches (due to scarcity or absence of corners), and hence it doesnt affect the overall BoF representation. In the latter case, the occluding object may also result in keypoint-based patches (due to presence of corners). These occluding patches would cast votes to the dictionary codewords, thereby adding noise to the overall BoF representation. There could be two sub-cases in such a scenario: (a) The occluding patches are very widely scattered in feature-space and hence cast scattered votes (noise per bin of the BoF histogram is very minimal), or (b) The occluding patches may be close in feature-space and hence add a considerable noise to the overall BoF histogram. In the former sub-case, since the noise is distributed, the overall BoF histogram's shape will not be affected considerably. However, in the latter sub-case, the overall BoF histogram's shape could be severely affected, leading to inaccurate predictions.

6.6 Testing in a Mobile Scenario

In order to test our BoF-VMMR in a mobile scenario, we conducted a simple experiment on a video captured from a moving vehicle in a parking lot. There were 6 different vehicles in this video: Audi Q5, Nissan Sentra, Volkwagen Jetta, Mazda 5, Audi A3, and Honda Civic. To the best of our knowledge, most VMMR works are based on images/videos from a fixed-camera only. In contrast, we make an attempt to test the feasibility of our approaches in a scenario where the camera is moving, representing a police vehicle patrolling in a parking lot.

The video was captured at a rate of 29 fps, by SONY HDR-PJ710V camera. The vehicle ROIs were manually constructed, assigning the respective ground-truth labels. For training, around 4-5 images for each make-model were used. Since every vehicle is captured in many number of frames and it would waste computational resources and time to operate on every frame, we run our BoF-SD on every 5th frame.

Figure 6.6: Some challenging cases of vehicles under occlusion (6.6a-6.6h), partially out of the camera's view (6.6i-6.6j), non-frontal views (6.6k-6.6m), or under low lighting (6.6m-6.6n). The BoF-based VMMR approaches were successful in predicting the make-model class in the above cases.

Figure 6.7: Testing BoF-SD in a mobile scenario

All 6 of the vehicle MMs were recognized correctly, as demonstrated in Figure 6.7. These results indicate the high potential of our BoF-VMMR approaches to be applied in real-world mobile scenarios.

6.7 Performance Summary of BoF-VMMR Approaches

A summary of *mACCR*'s and processing speeds of our BoF-based approaches for VMMR is given in Table 6.7. All our approaches yielded high accuracies for VMMR on the real-world dataset. Overall performance of BoF-SD is slightly better than that of BoF-MD. Interestingly however, for some classes, BoF-MD yielded higher accuracies than BoF-SD. We also observed in Section 5.3.1 that BoF-SD yields even higher accuracies (upto 96%), with a slight compromise on speed, when dictionary size is increased. So, depending on the specific application's requirements, respective parameters can be adapted.

Table 6.7: Performance Summary of our BoF-VMMR Approaches

Approach	Classifier	mACCR (%)				Speed (fps)				T_{DTr} (min)
		Max	Avg.	Min	95% CI	Max	Avg.	Min	95% CI	
BoF-SD	SSVM	95.77	94.84	94.00	93.7-96	7.6	7.5	7.25	7.4-7.51	101.5
	AB-SVM	94.53	93.02	91.46	92.5-93.6	5.01	4.97	4.94	4.95-4.99	
BoF-MD	SSVM	94.38	93.7	92.3	93.3-94.1	7.25	6.99	6.51	6.82-7.15	3.95
	AB-SVM	94.3	93.68	92.76	93.4-94	2.83	2.81	2.78	2.8-2.83	

With regards to dictionary training time, BoF-MD can re-train dictionaries 26x faster than BoF-SD. So, for applications with frequently changing requirements, BoF-MD is recommended. Comparing between the classification schemes, ABSVM performed with comparable accuracies but slightly slower speeds than SSVM.

Based on our findings, we recommend using a single multi-class SVM-based BoF-SD or BoF-MD for real-time VMMR systems, owing to their higher processing speeds and accuracies. The BoF-SD with ABSVM (which offers an average processing speed of around 5 fps) could also be a good choice, depending on the application's requirement.

6.8 Comparisons with Related Works on NTOU-MMR Dataset

The proposed BoF-based approaches for VMMR presented in this paper outperform several related VMMR works, both in terms of processing speed and classification accuracy. A performance comparison of our work with results of other related works on the NTOU-MMR Dataset is presented in Table 6.8. Both of our BoF approaches (BoF-SD and BoF-MD) significantly outperform the works of [65],[72], and [15]. The work in [65] employs a brute-force matching scheme of the local SIFT features, making it highly inefficient for real-time VMMR systems. The approach of [72] results in the worst performance, due to its reliance on edge pixels to build global representations of vehicle makes and models, which are prone to image noises and occlusions, and which lack discriminating power. The low accuracy of [15]'s work indicates the inefficiency of corner- and gradient-responses based global feature vectors for the VMMR problem.

Table 6.8: Performance Comparison of BoF-VMMR with Other Works

Method	mACCR (%)	Speed (fps)
[65]	90.52	-
[72]	67.33	-
[15]	85.9	-
FID-SRC-HDC [7]	91.1	0.46
BoF-SD-(SVM)	**94.84**	**7.5**
BoF-SD-(AB-SVM)	93.02	4.97
BoF-MD-(SVM)	93.7	6.99
BoF-MD-(AB-SVM)	93.68	2.81

More recently, a sparse representation scheme and a Hamming Distance-based classification for VMMR was proposed by Chen et al. [7]. Considering the average accuracy and speed of their best performing scheme (referred to as FID-SRC-HDC) on the 29 make-model classes of the NTOU-MMR Dataset [135], we see in Table 6.8 that our approaches prove to be significantly better (atleast 15x faster). Based on the above comparisons, one can conclude that the BoF-VMMR approaches presented in this thesis are superior in terms of both accuracy and processing speed.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis proposes and investigates unexplored approaches for highly robust and real-time automated vehicle make and model recognition (VMMR) based on the Bag-of-Features paradigm. The major contributions of this work are as follows: (1) Two distinct schemes for optimized Dictionary Building are proposed and investigated to address the multiplicity and ambiguity problems of VMMR; (2) The optimal dictionary sizes for both dictionaries are recommended via experimental evaluations in the context of VMMR; (3) Two different multi-class classification schemes are proposed and evaluated for accurate and efficient make-model prediction: (a) Single-SVM and (b) Attribute Bagging based Ensemble of SVMs (AB-SVM).

Both the Single and Modular dictionaries performed with high accuracies and speeds. While the former yielded slightly higher overall accuracy and speed, the latter demonstrated an advantage of having a drastically lower dictionary training time (around 26x faster than that of the former). So, in applications that require often re-training of dictionaries (e.g., due to changes in the list of makes and models being targeted), the latter serves as a better choice for VMMR. This thesis showed that the dictionary size is a parameter that can be adapted to increase accuracy with a slight compromise on speed, or vice-versa, depending on the application's requirements.

Amongst the classification schemes, both SSVM and ABSVM classifiers performed with high accuracies, with SSVM being slightly faster. However, tree-based classifiers such as Random Forest yielded very low accuracies, indicating their lack of discriminating capacity over the BoF-based global representations of vehicle makes and models.

This thesis also demonstrated that the way datasets are split into training and testing can yield varying accuracies. Hence, unlike most related works, the effectiveness and superiority of our approaches over the related works are validated using random training-testing splits of the recent real-world NTOU-MMR Dataset [135]. Thorough experimental evaluations have shown that our BoF-based VMMR approaches are highly suitable for real-time automated vehicle classification applications. The superiority of our approaches has been demonstrated even in challenging cases such as partial or non-frontal views, occlusions, low illumination, and in mobile scenarios.

7.2 Future Work

For future work, we plan to enhance state-of-the-art in VMMR along the following directions.

- As an enhancement to the proposed BoF-based VMMR approaches, the optimized dictionaries learned from different make-model classes could be further enhanced by pruning non-discriminative codewords. This could result in more compact, accurate and discriminative global features representations, thereby enhancing VMMR accuracy and speed.
- A plethora of feature encoding schemes have been proposed for other image classification problems. Many of them remain unexplored in the context of VMMR. We plan to investigate state-of-the-art feature encoding techniques to solve the VMMR challenges.
- All the more, necessitated by the lack of a standard publicly available benchmark dataset for VMMR, we plan to build a comprehensive VMMR dataset that exhibits the real-world challenges in VMMR and includes vehicles of a wider variety of colors, types, makes and models.
- Most current works focus on stationary VMMR systems, relying on the fixed traffic and surveillance cameras. A future direction of research is to develop real-time mobile VMMR

systems which could be installed in mobile devices such as smart phones, or even on patrolling security vehicles.

- Another direction to explore is towards a large-scale and distributed real-time VMMR system that can be deployed in a city-wide context, utilizing the networked vehicles and other components of the ITS infrastructure.

- A relatively untouched area of research is on Vehicle Make, Model and Generation Recognition (VMMGR), which involves identifying the year-range or generation of vehicles, besides their makes and models. This could possibly be achieved through a two-stage approach, in which the first stage identifies the make and model, while the second stage identifies the generation.

References

[1] ITS Canada, "Intelligent Transportation." Last accessed on 11-August-2015.

[2] Office of the Assistant Secretary for Research and Technology (OST-R), ITS Joint Program Office, U.S. Department of Transportation (US DOT), "RITA - Intelligent Transportation Systems." Last accessed on 11-August-2015.

[3] ITS Canada, "ITS in Soceity." Last accessed on 11-August-2015.

[4] NSERC DIVA Strategic Research Network, "DIVA — Developing Next Generation Intelligent Vehicular Networks And Applications." Last accessed on 11-August-2015.

[5] European Commission, "Home - Transport - Research and Innovation - European Commission." Last accessed on 11-August-2015.

[6] H. He, Z. Shao, and J. Tan, "Recognition of Car Makes and Models From a Single Traffic-Camera Image," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–11, 2015.

[7] L.-C. Chen, J.-W. Hsieh, Y. Yan, and D.-Y. Chen, "Vehicle Make and Model Recognition using Sparse Representation and Symmetrical SURFs," *Pattern Recognition*, vol. 48, no. 6, pp. 1979 – 1998, 2015.

[8] D. Llorca, D. Colas, I. Daza, I. Parra, and M. Sotelo, "Vehicle model recognition using geometry and appearance of car emblems from rear view images," in *17th IEEE International Conference on Intelligent Transportation Systems*, pp. 3094–3099, Oct 2014.

[9] M. Fraz, and E. A. Edirisinghe, and M. S. Sarfraz , "Mid-level-Representation Based Lexicon for Vehicle Make and Model Recognition," in *22nd International Conference on Pattern Recognition (ICPR)*, pp. 393–398, Aug 2014.

[10] J.-W. Hsieh, L.-C. Chen, and D.-Y. Chen, "Symmetrical SURF and Its Applications to Vehicle Detection and Vehicle Make and Model Recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 6–20, Feb 2014.

[11] B. Zhang, "Reliable classification of vehicle types based on cascade classifier ensembles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 322–332, March 2013.

[12] V. Varjas and A. Tanacs, "Car recognition from frontal images in mobile environment," in *Image and Signal Processing and Analysis (ISPA), 2013 8th International Symposium on*, pp. 819–823, Sept 2013.

[13] L.-C. Chen, J.-W. Hsieh, Y. Yan, and D.-Y. Chen, "Vehicle Make and Model Recognition using Sparse Representation and Symmetrical SURFs," in *16th International IEEE Conference on Intelligent Transportation Systems*, pp. 1143–1148, Oct 2013.

[14] M. AbdelMaseeh, I. Badreldin, M. Abdelkader, and M. El Saban, "Car Make and Model recognition combining global and local cues," in *21st International Conference on Pattern Recognition (ICPR)*, pp. 910–913, Nov 2012.

[15] G. Pearce and N. Pears, "Automatic make and model recognition from frontal images of cars," in *8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pp. 373–378, Aug 2011.

[16] A. Psyllos, C. Anagnostopoulos, and E. Kayafas, "Vehicle model recognition from frontal view image measurements," *Computer Standards & Interfaces*, vol. 33, no. 2, pp. 142 – 151, 2011. XVI IMEKO TC4 Symposium Ëxploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurementsänd XIII International Workshop on ADC Modelling and Testing.

[17] A. Boukerche, J. Feng, and X. Fei, "V-Square: An Accurate Time Synchronization Protocol for Wireless Video Sensor Networks," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–5, Nov 2008.

[18] A. Boukerche, "Wireless multimedia sensor and actuator system: A necessary public security and safety testbed for an urban emergency preparedness class of applications," in *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*, pp. 1–1, May 2010.

[19] C. Rezende, A. Mammeri, A. Boukerche, and A. A. Loureiro, "A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection," *Ad Hoc Networks*, vol. 17, pp. 1 – 17, 2014.

[20] C. Rezende, A. Boukerche, M. Almulla, and A. A. Loureiro, "The selective use of redundancy for video streaming over vehicular ad hoc networks," *Computer Networks*, vol. 81, pp. 43 – 62, 2015.

[21] C. Rezende, A. Boukerche, H. Ramos, and A. Loureiro, "A reactive and scalable unicast solution for video streaming over vanets," *IEEE Transactions on Computers*, vol. 64, pp. 614–626, March 2015.

[22] Y. Li and A. Boukerche, "Qugu: A quality guaranteed video dissemination protocol over urban vehicular ad hoc networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, pp. 55:1–55:23, June 2015.

[23] R. Wang, M. Almulla, C. Rezende, and A. Boukerche, "Video streaming over vehicular networks by a multiple path solution with error correction," in *2014 IEEE International Conference on Communications (ICC)*, pp. 580–585, June 2014.

[24] Y. Li, F. Naeimipoor, and A. Boukerche, "Video dissemination protocols in urban vehicular ad hoc network: A performance evaluation study," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pp. 2611–2616, April 2014.

[25] R. Wang, C. Rezende, H. Ramos, R. Pazzi, A. Boukerche, and A. Loureiro, "Liaithon: A location-aware multipath video streaming scheme for urban vehicular networks," in *2012*

IEEE Symposium on Computers and Communications (ISCC), pp. 000436–000441, July 2012.

[26] F. Naeimipoor, C. Rezende, and A. Boukerche, "Performance evaluation of video dissemination protocols over vehicular networks," in *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pp. 694–701, Oct 2012.

[27] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro, "Localization systems for wireless sensor networks," *Wireless Communications, IEEE*, vol. 14, pp. 6–12, December 2007.

[28] A. Boukerche, H. A. Oliveira, E. F. Nakamura, and A. A. Loureiro, "Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems ," *Computer Communications* , vol. 31, no. 12, pp. 2838 – 2849, 2008. Mobility Protocols for ITS/VANET.

[29] A. Boukerche, ed., *Algorithms and protocols for wireless sensor networks*. John Wiley Sons, 2008.

[30] H. de Oliveira, A. Boukerche, E. Freire Nakamura, and A. Loureiro, "An efficient directed localization recursion protocol for wireless sensor networks," *IEEE Transactions on Computers*, vol. 58, pp. 677–691, May 2009.

[31] H. A. Oliveira, A. Boukerche, E. F. Nakamura, and A. A. Loureiro, "Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm," *Performance Evaluation*, vol. 66, no. 35, pp. 209 – 222, 2009. Modeling and Analysis of Wireless Networks: Selected Papers from {MSWiM} 2007.

[32] O. Abumansoor and A. Boukerche, "A secure cooperative approach for nonline-of-sight location verification in vanet," *Vehicular Technology, IEEE Transactions on*, vol. 61, pp. 275–285, Jan 2012.

[33] S. Samarah, M. Al-Hajri, and A. Boukerche, "A predictive energy-efficient technique to support object-tracking sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 656–663, Feb 2011.

[34] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '04, (New York, NY, USA), pp. 157–164, ACM, 2004.

[35] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments," *Journal of Parallel and Distributed Computing*, vol. 66, no. 4, pp. 586 – 599, 2006. Algorithms for Wireless and Ad-Hoc Networks.

[36] R. W. Pazzi and A. Boukerche, "Mobile data collector strategy for delay-sensitive applications over wireless sensor networks," *Computer Communications*, vol. 31, no. 5, pp. 1028 – 1039, 2008. Mobility Management and Wireless Access.

[37] M. Elhadef, A. Boukerche, and H. Elkadiki, "A distributed fault identification protocol for wireless and mobile ad hoc networks," *Journal of Parallel and Distributed Computing*, vol. 68, no. 3, pp. 321 – 335, 2008. Wireless Mesh Networks: Behavior, Artifacts and Solutions.

[38] A. Boukerche, A. Martirosyan, and R. Pazzi, "An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks," *Mob. Netw. Appl.*, vol. 13, pp. 614–626, Dec. 2008.

[39] A. Boukerche, R. Pazzi, and R. Araujo, "Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol," in *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pp. 560–567, Nov 2005.

[40] A. Prati and F. Qureshi, "Integrating consumer smart cameras into camera networks: Opportunities and obstacles," *Computer*, vol. 47, pp. 45–51, May 2014.

[41] A. Boukerche, S. Hong, and T. Jacob, "An efficient synchronization scheme of multimedia streams in wireless and mobile systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, pp. 911–923, Sep 2002.

[42] A. Boukerche and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *Wireless Communications, IEEE*, vol. 14, pp. 64–69, October 2007.

[43] Z. Zhang, R. W. Pazzi, and A. Boukerche, "A mobility management scheme for wireless mesh networks based on a hybrid routing protocol," *Computer Networks*, vol. 54, no. 4, pp. 558 – 572, 2010. Advances in Wireless and Mobile Networks.

[44] Y. Ren, R. Pazzi, and A. Boukerche, "Monitoring patients via a secure and mobile healthcare system," *Wireless Communications, IEEE*, vol. 17, pp. 59–65, February 2010.

[45] A. Boukerche and Y. Ren, "A secure mobile healthcare system using trust-based multicast scheme," *Selected Areas in Communications, IEEE Journal on*, vol. 27, pp. 387–399, May 2009.

[46] Y. Ren and A. Boukerche, "Modeling and managing the trust for wireless and mobile ad hoc networks," in *Communications, 2008. ICC '08. IEEE International Conference on*, pp. 2129–2133, May 2008.

[47] A. Boukerche and Y. Ren, "A trust-based security system for ubiquitous and pervasive computing environments," *Computer Communications*, vol. 31, no. 18, pp. 4343 – 4351, 2008. Secure Multi-Mode Systems and their Applications for Pervasive Computing.

[48] A. Boukerche and X. Li, "An agent-based trust and reputation management scheme for wireless sensor networks," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 3, pp. 5 pp.–, Nov 2005.

[49] X. Ma and W. Grimson, "Edge-based rich representation for vehicle classification," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1185–1192 Vol. 2, Oct 2005.

[50] L. Zhang, S. Li, X. Yuan, and S. Xiang, "Real-time object classification in video surveillance based on appearance learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[51] N. Buch, J. Orwell, and S. A. Velastin, "3d extended histogram of oriented gradients (3DHOG) for classification of road users in urban scenes," in *British Machine Vision Conference, London, UK, September 7-10*, pp. 1–11, 2009.

[52] Z. Chen, T. Ellis, and S. Velastin, "Vehicle type categorization: A comparison of classification schemes," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 74–79, Oct 2011.

[53] Z. Chen, T. Ellis, and S. Velastin, "Vehicle detection, tracking and classification in urban traffic," in *15th International IEEE Conference on Intelligent Transportation Systems*, pp. 951–956, Sept 2012.

[54] R. Wang, L. Zhang, K. Xiao, R. Sun, and L. Cui, "Easisee: Real-time vehicle classification and counting via low-cost collaborative sensing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 414–424, Feb 2014.

[55] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, "Vehicle type classification using unsupervised convolutional neural network," in *22nd International Conference on Pattern Recognition (ICPR)*, pp. 172–177, Aug 2014.

[56] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–10, 2015.

[57] A. Psyllos, C.-N. Anagnostopoulos, and E. Kayafas, "Vehicle logo recognition using a sift-based enhanced matching scheme," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, pp. 322–328, June 2010.

[58] S. Yu, S. Zheng, H. Yang, and L. Liang, "Vehicle logo recognition based on bag-of-words," in *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pp. 353–358, Aug 2013.

[59] D. Llorca, R. Arroyo, and M. Sotelo, "Vehicle logo recognition in traffic images using hog features and svm," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pp. 2229–2234, Oct 2013.

[60] Y. Ou, H. Zheng, S. Chen, and J. Chen, "Vehicle logo recognition based on a weighted spatial pyramid framework," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1238–1244, Oct 2014.

[61] J. Xiao, W. Xiang, and Y. Liu, "Vehicle logo recognition by weighted multi-class support vector machine ensembles based on sharpness histogram features," *Image Processing, IET*, vol. 9, no. 7, pp. 527–534, 2015.

[62] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, pp. 1951–1960, Aug 2015.

[63] H. Peng, X. Wang, H. Wang, and W. Yang, "Recognition of low-resolution logos in vehicle images based on statistical random sparse distribution," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, pp. 681–691, April 2015.

[64] P. Badura and M. Skotnicka, "Automatic Car Make Recognition in Low-Quality Images," in *Information Technologies in Biomedicine, Volume 3* (E. Pitka, J. Kawa, and W. Wieclawek, eds.), vol. 283 of *Advances in Intelligent Systems and Computing*, pp. 235–246, Springer International Publishing, 2014.

[65] L. Dlagnekov, "Video-based Car Surveillance: License Plate, Make, and Model recognition," Master's thesis, University of California, San Diego, 2005.

[66] D. Lowe, "Object recognition from local scale-invariant features," in *7th IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[67] R. Baran, A. Glowacz, and A. Matiolanski, "The efficient real- and non-real-time make and model recognition of cars," *Multimedia Tools and Applications*, pp. 1–20, 2013.

[68] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[69] D. Jang and M. Turk, "Car-rec: A real time car recognition system," in *IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 599–605, Jan 2011.

[70] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893 vol. 1, June 2005.

[71] V. Petrovic and T. Cootes, "Vehicle type recognition with match refinement," in *17th IEEE International Conference on Pattern Recognition*, vol. 3, pp. 95–98 Vol.3, Aug 2004.

[72] D. T. Munroe and M. G. Madden, "Multi-Class and Single-Class Classification Approaches to Vehicle Model Recognition from Images," in *16th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 93–104, Sep 2005.

[73] X. Clady, P. Negri, M. Milgram, and R. Poulenard, "Multi-class vehicle type recognition system," in *Artificial Neural Networks in Pattern Recognition* (L. Prevost, S. Marinai, and F. Schwenker, eds.), vol. 5064 of *Lecture Notes in Computer Science*, pp. 228–239, Springer Berlin Heidelberg, 2008.

[74] P. Salembier and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[75] D. K. Park, Y. S. Jeon, and C. S. Won, "Efficient Use of Local Edge Histogram Descriptor," in *Proceedings of the 2000 ACM Workshops on Multimedia*, MULTIMEDIA '00, (New York, NY, USA), pp. 51–54, ACM, 2000.

[76] H. Lee, "Neural Network Approach to Identify Model of Vehicles," in *Advances in Neural Networks - ISNN 2006* (J. Wang, Z. Yi, J. Zurada, B.-L. Lu, and H. Yin, eds.), vol. 3973 of *Lecture Notes in Computer Science*, pp. 66–72, Springer Berlin Heidelberg, 2006.

[77] I. Zafar, E. A. Edirisinghe, S. Acar, and H. E. Bez, "Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition," *Proc. SPIE, International Conference on Real-Time Image Processing*, vol. 6496, pp. 649602–649602–8, 2007.

[78] I. Zafar, E. A. Edirisinghe, and B. S. Acar, "Localized contourlet features in vehicle make and model recognition," in *Proc. SPIE 7251, Image Processing: Machine Vision Applications II, 725105*, vol. 7251, pp. 725105–725105–9, 2009.

[79] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[80] U. Sinha, "SIFT: Generating a feature." Last accessed on 15-August-2015.

[81] A. Mammeri, A. Boukerche, and M. Zhao, "Keypoint-based binocular distance measurement for pedestrian detection system," in *Proceedings of the Fourth ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '14, (New York, NY, USA), pp. 9–15, ACM, 2014.

[82] "Public:surf [juergen's work wiki]." Last accessed on 25-July-2015.

[83] A. Mammeri, D. Zhou, A. Boukerche, and M. Almulla, "An efficient animal detection system for smart cars using cascaded classifiers," in *Communications (ICC), 2014 IEEE International Conference on*, pp. 1854–1859, June 2014.

[84] A. Mammeri, T. Zuo, and A. Boukerche, "Extending the Detection Range of Vision-based Driver Assistance Systems Application to Pedestrian Protection System," in *2014 IEEE Global Communications Conference (GLOBECOM)*, pp. 1358–1363, Dec 2014.

[85] A. Mammeri, A. Boukerche, J. Feng, and R. Wang, "North-american speed limit sign detection and recognition for smart cars," in *2013 IEEE 38th Conference on Local Computer Networks Workshops (LCN Workshops)*, pp. 154–161, Oct 2013.

[86] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Li, "Learning multi-scale block local binary patterns for face recognition," in *Advances in Biometrics* (S.-W. Lee and S. Li, eds.), vol. 4642 of *Lecture Notes in Computer Science*, pp. 828–837, Springer Berlin Heidelberg, 2007.

[87] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 971–987, Jul 2002.

[88] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3360–3367, June 2010.

[89] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3304–3311, June 2010.

[90] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 1704–1716, Sept 2012.

[91] X. Zhou, K. Yu, T. Zhang, and T. Huang, "Image classification using super-vector coding of local image descriptors," in *Computer Vision ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6315 of *Lecture Notes in Computer Science*, pp. 141–154, Springer Berlin Heidelberg, 2010.

[92] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, (Cambridge, MA, USA), pp. 487–493, MIT Press, 1999.

[93] F. Perronnin, J. Snchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6314 of *Lecture Notes in Computer Science*, pp. 143–156, Springer Berlin Heidelberg, 2010.

[94] Y. Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. II–506–II–513 Vol.2, June 2004.

[95] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, Oct 2003.

[96] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22, 2004.

[97] L. Nanni and A. Lumini, "Heterogeneous Bag-of-Features for Object/Scene Recognition," *Applied Soft Computing*, vol. 13, no. 4, pp. 2171 – 2178, 2013.

[98] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 923–930, June 2013.

[99] F. Shi, E. Petriu, and R. Laganiere, "Sampling strategies for real-time action recognition," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 2595–2602, June 2013.

[100] P. Pinto, A. Tome, and V. Santos, "Visual detection of vehicles using a bag-of-features approach," in *13th International Conference on Autonomous Robot Systems (Robotica)*, pp. 1–4, April 2013.

[101] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2169–2178, 2006.

[102] S. Singh, S. Choudhury, K. Vishal, and C. Jawahar, "Currency Recognition on Mobile Phones," in *22nd International Conference on Pattern Recognition (ICPR)*, pp. 2661–2666, Aug 2014.

[103] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1027–1035, 2007.

[104] L. Hazelhoff, I. Creusen, and P. de With, "Optimal Performance-Efficiency Trade-off for Bag of Words Classification of Road Signs," in *22nd International Conference on Pattern Recognition (ICPR)*, pp. 2996–3001, Aug 2014.

[105] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards Optimal Bag-of-features for Object Categorization and Semantic Video Retrieval," in *6th ACM International Conference on Image and Video Retrieval (CIVR)*, pp. 494–501, ACM, 2007.

[106] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, June 1998.

[107] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[108] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[109] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[110] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, pp. 1207–1245, May 2000.

[111] J. Weston, "Support Vector Machine (and Statistical Learning Theory) Tutorial." Last accessed on 5-August-2015.

[112] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learn. Res.*, vol. 2, pp. 299–312, Mar. 2002.

[113] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[114] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 278–282 vol.1, Aug 1995.

[115] A. Bosch, A. Zisserman, and X. Muoz, "Image classification using random forests and ferns," in *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007.*, pp. 1–8, Oct 2007.

[116] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Comput. Vis. Image Underst.*, vol. 106, pp. 59–70, Apr. 2007.

[117] Griffin, G. Holub, and P. A. Perona, "The Caltech 256," tech. rep., Caltech, 2006.

[118] F. Zaklouta and B. Stanciulescu, "Real-time traffic-sign recognition using tree classifiers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1507–1514, Dec 2012.

[119] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, pp. 1031–1044, June 2010.

[120] L. Wang, H. Wu, and C. Pan, "Manifold regularized local sparse representation for face recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, pp. 651–659, April 2015.

[121] J. Chen and Z. Yi, "Sparse representation for face recognition by discriminative low-rank matrix recovery ," *Journal of Visual Communication and Image Representation* , vol. 25, no. 5, pp. 763 – 773, 2014.

[122] J. Wang, C. Lu, M. Wang, P. Li, S. Yan, and X. Hu, "Robust face recognition via adaptive sparse representation," *IEEE Transactions on Cybernetics*, vol. 44, pp. 2368–2378, Dec 2014.

[123] S. Shekhar, V. Patel, N. Nasrabadi, and R. Chellappa, "Joint sparse representation for robust multimodal biometrics recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 113–126, Jan 2014.

[124] V. M. Patel and R. Chellappa, "Sparse Representation-based Object Recognition," in *Sparse Representations and Compressive Sensing for Imaging and Vision*, SpringerBriefs in Electrical and Computer Engineering, pp. 63–84, Springer New York, 2013.

[125] H.-F. Chiang, J.-W. Hsieh, C.-H. Chuang, K.-T. Chuang, and Y. Yan, "Modeling and recognizing action contexts in persons using sparse representation," *Journal of Visual Communication and Image Representation*, vol. 30, pp. 252 – 265, 2015.

[126] X. Zhu, J. Liu, J. Wang, C. Li, and H. Lu, "Sparse representation for robust abnormality detection in crowded scenes," *Pattern Recognition*, vol. 47, no. 5, pp. 1791 – 1799, 2014.

[127] S. H. Lee, K. Kostas Plataniotis, and Y. M. Ro, "Intra-Class Variation Reduction Using Training Expression Images for Sparse Representation Based Facial Expression Recognition," *IEEE Transactions on Affective Computing*, vol. 5, pp. 340–351, July 2014.

[128] K. B. Raja, R. Raghavendra, V. K. Vemuri, and C. Busch, "Smartphone based visible iris recognition using deep sparse filtering," *Pattern Recognition Letters*, vol. 57, pp. 33 – 42, 2015.

[129] M. Aharon, M. Elad, and A. Bruckstein, " K -SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311–4322, Nov 2006.

[130] S. Li and A. Jain, eds., *Handbook of Face Recognition*. Springer-Verlag New York, 2005.

[131] R. Bryll, R. Gutierrez-Osuna, and F. Quek, "Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets," *Pattern Recognition*, vol. 36, no. 6, pp. 1291 – 1302, 2003.

[132] N. Chawla and K. Bowyer, "Random subspaces and subsampling for 2-d face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 582–589 vol. 2, June 2005.

[133] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1088–1099, July 2006.

[134] "The Industrial Liaison Program of VBIE." Last accessed on 18-November-2014.

[135] "NTOU-MMR Dataset." Last accessed on 18-November-2014.

[136] L. Hazelhoff, I. Creusen, D. van de Wouw, and P. H. N. de With, "Large-scale classification of traffic signs under real-world conditions," in *Proc. SPIE 8304, Multimedia on Mobile Devices 2012; and Multimedia Content Access: Algorithms and Systems VI, 83040W*, vol. 8304, pp. 83040W–83040W–10, 2012.

[137] "Support Vector Machines Implementation." Last accessed on 3-March-2015.

[138] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[139] S. Sivaraman and M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1773–1795, Dec 2013.

[140] M. Cheon, W. Lee, C. Yoon, and M. Park, "Vision-based vehicle detection system with consideration of the detecting location," Sept 2012.

[141] A. Mammeri, E.-H. Khiari, and A. Boukerche, "Road-Sign Text Recognition Architecture for Intelligent Transportation Systems," in *80th IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–5, Sept 2014.

[142] A. Al-Ghaili, S. Mashohor, A. Ramli, and A. Ismail, "Vertical-edge-based car-license-plate detection method," *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 26–38, Jan 2013.

[143] B. Tian, Y. Li, B. Li, and D. Wen, "Rear-View Vehicle Detection and Tracking by Combining Multiple Parts for Complex Urban Surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 597–606, April 2014.

[144] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, "Application-oriented license plate recognition," *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 552–561, Feb 2013.

[145] M. Sarfraz, A. Shahzad, M. Elahi, M. Fraz, I. Zafar, and E. Edirisinghe, "Real-time automatic license plate recognition for cctv forensic applications," *Journal of Real-Time Image Processing*, vol. 8, no. 3, pp. 285–295, 2013.

[146] "CCTV Camera Pros." Last accessed on 5-March-2015.

[147] "Security Camera Warehouse (SCW)." Last accessed on 15-August-2015.