

Video Genre Classification using Compressed Domain Visual Features

by

Warwick J. Gillespie, B.E. (Hons.)

School of Engineering

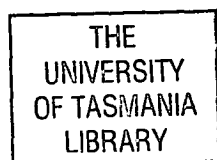
Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

University of Tasmania



November, 2006.

Science
Thesis
GILLESPIE
PhD
2006



Statement of Originality

This thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person, except when due reference is made in the text.

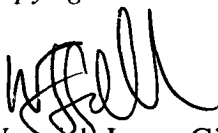


Warwick James Gillespie

November, 2006

Statement of Authority of Access

This thesis may be made available for loan and limited copying in accordance with the *Copyright Act 1968*.



Warwick James Gillespie

November, 2006

Abstract

With the rapid growth in the prevalence of digital video in the world comes the need for efficient and effective management of such information. The field of content-based video indexing and retrieval aims to achieve this through the automatic recognition of the structure and content of video data and the indexing of both low level formative features and high level semantic features. Two of the main problems facing this field are what low level features can be used, and how to ‘bridge the semantic gap’ between low level features and high level understanding of a video sequence. In this thesis we propose a new method that can successfully automatically classify video shots into broadly defined video genres. The classification serves as the first step in the indexing of a video sequence and its consequent retrieval from a large database by partitioning the database into more manageable sub-units according to genre, e.g. sport, drama, scenery, news reading.

As the transmission and storage of digital video is commonly in a compressed format (e.g. the MPEG-1, -2, and -4 standards), it is therefore efficient for any processing to occur in the compressed domain. In this thesis video files compressed in the MPEG-1 format are considered, although the majority of methods presented can be easily adapted for use with MPEG-2 and MPEG-4 formats. For indexing purposes an MPEG-1 file contains spatial information in DCT coefficients and temporal information using motion vectors. The reliability of the MPEG motion vectors is evaluated using a spatial block activity factor estimated from DCT coefficients, to discard the vectors which do not represent the true motion within a video sequence. The thesis also presents a robust camera motion estimation technique, based on Least Median-of-Squares regression, to minimise the influence of the outliers due to object motion and wrongly predicted motion vectors. The results produced by the proposed technique show a significant improvement in the sensitivity to object motion when compared to those produced by an M-estimator technique. Robust motion intensity metrics are also presented for camera and object motion, calculated from the estimated camera model and the MPEG motion

vector field after the filtering of unreliable vectors. A novel metric based on the activity factor used in the motion vector field filtering called activity power flow is introduced to effectively capture the spatio-temporal evolution of scenes through a video shot. These shot-based, low-level, global features represent both the spatial content of a shot, and the motion in a shot, both due to movement of the camera, and also of objects.

The thesis also compares several machine learning techniques to transform low level visual features into high level semantics, in particular Radial Basis Function (RBF) networks with a focus on a tree-based RBF network. In this network, the result of a binary classification tree is used to configure and to initialise the structure of the RBF network. Video shots in a database are classified into four video genres: Sport, News, Scenery, and Drama. This is believed to be the first shot based video classification algorithm and the first method which uses only compressed domain features. Experimental results show that this method is both efficient, as processing is undertaken in the compressed domain, and effective, providing a classification accuracy which is comparable, where possible, to previous techniques. For the genre set {sport, news, cartoon, commercial, music} the best classification accuracies seen in previous works are 83.1% [131] using just visual features and 87% [133] using combined audio and visual features compared with a classification accuracy of 83.6% presented in this thesis. For the genre set {sport, news, cartoon, drama, music} previous work [134] reported classification accuracies of 72.0% for visual features and 88.8% using combined audio and visual features compared with a classification accuracy of 86.2 % in this thesis.

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor Professor Thong Nguyen for his valuable guidance, encouragement and support. His advice and his experience have greatly benefited me academically as well as personally. His willingness and motivation has always provided me with confidence in my research work. I would especially like to thank him for his commitment to my work after his retirement from the University of Tasmania.

I am also grateful to my co-supervisor Associate Professor Michael Negnevitsky for his contributions in the latter stages of my candidature. I also gratefully acknowledge all the staff of the School of Engineering at the University of Tasmania, for their kindness and assistance and many thanks go to the technical staff of the School for their help and support for my research facilities especially Mr. Russell Twining for his assistance with the many computer related problems faced. I am also grateful to all my fellow postgrad students for their help and companionship, particularly Cameron Potter, Alan Henderson and Shao Ng.

I would also like to thank my parents, friends, and family, for their continued and unquestioning support. Recognition of the great work of Peter Degraes must also be noted. Finally, I would like also to express my love and appreciation to Liz for her love and patience and for helping me escape whenever I most needed it.

Contents

CHAPTER 1	INTRODUCTION	1
1.1	Objectives	3
1.2	Significant contributions	3
1.3	Structure of thesis	5
1.4	Supporting publications	7
CHAPTER 2	CONTENT-BASED VIDEO INDEXING AND RETRIEVAL	9
2.1	Syntax of video structure	10
2.2	Video segmentation	14
2.2.1	Shot detection	14
2.2.2	Shot representation	16
2.2.3	Scene / Logical Story Unit segmentation	18
2.3	Video management	19
2.3.1	Video indexing	21
2.3.2	Video retrieval	23
2.3.3	Video representation	26
2.4	Current systems	27
2.5	Conclusion	30

CHAPTER 3 SEMANTIC VIDEO PROCESSING	31
3.1 Genre specific processing	33
3.1.1 Logical Story Units.....	34
3.1.2 Events	38
3.1.3 Settings, objects, and people.....	40
3.2 High level genre classification.....	43
3.3 Conclusion	49
 CHAPTER 4 COMPRESSED DOMAIN VIDEO ANALYSIS.....	 51
4.1 MPEG-1 bit stream	54
4.1.1 GOP frame sequence	57
4.1.2 Spatial redundancy compression (Intra-coding)	59
4.1.3 Temporal redundancy compression (Inter-coding)	62
4.2 Possible visual features from MPEG domain.....	66
4.2.1 Frame sequence and block type distribution analysis	66
4.2.2 DCT coefficients.....	68
4.2.3 Motion vector field	69
4.3 Motion vector field filtering.....	70
4.3.1 Regions of unreliable motion vectors in BMA technique.....	70
4.3.2 Median filtering approach.....	71
4.3.3 Activity threshold approach.....	72
4.4 Conclusion	78
 CHAPTER 5 LOW-LEVEL VISUAL FEATURES	 79
5.1 Overview of visual features.....	80
5.2 Camera and object motion analysis	82

5.2.1 Camera motion models	82
5.2.2 Model estimation techniques	85
5.2.3 Comparison of model estimation techniques	93
5.2.4 Camera and object motion metrics.....	95
5.3 Activity power flow.....	98
5.4 Conclusion.....	102
 CHAPTER 6 HIGH-LEVEL SEMANTIC CLASSIFICATION	 103
6.1 Statistical classification approaches.....	105
6.1.1 Bayes decision theory	105
6.1.2 Discriminant functions and decision boundaries	106
6.1.3 Probability density estimation.....	107
6.1.4 Gaussian mixture models.....	108
6.1.5 Classification and regression trees.....	110
6.2 Artificial neural networks.....	115
6.2.1 Multi-layer perceptron networks.....	116
6.2.2 Training the network.....	117
6.2.3 Adaptive learning rates	119
6.2.4 Pre-conditioning of network	121
6.3 Radial basis function (RBF) networks.....	122
6.3.1 Conventional RBF networks.....	122
6.3.2 RBF Networks for classification.....	125
6.3.3 Training of output connection weights	126
6.3.4 Hidden layer basis function optimisation.....	127
6.3.5 Generalised RBF network.....	134
6.3.6 Comparison of RBF networks with MLP networks.....	136
6.4 Conclusion.....	137

CHAPTER 7 VIDEO GENRE CLASSIFICATION RESULTS	140
7.1 Experimental set-up and procedures	140
7.2 Classification results.....	142
7.2.1 Conventional RBF network initialised with <i>K</i> -means.....	142
7.2.2 Conventional RBF network initialised with tree.....	146
7.2.3 Conventional RBF network optimised with gradient descent.....	158
7.2.4 Generalised RBF network with input feature weights	162
7.3 Comparison of techniques.....	165
7.4 Comparisons with other work	167
 CHAPTER 8 CONCLUSIONS AND FURTHER RESEARCH	 171
8.1 Conclusions	171
8.2 Further Work.....	174
 REFERENCES	 177

Chapter 1

Introduction

As a result of the recent advances in computing and telecommunications technologies, society today is beginning to embrace the 'digital multimedia information age'. The amount of digital information produced, be it text/audio/images/video, is ever increasing and so is the need for technologies to store, search, access and retrieve such information effectively and efficiently. The focus of this thesis is digital video, and as an indication of the amount of digital video being produced today, if a video stream is encoded at approximately 20 Mb/s, then a 1 hour long video sequence will require approximately 8.8 GB of storage. If the amount of video material being produced just in the domain of broadcast/entertainment is considered (i.e. from sources such as television shows, internet video streaming, movies, music videos etc) it is obvious that the volume of information can quickly becoming overwhelming so there is a need for techniques to effectively and efficiently access this information. This is further compounded if domains such as educational surveillance video are also considered.

Until recently, the management of a video database was only possible using a text-based search of either filenames or a manually entered keyword index. This is not a satisfactory solution however, because the manual creation of a set of keywords for a large collections of video information is too time consuming for practical applications. The accuracy of keyword descriptions of a video sequence will also be inconsistent from person to person as each viewer may consider different elements important. The keyword approach may also be inaccurate because commonly keyword information will be domain specific so keywords relevant for one application may be irrelevant for another and vice versa.

In order to overcome these problems there has recently been a substantial amount of research in the field of content-based video indexing and retrieval. The goal of this

research field is to automate the video database population process to provide efficient browsing or searching and retrieval of relevant video material. Commonly this is achieved by assigning content-based labels to video documents through the analysis of low-level audio-visual signal patterns. These content-based labels can then be used to aid in the search and retrieval of videos from a database. There have been many different approaches in literature motivated by many different applications such as: program selection from a video on demand system, management of large digital libraries or broadcasting/production archives, video recommendation system to suggest video sequences according to a user profile, management of education resources, and video-conferencing. In general the annotation of video material with content based labels can be broken into three components: the segmentation of a video sequence into suitable units, the determination of a set of appropriate labels for the application, and finally determining a technique to calculate a value for the index. Each component contains a number of levels of abstraction and the optimal set of components is application dependent. The segmentation can occur at a number of temporal scales (e.g. labels may be determined at the frame, shot, scene or clip level) and spatial scales (e.g. labels may be determined for a whole frame or for a small region or object within a frame). The set of appropriate labels range from general low-level formative features such as simply determining the distributions of different colours within a frame, to more specific indexes containing high-level semantic information such as detecting and recognizing specific objects (e.g. a human face) within a sequence. The indexes may be calculated from visual, audio, or text signals within a video sequence and may be a continuous metric or a classification into one of a number of discrete values or classes. A general trend seen in the literature is that attempts to detect, recognise or classify more specific semantic content require the use of more complex features which are only practical for a limited video domain. This gives rise to a multi-resolution approach which first performs semantic classification at the highest levels of genre and/or sub-genre before using the more complex features only on video sequences from the appropriate genre. In this thesis, the first step in this process of semantic labelling - video genre classification, is investigated in particular for the genres {sport, news, scenery, drama, cartoon, music, commercial}.

1.1 Objectives

The main objectives of the work in this thesis focus on the problem of video genre classification. The main factors which affect the classification accuracy are the set of features used, the classifier used, and the experimental set of genres and data used. There are some current techniques in the literature that tackle this problem, some of which use visual features to describe segments of video, some use audio features and some use a combination of the two. In this thesis only visual features are considered because while audio can provide some useful information, the visual signal is a much richer information source. In general the approaches seen in the literature use visual features calculated from uncompressed video data, attempt the genre classification on video segments of a set length (usually between 20-60s), and only consider a single simple classification network.

The first objective of this thesis is to determine visual features suitable for video genre classification which can be calculated using compressed domain information. This is because as the storage of any digital video will be in a compressed format, it makes sense that the calculation of any visual features should be performed using information extracted directly from the compressed domain to reduce the computational complexity. The second objective of this thesis is to perform genre classification at the shot level. This is because in the case where a video sequence may contain a number of segments from different genres, the boundary between two genre segments is also at the shot level. As a result, if the segment of video used in genre classification is a shot, this will ensure each segment classified will contain a single genre. The final objective is to perform a detailed analysis of a number of classification techniques so as to find the most robust approach for the problem of video genre classification.

1.2 Significant contributions

The contribution of the work in this thesis can be divided into two components: extraction of compressed domain visual features from MPEG-1 files for CBVIR, and the analysis of a number of classifiers, with a particular focus on a tree-based radial basis function (RBF) networks, to the problem of video genre classification.

In the extraction of compressed domain information from MPEG-1 files the reliability of motion vectors is measured using a spatial activity metric calculated from discrete cosine transform (DCT) coefficients so unreliable vectors can be detected and discarded from further processing. A robust estimation of a parametric camera motion model from the filtered motion vectors using a least median of squares (LMedS) techniques is also proposed. From this estimated camera motion model, four features describing the distribution of the intensity of camera and object motion across a shot are proposed. Finally the spatial activity function used to measure the reliability of motion vectors is used to give a coarse measure of the spatial content of a video shot. To achieve this, the evolution of the spatial activity for blocks containing reliable, unreliable, and no motion vectors across a shot is described using two features per block type.

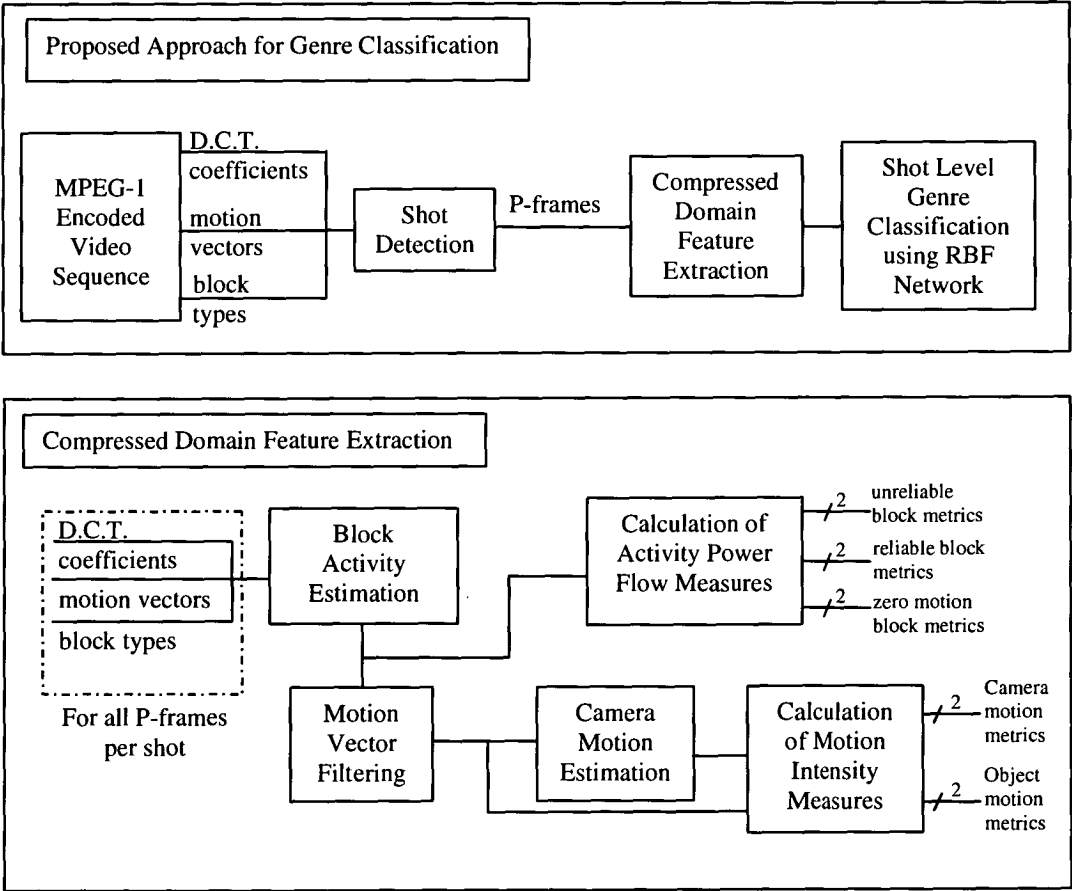


Figure 1.1: Overview of proposed approached to shot level video genre classification using compressed domain visual features.

The thesis also presents what is believed to be the first video genre classification results at a shot level as opposed to approaches seen in previous works which attempt genre classification for short video segments, from 20-60 seconds in length, which contain a number of shots. These results are based on the analysis of a number of classification schemes including a detailed analysis of the optimisation of a number of parameter values for each classifier. The classification accuracy for the best performing classifier, a tree-based radial basis function optimised with a gradient descent process, produces genre classification performance that compares favourably with previous video genre classification works.

1.3 Structure of thesis

Chapter 2 of the thesis provides an overview of the current research into content-based video indexing and retrieval (CBVIR). As the primary goal of any CBVIR system is to automatically understand the structure of the content in a video sequence, a standard hierarchical video structure from the literature that encompasses the majority of CBVIR approaches is given. This includes a description of a standard hierarchical video structure seen in the literature which is divided into three information perspectives: semantic, content and layout. Each perspective has its own hierarchical structure and subsequent techniques for video segmentation and the three components for video management: indexing, retrieval, and representation. A particular focus is given to low-level processing and the layout and content perspectives as the majority of initial research was concerned with these areas. A brief discussion of some of the current system implementations is also given.

In Chapter 3 an overview of current approaches for semantic processing at all levels is given. These include semantic segmentation of a video sequence into logical story units, the detection of semantic events, or at the lowest level the semantic detection and consequent labeling of settings, objects or people. From this discussion it is evident that the majority of current semantic processing approaches are designed for operation on a specific genre. As such, there is a need for a suitable technique for first classifying

video at a genre level and an overview of the current approaches to video genre classification and their limitations is given.

One of the objectives of the work in this thesis is to perform video genre classification using compressed domain features, so in Chapter 4 an introduction to compressed domain video processing is given. In particular, in this thesis the focus is on video sequences encoded in the MPEG-1 format so an overview of the compression techniques used in the MPEG-1 standard is given. The compressed domain information present in an MPEG-1 file that can be used in the extraction of visual features is also discussed, including the distribution of frame and blocks, discrete cosine transform (DCT) coefficients, and motion vectors. Finally, as the motion vectors in an MPEG-1 file can be unreliable if they are used to model the motion in a video sequence, a filtering technique is proposed to detect and discard unreliable motion vectors.

In Chapter 5 the low level visual features used for video genre classification are explained in detail. First, an overview of the visual features used in other approaches is given which shows the need for features that describe the global trends of visual properties such as motion, colour and texture across a number of frames. The features proposed in this work calculated over the length of a shot and include four metrics which describe the camera and object motion, and six metrics which describe the evolution of spatial activity. An analysis of the variation of each of the features with respect to a number of different video genres {sport, news, scenery, drama, cartoon, music, commercial} is also given.

Contained in Chapter 6 is a detailed analysis of a number of pattern recognition techniques which can be used to transform the low-level visual features into high-level genre classification. One possible approach is the use of a statistical based classifier and in particular an analysis of two statistical approaches) used by other authors for video genre classification, namely classification trees and Gaussian mixture models, is given. Another approach to the classification problem, which can be considered as an extension to the statistical approach, is the artificial neural network. This type of classifier is introduced using the commonly used multi-layer perceptron (MLP) trained with back-propagation, but a particular focus is on the use of radial basis function (RBF) networks

for classification. A number of different initialisation techniques for RBF networks are considered, including *K*-means clustering and tree-based initialisation, in addition to gradient descent optimization of RBF networks, and a generalised RBF network structure which includes feature weights for each of the input dimensions.

In Chapter 7, a detailed analysis of the classification results for a number of RBF network classifiers is presented. This includes an analysis of the dependence of the classification accuracy of each network on the variable network parameters. The initial analysis is performed on the genre sets {sport, news, scenery, drama} and {sport, news, scenery, drama, cartoon} to also gauge the dependence on the genre set. In order to better compare the results of the best approach used in this thesis with other works in the literature, further tests were conducted using the genre sets {sport, news, drama, cartoon, music} and {sport, news, cartoon, music, commercial}. Finally in Chapter 8 the conclusions from the work and possible directions for further research are presented.

1.4 Supporting publications

The work in this thesis has been published in the following papers:

D.T. Nguyen and W. Gillespie, "A Video Retrieval System Based on Compressed Data from MPEG Files," *IEEE Region 10 Conference - TENCON 2003*, vol. 2, pp. 555- 560, Bangalore, India, October, 2003.

W. Gillespie and D.T. Nguyen, "Filtering of MPEG Motion Vector Fields for use in Motion-Based Video Indexing and Retrieval," *International Symposium on Digital Signal Processing for Communication Systems*, pp. 8-11, Gold Coast, Australia, 2003.

W.J. Gillespie and D.T. Nguyen, "Robust Estimation of Camera Motion in MPEG Domain," *IEEE Region 10 Conference - TENCON 2004*, vol. 1, pp. 395-398, Chiang Mai, Thailand, 2004.

W.J. Gillespie and D.T. Nguyen, "Classification of Video Shots using Activity Power Flow," *IEEE Consumer Communications and Networking Conference*, pp. 336-340, Las Vegas, USA, 2004.

W.J. Gillespie and D.T. Nguyen, "Classification of Video Sequences in MPEG Domain," *Multimedia Systems and Applications*, vol. 27, pp. 71-86, Springer, New York, 2004.

W.J. Gillespie and D.T. Nguyen, "Video classification using a tree-based RBF network," *IEEE International Conference on Image Processing*, vol. 3, pp. 465-468, Genova, Italy, 2005.

Chapter 2

Content-based Video Indexing and Retrieval

The field of content-based indexing and retrieval of images and videos is fast growing and extremely relevant in today's age of digital multimedia. The reason for this research effort is due to the rapid rise in the amount of digital images and videos being stored, especially in large databases. The aim of content-based indexing and retrieval systems is to provide a quick and easy mechanism of indexing and storing visual content that will allow for easy searching and retrieval by potential users of the content. The users of such systems may come from varying domains, including people browsing web content, media or advertising agencies managing their visual data assets, or more specific applications such as face or finger print recognition to aid police [1]. With such a broad range of applications, the scope of a system (i.e. what kind of querying it is designed to handle and the domain of the video source) is very important. In this work the use of content-based indexing and retrieval systems for the management of generic broadcast video material, (i.e. video commonly found transmitted via TV, DVD, or VHS) is investigated for use in a web browsing or video on demand type environment.

Currently, most large video databases can only be searched either manually (a laborious task) or through the use of manually annotated text descriptions or filename searches. While this process can be suitable in some situations (such as small personal video collections) there are many downfalls to this method. In the case of text descriptions, there is a need for the description of every video to be entered manually by a user, resulting in a costly process for building the database (known as the population process). This is especially unfavourable for large databases that exist with no indexing. Another drawback is the fact that human influence on the population process will produce

inconsistencies in the indexing and searching process as descriptions of the same video by two different people may be vastly different. To take full advantage of the ever increasing multimedia technologies, there is a need for systems that can be populated automatically and searched efficiently and effectively, which has given rise to recent research efforts into content-based indexing and retrieval systems. These systems aim to populate multimedia (image, sound and/or video) databases automatically with an index of various features based on the content of the multimedia file.

Research in the field of content-based video indexing and retrieval has already developed to an extent where a common system structure and standard terminology has been adopted. The majority of works can be represented by a similar syntax of video structure, which is described in Section 2.1. This structure is hierarchical in nature to suit the many different applications which are designed at varying levels of abstraction in terms of information used within a system. One such variation is in the temporal scale at which video analysis occurs. While some systems may use features to index a video at a frame level or even a video sequence as a whole, but commonly a video needs to be segmented into small manageable units which are made up of a number of frames, which is considered in Section 2.2. The segmented units of video are the core elements of a video management system, and the processing of these units to achieve the goals of any system is considered in Section 2.3. A video management system is divided into three main components: indexing, retrieval, and representation. The indexing component is the techniques and features used to label the video units in a database, the retrieval component is the process by which a user queries the system, and the representation component is the method by which the query results are displayed to the user. Finally an overview of a number of significant video management systems currently available or in development is presented in Section 2.4.

2.1 Syntax of video structure

The primary goal of a CBVIR system is to automatically uncover the structure present in a video sequence. Commonly the structure of a video sequence is separated into formative and cognitive information. Formative information is generated using signal

processing techniques performed directly on low level signals resulting in low-level features representing properties such as colour, shape, or patterns in a sequence. In video processing this formative information is presented via three information channels, being visual, aural and textual, although the textual channel can be considered to be derived from the visual and aural channels via optical character recognition (OCR) and automatic speech recognition (ASR) techniques [2]. Cognitive information relates to the human understanding or interpretation of a sequence in terms of elements including objects, people, specific events, meaning, and emotion or mood. The most important factor to consider when describing the structure of a video sequence is the relationship between the formative and cognitive information.

In the literature, two independently derived syntaxes of video structure are proposed by Snoek and Worring [3] and by Roach et al [2,4]. Both are very similar with most differences being relatively minor and for the most part subjective in nature. A combination and adaptation of these syntaxes is shown in Figure 2.1, which defines a video sequence, as in [3], in terms of three perspectives - the semantic index, the content and the layout. This distinction is made in terms of the author of a produced video sequence (i.e. a sequence produced for broadcast via TV, DVD, VHS) as the semantic index contains the story or meaning which the author is trying to portray in the video, and the content and layout are the elements used to convey this information [3]. Each perspective could be used individually as a framework for a particular application in a CBVIR system, but there is considerable overlap between the perspectives and as such usually a combination of all perspectives is necessary.

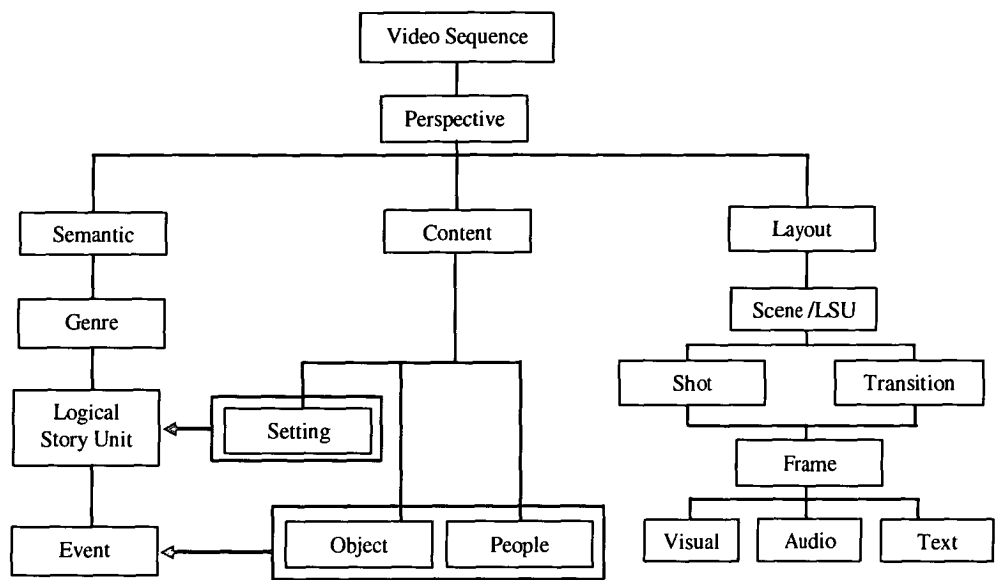


Figure 2.1: A syntax of video structure adapted from Snoek and Worring [3] and Roach *et al* [2,4]

The highest level in the semantic index is the genre, which is the broad class that the video sequence can be described as, and includes instances such as sport, news, feature film etc. Each broad genre can be further broken down into more genres (also known as sub-genres), i.e. the sport genre can be further separated into specific sports such as soccer, athletics, football, baseball etc. In practice a single sub-genre layer for each genre is usually sufficient to model the majority of classes of video sequence, although it is possible for more to exist i.e. a sport such as cycling could be further partitioned into road, track and mountain biking. The lowest level of genre is then segmented into Logical Story Units (LSU's). These contain a single semantic unit in the story and usually contain a single setting. For different genres, the LSU's used can be very different entities, i.e. in a drama sequence an LSU is usually a single scene that tells one segment of the story, usually occurring in a single setting, whereas in a news sequence an LSU could be a single story which contains an introduction by the newsreader followed the report itself. Within each LSU a number of events may occur, either specific to that LSU (e.g. a salient moment in the storyline of a drama) or a common event for a given genre (e.g. a goal in a football game). Further, each event may contain certain people or objects, which similarly can be specific to a given event or common to a given genre.

The content perspective encompasses the elements used to convey the meaning or story in a video sequence, such as setting, objects and people [5]. The setting is the time and place in which a particular dramatic event takes place, and is commonly specific to a LSU within a sequence. Objects are any important static or dynamic entities used to tell the story, a subset of which is people which are significant enough to be considered separately. The interaction of people and objects within a setting depict semantic events, and the appearance of these elements can convey the story, meaning, and mood of a particular LSU through lighting, camera angles and camera movement [3].

The layout perspective (defined in [4] as the editing effects) considers how the content of a video sequence is pieced together. In the visual mode the fundamental unit of a video sequence is a frame, and it is well known in literature that a group of frames recorded continuously from the same camera is called a shot. This is the definition of a visual shot, and similarly an audio shot can be defined as an uninterrupted recording of a sound source, and a text shot can be defined as an uninterrupted textual expression [3]. While only a single visual shot can be presented in a sequence at any one time, a number of audio or text shots, each with a different duration, may occur simultaneously (i.e. in an audio stream speech may overlay music), and it is common that the shots in each channel will overlap temporally (i.e. a single piece of dialogue may run over a number of visual shots). Within the layout perspective there are also transition effects such as cuts, dissolves, wipes, and fades which are used to concatenate one shot to the next. Similar neighbouring shots can be further grouped into Logical Story Units, or scenes, which at a semantic level usually correspond to segments which contain the same location, time or 'dramatic incident'.

In general the layout and content perspective can be viewed as low-level, formative information channels, and the semantic perspective is a high level human understanding of the layout and content perspectives. There is, however, a strong relationship and overlap between these perspectives as there are a number of properties with both formative and cognitive elements to them. These include the dual view of a Logical Story Unit from a semantic and layout perspective, the relationship between a setting and an LSU, and the relationship between objects / people and a semantic event.

2.2 Video segmentation

As the size of a video sequence can be extremely large, the first step in any CBVIR system is to segment a video sequence into smaller more manageable temporal units which can then be processed and indexed individually. Automatic video segmentation is the process of uncovering the layout perspective of the video structure described in Section 2.1. The fundamental unit of the layout perspective is the shot (only the visual shot is considered here), and the detection of visual shots in a video sequence, see Section 2.2.1, is usually the first step in any CBVIR system. Commonly, each shot is then represented by a key frame, which can be used in further processing to describe the contents of a shot, or to be used in a timeline of shot key frames to browse the contents of a video sequence. The choice of the most representative key frame for a shot is discussed in Section 2.2.2. Finally, a group of contiguous shots from a video sequence containing the same content setting, objects, and people can be grouped together into LSU's or scenes, which is described in Section 2.2.3.

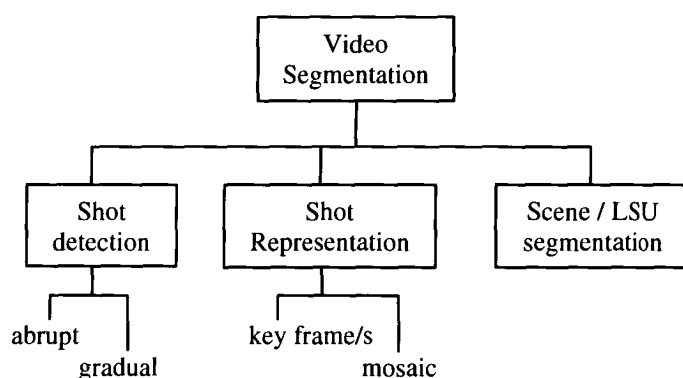


Figure 2.2: Overview of video segmentation techniques

2.2.1 Shot detection

As the segmentation of a video sequence into shots is seen as a fundamental, and commonly the first, step in any CBVIR system, the problem of detecting camera shots, or more precisely the transition boundaries between shots, in a video sequence is well researched. It is in fact a largely solved problem in some domains, with many different solutions proposed in the literature, and comprehensive reviews of shot boundary

detection techniques can be found in [6,7,8,9,10]. The transitions between shots are classified into two types – abrupt and gradual. An abrupt transition, also known as a cut, is an instantaneous transition between two shots, with no intermediate transition frames between the end frame of the current shot and the start frame of the next shot. A gradual transition is one which contains a number of transitional frames between the end of one shot and the start of the next. The transitional frames are some transformation of the final frames of the current shot and the initial frames of the next shot, with the three main transition types being a fade, a dissolve, and a wipe. A fade is a scaling of the pixel intensities over a number of frames either to black in a fade out or from black in a fade in, a dissolve is a mixture of the frames in the two shots by fading from one shot to the next, and in a wipe a specific pattern is used to progressively reveal the new shot and obscure the existing one. Initial shot transition detection attempts focused solely on abrupt transitions as they are simpler to detect than gradual transitions. An abrupt transition will produce a temporal visual discontinuity, and abrupt transition detection techniques generally use some feature to measure the frame to frame visual continuity of a video sequence. Common features used to compare the visual difference of two frames include simple pixel comparison or intensity/colour template matching, block based comparison techniques, a number of different global and local histogram comparison techniques, and edge detection and comparison, with initial works calculating these features from raw uncompressed video signals. As research progressed, compressed domain techniques were used to estimate these features and features specific to the compressed domain, such as bit-rate information and compression prediction statistics, were also developed. Once these features are calculated abrupt transitions are usually detected using either global or adaptive thresholds, clustering, or filtering techniques on the frame difference measures for subsequent frames. For gradual transitions a comparison of subsequent frames is not sufficient so alternate thresholding techniques are used, such as *twin comparison* and *plateau detection*. Twin comparison uses two threshold values and for any sequence of frames in which subsequent frames have a difference value greater than the lower threshold the difference values are summed. A gradual break is declared when the sum value exceeds the higher threshold value. In the plateau detection algorithm the i^{th} frame

is compared with the $i+k^{\text{th}}$ frame, a sequence of frame similarities is computed and significant plateaus are declared gradual transitions.

To measure the performance of different shot segmentation algorithms, two metrics from the field of image retrieval are commonly used: recall and precision. It is also common for these metrics to be used in other detection and classification problems in CBVIR systems. Recall is defined as the ratio of relevant items retrieved to all relevant items in the data set, which in shot detection is the ratio of correct boundary detections to all boundaries present. This requires the determination of a “ground truth” of the exact number of relevant items in the data set, which leads to obvious problems for large databases. Precision is the ratio of relevant items retrieved to all items retrieved in response to a query which in shot detection is the ratio of correctly detected boundaries to all boundaries detected.

$$\begin{aligned} \text{Recall} &= \frac{\text{No. Correct Detections}}{\text{No. Actual Shots}} \\ \text{Precision} &= \frac{\text{No. Correct Detections}}{\text{No. Detected Shots}} \end{aligned} \quad (2.1)$$

While shot transition detection is a well researched problem, it is still being tackled on a large scale today. It is one of a number of tasks in the TREC video retrieval evaluation project (TRECVID), a combined and collaborative effort which aims to promote progress in content-based retrieval from digital video via open, metrics-based evaluation. The TRECVID project, run annually since 2001, provides a new large set of video sequences annotated with shot transition ground truth each year to enable comparison of a large number of transition detection techniques. The most accurate shot detection results reported in the latest TRECVID were about 95% precision and recall for abrupt transitions and about 75% for gradual transitions on a test corpus of video which contained 4,535 shot transitions of which 60.8% were cuts [89].

2.2.2 Shot representation

After video shot segmentation, usually the next step in the analysis process is selection of representative frames from the segmented shots, called key frames. Representing

shots via single key frames has the advantage that established spatial image processing techniques used for content-based image querying can be used in the video domain. Initial key frame selection was as simple as choosing the first, last or some intermediate frame of a shot, however more sophisticated algorithms have since been proposed. One such algorithm uses optical flow techniques to determine the amount of motion in a shot and the key frame is then chosen on the basis of detected local minima in shot motion values [11], while in [46] an average frame colour histogram is calculated for a shot and the frame with a histogram closest to the average is chosen as the key frame.

One disadvantage of using a single key frame to represent a shot is that in the presence of camera and object motion one key frame may not be sufficient to accurately represent the content of a shot. This leads to the selection of a “representative set” of images from each shot to serve as key frames. One technique is simply to uniformly sample the frames in a shot, say 1 per second [12], or similarly uniformly sample the frames across a whole sequence without any form of shot detection. Another technique to select the frames in the representative set is to add a new key frame to the set if a frame is encountered that is sufficiently different from the previous frames selected for the set [13,21,32]. In [14] a clustering-based approach to automatic key frame selection is proposed. Firstly, the average and intersection colour histograms are computed for a particular shot. Each frame in the shot is then compared to these histograms separately then a clustering process returns either a single cluster or a group of clusters with centres no closer than a set merging-threshold. Key frames are then chosen as the sample closest to the centre of each cluster. In [15] a technique is presented which selects a set percentage of the original frames in a shot as key-frames using a clustering process. It is also claimed that this process can be applied to full video sequences without the need for shot detection. Another solution proposed by [16] is to calculate a measure of the ‘perceived motion’ in a shot and select a new key frame at the start and end of segments of a video shot which are deemed to contain high motion.

Rather than a set of representative frames, in [39,17] the creation of a “mosaiced” key frame to convey the content of the shot has been proposed. The process involves transforming all frames within a shot into the coordinate system of a reference frame using an estimation of camera motion. This concept is extended in [18] by first

computing the type of global motion in a shot and based on this computing the most appropriate key frame (e.g. a mosaic image for a pan, the first and last shot in a zoom, and an image of the targeted object in the case of object tracking).

2.2.3 Scene / Logical Story Unit segmentation

While the fundamental unit of a video sequence is a shot, in some circumstances the shot is not a useful unit of segmentation. Consider for example a user wanting to browse the contents of a video sequence via key frames of video shots. In long video sequences, such as feature films, there can be many hundreds or thousands of shots, and as such browsing at a shot level would become impractical. One solution to this is to group shots into semantic segments, known as scenes or logical story units, which can be considered as a number of consecutive video shots related to each other by their semantic content (i.e. containing a specific story or context which in itself may be useful for indexing and retrieval). In general, a scene can be defined by the continuity in the visual contents of shots due to a fixed physical setting or by the continuity of the ongoing actions [19]. The ability to segment a video sequence into scenes as well as shots will provide a further level of abstraction to use in video indexing, which will be especially useful in the semantic processing of video sequences.

Scene detection techniques are not as common in the literature as shot detection techniques, and the majority concentrate on identifying contiguous shots with similar content (i.e. similar settings) with no semantic analysis of ongoing actions. Every shot boundary in a sequence is a possible scene boundary, and actual scene boundaries are detected using a measure of the coherence of shot content between neighbouring shots. Generally, some measure of the similarity between shot on one side of a possible boundary and a number of shots on the other side of the boundary is computed, and the higher the number of similar shots, the higher the content coherence. If the content coherence across a boundary is high, then both sides of the boundary are from the same semantic segment, and across a boundary between two segments the content coherence is low.

A common shot similarity measure used is a histogram intersection comparison of key frames. This process is used in [18] using a computed key frame for each shot depending on the type of motion in the shot (see Section 2.2.2). In [20] a single key frame for each shot is used, with factors such as shot lengths and time difference between the compared shots also included and results are merged with an audio scene detection scheme to improve accuracy. In [21] a number of key frames are used for each shot and the potential scene boundaries detected from shot coherence using colour histograms. This set of potential boundaries is refined using a measure of motion in a scene, as it is argued the visual content in action scenes is not as coherent from shot to shot resulting in a number of false scene boundaries. In [22] shots are first clustered based on an audio classification scheme, then refined using a visual scheme that tracks dominant ‘colour objects’ (which do not always corresponds to actual objects in the frame) from frame to frame within a shot. The most common ‘colour objects’ are then used to determine the correlation between shots and to cluster shots into scenes. In [19] both color and motion information are used to compute shot similarities and similar shots are clustered into scenes using a weighted undirected graph called a shot similarity graph, in which each node represents a shot and the edges between the shots are weighted by their similarities. In [23] shot similarity is recursively computed using a global colour feature calculated from a sub-sampled DC image sequence, allowing the scene segmentation to occur in a single pass.

2.3 Video management

The management of a video database is a complex problem and the techniques used in a management system will depend greatly on the application or service the database is used for. Some of the factors affecting the system include the domain of video stored in the system, the range of users, the type of queries a user might present to the system, and the type of results the users desire. To illustrate this point consider the two very different applications of a database of stock footage for a television station and a digital television on demand system aiming to provide up-to-the-minute coverage of news stories. Typically, a query to the stock footage database will be made in order to provide a news story with footage, so a query to such a database may be ‘find footage containing

a police car’, or ‘find footage of a rainforest’. The result of such queries would ideally be a collection of video shots containing the desired scenes. A typical user query to the news on demand system may be ‘show the latest political headlines’ or ‘find reports on a recent event’ and the returned results should be not single shots but full news reports relating to the user query. As a result of the different user requirements, the design of a video management system needs to take into account how the system will be accessed and searched, the type of queries that can be expected, and whether the user is searching for specific frame/s, shot/s, scene/s or sequence/s.

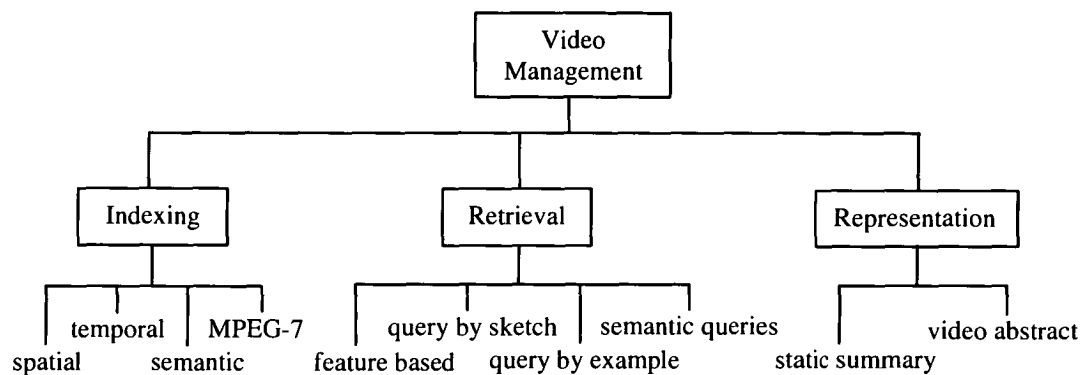


Figure 2.3: Overview of video management techniques

Regardless of the application, the video management process can be divided into three related components – video indexing, video retrieval, and video representation. Video indexing, also referred to in the literature as analysis, feature extraction or content representation, is the process in which video sequences are labeled, sorted, and stored in a database using content-based feature metrics. The goal of video indexing is to be able to describe an information rich video sequence as efficiently as possible and in way that can be effectively searched. Video retrieval encompasses the techniques used to search or query a video database and retrieve video sequences which match the query, and video representation is the form the database search or query results are returned in, such as textual descriptions, key frames or video abstracts. Like the video structure presented in Section 2.1, each component of the video management process can be performed using various levels of abstraction, from low-level formative techniques to higher level semantic techniques. For indexing purposes, formative or cognitive labels can be used, for retrieval formative or cognitive queries and search strategies can be used, and for

browsing a simple formative key frame representation or a cognitive video summary containing semantically important video segments could be used.

2.3.1 Video indexing

The task of video indexing, also known as feature extraction or content representation, is the process of automatically assigning content-based labels to video documents. Indexing is generally done off line to create meta-data to be used in further processing, or to be searched in a retrieval setting, or used to sort a video database for efficiency (i.e. to create clusters of similar frames/shots/scenes/sequences). Due to the broad scope in terms of application for video indexing, there is correspondingly a large variation in indexing techniques. The variation in indexing techniques can be divided into a number of independent aspects, including spatial or temporal features, spatial or temporal scale of the features, uncompressed or compressed domain features, and the semantic level of the features. Spatial indexing techniques are those derived from the field of image retrieval and are based on single video frames (i.e. indexing the key frames of a video shot). Temporal indexing involves extracting features based on the 3-dimensional nature of video shots to describe the variation in time of spatial features. The scale of spatial features can range from global features which index the frame as a whole to local features which index small regions or objects within a frame. Temporal features are also calculated on a number of scales from set time periods, to shot-based indexing, or object-based indexing [24] which attempts to track objects which may appear in a small fraction of a shot or across multiple shots. The semantic level of a feature could range from very low-level formative features such as global statistics describing the distribution of colour within a single frame, to high level cognitive features such as the detection of people, objects or events. Finally the actual computing of features may be performed on the uncompressed raw video signal or directly from a compressed video signal which can produce significant improvements in efficiency, provided the feature matches the characteristics of the compression algorithm.

Initially low-level formative spatial features were adopted from image retrieval applications and used to index key frames of video sequences, with reviews of many of these techniques in [8,25]. These low level features include various measures to

describe both the global and local distribution of various properties such as colour, texture, and shape within a single frame of a video sequence. The added complexity of indexing video sequences compared to images is due to the extra temporal information in video sequences. This leads to the need for indexing based on temporal features, primarily descriptions of the motion present in a sequence such as measures of motion magnitudes, camera and object motion characterisation, and object tracking. The next step seen in current research is looking at video indexing based on semantic features by transforming the low level features into higher level understanding. The TRECVID project contains a high level feature detection task, in which video shots are indexed based on the appearance or non-appearance of high level features such as, in the 2005 experiments [89], people walking or running, explosion or fire, map, US flag, building exterior, waterscape or waterfront, mountain, prisoner, sports, car. Semantic feature extraction is discussed further in Chapter 3.

The MPEG-7 standard is a multimedia content description interface which provides detailed formatting information and descriptions of low-level and high-level features which can be used to index digital multimedia content including images, video and audio. The MPEG-7 visual description scheme [26,27] specifies content-based descriptors based on color, texture, object shape, global motion, or object motion features for indexing both still images and video, based on visual content. An overview of MPEG-7 colour and texture descriptors can be found in [28]. Included in the standard is a generic *Scalable Colour Descriptor* (SCD), a color histogram which uses the HSV colour space uniformly quantised into 255 bins encoded by a Haar transform. The SCD is used either at a frame or group of frames level. There is also a *Dominant Colour Descriptor* which is a more compact description of global and local spatial color distribution than the SCD. Colours in a given region are clustered into a small number of representative colors. The descriptor consists of the percentages representative colors in a region, the spatial coherency of the colour, and the colour variance. A *Colour Layout Descriptor* is used to describe spatial regions for the dominant colour descriptor to be used on. A *Color Structure Descriptor* (CSD) is also used to describe local color features in images by computing a block based color histogram constructed by scanning an image in a sliding window approach. A *Homogenous Texture Descriptor* is used to

describe directionality, coarseness, and regularity of patterns in images. The descriptor is based on a filter bank approach employing scale and orientation sensitive filters and provides a quantitative characterization of textures with homogenous properties. In order to provide descriptions for non-homogenous texture images, MPEG-7 also defines a *Non-Homogenous Texture Descriptor (Edge Histogram)* which captures the spatial distribution of edges. An overview of MPEG-7 shape descriptors can be found in [29] including a *3-D Shape Descriptor* based on the shape spectrum concept, a *2-D/3-D Shape Descriptor* which creates a 3-D description from a number of 2-D viewpoints, a *2-D Region Based Shape Descriptor*, and a *2-D Contour Based Shape Descriptor*.

The motion descriptors in MPEG-7 are defined for either a video segment (i.e. a group of frames) or a moving region/object, and an overview can be found in [30]. For a video segment the *Motion Activity Descriptor* captures the overall activity level or pace of motion or action based on standard deviations of motion vector magnitudes which are quantized into five activity values. Optionally, the direction of motion, spatial distribution of motion activity, and the temporal distribution of motion activity can also be extracted. The movement of a camera, which often mimics the viewers' focus of attention, is described by the *Camera Motion Descriptor*. This descriptor expresses which camera motion types are present in a segment, giving their amplitude, and either their precise time localization in the segment or their relative importance in terms of duration. The global motion can also be described using a *Warping Parameter Descriptor* with reference to a global Mosaic of a video scene. The motion of a moving region/object can be described using local features such as the *Motion Trajectory Descriptor* which describes the coarse displacement using successive positions in time of a representative point for the region such as its center of mass. More detailed information about a region's displacement such as rotations or deformations can be captured by *Parametric Motion* which uses the same motion model and syntax as the *Warping Parameters*.

2.3.2 Video retrieval

Video retrieval is the process of selecting the most relevant video shots, scenes, or sequences from a video database in response to a user query. In general, the basis of

video retrieval is the matching of indexed features discussed above for each video sequence in the database against those of a user query, using some measure of similarity. The result is a list of candidate shots/scenes/sequences ranked according to the matching results. Video indexes are often in the form of feature vectors which can be compared by way of distance metrics i.e. a Euclidean distance. It is, however, important that videos returned as a result of a query must be perceptually close to that query, not just close under a metric [43], and for some features, Euclidian distance measures may not simulate human perception and thus other similarity measures are needed [25]. It is also possible to use the structure of the video indexing technique itself to aid in the retrieval process. To do this, the structure of the database can be chosen so that for a given query only a small percentage of the database needs to be searched to find the best match. One example of this is the use of R*-trees in the QBIC system [39].

Another component in the retrieval process is the type of query a user can make. Current techniques include low-level feature queries, query by example, query by sketch, and high level semantic queries. A low-level feature-based query simply involves the user specifying the low-level features that the required video should exhibit. An example of this is [31] in which video shots are retrieved based on similarity of low level motion characteristics using either a quantitative query of motion magnitude and dominant direction, or a qualitative query specifying the type of motion as pan left/right, zoom in/out, or up/down and the intensity of motion as low or high. Query by Example (QBE) is the process of retrieving videos from a database based on similarities to an image or video provided. Simple video queries by example include ‘find a video shot with a key frame like the given image’, or ‘rank the video clips in order of their similarity with a given video clip’, or ‘cue the given video clip to the frame like the given image’. These types of query can be solved by extracting relevant features from the given image/video segment and compare with the corresponding features from the database. Initial QBE for video were based on spatial image retrieval techniques simply used on a key frame for each shot in the database. The use of video clips as the basis for the query was considered in [32]. Two schemes were proposed: one based on a number of key frames selected for each video shot and one based on sub-sampled frames. The

key frame approach extended the spatial domain techniques to include a motion feature to represent the motion in a shot based on the frames around the key frame.

The theory behind QBE is that assuming the correct match does in fact lie in the database somewhere, then it makes sense to begin the search with an image from the database, and by a series of query by examples the user can be lead to the required image. One drawback of the QBE search method is the problem of which initial example image/video to use. A technique for sketch-based retrieval of images, Query by Sketch (QBS), which can be used as a stand alone search method or as a first step in a QBE system, was first proposed and implemented in the QBIC system [39]. The QBIC system involved the retrieval of images from a database on the basis of similarity to a sketch composed by a user, and the same principle has been applied to retrieval of key frames from video shots in [33]. In most cases the user sketch is comprised of simple shapes which the user can assign a colour and or texture. In a system called VideoQ [43, 44] the QBS is extended for use in video using the concept of an animated sketch to retrieve shots from a video database. This involves the user drawing a shape and an associated trajectory and combining this with feature-based query attributes such as colour or texture.

The next step in content-based video retrieval is to incorporate high level semantic queries into a system. The TRECVID project contains a high level search task in which video search systems are queried using a number of topics and were asked to return a ranked list of up to 1,000 of the most relevant shots from the videos in the search test collection. The topics are designed with the test data set in mind to include a number of basic search types such as generic/specific and person/thing/event where there are multiple shots in the test set coming from more than one video relevant to each topic. In the latest set of experiments [89], 24 high level search topics were used including; 'Find shots of Condoleeza Rice', 'Find shots of people shaking hands', 'Find shots of a helicopter in flight', 'Find shots of George Bush entering or leaving a vehicle, e.g., car, van, airplane, helicopter, etc - he and the vehicle both visible at the same time', and 'Find shots of tennis players on the court – both players visible at same time'. The majority of current techniques rely heavily on automatic speech recognition to achieve these tasks.

2.3.3 Video representation

In a video management system the aim of the video representation component, like the representation of a shot in Section 2.2.2, is to present the results of a user query to a video database in a format that is meaningful, concise, and visually pleasing. The best format of results will vary with the type of user query, whether the user is searching at the shot, frame or scene level, or whether the user is browsing the database and wishes to see an abstraction of a single video sequence, a small segment of the database, or to get an overview of the content of the database in its entirety. Video representation schemes can be broken into two types, static summaries and shortened video abstracts. Static summaries are created by automatically choosing only the most salient images or key frames and efficiently packing them into a pictorial summary. The use of a shortened video abstract, also known as video browsing or skimming, tries to summarise significant highlights of a sequence providing users with the capability to efficiently browse the contents of a video database.

In [34] the task of video representation is divided into two classes based on the hierarchical level of video used. Microscopic browsing is the term given to video browsing at the shot or frame level, most commonly achieved using key-frames arranged in a sequential layout, like a filmstrip, to summarise the content of a video sequence. Microscopic browsing is obviously only efficient for relatively short video sequences as the number of shots and hence key frames in motion pictures, for example, may be more than 1000 shots making a shot or frame level representation unfeasible to get an accurate idea of video content. Higher-level representation of scenes or story units is referred to as macroscopic browsing. Once again a key-frame filmstrip can be used at the scene level but choosing the best key-frame for a single scene is a difficult task, so other useful techniques include a scene transition graph (STG) or a video poster summary [35] and a comic strip approach [36]. An STG is a directed graph modelled with clusters of similar shot as nodes and transitions between shots in time as edges (e.g. in a two person dialogue all shots of one person will be in one cluster and all shots of the second person in another cluster, and a single key-frame is used to represent each cluster). The STG provides the viewer with a compact depiction of the content and temporal flow of a video sequence via the nodes and edges. The video poster approach creates a visual

summary of a sequence using a series of video posters, each representing a scene or story unit arranged in temporal order. The poster consists of one or more sub-images, with each sub-image selected to depict a collection of shots of similar visual content (i.e. a shot cluster as in the STG). The comic book approach uses similar clustering of key frames to find semantic segments in a video sequence. The importance of each segment is then ranked based on the length and rarity of the segment and segments with a high ranking are represented using larger key frames. The different sized key frames are presented in a compact comic book format to give a visually pleasing overview of a video sequence.

A different approach to macroscopic video browsing is the automatic creation of “video abstracts”, which provide a compact summary of an entire video sequence by piecing together relevant shots and audio, much like a film preview. The techniques used for constructing such abstracts usually differ depending on the type of video being abstracted (e.g. documentaries versus feature films) due to different requirements. Concentrating on motion pictures, the authors in [37] segmented the video into “shot clusters” and then attempted to extract special events, such as dialogue, gunfire, explosions and text from title sequences, to construct the abstract. Face detection algorithms were also used to select shots containing the lead actors. Another abstraction procedure, called video skimming, has also been proposed [38]. This process has been applied to news video or documentaries and requires a transcript of the video in question. The video and transcript are aligned through the process of word spotting and the audio track created by detecting important words. The video component is created from detected faces, text and camera operations

2.4 Current systems

A number of systems have been developed to solve the digital video library management problem. The majority of systems have been developed in an experimental framework, although some commercial systems do exist. The following is an overview of some of the more significant projects, including QBIC, WebSEEk, VideoQ, Informedia, Fischlar, Virage and CueVideo.

QBIC (Query by Image Content) [39], developed at the IBM Almaden Research Center, is probably the best-known of all image content retrieval systems. It is available commercially either in standalone form, or as part of other IBM products such as the DB2 Digital Library. QBIC allows queries on large image and video databases based on example images, user constructed sketches, selected colour and texture patterns, camera and object motion, as well as by text keyword. For video data, the database population process involves three main components being shot detection, mosaiced r-frame creation for each shot which are processed as still images for feature extraction, and finally automatic segmentation of independently moving objects from a static scene using a dominant 2D motion estimation method which divides a shot into a number of layers, each with its own 2D affine motion parameters and region of support in each frame.

WebSEEk [40,41,42] is a query by example type system developed by Columbia University for searching for images and videos on the World-Wide Web collected by automated agents using both text-based navigation and content-based technology. The system indexes still images and key frames from video sequences based on a combination of colour, location and a simple text description. Key frames of a sequence are detected by temporally sub-sampling at one frame for every second of video then shot detection is performed to eliminate duplicate shots. Upon retrieval from a query, the key frames appear to the user as animated samples of the original video.

VideoQ [43,44] is a content-based search system developed at Columbia University, designed specifically for video retrieval. The aim of *VideoQ* was to provide automatic video object segmentation and tracking, and query with multiple objects based on features such as color, texture, shape, size and motion. The *VideoQ* system uses colour regions and edge techniques to segment objects in each frame of video, and then uses optical flow techniques to track the objects throughout a shot. A user query to *VideoQ* is performed via an 'animated sketch' where each object drawn in a sketch (like any sketch for a still image system) can be assigned a colour, texture and shape and in addition motion attributes such as the speed and length of the object motion trajectory and the arrival sequence of objects.

One of the largest efforts in video indexing and access is the *Informedia* system [45] developed by Carnegie Mellon University which reportedly has more than 2,000 hours of news and documentary programs indexed in its database. Its overall aim is to allow full content search and retrieval of video by integrating speech and image processing. The system identifies video scenes (not just shots) from analysis of colour histograms, motion vectors, speech and audio soundtracks, and then automatically indexes these 'video paragraphs' according to significant words detected from the soundtrack, text from images and captions, and objects detected within the video clips. Thumbnails of key frames are then displayed with the option to show a sentence describing the content of each shot, extracted from spoken dialogue or captions, or to play back the shot itself.

The *Físchlar* [46] system was developed by Dublin City University for indexing and browsing broadcast TV content via a www interface. *Físchlár-TV* is a video recorder system with management techniques including shot boundary detection, key frame extraction, closed caption analysis, video streaming, XML architecture for video retrieval, and automatic recommendation and personalisation. *Físchlár-News* is an automated video archive of daily TV news in which stories can be searched by keywords, browsed by date/year, or browsed using full detail shot-level images and spoken dialogue. News stories can be followed over time and recommendations of interesting news stories sent to users. The *Físchlár-Nursing* video library allows easy browsing and playback of nursing-related video using a table of contents and shot-level content browsers.

Another commercial system is the *Virage VideoLogger* [47] which creates content-based segments based on scene changes or camera changes and distinct key frames are extracted to create a digital storyboard for visual content indexing. A video segment is indexed using features to describe color, texture, structure, composition motion content and motion uniformity and the system can also recognise faces, voices and types of sounds in the video, identify on-screen text/numbers, and convert spoken words to text. The video metadata can be further enhanced with manual annotations or remote processes.

CueVideo [48] is an IBM research project which uses keywords obtained from speech recognition to search through video data. It is a fully automated video indexing system including speech indexing, scene change detection, and the generation of multiple browsable views. It uses smart video browsing technology, including full video, audio-visual slide shows, a smart fast play which achieves 10-15 times faster playback without missing short events, and producing a visually pleasing video playback which is free of abrupt changes and jitters, and fast audio with natural pitch. The application interface is web-based using embedded streaming video with all streaming modes fully synchronized and supporting multiple input video formats including MPEG, QuickTime, AVI, and H.263.

2.5 Conclusion

In this chapter a common framework of approaches seen in the development of content-based video management systems has been considered. A hierarchical video model which segments a video sequence into the layout, content and semantic perspective to describe the information flow in these approaches has been adopted from the literature. Each perspective contains a number of levels of abstraction to provide a user with effective and efficient access to this information. The common approach is to segment each video sequence into units of a suitable length, such as the scene-shot-frame model from the layout perspective. Each video segment can then be indexed using a number of significant features and an indexed video database can be searched by specifying the desired features or by giving an example video/image as a query. The majority of initial techniques of segmentation, indexing and retrieval presented are performed at a formative level and do not take into account semantic information present in a sequence. While this is appropriate in some applications, the use of the information contained in the semantic perspective is vital to take the next step in content-based video management and will be considered in the next chapter.

Chapter 3

Semantic Video Processing

The majority of initial research in the field of content-based video indexing and retrieval was based on low-level processing of video sequences. As described in Chapter 2 this included segmentation schemes such as shot detection, indexing based on low level features such as colour shape and texture, representation schemes such as key frame browsing, and retrieval schemes such as Query by Sketch and Query by Example. In many possible practical applications though, such as a video on demand system, this low-level processing will not be sufficient to meet all the needs of the user. There is also a need to provide human understanding of a video sequence, at various levels of abstraction. This is commonly referred to in the literature as '*bridging the semantic gap*' between low level formative processing techniques and high level semantic descriptions of video sequences. Much of the current efforts in CBVIR are aimed at semantic video processing and like the formative processing techniques in the previous chapter, semantic techniques can be used in segmentation, indexing, representation, and retrieval problems.

In Chapter 2 the structure of a video sequence was broken into three perspectives, namely layout, content and semantic. In this chapter some techniques seen in literature to uncover the semantic structure will be introduced, as well as some techniques to assign semantic labels to some facets of the content and layout perspective. The structure of the semantic perspective is hierarchical from the highest level of genre, to lower level features such as objects and people which are also seen in the content perspective. This hierarchical semantic partitioning of video sequences can be used to guide techniques for segmentation, indexing, retrieval, and representation to achieve more perceptively accurate results. Video processing at a semantic level is commonly a classification or detection problem, and as the semantic structure contains many levels of

abstraction, so too do the techniques to uncover this structure. At each level of semantic abstraction different features are used to describe the semantic content at that level. In general global feature metrics are used to describe the higher level semantics such as genre, and more specialised local metrics, which are commonly region or object based are used to describe lower level semantics such as logical story units, and events. This is because at the lower level each genre will have events, objects and settings which are unique to that genre so more sophisticated features can be used with these semantic cues in mind, but at a higher level it is not feasible to perform such complex operations across all domains so coarser global features are necessary. Commonly the semantic processing approaches used to detect and label logical story units, events, settings, objects, and people are domain or genre specific, using domain specific knowledge and features. This is because in narrower domain of video more complex features can be used which are tuned to semantic knowledge of the domain. An overview of genre specific semantic processing techniques in the literature is presented in Section 3.1.

It follows that if techniques used for semantic processing at the object or event level are domain specific then there is a need for the partitioning of a video database at a genre and sub-genre level. As each semantic level in each genre may have different segmentation and indexing techniques the problem of genre classification is a necessary first step in any semantic processing scheme. This will allow domain specific processing techniques to be performed only on appropriate video sequences. The first video genre classification attempts were published in mid 1990's and in Section 3.2 a review of many of the subsequent approaches to genre classification is presented, which shows a number of different techniques used on a varied set of genres, using varying datasets, and various features using various signal modes (i.e. textual, audio visual and combinations of the three). It is the problem of genre classification that is the primary focus for this thesis and a discussion of some current approaches is presented in Section 3.3 and some of the limitations of these techniques are highlighted. This will provide a framework for the approaches developed in the remainder of the thesis.

3.1 Genre specific processing

The abstract model of a video sequence from a semantic perspective seen in Chapter 2 hierarchically segments a genre or sub-genre into logical story units, which are in turn made up of events, which in turn contain objects and /or people. The goal of semantic processing is to detect these entities, but current technology is not sufficiently advanced to be able to design a generic object or event detector to detect all occurrences in all video sequences. The particular logical story units, events, objects and people which occur in a video sequence though are commonly specific to a given genre so the semantic processing at lower levels is usually domain or genre specific. By using domain specific knowledge more complex features can be used which are tuned to semantic knowledge of the domain allowing more accurate detection of the semantic entities described. The domain specific knowledge is usually in the form of a model of production and editing rules and semantic content a different hierarchical video model can be used for each genre because within a single genre many sequences will contain similar content and have a similar layout. The model for a single genre is a specific instance of the abstract model from a semantic perspective and describes the typical logical units that genre can be segmented into i.e. a news sequence is usually segmented into stories. Within these logical story units for each genre will be occurrences of certain events, objects, people and settings which can be used to aid in the detection of the logical story units or for indexing the video sequence separately. Commonly techniques to detect more general elements of a model are designed to cope with a range of video sequences, but as a model becomes more specific, such as the definition of specific objects to be detected for semantic labeling, more complex features are needed and as such detection techniques will only be feasible on a limited domain of video sequences. As a result current research has focused on domains which contain an easy to define knowledge model with well defined production rules and specific content, such as sport and news broadcasts, although some techniques on other genres such as feature films also exists.

3.1.1 Logical Story Units

The segmentation of a video sequence into scenes or logical story units (LSU's) using only formative features is discussed in Section 2.2.3, in which an LSU in a video is by detected as a sequence shots with similar content. In some generic sequences, such as feature films or TV dramas, the techniques proposed will be sufficient to detect many LSU's, but in more specific sequences, such as in the sports genres, many scenes contain similar content so this scheme will not work, and in some applications (i.e. when searching for specific content) it may be necessary to index each LSU with a semantic label. From a semantic perspective, an LSU can be defined as a '*contiguous part of a video document's content consisting of a set of named events or other logical units which together have a meaning*' [3], and the detection of such segments is a very difficult problem in a general domain. With the aid of a knowledge model for a particular domain of video data, however, it becomes easier to index a video according to semantic content as the model can help to reveal the connection between low-level features and semantic content. In this sense, a model for a specific domain can be used as a template that is matched along a sequence in order to detect LSU's. The models proposed in the literature for each genre are dependent on the structure of a typical sequence of that genre. The news genre has a highly structured editing style so a very specific model can be used to detect LSU's and then labeled by topic, where as specific sports can be modeled and labeled using the rules and structure of the sport itself. Broader genres such as movie or TV sequences need a more general model, such as scenes with similar setting and dramatic incident which can then be labeled based on mood.

The segmentation of sports video sequences into logical story units is very useful in providing random access to the coverage of a sports game, and the problem is made significantly easier by the nature of such games which provide a natural segmentation scheme in many sports (i.e. very specific rules, scoring systems and time limits). As a result many sports broadcasts can be partitioned into game segments (i.e. first half, second half, breaks, studio intros) and then indexed using these labels as well as attributes in each segments such as key events and players. One example of this is seen in [49] which segments American Football games using textual information extracted from

closed-captions and visual shot segmentation into ‘live’ segments in which play is occurring, ‘others-game-related’ segments such as replays and breaks between live segments, ‘commercials’, and ‘others-game-unrelated’ segments such as studio segments. Due to the very specific structure of an American Football game each live segment followed by an others-game-related segment will make up a single play, known as a down, in the game. By detecting each play in the game, this information can be used to determine the full game structure within the video sequence, as shown in Figure 3.1. Similar concepts can be applied to segment most sports broadcasts with alterations due to variations in the structure of specific sports. While the structure of many sports is not as definite as that of American Football some form of LSU segmentation is usually possible. Some other examples of LSU segmentation in sports broadcasts include the detection of rally segments in Tennis matches in [50], and in [51] Tennis video sequences are segmented into the logical units first missed serves, rally, replay, and game break. The most common sport studied in the literature is Soccer with [52,53,54,55,56,57] all attempt to partition a soccer game into play and break segments. In sports sequences there is often an overlap between LSU and event detection which is discussed in Section 3.1.2.

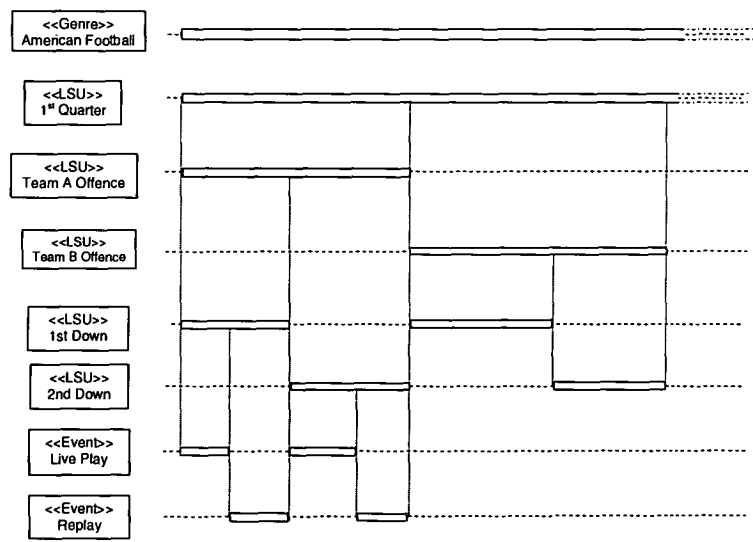


Figure 3.1: Structure of an American Football Sequence [49]

Commonly video sequences of an educational or informative nature such as news broadcasts or documentaries can be segmented into topics, stories or reports i.e. groups of contiguous shots which contain the same subject of discussion. In particular the well defined structure of the news genre has been used in a number of works [58,71,76,77,78,79,80,81] in order to segment a news broadcasts into logical story units using the standard temporal structure of each report or story being introduced by an anchorperson shot, which is then followed by the body of the news story, shown in Figure 3.2. In [126] a news sequence is segmented into talking head segments or narrative segments, where talking head segments are those containing either an anchor shot or interview sequence and all other segments are classified as narrative segments. Once the segmentation into stories is achieved further analysis can be performed, commonly on text transcripts obtained from closed captions or automatic speech recognition, such as in [59] which classifies each report into one of a number of subjects including politics, economy, and science, and similarly [60] which classifies a news story into one of the subjects politics, military, sports, weather, transport, business, health, entertainment, science & technology, and daily.



Figure 3.2: Key frames of shots from a typical news sequence with stories separated by anchor person shots

There has been some work in the genre of movies or films to detect logical story units in which a sequence is broken into scenes of contiguous shots unified by time, setting and dramatic incident which can then be indexed using some semantic label such as the type of dramatic event or the mood in the scene. While much of this work relies on the clustering of similar shots into scenes, as discussed in Section 2.2.3, there are some that take into account semantic information to segment movie sequences. These include the detection of dialogue or conversational scenes [61,62,63] which use the knowledge that these scenes are typically made up of repeated shots containing close ups of a single actor or a small group of actors in the scene, shown in Figure 3.3. In [64] the same rules are used to detect dialogue and action scenes, with shot length being used to differentiate the two. In [65] scenes with violent events are detected, while [66] classifies a scene as being dialogue, action, story unit, or generic (other), [67] detect dialogue, action, story unit, while in [126] a film is classified into segment of action or dialogue.



Figure 3.3: Key-frames from a typical dialogue scene in a movie sequence

3.1.2 Events

A semantic event can be defined as [3] ‘short segment which can be assigned a meaning that doesn’t change in time’ and is characterised by the specific interaction and inter-relation of objects and/or people in a setting over short period of time [2]. One event can be made up of other events and event detection can be used as a feature in LSU segmentation (e.g. a news report is usually introduced by an anchor person, so detection of the appearance of an anchor person could be used to indicate the start of a new story). Detection of some events may also be used in a semantic search and retrieval process as a video sequence can be indexed using a timeline of relevant events then a user may view only certain desired events (e.g. detection of goals in a Soccer video), or significant detected events could be used to create a summary of a video sequence for users to browse.

Once again techniques in the literature for the detection of events are most prevalent in the sport genre, as specific sports contain many events which are well defined and of semantic interest to the viewer. Some examples include [68] which develop separate models to detect important baseball events including home run, catch, hit, and infield play, [69] which detects the events break building, conservative play, escaping a snooker, shot-to-nothing in snooker games, and [70] which attempts to classify a tennis point into ‘baseline’ and net games. One heavily researched sport is soccer with works such as [71] which aims to detect the significant events goal, penalty, yellow card, red card, and substitution, [72] which aims to detect goals using visual and audio keywords, and both [56,57] attempt to detect the events shoot, corner kick, and free kick. In [73] highlight detection in American Football is attempted by detecting any event which changes the score i.e. a touch down, an extra point, or a field goal. The detection of video segments containing a slow motion replay in sports sequences is also used in [114,74] as they can indicate a highlight of significant interest to the user has occurred. Finally a technique using motion patterns is presented in [75] which first classifies shots as having no motion pattern, an object motion pattern, and a camera motion pattern. The object motion patterns are then further semantically classified into one of the object patterns diving, high jump, race, tee off in Golf, a goal shot in Soccer, team offense or

defense in Basketball, a penalty shot in Basketball, or wipe using a support vector machine designed for each pattern.

Event detection in the news genre is also common due to the well defined production and editing structure, and the presence of a number of common shot types used. In the news genre, events detected are usually used to segment a sequence into reports, which can then be labeled based on their semantic content. The most common event detected is an anchor person shot [71,76,77,78,79,80,81], as these shots are significant because they usually signal the start and end of a report. Other attempts to detect common events in news broadcasts include [78] in which news shots are classified into intro or highlight shot, anchor shot, anchor shot with two people, shot containing a gathering or meeting, still image shot, live-reporting shot, speech, weather, sport, text-scene, and special. In [79] shots and edit segments of a news are classified as one of the following types anchor, report, begin, end, weather forecast, interview, cut, frame translate, and window change. In [71] segments containing a reporting anchor, a monologue, a split-view interview, and a weather report are detected while in [80] report or speech shots, anchor shots, shots containing a reporter walking, gathering (i.e. conference or meeting) shots, and shots containing computer graphics are detected. In [81] anchor person shots and video captions are detected and then used with production rules of news video to segment into reports.

Some other examples of semantic event detection in other video genres include the detection of hunt segments in wildlife videos [82], the detection of violent events in feature films [83,84], the detection of flashlights, which can be used to indicate the occurrence of a dramatic event, in feature films [85], and in [86] a cooking video is segmented into face shots and hand shots, and the significant events of a cooking motion and the appearance of food in the hand shots are detected using motion based features. The TRECVID high level feature detection task also includes detecting video shots which contain a number of semantic events which include in the 2003 experiments [87] the events news subject face that is not anchor, news subject monologue, sporting event, weather news, and physical violence, in the 2004 experiments [88] the events basket scored in a Basketball game, an aeroplane taking off, people walking or running, and

physical violence, and in the 2005 experiments [89] the events people walking or running, an explosion or fire, and a sports event.

3.1.3 Settings, objects, and people

The detection of specific entities in the content perspective and subsequent semantic indexing is very much an unsolved problem in the domain of general broadcast video. Initial attempts at segmenting a video sequence into different layers of content such as settings, objects, and people concentrated on formative approaches such as detecting object as regions of similar colour and motion, separating the foreground from background in a scene, or detecting regions of flesh colour. These techniques had reasonable success on a general video domain, but labelling at a higher semantic level such as detecting a specific object such as a car, or a specific setting such as a seaside scene, or the recognition of a specific person proves to be much more challenging. Current techniques are seen in the literature are designed for specific content (i.e. one technique designed to detect cars for instance and would need a completely different technique to detect people), on a narrow video domain such as a specific genre or sub-genre. The occurrence of specific content can be detected using the visual mode using content-based features, in the audio mode by detecting sounds an object may make and in the textual mode by detecting references to content from the transcript of the narrative or from screen text.

The detection and indexing based on setting is very common in the sport genre as sports broadcasts are usually produced using a finite number of camera views of a fixed playing environment. Examples of this are proposed in the literature for a number of different sports and include the extraction of the pitcher scene in baseball [90], or [68] which detects the seven view classes pitch view, catch overview, catch close-up, running overview, running close-up, audience view, and touch-base close-up in a baseball sequence. In [91] the coverage of Snooker is segmented into the viewpoints large view, table close up, player close up, other table view, and crowd or journalist etc. The sport of Tennis is considered in [91] using the viewpoints large view, player close up, crowd or journalist etc, and in [92] which uses the viewpoints court-view-playing, medium-view-player-following, close-up head-tracking, site-bird-view, and audience. In [93]

each shot from American Football coaching video is classified into one of three types score board, end-zone, sideline. The structure of Basketball video is captured in [94] to classify basketball scenes into six classes fast-motion court-view scenes, slow-motion court-view scenes, penalty scenes, in-court medium scenes, out of court or close-up scenes, and bird-view scenes. Once again the sport of Soccer is commonly used such as [53] in which each shot is classified as either close-up, zoom-in, and [57] which detects close-up, and offensive scenes as well as events such as corner kicks and shots on goal using heuristic thresholding approach and finally [56] which takes transforms low level features into the views close-up view, medium view, midfield view, forward-field view, and goal view. The detection and semantic recognition of settings in general sense is attempted in the TRECVID high level feature detection task. Video shots which contain an outdoor setting, an indoor setting, a cityscape, or a landscape are searched for in the 2002 experiments [95], and in 2003 experiments [87] attempts to detect shots containing outdoor settings, buildings, roads, vegetation, or a non-studio setting were made. In the 2004 experiments [88] shots with a beach setting, or roads were detected and the 2005 experiments [89] concentrated on detecting shots containing building exteriors, waterscapes or waterfronts, and mountains.

The detection and semantic classification or recognition of objects in a sequence is a very broad proposition can occur at a number of different semantic levels. Initial low level attempts included simple segmentation of a frame into objects by separating the foreground and background layers in a scene, tracking objects through shots including appearance and disappearance, and using key frame indexing techniques at the object level rather than the frame level. The semantic labelling of objects however is a much more complex problem. The approaches for the detection of specific objects use complex visual features and are currently only feasible in domains limited to the context of a single genre or event and under assumed constraints. Some examples in the literature of semantic object detection are the tracking of moving balls in Snooker in [69], tracking of the ball in soccer in [96], detection of court lines in Tennis in [70], detection of moving cars from footage of city street intersections in [97], and the detection of settings which contain the objects rocks, sky, snow, water and forestry/greenery and also the setting outdoor is attempted in [98]. The TRECVID high

level feature detection task poses the problem of detecting shots containing a number of semantic objects in its various experiments including a building, a road, an animal, a car or truck or bus, and an aircraft in the 2003 experiments [87], a boat or ship, and a train in the 2004 experiments [88], and a map, a US flag, and a car in the 2005 experiments [89].

While the detection of people could be considered to be the same problem as object detection, as people can be seen as a subset of objects, because the occurrence of people is so common and usually the focal point in a scene, the detection and indexing of people should be considered separately. Like object detection and classification, the semantic detection of people can occur at a number of levels (i.e. from generic person detection to the recognition of a specific person). It is common in current techniques that generic people detection is performed in the visual mode by detecting faces or other body parts or using the audio mode by the presence of speech. Higher level semantic processing such as the recognition of a specific person however currently usually occurs using the textual mode by detecting names from a transcript or text overlay [3]. When designing a generic person detector a common approach is to detect a human face. A number of approaches for this have been developed, and a survey of many of the techniques is presented in [99]. The detection of a specific person using face recognition is a much more complex problem and is only possible in a constrained environment such as a full frontal view of face under good lighting [100] which may limit possible application in video management systems. Another approach to person detection is to detect the whole human body by detecting each body part separately before checking they are correctly connected [101]. The TRECVID high level feature detection task also attempts to detect both generic and specific features related to people. In the 2002 experiments [95] shots containing a face, and shots containing people (i.e. more than one human at least partially visible) were detected, as did the 2003 experiments [87] which also attempted to detect shots containing a specific person in Madeleine Albright. In the 2004 experiments [88] shots containing the specific people Madeleine Albright and Bill Clinton were detected along with the generic people walking or running and in the 2005 experiments [89] shots containing people walking or running, and shots containing a prisoner were detected.

3.2 High level genre classification

The problem of identifying the high level genre and/or sub-genre for a video sequence plays a key role in both the determining what subsequent domain specific processing can be used (as discussed in Section 3.1), and to assist in the partitioning of a large database into genre/sub genres specific clusters. The exact genres and sub-genres needed to encompass the vast range of video sequences in a real world situation is quite a subjective task with the literature including genres such as {sport, news, commercial, drama, music, talk show, sitcom, soap, etc} and so early research efforts have concentrated on well defined classes. In general the aim of classification at a genre level will be to partition a video sequence into segments of the same genre or sub genre. This will usually be at a program level rather than a scene or shot level, although many video sequences may have commercials interspersed within and between programs. To perform this program level classification many techniques perform classification at a clip level (i.e. a short fixed length segment of a video sequence) and combine these clip level results to find the program level segments, in a similar approach to scene detection through the clustering of similar shots. Very few techniques perform genre classification at the shot level, although this information may be more accurate when detecting boundaries between different genres, but it may also be useful in difficult cases such as a sport stories in a news program where the context of surrounding shots is needed in order to perform genre classification at the program level.

The first attempt at high level classification of genre appeared in 1995 [116], attempted to classify a video sequence into the set {news, commercial, cartoon, tennis, car racing} a mixture of genres and subgenres. A number of works have been published since, with significant variation in the set of genres including single genre sets, multiple genre sets, sets of subgenres in a specific genre, and some sets with a mixture of genres and subgenres. In [2], Roach defines two types of video classification: identification and detection/verification. The classification of a unit of video into a single genre is known as detection or verification i.e. a binary decision of 'in genre x ' or 'not in genre x '. In general, techniques considering a single genre are known as detection, and techniques considering multiple genres are known as verification. The classification of a video unit

into one of a given set of genres is known as identification. In the literature techniques for detection and identification are most common, but Roach argues that verification is the most representative of a real world problem as in the process of identification any sequence outside the set of defined genres is ignored and so accuracy of classification may be skewed higher. Verification testing however can be viewed as simply having a 'reject' class in the set of genres, and so should produce more realistic results with less sensitivity to the experimental set of genres.

A review of the genre classification sets from the literature can be seen in Table 3.1, which includes 19 different techniques from 13 different collaborations of authors. The table shows a large range in the genre sets including the genres sport, news, commercial, cartoon, music, drama, film, medical, talk show and also a number of sub-genres including the sport sub-genres Car Racing, Tennis, Basketball, and American Football, the news sub-genres news reports, weather forecasts, and financial news, and the drama sub-genres sit-com, soap, and comedy. It is worth noting however that almost all works include the sport and news genres in their genre set (or a mixture of subgenres from each genre in some of the earlier works) and commercials are used in a majority. This is because these genres contain the most defined production and editing rules and have the narrowest content domain, and because these genres are less sensitive to the subjective nature of determining which genres exist in the domain of broadcast video. Apart from these approaches designed for multiple genre sets, there are also a number of works in the literature designed for single genre detection. The genre for which single genre detection is most commonly seen in the literature is that of Commercials [102,103,104,105,106,107,108,109,110,111,112] as the separation of commercials from a broadcasted program is seen as an important filtering step prior to further processing. Other approaches have attempted to detect sport segments [113,114], and cartoons [115].

Table 3.1: Genre classification sets from the literature

		Sport	Sport sub-genres	News	News sub-genres	Commercial	Cartoon	Music	Drama	Drama sub-genres	Movie/Film	Medical	Talk Show
[116]	1995		✓ ₁	✓		✓	✓						
[117]	1997		✓ ₂		✓ ₁	✓							
[118]	1998		✓ ₂		✓ ₁	✓							
[119]	1999		✓ ₂		✓ ₁	✓							
[120]	2000	✓		✓						✓ ₁			
[121]	2001				✓ ₂	✓							✓
[122]	2001	✓		✓			✓						
[123]	2002			✓		✓							
[124]	2001		✓ ₂	✓		✓							
[125]	2002		✓ ₂	✓		✓							
[126]	2002	✓		✓							✓		
[127]	2002	✓		✓					✓				
[128]	2001	✓		✓						✓ ₂			
[129]	2003	✓		✓						✓ ₂			
[130]	2004			✓							✓	✓	
[131]	2000	✓		✓		✓	✓	✓					
[132]	2002	✓		✓		✓	✓	✓					
[133]	2003	✓		✓		✓	✓	✓					
[134]	2004	✓		✓			✓	✓	✓				
Sport sub-genres: 1: Car Racing, Tennis 2: Basketball, Football													
News sub-genres: 1: News Reports, Weather Forecasts 2: Financial News													
Drama sub-genres: 1: Sitcom, Soap 2: Comedy, Soap													

In Table 3.1 it is shown that many semantic video processing techniques are designed not for just a single genre, but even more specifically a single sub-genre, so it makes sense that sub-genre classification is also an important cognitive task. The sport genre is once again prevalent in the literature for the problem of sub-genre detection, including

[135] which attempts to classify video shots from a sports sequence into one of the following kinds {floor, high diving, field hockey, long horse, javelin, judo, soccer, swimming, tennis, track} based on colour and texture features calculated from key frames. In [136], a hidden markov model based on feature vectors calculated from MPEG-1 motion vectors is used for classifying sports sequences into one of three sub-genres {basketball, ice hockey, soccer}. In [137] rule based fuzzy inferencing is used on mosaiced key frames to classify cricket, golf, football sequences. In [138], TV sports news articles are classified into one of 8 sub-genres {American Football, baseball, golf, ice hockey, soccer, sumo wrestling, tennis, volleyball}. The genre of feature films or movies has also been explored in terms of sub genre classification. Two approaches have been proposed to classify film previews into different set of sub-genres. In [139] the set {Action + Drama, Drama, Comedy + Drama, Comedy, Action + Comedy, Horror} is used while the same authors in [140] classify a film preview as action or not action, and the not action sequences are further partitioned into the set {Comedy, Horror, Drama/Other}. In [141] films are classified into character (comedy, drama, romance) or action via HMM using ratio of shot duration to motion energy, and similarly in [142] films are classified into the two sub-genres comedy/ romance or action via shot duration and average shot activity.

For each of the techniques in Table 3.1, the factors affecting classification performance include low level features used, the signal mode used (i.e. visual, acoustic or combined), and the type of classifier. Unfortunately the comparison of these factors is made more complex as many result sets use different set of genres and a different experimental data set. As there is no standard data set for this problem, care has to be taken to ensure the experimental data set is representative of a real world situation. In terms of genre set, a comparison can be made of the works in [131,132,133] which use the same set {Sport, News, Cartoon, Music, and Commercial} and also [134] which substitutes Commercial for Drama. Following is an overview of these works and a more detailed look at some of the low-level features can be seen in Section 5.1 .

Genre classification in [131] was performed for 40, 60 and 80 second clips from a video sequence and 9 low-level visual features were used to represent each clip. Two features were designed to describe shot based editing, being the average shot length, and the

percentage of each shot transition type. Four motion/activity based features were also used being, camera motion magnitude, threshold based prevalence of special effects such as lighting changes and flashlights, threshold based prevalence of static scenes, and threshold based 'motion runs' to describe the length in time of consecutive active frames within a clip. Finally three HSV colour space statistics, being the variance of luminance levels, calculated from histograms, the average brightness, and the average saturation were also used. A C4.5 decision tree was used as a classifier for about 8 hours of TV material from a variety of channels and a variety of programs encoded in the MPEG-1 format. The majority of the features were calculated from the raw video stream with some using compressed domain information from the MPEG-1 file. Assuming a similar database size for each genre then for a 60 second clip this gives about 100 clips per genre to be split up using 60% for training and 40% testing. An average classification accuracy of 83.1% across all genres was achieved with a 60 second clip across 100 different randomly selected train/test sets, with a 60 second clip showing the best trade of between average error and standard deviation of error.

In [132] both visual and acoustic low level features are used. The visual feature is a motion metric calculated from the frame-to-frame difference in a sequence. This differential motion signal is represented using a DCT of short-term spectral estimates of 0.77 seconds with 50% overlap. The short-term spectral estimates are represented by an instantaneous window of 0.77 seconds and a transitional window of 1.54 seconds (i.e. three 50% overlapped instantaneous windows) centred on the current instantaneous window giving 64 DCT coefficients. A Gaussian mixture model (GMM) with 64 components is used as a classifier. A separate GMM is used for classification based on audio features using 14 DFT based Mel Frequency Cepstral coefficients (MFCC) generated in a similar process to the visual feature. Experimental results are obtained using one hour of video per genre, giving 180 20 second clips per genre, of which half are used for training and half for testing. Both GMM's also include a 'world model' that is subtracted from the GMM probability prior to comparison with an accept/reject threshold to provide verification rather than just identification of genre. Classification error rates of 73.6% using only the visual mode, 76.8% for just the audio and 84.3% for a linear combination of the GMM outputs of each mode are reported.

The same GMM classifier and the same data set is used in [133] but with a new 62-dimensional visual feature vector combined with the same 14-dimensional audio vector. The majority of the visual features are generated from the MPEG-7 standard including a 16-dimension Scalable Colour descriptor, and 12-dimension Colour Layout descriptor and a 32-dimension Homogenous Texture descriptor. A 2-dimensional motion feature, the mean and standard deviation of the magnitudes of MPEG video motion vectors, is also used. Each feature vector is calculated for every sample in a 1 second window (~25 samples for visual ~42 samples for audio) and Principal Component Analysis (PCA) is used to reduce the dimension and produce a 'super' feature vector. The super feature vectors in a decision window of 40 seconds are concatenated and the dimension of the stream of super feature vectors is further reduced with PCA before using the GMM for classification (with between 3 and 10 mixture components shown to be satisfactory) producing an error rate of 87%.

In [134] a video sequence is segmented into 60 second clips from which a number of low-level shot based and key frame based visual and audio features, are computed. The majority of features are from the MPEG-7 standard, including five audio features from the MPEG-7 standard, three visual features to represent colour using the MPEG-7 Scalable Color descriptor, five visual features to represent camera motion using MPEG-7 Camera Motion descriptor, one visual feature to represent motion intensity using an MPEG-7 Motion Activity descriptor computed from a shot's activity histogram, and the final feature is the number of shots in a clip. The classifier used to determine the genre of a single clip is a CART decision tree with the GINI splitting criterion [177] with error rates of 72.0% for visual features, 72.8% for audio features and 88.8% for combined features reported. The genre of a sequence was then determined by comparing the results of genre classification for a number of randomly selected clips from a sequence. Classification results were presented for when a video sequence was classified a genre if 2 out of 3 randomly selected clips were the same genre, and for when 3 of 5 clips were classified as the same genre, with classification rates improving as more clips were selected. The data set used contained 72 60 second video clips for each genre of which 47 clips are used for training the classifier.

3.3 Conclusion

Much of the current research relating to content-based video management involves some semantic processing to provide human understanding of the content of a video sequence. This is because initial attempts at indexing and retrieval of video sequences only relied on similarity measures based on low level features which could not always provide perceptually accurate results. The aim of semantic processing is to provide indexing and retrieval techniques based on the information in a video sequence that is of cognitive importance for humans. In order to achieve this, the semantic information is arranged into a number of different levels of abstraction from the high level genre segments which generally last for many minutes down to object information which may only be relevant for a number of shots or frames. The abstract levels used in this thesis, genre, sub-genre, logical story unit, event, and object/person/setting are those commonly used in the literature but the definition of specific instances at each level is a subjective task and often an overlap between the levels occurs. In this chapter an overview of a number of attempted classification tasks at each of the semantic levels is given, which shows that in general broad semantic classification such as at the genre level use more general global features and can be made across broad video domain. As the semantic classification gets more specific the features become more complex and so the video domain to which these features are relevant also become more specific (i.e. confined to a single genre or sub-genre). It is also common in the literature that low-level semantic information to be used in a bottom up approach to higher level semantic classification (i.e. the detection of objects can be used in event or logical story unit detection). In practice though as bottom up approaches require complex features in the first stage this is not currently feasible for use on a broad range of video with current technology. In practical systems a top down approach which first partitions the system into smaller domains such as genre or sub-genre would be used in combination with a bottom up approach on these more specific domains [2,98].

The remainder of this thesis will concentrate on the first stage of semantic classification, namely genre classification. In this chapter a comprehensive review of approaches currently used to tackle this problem was presented. It was seen that there was a diverse

range of genre sets used in the literature, so particular attention was given to four recent approaches on very similar genre sets. The main observations made from these approaches were that the features used were of a global nature to measure the long term spatio-temporal variations and editing structures in a sequence, and that all techniques perform genre classification at a clip level (i.e. across a number of shots) with clip lengths ranging from 20 to 80 seconds. One disadvantage of classifying genre at the clip level is that the boundary between two genres in a sequence will coincide with a shot boundary, so it is possible that a single clip will span two genres. In this thesis genre classification will be attempted at the shot level to counter this problem. Classification at the shot level should also help in the problem of ambiguous cases such as a news story containing sports footage. When viewed at the shot level the sports shot will be classified as sport but at the program level the shot should be classified as news. If the initial genre classification is at the shot level this will satisfy the shot based perspective, and if the genre classification of surrounding shots is used as context the sports shots could be reclassified as news to satisfy the program perspective. Another focus of this thesis is to develop new low level global compressed domain features to be used in the classification process. The compressed domain considered in this thesis is that of MPEG-1 video, which is described in the next chapter, and the low level features used are described in Chapter 5. As previous works have focused more on feature extraction rather than classifier design or any comparison of different classifiers, a number of classification techniques to transform the low level features into high level genre labels are compared in Chapter 6. The results include those used in current approaches such as a Gaussian mixture model and a decision tree, as well as new approaches to this problem using Radial Basis Function Networks with a particular focus on the use of decision trees to initialise the structure of the RBF network.

Chapter 4

Compressed Domain Video Analysis

If we consider a completely uncompressed digital video stream, containing no audio stream, of similar resolution to the PAL colour television standard (i.e. 25 frames per second with a visible frame size of 720 x 476 pixels and stored using 256 levels (8 bits) per colour channel (RGB), this will result in a bit rate of 196.1 Mbps (24.5 MBps). This will result in a storage size for a 30 minute video of approximately 43 Giga Bytes. As a result the efficient transmission and storage of digital video is dependent on the ability to compress the video stream. In fact, compression techniques are one of the major enabling technologies of the recent rise in the prevalence of multimedia content, and the vast majority of digital multimedia standards in common use today store data in a compressed form.

It is logical then, to minimize computational requirements, that the processing of any multimedia data, such as the extraction of low level visual features for a video sequence to be used in a CBIR system, should be performed in the compressed domain. The advantages of compressed domain processing include the avoidance of fully decoding the compressed stream to perform processing on pixel values, the ability to use some compressed data directly, or with minimal processing, as low level features, and the ability to perform processing either in parallel with compression, in bulk during the population of a database, or by the end user on a distributed basis, without any great variation in efficiency. In fact if we consider the MPEG compression standard, *“Extracting Frame Type, DCT, Motion Vectors takes less than 20% of computational load when decoding an MPEG file”*[143].

The aim of any compression scheme is to remove as much redundancy in a data source as possible so that only the minimum amount of information needs to be coded. This is similar in effect to the goal of low level visual features in a CBIR system, in that we want to find the least amount of information to describe the content of a video sequence. While the final goal of compression, i.e. the reduction of the bit rate of a video sequence, is quite different to the goal of a CBIR system, the fact that both are concerned with the most relevant information in a stream, it makes sense that, with knowledge of the inner workings of a compression scheme, the compressed domain information should be relevant to the computation of low level visual features.

When considering the use of compressed domain information to compute visual features it is important to understand what redundancy exist in video sequence and the type of compression that is used to remove it. The uncompressed digital video sequence described above is coded with a fixed bit rate (196.1 Mbps), but the information rate, a measure of the predictability in the sequence, is not constant as some segments of a video will have very little change from frame to frame. This difference between bit rate and information rate is the redundancy compression schemes try to remove. There are two categories to describe compression schemes, namely lossless and lossy coding. Lossless coding schemes, such as arithmetic coding and variable length coding, will produce a decoded stream that is identical to the original signal after the coding/decoding process. Variable length coding relies on the statistical distribution of a signal, in which the most commonly occurring samples are coded using a small number of bits, and the less frequent samples a large number of bits. Lossy coding schemes, such as transform coding and predictive coding, use signal processing techniques and produce a decoded signal which is not identical to the original. In predictive coding a prediction from one sample to the next is made and the prediction error, which is hopefully small, is coded rather than the sample itself. In transform coding, orthogonal transforms are commonly used to achieve energy compaction and transform coefficients with small energy are discarded. Lossy coding can be objective or subjective. The latter makes use of the known characteristics of the human visual perception for the quantization of the transform or predictive coefficients. In the case of a video compression scheme, the response of the human visual system to visual stimuli is

modelled, and the model is used to determine the minimum resolution needed to code a video sequence with no subjectively detectable errors despite the fact that the original uncompressed bit stream and the decoded bit stream are different. Perceptive coding techniques can provide a much higher compression rate than their objective counterparts.

In order to make use of compressed domain information when computing visual features the relationship between the compressed domain information, the raw visual data, and the video content needs to be investigated. These relationships are evident when observing what information redundancy each step in the compression process is trying to remove. For example video sequences can contain spatial redundancies as nearby pixels in a frame are likely to be a similar value, and also temporal redundancies as nearby frames are likely to contain similar content. It is also relevant to know how the performance of a compression scheme varies with respect to the type of video sequence. For example the temporal redundancies in a video with a large amount of action or motion will be harder to detect than a scene with little motion. Similarly the spatial redundancies in a frame with many edges are not as significant as a frame with more uniform content. It is also necessary to understand the distribution of various compressed domain information, and how these will change in the presence of different events such as editing effects or objects / background either entering or leaving a video shot. Finally, it is also necessary to be aware of any errors that can occur in compressed domain information with respect to calculating visual features considering that the compressed domain information is generated with the goal of maximizing the compression rate of a video sequence. This thesis considers digital video sequences compressed using the MPEG-1 [144] format so an overview of this standard is presented in Section 4.1. A discussion of the information present in the MPEG-1 which can be easily extracted and used in the calculation of visual features is presented in Section 4.2. One such source of information is one of the key components known as motion vectors which are used to compress temporal redundancies in a video sequence. In the MPEG standard however the motion vectors are calculated to maximise compression so may not represent the true motion in a sequence. A filtering technique to identify and discard erroneous motion vectors is presented in Section 4.3.

4.1 MPEG-1 bit stream

The compression scheme chosen in this work is MPEG-1 format, which is the first in a family of multimedia compression standards produced by the Moving Pictures Expert Group, namely MPEG-1 [144], MPEG-2 [145] and MPEG-4 [146]. All of these MPEG standards define the syntax for the bit stream of a MPEG compressed video sequence (see Figure 4.1), but not the specific details regarding the implementation of the encoder and decoder, which is left up to individual designers. This standard was chosen as it is a widely accepted and used family of open standards for video coding, with MPEG-1, released in 1992 being the most common format used in video delivery via the Internet and MPEG-2, released in 1996, being the standard used in DVD production. MPEG-4, which was first released in 2000, is the most recent and implementations of the standard as a codec are not yet in use as commonly as the earlier standards. MPEG-1 was chosen as the compression scheme for this work as the vast majority of compression techniques used in the MPEG-1 standard are also used in MPEG-2 and MPEG-4, with extensions and expansions to properties such as bit rates, picture sizes, picture formats, picture resolutions and coding techniques. The initial intention of MPEG-1 was to deliver video coding at a low bit rate of < 1.5 Mbps (MPEG-2 can deliver bit rates from 4 - 100 MBps), and this low resolution video provides reduced processing times, a desirable attribute in this research.

The MPEG-1 bit stream has a hierarchical structure, like many computing based protocols, with each layer providing a different level of abstraction. This layered structure allows logically distinct units to be separated, which can prevent ambiguity and also improve the coding/decoding process through the efficient handling of overheads. The lower layers of the MPEG-1 syntax describe elementary video and audio bit streams, which contain the coded bit stream of a single video sequence or audio signal. The higher layers of MPEG-1 are primarily concerned with the multiplexing and synchronisation of program streams and transport streams. Program streams contain the elementary video and audio streams of a single program multiplexed into a single synchronized stream, and are used when the transmission of the stream is not necessary (e.g. a DVD recording). A transport stream can contain multiple elementary streams,

from multiple programs multiplexed and synchronized for transmission (e.g. a digital television broadcast).

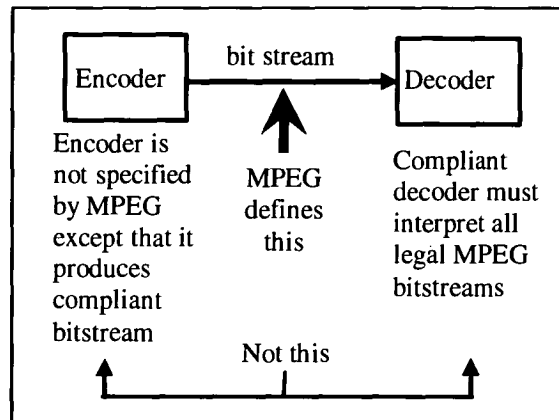


Figure 4.1: MPEG standards define the bit stream not the encoder/decoder [147]

The research presented in this thesis is concerned only with single elementary video streams, so a detailed discussion of the syntax of these follows. An elementary video stream is segmented into six different layers (Figure 4.2), with each supporting a specific logical or signal processing function (Table 3.1). The video sequence layer packages a single elementary video stream together and contains parameters relevant to the stream as a whole, such as picture height, picture width, pixel aspect ratio, frame rate, bit rate, buffer size. The group of pictures (GOP) layer (see 4.1.1 below) packages together groups of frames. This layer allows random access to the video stream as the first frame in a group of pictures is coded independently from other frames. It also provides error recovery as any errors from a previous GOP will not be propagated to the next. The picture layer is the primary coding unit as a video sequence is essentially a sequence of still pictures or frames. It contains parameters specific to a frame, such as the frame type, and any information needed for buffering. The slice layer segments the picture layer into a number of different horizontal strips of macroblocks. The function of the slice layer is as a resynchronisation unit either in the presence of coding or transmission errors. Resynchronisation occurs as the first macroblock in a slice will have its data coded absolutely and the following macroblocks will be differentially coded with reference to the previous macroblock in the slice. As a result greater compression is achieved through differential coding but errors will not propagate from slice to slice or

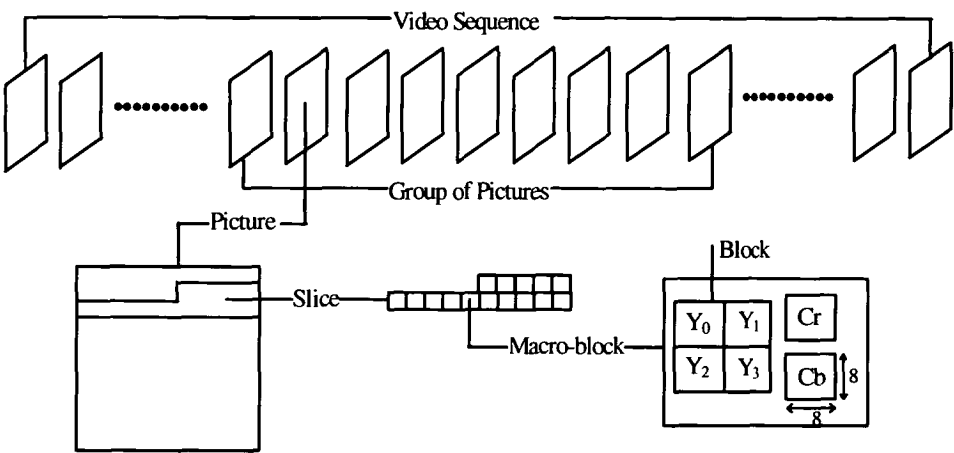


Figure 4.2: Layers of a single video stream in an MPEG-1 file

frame to frame as each slice resets or resynchronizes the differential coding. This allows each slice to have different parameters, to allow some slices to be coded more accurately than others. The exact number and location of slices within a frame is not defined in the MPEG standard so needs to be determined by the encoder. The macroblock layer is the motion compensation unit, as this is the level that temporal redundancy compression is performed at (see 4.1.3 below). The macroblock size in MPEG-1 is 16x16 pixels which are further segmented into a number of blocks containing the actual luminance and chrominance information. The number of blocks in a macroblock is variable in MPEG-1 due to the ability to code the luminance and chrominance information at different rates. The block layer is 8x8 pixels and is the level at which the spatial redundancy compression (see 4.1.2 below) is performed. It is called the DCT unit as the core function in spatial compression is the transformation of the spatial domain pixel values of a block into the frequency via the Discrete Cosine Transform (DCT).

Table 4.1: Six layers of syntax for MPEG video bit stream [148]

Syntax layer	Functionality
Sequence Layer	Random Access Unit: Context unit
Group of Pictures Layer	Random Access Unit: Context unit
Picture Layer	Primary Coding Unit
Slice layer	Resynchronisation unit
Macroblock layer	Motion compensation unit
Block Layer	DCT unit

It is this layered structure that provides the MPEG-1 standard its generality, flexibility and efficiency. The many parameters at each level are flexible enough to provide good quality compression for varying applications with varying demands. While the coding of these parameters provides an overhead in the compression scheme, the ability to code them at only the level necessary provides great efficiency. The flexibility to code many parameters at a number of levels also allows the scheme to vary the resolution of the coding from GOP to GOP down to macro-block to macro-block, which enables the encoder to deliver a low bit rate but high quality.

4.1.1 GOP frame sequence

MPEG-1 was designed as a somewhat generic video standard, in that it was not designed for use in any specific application. The initial requirements were designed to meet the demands of storage of video on CD-ROM, for any number of applications, and for video delivery via telecommunication networks such as video telephony or videoconferencing. The generic requirement were therefore to deliver a high quality compressed video at a high compression rate (approximately 1.5Mbps including audio). The main quality concerns, apart from the subjective viewing quality of the video, are robustness to error, and the ability for random access, that is the stream should be decodable with only a short delay, say 0.5 seconds, from any point within the stream.

In order to fulfil these requirements, MPEG-1 is a flexible standard containing a mix of intra-frame coding to deliver random access and inter-frame coding to achieve the high

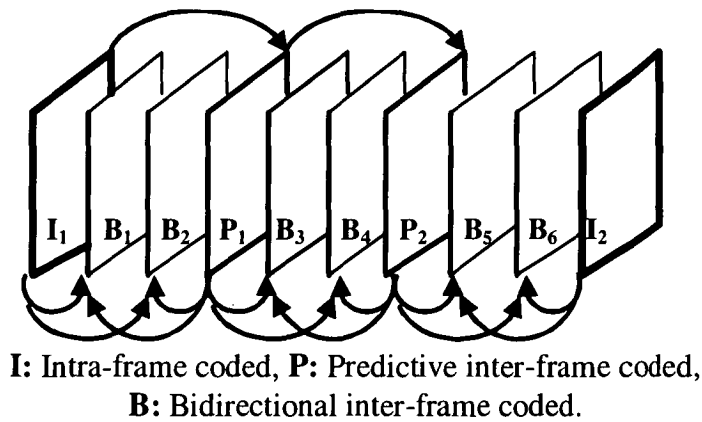


Figure 4.3: A typical Group of Pictures (GOP) in MPEG-1

compression rates necessary. It can be seen in Figure 4.3 that there are three picture types in the MPEG-1 standard. A group of pictures is defined as the sequence of frames between two intra-coded I-pictures (e.g. from I_1 to B_6 in Figure 4.3). The frames between the I-pictures are inter coded, and can be any combination of predictive P-pictures, which use the previous I or P frame as a reference for temporal redundancy, or bi-directional coded B-pictures, which can reference both the next and previous I- or P-picture. B-pictures are not used as references in inter-frame coding.

While the exact GOP structure is not specified in MPEG-1, the sequence in Figure 4.3 is the most common, which contains the reference frames spaced at approx $1/10^{\text{th}}$ second interval[148], which provides a reasonable result for the majority of video content. The variability in this structure lies in the number of B-pictures between the reference frames. As there is statistically more information to be used as reference when coding B-frames, B-pictures can handle events such as the occlusion and uncovering of the background by a moving object, providing greater compression (approximately 50:1) than I-pictures (approximately 7:1) or P-pictures (approximately 20:1). The trade off for this greater compression is as more B-pictures are used, the correlation between the reference frames and the B-pictures decreases, increasing the risk for error. Increased B-pictures also has the effect of increasing the complexity and delay in the coder and decoder, and reduces the ability for random access and editability of a stream. It is necessary to have regularly spaced I-pictures to give random access to the stream and to stop the propagation of errors due to inter-coding through the stream, as they can be decoded fully with no reference, and hence decoding, of another frame. P-frames provide a good reference to B-pictures as they are more resilient to errors as initial predictions are from I-frames, but also provide more compression than I-pictures. The ideal structure the GOP will be application dependent (i.e. if lots of editing necessary then fewer B-frames will be used).

It should be noted that the GOP structure in Figure 4.3 shows the order in which the frames will be displayed on the screen. When decoding the frame B_1 and B_2 both I_1 and P_1 need to be fully decoded first and placed in a buffer for further processing, so the MPEG-1 bit stream will contain the frames in decoding order, (i.e. $I_1 P_1 B_1 B_2 P_2 B_3 B_4 I_2 B_5 B_6$). The last B-frame in a GOP can reference the I-frame of the next GOP, so this

GOP structure is not a truly independent layer in the MPEG syntax. To achieve true independence, a GOP structure of IBBPBBPBBPI is needed, or it is possible to explicitly code the final B-frame only with references to the previous P-frame.

4.1.2 Spatial redundancy compression (Intra-coding)

The first step in MPEG-1 compression is a colour space conversion at the frame level and subsequent sub-sampling according to the sensitivity of the human visual system. A typical uncompressed colour digital picture will be stored using 3 colour channels (*R*:Red, *G*:Green, *B*:Blue) which when added together in different proportions can produce a large range of colours. In order to display a colour image on a monochrome screen, or vice versa, the *Y:Cr:Cb* colour space is used. The *Y* channel is the luminance, or the monochrome intensity image of the colour equivalent. The *Cr* and *Cb* channel contain the colour information in the form of difference images, the *Cr* channel containing *R-Y*, and *Cb* containing *B-Y*. The three channels can be combined using the transform in Figure 4.5 in order to recreate the RGB image. The *R-Y* and *B-Y* colour difference signals are used in favour of *G-Y* as the *G* channel is dominant so *G-Y* will be the smallest difference, and so more susceptible to noise.

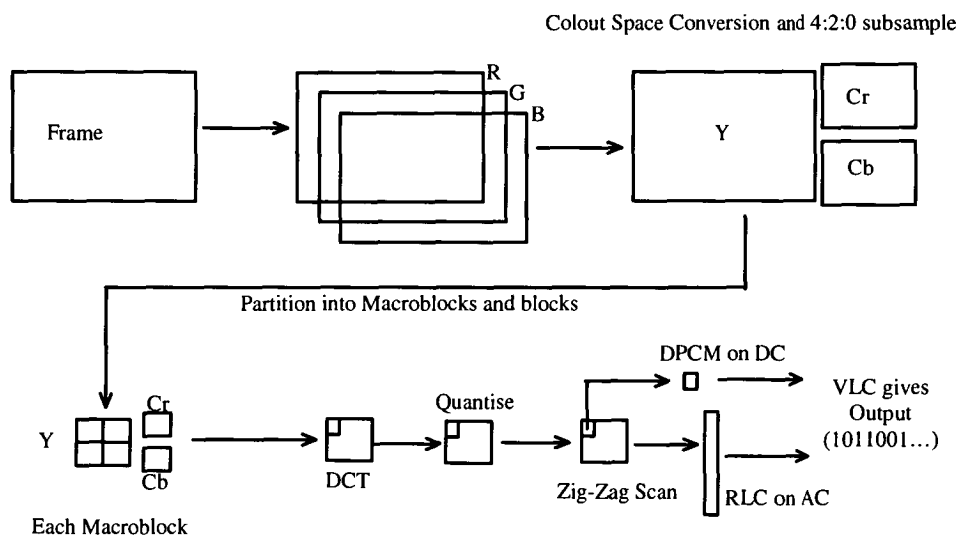


Figure 4.4: Intra coding process in MPEG-1

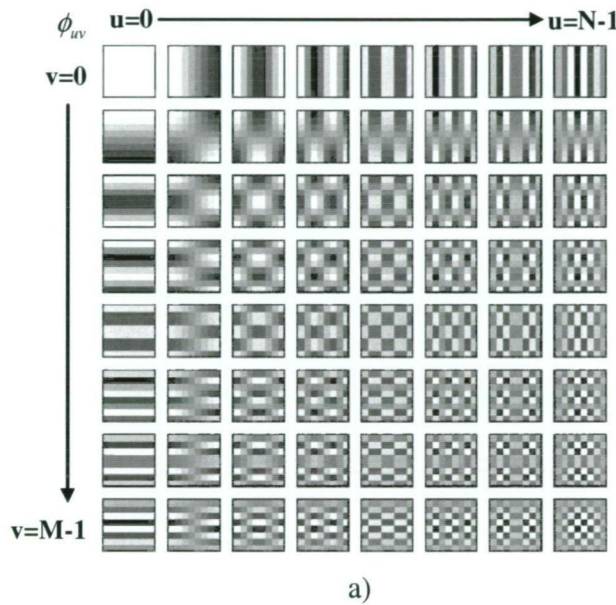
The human visual system is more sensitive to variation in the luminance channel Y , than the chrominance channels Cr and Cb . This results in the ability to encode the chrominance channels at a lower resolution than luminance by sub-sampling the chrominance channels up to a factor of two both horizontally and vertically, with no visible difference. The MPEG-1 standard is compatible with 4:4:4, 4:2:2, and 4:2:0 sub-sampling. In 4:4:4, for every 4 Y samples, there are 4 Cr and 4 Cb samples (i.e. each channel has the same resolution). In 4:2:2 for every 4 Y samples there are 2 Cr and 2 Cb samples, which is usually achieved by sub-sampling by two in the horizontal direction and keeping the same resolution in the vertical direction. In 4:2:0 for every 4 Y samples there is 1 Cr and 1 Cb sample, which is achieved by sub-sampling by two in both the horizontal and vertical direction. The most common sub-sampling rate used is 4:2:0 which results in the chrominance signals having a quarter the resolution of the luminance signals. After the colour transform and sub-sampling, each channel is split into blocks and macroblocks. The block size is 8x8 pixels to best suit the use of the properties of the DCT for compression. A number of blocks are then grouped together into a 16x16 pixel macroblock, the size of which is chosen to suit the temporal compression scheme (Section 4.1.3). A 16x16 macroblock is made up of 4 8x8 luminance blocks, and for 4:2:0 sub-sampling 1 8x8 Cr block and 1 8x8 Cb block.

$$\begin{aligned}
 Y &= (0.299R + 0.587G + 0.144B) \\
 Cr-128 &= 0.877(R-Y) \\
 Cb-128 &= 0.433(B-Y) \\
 \text{where } R, G, B &\in [0, 255]
 \end{aligned}$$

Figure 4.5: RGB -> YCrCb colour space transform

Each 8x8 pixel image block is then transformed into the frequency domain by the Discrete Cosine Transform (DCT). The DCT represents a block as the combination of 64 basis functions (see Figure 4.6) in different proportions. These proportions are known as the coefficients $C(u,v)$, where $u=v=0$ is the average or DC value for a block, and $u=v=8$ is the highest horizontal and vertical frequency corresponding to a basis function of 4 cycles per block in each direction. In effect, the DCT divides the image into 64 frequency bands, and as a result an 8x8 by pixel block is now represented by an

8x8 coefficient matrix, as each basis function $C(u,v)$, $u,v=0..7$, will have its own coefficient, resulting in no compression. The compression in fact comes from further processing of the DCT coefficients taking into account some of the statistical properties of the DCT and certain psycho-visual criteria. Generally the DC coefficient, which is always positive, is dominant by some orders of magnitude. The AC coefficients, which can be bipolar, generally decrease as the frequency increases, i.e. from top left to bottom right in the coefficient matrix.



DCT coefficient equation:

$$C(u,v) = \frac{C_u C_v}{4} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n) \phi_{uv}(m,n)$$

DCT basis function:

$$\phi_{uv}(m,n) = \cos\left(\frac{2n+1}{N} v\pi\right) \cos\left(\frac{2m+1}{M} u\pi\right)$$

b)

Figure 4.6: a) DCT basis functions b) DCT equations

Each of the DCT coefficients is then quantised to a discrete level, which is a lossy compression step as the original block will differ from the decoded block, i.e. some noise is added to the block. In effect this quantisation will reduce the number of bits necessary to code the coefficients because there are a finite number of possible coefficient values. Also many of the coefficients, especially at high frequencies, will be quantized to zero, so need not be coded as they carry no information. The harsher the quantization (i.e. the lower the number of discrete levels) the more compression, but also more noise is added to the block. The human visual system is less sensitive to noise at high frequency, so high frequency components can be quantized more coarsely. Using this fact and studies of the distributions of coefficients within video signals has lead to a number of default quantization tables to be defined in MPEG (e.g. a different

quantization table is used for luminance and chrominance blocks). The default quantization tables can be varied from block to block or from frame to frame allowing certain segments to be coded more accurately if the image content requires this, or if a certain bit rate needs to be met. The final compression steps are lossless arithmetic coding steps. As the DC coefficient is dominant, it is coded separately from the AC coefficients using Differential Pulse Code Modulation (DPCM). It also follows that as the DC coefficient is simply the average brightness of a block, it is likely that neighbouring blocks will have similar DC values. DPCM exploits this by coding the first macroblock in a slice absolutely then the difference from block to block is coded for the remaining blocks in a slice. The coding of absolute DC values at the beginning of each slice prevents the propagation of errors. The AC coefficients are reordered using a Zig-Zag scan from the top left to the bottom right of the coefficient matrix. This produces a sequence of sequence of coefficients with a high probability of containing long runs of zero coefficients. As a result the AC coefficients can be coded efficiently using Run Length Coding (RLC). RLC is a series of skip/value pairs containing the value of a non-zero coefficient and the number of zero coefficients to skip before then next non-zero coefficient. The final bit stream is then determined using a variable length coding (VLC) scheme on the AC DCT coefficients and DPCM coded DC DCT coefficients. The VLC uses short binary codes for commonly occurring values and longer codes for rare values.

4.1.3 Temporal redundancy compression (Inter-coding)

The main function of inter coded frames in MPEG (P and B frames) is to take advantage of temporal redundancies in the form of motion compensation. Motion compensation is performed at the macroblock level, and the current block is not coded in its entirety but instead a reference to a similar block in a reference frame and the difference between the current block and the reference block is coded. The residual difference will compensate for any small changes from frame to frame (e.g. a moving object may not maintain its exact appearance while moving). The MPEG-1 standard defines the motion vector as a two dimensional vector $\mathbf{u}(x,y) = [u_x, u_y]$ which gives a reference from the block at position (x,y) in the current frame to the block at position $(x+u_x, y+u_y)$ in the reference frame. Obviously this reference block will most likely not coincide with an actual

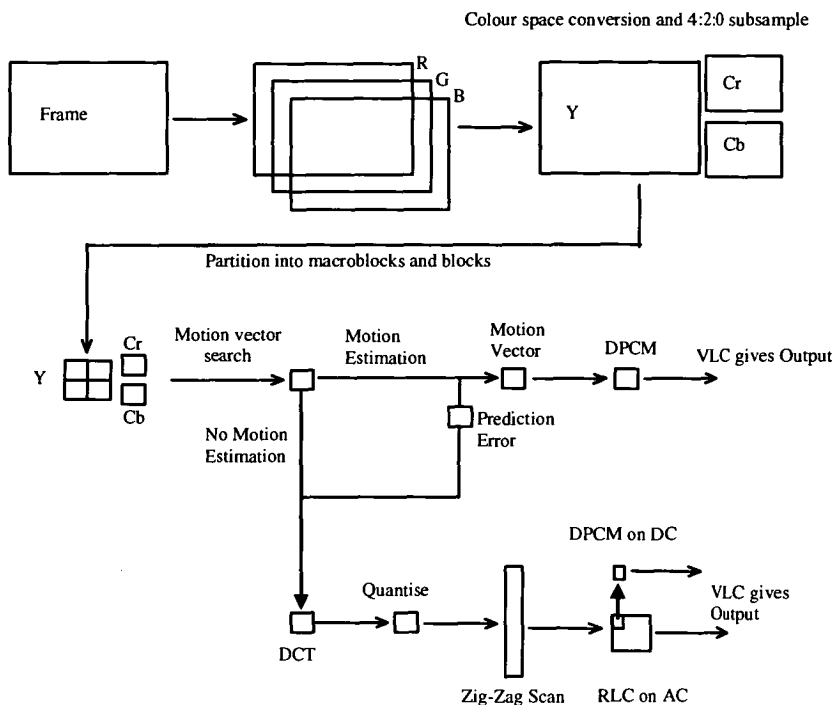


Figure 4.7: Inter-coding process in MPEG-1

macroblock in the reference frame so the pixel data of reference frames is stored in a buffer during the coding process. The reference frames are usually intra-coded and then decoded locally by the encoder before being placed in the buffer because then the encoder and the remote encoder will use the same reference data and the residual will remove any errors. The compression of inter-coded frames relies on the combination of the motion vector and the residual image will be coded with fewer bits than the block being intra-coded. As a result, the smaller the residual (i.e. the closer the reference block is to the current block) the greater the compression. This is why B-frames provide greater compression as they can reference both a past and a future frame it is more likely that a matching block with a small residual will be found.

It can be seen in Figure 4.7 that the first steps in interceding are the same as for intra-coded frames (i.e. colour conversion and sub-sampling followed by partitioning into macroblocks). The macroblock size of 16x16 pixels was determined as a suitable trade-off between a small block size giving a more accurate motion estimation process but resulting in a large number of motion vectors and residuals that need to be coded. For each macroblock a motion vector search will then be performed to find the best

matching block in the reference frame/s. The exact search algorithm is not specified in the MPEG-1 standard, but it is defined that motion vectors can be calculated up to half pixel accuracy by interpolation. Motion vectors are usually estimated in MPEG encoders using rather simple block matching algorithms (BMA). Block matching is only executed on the luminance component of a macroblock but the residual is then calculated for all luminance and chrominance blocks. A block matching technique will examine a search window in a reference frame/s to find the best matching block. The search window is not necessarily square and as motion is more likely to occur in the horizontal direction than vertical [149], rectangular search areas are often used.

The two performance criteria for a motion vector search algorithm are the ability to find the best matching block and also the computation time needed to perform the algorithm. The best match is the block that will give the least difference between the target and reference block and can be found by minimizing an appropriate cost function. The most common cost functions are the Mean Absolute Difference (MAD) or Mean Absolute Error (MAE) in Equation (4.1) and the Mean Squared Difference (MSD) or Mean Squared Error (MSE) in Equation (4.2) which both measure the average pixel by pixel difference of the target and reference block. It should also be noted that other methods could be used to calculate the best matching reference macro block (such as the mean absolute difference, pel difference classification, and integral projection) [150]. With respect to minimizing the cost function, the optimal search technique is known as the full search, in which the current macroblock is compared with every possible macroblock in the search window. In a search window size of 100 x 50 pixels, this would produce 2975 16x16 reference macroblocks, resulting in extensive computations.

$$MAE(u_x, u_y) = \frac{1}{MN} \sum_{m=0}^7 \sum_{n=0}^7 |I_t(x+m, y+n) - I_R(x+m+u_x, y+n+u_y)| \quad (4.1)$$

$$MSE(u_x, u_y) = \frac{1}{MN} \sum_{m=0}^7 \sum_{n=0}^7 |I_t(x+m, y+n) - I_R(x+m+u_x, y+n+u_y)|^2 \quad (4.2)$$

To reduce this computational expense, most coders will use a sub-optimal algorithm to calculate motion vectors. The first technique commonly used is to first test the

difference between the target block and the block in the same position in the reference frame (i.e. a motion vector of $[0,0]$). If this difference is below a threshold then zero motion is assumed and the search process can be avoided. If a motion vector search is needed, a sub-optimal algorithm is usually used, which do not compare the target macroblock with every possible macroblock in search window, but a select few. One technique is to compare the MSD for a set number of evenly spaced macroblocks around the target macroblock. The macroblock with the closest match is then used as the centre of the next level of search, using the same technique in a smaller area. This takes advantage of the principle of locality (macroblocks with very good matches are likely to be close to macroblocks with good matches) in order to select which macroblocks to compare. Some algorithms using this format are the two dimensional logarithmic search, the three step search, and the orthogonal search algorithm [150]. A second sub optimal technique uses down sampling (so there are less pixels in each macroblock) of the original image to perform the first full search. The macroblock with the closest match is then used as the centre of the next full search, in a smaller window, and with a slightly higher sampling rate. This process, called hierarchical motion estimation, can continue a number of levels of down sampling to increase the accuracy of the final motion vector. A final method is to first perform a full search using a less accurate, but computationally inexpensive matching method such as the pel difference classification. The block with closest match is then used as centre for next full search, in a smaller window but using a more accurate matching technique such as the MSD. As a result, the more accurate, but time consuming matching technique is used on less macroblocks. This technique is known as a signature based algorithm [150].

Once the motion vector and residual for each macroblock is found, like the intra-coded blocks, these are further compressed using statistical and arithmetic coding techniques (see Figure 4.7). Like the DC coefficients in I-frames, the motion vectors of surrounding blocks in inter-coded frames are likely to be similar so they are compressed using DPCM, with the slice layer being the start of each DPCM run. The residual images are DCT coded like I-blocks, but the statistics of the DCT coefficients are different resulting in a different quantization scheme and RLC. This is because assuming there is reasonably accurate motion estimation then residual images should

only be coded if the motion vector produces a close match between the target and reference blocks. As a result the residual images will contain mostly high frequency components and the DC and low frequency coefficients of the residual images should be close to zero. This means that the quantization can be much coarser and the default quantization matrix has a dead zone for low inputs causing more coefficients to be coded as zero than in I-blocks. The final bit stream is then determined using a variable length coding (VLC) scheme on the DCT coefficients and DPCM coded motion vectors. The VLC uses short binary codes for commonly occurring values and longer codes for rare values.

4.2 Possible visual features from MPEG domain

As any video file in a digital video database will be stored in a compressed format, it makes sense that any features used in a CBVIR scheme should be calculated using compressed domain information. This is because the compressed domain information can be extracted directly, or with minimal decoding, from the compressed video bit stream. When the MPEG-1 video format is considered it is possible to calculate compressed domain features from three information sources: the frame sequence and block type distributions, the DCT coefficients, and the motion vector fields.

4.2.1 Frame sequence and block type distribution analysis

One of the keys to the success of MPEG as a compression standard is its generality and flexibility to perform in a number of different applications. This is due to the fact that MPEG simply defines the syntax of a compliant bit stream, not the actual implementation of the coding or decoding system. This allows for, and in fact encourages, the concept of a complex but intelligent encoder coupled with a simple decoder. As a result an intelligent coder is flexible to make decisions about the coding process in order to produce the most efficient compliant bit stream. These decisions include quantisation schemes, block types, and direction of reference for motion vectors. These decisions will vary based on video content and editing effects, and the distribution of some of these quantities can be used as an aide to content-based indexing and retrieval.

The most important decision made during the encoding process is the macroblock types used within a frame. Obviously I-frames are coded only with I-blocks so no information can be gleaned from these frames. P- and B-Frames on the other hand can be coded with many different block types, and the following is a brief description of the different types of macroblocks in P and B frames, and some of the common situations when such blocks occur. If a close enough match cannot be found in the motion vector search, then the macroblock can be intra-coded (in the same fashion as all the blocks in an I-frame). These macroblocks occur in both P and B frames and usually occur due to the presence of object motion, as if an object rotates, is deformed in any way or uncovers some background at the edge of the object then a suitable reference macroblock may not be found. If a macroblock does not change between the reference and the frame currently being coded, then the coding of that block can be skipped. This would correspond to finding a reference block in the motion vector search with motion vector (0,0) and error term of 0. These macroblocks occur in both P and B frames usually in low activity areas. Inter-coded macroblocks can be forward coded, backward coded or interpolated. Forward coded macroblocks contain motion vector information referring to a past frame (i.e. the most recent I- or P-frame in the past). These macroblocks occur in both P- and B-frames. Backward coded macroblocks contain motion vector information referring to a future frame (i.e. the most recent I- or P-frame in the future). These macroblocks only occur in B frames. Interpolated macroblocks are coded with reference to both a past and a future frame. This is done by producing the least prediction error after averaging a macroblock in the past reference frame with a macroblock in the future reference frame. These macroblocks only occur in B-frames and can average any noise present between the two reference blocks.

An example of the use of frame and block type distribution can be seen in [151] for abrupt shot detection. By counting the number of interpolated macroblocks within a frame, the extent of correlation between the previous and the next I- or P-frames can be gauged. If this number is below a certain threshold, then it indicates a sudden scene change between these frames. If the number of backward predicted macroblocks for the first B-frame, B_1 , is high then this frame belongs to the future shot and B_1 itself must be the shot boundary. Similarly, if the number of backward predicted macroblocks in the

second B-frame, B_2 , is high, then B_2 is the shot boundary. Finally, if both B-frames have a low count of backward predicted macroblocks, then the shot boundary must be at the next I- or P-frame.

4.2.2 DCT coefficients

In an MPEG sequence each intra-coded macroblock and any error term from an inter-coded macroblock is coded with 4 8x8 DCT blocks for luminance information and 2 8x8 DCT blocks for chrominance information (in the 4:2:0 sub-sampling scheme). As a result any spatial domain colour or texture feature can be calculated from blocks containing this DCT information. Unfortunately, in a typical MPEG sequence this full DCT information is only present in I-frames which typically occur approximately every 10 or 15 frames as they are made up entirely of I-blocks. It is possible to estimate the average block values of inter-coded macroblocks (i.e. the DC DCT coefficient term) with minimal decoding of the bit stream using motion vectors. This is shown in Figure 4.8 where if the DC values of all blocks in a reference frame are known (which is the case for I-frames) then the DC DCT coefficient for the current macroblock in the target frame is the average of the DC coefficients of the up to four macroblocks the reference macroblock overlaps in the reference frame. Using this process an estimated DC image sequence can be calculated from the original MPEG sequence with minimal decoding of the bit stream. It is not possible to estimate the AC coefficients in the same way as these components change very fast both spatially and temporally.

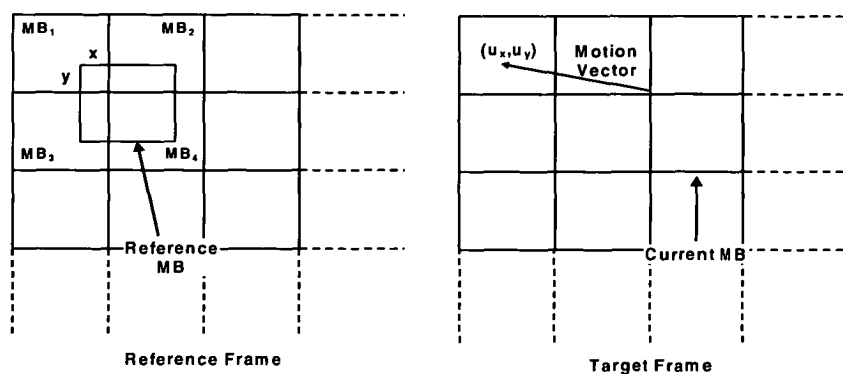


Figure 4.8: Estimation of DC DCT coefficients

It is possible to calculate an estimate of a number of colour and texture features from the DC block estimates. Any grayscale features can be estimated from the 4 four luminance blocks. Colour information however is stored in the chrominance blocks in the form of difference images so cannot be used directly to calculate features. They first need to be transformed into a suitable format such as the RGB colour space [152]. In this thesis only the luminance blocks is used as they carry the majority of information regarding the spatial content.

4.2.3 Motion vector field

The major visual feature which CBIR of digital video can take advantage of which still image CBIR cannot is the motion from frame to frame. The ability to use the motion vector information contained in MPEG bit streams is an obvious choice to exploit for this purpose, but to do so the makeup and possible limitations of the motion vector fields in a variety of video content must be explored first. The first problem when using MPEG motion vector fields to compute visual feature metrics is the presence of motion vectors which are referencing frames of varying lengths apart, in different directions in time, and the irregular spacing of interceded frames. One solution for this is to generate a normalised motion flow for each frame using the motion vector information of surrounding frames [153]. In this technique motion vectors from surrounding frames are transformed into reference frame so that all vectors point in the same direction over the same number of frames. In this thesis as the accuracy of a frame by frame motion description is unnecessary the motion vector information is simply sub-sampled in time by only using the information from P-frames. This also ensures that all motion vectors are pointing in the same direction and over the same number of frames.

Another problem that arises when using raw motion vector fields from an MPEG bit stream is a limitation inherent in the motion vector search itself. As with all of MPEG, the motion vector search is optimized with compression in mind. As a result, the goal of the motion vector search is to find a suitable reference block which will produce a small residual which will take minimal coding. This is usually achieved by using a threshold, so that any reference block which produces a residual error below this threshold will be

classed as suitable, and to minimise computation, the first suitable block reached in the search will be used as a reference.

As a motion vector search is usually confined to a small window to reduce computation, there is a possibility that the search will not find a suitable reference block within the search window. As described above, if the best match has a residual which is expensive to encode, the block will then be coded as an I-block using intra-coding techniques, hence motion flow information (i.e. motion vectors) will not be available in the MPEG bit stream. This will occur if the motion is too fast such that the new location is beyond the search window and therefore the block matching approach only works for slow moving blocks. I-blocks can also be produced when the motion of an object occludes or uncovers an image segment (i.e. near the object boundaries), or when object or camera motion causes image segments to leave or enter the screen (i.e. near the screen boundaries).

4.3 Motion vector field filtering

4.3.1 Regions of unreliable motion vectors in BMA technique

While the presence of I-blocks limits the amount of useful data describing motion within a frame, the main weakness of a BMA technique is the possibility of the search being prematurely terminated at a wrong location. This can occur because the search stops at the first pixel that the MSE is smaller than a given threshold T , and if this first pixel is not where the MSE is at a minimum, then the search gives an erroneous motion vector. This premature termination arises in the following types of image regions:

- In a dark region where the image grey level everywhere is too small resulting in a very small MSE below the threshold T ,
- In a low activity uniform region where the grey level everywhere is almost the same resulting also in a small MSE, and

- In a regular region where there are similar patterns, resulting in unpredictable termination of the LMSE search. (which usually occurs at boundaries between two segments of the first two region types)

It can be seen that in Figure 4.9 (which shows a P-frame and its corresponding motion vector field from a sequence that contains a camera pan to the right) that the majority of motion vectors in the area of frame corresponding to the houses are consistent with a camera pan to the right. This is because this area of frame contains high activity spatial content with many edges and gradients. In the low activity “uniform” areas of the frame, such as the sky and road however the motion vectors are not at all accurate in terms of the actual motion present. These vectors are not reliable data and should not be used in the computation of any motion-based parameters, so a filtering method needs to be devised to separate the reliable from the unreliable motion vectors.

4.3.2 Median filtering approach

A common approach is to use a median filter on the magnitude of horizontal and vertical components of MPEG motion vectors separately to remove random noisy vectors [154], but it can be seen in Figure 4.9 that this does not work in large uniform regions as it is common for a motion vector of (0,0) to occur. A more robust approach is to use some measure of the activity of a block to detect if a premature termination of the motion vector search is likely. A spatial domain solution was presented in [155] using a ‘texture filter’, which simply calculated the number of pixels in a block which had a gradient above a threshold. If this number is below another threshold, then the motion vector for such a macroblock is considered unreliable and discarded from further motion based processing. Following the discarding of unreliable vectors a median filter is used to correct rogue vectors, followed by a smoothing filter to correct for vector quantisation error caused by the median filter. While this approach was suitable for the application creating a panoramic mosaic image from a video sequence in [155] as the decoded frames were available in the spatial domain, it was also stated by the authors that a more efficient algorithm for general purpose applications should work the frequency domain information present in the compressed MPEG bit stream (i.e. the DCT coefficients).

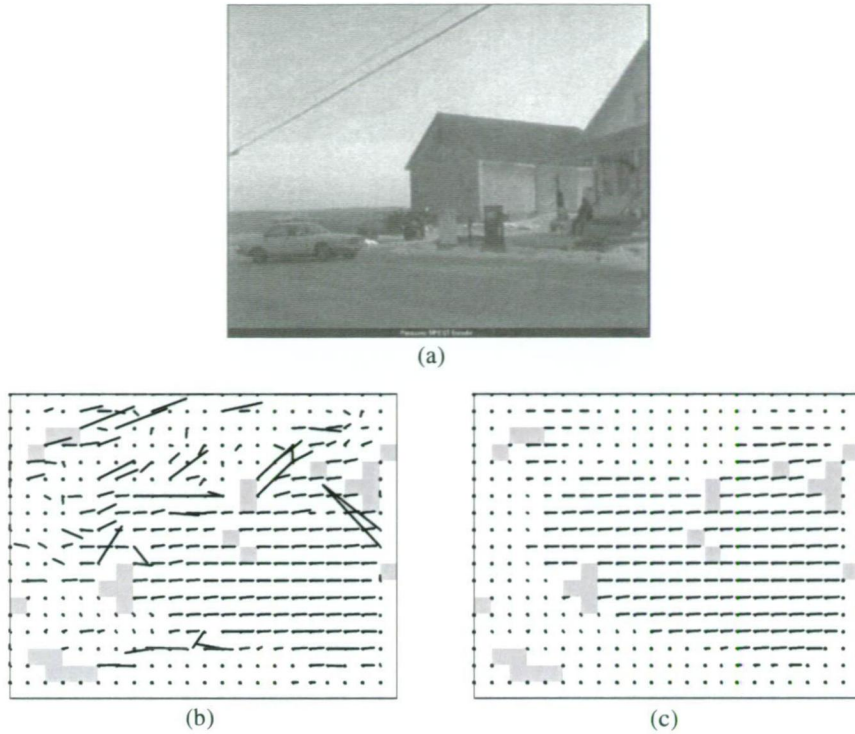


Figure 4.9: Erroneous motion vector field: (a) Frame from MPEG sequence, (b) Motion vector field from MPEG sequence, (c) Median filtered motion vector field

4.3.3 Activity threshold approach

In order to detect low activity, uniform blocks in the frequency domain, the quantitative relationship between the MSE of a block matching (calculated in the spatial domain) and some common frequency domain activity measures for the block has been successfully investigated by the author [156]. In the frequency domain a block with low activity will have small high frequency components (i.e. the energy in the high frequency components will be low). If we consider the target image block I_T in a motion vector search, this can be broken into a dc or average component, and an AC component which is simply the variations from the average.

$$I_T = \{DC_T\} + \{AC_T\} = \{\overline{I_T}\} + \{I_T - \overline{I_T}\} \quad (4.3)$$

Using this notation for I_T and similarly for the reference block I_R , the MSE in Equation (4.2) can be expressed in terms of expectations to better show the power in the image, i.e.

$$MSE = E \left[(DC_T + AC_T - DC_R - AC_R)^2 \right] \quad (4.4)$$

From the brightness constancy assumption it is assumed that the DC level does not change much in a common image block between consecutive frames. As the DC component in an image block is usually dominant compared to the AC component, it is only possible for a block to be a suitable reference block in the motion vector search if the DC level of the target and reference blocks is very similar, and hence the relation in Equation (4.4) can be simplified to:

$$MSE = E \left[(AC_T - AC_R)^2 \right] \quad (4.5)$$

An approximate bound on the maximum of this expression is when AC_R is equal and opposite to AC_T giving the expression

$$MSE \leq E \left[2(AC_T)^2 \right] \leq 4 \times Power_{AC} \quad (4.6)$$

Hence the AC power can be used as a measure of the activity of the block being coded, defined below in Equation (4.7), and Equation (4.6) gives an approximate relationship between the block matching MSE and the block activity measure. From this relationship, it follows that if the AC power is less than a quarter of the threshold T used in the block matching MSE calculations then there is a potential for the motion vector search to be prematurely terminated, therefore the reliability of the motion vector can be inconsistent.

The activity measure used in Equation (4.7) for an intra-coded macroblock is proportional to the AC power of a block and can be very simply calculated using the DCT coefficients present in the MPEG bit stream. As each macroblock consists of 4 DCT blocks luminance information and a number of DCT blocks chrominance information, the activity of each individual block is averaged to estimate the activity of the macroblock. Only the four luminance blocks are used in this process because chrominance signals carry no extra activity information.

$$A(MB_i) = \frac{1}{4} \sum_{b=0}^3 \sum_{u \& v \neq 0} C_{yb}^2(u, v) \quad (4.7)$$

where $C_{yb}(u, v)$ is the AC coefficient corresponding to position (u, v) in the DCT block Yb .

In the case of intra-coded macroblocks, the DCT coefficients are easily extracted from the MPEG bit stream with minimal decoding. Unfortunately inter-frame coded macroblocks (i.e. those with motion vectors encoded), only have DCT coefficients for the residuals directly available, hence the activity of these macroblocks must be estimated from the reference macroblocks using the motion vectors. In this work as motion features are only calculated on a sub-sampled set of frames, namely P-frames, we only consider the approximation of activity of blocks in P-frames. If we consider the frame sequence of a typical GOP in an MPEG stream (Figure 4.3), it can be seen that P-frames are encoded with reference to the previous I or P-frame in the GOP. Hence the activity for each macroblock encoded with a motion vector (i.e. non I-blocks) in the first P-frame in the GOP can be estimated from the I-frame in the GOP (as I-frames are made up completely of I-blocks). The activity of any I-blocks in the P-frame can be calculated directly from their DCT coefficients contained in the MPEG bit-stream.

Therefore, for each macroblock in the first P-frame of the GOP, the reference macroblock from the previous I-frame is found by projecting the motion for that macroblock back on to the I-frame. The reference macroblock may not overlay an exact macro-block in the previous frame. As a result, the activity measure is estimated from the macro-blocks that it does overlay, as shown in Figure 4.8. For each overlaid macroblock there is up to four macroblocks that the reference macroblock can overlay. The approximate activity is calculated using a simple weighted average (i.e. contribution of a macroblock is proportional to the area overlaid in that macroblock) as in Equation (4.8). This will estimate the activity measure for all blocks in the first P-frame of the GOP. The values of activity for all blocks in the first P-frame of the GOP can then be used to similarly estimate the value of activity for each block in the second P-frame of the GOP.

$$\begin{aligned}
A(MB_{ref}) &= \frac{1}{16 \times 16} \sum_{i=1}^4 w_i \times A(MB_i) \\
w_1 &= x \times y \\
w_2 &= (16 - x) \times y \\
w_3 &= x \times (16 - y) \\
w_4 &= (16 - x) \times (16 - y)
\end{aligned} \tag{4.8}$$

The motion vectors of macroblocks with an activity level below a threshold T_a would be discarded from further processing, as the reliability of such macroblocks is not certain. The relation in Equation (4.6) shows the threshold for activity, T_a , is simply a quarter of the threshold T used for the minimum MSE motion vector search in the codec used to create an MPEG file. Unfortunately, the threshold T is not defined in the MPEG standard and thus can vary from codec to codec, and is often not possible to determine directly (especially in proprietary video codecs).

To investigate the effect of the activity threshold T_a on the motion prediction error rate, the motion vector fields from a large number of frames from various MPEG sequences were studied, and motion vectors that did not correspond to actual object or camera motion within the frame were marked as unreliable. Then for varying levels of activity threshold T_a , the percentage of the unreliable motion vectors discarded was calculated, as was the percentage of the correct motion vectors that survived the thresholding (examples of which can be seen in Figure 4.10).

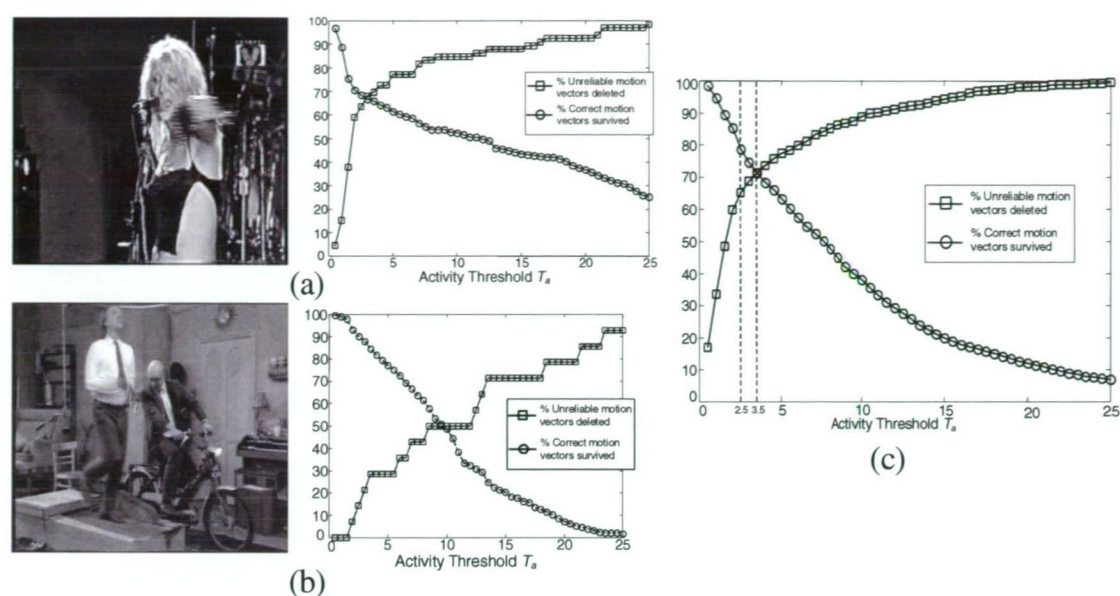


Figure 4.10: Effect of varying activity threshold on (a) Frame with high percentage of dark or low activity uniform regions, (b) Frame with low percentage of dark or low activity uniform regions, (c) A large sample of video frames

From Figure 4.10 (a) it can be seen that the majority of unreliable motion vectors are discarded using a threshold of between 2.5 and 3.5 for the activity, while still keeping the majority of correct motion vectors. This is because the majority of the unreliable motion vectors in this frame are due to the large dark or low activity uniform regions in the frame. Figure 4.10 (b) shows a frame where the majority of motion vectors are due to regular regions where there are patterns, resulting in unpredictable termination of the MSE search. In this case increasing the activity threshold above 3.5 simply discards reliable and unreliable motion vectors at the same rate, consequently increasing the threshold will only lose data rather than filtering out poor data, and further processing would be required to discard these motion vectors. From these observations, it is proposed that prior to using motion vectors in further processing, that those motion vectors from blocks with an activity less than the threshold T_a should first be discarded, leaving a sparse motion vector field, which more accurately represents the actual motion within a frame.

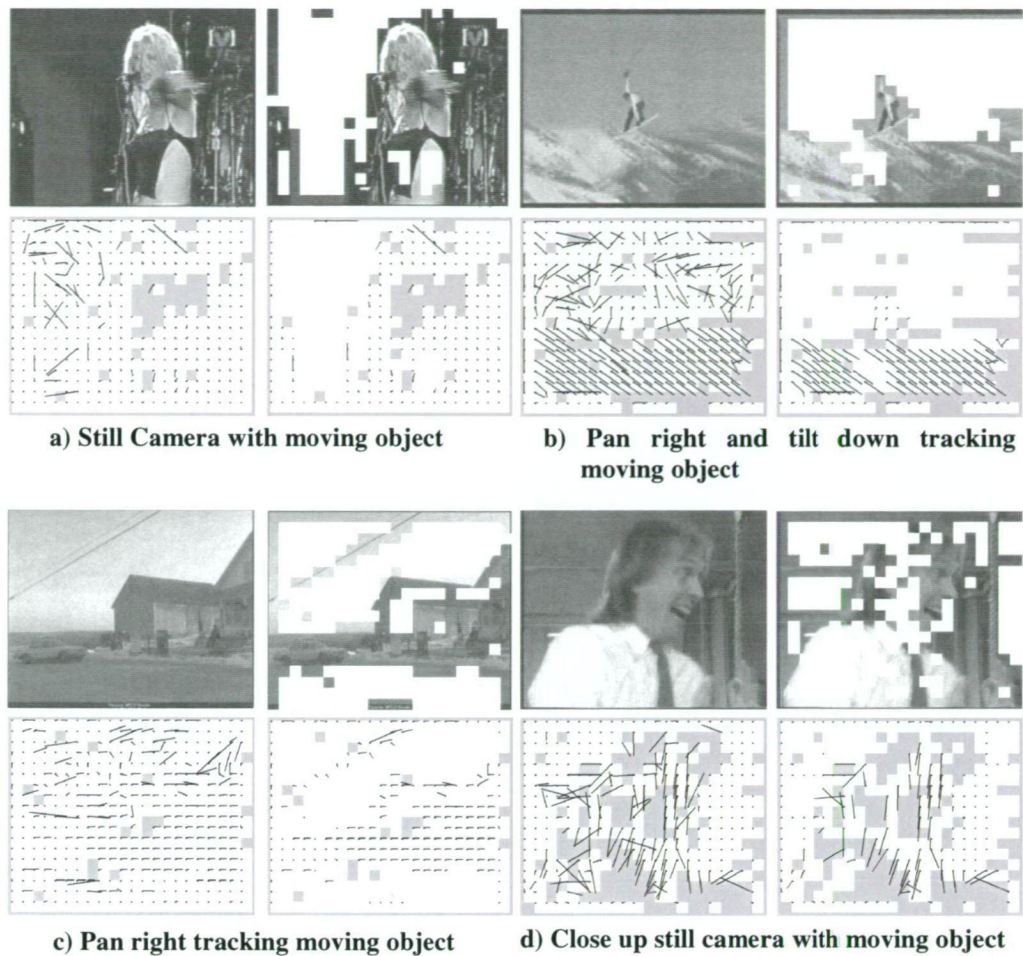


Figure 4.11: Results of motion vector filtering using activity threshold

The effects of the filtering technique are shown in Figure 4.11 for four frames from different MPEG sequences when the activity threshold T_a was set at 3. The figure shows the original frame, the frame with thresholded blocks highlighted in white, the original motion vector field, and the filtered motion vector field. The shaded areas in the motion vector fields correspond to intra-coded blocks. Figure 4.11 (a) shows a frame containing a still background and a moving object. The background on the left half of the screen is a low activity uniform region, and the corresponding motion vector field contains a number of erroneous motion vectors. It can be seen in the thresholded images that almost all of the unreliable motion vectors are deleted, and those left are due to regular patterned regions or the motion of the object. Figure 4.11 (b) shows a frame in which the camera is panning right and tilting down in order to track a moving object in the middle of the frame. It can be seen that the top half of the frame is of low activity,

and the corresponding region of the motion vector field is unreliable. The thresholded motion vector field contains only the small region of low motion in the centre of the frame corresponding to the object, and the motion vectors describing the camera motion in the bottom of the frame. Inaccuracies also exist near the borders of the frame, due to the high camera motion, and also to the black border surrounding the frame. Figure 4.11 (c) displays thresholding of unreliable motion vectors in the sky and road regions, but also shows the inability of this technique to discard erroneous motion vectors in regular or patterned regions. These often occur at the border between two different low activity segments such as the thin black wire or the edge of the roof of the houses in the above frame. Figure 4.11 (d) shows a close up of a moving object with a still background. The object is moving down and to the left in reference to the previous frame. Once again many of the erroneous motion vectors have been deleted. An interesting observation in this frame is the number of I-blocks, many due to the occlusion and uncovering of background near the border of the moving object.

4.4 Conclusion

A major focus of this thesis is the use of visual features calculated in the compressed domain. In particular the MPEG-1 video standard is considered as it is the most prevalent non-proprietary video compression standard used online and techniques developed in this standard can be easily adopted to work on videos of the MPEG-2 and MPEG-4 standard. In this chapter an overview of the various compression techniques used in the MPEG-1 format was given. In particular the focus was on the elements of the compression standard which can be used in the calculation of visual features for use in CBVIR and how these elements vary in the presence of different visual content. These elements include the distribution of frame and block types within a sequence, the DCT coefficients used for spatial compression, and the motion vectors used for temporal compression. Finally, as the motion vectors are calculated to maximise compression rather than describe then motion in a sequence a filtering technique to identify and discard erroneous motion vectors which may not reliably represent the true motion in a sequence was developed. In the following chapter these compression elements are used to calculate visual features suitable for use in the problem of video genre classification.

Chapter 5

Low-level Visual Features

In any CBVIR application the choice of the set of features used to index a segment of video is a very important decision and is highly dependent on the goals of the specific application. In the case of video genre classification the goal is to generate a set of features that will maximise the separation in the input space of the various genres to be classified. To achieve this, the features should be selected to describe the spatio-temporal trends unique to each genre.

The features used to index a video sequence can be calculated from visual and/or audio information. In this thesis the focus is on visual information and an overview of some of the visual features seen in the literature is presented in Section 5.1 . In general, it is evident that the features suitable for video genre classification are low-level global statistical features which describe the broad nature of each genre. It is also evident that many of the features used for video genre classification in the literature are computed in the uncompressed domain which can add unnecessary computational costs to the feature extraction process. With these observations in mind, the remainder of the chapter considers in detail the features used in this thesis, all of which are shot-based and computed using compressed domain information. The first group of features, discussed in Section 5.2, are based on the analysis of the camera and object motion present in a video sequence. This is achieved by applying a parametric camera motion model to a filtered MPEG motion vector field using robust estimation techniques. From this camera motion model a number of features measuring the intensity and variation of camera and object motion in a shot are calculated. The second group of features, discussed in Section 5.3, measure the spatial activity of various block types in a frame using the activity measure in Section 4.3.3. A number of metrics are then computed to describe the amount and variation of spatial activity across a shot.

5.1 Overview of visual features

In Section 2.3.1, an overview of the range of possible visual features that can be used in CBVIR systems was presented. These features aimed to describe a number of characteristics present in a video sequence including colour, texture and motion along with many others. These features could also be calculated over a large range of spatial and temporal scales. For the problem of video genre classification the visual features cannot be designed to be too local as they need to describe the broad spatial and temporal content unique to a broad genre of video clips. A very specific localised feature such as an object detector will not be suitable as such a feature may work well in one genre but fail in others. Instead, it is necessary to use global features that can be calculated over a number of frames or a shot within a video sequence. The majority of visual features used in the literature for video genre classification are calculated in the uncompressed domain. These include features to describe editing properties such as shot lengths [129,131,134] or the percentage of different shot transition types within a clip [131]. There is also a feature in [131] which measures the prevalence of special effects such as lighting changes and flashlights based on comparing the pixel luminance variance to a threshold. Another important visual property for video genre classification is the motion or action within a sequence. In [131] an estimate of the amount of camera motion is calculated using the magnitude of tilt and pan in a frame and then averaged over the frames in a clip. Also used in [131] is a feature which measures the prevalence of static scenes by comparing with a threshold the mean and variance of pixel luminance across frame transitions, and a feature which measures the average length of 'motion runs' which describe the length in time of consecutive active frames within a clip which have a sum of absolute pixel-wise difference between frames above a threshold. The motion metric in [132] is calculated from the frame-to-frame difference in a sequence which is represented across a number of frames within a clip using a DCT of short-term spectral estimates. In [129,133] the motion activity measure used is the mean MPEG motion vector magnitude with no filtering of the motion vector values and in [133] the standard deviation of MPEG motion vector magnitudes is also used. A visual feature used in [134] to represent motion intensity is the mean of the centroid value of a shot's activity histogram from the MPEG-7 temporal distribution of activity parameter. In

[134] five visual features to represent camera motion based on 3-D camera motion parameters from the MPEG-7 Camera Motion descriptor are also used. The five features are the average pan, tilt, zoom, and fixed parameters for a clip and the fifth feature is the sum of the pan, tilt and zoom parameters. A number of colour and texture descriptors have also been used as features for video genre classification. In [129] the colour feature is the average pixel luminance and chrominance, and the texture feature used is the variance of pixel luminance averaged over a shot, with both features calculated from using uncompressed frame pixel data. In [131] three HSV colour space statistics, also calculated in the uncompressed domain, are used. They are the variance of luminance levels calculated from histograms, the average brightness, and the average saturation. The colour and texture features in [133] are generated using MPEG-7 descriptors, including a 16-dimensional *Scalable Colour* descriptor, a 12-dimensional *Colour Layout* descriptor and a 32 dimensional *Homogenous Texture* descriptor. In [134], three MPEG-7 colour descriptors are used to represent colour information in HSV coordinates: the means of Hue, Saturation, and Value in the MPEG-7 Scalable Color descriptor calculated in the uncompressed domain. It is also possible to include audio features, as in [132,133,134], but in this thesis the focus is on visual features only so the specific audio features will not be considered. It should be noted however that results in the literature have in general shown that a combination of visual features with audio features will outperform just visual features in a video genre classification problem.

It can be seen that the features above attempt to model the global trends of the spatio-temporal content of a video sequence across a large number frames. This is done by describing the evolution of properties such as motion, colour, and texture by averaging features across a number of frames within a sequence. It is also evident that many of the features above are performed in the uncompressed domain which can add extra computational time to the feature extraction process. The remaining sections of this chapter present a number of compressed domain shot-based visual features designed for video genre classification, some of which are adapted from the above features, and some of which are new. The features include a number of motion intensity features which describe the evolution of both camera and object motion across a shot, and a number of

metrics to describe the evolution of the average spatial activity of various block types within a frame across a shot.

5.2 Camera and object motion analysis

The most important feature in dynamic scene analysis in various applications, including object tracking and motion-based automatic video indexing and retrieval, is the motion within a video clip. When describing the motions in a video clip it is usually necessary to separate various object motions from the camera motion. As the regions of support for model estimation for object motions are not known *a priori* and usually are not large enough for reliable estimation, it is therefore customary to estimate the camera motion first because the region of support for its estimation is global

5.2.1 Camera motion models

The definition of motion in a video sequence is the movement of points in a scene from one frame to another. This can be seen in Equation (5.1) where the intensity at point (x,y) in the current frame is the same as the point (x',y') in a reference frame. The point (x', y') can be modelled by a transformation $F(x,y,\theta)$ from the point (x,y) due to the motion between the current and reference frames, where θ are the parameters of that motion.

$$I_{cur}(x, y) = I_{ref}(x', y') = I_{ref}(F(x, y, \theta)) \quad (5.1)$$

One simple set of motion parameters, θ , is the optical flow or image velocity field (u, v) from the current frame to the reference frame (in Equation 5.2) which is defined as the observable positional shifts of prominent grey value structures, or simply the difference between the points (x,y) and (x',y') which defines a two dimensional vector for each pixel in the image. The MPEG motion vector field is a sampled version of this, with a single flow vector relating to all the pixels in a macroblock. Equation (5.3) shows this transformation written in the form of Equation (5.1).

$$\theta = (u_x, u_y) = (x', y') - (x, y) \quad (5.2)$$

$$(x', y') = F(x, y, \theta) = (x, y) + \theta = (x, y) + (u_x, u_y) \quad (5.3)$$

A common technique to describe the global motion is to use a parametric 2-D model for the optical flow field, of which the 6-parameter affine flow model, which is used in [157,158,159,160,161,162], is most popular and can be seen in Equation (5.4), i.e. first-order polynomials in the co-ordinates (x, y)

$$\begin{aligned} \begin{bmatrix} x' \\ y' \end{bmatrix} &= F(x, y, \theta) \\ &= \begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \\ &= \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \theta \\ \theta &= [a1 \quad a2 \quad p_x \quad a3 \quad a4 \quad p_y]^T \end{aligned} \quad (5.4)$$

In the spatial domain these camera motion parameters can be found by solving a set of equations, with each equation representing a set of corresponding feature points (x, y) and (x', y') found in separate frames. Other recent techniques for motion estimation use an optical flow approach in the uncompressed domain [160]. First the optical flow is calculated, often using a gradient based approach in the spatial domain, i.e. directly on the decoded video sequence, and the camera motion model is then fitted to the optical flow field. The optical flow method is usually computationally expensive so this research concentrates on compressed domain techniques that make use of data readily available from video files such as motion vectors and DCT coefficients in the MPEG files without the need for decoding these files.

The optical flow vector or image velocity is the solution to a differential equation which cannot be solved unless local gradient or smoothness constraints are known [163]. Under the assumptions that the instantaneous camera motion and the 3-D surface element $dx dy$ imaged at location (x, y) are more or less parallel to the image plane, and that its illumination does not vary substantially between two consecutive frames, we can approximate the optical flow vector by the 2-D motion vector $\mathbf{u}(x, y)$. In fact, a BMA that minimises the sum-of-squared errors (MSE) or the equivalent minimum absolute

difference (MAD) is a region-based matching technique – a more robust alternative to the differential technique, for computing optical flow in noisy or distorted video sequence [163]. Therefore, it seems reasonable to use the motion vectors from an MPEG file as estimates of optical flow vectors, so the camera motion model can be refined to become a function of the MPEG motion vector (u_x, u_y) and the location of the corresponding macroblock (x, y).

$$\begin{bmatrix} u_x + x \\ u_y + y \end{bmatrix} = \begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \quad (5.5)$$

These parameters can be related to the various meaningful camera motion components: pan (horizontal tracking), tilt (vertical tracking), zoom (forward/backward tracking), and rotation of the camera [154] i.e.

$$\begin{aligned} Pan &= P_x & Tilt &= P_y \\ Zoom &= \frac{1}{2}(a_1 + a_4) & Rotation &= \frac{1}{2}(a_3 - a_2) \end{aligned} \quad (5.6)$$

It follows from these relations that the average of parameters a_1 and a_4 provides a single parameter to describe the zooming factor, and the average of parameters of a_2 and a_3 in opposite polarity provide a single parameter to describe the camera rotation. This 4-parameter model (see Equation (5.6)) was shown in [164] to provide very similar results to the 6-parameter affine model. Higher order models can also be used, usually when applied to the more dense optical flow fields, but the scarcity of the motion vector field does not benefit from such models and the affine and 4-parameter models are in fact more resilient to the noise present in these fields.

$$\begin{bmatrix} u_{xi} \\ u_{yi} \end{bmatrix} = \begin{bmatrix} a1 & a2 \\ -a2 & a1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} P_x \\ P_y \end{bmatrix} \quad (5.7)$$

As the motion vectors from an MPEG file are used as the inputs, therefore for a frame of N pixels, we have N sets of linear equations to describe the global motion in the frame. In matrix form we have

$$Y = X\theta \quad (5.8)$$

$$\begin{bmatrix} u_{x1} \\ u_{y1} \\ \vdots \\ u_{xn} \\ u_{yn} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ y_1 & -x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 \\ y_n & -x_n & 0 & 1 \end{bmatrix} \begin{bmatrix} a1 & a2 & Px & Py \end{bmatrix} \quad (5.9)$$

5.2.2 Model estimation techniques

5.2.2.1 Least-squares regression

The estimation of the model parameters θ is a classical regression problem, in which the sample set of N noisy input data $\{\hat{y}_i, x_i\}$, $i = 1..N$, are fitted to a mathematical model (see Equation (5.8)).

$$y_i = f(x_i, \theta) = x_i \theta \quad (5.10)$$

The difference between the observed data input, \hat{y}_i , and the model solution, y_i , known as the residual, needs to be minimised for accurate fitting to occur. It is evident from Equation (5.5) that the input data in the camera motion model is a motion vector so the residual is the magnitude of the difference vector because the horizontal and vertical component of the residual should not be minimised independently. The optimal set of parameters, θ_{opt} , is traditionally calculated using the Least Mean Squared Error i.e. minimising the mean of the squared residuals

$$\theta_{opt} = \arg \cdot \min_{\theta} \left(\sum_{i=1}^N r_i^2 \right) = \arg \cdot \min_{\theta} \left(\sum_{i=1}^N (\hat{y}_i - y_i)^2 \right) \quad (5.11)$$

In order to estimate the model parameters the classical Least Squares (LS) regression technique which the closed form pseudo-inverse solution for θ_{opt} is found setting the derivative of Equation (5.11) to zero. In matrix form this gives

$$\begin{aligned} \frac{d(\hat{Y} - X\theta)^T (\hat{Y} - X\theta)}{d\theta} &= 0 \\ \Rightarrow 2X^T (X\theta - \hat{Y}) &= 0 \\ \therefore \theta_{opt} &= (X^T X)^{-1} X^T Y \end{aligned} \tag{5.12}$$

This technique, whilst being optimal for data contaminated by Gaussian noise, is extremely inaccurate in the presence of outlier data. Unfortunately, in MPEG motion vector fields, this outlier data can be very common due to a number of factors, including object motion and shortcoming in the motion prediction algorithm used. This is acceptable in MPEG coding because the goal of MPEG is to maximise compression, not to describe the true motion of a frame, and any prediction error will be compensated at the decoder. As a result, these outliers need to be discarded, whether by pre-filtering of the motion vector field, or as part of a robust parameter estimation algorithm. While the pre-filtering technique described above in Section 4.2.3 will remove the majority of erroneous motion vectors, any object motion in a video sequence will cause the LMS estimation of the camera motion to fail so robust estimation is also required.

5.2.2.2 M-estimator technique

The purpose of any robust technique is to minimize effect of outliers on estimation by detecting them and discarding from estimation so the final estimation is performed using only inliers (i.e. good points). For example [162,165] use an iterative process in which the residuals are classified as outliers or inliers using a 90% Gaussian confidence interval on the residual error terms which are continually updated until the camera motion parameters settle. The most common technique currently used to deal with data contaminated by outliers is the M-estimator technique which is essentially a weighted least square technique which ‘*weights down*’ the effect of outliers by using an *influence* function [166]. This technique has been used recently in [160] for global motion estimation in MPEG domain and by [167] in the spatial domain. In this technique, instead of minimising the squared residuals, a symmetrical, positive definite function of the residuals $\rho(r_i, \sigma)$ is minimised, i.e.

$$\theta_{opt} = \arg.\min_{\theta} \sum_{i=1}^N \rho(r_i, \sigma) \quad (5.13)$$

The derivative of the function $\rho(r_i, \sigma)$ is called the influence function, ψ , and measures the influence a sample with a certain residual has on the estimation of the parameters. Some common influence functions can be seen in Figure 5.1, of which the L_2 corresponds to the MSE solution, L_1 corresponds to the MAE solution, L_1-L_2 is used in [160], and *Tukey* is used in [167]. The estimation of parameters is usually performed by converting the minimisation in Equation (5.13) into a weighted least squares minimisation. The optimal parameter set is found using an iterative technique in which Equation (5.14) is minimised at each iteration generating a set of residuals from which a new set of weights can be calculated and used in the next iteration and this is continued until convergence or a stopping criteria is met.

$$\theta_{opt} = \min_{\theta} \sum_{i=1}^N w_i r_i^2 \quad (5.14)$$

The weight is found by differentiating Equation (5.13) and Equation (5.14) with respect to r_i giving Equation (5.15). This weight will give the effect of reducing the effect of those points with large outliers, effectively reducing their influence on the estimation of the model parameters.

$$w_i = \frac{1}{r_i} \frac{d\rho(r_i)}{dr_i} = \frac{\psi_i}{r_i} \quad (5.15)$$

	$\rho(x)$	$\psi(x)$	$w(x)$
L_2	$x^2/2$	x	1
L_1	$ x $	$\text{sgn}(x)$	$1/ x $
$L_1 - L_2$	$(\sqrt{1 + x^2/2} - 1)$	$\frac{x}{\sqrt{1 + x^2/2}}$	$\frac{1}{\sqrt{1 + x^2/2}}$
$Tukey \begin{cases} x \leq c \\ x > c \end{cases}$	$\begin{cases} \frac{c^2}{6} (1 - [1 - (x/c)^2]^3) \\ c^2/6 \end{cases}$	$\begin{cases} x[1 - (x/c)^2]^2 \\ 0 \end{cases}$	$\begin{cases} 1 - (x/c)^2 \\ 0 \end{cases}$

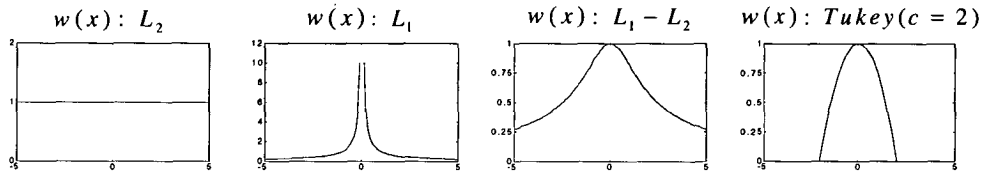


Figure 5.1: Some common M-estimator influence functions

5.2.2.3 Least median-of-squares (LMedS) technique

In [166] Rousseeuw proposes the Least Median-of-Squares (LMedS) technique in which the optimal estimate is obtained by solving the non-linear minimisation problem in Equation (5.16), that is, the optimal estimate gives the smallest value for the median of the squared residuals of the entire data set.

$$\theta_{opt} = \arg \min_{\theta} \text{med}_i(r_i^2) \quad (5.16)$$

As the the input data in the camera motion model is a motion vector the residual is the magnitude of the difference (Equation (5.17)) between the model vector and the MPEG motion vector because the horizontal and vertical component of the residual should not be minimised independently.

$$r_i = \hat{y}_i - y_i = \sqrt{(\hat{u}_{xi} - u_{xi})^2 + (\hat{u}_{yi} - u_{yi})^2} \quad (5.17)$$

The LMedS minimisation problem cannot be reduced to a LS-based solution as for the M-estimators, but instead must be solved by a search in the space of possible p -tuple groups of data points to give the optimum model estimate of p parameters. Since this space is too large, only a randomly chosen subset of m samples of p -tuples can be analysed. The concept behind a LMedS estimator is that for a p -parameter model, in this case the camera motion model has $p=4$, a p -tuple sample is a good fit to the estimator model if the sample contains p good data points (i.e. without outliers). As a result we can choose m p -tuple sub samples to fit the estimator model, and then use the median of the residuals to select the best fit. We must first choose a suitable number of sub samples, m , which will provide acceptable accuracy. Let ε be the fraction of outliers in the data, then the probability of error that can be tolerated by taking only m randomly chosen p -tuples instead of all possible combination of tuples is [166]:

$$P = [1 - (1 - \varepsilon)^p]^m \quad (5.18)$$

Therefore, we must have

$$m \geq \frac{\log P}{\log[1 - (1 - \varepsilon)^p]} \quad (5.19)$$

When choosing exact values for the properties P and ε , there is a trade off between the accuracy needed and the computational requirements. Results presented in this work are achieved using conservative values of $P=0.01$ and $\varepsilon=0.4$, giving $m=33$ sub-samples. The value of ε depends primarily on the extent of object motion in the frame, and a value of 0.4 will mean at least 60% of reliable motion vectors available in a frame will need to correspond to that of the camera motion to ensure less than 1% error in the camera motion model.

To select the m random p -tuples, a Monte Carlo type sampling technique can be used. It is important though that the chosen samples are spread out in the motion vector field to best represent the global motion, as if all samples are in close proximity to each other could lead to an unstable solution. This is achieved in [168] using a regularly random selection method based on a bucketing technique. The activity filtered motion vector field for a frame is evenly divided into 3×3 regions or buckets as in Figure 5.2. Within each bucket is a set of reliable motion vectors, so each p -tuple is generated by selecting p mutually different buckets at random (buckets containing no reliable motion vectors should not be chosen) and then randomly selecting one motion vector from each of the p selected buckets.

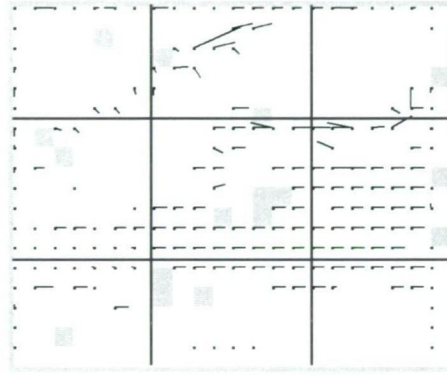


Figure 5.2: Bucketing technique for LMedS estimator

The number of random p -tuples selected, m , is determined by Equation (5.19) on the assumption that all samples (both inlier and outlier points) are distributed evenly throughout the motion vector field. As a result it is necessary that all points have an equal probability of being chosen in the random selection of p -tuples. If each bucket has the same likelihood of being chosen though, those points in buckets with fewer samples have a higher probability of being chosen in a p -tuple. To overcome this, buckets containing a higher number of samples need to be chosen more frequently. This is achieved in [168] by segmenting the interval $[0 \ 1]$ into as many interval as there are buckets containing samples, using the probability of each bucket as a mapping (Figure 5.3). The width of the i^{th} interval is the desired probability, $p(i)$ that the i^{th} bucket is chosen, shown in Equation (5.20) where n_i is the number of sample points in bucket i .

Thus every time a number produced by a $[0 \ 1]$ uniform random number is in the i^{th} interval the i^{th} bucket is selected.

$$p(i) = \frac{n_i}{\sum_i n_i} \quad (5.20)$$

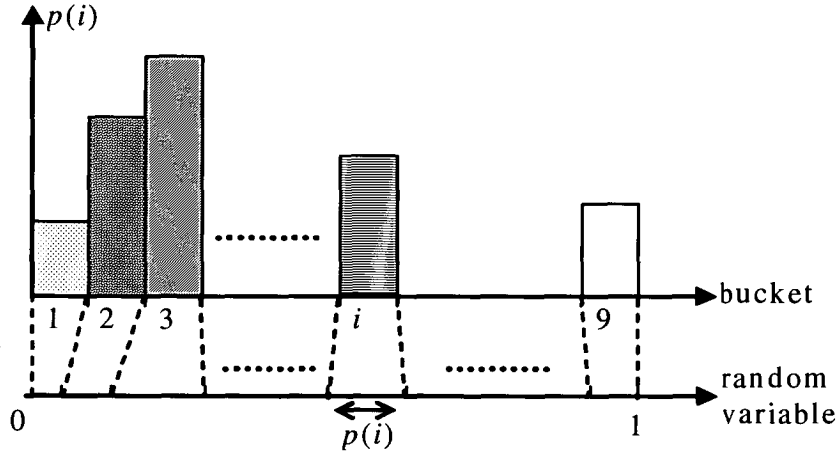


Figure 5.3: Bucket selection probability mapping [168]

For each randomly selected p -tuple sub-sample, j , the LS estimation in Equation (5.12) is used to calculate the model parameters θ_j . The global motion field can then be calculated using Equation (5.7) for every macroblock, i , in the frame containing a reliable motion vector. This is done iteratively, like the M -Estimator approach, with each observation, i , given a weight w_i using a scale estimate σ^* [166] of the dispersion from the median M_j , i.e.

$$w_i = \begin{cases} 1 & \text{if } r_i^2 \leq (2.5\sigma^*)^2 \\ 0 & \text{otherwise} \end{cases} \quad (5.21)$$

where

$$\sigma^* = \sqrt{\frac{\sum_i w_i r_i^2}{\left(\sum_i w_i\right) - p}} \quad (5.22)$$

Thus in each iteration outliers are effectively discarded from data by Equation (5.21) as those observations having zero weight are not used in the minimisation of the residual error. In order to start the outlier identification process in the LMedS technique, we need an initial value for the scale estimate σ^* . This is proposed in [166] as

$$\sigma^0 = 1.4826 \left(1 + \frac{5}{n-p} \right) \sqrt{M_j} \quad (5.23)$$

$$M_j = \text{med}\{r_i^2\} \quad (5.24)$$

in which for each sub-sample j , the median M_j of the squared residuals r_i in Equation (5.24) is calculated from residuals r_i of the first LS- estimate using all data points (i.e. all weights w_i are 1) using the residual given in Equation (5.11). The factor 1.4826 is for consistent estimation in the presence of Gaussian noise, the term $5/(n-p)$ is the finite sample correction factor. Each iteration of Equation (5.21) and Equation (5.22) discards some more outliers and the iteration stops either when σ^* converges to less than 1% of its previous value or when the set limit of say 20 or 30 iterations reached.

The optimal p -tuple from the original m random selections is the one with the minimum sum of weighted squared residuals in Equation (5.25) after the iterative process described above. It is suggested in [169] that once the above LMedS technique has discarded most outlier data points, the technique can be refined by a LS procedure on the remaining mainly inlier data points using the weights from the optimal p -tuple to determine the inliers and outliers.

$$\theta_{opt} = \arg \min_{\theta} \sum_{i=1}^N w_i r_i^2 \quad (5.25)$$

A similar technique for camera motion estimation which also uses the LMedS estimation technique was developed independently in [161]. This technique however, was designed with the application of creating mosaiced background images in mind, so all frames were used in the estimation process, rather than the sub-sample of just P-frames used in this thesis, giving a more dense motion flow field. As a result, unlike in our technique,

no pre-filtering of the motion vectors is used and instead the residuals of the estimation technique are used to determine any erroneous camera motion estimation and these frames are discarded.

5.2.3 Comparison of model estimation techniques

The performance of the LMedS algorithm was compared with that of the robust M-estimator approach presented in [167]. Both algorithms were tested on a number of video clips, and found that in clips with little object motion and frames which had limited low activity (smooth texture) areas (i.e. clips/frames which had a low percentage of outliers) that both approaches produced very similar results. It was observed however, that when the percentage of outliers within a frame was increased, i.e. due to a greater quantity of object motion, the performance of the LMedS estimator was better than that of the M-estimator [170]. Results of a specific 120-frame long shot from a

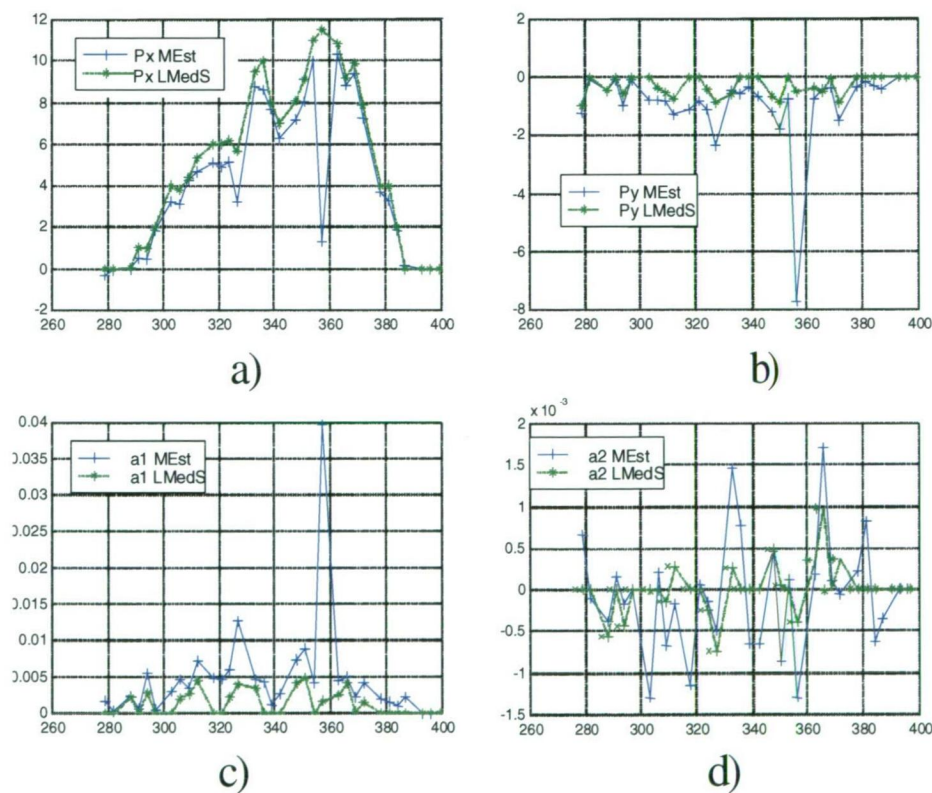


Figure 5.4: Results of camera motion estimation for a video shot containing a pan to the right, tracking a car pulling in to the curb. (a) P_x parameter, b) P_y parameter, c) a_1 (zoom) parameter, d) a_2 (rotate) parameter.

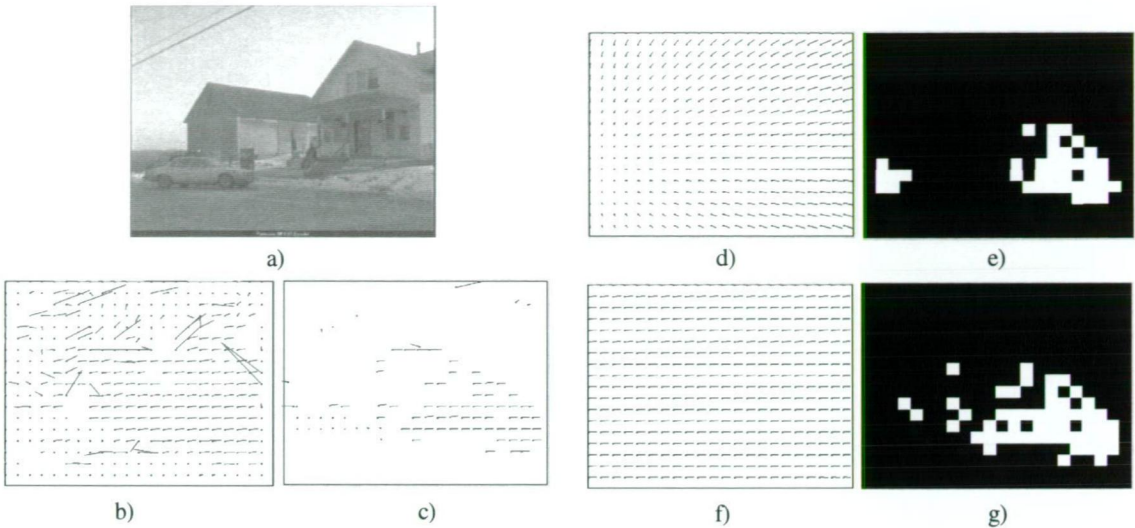


Figure 5.5: Frame from MPEG sequence with high percentage of outliers. a) Actual Frame, b) MPEG motion vector field c) filtered motion vector field, d) Global motion model using M-estimator, e) outlier rejection mask (black blocks are rejected) at final iteration of M-estimator, f) Global motion model using LMedS technique, g) outlier rejection mask at final iteration of LMedS technique.

video sequence are shown in Figure 5.4. The clip contains a pan to the right, as the camera follows a car as it pulls into the curb. Note that only P-frames are used so model parameters are not calculated for every frame in the shot. Figure 5.4 shows the four estimated model parameters for both estimation techniques. The P_x parameter is the dominant parameter (as expected for a shot with a pan to the right) and both techniques produce similar results for this parameter in most frames. In general, the LMedS results are also much smoother in the non-dominant parameters, indicating a more robust estimator in the presence of outliers.

The frame in which there is a large drop in the P_x parameter is further investigated in Figure 5.5. It can be seen in the motion vector field for this frame (Figure 5.5 (b)) that there is large percentage of outliers, primarily in the smooth textured regions. Even after filtering these outliers), there is still a high fraction of outliers due to object motion and at the boundaries of smooth textured regions because there are large areas of smooth texture in the frame (Figure 5.5 (c)). The global motion vector fields in Figure 5.5 (d) and (f) show the estimated global motion using the M-estimator and LMedS techniques

respectively. It can be seen that the LMedS estimator has correctly modelled the global motion as a ‘pan right’, whereas the M-estimator has incorrectly modelled a ‘zoom in’ (with zoom centre near the bottom left of the frame). This explains the large drop in the P_x and P_y parameters, and the large increase in the a_1 parameter (zoom) in Figure 5.4. It can be seen from the outlier rejection masks (white blocks are inliers, black block are outliers) in Figure 5.5 (e) and (g) that the LMedS technique has correctly rejected all outliers, whereas the M-estimator technique has not rejected some of the outliers due to the object motion of the car in the bottom left hand corner. The 0.5 breakdown point of the LMedS technique is mostly avoided due to the pre-filtering of the motion vector field to discard wrongly predicted motion vectors, so is more likely to detect object motion as outliers than the M-Estimator technique which has a breakdown point of less than 0.25.

5.2.4 Camera and object motion metrics

Using the LMedS technique shown in Section 5.2.2.3 a camera motion model can be estimated, and hence a camera motion vector field can be produced for each P-frame in a video shot. From this, a metric which describes the evolution of the camera motion and object motion from frame to frame within a shot can be calculated. This is only calculated for the P-frames because they are generally evenly spaced within an MPEG sequence so are a suitable temporal sample of the sequence, and because all the motion vectors in a P-frame refer to the same reference frame. For a single P-frame, p , the global motion intensity (i.e. the average magnitude of the camera motion vector field) is calculated from the camera motion vector field using Equation (5.26), in which C is the number of macroblocks with reliable motion vectors and $\mathbf{u}_i = [u_{ix} \ u_{iy}]$ is the global motion vector of macroblock MB_i .

$$G_p = \frac{1}{C} \sum_{i=1}^C \sqrt{(u_{ix})^2 + (u_{iy})^2} \quad (5.26)$$

The original motion vector field contains motion vectors due to camera and object motion. To produce an object motion field, the camera motion vector field can be subtracted from the MPEG motion vector field, using only the reliable motion vectors.

The object motion intensity for frame p , O_p , can then be calculated using Equation (5.27), in which C is the number of macroblocks with reliable motion vectors, $\hat{\mathbf{u}}_i = [\hat{u}_{ix} \ \hat{u}_{iy}]$ is the MPEG motion vector of macroblock MB_i and $\mathbf{u}_i = [u_{ix} \ u_{iy}]$ is the global motion vector, to give a measure of the average magnitude of object motion in a frame.

$$O_p = \frac{1}{C} \sum_{i=1}^C \sqrt{(\hat{u}_{ix} - u_{ix})^2 + (\hat{u}_{iy} - u_{iy})^2} \quad (5.27)$$

Once the global and object motion intensities have been calculated for each P-frame within a shot, a measure of the temporal evolution of the distribution of motion intensity across a shot needs to be calculated. This is represented by the second order statistics, the mean μ_G and μ_O , and variance σ_G^2 and σ_O^2 of the motion intensity metrics. This is shown in Equation (5.28) for the global motion of a shot where P is the number of P-frames within the shot.

$$\begin{aligned} \mu_G &= \frac{1}{P} \sum_{p=1}^P G_p \\ \sigma_G^2 &= \frac{1}{P} \sum_{p=1}^P (G_p - \mu_G)^2 \end{aligned} \quad (5.28)$$

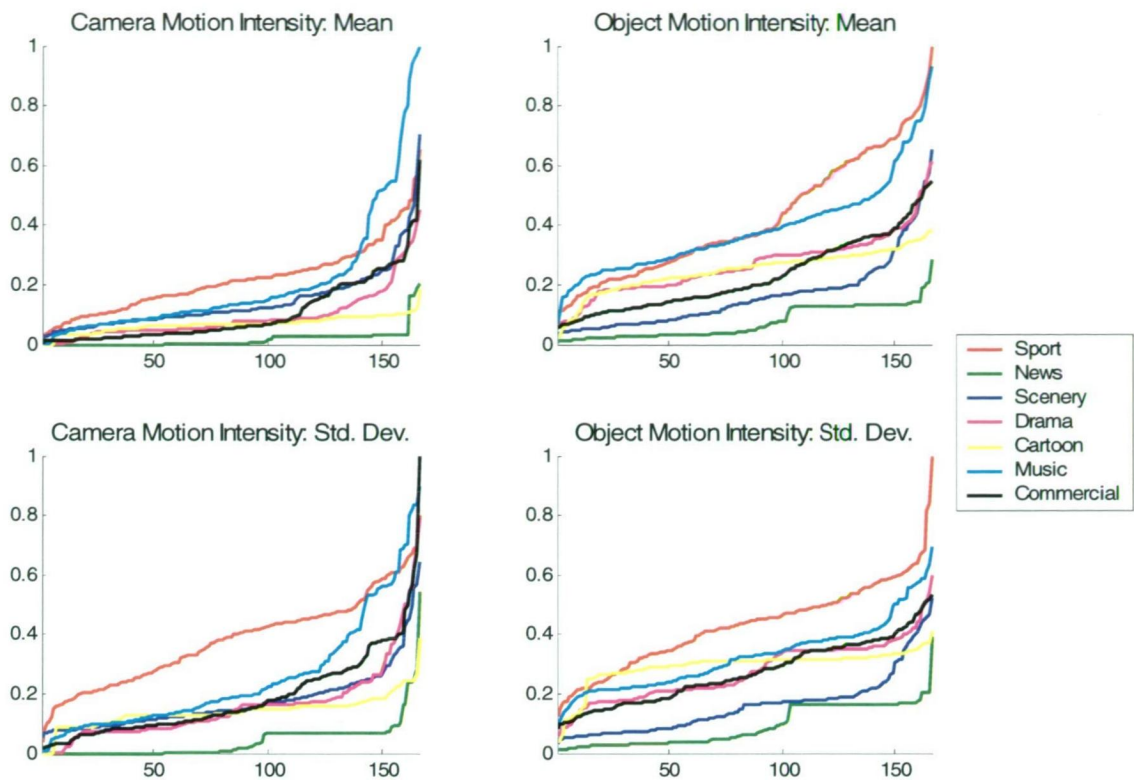


Figure 5.6: Motion intensity feature metrics for samples of each genre {sport, news, scenery, drama, cartoon, music, commercial} in ascending order

In Figure 5.6 the four motion intensity metrics from 166 shots from each video genre {sport, news, scenery, drama, cartoon, music, and commercial} are graphed in ascending order. It is evident that for the sports genre the mean and standard deviation of the camera motion intensity are generally much higher than in other genres. This is because camera motion is used to follow the action occurring in a scene and in the sports genre the action can be fast and erratic resulting large and variable camera motion. In the other genres the camera motion is generally planned, and as such much less erratic, so in general the standard deviation of the camera motion intensity for the music, scenery, drama, cartoon and commercial genres are very similar and much less than shots from the sport genre. There are, however, some music shots that contain erratic camera motion as evident in the graph. The news genre shots contain almost no camera motion resulting in small values for mean and standard deviation of camera motion intensity for this genre. It is also evident that the cartoon genre contains a small amount of camera motion. The music, scenery, drama, and commercial genres contain a large range of

mean camera motion intensity values, although in general shots from the music and scenery genres are slightly higher. It is evident from the object motion intensity graphs that shots from sports genre contain a large amount and also erratic object motion. The object motion in the music genre is also high as they usually contain dancing but the motion is usually less erratic as the intensity of motion is more constant throughout a shot. The scenery and news genres show little object motion as expected. The other genres are similar to each other as expected showing mid-range results.

5.3 Activity power flow

In [156] reverse motion compensation was used to calculate the activity factor (loosely defined as the AC power), of the macro-blocks in an MPEG frame. A relationship was then established between the threshold of the activity factor of a macro-block and the threshold of the mean squared error of a BMA (see Section 4.2.3). This allowed the identification of an unreliable motion vector predicted for a low activity macro-block. This is because a BMA search for a match in the previous reference frame, of a low activity block in the current frame is usually terminated prematurely at a wrong location giving a wrong motion vector. From this, the macro-blocks of an MPEG frame can be classified into three categories: those with reliable predicted motion vectors (C), those with unreliable predicted motion vectors (W), and those that are intra-frame coded (I). The total number of macro-blocks in a frame is therefore

$$N = I + C + W \quad (5.29)$$

The total activity power in one frame in the three categories, P_I , P_C , P_W , is then defined in Equation (5.30), respectively. The average activity factor, AF_b , $b=I,C,W$, for each of the categories in a frame is the average activity of all the blocks from each category in a frame and gives a coarse measure of the spatial content of a frame. By multiplying the average activity by the number of blocks in each category, the power flow will also be dependent on the area taken up by a category in a frame.

$$\begin{aligned} P_I &= I \times AF_I \\ P_C &= C \times AF_C \\ P_W &= W \times AF_W \end{aligned} \tag{5.30}$$

To understand the potential for the *activity power* to characterise a video shot, we need to investigate how the *activity power* of the three categories in any given frame varies depending on different characteristics. P_I is dependent on the number of I-blocks, which occur in P-frames of MPEG bit streams when the residual after motion compensation is too large to code efficiently. This is primarily due to object motion when an object is rotated or deformed, or background is uncovered near the edges of objects. Thus,

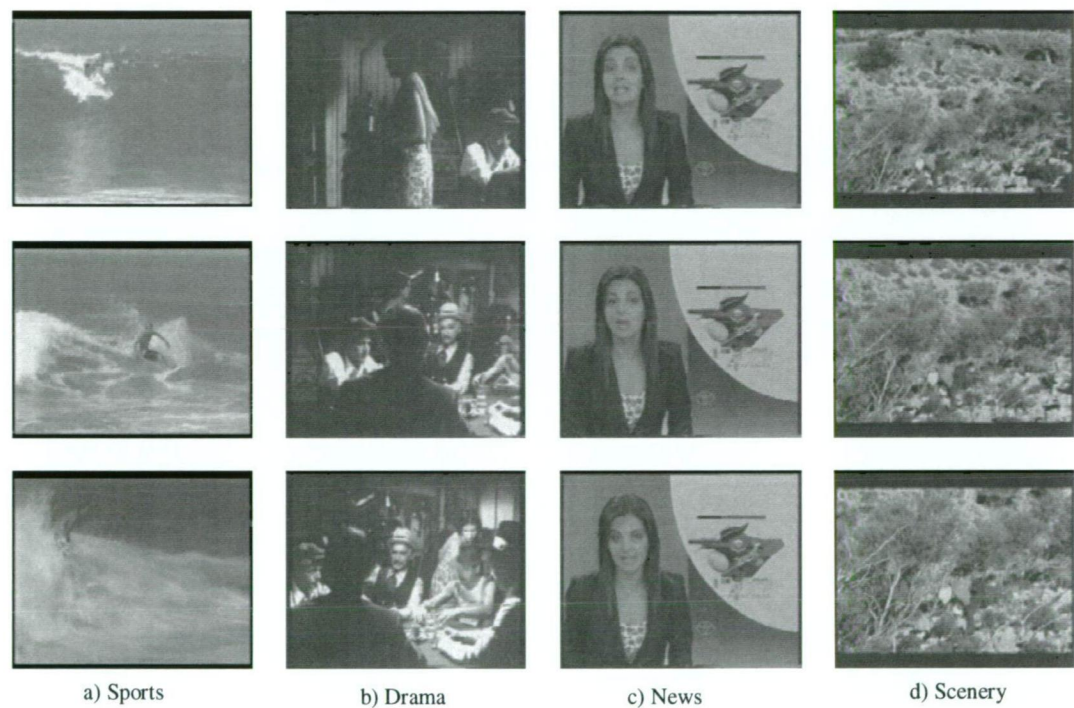


Figure 5.7: The first, middle and last frame in a video shot from each category, displaying the temporal changes within a shot. In sports shots, (a), the camera and object motion is often irregular, shown by the vast differences in frame make up within the shot. The make up of a frame within drama shot, (b), can also change, due to camera and object motion, but the changes are generally much smoother. News shots, (c), usually have no change within a shot, while scenery shots, (d), may change slightly, due to camera panning or zooming, but not object motion.

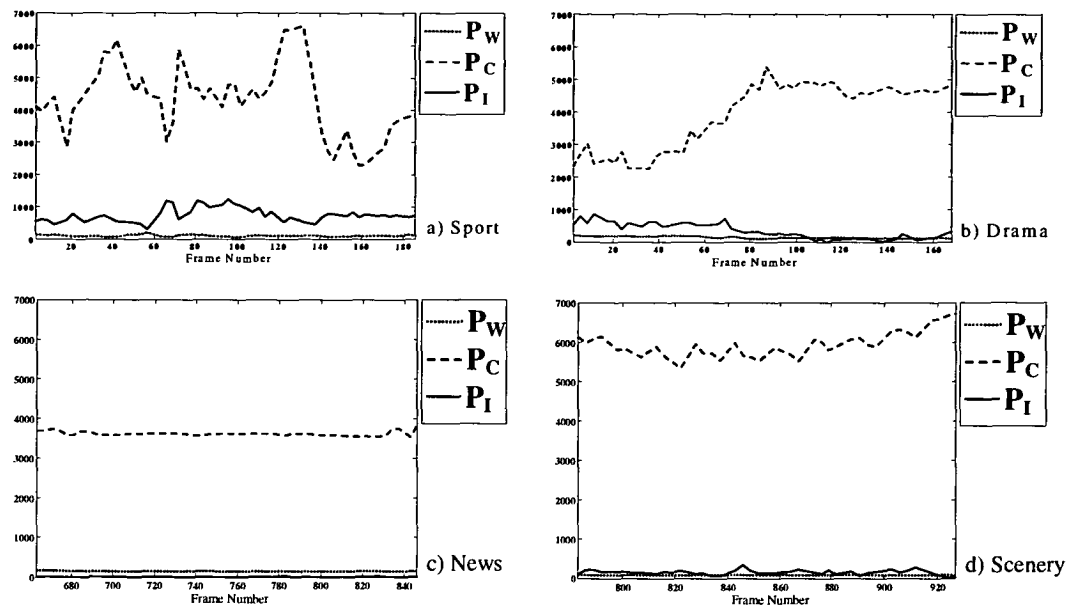


Figure 5.8: Example of activity power flow in the three categories of macro-blocks in an MPEG video shot, (a) Sport, (b) Drama, (c) News, (d) Scenery

frames containing a large quantity of object motion will have a high P_I . The other two categories represent a coarse description of the spatial content of a frame. Frames with many uniform areas will have a large P_W where as frames with more high activity areas will have a large P_C .

The plot of P_b for the sequence of P-frames in a shot (Figure 5.8) gives the *activity power flow* during the shot. Like the motion intensity metrics, this is only calculated for the P-frames because they are generally evenly spaced within an MPEG sequence giving a suitable temporal sample. The *activity power flow* represents the temporal variations of a coarse measure of the spatial content of a video shot. These features represent both the spatial content of a shot, and the motion in a shot, both due to movement of the camera, and also of objects. The use of the *activity power flow* in a CBIR application is based on the premise that the combination of the temporal changes of this flow and the spatial distribution of the frame's activity power between the three macro-block categories is sufficient to characterise the video sequence. Like the global and object motion metrics, the second-order statistics, i.e. the mean μ_b and the variance σ_b^2 , are used as metrics for the activity power flow. The mean gives a coarse measure of the spatial content of a sequence, and the variance represents the camera and object motion.

It can be seen from Figure 5.8 (a) that there are large variations in P_C and P_I , showing that sports shots contain highly irregular camera and object motion. In contrast, Figure 5.8 (b) shows that Drama shots can contain variations in camera and object motion, but the variations are generally much smoother. This is generally explained by the fact that any action in a drama sequence is usually pre-planned, unlike the motion in a sports sequence. Video shots of News (Figure 5.8 (c)) and Scenery (Figure 5.8 (d)) contain very low P_I due to the lack of object motion, but scenery clips generally contain some camera motion resulting in a slight variation in P_C .

In Figure 5.9 the six activity power flow metrics from 166 shots from each video genre {sport, news, scenery, drama, cartoon, music, and commercial} are graphed in ascending order. As the activity power flow metrics are a coarse measure of global spatio-temporal content it is harder to relate to genre specific content than the motion intensity metrics. It is evident however that the cartoon, music and commercial genres in general contain

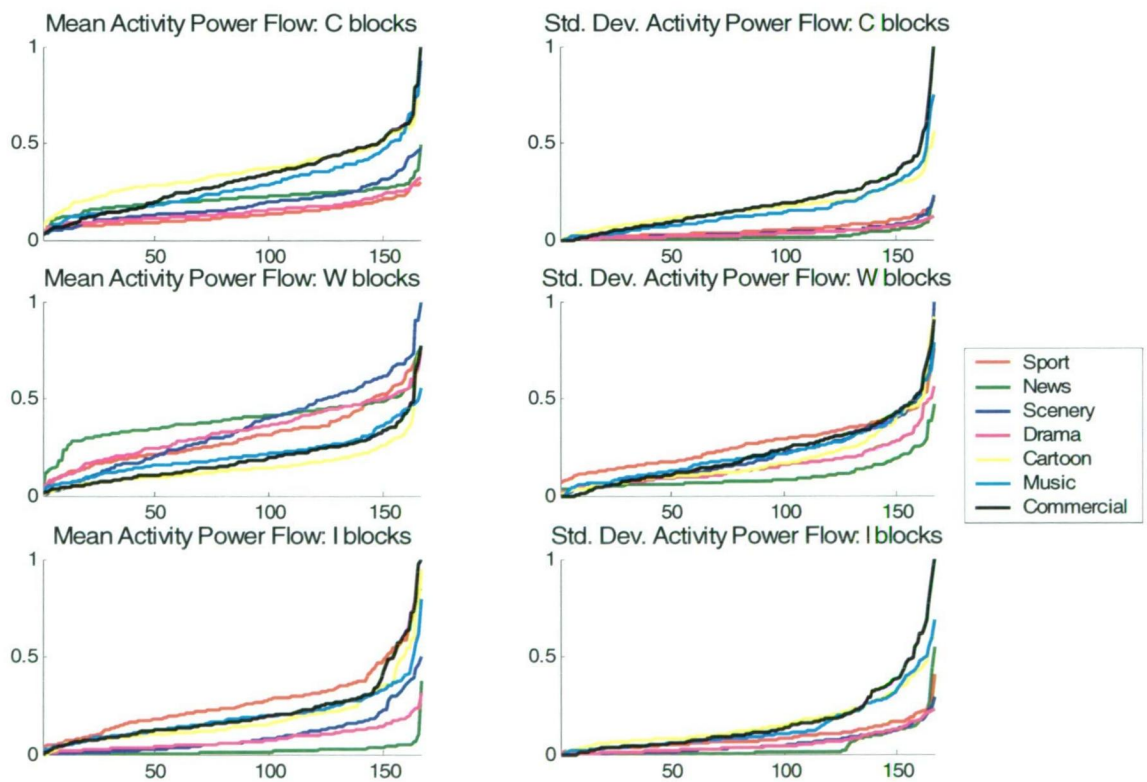


Figure 5.9:Activity power flow feature metrics for samples of each genre {sport, news, scenery, drama, cartoon, music, commercial} in ascending order

larger mean and standard deviation values for the activity power flow of *C* blocks, which indicates more high spatial activity regions in these genres which relates to the more common use of synthetic content as opposed to the more natural content occurring in the other genres. The inverse is true for the activity power flow of *W* blocks as if there are larger regions of *C* blocks there is consequently smaller regions of *W* blocks. The activity power of *I* blocks is an indication of motion which can not be describe by the 2-D motion vectors present in an MPEG file such as rotating objects, object motion which uncovers or covers background, and some computer graphic or special effects. This type of content is most prevalent in the sport, music, cartoon, and commercial genres as evident in the graph.

5.4 Conclusion

In this chapter, an overview of the type of visual features that can be used in the video genre classification problem was presented. In general, it is evident that the features suitable for video genre classification are low-level global statistical features which describe the broad nature of each genre. These features generally provide a coarse measure of the spatio-temporal evolution of properties such as motion, colour, and texture across a number of frames within a video sequence. A particular focus of this thesis is the use of compressed domain information, and a 10-dimensional feature vector to describe a video shot is designed that is calculated solely from information present in an MPEG-1 bit stream. In this feature vector are 4 features which describe the mean and standard deviation of the intensity of camera and object motion within a shot. The other 6 features describe the distribution of spatial activity within a shot, using the mean and standard deviation across a shot of the spatial activity of three block types within a frame: blocks with no motion information, blocks with reliable motion information, and blocks with unreliable motion information. The next step in the video genre classification problem is to transform this 10-dimensional feature vector into a high-level semantic genre label for each video shot, and in the next chapter a number of techniques for this problem are presented.

Chapter 6

High-Level Semantic Classification

The goal of a classification problem is to sort a collection of entities into one of several discrete and usually mutually exclusive classes based on a set of characteristic features. As such, a classification network can be used to perform high level semantic analysis of a system by either classifying new data or by analysing the underlying class structures of the relationship between the features for each class. This task can be viewed as one of pattern recognition and may be defined as either supervised classification in which the input pattern is identified as a member of a class predefined by the system designer, or unsupervised classification in which the pattern is assigned to an unknown class based on the similarity of patterns. The task of classification is thus to discover the fundamental mapping between the input feature space X^N to the output category space C^M

$$f : X^N \rightarrow C^M \quad (6.1)$$

where f is the hidden functional relationship between an input feature or pattern vector $x \in \mathbb{R}^N$ and an output class or concept vector $c = \{0,1\}^M$. In the case of supervised classification, this mapping is estimated using a training set of K noisy observations of input-output pairs $\{x(k), t(k)\}$, $k=1..K$, where $x(k) = [x_1(k) \dots x_n(k) \dots x_M(k)]$ is an input feature vector of N attributes to describe the k^{th} pattern. These attributes can be either continuous or discrete and combined should provide a full description of the entity for the classification task. The target output vector $t(k)$ gives the desired output of the classification network $c(k)$. In this thesis the classification of a pattern into 1 of M classes in which the outputs $c(k)$ are coded using 1 of c coding is considered. In this

coding scheme the output vector $c(k) = [c_1(k) \cdots c_m(k) \cdots c_M(k)]$ is a class or concept label in which $c_m = 1$ if the k^{th} observation belongs to class c_m , and zero otherwise, which can be interpreted as a probability of 1 that the k^{th} observation belongs to class c_m and probability of 0 that it belongs to any other class. In many applications the definition of appropriate classes can be a subjective problem so it is possible to include in the M classes the class ‘doubt’ for patterns which a reliable classification can not be made, and the class ‘outlier’ for patterns which definitely do not belong in any other class.

The four best known approaches for pattern recognition are: 1) *template matching*, in which a pattern is classified based on the similarity between the pattern to be recognized and a stored pattern template or a prototype, 2) *statistical classification*, in which the decision boundaries between classes are determined by the probability distributions of each class with pattern vectors belonging to different classes ideally occupying compact and disjoint regions in an N -dimensional feature space, 3) *syntactic or structural matching*, in which a pattern is viewed as being composed of a hierarchical set of sub-patterns to allow a large collection of complex patterns to be described by a small number of sub-patterns and grammatical rules, and 4) *neural networks*, which attempt to model the processing of the human brain and are composed of many simple processing units, called neurons, in a highly inter-connected network. These approaches are not necessarily independent and some pattern recognition methods exist with different interpretations. Among the various frameworks of pattern recognition developed, the statistical approach is the most intensively studied and used in practice [171]. This approach is discussed in Section 6.1 including details of two approaches that have been applied to video genre classification: Gaussian mixture models [132,133] and classification trees [131,134]. In recent times neural network approaches, such as multi-layer perceptrons (discussed in Section 6.2) and radial basis function networks (discussed in Section 6.3), have become popular due to their capability to learn complex non-linear input-output relationships with a robust learning inference and generalisation from the training data. In spite of what seem to be differences, neural network models have a close relationship to statistical pattern recognition [172,173]. In many applications no single approach for classification is optimal and multiple approaches

have to be used, such as in Section 6.3.4 in which a number of statistical approaches are used to guide the training of neural approaches.

6.1 Statistical classification approaches

6.1.1 Bayes decision theory

The fundamental theory behind pattern classification is the statistical approach known as Bayesian decision theory which states the problem in probabilistic terms. So if the classification problem above is considered, in which a set of K N -dimensional patterns $\mathbf{x}(k)$ are to be classified into one of M classes c_m , then assuming the following *a priori* probability values and probability densities are known

- probability $p(\mathbf{x}(k))$ that a pattern is $\mathbf{x}(k)$
- *a priori* probability $P(c_m)$ that a pattern belongs to class c_m
- conditional probability $p(\mathbf{x}(k)|c_m)$ for pattern $\mathbf{x}(k)$ given it belongs to class c_m

then using the following expressions of the joint probability $P(c_m, \mathbf{x}(k))$ that the pattern is $\mathbf{x}(k)$ and belongs to class c_m

$$P(c_m, \mathbf{x}(k)) = p(\mathbf{x}(k) | c_m)P(c_m) = P(c_m | \mathbf{x}(k))p(\mathbf{x}(k)) \quad (6.2)$$

and considering the following normalisation conditions

$$\sum_{m=1}^M P(c_m) = 1 \quad \text{and} \quad \sum_{k=1}^K p(\mathbf{x}(k)) = 1 \quad (6.3)$$

then the *a posteriori* conditional probability $P(c_m | \mathbf{x}(k))$ that a pattern is class c_m given the pattern $\mathbf{x}(k)$ can be determined using Bayes' theorem i.e.

$$P(c_m | \mathbf{x}(k)) = \frac{p(\mathbf{x}(k) | c_m)P(c_m)}{p(\mathbf{x}(k))} \quad (6.4)$$

If the cost, $C(c_m|c_j)$, of classifying an event as class c_m when in fact it is class c_j is defined, then the conditional risk of classifying pattern $\mathbf{x}(k)$ as class c_m [174] is

$$R(c_m | \mathbf{x}(k)) = \sum_{j=1}^M C(c_m | c_j) P(c_j | \mathbf{x}(k)) \quad (6.5)$$

The optimal classification decision will give the minimum risk (i.e choose class c_m such that $R(c_m|\mathbf{x}(k))$ is minimum for $m = 1..M$) and a common choice of the costs is known as zero-one loss in which $C(c_m|c_j) = 0$ if $m = j$ and 1 otherwise in which the risk simplifies to

$$R(c_m | \mathbf{x}(k)) = \sum_{j=1, j \neq m}^M P(c_j | \mathbf{x}(k)) = 1 - P(c_m | \mathbf{x}(k)) \quad (6.6)$$

and as such the decision c_m providing minimum risk corresponds simply to the class with the maximum probability $P(c_m | \mathbf{x}(k))$.

6.1.2 Discriminant functions and decision boundaries

An alternate way to view a classification problem is to construct discriminant functions which define the boundaries between the classes in the feature space, as a combination of discriminant functions will segment the feature space into regions corresponding to one of the M possible classes. A multi-category classifier can be represented in a number of ways using discriminant functions, but are generally expressed in terms of a set of functions, $g_m(\mathbf{x})$, $m = 1..M$, and will assign a feature vector \mathbf{x} to class c_m if $g_m(\mathbf{x}) > g_j(\mathbf{x})$, $\forall j \neq m$ [174]. This will result in a number of decision regions R_m and decision boundaries ∂R_m such that

$$\begin{aligned} R_m &= \{\mathbf{x}; g_m(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq m\} \\ \partial R_m &= \{\mathbf{x}; g_m(\mathbf{x}) = g_j(\mathbf{x}) \quad \forall j \neq m\} \end{aligned} \quad (6.7)$$

Bayes theorem shows that a suitable discriminant function is $g_m(\mathbf{x}) = P(c_m|\mathbf{x}) = p(\mathbf{x}|c_m)P(c_m)$, but any monotonic function of $g_m(\mathbf{x})$ will suffice, i.e. the natural logarithm

$$g_m(\mathbf{x}) = \ln P(c_m | \mathbf{x}) = \ln p(\mathbf{x} | c_m) + \ln P(c_m) \quad (6.8)$$

The decision boundary between two contiguous regions R_i and R_j is $g_i(\mathbf{x}) = g_j(\mathbf{x})$. For the case where the class conditional probability $p(\mathbf{x}|c_m)$ is an N -dimensional Gaussian distribution with mean vector $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$ with determinant $|\boldsymbol{\Sigma}_m|$

$$p(\mathbf{x} | c_m) = \frac{1}{(2\pi)^{N/2} |\boldsymbol{\Sigma}_m|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{x} - \boldsymbol{\mu}_m)\right] \quad (6.9)$$

$$\begin{aligned} \boldsymbol{\mu}_m &= E[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x}) d\mathbf{x} \\ \boldsymbol{\Sigma}_m &= E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \int (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

then the resulting discriminant function, ignoring constant terms, is [191]

$$g_m(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{x} - \boldsymbol{\mu}_m) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_m| + \ln P(c_m) \quad (6.10)$$

6.1.3 Probability density estimation

It can be seen from the previous sections that in order to design a classifier an estimate of the probability density function for each class c_m is needed. The possible approaches are to model the *a posteriori* probabilities $P(c_m|\mathbf{x})$ directly, to model the class-conditional probabilities $p(\mathbf{x}|c_m)$ and combine with the prior probabilities $P(c_m)$ using Bayes theorem to estimate $P(c_m|\mathbf{x})$, or to determine a set of discriminant functions and decision boundaries directly. In [191] the various approaches are grouped into parametric, non-parametric, semi-parametric techniques. In parametric estimation techniques, probability distributions are modeled by an assumed functional form which is governed by a few free parameters (e.g. the Gaussian distribution which is described by the mean and variance parameters). The free parameters are learnt from training data using maximum likelihood estimation or Bayesian inference and quite often have a 'physical meaning' to problem. Parametric techniques are usually quick and easy to construct so a number of setups can be compared, but if the wrong model family is chosen the model can not be fitted exactly to the distribution which is known as model bias error. In non-parametric techniques, no assumption is made about the underlying model of the probability distribution so there is no model bias error. Instead many free parameters are estimated, with the parameters usually having no physical meaning to the

problem. Non-parametric models are complex to construct and sensitive to the training set used, which is known as model variance error. This stems from the possibility of overtraining a model to a particular training set resulting in an inability of the model to generalise to unseen examples. In general non-parametric techniques attempt to model a probability distribution $P(\mathbf{x})$ by estimating a multi-dimensional histogram from the training data by determining the number of samples K that fall into the multidimensional volume V compared to the total number of training samples N (i.e $P(\mathbf{x}) = K/NV$). The approaches used are of two types: those which fix the volume V and then find K , such as Kernel Methods, and those which fix K and then find V such as K -Nearest Neighbour methods. The disadvantage of non-parametric methods is that typically the number of parameters grows with the data set so models can become large and hard to deal with. Semi-parametric methods, which can be considered a subset of non-parametric methods, attempt to combine the benefits of both parametric and non-parametric approaches in order to find a suitable trade off between the model bias and model variance errors. A semi-parametric technique uses a general model of the probability distribution which is more flexible than in a parametric approach. Also, the number of free parameters controlling the model can be varied to suit the complexity of the problem rather than the number of training samples. Examples of semi-parametric methods which have been applied to the video genre classification problem include Gaussian mixture models and classification trees. Other techniques which can also be regarded as semi-parametric probability density estimators are Artificial Neural Networks and Radial Basis Function Networks, discussed in Section 6.2 and 6.3.

6.1.4 Gaussian mixture models

A Gaussian mixture model is a semi-parametric approach to estimate the probability density $p(\mathbf{x}|\lambda)$, for a set of model parameters λ , from a set of training samples $\mathbf{x}(k)$, $k=1..K$. In a Gaussian mixture model the density $p(\mathbf{x}|\lambda)$ is formed by the linear combination of H mixture densities, where H is typically much less than the number of training samples K , and each mixture component j is an N -dimensional Gaussian distribution $p(\mathbf{x}|j)$ with a mixing coefficient $P(j)$, i.e.

$$p(\mathbf{x} | \lambda) = \sum_{j=1}^H p(\mathbf{x} | j) P(j) \quad (6.11)$$

where $p(\mathbf{x}|j)$ is given in Equation (6.9), so the set of model parameters λ contains the set of mixing coefficients, means and covariance matrices for each mixture component j (i.e. $\lambda = \{\{P(j)\}, \{\mu_j\}, \{\Sigma_j\}\}, j = 1..H$). To normalise the mixture model, the sum of mixing coefficients $P(j)$ should sum to 1 and each coefficient should lie between 0 and 1. Similarly the area under each component density $p(\mathbf{x}|j)$ should be 1 so each component can be regarded as a class-conditional density. As such, using Bayes rule, the *a posteriori* probability that mixture component j was responsible for \mathbf{x} is

$$P(j | \mathbf{x}(k)) = \frac{p(\mathbf{x} | j) P(j)}{\sum_{j=1}^H p(\mathbf{x} | j) P(j)} \quad (6.12)$$

The most common approach used to learn the set of model parameters in λ is the expectation-maximisation (EM) algorithm [175]. This is an iterative process which begins with an initial set of model parameters λ^{old} from which a new model λ^{new} can be estimated. This new model then becomes the old model λ^{old} for the next iteration and a new set of parameters using the following re-estimation formulas

$$\begin{aligned} P^{\text{new}}(j) &= \frac{1}{K} \sum_{k=1}^K P^{\text{old}}(j | \mathbf{x}(k)) \\ \mu_j^{\text{new}} &= \frac{\sum_{k=1}^K P^{\text{old}}(j | \mathbf{x}(k)) \mathbf{x}(k)}{\sum_{k=1}^K P^{\text{old}}(j | \mathbf{x}(k))} \\ (\sigma_j^{\text{new}})^2 &= \frac{1}{N} \frac{\sum_{k=1}^K P^{\text{old}}(j | \mathbf{x}(k)) \mathbf{x}(k)^2}{\sum_{k=1}^K P^{\text{old}}(j | \mathbf{x}(k))} - (\mu_j^{\text{new}})^2 \end{aligned} \quad (6.13)$$

According to [176] there are no good theoretical means to find the initial model parameters λ and the number of mixture models H so they should be determined experimentally for a given application. In an M class classification problem a separate

mixture model $p(x|\lambda_m)$ can be estimated for each class c_m using only the training samples belong to that class. As such a sample $x(k)$ will be classified as the class c_m with the maximum mixture model probability $p(x|\lambda_m)$.

6.1.5 Classification and regression trees

A classification or regression tree [177,178] (which has been applied in video genre classification in [131,134]) recursively splits the input space into two along boundaries parallel to the axes of the input feature vector, i.e. each split is a binary split in an attribute component resulting in two clusters of data. This approach separates the instance space into hyper-rectangular regions (or nodes in the tree), in which ideally each region, or cluster, contains only inputs of the same class.

6.1.5.1 Growing the tree

In the example classification problem of M classes with K observations from an N -dimensional input space, a tree is grown from a root node, which is the smallest hyper-rectangle that encloses the set of K examples in the instance space, S . The root node is an N -dimensional hyper-rectangle with size (half width), s_n , and centre, c_n , along the n^{th} (attribute) dimension, i.e.

$$s_n = \frac{1}{2} \left(\max_{k \in S} (x_{kn}) - \min_{k \in S} (x_{kn}) \right) \quad (6.14)$$

$$c_n = \frac{1}{2} \left(\max_{k \in S} (x_{kn}) + \min_{k \in S} (x_{kn}) \right) \quad (6.15)$$

In a classification problem the tree is grown by splitting each node into two children nodes using the bifurcation which produces the purest children nodes possible. A split, sp , of a tree node, j , divides the instance space of that node, S_j , into left, j_L and right, j_R subsets (child nodes) on either side of a boundary b_n in the n^{th} attribute dimension such that

$$\begin{aligned} S_{j_L} &= \{S_j : x_n \leq b_n\} \\ S_{j_R} &= \{S_j : x_n > b_n\} \end{aligned} \quad (6.16)$$

The proportions of each class, c_m , in a node can be thought of as the conditional probability $P(c_m|S_j)$ which sum to 1 over the $m = 1..M$ classes. The impurity of a node, j , is then defined [177] as $i(j) = \varphi(P(c_m|S_j))$ where φ is a non-negative function which is maximum when all classes are equally likely, and zero when only one class exists in a node i.e. the node is perfectly pure. One such function is the entropy of a node [177], i.e.

$$i(j) = -\sum_{m=1}^M P(c_m | S_j) \log_2 P(c_m | S_j) \quad (6.17)$$

The goodness of partition sp of node j is defined in [177] as the reduction in impurity

$$\Delta i(sp, j) = i(j) - \frac{K_{j_L}}{K_j} i(j_L) - \frac{K_{j_R}}{K_j} i(j_R) \quad (6.18)$$

where K_j , K_{j_L} , and K_{j_R} are the number of input examples in the current, left child, and right child nodes respectively. The best partition, sp^* , for node j is the one that gives the largest reduction of impurity over the set of all possible partitions, SP , i.e.

$$\Delta i(sp^*, j) = \max_{sp \in SP} \Delta i(sp, j) > 0 \quad (6.19)$$

So sp^* can be found by a simple discrete search over all N attributes and all K_j examples in the instance space of node j , and it is obvious that the impurity of a node never increases when the node splits. This process of splitting nodes starts at the root node and repeats on the children nodes to grow the tree to grandchildren nodes and so on. A family branch stops growing, and the node becomes a terminal node (a node without children), if the resulting node has a remaining instance space of fewer than K_{min} examples, or if there is no further significant reduction in impurity.

For any terminal node, j_t , in the set of terminal nodes, \tilde{T} , that node will be assigned to a class, c_{m^*} , in such a way that the cost or risk of misclassifying a sample, $r(j_t)$, is minimised. Like the conditional risk function applied to Bayes theory in Equation (6.10), a set of conditional misclassification costs, $C(i|c_m)$, are usually given or known a priori. As such the expected misclassification cost $r(j_t)$ for the class i will be

$$r(j_i) = \sum_m C(i | c_m) P(c_m | j_i) \quad (6.20)$$

and the node will be labelled as the class $c_{m^*} = i$ which minimises the above. In the case of unit misclassification, $C(i | c_m) = 1$ for $i \neq m$ then class label, c_{m^*} , will be that of the majority of samples within that node as the misclassification cost reduces

$$r(j_i) = 1 - P(c_{m^*} | j_i) \quad (6.21)$$

The probability of a sample falling into terminal node j_i is $P(j_i) = K_{j_i} / K$, so the node misclassification rate, $R(j_i)$, and hence the misclassification rate of the whole tree, $R(T)$ are respectively

$$R(j_i) = r(j_i) P(j_i) \quad (6.22)$$

$$R(T) = \sum_{j_i \in \tilde{T}} R(j_i) \quad (6.23)$$

and it can be shown simply that the overall misclassification rate of a tree never increases when a node splits[177].

6.1.5.2 Tree pruning

When a fully grown tree, T_{max} , is grown so that the terminal nodes will contain a minimum number of examples, K_{min} , it is likely that the tree will be larger than necessary for the data, and hence while the misclassification rate, $R(T)$, on the training set decreases as the tree size increases, it is possible that the tree is overfitting the data, resulting in the actual expected classification rate, on a different test set, $R^*(T)$ will be lower. In order to come up with the best sized tree, to maximise $R^*(T)$, there are two options: either use a stopping criterion to inhibit the tree from becoming too large, or, more commonly, to prune a tree to remove the overfitting. A number of pruning algorithms have been proposed, as surveyed in [179], one of which is minimum cost-complexity pruning [177].

In minimum cost-complexity pruning, the cost-complexity measure, $R_\alpha(T)$ of a sub-tree $T < T_{max}$ is a linear combination of the misclassification cost, $R(T)$, and a cost for greater tree complexity $|\tilde{T}|$, defined as the number of terminal nodes in T , weighted by a real number complexity parameter, α , i.e.

$$R_\alpha(T) = R(T) + \alpha |\tilde{T}| \quad (6.24)$$

The ‘best’ sub-tree, $T(\alpha)$, can then be found for any value of α by minimising Equation (6.24), such that if α is small, $T(\alpha)$ will be large, and vice versa. Although the value of α is continuous, because there are a finite number of sub-trees of T_{max} , there is a sequence of minimising sub-trees $T_1 > T_2 > \dots > T_k \dots > \{t_l\}$, where $\{t_l\}$ is the root node of T_{max} , and each tree, T_k , is the minimising sub-tree for the range of α values $\alpha_k \leq \alpha < \alpha_{k+1}$. To start the pruning process, rather than using T_{max} , the minimising sub-tree for $\alpha = 0$ (i.e. $T_l = T(0)$) is used. As such T_l is the smallest sub-tree of T_{max} where $R(T_l) = R(T_{max})$.

If the minimising sub-tree T_k and the corresponding complexity parameter α_k is known (i.e. initially $T_l = T(0)$ and $\alpha_l = 0$) then the next minimising sub-tree T_{k+1} and complexity parameter α_{k+1} can be found. Consider the branch T_t of T_k which has node t as its root, then the misclassification cost $R(T_t)$ can be calculated using Equation (6.23) for the set of terminal nodes \tilde{T}_t . If $\{t\}$, the sub-branch of T_t containing only the node t , is considered then we have following cost-complexities

$$R_\alpha(\{t\}) = R(t) + \alpha \quad (6.25)$$

$$R_\alpha(T_t) = R(T_t) + \alpha |\tilde{T}_t| \quad (6.26)$$

While $R_\alpha(T_t) < R_\alpha(\{t\})$, the branch T_t is preferable to that of $\{t\}$, but as α increases there will be a point when the cost-complexities are equal, and the sub-branch $\{t\}$ will become preferable. The value of α where this occurs can be found using the following function $g_k(t)$, for all nodes t in T_k .

$$g_k(t) = \begin{cases} \frac{R(t) - R(T_k)}{|\tilde{T}_k| - 1}, & t \in T_k, t \notin \tilde{T}_k \\ +\infty, & t \in \tilde{T}_k \end{cases} \quad (6.27)$$

The weakest link, \bar{t}_k , in T_k is the first node as α increases that node t is preferable to branch T_i i.e. the minimum $g_k(t)$ for all t . Hence α_{k+1} and the minimising sub-tree T_{k+1} at this value of α are respectively

$$\alpha_{k+1} = g_k(\bar{t}_k) = \min_{t \in T_k} g_k(t) \quad (6.28)$$

$$T_{k+1} = T_k - T_{\bar{t}_k} \quad (6.29)$$

Note that if at any stage in the process there are multiple ‘weak links’, i.e. two nodes \bar{t}_k and \bar{t}_k' with equal $g_k(\bar{t}_k)$ and $g_k(\bar{t}_k')$ then the next minimising sub-tree T_{k+1} is

$$T_{k+1} = T_k - T_{\bar{t}_k} - T_{\bar{t}_k'} \quad (6.30)$$

This process is continued to generate a decreasing sequence of sub-trees until the minimizing sub-tree $\{t\}$, or the root node of the original tree, is reached. If the minimum misclassification cost $R(T_k)$ is used to select the best sub-tree from the sequence produced above, the largest tree T_l will always be chosen so an estimate of the expected misclassification cost, $R^*(T_k)$, for each tree T_k is needed. The best technique for estimating these expected costs when the training set is not large is cross validation. In V -fold cross validation, the training set is divided into V randomly selected subsets with each subset containing a distribution of classes as close to the original training set as possible. Each subset can be used as an unseen test set to estimate the expected misclassification cost of a tree grown using the remaining training samples. So, along with the original tree, T_{max} , grown on of the original training set, V auxiliary trees, $T_{max}^{(v)}$, $v = 1..V$ are also grown. Each of these trees can be pruned using minimum cost complexity pruning, giving a sequence of trees $\{T(\alpha_k)\}$ and $\{T(\alpha_k^{(v)})\}$ with corresponding sets of complexity parameter $\{\alpha_k\}$ and $\{\alpha_k^{(v)}\}$. If V is suitably large, the misclassification cost of a tree $T(\alpha)$ should be almost the same as auxiliary tree $T^{(v)}(\alpha)$ so the expected misclassification cost $R^*(T_k)$ of $T(\alpha_k)$ can be estimated by averaging the

misclassification costs across the V training sub-samples. As such the best sized pruned tree is the tree from set of minimum cost complexity trees of T_{max} with the minimum expected misclassification cost $R^*(T_k)$ and if more than one tree has the same minimum value, the smallest tree is chosen.

6.2 Artificial neural networks

Another approach to pattern recognition is an artificial neural network (ANN), which attempts to model the processing of the human brain. Neural network approaches first appeared in 1943 [180] and have become a popular tool in recent decades due to the technological breakthroughs in computing power. A neural network is, like the brain, composed of many simple processing units, called neurons, connected in a network structure. It can be seen to some extent as a 'black box' which will learn the relationship between training input signal and the desired network response or output (e.g. classification). The three properties which characterise an ANN are [181]

1. Network topology, or interconnection of neurons,
2. Characteristics of individual neurons, and
3. Strategy for pattern learning or training.

Every neuron will contain a number of inputs which are transformed into a single output. The topology of a network describes which neuron outputs are connected to which neuron inputs. The connections can be feed-forward where all signals flow in the same direction (from the input layer to the output layer), or feed-back where signals can be fed back into previous layers. The connections are usually made via connection weights which alter the strength of the connection. The most common learning approach is to vary the connection weights to produce the desired network response. As the internal structure to some extent is learnt by the system, an ANN approach to classification lends itself to problems where the a priori knowledge of the feature data is limited (e.g. no physical model is available for the process) but the collection of suitable training data is

possible. The ANN approach is also very suited to cases where the underlying features are nonlinear as in general each neuron is a non-linear device [182].

6.2.1 Multi-layer perceptron networks

The most common ANN is the multi-layer perceptron (MLP) network, a feed forward network in which the outputs of one layer are fed via connection weights to the inputs of each neuron in the next layer. The network contains an input layer which simply stores the input values, one or more hidden layers which sequentially transform the inputs and results of previous hidden layers to a more separable representation and an output layer at which the network response is seen. Each layer is made up of a number neurons and the output of a neuron is calculated using an activation function, f , of the input or internal activation, net_j , where f is non decreasing and differentiable and is usually the same for each neuron in a single layer. So for a given neuron, j , the output due to the input $x(k)$ is

$$o_j(k) = f(net_j(k)) \quad (6.31)$$

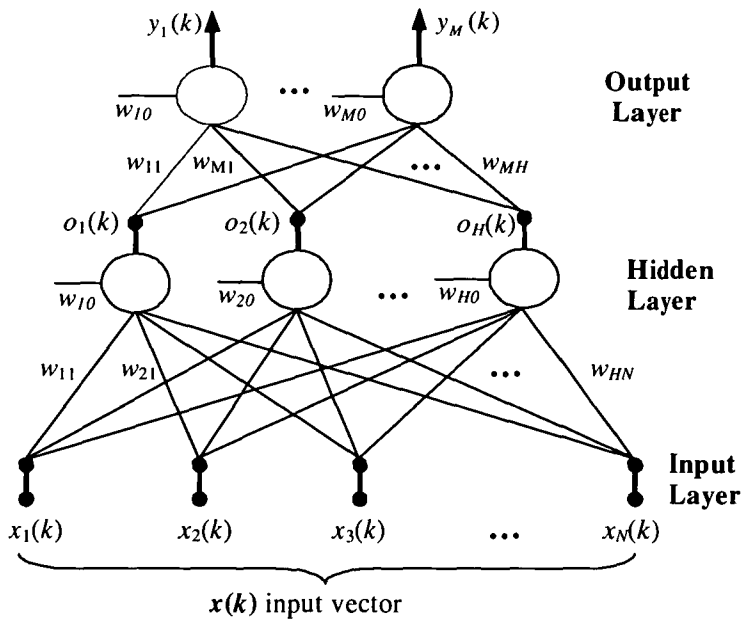


Figure 6.1: Multi-layer perceptron network containing single hidden layer

The internal activation is a linear combination of the output of the neurons in the previous layer and the corresponding weight w_{ji} connecting neuron i in the previous layer to neuron j . A neuron is normally also connected to a bias term which can be represented using a bias weight w_{j0} connected to an extra input, with index $i=0$, which has a value ± 1 . If there are I neurons in the layer previous to neuron j the internal activation of j is

$$net_j(k) = \sum_{i=0}^I w_{ji} o_i(k) \quad (6.32)$$

A common activation function used is the logistic sigmoid because the outputs of these units can be interpreted as probabilities of features in the input space, i.e.

$$o_j(k) = \frac{1}{1 + \exp[-net_j(k)]} \quad (6.33)$$

For classification purposes the network outputs can be viewed as class conditional probabilities, i.e. $y_m = p(c_m | \mathbf{x}(k))$. As a result the requirements of the output neurons are that the activation functions range between zero and one and the sum of all outputs for a given input example should sum to one. In the multiple class case in which the output layer contains a neuron for each class, a generalisation of the logistic sigmoid called the ‘softmax’ function is used for the output activation function [183]

$$y_m(k) = p(c_m | \mathbf{x}_k) = \frac{\exp(net_m(k))}{\sum_{j=1}^M \exp(net_j(k))} \quad (6.34)$$

6.2.2 Training the network

The most common training method for a MLP is known as back-propagation [184]. The training process uses a set of training input/output target pairs $\{\mathbf{x}(k), \mathbf{t}(k)\}$ for $k = 1..K$ to optimise the weights connecting each neuron in the network. This is achieved by minimising some function E of the error between the actual network output $\mathbf{y}(k)$ and the target output $\mathbf{t}(k)$ for the input $\mathbf{x}(k)$. The minimum occurs when the derivative of the

error function with respect to each of the weights w_{ji} connecting neuron i to neuron j is 0, i.e.

$$\frac{\partial E(k)}{\partial w_{ji}} = \frac{\partial E(k)}{\partial net_j(k)} \frac{\partial net_j(k)}{\partial w_{ji}} = \delta_j o_i(k) = 0 \quad (6.35)$$

where the delta term δ_j is

$$\delta_j = \frac{\partial E(k)}{\partial net_j(k)} = \frac{\partial E(k)}{\partial o_j(k)} \frac{\partial o_j(k)}{\partial net_j(k)} = \frac{\partial E(k)}{\partial o_j(k)} f'(net_j(k)) \quad (6.36)$$

For neurons in the output layer, the delta terms can be calculated directly as the error function E is a function of the neuron outputs o_j . For neurons in the hidden layers though, this is not the case so the delta terms have to be estimated using the delta terms in the next layer, i.e. for a hidden neuron j connected to the M neurons in the output layer

$$\frac{\partial E(k)}{\partial o_j(k)} = \sum_{m=1}^M \frac{\partial E(k)}{\partial net_m(k)} \frac{\partial net_m(k)}{\partial o_j(k)} = \sum_{m=1}^M \delta_m w_{mj} \quad (6.37)$$

This results in a two phase training process, where initially a training sample is fed forward through the network to calculate the outputs of all the neurons layer by layer. Then the output layer neuron outputs are used to calculate the output layer delta terms which can then be propagated back through the layers of the network to calculate the weight changes at each level. The weight update can be done online (i.e. making a change after each training sample) or iteratively using batch learning (i.e. in a single iteration every training sample will be presented to the network and a single change is made for each iteration). The weight update, known as the delta rule [184], is proportional to the error gradient but in the opposite direction. The amount of change is determined by learning rate η , i.e. for a single iteration in batch learning

$$\Delta w_{ji} = -\eta \sum_{k=1}^K \frac{\partial E(k)}{\partial w_{ji}} \quad (6.38)$$

The goal of a classification network is to maximise the likelihood of the conditional probability $p(\mathbf{t}|\mathbf{x})$, i.e. the probability of training target set given the training input set. In a network containing an output y_m for each class, then for a training sample k , belonging to class c_m , the probability of observing $\mathbf{t}(k)$ for the given input $\mathbf{x}(k)$ is simply $p(c_m|\mathbf{x}(k)) = y_m$, i.e. the network output of the neuron representing class c_m . As such, the conditional probability for that sample $p(\mathbf{t}(k)|\mathbf{x}(k))$ can be written as [191]

$$p(\mathbf{t}(k) | \mathbf{x}(k)) = \prod_{m=1}^M (y_m(k))^{t_m(k)} \quad (6.39)$$

As is common in maximum likelihood estimation, instead of maximising the above, the negative log of the likelihood, $-\ln(p(\mathbf{t}(k)|\mathbf{x}(k)))$, is minimised. So for classification purposes, with a single output for each class using ‘1 of c ’ coding for the target vector, the most suitable error measure, known as the cross entropy, is [191]

$$E = -\sum_{k=1}^K \sum_{m=1}^M t_m(k) \ln y_m(k) \quad (6.40)$$

Commonly the sum of squares is used as the error function which is a viable alternative when the training data set is large [182].

6.2.3 Adaptive learning rates

The amount of change in weight at any iteration of the back propagation algorithm is determined primarily by the learning rate parameter. If the learning rate is too small, the training will be very slow as each layer is highly interconnected resulting in a large number of weights to be updated. A small learning rate can also cause the training to become stuck in a local minimum of the error function. Conversely, a large error rate can result in the system to oscillate and prevent convergence. As a result the rate of learning is often adapted during the training process, guided by four heuristics first proposed in [185] which relate the network parameters to the properties of the error surface. Firstly, the error surface is very different in each parameter, so it may be useful for each weight to have its own individual learning rate. Also different portions of the error surface will behave very differently so each learning rate should vary from

iteration to iteration as the error surface is traversed. Finally in smooth sections of the error surface (i.e. when the gradient is in the same direction for consecutive iterations) the learning rate can be increased, and when the direction gradient of the alternates in consecutive iterations the learning rate should be decreased.

A popular method for speeding up the training process is by adding a momentum term to the weight update equation [184]. The momentum term adds a proportion of the weight update in the previous iteration to the weight update in the current iteration according to the following equation, commonly referred to as the generalised delta rule, in which α is a positive momentum constant between 0 and 1, commonly chosen to be approximately 0.9. The momentum term guides each update in the same direction as the previous update, helping the system to prevent oscillations and escape local minima.

$$\Delta w_{ji}^{(n+1)} = -\eta \sum_{k=1}^K \frac{\partial E(k)}{\partial w_{ji}^{(n)}} + \alpha \Delta w_{ji}^{(n)} \quad (6.41)$$

Even with the inclusion of momentum, back propagation training is still a slow process. More efficient techniques of adaptive learning, such as the *delta-bar-delta* rule [185], *quickprop* [186], and *RPROP* (resilient back-propagation) [187] all use an individual learning rate for each weight parameter which is also adapted at each step in the training process. The RPROP algorithm is an efficient batch learning technique which combines the ‘*Manhattan*’ update step which uses only the sign of the error gradient to determine the direction of the change with an individual learning rate η_{ji} for each weight w_{ji} , i.e. for the n^{th} iteration

$$\Delta w_{ji}^{(n)} = -\eta_{ji}^{(n)} \text{sign} \left[\sum_{k=1}^K \frac{\partial E(k)}{\partial w_{ji}^{(n)}} \right] \quad (6.42)$$

The learning rate is adapted after each iteration so the error gradient has the same sign as the previous epoch the learning rate is increased, by the factor γ^+ , and if the error gradient changes sign the learning rate is decreased by the factor γ^- , where $0 < \gamma^- < 1 < \gamma^+$, i.e.

$$\eta_{ji}^{(n)} = \begin{cases} \gamma^+ \eta_{ji}^{(n-1)} & \text{if } \partial E_{w_{ji}}^{(n)} \partial E_{w_{ji}}^{(n-1)} > 0 \\ \gamma^- \eta_{ji}^{(n-1)} & \text{if } \partial E_{w_{ji}}^{(n)} \partial E_{w_{ji}}^{(n-1)} < 0 \end{cases} \quad (6.43)$$

Typically $\gamma^- \approx 0.5$ and $\gamma^+ \approx 1.1$ so the learning rates are increased slowly but decreased rapidly if oscillations occur. It is also common to set upper and lower limits on the learning rates so they do not become too large, too small or negative. The RPROP can only be used in batch learning as the learning rate updates become noisy if the error gradient is calculated over a single training sample.

6.2.4 Pre-conditioning of network

The efficiency and effectiveness of the back-propagation training process can be further improved by preconditioning the network with appropriate initialisation of network parameters, inputs, and outputs. As the values of input variables may differ by orders of magnitude, it is common to normalise the input variables using the following equation, where \bar{x}_n and σ_n are estimates of the mean and standard deviation respectively of attribute n to ensure they have zero mean and a unit standard deviation.

$$\hat{x}_n = \frac{x_n(k) - \bar{x}_n}{\sigma_n} \quad (6.44)$$

This transformation ensures that all inputs have similar values (of order unity) and if appropriate activation functions are chosen then the network weights will also be order unity. It is also important that the target values are set within the range of the output activation function chosen. For the logistic sigmoid activation function if the saturation values of the function (i.e. 0 and 1) are used as target values the trained weights will tend towards infinity slowly the training process. Instead target values a small amount, ε , inside the saturation values should be used (i.e. $0 + \varepsilon$, and $1 - \varepsilon$) [182]. The network connection weights are commonly initialised to small random values. The initial weights must be small enough that the neuron activations are not driven to saturation point, but not so small that very small error gradients are produced resulting in training to be very slow. To achieve this, assuming all inputs have been appropriately normalised, for neuron j each weight w_{ji} at the input to j should be generated from a

Gaussian distribution with zero mean and standard deviation proportional to $1/\sqrt{I_j}$ where I_j is the number of input connections to neuron j [191].

6.3 Radial basis function (RBF) networks

Another ANN type classification technique, similar in structure to an MLP Network, is the radial basis function (RBF) network [188,189]. The structure of an RBF network, shown in Figure 6.2, contains a single hidden layer of H neurons, an N -dimensional input layer for the input feature vector, and an M -dimensional output layer containing one neuron for each class in a classification problem. The role of the hidden units is to perform a non-linear transformation of the input space into the space of the hidden unit activations which gives the RBF network a much greater representational power than the linear perceptron. The hidden layer neuron activation functions are non-monotonic radial basis functions as opposed to the monotonic sigmoid activation functions in an MLP network. A radial basis function produces a maximum activation at the centre and the response decreases with distance from the centre, which causes each hidden neuron to be 'locally tuned' to a cluster of input samples. A specific neuron will have a receptive field for which input samples with a feature vector close to the centre of the basis function will provide the greatest response for that neuron. In a classification problem a single node will often provide a larger activation for a single class, and as such the hidden layer is designed to transform the input space into a space that enhances the separability of the different classes. The output layer is a simple linear summation as opposed to the commonly used logistic sigmoid activation in the MLP network. Like an MLP network, in a classification problem the outputs can be thought of as probabilities so an input sample k is defined as class m if the output neuron y_m has the maximum activation of all the output neurons.

6.3.1 Conventional RBF networks

In a conventional RBF network, the inputs are fed directly into the hidden layer, i.e. the input weights w_{hn} are all set to 1. The most commonly used radial basis function is the Gaussian function, so the non-linear activation of the hidden neuron h due to the k^{th} input feature vector $x(k)$ with respect to the internal node activation, net_h , is

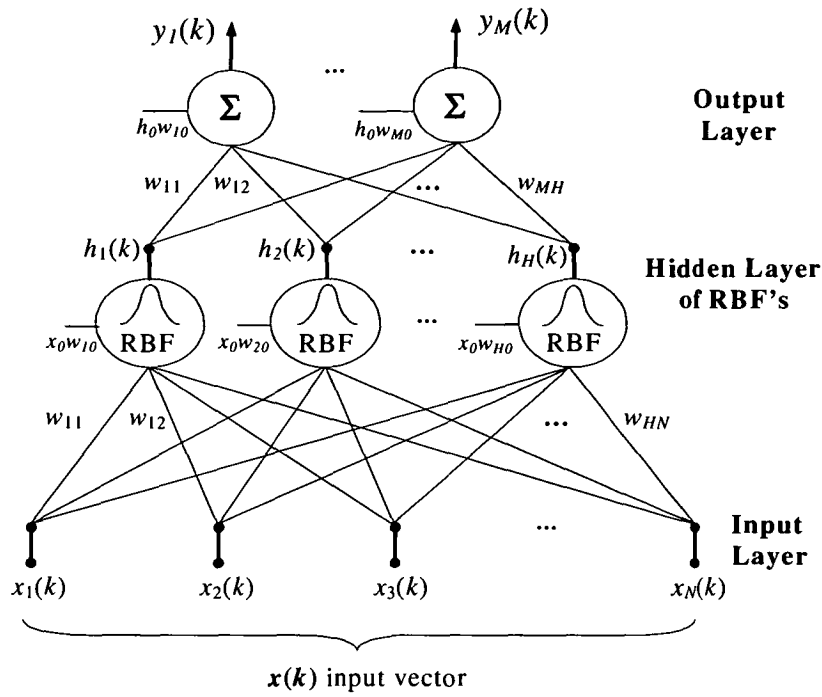


Figure 6.2: Radial basis function (RBF) network structure

$$a_h(\mathbf{x}(k)) = \exp[-net_h(\mathbf{x}(k))] \quad (6.45)$$

The internal node activation, net_h , is a measure of the distance between the N -dimensional input sample $\mathbf{x}(k)$ and the N -dimensional position vector of the node centre $\boldsymbol{\mu}_h$, giving a maximum node activation for samples closest to the node centre with the response of the node decreasing with radial distance from the node centre. The distance measure is usually weighted using some variance measure to control the rate of decrease (i.e. the width or spread of the Gaussian function) in each dimension of the basis function for each node. If the same variance is used in each dimension, then net_h is simply a Euclidean distance. In most practical applications though, the distribution of samples in each dimension of the N -dimensional input space is different, so a further generalisation known as the *Mahalanobis* distance [190], in Equation (6.46), is commonly used. This allows for a different spread in each dimension by using the covariance matrix Σ_h^{-1} of the samples clustered into the h^{th} node as a scaling factor.

$$net_h(\mathbf{x}(k)) = (\mathbf{x}(k) - \boldsymbol{\mu}_h)^T \Sigma_h^{-1} (\mathbf{x}(k) - \boldsymbol{\mu}_h) \quad (6.46)$$

This function produces a scalar internal activation for each node as each of the components are N -dimensional, the same as the input feature vector, with each dimension having a different centre and spread. Generally, if the input features are uncorrelated, i.e. they are independent but with different variances, then the covariance matrix can be simplified to that in Equation (6.47). This has the effect that nodes and dimensions with a large variance will have little significance on the activation of the node because the spread of that node or dimension will be large resulting in little change to the internal activation as the radial distance from the centre increases.

$$\Sigma_h^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2^2} & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & 0 & \frac{1}{\sigma_N^2} \end{bmatrix} \quad (6.47)$$

The output layer consists of M linear summation nodes with linear activation in contrast to the logistic sigmoid activation of output nodes in a MLP network

$$y_m(\mathbf{x}_k) = net_m(\mathbf{x}_k) \quad (6.48)$$

where the internal node activation of node m is, like an MLP, the weighted sum of all the hidden node activations in Equation (6.49). Each hidden node h has a different weight w_{mh} connecting itself to each output neuron m . There are $H+1$ connections into each output neuron, where w_{m0} represent bias weights which are connected to constant unit inputs, i.e. $a_0 = 1$. These bias weights are needed to compensate for the difference between the mean (over the training set) of the output vector for the network and the corresponding mean of the target data [191]. In a conventional RBF network, as there are no weights connecting the inputs to the hidden layer, there are no bias terms in the hidden layer.

$$net_m(\mathbf{x}(k)) = \sum_{h=0}^H w_{mh} a_h(\mathbf{x}(k)) \quad (6.49)$$

The training process in a conventional RBF network involves the optimization of four network parameters with respect to the classification accuracy. These parameters are the number of neurons in the hidden layer, the position vectors of the centre of each neuron, the spreads of each neuron, and the output weights. A number of different techniques, including statistical, heuristic, supervised and unsupervised processes have been proposed in the literature for each parameter. The most common implementations of a conventional RBF network use a two-stage training process which first determines the hidden layer activations and then uses a supervised learning process to calculate the output layer weights. The main advantage of the two stage process is that the training of each layer is decoupled meaning the training time is much reduced when compared to the computationally expensive non-linear back propagation training of a MLP network. Other authors [191,192,193,197] present a third gradient descent based training stage to optimise all parameters simultaneously.

6.3.2 RBF Networks for classification

If the RBF network above, with H neurons in the hidden layer and M output neurons, is considered in a probabilistic form, an insight into the use of RBF networks for classification can be achieved [191]. In a M -class classification problem, each neuron, m , in the output layer is considered to be the posterior probability $P(c_m|\mathbf{x})$ for each class, c_m , given the input vector \mathbf{x} , which from Bayes theorem are

$$P(c_m | \mathbf{x}) = \frac{p(\mathbf{x} | c_m)P(c_m)}{p(\mathbf{x})} \quad (6.50)$$

The RBF network can be compared to a mixture model in which the class-conditional distributions $p(\mathbf{x}|c_m)$ are calculated using a mixture of the H basis functions produced by the hidden neurons, and the unconditional density $p(\mathbf{x})$ is found by summing across the all the basis functions i.e.

$$p(\mathbf{x} | c_m) = \sum_{j=1}^H p(\mathbf{x} | j)P(j | c_m) \quad (6.51)$$

$$p(\mathbf{x}) = \sum_{j=1}^H p(\mathbf{x} | j) P(j) \quad (6.52)$$

Combining Equations (6.51) and (6.52), with a factor of $P(j)/P(j) = 1$ inserted gives the following probabilistic model of an RBF network

$$P(c_m | \mathbf{x}) = \frac{\sum_{j=1}^H p(\mathbf{x} | j) P(j | c_m) P(c_m)}{\sum_{j'=1}^H p(\mathbf{x} | j') P(j')} \frac{P(j)}{P(j)} = \sum_{j=1}^H w_{mj} a_j(\mathbf{x}) \quad (6.53)$$

where the activation, $a_j(\mathbf{x})$ of hidden neuron j and output layer weight w_{mj} connecting hidden neuron j and output neuron m are

$$a_j(\mathbf{x}) = \frac{p(\mathbf{x} | j) P(j)}{\sum_{j'=1}^H p(\mathbf{x} | j') P(j')} = P(j | \mathbf{x}) \quad (6.54)$$

$$w_{mj} = \frac{P(j | c_m) P(c_m)}{P(j)} = P(c_m | j) \quad (6.55)$$

So the each hidden neuron can be thought of as a detector of some feature in the input space, where the activation of each neuron is the posterior probability of that feature occurring, and the output weights are the posterior probability of class membership given the presence of a feature.

6.3.3 Training of output connection weights

Assuming the hidden layer parameters have been determined, the output of the m^{th} output neuron of an RBF network with H hidden nodes due to input $\mathbf{x}(k)$ is linear combination of the hidden neuron activations so the output matrix, $\mathbf{y}(k)$, of the output layer can be conveniently written in matrix form using the $H \times M$ weight matrix, \mathbf{w} , and the $K \times H$ design matrix of the RBF, \mathbf{H} , in which H_{kh} is the activation of the h^{th} hidden node due to the k^{th} input training example i.e.

$$\mathbf{y}(\mathbf{x}(k)) = \mathbf{H}(\mathbf{x}(k)) \mathbf{w} \quad (6.56)$$

The weights connecting the hidden and output layers in the network can be determined by minimising a sum of squares error function using a least squares technique as in Section 5.2.2.1. As the sum of squares error is a sum over all patterns in the training set it the error in matrix form, where \mathbf{t} is the $K \times M$ training output matrix, is

$$E = \frac{1}{2} \sum_{k=1}^K (\mathbf{t}(k) - \mathbf{H}(k)\mathbf{w})^2 \quad (6.57)$$

This is solved for \mathbf{w} by computing the derivative of E with respect to \mathbf{w} and setting this to zero which has the closed form pseudo-inverse solution

$$\mathbf{w} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{t} \quad (6.58)$$

This solution is only valid for linear output neurons using a sum squared error and for the case where the matrix $\mathbf{H}^T \mathbf{H}$ is non-singular. It is not uncommon in practice in the presence of noisy data sets for this matrix to be singular or nearly singular which will result in numerical problems due to very large weight values. This problem is overcome by using the singular value decomposition technique to compute the pseudo-inverse. If non-linear output units or a different error function is used other techniques such as the more general gradient descent approach described in Section 6.3.4.3 can be used to compute the weight values.

6.3.4 Hidden layer basis function optimisation

In an RBF network, the crucial concern is the selection of the hidden layer parameters: the number of neurons, H , the neuron centres μ_h , and their widths, σ_h , so that the model can fit closely to the training space. One simple approach is to have as many basis functions as there are training samples and set the centres of the basis function to the sample positions. While this will provide a solution that will model the training data exactly, it will result in a very large network and in the presence of noise in the training data and the model will not generalize well when presented with an unseen test set. This is because the model is too complex and has learnt to match the unwanted noise as well as the desired underlying solution in the training data.

6.3.4.1 K-means clustering

One common solution to selecting a subset of centres is to use a clustering technique which can more accurately reflect the distributions of the data in the training set. Such a technique is *K*-means clustering, used in [189], in which a random set of *K* centres, μ_j for $j=1..K$, are chosen from the data set and then each sample in the training set is assigned to the cluster S_j belonging to the nearest cluster centre μ_j . The aim is then to find a group of centre points to minimize the sum squared error of the distance between the training samples and the cluster centres, i.e.

$$SSE_{k-mean} = \sum_{j=1}^K \sum_{p \in S_j} \|\mathbf{x}(p) - \mu_j\|^2 \quad (6.59)$$

The minimisation is achieved by iteratively updating the cluster centres to be the mean of the data points in the each cluster S_j , then recalculating the clusters using the new cluster centres and repeating until the SSE_{k-mean} is at an acceptable level, or has settled to within a set threshold or a set number of iterations has been reached. Once the cluster centres have been determined, they can then be used to determine suitable values for the spreads of each neuron. The *N*-dimensional spreads or scaling factors for each node need to be chosen so that the node activations cover all the samples in the training set, allowing a smooth fit to the desired network outputs. This means that the scaling factors need to be large enough so that neighbouring nodes overlap, but not so large that the activation of a node loses the property of being locally tuned. A heuristic solution to this problem proposed by [194] used the distance *P*-nearest node centres to calculate the spread ($P=2$ was used in [194]), i.e. for node *h* in a single dimension *n* the spread σ_{hn} is

$$\sigma_{hn} = \sqrt{\frac{1}{P} \sum_{k=1}^P \min_k \|\mu_{hn} - \mu_{kn}\|^2} \quad (6.60)$$

Another technique to calculate the spreads is to use the covariance matrix of each node *j* by calculating the standard deviation σ_{hn} of the N_h samples assigned to the h^{th} cluster S_h , as in Equation (6.61), and these standard deviation's are usually multiplied by a constant

to ensure sufficient overlap of neighbouring nodes (i.e. to ensure the activation at the border between two nodes is greater than say 0.5).

$$\sigma_{hn} = \sqrt{\frac{1}{N_h} \sum_{k \in S_h} \|x_n(k) - \mu_{kn}\|^2} \quad (6.61)$$

The speed of convergence of the clustering can be improved by using prior knowledge of the input data to select the initial centres rather than a random initialization. For instance in the classification domain it might be useful to define a single cluster for each class, using the means of the training data in each class as the initial centres. This leads to one of the drawbacks of the K -means clustering procedure in that the number of centres K has to be selected in advance, providing little control over the classification model complexity. As a result the correct number of centres to provide accurate classification on an unseen test set is difficult to determine, especially without significant prior knowledge of the input data, i.e. in the above example whether each class can be described by a single cluster or a number of smaller ones.

6.3.4.2 Tree-based initialisation

Another clustering type approach, first proposed by [195] and further explored by [196,197] is the use of classification or regression trees [177,178] to initialize the hidden layer of an RBF network. The advantage of this approach is that it is a supervised clustering technique which separates the instance space into hyper-rectangular regions (or nodes in the tree), in which ideally each region, or cluster, contains only inputs of the same class. As a result the position and size of any node in a tree can be used to determine the centre and spread of an RBF neuron. Also the structure of the tree can be used to calculate the number of hidden neurons needed, and as each split in the tree is based on a single component of the feature vector, irrelevant features will not appear in the tree and so can be ignored in the RBF network calculations.

Each node, j , in the tree is associated with a hyper-rectangle with the N -dimensional centre position vector c_j and N -dimensional size vector s_j in the input space. As such,

each node in the tree can be converted to a hidden neuron in an RBF network with the activation of a hidden neuron corresponding to node j in the tree being

$$a_j(\mathbf{x}) = \exp\left\{-\sum_{n=1}^N \frac{(x_n - \mu_{jn})^2}{2\sigma_{jn}^2}\right\} = \prod_{n=1}^N \exp\left\{-\frac{\alpha^2(x_n - c_{jn})^2}{2s_{jn}^2}\right\} \quad (6.62)$$

where the neuron centres μ_{jn} are calculated from the tree node centre c_{jn} , and the neuron spreads σ_{jn} are proportional to the tree node sizes s_{jn} scaled by a constant, i.e.

$$\sigma_{jn} = \frac{s_{jn}}{\alpha} \quad (6.63)$$

The scaling constant α will set the node activation at the point where bordering nodes overlap. It is commonly the same value in all dimensions n and all nodes j to ensure the ratio between the node size and neuron spread is constant and the sensitivity of the overall network performance compared to the exact choice of α is low [195]. In [196] it is claimed that a greater sensitivity to α exists so a number of trial values for the scale factor are used and the value that produces the best performing network is selected. From Equation (6.62), the activation of a node is the product of the response in each dimension separately. For a given tree node, the branch leading to that node will consist of a number of splits, each in a single dimension, but not all dimensions are necessarily used in the splitting process, resulting in a number of irrelevant attributes, which can be ignored by setting the activation to 1 in the corresponding dimension.

The exact choice of centres and spreads requires that the hidden node activations at the borders in each dimension of the hyper rectangles are the same for bordering nodes. The method proposed by [195] was to use the centre of a tree node as the centre of a hidden neuron, except if the border of the tree node is the minimum or maximum value in a single feature dimension. In this case, the centre will be placed at the border of the node in that dimension. The spreads are set to be the size of tree node in any dimension not touching the boundary, and double the node size for boundary cases. In [196] a comparison of the above method and always placing the neuron centre in the middle of the tree node was performed and results indicated that the exact positioning of centres

did not clearly affect the performance of classification. In both cases to achieve an activation of 0.5 at the border of two nodes as above will require a value of 2.35 for α . Another technique proposed in [197] was take c_j to be the centre of gravity of the data in the tree node in each dimension and the spread s_j is simply the Euclidean distance between c_j and its nearest neighbour. Once again α is used to control the activation at the border of two nodes.

If all nodes in the regression tree are used as hidden neurons in the RBF network, the result will be a very large network prone to over-fitting, poor generalisation and slow training so a subset of RBF hidden neurons from the regression tree needs to be selected. The solution in [195] is to use only the leaves of a suitably pruned tree as hidden neurons, with experiments giving the best classification accuracy for moderate if any pruning as it seems the RBF network can compensate for any errors introduced due to over-fitting in the tree. An unpruned tree leads to a large RBF network though, so to reduce network size, instead of using a pruning algorithm, it is proposed that it would be sufficient to use some vaguely defined 'upper part of tree' i.e. the top third. In [196] the RBF network is grown, guided by the structure of the tree, using a technique combining forward selection and backward elimination to add/remove nodes to/from the RBF network so as to maximise some model selection criteria (MSC). Intuitively, the tree nodes corresponding to largest hyper-rectangles (i.e. with largest s_j) should be added to the RBF network first to model coarse details, and smaller nodes later to model fine details. As a result the best order to process the tree nodes is to start at the root node (the largest node) and work down through its children and finally to the terminal nodes. The process starts with an RBF network containing only the root node of the tree. An 'active list' of nodes (also initialised with only the root node of the tree) is iteratively processed and updated as nodes are added/removed to/from the RBF network. At each iteration, for every non-terminal node in the active list a new RBF network is created by considering all possibilities of deleting the current node, and adding neither, one or both of the children nodes to the RBF network. Of all possible changes to the model, the one which decreases the MSC by the most is selected, the node the change effects is removed from the active list, and that nodes children are added to the active list. This iterative process terminates when either the active list contains only terminal nodes or if

at any iteration none of the possible modifications will decrease the MSC. The MSC used to select the best subset of hidden neurons commonly reported in literature to give the best results is the Bayesian information criterion (BIC) [196]

$$\sigma_{BIC}^2 = \frac{K + (\ln(K) - 1)H}{K(K - H)} E$$

where E is the error of the RBF network being considered. It is proposed in [196] that the above process is performed on a number of trees grown with different K_{min} values, and for each tree a number of different scale values, α , is also considered.

6.3.4.3 Gradient descent optimisation

The above techniques involve a two stage training process, with the first stage determining the hidden layer parameters and the second stage the output connection weights. This can produce classification errors as the positions of cluster centres and their widths may be sub-optimum. If the training space is highly overlapped between classes the K -means clustering technique cannot give the optimum cluster centre positions since it does not require the knowledge of the desired outputs, and in the regression tree technique the centres are simply placed at the middles of the hyper-rectangles. In [191,192,193,197] a three stage training process is proposed to overcome this. After the first two stages described above, the third stage uses gradient descent on each of the network parameters, so the neuron centres and spreads can be iteratively updated in parallel with the output layer weights until an optimal solution for the RBF network is reached. The gradient descent procedure is a generalisation of the LS technique solved by the pseudo inverse above, as it can be used to minimise any error function that is differentiable with respect to the network parameter being optimised. In the case of an RBF network the sum of squares error function is chosen to be minimised as the output neurons have linear activation functions. The total network error E is the sum over K training samples of the network error $E(k)$ for each input pattern $\mathbf{x}(k)$, where $E(k)$ is the sum over each output neuron m of squared difference between the network response $y_m(k)$ and the desired response $t_m(k)$, i.e.

$$\begin{aligned}
 E &= \frac{1}{K} \sum_{k=1}^K E(k) \\
 E(k) &= \frac{1}{2} \sum_{m=1}^M [t_m(k) - y_m(k)]^2
 \end{aligned} \tag{6.64}$$

The minimisation is achieved using the delta rule, which updates a network parameter a short distance in the direction of the negative gradient of the error function with respect to the parameter being optimized. The amount of change in a parameter at each step is determined by a learning rate for each parameter, η_w for the weights, η_μ for the centres, and η_σ for the spreads i.e. the update for each parameter due to input sample $\mathbf{x}(k)$ is

$$\begin{aligned}
 \Delta w_{mh}(k) &= -\eta_w \frac{\partial E(k)}{\partial w_{mh}} \\
 \Delta \mu_{hn}(k) &= -\eta_\mu \frac{\partial E(k)}{\partial \mu_{hn}} \\
 \Delta \sigma_{hn}(k) &= -\eta_\sigma \frac{\partial E(k)}{\partial \sigma_{hn}}
 \end{aligned} \tag{6.65}$$

To compute the update to the weight parameter Δw_{mh} connecting the h^{th} hidden node to the m^{th} output node the partial derivative of the error with respect to the weight can be calculated using the chain rule, i.e.

$$\begin{aligned}
 \frac{\partial E(k)}{\partial w_{mh}} &= \frac{\partial E(k)}{\partial y_m(k)} \frac{\partial y_m(k)}{\partial net_m(k)} \frac{\partial net_m(k)}{\partial w_{mh}} \\
 &= -[t_m(k) - y_m(k)] a_h(\mathbf{x}(k))
 \end{aligned} \tag{6.66}$$

Similarly, the network parameter update equations for the centres μ_{hn} are,

$$\begin{aligned}
 \frac{\partial E(k)}{\partial \mu_{hn}} &= \sum_{m=1}^M \frac{\partial E(k)}{\partial y_m(k)} \frac{\partial y_m(k)}{\partial net_m(k)} \frac{\partial net_m(k)}{\partial a_h(k)} \frac{\partial a_h(k)}{\partial net_h(k)} \frac{\partial net_h(k)}{\partial \mu_{hn}} \\
 &= -a_h(\mathbf{x}(k)) \times \frac{(x_n(k) - \mu_{hn})}{\sigma_{hn}^2} \sum_{m=1}^M [t_m(k) - y_m(k)] \times w_{mh}
 \end{aligned} \tag{6.67}$$

and for the spreads σ_{hn} the update rule is dependent on RBF structure as it is possible the same spread for all neurons, a single spread for each neuron or in this case a different spread for each dimension in each neuron.

$$\begin{aligned} \frac{\partial E(k)}{\partial \sigma_{hn}} &= \sum_{m=1}^M \frac{\partial E(k)}{\partial y_m(k)} \frac{\partial y_m(k)}{\partial net_m(k)} \frac{\partial net_m(k)}{\partial a_h(k)} \frac{\partial a_h(k)}{\partial net_h(k)} \frac{\partial net_h(k)}{\partial \sigma_{hn}} \\ &= -a_h(\mathbf{x}(k)) \times \frac{(x_n(k) - \mu_{hn})^2}{\sigma_{hn}^3} \sum_{m=1}^M [t_m(k) - y_m(k)] \times w_{mh} \end{aligned} \quad (6.68)$$

The optimization of each parameter is an iterative process where for each iteration, i , the total update for each weight connecting the hidden to the output layer is the sum of individual updates for each input sample $\mathbf{x}(k)$, i.e.

$$w_{mh}^{(i+1)} = w_{mh}^{(i)} + \sum_{k=1}^K \Delta w_{mh}^{(i)}(k) = w_{mh}^{(i)} - \eta_w \sum_{k=1}^K \frac{\partial E^{(i)}(k)}{\partial w_{mh}} \quad (6.69)$$

Any of the adaptive learning and momentum techniques in Section 6.2.3 can be applied to this update formula in order to improve the efficiency and robustness of the training process. With these improvements and initialisation of network parameters using the first two stages of training provide an accurate network which is trained more efficiently than a back-propagation trained MLP network.

6.3.5 Generalised RBF network

In an RBF network, the hidden layer is the most important layer, providing the majority of the discriminant power of the classification process. The output layer of the RBF network consists simply of linear summation units with linear activation which provides only a minimal contribution to the classification. In order for the training space to be optimally separable, a generalised RBF network is proposed in [198] which includes input feature weights in contrast to the direct connection of the input to the hidden layer of a conventional RBF. The feature weights give more influence to the components in the feature vector that are more important or more discriminating than other features. In a typical RBF network, the node's widths σ_{jn} also play the role of weights for the input features, but are not sufficient to enhance the discriminant effect of relevant or dominant

features or to eliminate the effect of irrelevant features for each particular class. The activation of hidden node j with the added input feature weights is now

$$a_j(\mathbf{x}(k)) = \exp \left\{ - \sum_{n=1}^N \frac{w_{jn} (x_n(k) - \mu_{jn})^2}{2\sigma_{jn}^2} \right\} \quad (6.70)$$

The cluster centres μ_j and their widths σ_j can be determined by one of the techniques described above, and the output connection weights are determined using the pseudo-inverse LS solution described in Section 6.3.3. The input feature weights w_{jn} need to be trained to optimise the hidden layer activations, and in order to keep the weight training of the two layers in the RBF decoupled, this is done using a gradient descent approach which only involves the input layer and hidden layer of the RBF. To achieve this, a target vector for the hidden layer activations, $\mathbf{t}(k)$, for each training input, $\mathbf{x}(k)$, is needed as the output layer target vectors are not relevant at the hidden layer. So the target vector can be in the range [0..1], a new variable $o_j(k)$ is generated which normalises the activation $a_j(k)$ of hidden node j due to input $\mathbf{x}(k)$ to the highest value of all hidden nodes $(a_j(k))_{\max}$ to produce hidden activation values in a range of [0,1], i.e.

$$o_j(k) = \frac{a_j(k)}{(a_j(k))_{\max}} \quad (6.71)$$

Each hidden neuron will respond the most for a cluster of input samples within the neighbourhood of the neuron centre, the majority of which will have the same output target class. Each neuron can now be assigned the same class label as the majority of samples in the cluster and depending on the distribution of training samples a class may be represented by more than one cluster in the training space. The goal of the feature weight training is to achieve greater discrimination between classes in the hidden layer space. To do this the response of a neuron needs to be maximised for the samples that have the same class as that assigned to the neuron, both in the cluster and within a neighbourhood of the neuron. Also the response of a neuron for samples with a different class to that of the neuron needs to be minimised. To achieve this, the target for $t_j(k)$ of the normalised activation output $o_j(k)$ is '1' for a sample $\mathbf{x}(k)$ for which neuron j has the maximum response of all neurons with same target class as $\mathbf{x}(k)$, and '0' for all other

samples. The output error of the j^{th} hidden node in response to the k^{th} training sample is the difference between the hidden neuron target $t_j(k)$ and the normalised activation $o_j(k)$ so the sum squared error of the hidden layer in response to the input pattern $\mathbf{x}(k)$ is

$$E(k) = \frac{1}{2} \sum_{j=1}^H (t_j(k) - o_j(k))^2 \quad (6.72)$$

Using a gradient descent approach, the hidden layer error is minimised by iteratively adjusting the weights with the delta rule as in Equation (6.69). The derivative $\partial E(k)/\partial w_{hn}$ can be evaluated using the chain rule (for full derivation see Appendix), i.e.

$$\begin{aligned} \frac{\partial E(k)}{\partial w_{hn}} &= \frac{\partial E(k)}{\partial o_h(k)} \frac{\partial o_h(k)}{\partial a_h(k)} \frac{\partial a_h(k)}{\partial net_h(k)} \frac{\partial net_h(k)}{\partial w_{hn}} \\ &= o_h(k)(t_h(k) - o_h(k)) \frac{|x_i(k) - \mu_{hi}|^2}{\sigma_{hi}^2} \end{aligned} \quad (6.73)$$

Once again any of the adaptive learning techniques discussed above can be used to improve the efficiency of the training process. After each iteration the training samples are assigned to the neuron with the greatest activation and from this new set of clusters each neuron centre μ_h and its width σ_h are updated, respectively, to the mean and the variance of the samples in its new cluster. The training is repeated until it converges or a set number of iterations has been reached. After determining the input feature weights the connection weights at the output layer in the network are found as in a conventional RBF network using the pseudo-inverse solution to the linear LS problem.

6.3.6 Comparison of RBF networks with MLP networks

Some interesting comparisons of the performance of RBF networks and MLP networks are reported in [194] and [191]. In a MLP network the hidden units have a linear internal activation followed by a monotonic activation function giving constant activation on surfaces with parallel $N-1$ dimensional hyper-planes in the input space. In the hidden layer activation space an MLP has a distributed response as a number of hidden units will respond for a given input to produce the desired output. In an RBF network the internal activation of the hidden units are proportional to the distance from

the centre of the receptive field of a neuron. The hidden unit response is then determined using locally tuned activation functions giving constant activation on concentric $N-1$ dimensional hyper-ellipsoids. As a result in the hidden layer space an RBF network has a local activation as generally only a few hidden units will respond to a given input. Due to the distributed response in the hidden layer of an MLP there is significant interference and cross-coupling between hidden neurons. This results in a highly non-linear training process and local minima/nearly flat regions in the error function which leads to very slow training. In general the training time of an RBF network is an order of magnitude faster than the training of a back-propagation network of comparable size. This is because the hidden layer units are locally tuned and the training of the two layers in an RBF is decoupled (i.e. the hidden layer parameters are determined independently from the output layer connection weights). Even RBF networks optimised with a third training phase will be trained more efficiently than an MLP as the final gradient descent procedure should be efficient as it is initialised with suitable values. The global learning ability and the monotonic nature of the hidden layer's activation in an MLP make these networks more "adventurous" and in the presence of a noisy data set therefore more prone to interpolation error. As an RBF network only has a local learning capability it can produce decision surfaces which are less sensitive to outliers in the training set to avoid the possibility of interpolation errors. This is because the hidden node centres can be conservatively placed to cover the input space adequately and due to the non-monotonic nature of the radial basis function. Classification error in an RBF network is usually due to decision error alone (i.e. due to stochastic overlap of the classes) while in an MLP network any error could be due to both decision and interpolation sources. Conversely the performance of an RBF network may be less robust than an MLP classifier in dealing with situations in which samples to be classified during an application fall outside the training set space boundary (e.g. due to drifts in the practical data acquisition systems).

6.4 Conclusion

The underlying concept of the video genre classification task is one of bridging the semantic gap between low level features and high level semantics. This task can be

viewed as a pattern recognition problem, where for each semantic class to be detected a number of features or patterns which distinguish each class need to be identified. In this chapter a number of approaches, in particular a variety of statistical and neural network approaches, designed to classify a sample feature vector $\mathbf{x}(k)$ into one of M classes have been discussed. The concept of statistical decision theory is to determine the decision boundaries in an N -dimensional feature space between classes based on a probabilistic perspective. The foundation of a statistical classifier is the deterministic Bayes decision rule which requires prior knowledge of all the class-conditional densities to be specified. However, in practice the information of the class distributions is normally not known and an estimate must be learned from the available training data. Depending on what data and prior knowledge of the density to be estimated is available, a number of techniques can be used. If the specific model of the density is known (e.g. a Gaussian distribution) a parametric approach can be used to estimate the unknown model parameters. Otherwise a general model must be used and a non-parametric approach is needed in which many free parameters are used to estimate the class distributions or construct the decision boundaries directly from the training data. In practice there is a trade off between the possible bias error when using a specific model and the possible variance error when using a general model. As a result it is usually best to use a technique in which the model complexity can be varied to suit the complexity of the problem at hand, sometimes called semi-parametric estimation. Two such approaches that have been applied to the problem of video genre classification are Gaussian mixture model and classification trees.

Better classification accuracy can be achieved with a more sophisticated classifier such as an MLP or RBF network. An MLP can be viewed as a nonparametric method that constructs the decision boundaries from the training data. An RBF network is required to estimate the class distribution from the training data. An RBF is a special class of neural network that contains a hidden layer which, if Gaussian activation functions are used, can be viewed as a Gaussian mixture model. An MLP is typically trained using the back-propagation algorithm starting the training procedure with a random initialisation of the MLP's parameters, whereas an RBF network can be trained in many different ways. The first step in training an RBF network is closely related to density

estimation and statistical approaches such as *K*-means clustering, classification trees or Gaussian mixture models can be used to determine appropriate hidden layer parameters. While the training of a conventional RBF network is significantly quicker than for an MLP classifier it has the weakness of having only a local learning capability and a limited learning inference from the training data. As a result in some situations the classification accuracy of an RBF network can be inferior to an MLP. It has been shown in [198] that using the hidden layer scaling parameters in an RBF network to weight the input features can be insufficient to enhance the discriminant power of relevant or dominant features or to eliminate the effect of irrelevant features for each particular class. While initialising the hidden layer using the regression tree technique can make a network less sensitive to irrelevant attributes by only connecting relevant features which appear in the tree, less relevant features, when incorporated with appropriate feature weights to different classes, do in fact increase the discriminant level between classes. One solution proposed is to optimise the hidden layer parameters using a third training phase based on gradient descent, while another is to include feature weights at the input layer to adaptively separate the class regions. In each approach the network has the ability to deal with complicated problems that have a high degree of interference in the training data, and achieves a higher classification rate over the current classifiers using a conventional RBF. While each approach takes longer to train than the conventional RBF network, it is still much faster than that of a back-propagation network of the same network size.

It is commonly reported in the literature that no one classifier will produce optimal results for all situations. The performance of each classifier discussed in this chapter will depend on what data is available and the properties of the input feature vector. The different classifiers also have a number of parameters which need tuning by experimentation to find the best setup for each classifier. As a result for a given application it is necessary to compare a number of techniques and setups to find a suitable classifier for that application. In the following chapter the experimental results of a number of classifiers designed for the video genre classification problem are presented.

Chapter 7

Video Genre Classification

Results

7.1 Experimental set-up and procedures

When testing the effectiveness of each of the high level semantic classification techniques discussed in Chapter 6 for the problem of video genre classification, there are a number of factors beside the parameters specific to a given technique which can affect the classification performance. To obtain experimental results which will generalise reasonably well to any other video database the experimental database needs to be representative of the each genre. This will limit the sensitivity of the results to the data used to train a classifier. It is also important to use experimental procedures which also reduce or give some measure of the sensitivity of a classifier to the dataset used for training and testing. The sensitivity of classification performance to the actual set of genres is also considered in this work. Initial experiments to investigate each of the classifiers were conducted with a four genre set {sport, news, scenery, drama} and a five genre set {sport, news, scenery, drama, cartoon}. Then in order to make some comparison with previous video genre classification works the genre sets {sport, news, cartoon, music, commercial} and {sport, news, cartoon, music, drama} were also considered.

The experimental database used in these experiments consists of a number of short video clips for each genre. Each video clip is encoded using MPEG-1 format using a frame size of 352x288 pixels and a frame rate of 25 frames per second. Each video clip is approximately 1 minute long and contains between 10 -20 shots. Each clip is taken from a separate source (e.g. each cartoon clip is from a different cartoon program, each music

clip from a different song, each sports clip from a different sport sub-genre). As the video clips in the database come from between 10-20 separate sources for each genre this should provide a database which suitably represents each genre so that the classifier will learn to recognise a genre in general, not just the patterns of a sub-set of samples from a genre.

For each video genre, a selection of 166 video shots was chosen to represent each class. Broad selections of clips were chosen to best represent each category. The sports genre clips come from a wide variety of sports broadcasts including Aussie Rules Football, Rugby, Field Hockey, Cycling, Surfing, Netball, Cricket, Soccer and others. The news genre clips were all of newsreaders in the studio (i.e. anchor person shots) from a number of different news programs, as it is believed this type of shot is a good semantic cue for the news genre. The scenery genre contains clips with both cityscapes and natural scenes, but with no object motion and smooth camera motion, as is typical for this genre of program. This genre was included in initial experiments as it seemed to have a well defined spatio-temporal structure. The drama genre is the broadest, and always contains human interaction, but the spatio-temporal content can vary greatly compared to the other genres. The clips are taken from a broad range of drama programs including movies, sitcoms, and other television series. The cartoon category is taken from a number of cartoon programs with a broad range of content. The music genre clips are taken from a range of music film clip programs from a number of modern rock/pop/dance songs. The commercial genre is made up of a number of commercials recorded from a number of television channels from a number of television shows from a range of times of the day to ensure a range of the type of products and services being advertised.

For each video clip in the database the shots are manually detected to eliminate the effect of errors in automatic shot detection on the genre classification process. For each shot a 10-dimensional feature vector is then calculated, containing the features discussed in Chapter 5 (i.e. 4 motion intensity metrics and 6 activity power flow metrics). For each of the classifiers discussed in Chapter 6 (i.e. a conventional RBF network initialised using *K*-means clustering, a conventional RBF network initialised using the leaves of a pruned tree, a conventional RBF network initialised using a tree with forward

selection and backward elimination, a conventional RBF network optimised with gradient descent, and a generalised RBF network), a random selection of 100 shots for each category is used to build the classifier and the remaining shots used as a test set to determine the classification accuracy. For each classifier type a number of parameters can be varied and this process was run 10 times (with different randomly selected training and test sets) for each set of parameters to determine the best parameter set. The comparison across the 10 simulation runs for each classifier type and set of parameters was done using the mean and standard deviation of the classification accuracy for each genre and across all genres for the 10 runs.

7.2 Classification results

7.2.1 Conventional RBF network initialised with *K*-means

When designing a conventional RBF network initialised with *K*-means clustering the most significant parameter controlling the complexity of the classifier is the number of cluster centres *K* (i.e. the number of hidden nodes in the network). This has to be determined either heuristically or by experiment. In Figure 7.1 the classification rate averaged across the four classes {sport, news, scenery, drama} for a number of different values of *K* is shown. The figure contains the average and standard deviation of the training and testing classification rate of 10 simulation runs using different training and test sets. It can be seen that the training classification rate continues to grow as the number of hidden units is increased. At lower values of *K* the testing rate is about 3% less than the training rate but begins to diverge significantly from the training rate at a *K* value of about 25. The average testing classification rate at this point continues to rise but at a smaller rate. For values of *K* greater than 66 the testing classification rate begins to plateau at approximately 84%, above which a minimal increase in classification accuracy is achieved for added network complexity.

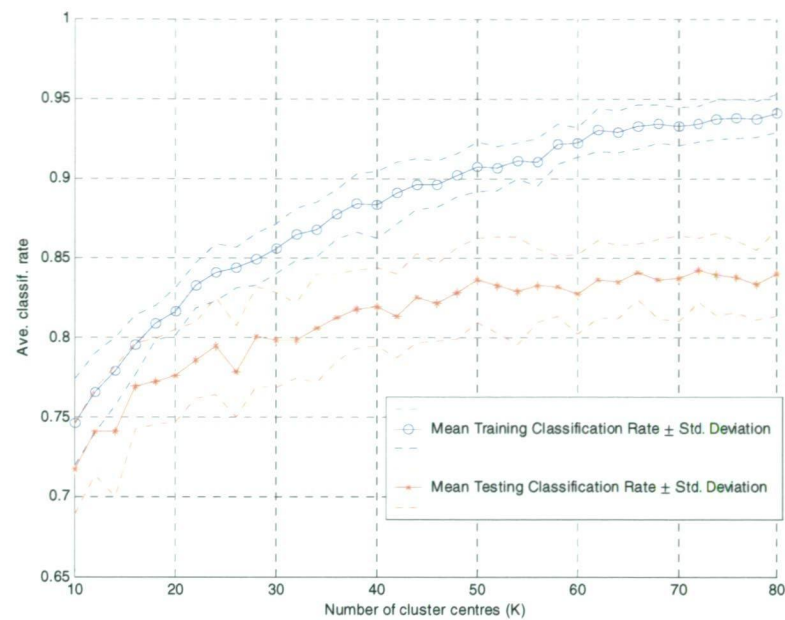


Figure 7.1: Four genre classification results for RBF network initialised with K -means clustering using varied number of cluster centres K .

As a result $K = 66$ can be chosen as a suitable number of cluster centres and a more detailed analysis of this classifier can be seen in the confusion matrix in Table 7.1. This shows the average classification accuracy of each genre across the ten simulation runs, as well as the confusion between the different genres. This shows that the news and sport genres are classified most accurately and rarely confused with each other. These results are intuitive when considering the typical spatio-temporal content of each genre (e.g. the sports genre contains highly erratic motion where as the news genre contains none). The sport genre is most commonly confused with the drama genre and vice versa, which occurs because it is possible for shots from the drama and sport genres to contain similar levels of camera and object motion. The sports genre is also confused, to a lesser degree, with the scenery genre, which can be attributed to long ‘panoramic’ shots in the sports genre that occasionally coincide with little object motion. It is also evident that the scenery and drama genres can be confused with all other genres. The standard deviation of the classification accuracy across the simulation runs for each class is large for some genres (e.g. drama), because the difference in training set for each run will cause the classifier to be tuned differently for samples on the boundary between classes. This can be seen more clearly when compared with the standard deviation of

the average classification across all genres from Figure 7.1, which is about 2.5%. This value is lower than the standard deviation of individual genres, because for different training sets the classification accuracy may increase for one genre and decrease for another genre, producing a similar average classification accuracy across all genres.

Table 7.1: Four genre confusion matrix for average classification and standard deviation for conventional RBF Network initialised with K-means clustering using K=64

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	86.1%	0.6%	4.2%	9.1%
	N	1.4%	92.1%	2.4%	4.1%
	Sc	4.7%	7.3%	77.4%	10.6%
	D	12.1%	2.1%	4.8%	80.9%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	3.8%	1.0%	3.4%	3.3%
	N	1.1%	3.6%	2.2%	3.7%
	Sc	2.6%	2.1%	3.9%	2.8%
	D	3.9%	2.2%	2.4%	5.0%

A value of $K = 66$ for four genres corresponds to an average of 16-17 hidden neurons per class in the RBF classifier. When a fifth genre, cartoons, is added 16-17 nodes per genre will give 80-85 hidden nodes. In Figure 7.2 the average classification across five genres for a number of K values is shown. The behavior is the same as the four class case although as expected the testing classification rate plateaus at a higher value of approximately $K = 78$ and the performance of this classifier is further analysed in Table 7.2.



Figure 7.2: Five genre classification results for RBF network initialised with K -means clustering using varied number of cluster centres K .

It can be seen from the confusion matrix in Table 7.2 that the behaviour of the five genre case is similar to the four genre case. The accuracies of the scenery and drama genres have not changed, and they again have the least classification accuracy. It seems that when the cartoon genre is also considered, fewer scenery shots are classified as drama or sport and fewer drama shots are classified as sport, but for both genres there is also some confusion with the cartoon genre. The accuracy of the news genre is again the best with very few confused with the cartoon class. The classification accuracy of the sport genre has fallen slightly as some of these shots are confused with the cartoon genre, although the confusion between the sport and drama genres and the sport and scenery genres have both decreased slightly. The cartoon genre has the second most accurate classification rate with the majority of confusion being with the drama genre and some confusion with the sport genre. The values of the standard deviation of the classification accuracy across the 10 simulation runs seem to be very similar to the four genre case. The standard deviation is slightly higher for the sport, scenery and drama genres, which can be expected as the classification accuracy for these genres is not as robust.

Table 7.2: Five genre confusion matrix for average classification and standard deviation for conventional RBF Network initialised with K-means clustering using $K=78$

Average Classification Confusion Matrix						
Video genre	RBF Classification					
		Sp	N	Sc	D	Ca
	Sp	84.8%	0.8%	2.4%	7.3%	4.7%
	N	0.8%	93.0%	2.4%	2.4%	1.4%
	Sc	2.6%	7.3%	77.7%	8.9%	3.5%
	D	9.1%	2.7%	4.2%	80.5%	3.5%
	Ca	3.0%	1.7%	1.7%	6.4%	87.3%
Classification Std. Dev. Confusion Matrix						
Video genre	RBF Classification					
		Sp	N	Sc	D	Ca
	Sp	4.9%	1.0%	1.7%	3.4%	2.8%
	N	0.8%	2.6%	2.1%	2.2%	1.3%
	Sc	1.9%	3.1%	4.1%	4.7%	1.4%
	D	3.2%	2.3%	2.9%	5.9%	1.8%
	Ca	1.8%	1.7%	1.3%	2.1%	1.8%

7.2.2 Conventional RBF network initialised with tree

7.2.2.1 Using leaves of pruned tree [195]

The tree-based initialisation in [195] used the leaves of a pruned decision tree as hidden neurons in an RBF network. There are a number of parameters to be set by the user to alter the classification accuracy, so some preliminary experimentation is needed to determine the best set of parameters. In Figure 7.3, the accuracy of a number of classifiers using the four genres {sport, news, scenery, drama} designed with a variety of network parameter values is shown. This includes the comparison of three options to transform the tree nodes into neuron centres and widths (i.e. always using the tree node centres as the neuron centres, or using the edge of nodes for boundary nodes of the input space and the node centre for all other neurons, or using the mean of all samples within a node as the neuron centre). There is also a comparison of setting all input weights to 1

and using a weight of 1 for relevant features and 0 for irrelevant features, where relevant features are those used in a tree branch to determine the node splits within that branch. The size of the tree will also affect the performance of the classifier as it determines the number of hidden nodes. As the terminal nodes of a pruned tree are used as hidden neurons the minimum node size K_{min} should be small (in these experiments a value of 2 is used) because the tree pruning will overcome any over-fitting. As the amount of pruning controls the complexity of the RBF network, the effect of 5 levels of pruning, which ensure the pruned tree contains at least 15, 20, 25, 30 or 35 terminal nodes are compared. Finally, the dependence on the scale factor α is also determined.

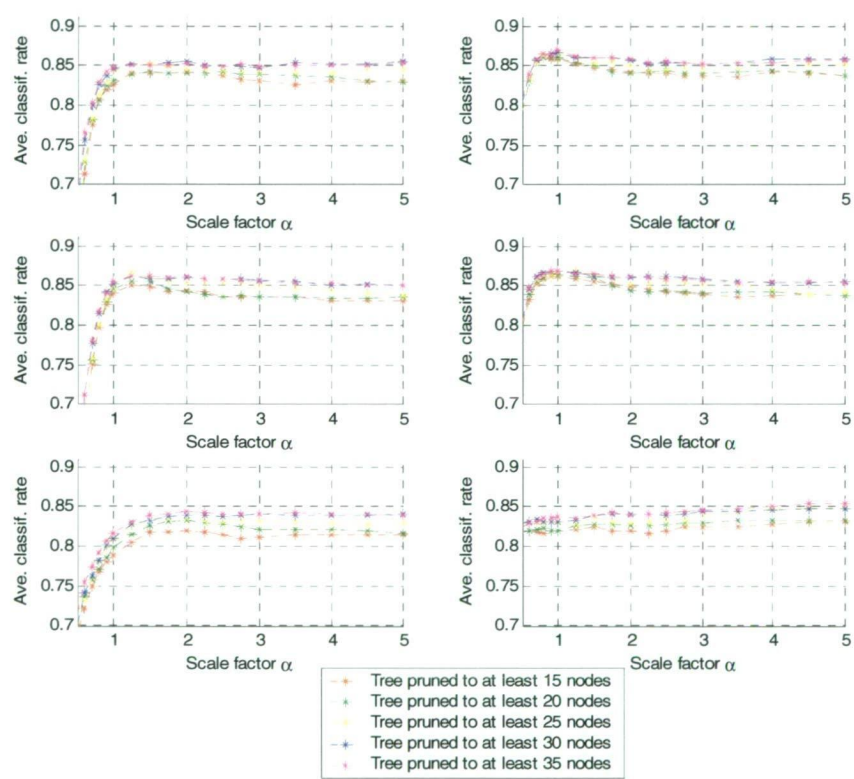


Figure 7.3: Average four genre test set classification rates for RBF network initialised with leaves of a classification tree with various amounts of pruning:

a) using tree node edge and centre with input weights all 1's	b) using tree node edge and centre with input weights 0 or 1
c) using tree node centre with input weights all 1's	d) using tree node centre with input weights 0 or 1
e) using mean of samples in tree node with input weights all 1's	f) using mean of samples in tree node with input weights 0 or 1

It can be seen in Figure 7.3 that for scale factor values greater than about 1 the classification accuracy for most classifiers is unaffected. The classifiers which use the mean of the samples in a tree node for the neuron centres are consistently less accurate by 1-2% than the other centre selection techniques. There is, however, very little difference in accuracy between the two other techniques. Using the centre of a tree node for all neurons seems to be more stable across a broader range of scale factor values. It is also evident that using input weights of zero for irrelevant features can improve the classification accuracy by 1-2%. As the amount of pruning decreases the classification accuracy tends to increase. This is expected as less pruning results in a larger tree hence more hidden nodes and a more complex classifier.

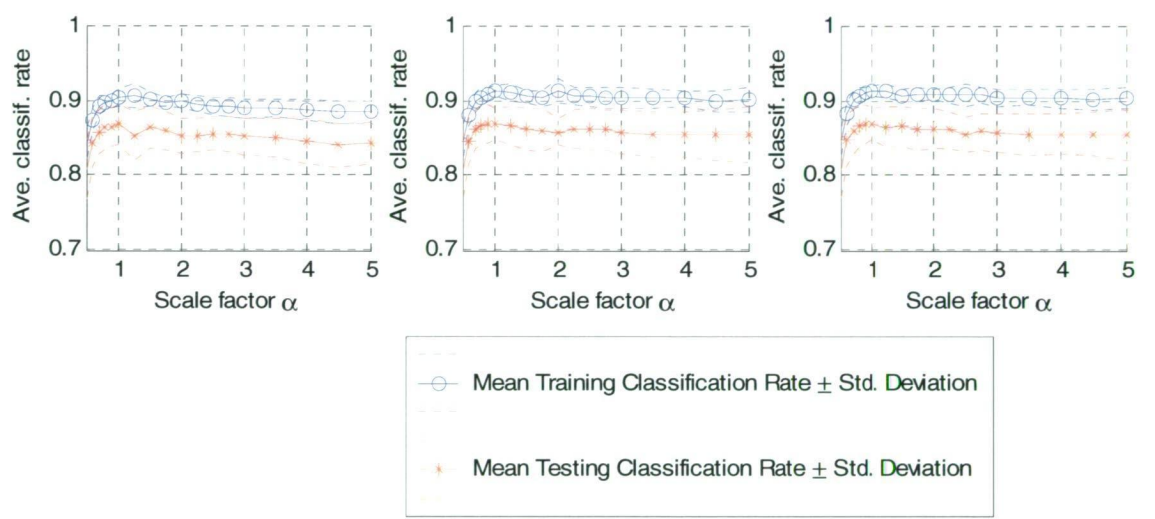


Figure 7.4: Four genre training and testing classification rate of RBF network with input weights 1 or 0 initialised with of leaf node centres using a tree pruned to:
a) at least 25 leaf nodes b) at least 30 leaf nodes c) at least 35 leaf nodes

The classification accuracies for the classifiers using trees pruned to at least 25, 30 or 35 are very similar however, so a comparison of the training and testing classification accuracy for these classifiers, using tree node centres and input weights of 0 or 1, can be seen in Figure 7.4. It is evident that as the amount of pruning decreases the difference between the training and testing accuracy increases. This is very similar to the performance of a classification tree in which the training classification rate will

continually grow as the tree increase in size, but at some point the tree will over fit to the training data and the testing classification accuracy will fall. When the tree is used to initialise an RBF network the amount of pruning needed is much less than a classification tree on its own, as the RBF network can compensate for the overtraining. It is evident from Figure 7.4 that while there is a slight peak at a scale factor of 1 using a tree pruned to at least 25 nodes, a more robust classifier would be to use a scale factor of 2 and a tree pruned to at least 30 nodes. This is because at this point the classification accuracy is very close to the maximum across all parameters and remains stable if the scale factor is increased or decreased.

Table 7.3: Four genre confusion matrix for average classification and standard deviation for conventional RBF Network initialised with leaves of a pruned tree

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	86.2%	1.5%	2.1%	10.2%
	N	0.9%	94.5%	2.3%	2.3%
	Sc	5.6%	4.5%	82.9%	7.0%
	D	11.1%	5.0%	3.5%	80.5%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	4.6%	1.4%	1.8%	3.7%
	N	1.0%	2.2%	1.4%	1.6%
	Sc	2.1%	2.1%	4.9%	4.9%
	D	5.5%	2.7%	2.1%	6.5%

The confusion matrix for the classifier using a scale factor of 2 and a tree pruned to at least 30 nodes can be seen in Table 7.3, which shows that the major improvement when compared to the *K*-means initialised RBF classifier is in the scenery genre, in which fewer shots are confused with the drama and news genres. The classification accuracies of the sport and drama genres are about the same, while the news genre has increased slightly. The increase in the scenery class is because the tree-based method takes into account the class of the training data when choosing the neuron centres. Across the 10 simulation runs for this classifier the number of hidden nodes is 30.7 ± 1.0 , so compared to the RBF initialised with *K*-means clustering, this classifier can achieve greater classification accuracy while using less than half the number of hidden neurons.

When four genres were used, trees pruned to between 25 and 35 leaf nodes produced quite similar results. This corresponds to between 6 and 9 hidden neurons per genre so if the cartoon genre is now included it would be expected that a tree pruned to between 30 and 45 leaf nodes would be sufficient. The comparison of the same classifier setups as above using trees pruned to 25, 30, 35, 40, and 45 leaf nodes can be seen in Figure 7.5.

The results are very similar to the four genre case with the classifier using tree node centre with input weights 0 or 1 performing the best, and the classification accuracy increases as the amount of pruning decreases. For this setup using a tree pruned to between 35 and 45 leaf nodes gives very similar results, so a more detailed analysis of this is shown in Figure 7.6. From this it is evident that the best set of parameters to use is using input weights set to 0 or 1, with tree node centres from a tree pruned to at least 35 leaf nodes using a scale factor of 2.5. The confusion matrix of this set-up can be seen in Table 7.4.

When this confusion matrix is compared to the five genre classification results using the *K*-means initialisation it can be seen that the increase in classification accuracy is spread across all genres although the scenery and cartoon genres improve the most. In this classifier the number of hidden nodes across the 10 simulation runs is 36 ± 1.0 and when compared to the *K*-means clustering results a slight improvement in classification accuracy is achieved using about half the number of hidden nodes. The standard

deviation of the classification accuracy across the 10 simulation runs is largest for the scenery and drama categories as these are the most prone to confusion with other genres.

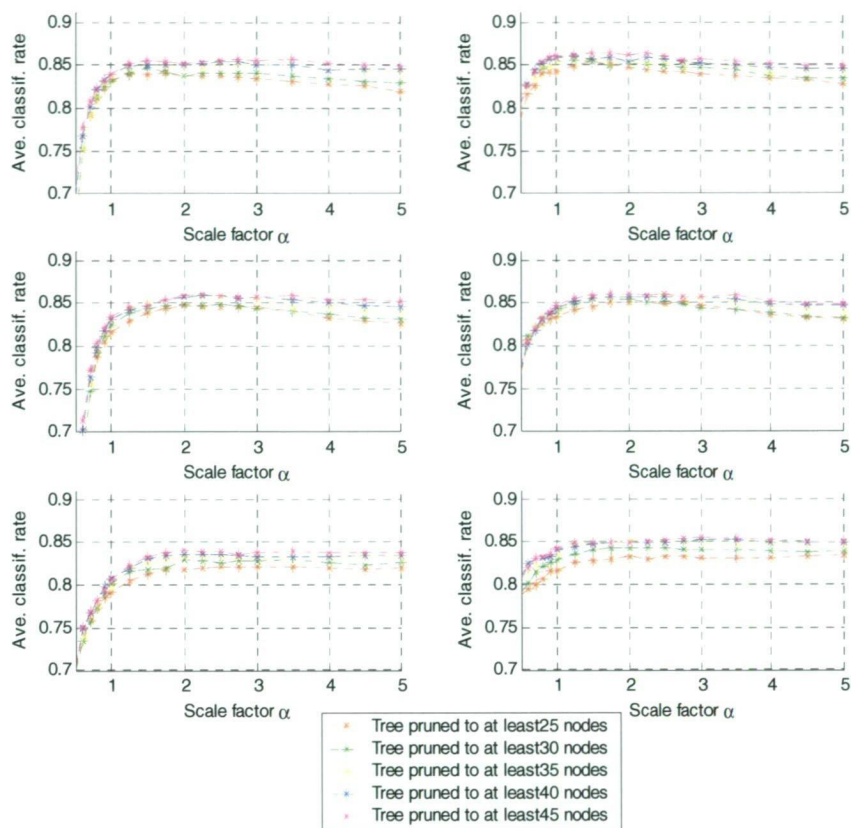


Figure 7.5: Average five genre test set classification rates for RBF network initialised with leaves of a classification tree with various amounts of pruning:

a) using tree node edge and centre with input weights all 1's	b) using tree node edge and centre with input weights 0 or 1
c) using tree node centre with input weights all 1's	d) using tree node centre with input weights 0 or 1
e) using mean of samples in tree node with input weights all 1's	f) using mean of samples in tree node with input weights 0 or 1

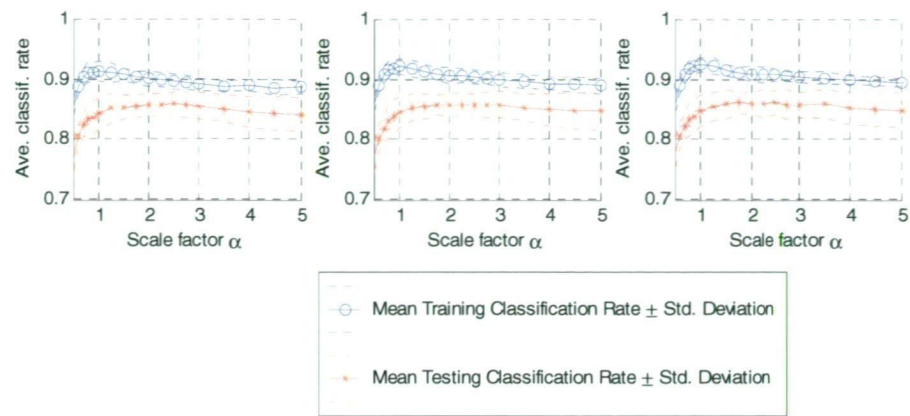


Figure 7.6: Five genre training and testing classification rate of RBF network with input weights 1 or 0 initialised with of leaf node centres using a tree pruned to:
a) at least 35 leaf nodes b) at least 40 leaf nodes c) at least 45 leaf nodes

Table 7.4: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with leaves of a pruned tree

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	84.2%	1.7%	1.5%	10.2%	2.4%
	N	1.1%	94.5%	1.8%	1.4%	1.2%
	Sc	3.2%	5.0%	80.9%	7.9%	3.0%
	D	10.8%	5.0%	3.0%	79.1%	2.1%
	Ca	2.4%	0.8%	0.6%	5.9%	90.3%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	3.3%	1.1%	1.4%	3.3%	1.8%
	N	0.7%	2.4%	0.9%	1.8%	1.1%
	Sc	1.7%	1.4%	4.0%	2.9%	2.7%
	D	6.0%	3.5%	1.7%	6.3%	1.5%
	Ca	1.8%	1.0%	1.0%	2.6%	3.5%

7.2.2.2 Using forward selection and backward elimination [196]

The concept of the tree based RBF initialisation in [196] is to traverse the tree from the root node down to grow an RBF network by adding and deleting tree nodes to improve the RBF classification accuracy. Like the previous tree based RBF initialisation method, the effect of various network parameters needs to be investigated. Once again the effect of the neuron centre selection and the scale factor is investigated. In this method as the tree is not pruned the effect of the minimum tree node size K_{min} is also investigated. In the previous method each hidden neuron had a unique receptive field as each neuron was initialised from a leaf node belonging to a unique region of the input space. In this method, however, a number of nodes from the same branch in a tree may be included in the RBF network so the receptive field of a number of nodes may cover the same area of the input space. As a result the input weights should all be set to 1 to ensure no information is discarded from nodes from higher in the tree.

The effect of each of the factors on the training and testing classification rate can be seen in Figure 7.7. In general using the mean of samples in a tree node gives a slightly degraded accuracy and although the two other methods are very similar, always using the node centre seems to be slightly more robust across the other parameters. Using a K_{min} value of 14 gives a slight degradation in classification accuracy as the classifier is not complex enough because there are insufficient hidden nodes. While the other K_{min} values produced similar classification accuracy the best results are achieved by always using the centre of a tree node with scale factor of 2.5 and K_{min} value of 2. For this set of parameters, the confusion matrix is given in Table 7.5.

It can be seen in Table 7.5 that compared to the four genre classification results for RBF network initialised with K -means clustering, there is a vast improvement in the classification of the scenery genre due to a large reduction in the confusion between the scenery genre and the news and drama genres. There is also an increase in the accuracy of the sport and news genres and a slight decrease in the drama genre. The sport genre is primarily confused with the drama genre, the drama genre is confused with all other genres but more so with the sport genre, and the scenery genre is equally confused with all other genres. The number of hidden nodes across the 10 simulation runs is 23.5 ± 4.0

so this increase in classification accuracy is achieved using slightly more than a third of the hidden neurons needed in the *K*-means clustering initialisation.

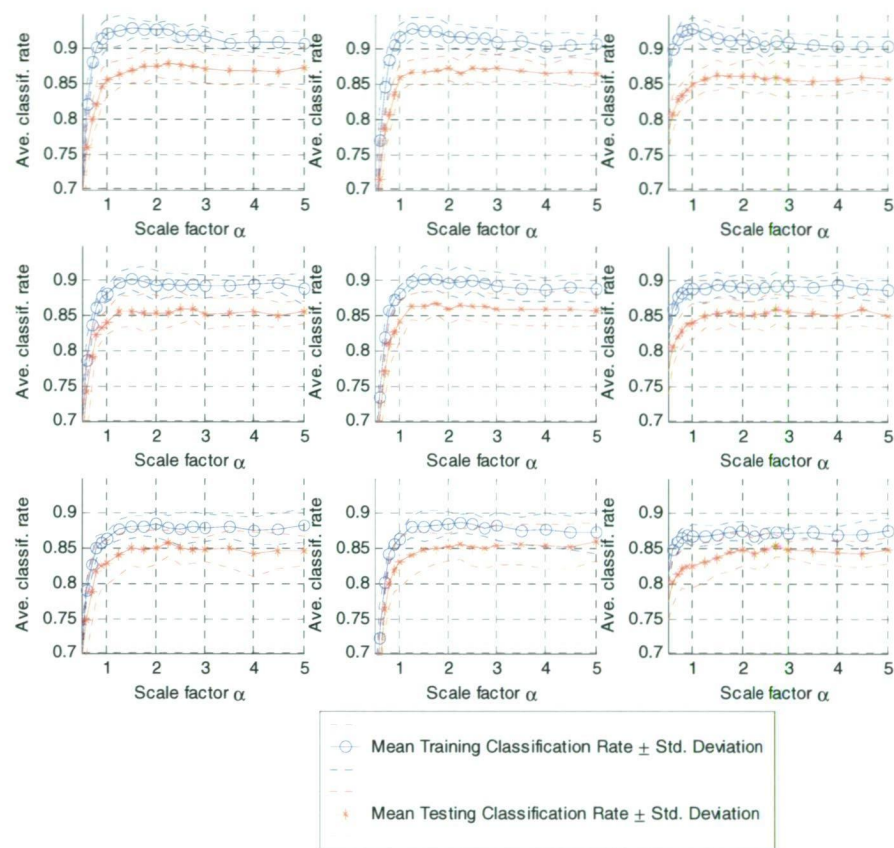


Figure 7.7: Four genre classification results for RBF network initialised using forward selection and backward elimination method:

(a) using edge and centre of tree node with $K_{min} = 2$	(b) using tree node centre with $K_{min} = 2$	(c) using mean of samples in tree node with $K_{min} = 2$
(d) using tree node edge and centre with $K_{min} = 8$	(e) using tree node centre with $K_{min} = 8$	(f) using mean of samples in tree node with $K_{min} = 8$
(g) using tree node edge and centre with $K_{min} = 14$	(h) using tree node centre with $K_{min} = 14$	(i) using mean of samples in tree node with $K_{min} = 14$

Table 7.5: Four genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with tree using forward selection and backward elimination

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	88.5%	1.4%	2.0%	8.2%
	N	1.2%	95.9%	2.1%	0.8%
	Sc	4.7%	4.7%	85.0%	5.6%
	D	10.8%	4.4%	4.5%	80.3%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	4.5%	1.1%	2.3%	2.8%
	N	1.1%	2.4%	1.5%	1.2%
	Sc	2.2%	2.5%	4.3%	2.1%
	D	5.7%	2.9%	2.6%	6.3%

When the cartoon genre is also considered, once again very similar results are seen. The same comparison of the various parameters for five genre classification is shown in Figure 7.5. It can be seen that in this case all three options for the choice of node centre produce very similar results, although once again always choosing the node centre seems to be the most robust to variations in the other parameters. Once again it seems the best set of parameters is always using the centre of a tree node with scale factor of 2.5 and K_{min} value of 2.

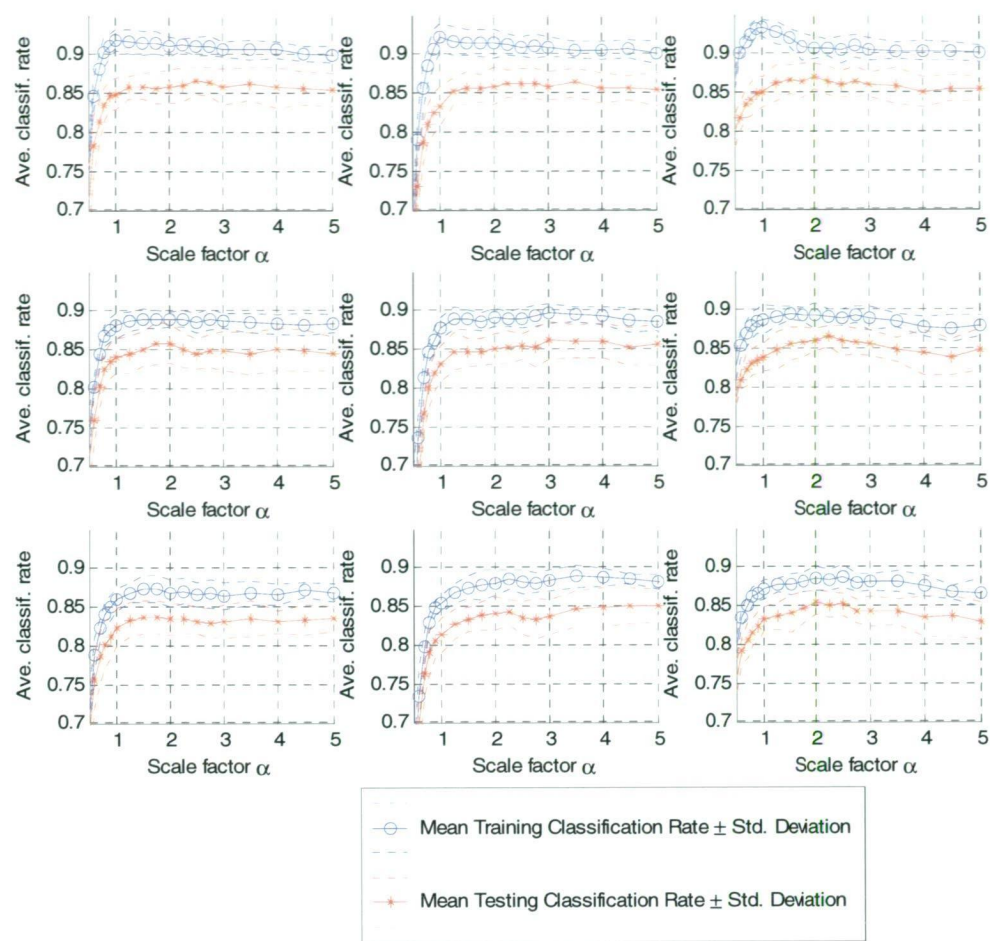


Figure 7.8: Five genre classification results for RBF network initialised using forward selection and backward elimination method:

(a) using edge and centre of tree node with $K_{min} = 2$	(b) using tree node centre with $K_{min} = 2$	(c) using mean of samples in tree node with $K_{min} = 2$
(d) using tree node edge and centre with $K_{min} = 8$	(e) using tree node centre with $K_{min} = 8$	(f) using mean of samples in tree node with $K_{min} = 8$
(g) using tree node edge and centre with $K_{min} = 14$	(h) using tree node centre with $K_{min} = 14$	(i) using mean of samples in tree node with $K_{min} = 14$

The confusion matrix for the best parameter set is shown in Table 7.6. If these results are compared to the K -means initialized it is evident that the classification accuracy sport, news, scenery and cartoon genres have all improved by about 2-4%, while the classification accuracy of the drama genre has remained the same. There is no

significant change in the confusion between any two genres although interestingly the confusion between the drama and sport genres has increased while the confusion between the drama and scenery genres and the drama and cartoon genres has decreased which has resulted in no change in the classification accuracy of the drama genre. The number of hidden nodes used in this classifier across the 10 simulation runs is 29.6 ± 5.3 which once again is slightly greater than a third of the hidden nodes used in the *K*-means initialisation. The standard deviation of the classification accuracy across the 10 simulation runs is largest for the drama genre indicating this genre gives the least robust classification results.

Table 7.6: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with tree using forward selection and backward elimination

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	87.4%	1.5%	1.7%	6.1%	3.3%
	N	0.9%	95.2%	1.8%	1.2%	0.9%
	Sc	4.5%	5.0%	81.1%	7.3%	2.1%
	D	11.8%	2.7%	2.7%	80.3%	2.4%
	Ca	2.1%	0.6%	0.6%	5.5%	91.2%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	2.4%	1.5%	1.3%	3.2%	1.9%
	N	0.7%	2.1%	1.3%	1.1%	0.7%
	Sc	3.2%	1.4%	3.0%	1.1%	1.7%
	D	6.3%	2.6%	1.3%	7.9%	1.5%
	Ca	1.9%	1.0%	0.7%	2.6%	3.4%

7.2.3 Conventional RBF network optimised with gradient descent

The initialisation of an RBF network with K -means clustering or the leaves of a pruned tree can result in the sub-optimal basis functions. The classification accuracy can be increased by optimising the hidden and output layer parameters with gradient descent, which is investigated for the best performed classifiers for the above RBF initialisation methods. The confusion matrices of a gradient descent optimised conventional RBF network initialised with K -means clustering using four and five genres are shown in Table 7.7 and Table 7.8 respectively. The confusion matrices of a gradient descent optimised conventional RBF network initialised with the leaves of a pruned tree for four and five genres are shown in Table 7.9 and Table 7.10 respectively. The tree-based forward selection and backward elimination method of RBF initialisation is not suitable for optimisation with gradient descent as there are many overlapping neurons which can cover the space of a number of genres in the input space.

Table 7.7: Four genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with K -means with $K=64$ optimised with gradient descent

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	86.2%	1.1%	3.5%	9.2%
	N	0.8%	94.5%	2.4%	2.3%
	Sc	4.4%	7.6%	77.6%	10.5%
	D	9.4%	3.2%	4.5%	82.9%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	5.0%	1.9%	2.6%	4.5%
	N	0.8%	2.3%	1.7%	1.8%
	Sc	1.4%	2.8%	2.8%	2.6%
	D	3.3%	2.6%	2.7%	4.1%

It can be seen in Table 7.7 that for the four genre case there is only slight improvement shown in the drama genre due to a decrease in confusion with the sport genre, and a slight improvement in the news genre which is primarily due to a decrease in the confusion with the drama genre. In all of the genres the standard deviation of the classification accuracy across the 10 simulation runs has also decreased. In the five genre case in Table 7.8 there is also little improvement shown, except for a slight improvement in the classification accuracy of the scenery genre due to a decrease in the confusion with the drama genre, and a slight improvement in the drama genre due to a decrease in the confusion with the sport genre. This indicates that the use of *K*-means clustering to initialise an RBF network is not sufficient in this application as the classifier produced is too complex and does not generalise well, even after gradient descent optimisation.

Table 7.8: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with *K*-means with *K*=78 optimised with gradient descent

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	84.8%	0.8%	2.6%	7.0%	4.8%
	N	0.8%	93.2%	2.6%	2.3%	1.2%
	Sc	2.4%	7.6%	79.1%	7.7%	3.2%
	D	8.9%	2.6%	4.1%	81.2%	3.2%
	Ca	3.0%	1.5%	1.5%	6.4%	87.6%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	4.5%	1.0%	1.7%	3.3%	2.6%
	N	0.8%	2.5%	1.9%	2.3%	1.3%
	Sc	1.8%	3.0%	3.7%	3.9%	1.6%
	D	2.8%	2.4%	3.0%	5.3%	1.6%
	Ca	2.2%	1.7%	1.2%	2.5%	1.8%

It can be seen in Table 7.9, however, that when the leaves of a pruned tree are used to initialise an RBF network it provides a good starting point for the gradient descent optimisation. This is because the network produce is less complex so is more robust to minor alterations of the gradient descent process. In the four genre case the classification accuracy of the news genre is unchanged (but is the most accurate of all genres) and the classification accuracy other genres are improved by 2.5-5%. The improvement in classification of the sport genre is primarily due to a decrease in the confusion with the drama genre, the improvement in the scenery genre due to a decrease in confusion with the sport genre, and the improvement in the drama genre due to a decrease in the confusion with the sport and news genres.

Table 7.9: Four genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with leaves of pruned tree optimised with gradient descent

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	89.5%	1.2%	1.2%	8.0%
	N	1.1%	94.8%	1.4%	2.7%
	Sc	3.9%	4.1%	85.3%	6.7%
	D	9.4%	2.4%	2.9%	85.3%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	4.2%	0.9%	1.3%	3.1%
	N	1.2%	2.5%	1.4%	1.9%
	Sc	1.4%	2.4%	3.0%	2.4%
	D	4.9%	2.2%	2.2%	5.2%

From Table 7.10, it is evident that in the five genre case, once again, the news genres is most accurate and does not improve, while the classification accuracy of all other genres is increased by 2-4%. The increase in classification accuracy of the sport genre is once

again due to a decrease in the confusion with the drama genre, the scenery genre is less confused with the sport and news genres, the drama genre is less confused with the news genre, and the cartoon genre is less confused with the drama genre. It is also evident from Table 7.9 and Table 7.10 that in both the four genre and the five genre case, there is a slight decrease in the standard deviation of the classification accuracy across all genres indication that the gradient descent optimised classifier is more robust across different training sets.

Table 7.10: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with leaves of pruned tree optimised with gradient descent

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	88.2%	1.5%	1.2%	6.5%	2.6%
	N	0.9%	94.7%	1.5%	1.7%	1.2%
	Sc	2.3%	3.9%	83.6%	7.7%	2.4%
	D	10.0%	2.6%	2.6%	82.6%	2.3%
	Ca	2.7%	0.2%	0.5%	4.1%	92.6%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	3.3%	1.0%	1.3%	3.2%	1.7%
	N	0.7%	1.8%	1.2%	1.6%	1.1%
	Sc	1.7%	1.7%	3.0%	2.5%	1.4%
	D	5.0%	2.0%	1.9%	6.0%	1.6%
	Ca	2.0%	0.5%	0.7%	1.7%	2.4%

7.2.4 Generalised RBF network with input feature weights

The generalised RBF network structure attempts to optimise the classification accuracy by increasing the separation between classes at the hidden layer using input feature weights. The effectiveness of this structure is investigated for the best performed classifiers for the RBF initialisation methods above. The confusion matrices for four and five genre classification of a generalised RBF network initialised with *K*-means clustering are shown in Table 7.11 and Table 7.12 respectively. The confusion matrices for four and five genre classification of a generalised RBF network initialised with the leaves of a pruned tree are shown in Table 7.13 and Table 7.14 respectively. The tree-based forward selection and backward elimination method of RBF initialisation was not suitable for initialising a generalised RBF network, as there are many overlapping neurons which can cover the space of a number of genres in the input space so the determination of hidden neurons targets is not possible using the method proposed.

Table 7.11: Four genre confusion matrix for average classification and standard deviation for a generalised RBF network initialised with *K*-means clustering

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	84.5%	1.4%	3.9%	10.2%
	N	1.1%	93.6%	2.0%	3.3%
	Sc	3.2%	7.7%	79.4%	9.7%
	D	9.7%	2.9%	3.5%	83.9%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	2.8%	1.4%	2.5%	3.3%
	N	1.2%	3.2%	2.3%	2.2%
	Sc	1.7%	2.9%	3.4%	3.5%
	D	3.7%	2.2%	1.8%	4.4%

It can be seen in Table 7.11 and Table 7.12 that there is only a slight improvement using a generalised RBF network when the network is initialised using *K*-means clustering. This is due to the large number of hidden nodes compared to the training data making the determination of hidden layer target values inaccurate. In the four genre case, there is a slight decrease in classification accuracy for the sport genre due to an increase in confusion with drama genre, an increase in the accuracy of the drama genre primarily due to a decrease in the confusion with the sport genre, a slight increase in the news genre due to a decrease in confusion with the drama genre, and a slight increase in the scenery genre due to a decrease in the confusion with the sport genre.

Table 7.12: Five genre confusion matrix for average classification and standard deviation for a generalised RBF network initialised with *K*-means clustering

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	85.3%	0.8%	2.4%	7.6%	3.9%
	N	0.8%	92.9%	2.4%	2.7%	1.2%
	Sc	2.6%	7.1%	78.9%	7.3%	4.1%
	D	9.2%	2.7%	3.5%	81.2%	3.3%
	Ca	2.9%	1.2%	2.0%	6.1%	87.9%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	4.4%	1.0%	2.6%	2.5%	1.5%
	N	0.8%	2.9%	1.9%	2.0%	1.3%
	Sc	2.0%	3.2%	4.0%	4.5%	1.7%
	D	3.4%	2.3%	2.3%	6.2%	1.9%
	Ca	1.6%	1.1%	1.4%	1.9%	3.0%

In the five genre case, the classification accuracies of the sport, news, drama and cartoon genres do not change significantly, while there is a slight improvement in the accuracy of the scenery genre due to a decrease in the confusion with the drama genre. There is also a tendency in the five genre case for the standard deviation of the classification

accuracy across the 10 simulation runs to be worse using a generalised RBF network indicating that using *K*-means clustering does not produce a robust classifier.

It can be seen from the confusion matrices in Table 7.13 and Table 7.14 that when the leaves from a pruned tree are used to initialise a generalised RBF network, the classification accuracy does improve slightly. In the four genre case all but the news genre, which still has the highest classification accuracy, improve by about 2%, with the improvement in both the sport and scenery genres due to a decrease in the confusion with drama genre, and the improvement in the drama due to a decrease in the confusion with the news and sport genres.

Table 7.13: Four genre confusion matrix for average classification and standard deviation for a generalised RBF network initialised with leaves of pruned tree

Average Classification Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	87.9%	1.4%	2.3%	8.5%
	N	0.9%	94.5%	2.3%	2.3%
	Sc	5.2%	4.2%	84.7%	5.9%
	D	10.3%	3.3%	3.6%	82.7%

Classification Std. Dev. Confusion Matrix					
		RBF Classification			
		Sp	N	Sc	D
Video genre	Sp	5.1%	1.4%	1.4%	4.5%
	N	0.7%	3.3%	1.2%	2.6%
	Sc	2.6%	1.8%	4.1%	3.3%
	D	6.1%	3.0%	2.2%	7.5%

In the five genre case, the classification accuracy of all genres improved between 1-2%, in which the improvement in the sport and scenery genres is due to a decrease in the confusion with the drama genre, the improvement in the news genre is spread across a

number of genres, the improvement in the drama genre is due to decrease in confusion with the sport genre, and the improvement in the cartoon genre is due to decrease in confusion with drama and news genres. In both cases there is no significant change in the standard deviation of classification accuracy across the 10 simulation runs.

Table 7.14: Five genre confusion matrix for average classification and standard deviation for a generalised RBF network initialised with leaves of pruned tree

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	86.4%	1.5%	1.8%	7.3%	3.0%
	N	0.8%	95.5%	1.8%	1.2%	0.8%
	Sc	3.2%	5.5%	81.7%	5.9%	3.8%
	D	9.7%	5.0%	2.6%	79.8%	2.9%
	Ca	2.3%	1.7%	0.5%	4.4%	91.2%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Sc	D	Ca
	Sp	4.1%	1.0%	2.1%	3.5%	1.8%
	N	0.8%	2.4%	1.3%	1.5%	1.2%
	Sc	2.3%	1.9%	3.9%	2.1%	2.4%
	D	6.1%	3.6%	1.8%	6.7%	1.4%
	Ca	1.9%	1.6%	0.7%	2.7%	2.6%

7.3 Comparison of techniques

A comparison of the average classification accuracy across all genres for the above classifiers can be seen in Table 7.15 and Table 7.16 for the four genre set and the five genre set respectively. In both genre sets all RBF network classifiers which are initialised with *K*-means clustering result in less accurate performance than the tree-based initialisation techniques. The *K*-means initialised classifiers which are more complex than necessary (i.e. too many hidden neurons) as the output class of the training

data is not used in the initialisation process which leads to a less robust classifier in the presence of overlapping data. The tree-based classifiers use between a third and half as many hidden neurons (approximately 25-30 in the four genre case and 30-35 in the five genre case) as the *K*-means initialised classifiers. As a result the tree-based classifiers that are more robust to a noisy data set as they do not over fit to the training data. This is evident when comparing the difference between the training and testing classification accuracies for each classifier which is smaller for the tree-based classifiers.

Table 7.15: Comparison of classification accuracy across four genres {sport, news, scenery, drama} for a number of classifiers

Classifier		Classification Accuracy (Training)	Classification Accuracy (Testing)	No. hidden nodes	Training time
RBF Type	Initialisation				
Conventional	<i>K</i> -means	93.2 ± 1.3%	84.1 ± 1.7%	66	0.18 ± 0.01s
Conventional (optimised with grad. desc.)	<i>K</i> -means	93.4 ± 2.7%	85.3 ± 1.2%	66	148.3± 10.1s
Generalised	<i>K</i> -means	93.3 ± 1.2%,	85.4 ± 1.6%	64	23.5 ± 1.2s
Conventional	Tree-based 1	90.5 ± 1.0%	86.0 ±2.8%	30.7 ± 1.0	11.2 ± 0.4s
Conventional (optimised with grad. desc.)	Tree-based 1	93.4 ± 1.3%	88.8 ± 1.9%	30.7 ± 1.0	90.5 ± 3.5s
Generalised	Tree-based 1	89.3 ± 1.5%	87.5 ± 2.8%	30.7 ± 1.0	35.6 ± 1.0s
Conventional	Tree-based 2	91.5 ± 1.7%	87.4 ± 2.1%	23.5 ± 4.0	3.0 ± 0.2s
RBF initialisation Key: Tree-based 1 = leaves of pruned tree, Tree-based 2 = forward selection and backward elimination					

The conventional RBF initialised with a tree using forward selection and backward elimination and the generalised RBF initialised using the leaves of a pruned tree produce very similar classification accuracies for both genres sets. The difference between the training and testing classification is slightly less for the generalised RBF but the training time is much less for the RBF network initialised with a tree using forward selection and backward elimination. The best classification accuracy is achieved using a conventional

RBF network initialised with the leaves of a pruned tree which is optimised with gradient descent. The downfall to this method is that the training time is much longer.

Table 7.16: Comparison of classification accuracy across five genres {sport, news, scenery, drama, cartoon} for a number of classifiers

Classifier		Classification Accuracy (Training)	Classification Accuracy (Testing)	No. hidden nodes	Training time
RBF Type	Initialisation				
Conventional	K-means	92.4 ± 1.1%	84.7 ± 2.5%	78	0.3 ± 0.03s
Conventional (optimised with grad. desc.)	K-means	92.7 ± 0.8%	85.2 ± 2.2%	78	197.2± 15.3s
Generalised	K-means	92.4 ± 1.0%	85.2 ± 2.3%	78	25.3 ± 9.2
Conventional	Tree-based 1	89.5 ± 1.2%	85.8 ± 2.0%	36 ± 1.0	16.7 ± 1.1s
Conventional (optimised with grad. desc.)	Tree-based 1	92.0± 1.6%	88.3± 1.8%	36 ± 1.0	130.5 ± 8.9s
Generalised	Tree-based 1	89.5± 0.7%	86.9 ± 1.9%	36 ± 1.0	60.2 ± 10.6s
Conventional	Tree-based 2	90.7 ± 1.5%	87.0 ± 1.6%	29.6 ± 5.3	5.49 ± 0.8s
RBF initialisation Key: Tree-based 1 = leaves of pruned tree, Tree-based 2 = forward selection and backward elimination					

7.4 Comparisons with other work

There are a number of factors that make an exact comparison between the classification approaches presented in this thesis and previous works seen in the literature hard to make. The two major ones are the different video databases used for testing, and the set of genres used. To overcome the problems of different data sets, the video clips in the database are chosen to give the best representative view of each genre. In order to compare results with the approaches in [131,132,133] further experiments were conducted using the genre set {sport, news, cartoon, commercial, music}, and to compare with the results seen in [134] further experiments were conducted using the genre set {sport, news, cartoon, drama, music}. The experiments for each genre set used

a conventional RBF network initialised with the leaves of a classification tree and optimised with gradient descent. The results of these experiments can be seen in Table 7.17 which shows that the techniques used in this thesis compare favourably with the other techniques from the literature, both those using just visual features, and those using a combination of visual and audio features. An important consideration when this comparison is made is the length of video clip to be classified for each of the techniques, as it is shown in [131,133] that in general as the clip length increases the classification accuracy also increases. For the other techniques a clip length of between 20-60 seconds is used, whereas in this work the classification is at a shot level which on average is between 1 and 5 seconds in length, and rarely longer than 10 seconds.

Table 7.17: Comparison of video genre classification approaches seen in the literature

Genre Set	Authors	Classifier	Clip length (seconds)	Database size (clips per genre)	Average classification rate	Gradient descent optimised TB-RBF classification rate
Sport, News, Cartoon, Commercial, Music	[131] (2000)	C4.5 decision tree	60	100	83.1% (visual features)	83.6 ± 1.8%
	[132] (2002)	GMM (64 components)	20	180	73.6% (visual features) 76.8% (audio features) 84.3% (combined)	
	[133] (2003)	GMM (3-10 components)	40	90	87%. (combined)	
Sport, News, Cartoon, Drama, Music	[134] (2004)	CART (GINI splitting criterion)	60s	72	72.0% (visual features) 72.8% (audio features) 88.8% (combined)	86.2 ± 1.5%

The confusion matrix for the genre set {sport, news, cartoon, music, commercial}, using the conventional RBF network initialised with the leaves of a pruned tree and optimised with gradient descent, can be seen in Table 7.18. In this case, the news genre is well classified and is not significantly confused with any other genre. The classification accuracies for the sport and cartoon genres are also good, although the sport genre confused with music genre and to a lesser extent the cartoon and commercial genres and the cartoon genre is significantly confused with the commercial genre and to a lesser extent with the music and sport genres. The classification accuracy of the music genre is not as good and it is significantly confused with the commercial and sport genres and to a lesser extent with the cartoon genre. The classification accuracy of the commercial genre is significantly worse than the other genre and it is significantly confused with the music, cartoon and sport genres.

Table 7.18: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with leaves of pruned tree optimised with gradient descent for the genre set {sport, news, cartoon, music, commercial}.

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Ca	M	Co
	Sp	87.3%	1.4%	3.5%	5.5%	2.4%
	N	1.1%	97.1%	0.2%	0.0%	1.7%
	Ca	1.8%	1.1%	85.6%	2.9%	8.6%
	M	7.7%	0.2%	3.5%	78.2%	10.5%
	Co	7.1%	1.1%	10.5%	11.7%	69.7%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	Ca	M	Co
	Sp	5.1%	1.3%	2.4%	3.1%	1.8%
	N	1.0%	1.4%	0.5%	0.0%	1.3%
	Ca	1.9%	1.5%	3.6%	2.3%	2.0%
	M	3.4%	0.5%	2.7%	5.3%	4.3%
	Co	1.7%	1.8%	3.1%	3.8%	4.5%

The confusion matrix for the genre set {sport, news, drama, cartoon, music}, using the conventional RBF network initialised with the leaves of a pruned tree and optimised with gradient descent, can be seen in Table 7.19. The news genre is again classified accurately and is not significantly confused with any other genre. The cartoon genre is also classified accurately, although it is sometimes confused with the music genre and to a lesser extent with the sport and drama genres. The sport and music genres have good classification accuracies, with the sport genre sometimes being confused with music and drama genres and less so with the cartoon genre, and the music genre is confused with the sport, drama, and cartoon genres. The drama genre has the least accurate classification accuracy, although it is still reasonable, and it is significantly confused with the sport and music genres.

Table 7.19: Five genre confusion matrix for average classification and standard deviation for conventional RBF network initialised with leaves of pruned tree optimised with gradient descent for the genre set {sport, news, drama, cartoon, music}.

Average Classification Confusion Matrix						
RBF Classification						
Video genre		Sp	N	D	Ca	M
	Sp	83.6%	1.4%	6.5%	3.2%	5.3%
	N	1.1%	95.6%	1.4%	0.8%	1.2%
	D	8.9%	2.0%	79.7%	2.0%	7.4%
	Ca	2.3%	0.5%	2.4%	89.2%	5.6%
	M	6.8%	0.3%	5.0%	5.0%	82.9%
Classification Std. Dev. Confusion Matrix						
RBF Classification						
Video genre		Sp	N	D	Ca	M
	Sp	4.7%	1.6%	3.7%	2.7%	1.9%
	N	1.2%	2.2%	1.3%	1.0%	1.5%
	D	4.6%	2.3%	6.0%	1.7%	3.1%
	Ca	1.7%	0.7%	1.8%	2.3%	2.3%
	M	2.9%	0.9%	4.1%	1.9%	4.3%

Chapter 8

Conclusions and Further Research

8.1 Conclusions

The work in this thesis investigates the problem of the automatic classification of high level video genres. There has been some previous work in this area by other authors which in general use visual features, audio features, or a combination of both in the classification process. The features are generally computed using raw, uncompressed visual or audio signals, and the classification is performed on video segments between 20-60 seconds in length. The first objective of this thesis is to investigate the use of visual features which are computed using compressed domain information extracted directly, or with minimal decoding, from the compressed bit stream. The second objective is to perform the classification at a shot level (i.e. each video segment used in the classification process corresponds to a video shot) as this will ensure each segment classified will only contain a single genre. The final objective is to perform a detailed analysis of a number of classification techniques so as to find the most robust approach for the problem of video genre classification.

In the investigation of possible compressed domain visual features to be used in video genre classification, the author has made a number of contributions to the field. The use of motion vectors from an MPEG-1 file to compute features relating to the motion in a sequence was first explored. It was seen that in certain circumstances the motion vectors in an MPEG-1 file do not accurately describe the motion in a video sequence. This primarily occurs in uniform regions where the spatial activity is low, so a spatial activity metric calculated from discrete cosine transform (DCT) coefficients present in an

MPEG-1 sequence is proposed. It is shown that this metric can be used to detect unreliable motion vectors and discard them from further processing. A robust estimation of a parametric camera motion model from the filtered motion vectors using a least median of squares (LMedS) technique is also proposed. The LMedS technique is shown to be more robust and can cope with more outliers as a result of unreliable motion vectors or object motion than the commonly used M-estimator technique. To ensure the motion vectors used in the camera estimation all reference across the same number of frames in the same temporal direction, only the motion vectors from P-frames are used as a sub-sample of the motion in a sequence. From this estimated camera motion model, four features describing the distribution of the intensity of camera and object motion across a shot are proposed. First, for each P-frame in a shot the average camera and object motion intensity values are calculated. The camera motion intensity is the average motion vector magnitude for the estimated camera motion model and the object motion intensity is the average magnitude of the difference between the camera motion model and the MPEG motion vectors. The four motion features for a shot are then the mean and standard deviation of camera and object motion intensity values across a shot. Results show that these motion features are representative of the motion seen in shots of each genre. For example the features produce high values for mean and standard deviation of camera and object motion for shots in the sport genre which is expected because the camera is continuously following erratically moving objects. In the music genre however, the mean object motion values are high but the other features are slightly lower because in the music genre there is commonly a large amount of object motion in the form of dancing, but the motion is more likely to be staged so there is less camera motion, and the intensity of the motion is more likely to be less stop-start than in the sport genre.

A further six visual features were proposed which use the *spatial activity function* to measure the reliability of motion vectors to give a coarse measure of evolution of the spatio-temporal content of a video shot. To achieve this, the sum of the spatial activity for blocks containing reliable, unreliable, and no motion vectors (i.e. I-blocks) within a frame is calculated. The evolution of these values across a shot, known as the *activity power flow*, is then measured for each block type using the mean and standard deviation

of the activity power flow. An analysis of the activity power flow metrics across a number of video genres show that they are harder to relate to genre specific content than the motion intensity metrics. It is evident, however, that the cartoon, commercial, and music genres in general contain more high spatial activity regions which relates to the more common use of synthetic content as opposed to the more natural content occurring in the other genres. It is also evident that the sport, music, cartoon, and commercial genres contain large values of activity power flow for I-blocks which is an indication of motion which can not be describe by the 2-D motion vectors present in an MPEG file such as rotating objects, object motion which uncovers or covers background, and some computer graphic or special effects.

The thesis also presents the first shot-based video classification results based on the analysis of a number of classification techniques. The classifiers analysed are a conventional RBF network initialised with *K*-means clustering, a conventional RBF network initialised with the leaves of pruned classification tree, a conventional RBF network initialised using classification tree-based forward selection and backward elimination, a conventional RBF network optimised with gradient descent, and a generalised RBF network. Initial experiments included a detailed analysis of the optimisation of a number of parameter values for each classifier on the genre sets {sport, news, scenery, drama} and {sport, news, scenery, drama, cartoon}. It was observed the accuracy of the classifiers initialised with *K*-means clustering performed slightly worse than other classifiers and also used between two and three times more hidden neurons than the tree-based classifiers. As a result the tree-based techniques produce classifiers that are less complex, more robust to a noisy data set as they do not over fit to the training data, and more robust in the presence of overlapping data. Of the tree-based techniques, the conventional RBF network initialised with a tree using forward selection and backward elimination and the generalised RBF initialised using the leaves of a pruned tree produced very similar classification accuracies for both genres sets, but the best classification accuracy is achieved using a conventional RBF network initialised with the leaves of a pruned tree which is optimised with gradient descent. The only downfall to this method is that the training time is much longer.

The classification accuracy for the tree-based radial basis function optimised with gradient descent process was then tested for the genre sets {sport, news, cartoon, commercial, music} and {sport, news, cartoon, drama, music} in order to provide some comparison with other approaches seen in the literature. The classification accuracy of the most robust classifier analysed in this work compares favourably with previous video genre classification works. For the genre set {sport, news, cartoon, commercial, music} previous works reported classification accuracies of 83.1% [131] and 73.6% [132] using just visual features and 84.3% [132] and 87% [133] using combined audio and visual features compared with a classification accuracy of 83.6% in this thesis. In this genre set the commercial genre proved to be the hardest to classify by a significant margin followed by the music genre. For the genre set {sport, news, cartoon, drama, music} previous work [134] reported classification accuracies of 72.0% for visual features and 88.8% using combined audio and visual features compared with a classification accuracy of 86.2 % in this thesis. In this genre set the drama and music genres were the hardest to classify. These comparisons are more favourable when considering only visual features were used in this work and the classification is performed at the shot level, which on average has a length of 1-5 seconds, where as previous works classified video segments that were between 20-60 seconds long.

8.2 Further Work

The primary function of a video genre classification scheme is to segment a video sequence into regions of the same genre. The process of genre classification will in general be used as one of the first steps in any CBVIR system, either to identify relevant video segments which genre specific processing techniques can be performed on, or to separate a database into clusters of similar genre to reduce the search space. The results shown in this thesis show great promise although some further research is needed to ensure the classification accuracy is robust enough to be used in a commercial system.

In the extraction of visual features, currently the slowest component is the least median-of-squares camera motion estimation step. It was evident in the comparison of the M-estimator approach and the LMedS approach that much of the time both approaches

produce similar results. In order to improve the computational efficiency of the estimation process, while retaining the robustness of the LMedS approach, it may be possible to use the M-estimator as an initial guess for the camera motion model and use the LMedS techniques when the M-estimator fails. The reliability of the M-estimation technique could be tested by comparing the camera motion parameters from subsequent estimations as it is unlikely there will be a large change in camera motion from one frame to the next. Parallel computation could also be used to improve the efficiency of the LMedS process as each p -tuple can be processed simultaneously. Finally, it may also improve the efficiency if the camera motion model from the previous frame is used to initialise the calculations for current frame. In terms of the features used in this thesis, the inclusion of audio features would more than likely improve the classification result. There is also a need for more research into possible features which are geared towards improving the classification accuracy of some of the harder to classify genres, in particular the commercial genre but also the drama, music, and scenery genres. The commercial genre was significantly the worst performed genre in this work and a possible feature to use is a text detector as many commercial shots contain a lot of overlaid text. It may be possible to implement this detector in the compressed domain with minimal decoding using the DCT coefficients present in I-frames. Another approach may be to use a separate method for commercial detection prior to the genre classification step.

In terms of genre classification, the best classifier used in this thesis seems to be quite robust so the next step to take is to look at determining the exact shots where the boundaries between two genres occur within a video sequence. One approach for this is to combine the shot-level genre classification results over time to locate possible groups of shots which may contain a genre boundary, and use further processing to locate the exact boundary. It is also necessary to build the range of genres for which the classifier will work to include not only other genres but also sub-genres. The inclusion of sub-genres may be especially important in a genre such as sport as the further processing required by each sub-genre (i.e. each specific sport) will most likely be quite unique. To include sub-genres, it may be necessary to build a hierarchical classifier that first

classifies a sequence at the genre level, and a separate classifier is built for each genre for appropriate sub-genre classification.