Motion based video content indexing

Yi, Haoran

2007

Yi, H. R. (2007). Motion based video content indexing. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/35741>

<https://doi.org/10.32657/10356/35741>

Motion Based Video Content Indexing

Yi Haoran

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2007

Abstract

This thesis addresses the problem of content-based video indexing by motion. Our research works cover the 3 fundamental and challenging issues in content-based video indexing: video structure analysis, low-level feature index, high-level semantic description.

First, we propose a new rapid shot boundary detection algorithm. The proposed algorithm is able to detect both hard cut and gradual shot boundaries. With the detected shots as building blocks, we develop an adaptive hierarchical tree structure, called ‘Scene Tree’, for video content description. The ‘scene tree’ facilitates content based indexing, browsing and retrieval. We also propose a set of motion features in compressed domain: total motion (TM), camera motion (CM), and object motion (OM) as indices for the video content. The ‘Scene Tree’ together with the proposed $TM/CM/OM$ features provides an integrated approach for parsing, indexing and retrieval of compressed videos.

Secondly, we propose a new representation for video motion content. The motion content of the video is represented as a *Pixel Change Ratio Map (PCRM)*. With the proposed video motion representation, two new motion features for video motion content indexing are proposed: 1) motion histogram and 2) motion correlogram. We demonstrate the usefulness of the motion features with three applications: (i) video retrieval, (ii) video clustering and (iii) video classification. With the $PCRM$ based video motion representation, a new motion based key frame extraction method is developed. The proposed key frame extraction method is computationally efficient and flexible. It can rapidly generate key frame based video summary of any desired length.

We also study another object based motion descriptor standardized by MPEG-7, motion trajectory. We present an automatic trajectory extraction approach for compressed sports videos. The proposed approach is able to extract arbitrary number of motion trajectories in both stationary and moving camera. The trajectories are extracted through

a four stage process of (1) global motion estimation, (2) motion blob detection, (3) trajectory evolution and (4) trajectory refinement. The extracted motion trajectories are output as MPEG-7 motion trajectory descriptors.

Finally, we explore the problem how to build up semantic index for broadcasting videos. We present two approaches to index the video content at semantic level. The first approach is based on the classification scheme using multiple low-level features. We provide a video classification system using *HMM*. We use color and motion features as the observation of the video *HMM*. The *HMM* video classification has been proved to be a viable method for video topic classification. The features extracted from compressed domain are proved to be effective and efficient for the video classification task. The second approach stems from a different theme. We apply modern information retrieval techniques to analyze the subtitles of the video to get the semantics directly for video indexing. The semantic video index is extracted by analyzing the subtitle file in a DVD video and represented by the vector based model. The semantic script vector index is shown to be effective for semantic video retrieval and video summarization.

Acknowledgments

First of all, I would like to express my profound gratitude to my supervisor, Dr. Deepu Rajan for his support since the beginning, and for his patience and guidance. I am especially grateful to Associate Professor Chia Liang-Tien, Clement for his supervision, guidance, friendship, and moral support. I regard myself fortunate to have him as my co-supervisor. Without their continuous encouragement and advice, I could never complete this thesis.

I must also thank Associate Professor Lee Bu-Sung, Associate Professor Chan Syin and Dr. Sourav Saha Bhowmick for their proposal of the Pet-Device project.

I would like to thank all my friends and colleagues in Center of Multimedia and Network Technology(CeMNet), especially Wang Surong, Liu Song, Chen Ling, Chu Yang and Hu Yiqun, with whom I have worked together, for their precious discussions and friendship.

My final thanks reserved to my parents for their love, understanding, encouragement and support during my Ph.D study.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	vii
List of Tables	xi
1 Introduction	1
1.1 Issues in Video Indexing	2
1.2 MPEG-7 Standard	4
1.3 Literature Survey	7
1.3.1 Video structure analysis	7
1.3.2 Feature indexing	10
1.3.3 Semantic indexing	14
1.4 Research Objectives and Main Contributions	16
1.4.1 Main Contributions	17
1.5 Dissertation Organization	18
2 A Scene Tree for Browsing and Retrieval	20
2.1 Shot Boundary Detection	21
2.1.1 Detection of Hard Cuts	23
2.1.2 Detection of Gradual Transitions	26
2.2 Scene Tree	27
2.2.1 Scene Tree Building Algorithm	28
2.2.2 Illustration of a Scene Tree Construction	32
2.2.3 Scene Tree Representation	33
2.3 Motion based Indexing and Retrieval	36

2.3.1	Motion Extraction	37
2.3.2	Video Similarity Measure	42
2.4	Experimental Results	43
2.4.1	Shot Boundary Detection	43
2.4.2	Adaptive Scene Tree Construction	46
2.4.3	Motion Estimation	55
2.4.4	Motion Based Retrieval	56
2.5	Conclusion	59
3	New Motion Features and their Applications	61
3.1	Pixel Change Ratio Map	63
3.2	The Motion Histogram	65
3.3	The Motion Correlogram	66
3.4	Applications of Motion Histogram and Correlogram	68
3.4.1	Video Retrieval	68
3.4.2	Video Classification via Probabilistic Modeling	74
3.4.3	Video Classification via Pattern Recognition Techniques	80
3.5	Key Frame Extraction	81
3.5.1	Pixel Change Map	82
3.5.2	Global Motion Compensation by Adaptive Filter	83
3.5.3	Key Frame Selection	85
3.5.4	Experimental Results	86
3.6	Conclusion	91
4	Automatic Motion Trajectory Extraction	94
4.1	Tracking and Trajectory Extraction	95
4.2	System Overview	96
4.3	Shot Classification	98
4.3.1	Feature Extraction	99
4.3.2	Support Vector Machine	100
4.4	Global Motion Estimation	103
4.5	Trajectory Extraction	104

4.5.1	Motion Blob Detection	106
4.5.2	Trajectory Evolution	108
4.5.3	Trajectory Refinement	111
4.6	XML Document for Motion Trajectory Descriptor	112
4.7	Experimental Results	114
4.7.1	Shot Classification	114
4.7.2	Global Motion Estimation	115
4.7.3	Motion Trajectory Extraction	117
4.8	Conclusion	125
5	Towards Semantic Video Analysis	127
5.1	HMM based Video Classification	128
5.1.1	Hidden Markov Models	129
5.1.2	Feature Extraction	131
5.1.3	Video classification	133
5.1.4	Experimental Results	136
5.2	Semantic Video Indexing Using Subtitles	139
5.2.1	Semantic Video Indexing	140
5.2.2	Applications	143
5.2.3	Experimental Results	146
5.2.4	Discussion	149
5.3	Conclusion	149
6	Conclusion	151
6.1	Summary of Contributions	151
6.2	Future Directions	154
	References	157
	List of Publications	170

List of Figures

1.1	(a) A graphical view of the relation between the different MPEG-7 elements[1]. (b)An abstract representation of possible applications using MPEG-7[1]	5
2.1	Illustration of scene tree construction algorithm.	34
2.2	Example of XML representation for scene tree.	36
2.3	Histogram of magnitudes of residual motion vectors between two frames in an example ‘soccer’ video.	40
2.4	Block Rejection Map.	41
2.5	(a) <i>MFDR</i> for a ‘news’ sequences; (b) Local sum of <i>DMBC</i> in a dissolve.	44
2.6	Comparison of (a) ‘Recall’ and (b) ‘Precision’ rates using the proposed SBD algorithm and Jun’s algorithm [2] respectively for gradual shot change detection.	46
2.7	Scene tree of MPEG-7 test video.	47
2.8	Scene tree of TV program “Opening The New Era”.	49
2.9	Scene tree of MPEG-7 test video (CD20#1).	51
2.10	Scene tree of MPEG-7 test video (CD20#2).	53
2.11	(a) Video annotation tool (VideoBuddy) interface; (b) key frame annota- tion.	54
2.12	Motion estimation. Each row consists of a key frame of the shot, the <i>CM</i> image, the <i>OM</i> image and the <i>TM</i> image of News anchor person (first row), gallery (second row), soccer (third row) and moving person (fourth row)	56
2.13	Motion based video retrieval. (a) Five example query results using <i>TM</i> matrix. (b) The same five query results using <i>CM</i> , <i>OM</i> and <i>TM</i> matrices.	57

2.14	Precision as a function of the number of video clips retrieved. Blue line corresponds to retrievals using the <i>TM</i> , <i>OM</i> and <i>CM</i> ; red line corresponds to retrievals using <i>TM</i> only; pink line corresponds to retrieval using <i>MV+PCA</i> (35D); black line corresponds to retrieval using motion histogram (8D).	59
3.1	(a) Key frame (middle frame) of a 'Football' sequence, (b) <i>Pixel Change Ratio Map</i> of the video sequence in (a)	64
3.2	Histogram of <i>PCRM</i> images accumulated over all the sequences in the database.	65
3.3	(a) Histogram of Figure 3.1(b) before re-quantization, (b) Histogram of Figure 3.1(b) after 8-level nonuniform quantization.	67
3.4	Examples of video retrieval using motion histogram (showing the key frame of each sequence).	69
3.5	Average precision of retrieval for (a)'soccer&basketball' and (b)'talking head' sequences using motion histogram with varying quantization levels and <i>MV+PCA</i> (32D) features.	72
3.6	Average precision of retrieval for (a)'soccer&basketball' and (b)'talking head' sequences using motion histogram for motion histogram(8D), <i>MV+PCA</i> and <i>TM+CM+OM</i> features.	73
3.7	Examples of video retrieval with (a) motion correlogram and (b) motion histogram features (showing the key frame of each sequence).	74
3.8	Comparison of average precision of retrieval for (a) 'soccer&basketball' and (b) 'talking head' videos with different features.	74
3.9	(a) Bayes Information Criterion (<i>BIC</i>) for different number of clusters with motion histogram. (b) Scatter plot of feature vectors reduced from 8D motion histogram to 2D using <i>PCA</i>	77
3.10	Video clustering using 8-level motion histogram (the key frames are shown).	79
3.11	(a) Examples of video sequences (key frames) from two classes - 'high motion' (<i>Top Row</i>) and 'low motion' (<i>Bottom Row</i>), (b) Average classification rate with different number of training samples.	81

3.12 (a) The simulated pixel change sequence. Convergence of weights using (b) <i>LMS</i> algorithm and (c) <i>SLMS</i> algorithm.	85
3.13 (a) Prediction error in pixel change, PCM_e , per frame (synthetic data), (b) Cumulative PCM_e with location of extracted key frames.	87
3.14 Frame from video clip with (a) smooth and (c) irregular camera motion. (b) and (d) original, predicted and prediction errors of pixel changes for video clip of (a) and (c), respectively (Horizontal axis shows the frame number.)	88
3.15 DC images of (a) 4 keyframes (frame #1, #67, #139, #374) and (b) 5 keyframes (frame #1, #45, #96, #192, #374) from an MPEG-7 test sequence.	89
3.16 DC images of key frames extracted by (a) <i>CTK</i> method (frame #37, #112, #211, #256) and (b) the proposed method (frame #106, #214, #241, #277).	90
3.17 Key frame selection result: (a) 'News' video, (b) part of the 'News' video, (c) 'Basketball' video 1, (d) 'Basketball' video 2, (e) 'Soccer' video 1, (f) 'Soccer' video 2, (g) 'TV program' video 1, (h) 'TV program' video 2.	92
4.1 Overview of the proposed trajectory extraction approach	97
4.2 (a) Distribution of co-occurrence matrix based texture features extracted from 150 frames of three sports videos, (b) 2D plot of Energy v/s Contrast of (a) and (c) 2D plot of Entropy v/s Contrast of (a); (d) Distribution of run-length matrix based texture features, (e) 2D plot of Run Length Entropy v/s Long Run Emphasis of (d) and (f) 2D plot of Run Length Entropy and Shot Run Emphasis of (d).	101
4.3 The transformation matrix between frames F_n and F_1 given by $M_1 \times M_2 \times$ $\dots \times M_{n-1}$	103
4.4 Motion vector field and estimated camera motion optical flow field for zooming and panning of tabletennis shot (a) motion vector field for zoom- ing and (b) estimated camera motion field (zooming), (c) motion vector field for panning and (d) estimated camera motion field (panning)	105
4.5 Occlusion illustration	110
4.6 (a) Frame 153 and (b) frame 183 of the soccer sequence	115

4.7	Mosaic of frames 153 and 183 of the soccer sequence using (a) gradient based algorithm and (b) the proposed motion vector based algorithm . . .	116
4.8	Mosaic generated from 300 frames of soccer video	116
4.9	Motion blob detection in sample frames of the badminton sequence. . . .	117
4.10	Trajectories extracted from (a) frames 1 to 123 and (b) frames 1 to 246 and (c) final trajectory extracted after refinement from 369 frames in the badminton sequence.	118
4.11	Motion trajectory descriptors for "Badminton video" in MPEG-7 compliant XML format.	119
4.12	Motion blob detection in sample frames of the soccer sequence.	120
4.13	Trajectories extracted from (a) frames 1 to 100 and (b) frames 1 to 200 and (c) final trajectory extracted after refinement from 300 frames in the soccer sequence.	120
4.14	Motion trajectory descriptors for 'Soccer video' in MPEG-7 compliant XML format.	121
4.15	Motion blob detection in sample frames of the 'tennis' sequence.	122
4.16	Trajectories extracted from (a) frames 1 to 104 and (b) frames 1 to 208 and (c) final trajectory extracted after refinement from 312 frames in the tennis sequence.	122
4.17	Decoded MB ratio for the experiment sequence: 'soccer', 'tennis', 'badminton'.	124
4.18	An example frame and the motion blob detection result for the 'basketball' video.	125
5.1	Framework for pattern classification	128
5.2	A diagram of training and classification of video clips using <i>HMM</i>	136
5.3	The proposed approach to build a semantic index.	139
5.4	Script example file	141
5.5	(a) Time gap between the script element of 'friends' video. (b) Energy ratio VS feature rank (<i>PCA</i>).	147
5.6	Example retrieved results: (a) 'bride' query, (b) 'dance' query, (c) 'groom' query, (d) 'wedding' query	148

List of Tables

2.1	Summary of SBD results	44
3.1	Description of clusters using motion histogram.	78
3.2	Average histogram errors for <i>CIK</i> , <i>CTK</i> and the proposed method. . . .	91
3.3	Test video clips and number of extracted key frames.	93
4.1	Shot classification using co-occurrence and run-length features.	114
4.2	Average decoded MB ratio.	124
5.1	Training and test set	137
5.2	Classification result with HMM (2-4 hidden states) HMM and 3-component GMM	137
5.3	Log likelihood distance measure with different combinations of color spaces and compression methods	137

Chapter 1

Introduction

With the progress in video compression and communication, a large amount of digital video content is available online and ready to be delivered live or on-demand over the Internet. Home users have high bandwidth cable or DSL connections to view TV-quality videos. Storage and transmission technologies have advanced to a stage where they can accommodate the demanding volume of video data within practical constraint. Encoding technologies such as MPEG-1, 2, 4 have enabled access to digital video within the constraints of current communications infrastructure and technology. Even the production of digital video has become available to the masses with the introduction of high performance, low-cost digital capture and recording devices. As a result, a huge volume of digital video content is available in digital archives, on the World Wide Web, in broadcast data streams, and in personal and professional databases [1][3].

The vast amount of the content information requires effective and efficient techniques to find, access, filter and manage video data. Search engines and database management systems (DBMS) exist for text information. However, videos are much more rich in information compared to text documents and they lack a well organized structure. The diversity of content and lack of structure makes management of video data a challenging task. New technologies capable of efficient processing, effective indexing and organization of video contents need to be developed. In traditional relational databases, the data

CHAPTER 1. INTRODUCTION

are organized in the form of tables, attributes etc. The *indices* for each item in the database are the key fields. When a query is posed, the stored data is searched through alphanumeric matching. Each item in the database has several key fields (indices) by which the query is matched. The notion of index with respect to images and videos undergoes a paradigm shift in that an image is indexed using low-level pictorial features (e.g. color, texture, and shape etc.) and a video is indexed using low-level features extracted from individual frames or a collection of frames. Since video can be viewed as a 3-d signal, temporal characteristics are also important for video indexing.

1.1 Issues in Video Indexing

The immediate problem of the video indexing is to facilitate location of the desired content. Retrieval of content from the digital multimedia libraries/databases is the key problem for a wide range of applications such as filtering and selection of broadcast channels, summarization of a long video program with key-frames, detection of interesting events in sports videos, etc. The typical approach for video content description and indexing/retrieval is as follows. First, the video is segmented into temporally consistent units. Then, various features, e.g. color, texture, motion, etc., are extracted from the video units to index the video content. Using the feature index, the videos are retrieved and browsed based on the similarity of the features. These approaches are known as the “content based” techniques. The key component is the feature, which provides quality in the search for similar sequences. Many studies have been done in this area, such as QBIC [4], PhotoBook [5], VisualSeek [6], MARS [7] and VideoQ [8]. The research objectives are to develop features with better search capability and to automate the conventional manual annotation for video content.

In addition to features, another area of research in video content index aims at parsing and discovering structure of the video data [9, 10]. As mentioned before, video lacks

CHAPTER 1. INTRODUCTION

structure while traditional relational database and text documents are structured or semi-structured. Video data can be indexed at different spatio-temporal granularity, i.e. region level, frame level, shot level, story unit level etc. The research objective here is to build up a structural description of the video content which will facilitate easy browsing and retrieval. Ideally, a non linear browsing capability is desirable as opposed to standard techniques like fast forward or fast reverse. This can be achieved, preferably, using a structure that represents video information as a hierarchy of various semantic levels. Such a multi-layer abstraction makes it not only more convenient to reference video information but also simplifies video indexing and storage organization.

Another important problem associated with video indexing is the assignment of semantic indices to video content. Human beings use high-level concepts in everyday life and tend to describe the video content using semantics. One way to obtain semantic information for video content is through manual annotation. However, it is unrealistic for large video database. It is desirable to develop approaches to extract semantic information of the video content automatically or semi-automatically. This is a challenging problem because there is a huge gap between low-level features and high-level semantics. New technologies and methods are needed to bridge this gap.

Besides, an important difference between videos and images is that video are dynamic data with temporal dimension. Videos are presented continuously at certain rates. A TV-quality video has 25 to 30 frames per second. On one hand, the continuous characteristic results in a large amount of information and make it further challenging to process and manage videos. On the other hand, there are more information to be exploited, particularly motion information. Motion features capture the essential characteristics of the video content and provide an opportunity for better video content indexing. Automatic extraction of motion features for video indexing has been a vital research topic.

In summary, we can identify three fundamental problems and challenging issues in video content indexing: (i) video structure analysis, (ii) feature based video content analysis, and (iii) semantic video analysis.

1.2 MPEG-7 Standard

Since its advent, video indexing has attracted many researchers from various research communities including computer vision, image/video processing, library and information science and database management systems, etc. A lot of research effort has been put into this area ranging from government, industry, to universities. The International Standard Organization (ISO/IEC) has put out the MPEG-7 standard aimed for multimedia content indexing and description. MPEG-7 provides a comprehensive and standardized description of multimedia content which covers various aspects of the video content and supports a wide range of applications.

The MPEG-7 standard, formally called “Multimedia Content Description Interface”, provides a set of “Descriptors” and “Description Schemes”, along with a “Description Definition Language”, to describe multimedia content [11]. The scope of the standard covers various forms of media including audio, video and images, both in digital and analog format, although most applications involve only digital content. MPEG-7 allows a standard description scheme of various aspects of multimedia material such as low-level feature index, video structure description, high-level semantic index etc. MPEG-7 also enables many applications aimed at the end user (e.g. a VCR that records only preferred content) or automatic systems (e.g. a surveillance system).

MPEG-7 defines a set of visual descriptors that represent various “features” of media content. For example color is a feature of a piece of visual content. MPEG-7 provides a number of descriptors to represent the color content, e.g. the Dominant Color Descriptor (DCD), the Color Layout Descriptor (CLD), etc. of a given piece of visual content. Other

CHAPTER 1. INTRODUCTION

Figure 1.1: (a) A graphical view of the relation between the different MPEG-7 elements[1]. (b)An abstract representation of possible applications using MPEG-7[1]

CHAPTER 1. INTRODUCTION

descriptors for the shape, texture, and motion features of visual content are provided as well. Essentially, the visual descriptors provide the low-level feature based index of the multimedia content.

Besides visual descriptors, MPEG-7 also provides predefined structures called “Description Schemes (DS)”. They are made up of various descriptors. For example, the video segment DS shown in Figure 1.1(a) includes various visual descriptors to describe the video segment. The Multimedia Description Scheme (MDS) can be used as a tool for description of the video structure and organize the video content. Figure 1.1(b) shows an abstract representation of an MPEG-7 application. MPEG-7 also defines a set of ‘Descriptors’ and ‘Description Schemes’ for semantic description of the video content. The ‘Content Semantics Description Scheme’ is a tool provided in MPEG-7 for describing the semantics of multimedia content. The semantic DS is capable of describing the entities, events, concepts, time, place and all other semantic information for the multimedia data.

The above description shows that MPEG-7 provides tools for low-level feature index, video structure description and semantic description. However, an important aspect of MPEG-7 is that it standardizes content description but does not specify how the description is produced. Some of the descriptors in MPEG-7 are automatically extracted from the content, such as color histograms or motion descriptors. However, most of the high level descriptors and semantics are very hard to generate automatically, such as the description of the theme of a movie, or segmentation of a drama video into semantically meaningful scenes. Extraction of the semantic information automatically or semi-automatically in a most accurate and effective way still remains a great challenge and is an open research area that leaves room for future improvements in multimedia content analysis. MPEG-7 does not specify how the content description will be used as well. Again, this leaves room for improvements in intelligent use of available information to arrive at the higher level understanding and better utilization of the content.

In conclusion, MPEG-7 provides a standard interface for the exchange of information that describe multimedia content. Extraction and consumption of this information in the most accurate, efficient and effective way is open for research. MPEG-7 provides a common framework and a widely accepted platform that facilitates research and improvement on multimedia content analysis and description.

1.3 Literature Survey

In the following, literature related to video indexing methods is classified into the three fundamental tasks that we address in this thesis, i.e., video structure analysis, feature indexing and semantic indexing.

1.3.1 Video structure analysis

Many research efforts have been put on the video structure analysis, which is a fundamental task in video indexing. It is the first step towards automatic analysis of video content. A fundamental step in video structure analysis is shot boundary detection. It segments video data into the basic unit, ‘shot’. A shot in a video sequence refers to a contiguous recording of one or more video frames depicting a continuous action in time and space. In a video database, the shot is considered to be the basic unit in video indexing and is appropriate for video browsing and content based video retrieval [12]. In general, there are 2 types of shot changes: 1) abrupt change or hard cut, and 2) gradual change due to the various video editing effect, such as fade in, fade out, dissolve etc. Abrupt changes (hard cuts) are simple shot changes. They occur in a single frame when stoping and restarting the camera. Although many kinds of cinematic effects could be applied to artificially combine two shots to create gradual shot transitions, most often ‘fades’ and ‘dissolves’ are used. A ‘fade out’ is a slow decrease in brightness resulting in a black frame; a ‘fade in’ is a gradually increase in intensity starting from a black image.

CHAPTER 1. INTRODUCTION

‘Dissolve’ shows one image superimposed on the other as the frames of the first shot get dimmer and those of the second one get brighter.

There is a rich literature of algorithms for detecting video shot boundaries [13][14][15]. In the initial phase of research in SBD, the data mostly consisted of uncompressed video[9][13]. The SBD algorithms for those data can be broadly classified into methods that use one or more of the following features extracted from the video frames: (i) pixel differences [9], (ii) statistical differences [16], (iii) histogram [17] and (iv) edge tracking [18][19]. With the emergence of the MPEG compression standard, most of the videos are stored using the MPEG compression standard now. However, the approaches above are working in the spatial domain and need to fully decode the video. It is very time consuming to fully decode the video sequences. Many algorithms have been proposed to parse a video sequence directly from the compressed data [20] [21] [22]. Most of these methods use either the DC information [21] or the bit-rate of different MPEG picture types [22] in their algorithms. Feng et al. [20] used bit rate information to detect shot boundary. The shot boundary is identified where the bit rate increase dramatically. Yeo et al. [21] used the color histogram extracted from the DC image for shot boundary detection. Meng et al. [22] extracted the macroblock type information from the MPEG compressed video and used the percentage of the forward predicted macroblocks for shot boundary detection.

Although the shot is the basic unit in the video, it does not convey semantic meaning. The shot boundary is the physical boundary of the video. It supports only linear browsing of the video content. Besides, the number of shots may vary from several hundreds to thousands in a typical full length movie. Ideally, a non linear browsing capability is desirable as opposed to standard techniques like fast forward or fast reverse. This can be achieved, preferably, using a structure that represents video information as a hierarchy of various semantic levels. Such a multi-layer abstraction makes it not only

CHAPTER 1. INTRODUCTION

more convenient to reference video information but also simplifies video indexing and storage organization. Several multilevel structures have been proposed in the literature. The theme behind those algorithms is to sequentially group similar shots into semantic ‘scenes’. The ‘scene’ provides semantic structure of the video. Yeung et al. [23] proposed a graphical representation of videos by constructing a scene transition graph (*STG*). Each node in *STG* represents a shot, and the edges represented the transitions between the shots based on the visual similarity and temporal locality. The *STG* was then split into several subgraphs using the complete-link method of hierarchical clustering. Each subgraph satisfied a similarity constraint based on the color and represented a scene. Hanjalic et al. [24] used a similar approach for shot clustering using a graph and found logical story units. The linking of shots was done by defining an inter-shot dissimilarity measure among the shots. Rui et al. [25] proposed the construction of table-of-contents for videos. A time-adaptive grouping of shots was done by finding the visual similarity between them. This similarity was a function of color and activity features of the shots, weighted by their temporal locality. Shots were merged together to form groups by defining a group threshold and groups were merged together to form scenes by defining a scene threshold. Zhou et al. [26] exploited film editing techniques and discovered that certain regions in the video frames are more robust to noise for computing shot similarities. The clustering method is, however, similar to [24]. Adams et al. [27] proposed to parse the videos by computing the tempo in feature movies. Their approach was inspired by the existing cinematic conventions known as film grammar. Camera motion parameters and shot lengths in the video were used to construct a tempo plot. The proposed method detects edges in the tempo function and identifies instances where the tempo of the movie changed with time.

1.3.2 Feature indexing

Videos are visual representation of information. Many methods have been developed for indexing of the video content based on their low-level features. Low-level features provide a compact, numerical representations of the physical properties of the audiovisual data. They greatly reduce the amount of data to be analyzed, provide mathematical representation for comparison, and serve as the foundation for indexing, analysis, high-level understanding, and classification. As shown in Figure 1.1(a), most visual descriptors in MPEG-7 come from level features. Feature extraction methods have been proposed in both compressed and uncompressed domain. We focus our review on the features extracted from the *compressed* domain, as presented in the following chapters. Most of the multimedia content available today is in compressed format. Using features that can be extracted from compressed video directly avoids the need to decode video. This makes efficient and real-time video indexing and analysis possible. The other reason for extracting features from the compressed domain is the availability of information generated during the encoding process. A typical example is the motion vector. Of course, most features can be obtained from uncompressed data as well. Generally, those features computed from the uncompressed domain are of a higher precision but require a much higher computational cost. But the high accuracy achieved in uncompressed domain may not be necessary for video indexing. A video index requires a compact representation of the characteristics of the video and should be quite general. In most cases, those features should be scaled down, e.g. high resolution histogram might be scaled down by merging adjacent bins. Thus it may be a waste of computation resources to process uncompressed videos.

Visual feature extraction

Visual features can be extracted from MPEG compressed video. These features can be used to characterize individual video frames, as well as JPEG-compressed images.

CHAPTER 1. INTRODUCTION

They can also be applied to a group of video frames to characterize their aggregated visual property. An important video representation in compressed domain is through DC images. DC image sequence is an iconic version of the original video. Although it is of a lower resolution, it captures the key contents and is very efficient for visual feature extraction. All *DCT* coefficients, including the DC values, are readily available from *I* frames in MPEG video. For *P* and *B* frames, the residual error resulting from motion compensation is coded with *DCT*. To get the *DCT* coefficients in motion-compensated frames, Chang et al. [28] develop a method based on the fact that *DCT* is a linear transformation. The algorithm is very efficient if only part of the *DCT* coefficients (e.g. the DC coefficient) is needed. Yeo et al. [29], as well as Shen et al. [30], also propose efficient algorithms to approximate DC values for *P*- and *B*-frames. Once the DC image sequence is constructed, features can be extracted. With the DC images, many visual features have been proposed for video indexing. Color histogram and other statistical measures of color extracted from the DC sequence for digital video indexing are proposed in [31, 32, 33]. Coefficients in the 8×8 *DCT* block can be classified into frequency bands that roughly correspond to smooth area, horizontal and vertical edges, and noisy areas [34]. Bao et al. [35] use energy histogram of low frequency coefficients as the matching features for video frame similarity. This feature can be seen as a rough description of the global texture pattern in the video frame. Zhong et al. [36] compute sums of amplitude of AC coefficients in the first row and first column of the *DCT* coefficients matrix and use them as the criteria for initial segmentation of caption blocks in compressed video. This works because rapid changes in intensity (in both direction) introduced by character lines raise the energy level in the corresponding *DCT* frequency bands. In other words, caption blocks have a distinctive texture pattern that is reflected in the *DCT* coefficients. There also have been some efforts on edge extraction from compressed video. Shen et al. [37] show that AC energy of *DCT* blocks can be used to detect areas of high activity.

CHAPTER 1. INTRODUCTION

Observing that AC coefficients of the *DCT* are linear combination of pixel intensity values in the 8×8 block, they use the ideal step-edge model and derive approximate edge orientation, offset and strength parameters using only *DCT* AC coefficients. The result is coarse edge segments that can be used for video indexing task. They also investigate convolution-based edge detection in compressed images by merging symmetric convolution kernel for edge detection with the 2D *IDCT* procedures[38]. Compared with conventional convolutional-based edge detection, a speedup of 3 to 10 times is achieved with comparable result. However, large convolution kernel is necessary to reduce the artifacts introduced by JPEG or MPEG compression scheme. Therefore, this method becomes less efficient and can be applied to only a small subset of MPEG video frames due to the computational complexity. Song et al. [39] use edge block energy to classify *DCT* blocks into edge blocks and non-edge blocks. Using the 3×3 Sobel operators, they show that the energy of the resulted gradient block can be approximated using the first few AC coefficients. The result is a rough block edge map that is $1/64^{th}$ the size of the original video frame, which indicates high visual activity areas. Arman et al. [40] use a subset of *DCT* coefficients of a subset of blocks to form feature vectors for each frame. The angle between the feature vectors of different frames is used to measure their similarity.

Motion feature extraction

In addition to visual features, motion features have been proposed for video indexing. In compressed video, motion information is captured in the form of motion vectors in MPEG videos. However, motion vectors are only a rough and sparse approximation to real optic flow fields and are prone to be inaccurate when used to indicate real motion of *MBs*. In MPEG video frames, the following areas are most likely to have erroneous motion vectors: (i) boundary blocks and (ii) large smooth areas. Usually some kind of morphological or

CHAPTER 1. INTRODUCTION

median filter should be applied to the motion vector field to remove anomalies before they are used for indexing and analysis. Also motion vectors are more sensitive to noise in magnitude than in direction. Therefore, a median filtering based on magnitude is usually sufficient. The authors in [41] use global motion field to model the global motion in video sequences for similarity query. The idea is to divide a frame into 4 to 16 quadrants with each quadrant a motion feature in terms of magnitude and direction. For magnitude, an average value is used and the query can be specified by high or low magnitude. For direction feature, either average or dominant motion vector direction (such as the largest bin in angle histogram), as well as an angle histogram (90 directions, 4 degree each) is computed. Dimitrova et al. [42] use the MPEG motion vector in P and B frames to approximate the movement of macroblocks. Macroblock tracing is used to retrieve low-level motion information. In the middle-level motion analysis, averaging or clustering method is used to extract object motion for rigid and non-rigid objects, respectively. At high-level analysis, a set of trajectories is associated with an activity defined using domain knowledge. There are several representations of trajectories for various matching purpose: exact motion trajectory (start position and trajectory coordinates), B-spline curves, chain code (for approximate matching), and differential chain code (for qualitative matching). Wavelet transform can be used to generate a multi-resolution representation of trajectories. Camera motion is also an important feature in video indexing. Foreground moving objects can be extracted by distinguishing camera motion and object motion. Furthermore, in some specific applications, models of camera motion can be used to detect important events, e.g. fast breaks in basketball games. Camera operation usually causes a global and dominant motion flow in the video sequences. However, when dominant motion of a large object is present concurrently, it changes the global motion field and camera operation estimation becomes less reliable. One solution is to remove the outliers of motion vectors that do not fit well with the global motion model and iterate the

CHAPTER 1. INTRODUCTION

estimation process of camera motion. Divakaran et al. [43] propose the amount of motion descriptor for video indexing. The average motion vector magnitude is used as one of the motion features. They use this average magnitude to threshold all the MV magnitudes in the frame and count the number of short, medium and long runs of zeros in raster scan order. These numbers indirectly expresses the size, shape, and number of moving object(s) and hence are good indicators of the motion content of the video frame. For example, a frame with a single large object can be easily distinguished from a frame with several small moving objects. Wolf et al. [44] uses the amount of motion in video frames to select key frames in a video sequence. Although in [44] motion is estimated using optical flow, the idea can be applied to MPEG video using motion vectors. The amount of motion is defined as the summation of magnitude of motion vectors in a frame, and a curve of motion amount is drawn with respect to frame numbers. Key frames are then selected by detecting valleys between large peaks in this curve, which correspond to relatively still frames between two video segments with large amount of motion.

1.3.3 Semantic indexing

There are also a lot of research works on the extraction of semantics from video content. The work in [45] is an example of high level semantic indexing. This approach uses a set of predefined index terms for annotating video. The index terms are organized based on high level ontological categories like action, time, space, etc. The high level indexing techniques are primarily designed from the perspective of manual indexing or annotation. This approach is suitable for small quantities of news video and for accessing previously annotated databases.

To deal with large databases, automatic or at least semiautomatic methods are desired. One popular type of methods to extract semantic information are to combine human annotation with machine learning [46]. Such methods are semiautomatic and

CHAPTER 1. INTRODUCTION

complex because the initial training set need to be labelled by human and the learned classifiers may also need to be tuned for different videos. However, video classification still has been used widely as tool for extracting semantics. In [47], the authors use nearest neighbor clustering to do video classification. This classification method is simple, but it does not take the temporal features into consideration. The authors in [48] analyze the face and text trajectories and then use *HMM* to classify video into predefined types. The chromaticity of keyframes in video summarization is used for video classification in [49]. An *HMM* framework using audio and visual features is built for video classification in [50]. The use of both audio and visual features can improve classification accuracy, but the system becomes complicated and computational intensive.

Several domain specific semantic indexing technologies have been developed in literature as well. By narrowing the input videos from wide domain to specific domain, the gap between low-level features and high level semantics is reduced. Domain constraint and information helps to extract the semantics. The context information in the domain limits the possible semantics and makes it is easier to detect them. For example, if you detect a lot of dust and red colors in a scene during the car racing game, there is a high probability that an accident is taking place. In real-life, many domain specific videos are available. Medical videos are recorded and used to help diagnosis of illness; surveillance video are recorded and used for security monitoring. The ‘illness’ and ‘security’ information all belongs to a higher level semantic index. Another interesting domain is sports videos. Many sports video analysis and semantic event detection methods have been developed in literature. Ekin et al. [51] use low-level soccer video processing algorithms and object detection algorithms for automatic, real-time soccer video analysis and summarization. Lu et al. [52] use color and motion features for dominant scene clustering and event detection. In [53], the author presents a method that tries to detect the complete set of semantic events, which may happen in a soccer game. This method uses the position

information of the player and of the ball during the game as input, and therefore needs quite a complex and accurate tracking system to obtain this information. In [54] and [55] the correlation between low-level descriptors and the semantic events in a soccer game have been studied. In particular, in [54], it is shown that the low-level descriptors are not sufficient, individually, to obtain satisfactory results. In [55] the temporal evolution of the low-level descriptors in correspondence with semantic events has been exploited, by proposing an algorithm based on a finite-state machine. This algorithm gives good results in terms of accuracy in the detection of the relevant events, whereas the number of false detections remains still quite large. Generally, the methods for sports video analysis combine both domain rules and low-level visual and audio features to infer semantics.

1.4 Research Objectives and Main Contributions

The objectives of this research are aligned towards the three major issues related to video indexing as mentioned in section 1.1. Since video is a continuous media, its temporal characteristics should be explored for content indexing.

A content-based video indexing system is hard to build because of video's length and unstructured format. The challenge is to build a hierarchical tree structure description of video content that not only conveys meaningful semantics of the scene, but also achieves scalability in content. Such a structure should facilitate content based indexing, browsing and retrieval. We have developed a new algorithm to build a browsing hierarchy, *adaptive scene tree*, for video content description. Our scene tree algorithm works on the compressed video directly and is very efficient. Furthermore, the proposed scene tree description is represented as an XML document which is compliant to the MDS (Multimedia Description Schema) of MPEG-7 standard.

The critical role of features in a video indexing system is exemplified by the MPEG-7 standard which provides a comprehensive and standardized descriptors of multimedia

CHAPTER 1. INTRODUCTION

content. The utility of motion features proposed in this research is discussed. In particular, we focus on evaluating motion features from both compressed and uncompressed domain. We also present an automatic approach to extract motion trajectory descriptors from compressed sport videos.

Automatic extraction of high-level semantic information from video is a challenging task. In general, the low-level features do not map directly onto high level concepts. However, a successful mapping would then enable the high level semantics itself to be used as an index. We present two approaches to index the video content at a semantic level. The first approach is based on the classification scheme using color and motion features as the observations in a hidden Markov model(*HMM*) framework. The features extracted from compressed domain is proven to be effective and efficient for the video classification task. The second applies modern information retrieval techniques to analyze the script of the video to obtain the semantics.

1.4.1 Main Contributions

The main contributions of the thesis are as follows:

- An integrated approach for parsing, indexing and retrieval of compressed video. We develop a rapid shot boundary detection algorithm in compressed domain. With the detected shots as building blocks, a novel adaptive hierarchy structure, called the ‘scene tree’ is proposed for *compressed* video content organization. A set of novel motion features are proposed as indices for the video content and integrated into the scene tree.
- A novel and efficient representation of the video motion content using Pixel Change Ratio Map (*PCRM*). With the proposed *PCRM*, effective motion features of motion histogram and correlogram are proposed. The efficacy of the proposed motion

features are demonstrated through applications in video retrieval, clustering and classification. We also propose a new video abstraction method based on *PCRM*. The proposed method is computationally efficient and adaptive. It can rapidly generate key frame based video summary of any desired length.

- A new framework for automatic motion trajectory extraction. The proposed framework consists of four stages: (1) global motion estimation, (2) motion blob detection, (3) trajectory evolution and (4) trajectory refinement. The method is capable of extracting an arbitrary number of trajectories for multiple objects. Moreover, the method is applied directly on the MPEG compressed video without fully decoding of each frame. The extracted motion trajectories are output as MPEG-7 descriptors in an XML document.
- Two novel approaches for video indexing at semantic level. The first approach is a video classification system based on *HMM* using multiple visual cues in the compressed domain. The *HMM* video classification has been proven to be a viable method for video topic classification. The feature extraction method in compressed domain is proven to be effective and efficient. The second approach is based on mining the scripts. The proposed semantic video index by script mining is shown to be effective for semantic video retrieval and video summarization.

1.5 Dissertation Organization

To clearly present the content of the research, the thesis is organized as follows. Chapter 2 deals with video structure analysis. We present our shot boundary detection algorithm, scene tree construction algorithm and motion features for video indexing. In chapter 3, we describe a new video motion representation, called Pixel Change Ratio Map (*PCRM*) and extract motion histogram and motion correlogram features for video indexing. In

CHAPTER 1. INTRODUCTION

chapter 4, we study the MPEG-7 motion trajectory descriptor and present a framework for automatic motion trajectory extraction in compressed sports video. Then, in chapter 5, we explore the topic of video indexing at semantic level. We present two approaches in different themes: 1) semantic video category extraction by *HMM* video classification scheme with multiple cues; 2) semantic video indexing by mining the script. Finally, we provide conclusions and suggest some future research directions in chapter 6.

Chapter 2

A Scene Tree for Browsing and Retrieval

There are three tasks that are fundamental to build an efficient video management system:

(i) The entire video sequence must be segmented into shots, where a shot is defined as a collection of frames recorded from a single camera operation. This is akin to a tuple which is the basic structural element for retrieval in conventional text based database management system. (ii) Even though a shot determines a physical boundary in a video sequence, it does not convey any meaningful semantics within it. Hence, shots that are related to each other must be grouped together into a scene [56, 57, 58]. (iii) Finally, a robust retrieval method depends on a model that captures similarity in the semantics of video content.

We address the above fundamental tasks to provide an integrated approach for managing *compressed* video content. Since video is mostly available in compressed form, there is a need to develop algorithms to process compressed video directly without paying for overheads to decode them before processing. Tasks such as restoration, resolution enhancement, and tracking in compressed videos have been reported recently [59, 60, 61, 62]. Our objective is to develop a fully automatic technique for content based organization and management of MPEG compressed videos. To this end, the chapter

- (i) describes a novel shot boundary detection algorithm that is capable of detecting both abrupt and gradual shot changes like dissolve, fade-in/fade-out;
- (ii) describes a scene tree that acts as an efficient structure to facilitate browsing;
- (iii) presents a video similarity measure that enables efficient indexing of video content based on motion and which is shown to be useful in video retrieval.

As noted earlier, these three tasks provide an integrated approach to browsing and retrieval in large *compressed* video databases.

2.1 Shot Boundary Detection

Shot boundary detection (SBD) is the first fundamental step in video content analysis. It segments video data into the basic unit, shot, for indexing and retrieval. A shot in a video sequence refers to a contiguous recording of one or more video frames depicting a continuous action in time and space. In a video database, the isolation of shots is of interest because the shot level organization of video sequences is considered to be the basic unit in video indexing and is appropriate for video browsing and content based video retrieval [12]. In general, there are 2 types of shot changes: 1) abrupt change or hard cut, and 2) gradual change due to the various video editing effect, such as fade in, fade out, dissolve and wipe. There is a rich literature of algorithms for detecting video shot boundaries [13][14][15]. In the initial phase of research in SBD, the data mostly consisted of uncompressed video [9][13]. They can be broadly classified into methods that use one or more of the following features extracted from the video frames: (i) pixel differences [9], (ii) statistical differences [16], (iii) intensity histogram [17] or principal color histogram [63], and (iv) edge tracking [18][19]. With the emergence of the MPEG compression standard, most of the videos are stored using the MPEG compression standard now. However, the approaches above are working in the spatial domain and need

to fully decode the video. It is very time consuming to fully decode the video sequences. Therefore, it is prudent to develop SBD algorithms in the compressed domain. Some algorithms have been proposed to parse a video sequences directly from the compressed data [20][21][22]. Most of these methods use either the DC information [21] or the bit-rate of different MPEG picture types [22] in their algorithms; the methods using DC coefficients, which requires Huffman decoding of the variable length encoded DCT coefficients, involve significant decoding of the MPEG-compressed video, while the methods using bit-rate do not yield satisfactory detection. Moreover, these methods are incapable of detecting gradual transitions. Recently, many researchers also address the problem of gradual transition detection [64]. Twin comparison method [9] is the first attempt to detect gradual shot transitions. In this approach, two thresholds, one lower and one higher, are applied on the intensity histogram to detect gradual shot changes. This twin comparison method can partially detect the gradual shot transitions and is sensitive to camera and object motion. Another well-known method [22] relies on the intensity variance of successive frames for the detection of gradual shot changes. The method utilizes DC images for the detection. The algorithm is efficient but the accuracy is not high. Some analytical models for gradual shot detection are also suggested in [65], [66] and [67]. These techniques rely on the fact that the special effect, i.e. dissolve, is produced by linearly combining the starting frame of the new shot with the ending frame of the previous shot. These approaches require significant computational cost. In [68], the authors proposed to detect gradual shot boundary using Hidden Markov Model. This approach outperforms the simple thresholding approaches. However, one major drawback of this approach is that it requires labeling of the training set. In [69], the authors proposed a rule based refinement postprocessing scheme to detect gradual transitions in digital videos. Different rules are applied to filter out false positives. In [2], the authors proposed to detect gradual shot boundary using the Macro-block type information. In

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

this approach, the detection of gradual shot changes is based on the spatial-temporal distribution of the Macro-block types. Only a few of these algorithms can be used as a unified approach to detect all types of shot changes. Most of these methods are incapable of handling both hard cuts and gradual transitions (i.e. fade, dissolve, etc.) simultaneously. The hard cuts and various gradual transitions are often treated separately because of the different characteristics of hard cut and gradual transitions. It is difficult to find a uniform representation for both hard cut and different types of gradual shot changes. Therefore, different representation and similarity measures are used to detect different types of shot boundaries, respectively. Lastly, most of them involve tight thresholds that are largely dependent on the video sequence. In this section, we describe a method of detecting not only hard cuts but also gradual transitions. Our algorithm requires only partially decoding of the video stream. Without the need to perform inverse *DCT*, our algorithm works fast. We exploit the number of macroblock (MB) types in each of the *I*-, *P*-, and *B*-frames to derive the SBD algorithm.

2.1.1 Detection of Hard Cuts

We note that not all the macroblocks (MBs) in a *P*-frame are forward motion predicted and not all the macroblocks in a *B*-frame are bidirectionally predicted[70]. Each MB in the *P*-frame is either forward predicted, skipped or intracoded. Similarly, each MB in the *B*-frame is either forward predicted, backward predicted, bi-directionally predicted, skipped or intracoded. However, a skipped MB is encoded in the same way as the first MB in the slice to which it belongs[71]. Hence, we are left with basically four types of MBs which we abbreviate as *In*, *Fw*, *Bk* and *Bi* for intracoded, forward predicted, backward predicted and bidirectionally predicted, respectively.

The number of each type of MBs in a frame is an indication of the similarity/dissimilarity of that frame with its neighboring frames. Since only the *B*-frame has all the four types of

MBs, we use it to compare the dissimilarity through a Frame Dissimilarity Ratio (FDR), which is defined as

$$FDR_n = \begin{cases} \frac{Fw_{n-1}}{Bi_{n-1}} & \text{for Reference frame} \\ \max\left(\frac{Fw_n}{Bi_n}, \frac{Bk_n}{Bi_n}\right) & \text{for B frame} \end{cases} \quad (2.1)$$

where the Reference frame refers to either an I-frame or a P-frame and the index n denotes the frame number. If a shot change takes place at a reference frame, most of the MBs in the previous frame (which has to be a B-frame) are predicted from the previous reference frame. In other words, Fw in the previous frame will be high resulting in a high FDR . On the other hand, if the shot change takes place in a B-frame, all the frames lying between the previous and the following reference frames are either forward predicted or backward predicted. If the number of bi-directionally predicted MBs is small, it once again results in a high FDR for these B-frames. The advantage of using only the B-frames for comparison is that we can avoid the need for normalizing the FDR as well as choosing a different threshold on the FDR for each type of frame. However, we observe that if a shot change takes place at a B-frame, all the B-frames lying between the previous and the next reference frames will have high FDR s. Consider the following frame structure in an MPEG bit stream :

$$\dots I_1 B_2 B_3 B_4 P_5 B_6 B_7 B_8 P_9 \dots$$

If the shot change takes place at B_3 , FDR s for B_2 , B_3 and B_4 will be very high. In order to determine the exact location of the shot boundary, we observe that B_2 is mostly forward predicted while B_3 and B_4 are mostly backward predicted. Thus, at the shot boundary there is a change in the dominant MB type of the B-frame. So, we define a dominant MB change ($DMBC$) for frame n as

$$DMBC_n = \begin{cases} 1 & \\ 0, & \text{if } (Bk_n - Fw_n)(Bk_{n-1} - Fw_{n-1}) > 0 \\ 1, & \text{if } (Bk_n - Fw_n)(Bk_{n-1} - Fw_{n-1}) \leq 0 \end{cases}$$

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

where the first line in the above equation applies to the reference frames and the other two lines apply to the B-frames. The modified FDR ($MFDR$) for frame n is then defined as

$$MFDR_n = FDR_n \times DMBC_n \quad (2.2)$$

While the $MFDR$ is the same as the FDR for a reference frame, there will be one or two consecutive high $MFDR$ s for a shot change in a B-frame. If there is only one high $MFDR$, the corresponding frame indicates the beginning of a new shot. If there are two consecutive high $MFDR$ s, the former indicates the end of a shot and the latter indicates the beginning of a new shot. In the example frame structure considered earlier, the $MFDR$ is retained for frames B_2 and B_3 , while it drops to zero for frame B_4 .

An adaptive threshold mechanism based on a sliding window as proposed in [72] is used to detect hard cuts from the $MFDR$. The $MFDR$ values are compared over a window whose length l is set to be less than the duration of a shot, typically 10 frames. The presence of an abrupt change is detected at each window position, in the middle of the window, using the following adaptive threshold:

$$\begin{aligned} T &= \frac{\max(MFDR_n, \dots, MFDR_{n+l})}{\max(T_g, \text{submax}(MFDR_n, \dots, MFDR_{n+l}))} \\ &\geq \alpha \end{aligned} \quad (2.3)$$

Here, $\max$ and submax refer to the largest and the third largest values, respectively, of their arguments, the parameter α can be thought of as a shape parameter of the boundary pattern characterized by an isolated sharp peak in a series of discontinuity values [24] and T_g is a global threshold which prevents the detected $MFDR$ to be very small. At first glance, it would appear that the presence of α and T_g imposes a tight constraint diluting the adaptive nature of the thresholding process. It is interesting to note, though, that an accurate choice of α and T_g is not crucial to the success of the algorithm. We have

observed that an empirical choice of $\alpha = 5$ and $T_g = 1$ works well with a wide variety of video sequences.

2.1.2 Detection of Gradual Transitions

We consider two types of gradual transitions, viz., fade and dissolve. A fade is characterized by a gradual darkening of a frame and its replacement by another image which either fades in or appears abruptly [73]. A fade-in occurs when the image gradually appears from a blank screen and a fade-out occurs when the image disappears gradually, leaving a blank screen. A dissolve occurs when one image fades away while another image appears. It can be viewed upon as a linear transition from the ending frame in one shot to the starting frame in the next shot with pictures from both the shots occupying the region of transition. From this perspective, fade-in and fade-out can be considered as special instances of a dissolve in which case, the former is a dissolve that begins with a blank frame and the latter is a dissolve that ends in a blank frame.

We use the *DMBC* defined in equation (2.2) to detect dissolves. Consider the following frame structure from an MPEG video: $\dots I_1 B_2 B_3 P_4 B_5 B_6 P_7 \dots$. If a dissolve takes place in this sequence, the dominant MB type in frame B_2 will be *Fw* since B_2 is nearer to the reference frame I_1 , and that in frame B_3 will be *Bk* because it is nearer to P_4 . Thus, we observe that during a dissolve the *DMBC* changes rapidly during a dissolve and consequently a high local sum of the *DMBC* indicates the presence of a dissolve. If w is the length of a window centered around frame n , then the local sum of the *DMBC* for that frame is given by

$$F_n = \sum_{n-w}^{n+w} DMBC_k \quad (2.4)$$

If F_n is greater than a threshold τ , then frame n belongs to the set of frames that makes up the dissolve. It is heuristically found that the value of τ can be chosen to be 0.8 times the width of the window for *several* kinds of video sequences (news, movie, cartoon,

sports). This choice is a tradeoff between the *precision* (the ratio of the number of shot changes detected correctly over the total number of shot changes detected) and *recall* (the ratio of the number of shot changes detected correctly over the actual number of shot changes). If we lower the threshold, we are able to increase the recall but the precision decreases. On the other hand, if we increase the threshold, the precision increases but the recall decreases. The width of the window is taken to be slightly less than the width of a dissolve, which is typically 10 frames. We choose our threshold to be about 80% percent of window length. If the dissolve lasts longer than the window, the algorithm is still capable of detecting the high plateau as shown in Figure 2.5 (section 2.4). If the dissolve lasts only a few frames, for example 3 to 4 frames, it would be missed by equation (2.4) because it is too short to meet the threshold. But it might result a high *MFDR* in equation (2.2) due to the relative large amount of changes between frames, which would then be potentially detected by equation (2.3). That is why we do not set the window size to be too narrow, which would cause many false detections. Thus, we note that the presence of the threshold τ does not render the algorithm inflexible over different types of sequences, as we show in our experiments. A further refinement of the detected gradual transitions can be achieved by merging those transitions whose temporal distance are less than the width of a typical dissolve.

2.2 Scene Tree

The objective of content based indexing of videos is to facilitate easy browsing and retrieval. Ideally, a non linear browsing capability is desirable as opposed to standard techniques like fast forward or fast reverse. This can be achieved, preferably, using a structure that represents video information as a hierarchy of various semantic levels. Such a multi-layer abstraction makes it not only more convenient to reference video information but also simplifies video indexing and storage organization. Several multilevel structures

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

have been proposed in the literature [12][74][57][75][58]. However, all of them use fixed structures, like shots and scenes, to describe the video content. The underlying theme is to group together frames that are ‘similar’ to each other where similarity could be defined in primitive terms such as temporal adjacency [76] or in terms of video content. The latter results in the entity called *scene*, which should convey semantic information in the video being viewed. Hence, our objective is to build a browsing hierarchy whose shape and size are determined only by the semantic complexity of the video. We call such a hierarchical structure as an *adaptive scene tree*. The scene tree algorithm described in this section is motivated by the work of Oh and Hua [10] who build a scene tree for uncompressed videos. Our algorithm described in subsection 2.2.1 is an extension and an improvement over [10] that works for compressed videos. Subsection 2.2.2 provides an illustration example of the scene tree, which helps the reader to understand the scene tree construction algorithm. Subsection 2.2.3 describes the representation of the scene tree as an MPEG-7 compliant XML document.

2.2.1 Scene Tree Building Algorithm

The main idea of the scene tree building algorithm is to sequentially compare the similarity of the key frame from the current shot to the key frames from the previous w shots. Based on a measure of similarity, the current shot is either appended to the parent node of the previous shot or appended to a newly created node. The result of the scene tree building algorithm is a browsing tree whose structure is adaptive, i.e. the number of levels of the tree are larger for complex content and smaller for simple content. The details of the algorithm, which takes in a sequence of video shots and outputs the scene tree are summarized in Algorithm 1.

The first *for* loop (line 1 through line 4) is the initialization phase of the algorithm. We Create a scene node SN_i^0 at the bottom level of the scene tree for each $shot_i$. The

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Algorithm 1 Build Scene Tree**Input:** Sequence of video shots.**Output:** Scene Tree, SN .

```

1: for  $i = 1$  to  $NumOfShots$  do
2:   Create  $SN_i^0$  for  $shot_i$ .
3:    $SN_i^0.keyframe = SelectKeyframe(shot_i)$ .
4: end for
5:  $i \leftarrow 3$ .
6: while  $shot_i \neq NULL$  do
7:    $found = false$ .
8:   for  $j = i - 1$  to  $i - w$  do
9:     if  $isSimilar(shot_i, shot_j)$  then
10:       $found = true$ .
11:      if  $SN_{i-1}^0$  and  $SN_j^0$  do not currently have a parent node then
12:        Connect all scene nodes,  $SN_i^0$  through  $SN_j^0$  to a new empty node as their
        parent node. //Condition (a)
13:      else if  $SN_{i-1}^0$  and  $SN_j^0$  share an ancestor node then
14:        Connect  $SN_i^0$  to this ancestor node. //Condition (b)
15:      else
16:        Connect  $SN_i^0$  to the current oldest ancestor of  $SN_{i-1}^0$ , and then create a
        new empty node and connect the oldest ancestor of all nodes from  $SN_j^0$  to
         $SN_{i-1}^0$  to it as its children. //Condition (c)
17:      end if
18:      break;
19:    end if
20:  end for
21:  if  $found = false$  then
22:    Create a new empty node and connect  $SN_i^0$  to it as its child node.
23:  end if
24:   $i = i + 1$ 
25: end while
26: if All the  $SN$ s are not connected then
27:   Connect all the nodes currently without a parent to a new empty node as their
   parent.
28: end if
29: Traverse the scene tree in bottom-up order. When an empty node visited (with no
   key frame), we identify its child node which contains the largest number of frames
   and assign the child node's representative frame as the representative frame for this
   node.

```

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

subscript indicates the shot (or scene) from which the scene node is derived and the superscript denotes the level of the scene node in the scene tree. For each SN_i^0 (corresponding to $shot_i$), we select the key frame as its representative frame by choosing the I frame in a shot whose DC value is closest to the average of the DC values of all the I frames in the shot. Since we do not desire to decode all the frames in the video sequence (recall that we would like to process in the compressed domain), we choose the DC value of the I frame to make the algorithm more efficient. The *while* loop (line 6 through line 25) performs the sequential comparison of the similarity between the key frame from the current shot and the key frames from the previous w shots (in descending order) to build up the scene tree. The similarity is calculated using a function *isSimilar()* (line 9), which will be described shortly. The comparisons stop when a similar shot, say $shot_j$, is found. The following *if* statements (line 9 through 19) grow the scene tree according to 3 different conditions: a) if SN_{i-1}^0 and SN_j^0 do not currently have a parent node, we connect all scene nodes, SN_i^0 through SN_j^0 to a new empty node as their parent node; b) if SN_{i-1}^0 and SN_j^0 share an ancestor node, we connect SN_i^0 to this ancestor node; c) if SN_{i-1}^0 and SN_j^0 do not currently share an ancestor node, we connect SN_i^0 to the current oldest ancestor of SN_{i-1}^0 , and then create a new empty node and connect the oldest ancestor of all nodes from SN_j^0 to SN_{i-1}^0 to it as its children. If there is no similar shot found, a new empty node is created and connected to SN_i^0 as its parent node (line 21 through 23). If the scene tree is split into multiple parts, we merge them into one tree by connecting all the nodes currently without a parent to a new empty root node (line 26 through 28). In the end, we traverse the scene tree from the bottom to the top. For each empty node visited, we identify the child node which contains the largest number of frames and assign the child node's representative frame as the representative frame for this node (line 29).

We now return to the function *isSimilar()* (line 9 in Algorithm 1) used to compute the similarity between key frames. The computation of the similarity involves dividing the

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

frames into background area and foreground area. Instead of using a fixed background and a fixed object area as in [10], we use a byproduct of the camera motion estimation algorithm described in section 2.3. This byproduct, which we call a “Block Rejection Map” consists of those macroblocks (MBs) which corresponds to moving foreground object region in the scene. The reason for such a nomenclature will be evident in section 2.3; however for the moment, it will suffice to understand the *BRM* as a partition of the frame into foreground and background areas. Figure 2.4 (section 4) shows an example of a *BRM*. Operating on the HSV color space, we compute the color histograms of the foreground and background area of two key frames k_1 and k_2 . The Hue(H) and Saturation(S) values are quantized into 8 bins and 4 bins respectively, while we ignore the Illumination(V) component.

The similarity measure is defined as

$$Sim = w_1 \times |hist_b(k_1, k_2)| + w_2 \times |hist_o(k_1, k_2)| \quad (2.5)$$

where $|hist_b(.)|$ and $|hist_o(.)|$ are the Euclidean distances of histograms of the background and object areas, respectively, between frames k_1 and k_2 and w_1, w_2 are weights for the background and object areas, respectively. We choose $w_1 = 0.7$ and $w_2 = 0.3$ so as to give more weight to the background since a change in the background area is a stronger indication of a change in the scene. The value of *Sim* in equation (2.5) is compared with a threshold to determine if two key frames are similar.

Although the above adaptive scene tree building algorithm is motivated by [10], there are some important differences, which we highlight now. As mentioned earlier, our algorithm works directly on MPEG-2 compressed video in the sense that we need only partially decode the video to extract the motion vectors and DC images. Moreover, a full decoding is required only for one frame per shot that represents the key frame. Secondly, the similarity between shots is ascertained through a meaningful predefined window, w ,

as opposed to all the shots from the beginning of the video. We now compare the computational complexity of our algorithm to [10]. The shot similarity determination can be done in $O(w * s)$, where w is the length of the window and s is the number of shots. Generally, the number of frames in a shot, f is much larger than s and s is larger than w ($f \gg s > w$). The scene tree construction algorithm involves traversal in step 4 and step 6. The computational complexity of tree transversal is $O(\log(s))$. Therefore, the worst-case computational complexity of building the tree is of $O(s * \log(s))$. Thus, the total computational complexity of our algorithm is of $O(s * \log(s))$, while that in [10] is $O(s * f^2)$. Finally, we partition the frame into background and foreground areas using “Block Rejection Map” dynamically, while the frame is partitioned into fixed background/object area partition in [10]. The usage of “Block Rejection Map” gives more accurate partitioning of object and background areas, and hence results more accurate scene similarity calculation.

2.2.2 Illustration of a Scene Tree Construction

In this section, we walk through the steps leading to the construction of a scene tree, using a symbolic video sequence consisting of 10 shots. The shots denoted by the same gray color in Figure 2.1(a) are regarded as similar by the *isSimilar()* function.

The scene tree construction algorithm is initialized by attaching a scene node, $SN_{0,i}$ for each shot i , $i = 1, \dots, 10$ as in Figure 2.1(a). The shots are scanned to check the similarity between scene node pairs. $SN_{0,4}$ is found to be similar to $SN_{0,2}$. Since the previous node $SN_{0,3}$ and $SN_{0,2}$ do not have a parent node, $SN_{1,1}$ is created in the upper level and all scene nodes from $SN_{0,2}$ to $SN_{0,4}$ are connected to $SN_{1,1}$ (condition (a) at line 12 in Algorithm 1). Next, $SN_{0,5}$ is found to be similar to $SN_{0,3}$. Since the previous $SN_{0,4}$ and $SN_{0,3}$ share a common ancestor node, $SN_{1,1}$, $SN_{0,5}$ is connected to $SN_{1,1}$ as the child node (condition (b) at line 14 in Algorithm 1). Figure 2.1(b) shows the constructed

scene tree at this stage. Since $SN_{0,6}$ is not similar to any of the previous three shots, $SN_{1,2}$ is created as its parent. However, $SN_{0,7}$ is similar to $SN_{0,4}$ and since $SN_{0,6}$ and $SN_{0,4}$ do not share a common parent, a new node $SN_{2,1}$ is created with children $SN_{0,1}$, $SN_{1,1}$ and $SN_{1,2}$ (condition (c) at line 16 in Algorithm 1). Next, $SN_{0,8}$ is similar to $SN_{0,6}$ and since $SN_{0,6}$ and $SN_{0,7}$ share the same parent node, $SN_{0,8}$ is also connected as their sibling. The scene tree at this stage is illustrated in Figure 2.1(c). We observe that if the comparison of the scene nodes are not restricted to within a window, $SN_{0,6}$ would be deemed similar to $SN_{0,1}$ causing loss of the inherent semantic information in the tree. $SN_{0,9}$ and $SN_{0,10}$ are found to be different from all other shots considered so far and a new parent node $SN_{1,3}$ is created for them. Finally, the scene node $SN_{2,1}$ and $SN_{1,3}$ are connected to the root node $SN_{3,1}$ of the scene tree, which is shown in Figure 2.1(d).

2.2.3 Scene Tree Representation

The algorithm in the subsection 2.2.1 outputs a description for the video content in the form of a tree structure called the Scene Tree. If the ‘Scene Tree’ is not represented in a standard way, it would not be possible to access content across several repositories to enable content exchange using different description methods. These are interoperability issues. We can define a standardized description to solve the interoperability issue so that the scene tree information can be shared and reused. The MPEG-7 standard is aimed at addressing this problem. Formally known as “Multimedia Content Description Interface”, it provides a set of “Descriptors” and “Description Schemes”, along with a “Description Definition Language”, to describe multimedia content [77]. The scope of the standard covers various forms of media including audio, video and images, both in digital and analog format, although most applications involve only digital content. MPEG-7 allows a standard description scheme of various aspects of multimedia material that can be used by MPEG-7 enabled applications aimed at end user, or automatic systems. Thus, we choose to represent the information in a scene tree in MPEG-7 compliant XML document.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.1: Illustration of scene tree construction algorithm.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

XML (eXtensible Markup Language) [78] is a simple and flexible text format derived from SGML (ISO 8879). It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification introduced by the World Wide Web Consortium (W3C). XML schema has been widely used as a schema language for constraining the structure and content of XML document. After a detailed evaluation of XML and XML schema, MPEG-7 chose to adopt and extend XML and XML schema as the description definition language (DDL) for MPEG-7 document. Thus the MPEG-7 documents are XML documents that conform to particular MPEG-7 schemas (expressed in DDL) and that describe audiovisual content. In MPEG-7, there are a collection of schemas called “Multimedia Description Schema (MDS)”. The MDS provides a set of standardized tools to facilitate multimedia content description. In MDS, there is a specific schema for temporal video decomposition, which matches our scene tree based video description.

An example of an MPEG-7 compliant XML document is shown in Figure 2.2. The XML document starts with `< Mpeg7 >` as the root node. This node has two children: `< DescriptionMetadata >` and `< Description >`. The `< DescriptionMetadata >` is used to describe the metadata information about the XML document, such as author, confidence, version etc. and the `< Description >`, which is of `ContentEntityType`, is used for multimedia content description. The scene tree information is contained in the `< TemporalDecomposition >` node under “Mpeg7/ Description/ MultimediaContent/ Video/”. Since there is no gap in the scene tree decomposition and parent scene nodes are overlapped with child scene nodes, the *gap* and *overlap* attributes of `< TemporalDecomposition >` are set to false and true, respectively. Each node in the scene tree is represented by a `< VideoSegment >`, which contains a `< StartFrame >`, a `< EndFrame >` and one or more `< VideoSegment >(s)` depending on number of child scene nodes that the current scene node has. Note that other MPEG-7 visual descriptors such as color, texture, motion etc. can be inserted to the `< VideoSegment >` too.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

```

<?xml version="1.0" encoding="iso-8859-1"?> <Mpeg7>
  <DescriptionMetadata>
    .....
  </DescriptionMetadata>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>file:///c:/example.mpg</MediaUri>
        </MediaLocator>
        <TemporalDecomposition gap="false" overlap="true">
          <VideoSegment>
            <StartFrame>0</StartFrame>
            <EndFrame>2999</EndFrame>
            <Keyframe>1020</Keyframe>
          <VideoSegment>
            <VideoSegment> ... </VideoSegment>
            ...
          </VideoSegment>
          .....
        </VideoSegment>
      </TemporalDecomposition>
    </Video>
  </MultimediaContent>
</Description>
</Mpeg7>

```

Figure 2.2: Example of XML representation for scene tree.

We will illustrate the XML representation of actual scene trees derived from video sequences in section 2.4.

2.3 Motion based Indexing and Retrieval

Video retrieval and browsing can be envisioned as an integrated and interactive process. This process can be seen to be analogous to browsing through books with a Table-of-Contents in the front and an Index at the back of the book. In order to search for a specific topic of interest, one may consult the index to locate the pages where it appears. Consequently, if one wants to know more about the topic, he may refer to the Table-of-Contents to browse through chapters, sections and so on. Similarly, for video data, the feature vectors serve as the Index and the scene tree serves as the Table-of-Contents. The

search for relevant content in the video is initiated by locating the video shot by feature matching/retrieval and then carried forward by browsing up and down along the scene tree (table of content) to further explore the relevant content. Hence, the feature vector and the scene tree are closely related tools for the management of the video content. In this section, we describe a new motion feature for video indexing and retrieval.

The motion content is represented by three components, viz., the camera motion (CM), the object motion (OM), and the total motion (TM). Each of CM , OM and TM are matrices and it is these matrices that serve as indices for video retrieval.

2.3.1 Motion Extraction

Recall that all the videos considered here are compressed according to MPEG-2 format. Hence, the MV for frame n is obtained as

$$MV_n = \begin{cases} -Bk_{n-1}, & \text{for I frame} \\ Fw_n - Fw_{n-1}, & \text{for P frame} \\ (Fw_n - Fw_{n-1} + Bk_n - Bk_{n-1})/2, & \text{for B frame} \end{cases} \quad (2.6)$$

Refer to section 2.1 for meaning of the notations. The MV fields (MVF) obtained are then smoothened using a spatial $(3 \times 3 \times 1)$ median filter followed by a temporal $(1 \times 1 \times 3)$ median filter. The smoothing effect of this two-step median filter is similar to that of a joint $3 \times 3 \times 3$ spatio-temporal median filter. The joint spatio-temporal median filter returns the 14th ranked MV; while the two-step median filter returns a MV that is ranked between 10th and 18th. But the two-step median filter is slightly more efficient than the joint spatio-temporal filter. The smoothened MVFs are then used to calculate the camera motion and object motion. The total motion component at each MB is just the smoothened motion vector at its center.

Parametric model based camera motion estimation has been proposed in literature [79] [80] [81]. In [79], a four parameter model is used to estimate the camera motion. In [80], the authors studied several parametric models for camera motion estimation. In a

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

recent paper [81], the nonlinear perspective model is used to model the camera motion. All the above approaches for camera motion estimation involve the computation of a Hessian matrix and its inverse, which is computational expensive. In the following, we provide our efficient camera motion estimation algorithm. We derive the equations for the parameters of the camera motion model explicitly. The proposed algorithm avoids the need to compute the inverse of the Hessian matrix.

We use a six parameter affine model to estimate the camera motion which is modelled as

$$\begin{aligned} mv_x(i, j) &= p_1 \cdot i + p_2 \cdot j + p_3 \\ mv_y(i, j) &= p_4 \cdot i + p_5 \cdot j + p_6 \end{aligned} \quad (2.7)$$

where $mv_x(i, j)$ and $mv_y(i, j)$ are the x and y components of the motion vector for a MB centered at (i, j) , ps are the affine parameters. Consider a group of MBs, G , whose affine parameters $P = \{p_1, \dots, p_6\}$ need to be determined. We will explain the significance of G shortly, when we present the iterative algorithm for camera motion estimation. Using the method of least squares, from equation (2.7), P is obtained by minimizing

$$\sum_G (mv_x(i, j) - p_1 \cdot i - p_2 \cdot j - p_3)^2 + (mv_y(i, j) - p_4 \cdot i - p_5 \cdot j - p_6)^2 \quad (2.8)$$

The two terms inside the summation can be minimized independently. Considering the first term only and differentiating it with respect to p_1, p_2 and p_3 and setting the resulting equations to zero, we get

$$\sum_{(i,j) \in G} (mv(i, j) - p_1 \cdot i - p_2 \cdot j - p_3)i = 0 \quad (2.9)$$

$$\sum_{(i,j) \in G} (mv(i, j) - p_1 \cdot i - p_2 \cdot j - p_3)j = 0 \quad (2.10)$$

$$\sum_{(i,j) \in G} (mv(i, j) - p_1 \cdot i - p_2 \cdot j - p_3) = 0 \quad (2.11)$$

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

If the origin of the frame is taken to be at its center (instead of the top left corner as is conventionally done), $i' = i - \frac{V+1}{2}$ and $j' = j - \frac{U+1}{2}$, $U \times V$ is the size of the motion vector field, then $\sum_G i' = 0$ and $\sum_G j' = 0$ and the affine parameters can be easily shown to be

$$\begin{aligned} p_1 &= \frac{IX \cdot YY - JX \cdot XY}{A}, & p_2 &= \frac{JX \cdot XX - IX \cdot XY}{A}, \\ p_3 &= \frac{\sum_G mv_x(i, j)}{g}, & p_4 &= \frac{IY \cdot YY - JY \cdot XY}{A}, \\ p_5 &= \frac{JY \cdot XX - IY \cdot XY}{A}, & p_6 &= \frac{\sum_G mv_y(i, j)}{g}. \end{aligned} \quad (2.12)$$

where

$$\begin{aligned} IX &= \sum_G i' \cdot mv_x(i', j'), & JX &= \sum_G j' \cdot mv_x(i', j'), \\ IY &= \sum_G i' \cdot mv_y(i', j'), & JY &= \sum_G j' \cdot mv_y(i', j'), \\ XX &= \sum_G i'^2, & YY &= \sum_G j'^2, & XY &= \sum_G i' \cdot j', \\ A &= XY \cdot XY - XX \cdot YX, \end{aligned}$$

and g is the number of macroblocks in the group G . $mv(i', j') = mv(i, j)$ since they refer to the same MB.

An iterative algorithm with outlier rejection algorithm is used to estimate the camera motion parameters. Since the motion vector field is composed of both camera motion and object motion, the camera motion should be estimated from the homogenous camera motion field. By iteratively rejecting the ‘outlier’ blocks, which do not fit into the camera motion, the homogenous camera motion field is refined and hence a more accurate estimation is resulted. The algorithm to estimate the affine parameters starts by labelling all the MBs in a frame as ‘inliers’. Then the parameters are estimated for all the inliers using equation (2.12). A new set of motion vectors are reconstructed for each inlier using the estimated parameters. If the magnitude of the residual motion vector R_{mv} , calculated

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.3: Histogram of magnitudes of residual motion vectors between two frames in an example ‘soccer’ video.

as the difference between the original motion vector and the reconstructed one, is greater than an adaptive threshold $T = \max(\text{median}(R_{mv}), \beta)$ for a particular MB, then that MB and the one situated diagonally opposite it are marked as ‘outliers’. The role of β is to prevent the rejection of a large number of MBs if the median of the residuals is very small. We choose β to be 1. Figure 2.3 shows the histogram of the magnitudes of residual motion vectors between two frames in an ‘example’ soccer video. Note that the diagonally opposite MBs are also marked as ‘outliers’ due to the shifting of the co-ordinate axes to the center of the frame. After each iteration, some MBs are marked as ‘outliers’; these MBs correspond to areas in which the motion is associated with moving objects. The steps in the algorithm to estimate the camera motion are shown in Algorithm 2.

When Algorithm 2 is terminated, the MBs that are rejected as outliers correspond to moving foreground regions. The background area consisting of MBs labelled as inliers are marked with zero intensity. Such a partition of a frame into foreground and background regions results in the Block Rejection Map; an example of which is shown in Figure 2.4. As mentioned in the previous section, we can utilize the *BRM* for the calculation of the similarity between two scenes since the meaningful calculation of scene similarity requires different treatments of background and foreground areas.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Algorithm 2 Camera Motion Estimation

- 1: Mark all the MBs as 'inliers'.
- 2: Estimate the motion affine parameters for 'inliers' (equation (2.12)).
- 3: Reconstruct the global motion vector at each macroblock with the estimated affine parameters.
- 4: Calculate residual motion vector (R_{mv}) as the difference between the original and reconstructed motion vector.
- 5: If R_{mv} is greater than $\max(\text{median}(R_{mv}), \beta)$, then mark this MB and its opposite diagonal MB as 'outliers'. $\text{median}(R_{mv})$ is the median of the all the R_{mv} s.
- 6: Go to step 2 until there are no more new 'outliers' or more than two thirds the MBs are marked as 'outliers'.
- 7: If more than two thirds of the MBs are 'outliers', the affine parameters are set to zero.

Figure 2.4: Block Rejection Map.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

After estimating the camera motion, we now describe the CM , OM , TM matrices for motion content indexing. The CM , OM , TM are designed to measure the amount of motion. Hence, we ignore the direction of the motion vectors and only use the magnitude as the measure. The total amount of motion contains the amount of camera motion and object motion and should be greater than those two. Since it is possible that the $|cm|$ is larger than $|mv|$ i.e. the object motion vector and camera motion vector have opposite directions, for simplicity we choose the minimal between mv and cm as the camera motion as follows:

$$|cm(i, j)| = \min(|mv(i, j)|, |cm(i, j)|) \quad (2.13)$$

where $cm(i, j)$ and $mv(i, j)$ are the camera motion vector and total motion vector, respectively, at the MB centered at (i, j) . For the same reason

$$|om(i, j)| = \max(0, |mv(i, j) - cm(i, j)|) \quad (2.14)$$

where $om(i, j)$ is the object motion vector at the MB centered at (i, j) . The CM , OM and TM matrices are now formed for a shot by accumulating $|cm(i, j)|$, $|om(i, j)|$ and $|mv(i, j)|$, respectively, over all the frames in the shot, i.e., $CM(i, j) = \sum_{l=1}^n cm_l(i, j)$, $OM(i, j) = \sum_{l=1}^n om_l(i, j)$, and $TM(i, j) = \sum_{l=1}^n mv_l(i, j)$, where n is the number of frames in the shot. These matrices, which represented the integrated amount of motion for the whole video segment, serve as indices for retrieval, as explained in the next subsection.

2.3.2 Video Similarity Measure

Each shot is characterized by the three matrixes CM , OM and TM . In order to compute the similarity between two shots, we assume that the matrices belong to the Hilbert sequence space l^2 with metric defined by

$$d(M_x, M_y) = \left[\sum_{u=1}^U \sum_{v=1}^V |M_x(u, v) - M_y(m, n)|^2 \right]^{\frac{1}{2}} \quad (2.15)$$

where M_x and M_y are the motion matrices of order $U \times V$. However, instead of comparing matrices, we reduce the dimension of the feature space by projecting the matrix elements along the rows and the columns to form one dimensional feature vectors $M_x^r(v) = \sum_{u=1}^U M_x(u, v)$ and $M_x^c(u) = \sum_{v=1}^V M_x(u, v)$. These projections can be thought of a special case of a linear projection.

The metric in equation (2.15) can then be rewritten as

$$d(M_x, M_y) = \left[\sum_{v=1}^V \frac{1}{U} |M_x^r(v) - M_y^r(v)|^2 + \sum_{u=1}^U \frac{1}{V} |M_x^c(u) - M_y^c(u)|^2 \right]^{\frac{1}{2}}. \quad (2.16)$$

This leads us to the distance function between two shots s_x and s_y to be defined as

$$\hat{d}(s_x, s_y) = \omega_C \cdot d(CM_x, CM_y) + \omega_O \cdot d(OM_x, OM_y) + \omega_T \cdot d(TM_x, TM_y), \quad (2.17)$$

where the $d(\cdot)$'s are computed according to equation (2.16) and the ω 's are the weights for each motion component. The weights are assigned in such a way that there is equal contribution from each of the components to the distance function. Use of the distance function enables current matrix based indexing techniques, like M-tree, R-tree etc. to be applied as the index to the feature vectors to facilitate efficient search.

2.4 Experimental Results

Our experiments were designed to assess the performance of the proposed techniques for SBD, scene tree construction, motion estimation and motion based indexing and retrieval of shots. In the following subsections, we present the experimental results for shot boundary detection, scene tree construction, motion estimation and motion based indexing and retrieval of shots.

2.4.1 Shot Boundary Detection

We have tested our algorithm on a wide variety of video sequences like soccer, news, movie and cartoons which were encoded in the MPEG-2 format with a *GOP* length of

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.5: (a) $MFDR$ for a ‘news’ sequences; (b) Local sum of $DMBC$ in a dissolve.

Table 2.1: Summary of SBD results

Video (Frames)	Cut (Correct,False)	Gradual (Correct,False)	Recall/Precision Cut	Recall/Precision Gradual
News(32640)	43(42,4)	12 (11,2)	97.67/91.3	91.67/84.62
Movie(29010)	17(16,2)	3 (3 ,1)	94.12/88.89	100/75
Cartoon(1498)	13(13,2)	0 (0 ,0)	100/86.67	—
Soccer(21310)	32(31,2)	10 (9,3)	96.88/93.94	90/75.00
Total(84458)	105(102,10)	25 (23,6)	97.41/91.75	92.00/79.31

12 frames. The encoding pattern was

$$\dots IBBPBBPBBPBB \dots$$

There were a total of 105 hard cuts and 25 gradual transitions of which 20 are dissolves, 3 are fade-ins and 2 are fade-outs. Figure 2.5(a) shows the $MFDR$ for 373 frames of a ‘News’ video. As expected, the hard cuts at frames 120 and 300 have a very high $MFDR$, which enables easy detection of the shot boundary. As for gradual transitions, local sum of $DMBC$ for a ‘Movie’ sequence is shown in Figure 2.5(b). We note that the beginning of the dissolve takes place at frame 43 and ends at frame 68. Table 2.1. summarizes the results of shot boundary detection. In the experiment, the global threshold T_g in equation (2.3) for detecting hard cuts was set to 1. For detecting gradual transitions, the length of the window over which the local sum is computed is set to 12, with τ (the threshold for F_n in equation (2.4)) set to 10. The video sequences are chosen so as to

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

include various kinds of camera and object motions. We use *precision* (the ratio of the number of shot changes detected correctly to the actual number of shot changes) and *recall* (the ratio of the number of shot changes detected correctly over the total number of shot changes detected) to measure the performance of the SBD algorithm.

The recall rates for hard cuts are consistently above 95% over all the sequences reinforcing the fact the choice of the thresholds are not critical to the success of the method. The missed cuts are mostly due to large motion at the shot boundary. The overall recall rate for gradual transitions is 92% (only 2 of the 25 transitions were missed). Note that τ is chosen the same for all the sequences. The false detection of hard cuts is attributed to the sudden change in brightness of the frame due to flash in the ‘News’ sequence, a lightning bolt in the ‘Movie’ sequence and the sudden appearance of text in the ‘Cartoon’ sequence. We must mention that the ‘News’ sequence has 45 instances of flashes which were potential candidates for shot boundaries, out of which only 4 were falsely detected, and yields the precision rate of 91.3%. However, the overall precision obtained for hard cuts is 91.75%. The false detection of gradual transitions is mainly due to the undesirable shaking of the camera which pulls down the camera motion characterization parameters to near zero. This is evident from the relatively low precision obtained for ‘Soccer’ and ‘Movie’ sequence where such jerky movements of the camera are present.

We compare the results for gradual shot boundary detection with Jun’s method [2]. Both of these two methods use MB type information to detect shot boundaries. In [2], the shot changes are detected by a set of rules based the spatial-temporal distribution of the MB type information. Figure 2.6 shows the ‘Recall’ and ‘Precision’ rates for gradual shot boundary detection using our method and Jun’s method [2]. Our method detects more gradual shot changes than [2] with slightly more false positives. Compared to the slightly worse ‘precision’, the high ‘recall’ rate of our method is more promising because low recall rate in the initial stage is not desirable. Since both methods are very fast algorithms in

Figure 2.6: Comparison of (a) 'Recall' and (b) 'Precision' rates using the proposed SBD algorithm and Jun's algorithm [2] respectively for gradual shot change detection.

compressed domain, they can be used as the first phase of shot detection. To increase the precision, we can apply other shot boundary algorithms in the uncompressed domain, as a postprocessing step. If initially the recall rate is low, there is no way to recover the missed shot boundaries.

2.4.2 Adaptive Scene Tree Construction

To evaluate the scene tree building algorithm, we run the algorithm described in section 3 for various videos. Since we cannot quantify the effectiveness of the algorithm to produce the scene trees, we show four examples of scene trees in Figure 2.7, Figure 2.8, Figure 2.9 and Figure 2.10 and describe the main events occurring in the video sequences from which the trees are constructed.

The scene tree in Figure 2.7 is built from a three minute video clip available from the MPEG-7 test video set (CD 21 Misc2.mpg). In this sequence, two women are watching the TV, and talking to each other (the first 8 shots, leaf nodes). Then one of the women leaves while the other continues watching TV (next 4 shots). After a while, this woman also leaves (starting from frames 3300). Then both the women appear together. One of

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.7: Scene tree of MPEG-7 test video.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

them puts on a coat and leaves while the other starts cleaning the table (the last 4 shots, leaf nodes). The output XML file for the scene tree is shown below:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Mpeg7>
  <DescriptionMetadata>
    <Confidence>0.85</Confidence>
    <Version>1.4</Version>
    <LastUpdate>2003-09-1T10:00:00+00:00</LastUpdate>
    <Comment>
      <FreeTextAnnotation>
        Scene Tree.
      </FreeTextAnnotation>
    </Comment>
  </DescriptionMetadata>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>
            file:///c:/MPEG-7_tv.mpg
          </MediaUri>
        </MediaLocator>
        <TemporalDecomposition gap="false" overlap="true">
          <VideoSegment>
            <StartFrame>0</StartFrame>
            <EndFrame>4432</EndFrame>
            <Keyframe>2820</Keyframe>
            <VideoSegment> ... </VideoSegment>
            .....
          </VideoSegment>
        </TemporalDecomposition>
      </Video>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

The <DescriptionMetadata> carries the meta data information of the scene tree XML document (e.g. Confidence, Version, LastUpdate, Comments). The main content of the scene tree is stored in the <TemporalDecomposition> descriptor under “/Description /MultimediaContent /Video /”. Each scene node is represented by a <VideoSegment>. In total, there are 26 <VideoSegment>s which correspond to the number of scene nodes in the scene tree shown in Figure 2.7. The <MediaLocator> descriptor under the <Video> indicates the source of the video.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.8: Scene tree of TV program "Opening The New Era".

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

The scene tree in Figure 2.8 is built from a two-minute video clip from the TV program “Opening The New Era”. This video sequence consists of three scenes. A man and a woman are taking a picture together. Then the man drives the woman to her home. The woman goes inside her home and reads the newspaper. If we traverse the scene trees from the top to bottom, which is in essence a non-linear browsing of the video, we get the above stories. The scene tree give a hierarchical view of the scenes as well as provides a summary of the video sequence. The output XML document for the scene tree is shown below:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Mpeg7>
  <DescriptionMetadata>
    <Confidence>0.85</Confidence>
    <Version>1.4</Version>
    <LastUpdate>2003-09-1T10:00:00+00:00</LastUpdate>
    <Comment>
      <FreeTextAnnotation>
        Scene Tree.
      </FreeTextAnnotation>
    </Comment>
  </DescriptionMetadata>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>
            file:///c:/csj_tv.mpg
          </MediaUri>
        </MediaLocator>
        <TemporalDecomposition gap="false" overlap="true">
          <VideoSegment>
            <StartFrame>0</StartFrame>
            <EndFrame>2999</EndFrame>
            <Keyframe>1020</Keyframe>
            <VideoSegment> ... </VideoSegment>
            .....
          </VideoSegment>
        </TemporalDecomposition>
      </Video>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

The scene tree information is stored in the <TemporalDecomposition> descriptor.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.9: Scene tree of MPEG-7 test video (CD20#1).

There are 24 nodes in the scene tree, thus there are 24 `<VideoSegment>` under the `<TemporalDecomposition>` descriptor.

Figure 2.9 shows the scene tree extracted from another MPEG-7 test video (CD20 Misc1.mpg). The length of the video clip is one minute. This video clip contains two different events. The first event happens between a boy and a girl. The boy sits on a chair and plays with a dog in the room. A girl enters the room and sees the boy. She begins to talk to him. Then the girl begins to feed the goldfish. The next event in this video clip happens between the boy and two other men. One man introduces the boy to the other. The output XML document for the scene tree is as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Mpeg7>
  <DescriptionMetadata>
    <Confidence>0.85</Confidence>
    <Version>1.4</Version>
    <LastUpdate>2003-09-1T10:00:00+00:00</LastUpdate>
    <Comment>
      <FreeTextAnnotation>
        Scene Tree.
      </FreeTextAnnotation>
    </Comment>
  </DescriptionMetadata>
</Mpeg7>
```

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

```

    </Comment>
  </DescriptionMetadata>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>
            file:///c:/MPEG-7_CD20_1.mpg
          </MediaUri>
        </MediaLocator>
        <TemporalDecomposition gap="false" overlap="true">
          <VideoSegment>
            <StartFrame>0</StartFrame>
            <EndFrame>1499</EndFrame>
            <Keyframe>915</Keyframe>
            <VideoSegment> ... </VideoSegment>
            .....
          </VideoSegment>
        </TemporalDecomposition>
      </Video>
    </MultimediaContent>
  </Description>
</Mpeg7>

```

Figure 2.10 shows another scene tree extracted from the same video as in Figure 2.9. This video clip lasts one minute. The story happens in the classroom. There are three events in this video clip. The first event is that the teacher walks into the classroom and begins the class. The second event is that the students hear the bell and think the class is over. They begin to leave the classroom. The teacher gets angry and orders the student to go back to their seats and continues the class. The corresponding XML document of the scene tree is shown below:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<Mpeg7>
  <DescriptionMetadata>
    <Confidence>0.85</Confidence>
    <Version>1.4</Version>
    <LastUpdate>2003-09-1T10:00:00+00:00</LastUpdate>
    <Comment>
      <FreeTextAnnotation>
        Scene Tree.
      </FreeTextAnnotation>
    </Comment>
  </DescriptionMetadata>

```

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.10: Scene tree of MPEG-7 test video (CD20#2).

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.11: (a) Video annotation tool (VideoBuddy) interface; (b) key frame annotation.

```
<Description xsi:type="ContentEntityType">
  <MultimediaContent xsi:type="VideoType">
    <Video>
      <MediaLocator>
        <MediaUri>
          file:///c:/MPEG-7_CD20_2.mpg
        </MediaUri>
      </MediaLocator>
      <TemporalDecomposition gap="false" overlap="true">
        <VideoSegment>
          <StartFrame>0</StartFrame>
          <EndFrame>1499</EndFrame>
          <Keyframe>1335</Keyframe>
          <VideoSegment> ... </VideoSegment>
          .....
        </VideoSegment>
      </TemporalDecomposition>
    </Video>
  </MultimediaContent>
</Description>
</Mpeg7>
```

As described in previous sections, the scene tree can be viewed as “table-of-content” of a video. It can be used as a guide for video content browsing and navigation. It has a very wide range of practical applications. We have integrated the “scene tree” into a semiautomatic video annotation tool, called “VideoBuddy”, whose interface is shown

in Figure 2.11(a). The user selects the input video to be annotated. Our algorithm automatically performs the shot boundary detection, key frame extraction and builds up the scene tree. The generated scene tree is shown in the bottom left part in the interface. The key frames from each shot are shown as thumbnail images on the right panel besides the scene tree panel. Then, the user can navigate the “scene tree” and select interesting scene nodes for annotation. When the user select one scene node, the corresponding scene node information from the scene tree is shown in “shot information” window. The user can view the selected scene in the embedded media player. After that, he can annotate the video with key words by ticking the appropriate checkbox. The user can also perform the key frame region annotation by drawing rectangles over the annotated regions (Figure 2.11(b)). Although “VideoBuddy” is still a semiautomatic video annotation tool, the automatically extracted “scene tree” provides valuable information for video navigation and facilitates the annotation of the video content.

2.4.3 Motion Estimation

In this subsection, we evaluate the proposed algorithm for motion estimation. We consider video sequences that are compressed using the MPEG-2 standard. Figure 2.12 shows four examples of such videos. Each row consists of the key frame in a particular shot, camera motion, object motion and total motion. The motion information contained in the CM , OM and TM matrices are represented as monochrome images for visualization purposes; the brighter pixels indicate higher motion activity. The first row in Figure 2.12 shows the shot of a news anchor person in which the camera is stationary. This is reflected in the black CM image while the motion of the object (face of the person) is clear in the OM image. Thus the total motion consists essentially of the object motion only. Similarly in the second row which shows a scene with a moving camera while the objects in the scene are stationary, the OM image is black (indicating

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

no motion) while the *CM* image is uniformly bright. The third row illustrates the case of both object motion and camera motion contributing to the total motion in the shot from a soccer game. We can clearly see the zooming in of the camera in the *CM* image while object motion is indicated by some bright pixels in the *OM* image. Finally, we show a shot of the camera tracking a person and its corresponding motion images which are all mostly bright implying that the motion contains both camera as well as object motion. We see that the proposed motion estimation algorithm has been able to extract the motion information quite well.

Figure 2.12: Motion estimation. Each row consists of a key frame of the shot, the *CM* image, the *OM* image and the *TM* image of News anchor person (first row), gallery (second row), soccer (third row) and moving person (fourth row)

2.4.4 Motion Based Retrieval

We noted earlier that the video retrieval process is implicitly contained within the process of browsing the adaptive scene tree. However, the proposed retrieval method can also be considered as a ‘standalone’ process. To evaluate the performance of our proposed method for motion indexing and retrieval, we build a database consisting of video shots using the SBD algorithm on the 643 video sequences of the MPEG-7 test set. The lengths of the video clips range from 5 seconds to 30 seconds.

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

Figure 2.13: Motion based video retrieval. (a) Five example query results using TM matrix. (b) The same five query results using CM , OM and TM matrices.

Using the CM , OM and TM matrices as indices and the video similarity measure developed in section 2.3, we retrieve the top N video shots from the database. In Figure 2.13, we show the top 3 results of the query shown in the first column. We compare the retrieval results when only TM is used with the case when all the three motion matrices, viz., CM , OM and TM , are used to retrieve. Figure 2.13(a) shows the retrieval results with the motion feature extracted from TM matrix only, while Figure 2.13(b) shows the retrieval result when CM , OM and TM matrices are considered. The first columns of Figure 2.13(a) and Figure 2.13(b) are the key frames from the query video shots. The second, third and fourth columns of Figure 2.13(a) and Figure 2.13(b) are the first, second and third retrieval results. From these examples, we observe that both models retrieve video clips which have similar motion content. However, we see that the video similarity model with CM , OM and TM matrices gives better retrieval result in both the motion and semantic sense than the model that uses only TM matrix.

In the following experiment, we compare the efficiency of the proposed features with

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

PCA + motion vector features used in [82] and our motion histogram feature proposed in the following chapter. In [82], the authors illustrate the effectiveness of the proposed feature through two video applications: 1) video classification and 2) video motion content description. The paper shows that the first few principal components give a representation of the amount of camera motion. Since both our method and Zakhor's [82] work directly on the motion vectors extracted from the compressed domain and both methods are able to differentiate camera motion from object motion, we compare their performance for the task of video retrieval. We randomly choose 50 video shots from the video database as queries and retrieved the top 30 videos. *Precision* is used as the measure for performance, where *precision* is defined as the ratio of the number of relevant shots retrieved to the total number of shots retrieved. Two video shots are considered as relevant if they have close amount of motion in terms of total motion, camera motion and object motion. The frame size is 320 by 240. The motion vector field size is 20 by 15. Since we are using projection vector of the motion vector field, the actual dimension of our feature vector is $20 + 15 = 35$, the dimension of the *PCA*+motion vector feature[82] is set to 35 in the comparison. For all the 643 videos in our video database, the original dimension of the motion vector field is $9 \times 7 = 63$ (same as in [82]) and the first 30 principal components captures 98.3% of the total energy. We plot *precision* as a function of the number of retrieved video clips are shown in Figure 2.14. The average precision (computed as average of the precision with the number of top return video clips varying from 1 to 30) when *CM*, *OM* and *TM* are considered is 91% compared to 85% when only *TM* is used as an index and 87% when *PCA*+MV feature is used as an index. The performance of the features is $TM + OM + CM > MV + PCA > only TM$. We also find that both our proposed feature and the MV+PCA feature outperform the motion histogram feature in this experiment. The motion histogram feature is not as good as the $TM + CM + OM$ feature to differentiate the camera motion and object motion.

Figure 2.14: Precision as a function of the number of video clips retrieved. Blue line corresponds to retrievals using the TM , OM and CM ; red line corresponds to retrievals using TM only; pink line corresponds to retrieval using $MV+PCA$ (35D); black line corresponds to retrieval using motion histogram (8D).

2.5 Conclusion

In this chapter, we have presented a fully automatic content-based approach to organizing and indexing *compressed* video data. The three main steps in our approach are (i) shot boundary detection using motion prediction information of MPEG-2, (ii) development of a browsing hierarchy called adaptive scene tree and (iii) video indexing based on camera motion, object motion and total motion. The SBD algorithm can detect both abrupt cuts and gradual transitions. Unlike existing schemes for building browsing hierarchies, our technique builds a scene tree automatically from the visual content of the video. The size and shape of the tree reflect the semantic complexity of the video sequence. A video similarity measure based on the CM , OM and TM matrices is shown to give good retrieval results. The estimation of these matrices using the affine parameter model is also shown to perform well.

The hunt for highly discriminating features is a continuous one. From our study, we find that the CM , OM and TM matrices have excellent discriminatory characteristics. However, during computation of video similarity, the conversion of the matrices to a one dimensional vector might result in loss of valuable information. A more elegant and robust

CHAPTER 2. A SCENE TREE FOR BROWSING AND RETRIEVAL

use of these matrices is one of the future directions for this research. The proposed scene tree are integrated into the MPEG-7 standard description scheme to generate a MPEG-7 compliant XML metadata. In the MDS of MPEG-7, there is a descriptor to describe the temporal segment of the video. We use this segment descriptor to represent the scene tree since each node in the scene tree is a video segment. To check the effectiveness of the scene tree building algorithm, it will be useful to conduct user studies in the form of presenting the summarized video to several users and asking them to narrate the story as they understand it. However, it remains to be seen if a meaningful scene tree can be built for sports videos.

Chapter 3

New Motion Features and their Applications

Early video indexing methods were based on detecting shot boundaries followed by extracting key frames from which visual features like color, texture, shape, edge etc. were extracted to be used as indices [83, 84, 85, 86, 87]. However, such indexing techniques do not take into account the essential characteristic of video, viz., its temporal dimension. Recent works have used the spatio-temporal relationship among video frames by extracting motion information inherent in them [88, 89, 90, 82]. In [88], the authors use texture features extracted from temporal slices to index motion content. While patterns in spatio-temporal slices reveal camera motions (pan and zoom) and direction of motion, they do not indicate the *intensity* of motion in a video sequence. In [89], trajectory of the moving object is used as the index to motion content. In [90], Markov Random Fields is used to characterize the optical flow field of video clips. This method is computationally intensive and is, therefore, not suitable for long video clips in large video databases. In [82], the authors use block based motion vectors and principal component analysis to represent motion content, but this is done at a very coarse level. The MPEG-7 standard provides motion descriptor to describe motion activity. The extraction of this descriptor is based on aggregation of motion vectors contained in the compressed bitstream [91]. However, as in [82], motion vectors in MPEG compressed video streams are computed

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

over macroblocks of size 16×16 and hence they are only a coarse representation of the actual motion. It is desirable to consider motion features that can be determined at the pixel level in order to obtain motion information at a finer resolution.

Key frames provide an abridged representation of the original video sequence and can be used for video indexing, retrieval, browsing and summarization. Most of the existing key frame extraction approaches are based on color features. A naive approach, but effective only for low motion videos, is to choose the first frame of a shot as the key frame. A more robust approach was proposed in [24], in which they cluster the colour histograms of the frames and choose the key frames nearest to the centroids of the clusters. The optimal number of key frames is determined by a cluster validity analysis. Motion based approaches to key frame extraction include [92] that uses optical flow, while [93] utilizes the MPEG-7 motion activity descriptor as a measure of the summarizability of the video sequence. Lee and Kim describe a method to select key frames based on temporal variation of the mean intensity in a frame to minimize a distortion function [94].

In this chapter, we present a simple and efficient method to represent the motion content of video at the pixel level. Through a simple procedure that uses frame differencing, we create what we call a *Pixel Change Ratio Map (PCRM)* that indicates moving regions in a particular video sequence. Histogram and correlogram have been widely used on gray level or color images for indexing. Histogram is an approximation of the distribution of the pixel values while the correlogram indicates spatial distribution as well as pixel value distribution information as well [95]. We propose to extract the histogram and correlogram from each of the *PCRM*s as the motion features for video indexing. We illustrate the utility of the proposed motion features through applications in video retrieval, video clustering and video classification.

We also propose a new key frame extraction scheme utilizing the proposed motion cue. After compensation the global motion by adaptively filtering the pixel change map using

a modified version of the least mean square (*LMS*) algorithm, the prediction errors thus obtained are used to subsequently select the key frames. The key frames are selected so that the cumulative prediction error is partitioned into equal amounts in each segment. The entire procedure is computationally simple and flexible.

3.1 Pixel Change Ratio Map

In this section, we present the algorithm to compute the *Pixel Change Ratio Map* (*PCRM*), which characterizes motion content in a video segment. We assume that each segment consists of a single shot. However, if the segment consists of more than one shot, a shot detection algorithm can be used to locate the shot boundaries before determining the *PCRM* for each shot. The algorithm to generate the *PCRM* is motivated by the fact that the human visual system perceives motion if the intensity of motion is high and the motion continues for a reasonably long duration. By intensity of motion, we mean how fast a certain object moves, implying that a high intensity of motion results in a large change in pixel intensities over the frames. Similarly, object motion that is spread over several frames, i.e., one that has a long duration, is more perceptible than motion over a few frames.

Based on the above observations, we accumulate the changes in pixel intensity over all the frames in a video segment to generate the *PCRM* for that segment. We initialize the *PCRM*, which is of the same size as that of the frame, to all zeros. If the frame size is $M \times N$, then the *PCRM* is simply a matrix of size $M \times N$ initialized to zeros. For the current frame i , we add the absolute values of the frame differences $p_i - p_{i-1}$ and $p_{i+1} - p_i$, i.e., $DI_i = |p_i - p_{i-1}| + |p_{i+1} - p_i|$. For each pixel in this frame, if the sum DI_i is greater than a threshold, the corresponding location in the *PCRM* is incremented by 1. This procedure is carried out for all the frames in a video segment. The *PCRM* values are then divided by the number of frames and normalized to lie in $[0,1]$. Thus, it

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.1: (a) Key frame (middle frame) of a 'Football' sequence, (b) *Pixel Change Ratio Map* of the video sequence in (a)

represents the ratio of the number of pixels whose intensities have changed as a result of significant motion, where significant qualifies intensity and duration. The more intensive the motion, the more pixel changes per frame and the longer the duration of motion, the pixel changes will last for more frames. The comparison of DI_i with a threshold is simply to undo the effect of any noise associated with the camera or the decoding process when dealing with compressed video. In this work, the threshold is chosen to be 10 for all the experiments discussed and hence it is not dependent on any particular type of video. Figure 3.1(a) shows the 50th frame of a video segment from a soccer sequence consisting of 100 frames. The *PCRM* for this segment, shown as an intensity image in Figure 3.1(b), clearly indicates motion regions. Note that both the local motion (of the players) as well as the global motion (of the camera) is captured in the *PCRM*. It is the camera motion component of the total motion that causes the lines of the penalty box to show up. The brighter the pixel intensity in the *PCRM*, higher is the intensity of motion.

Figure 3.2: Histogram of *PCRM* images accumulated over all the sequences in the database.

3.2 The Motion Histogram

The representation of the *PCRM* as an intensity image enables us to consider its histogram as a reliable feature to index motion content. However, a histogram with uniform quantization of the bins will not achieve the purpose. This is because most of the values in the *PCRM* will be cluttered at low values. This is indicated in Figure 3.2 which shows the histogram of the *PCRM* images accumulated for all the 643 video segments in the database that we use in our experiments. Here, the range $[0,1]$ of the *PCRM* values are quantized uniformly into 1000 bins. Clearly, in order for the proposed motion histogram to have a high discriminative ability among different classes, it is required to requantize the bins non-uniformly such that the step size is finer where the distribution is high and coarser where the distribution is low. Thus, in Figure 3.2, we would like ‘narrower’ bins to be placed close to the origin while the bins get larger as we move away from it. Once the quantization step sizes are determined for the accumulated histogram, the same is used to requantize the histograms of individual *PCRM* images, which is indeed the proposed motion histogram for each video sequence.

Suppose there are m quantization levels and it is required to determine the step size for each of them. We obtain the cumulative histogram from the accumulated histogram

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

and traverse it, marking off points at $\frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}, 1$, where m is the number of bins. The abscissa of each of these points marks the boundary between two bins. This is a way of simply ensuring that the distribution of the *PCRM* values among the bins is uniform. Figure 3.3(a) shows the histogram of Figure 3.1(b) with uniform quantization; this histogram is then subjected to a non-uniform quantization and the result is shown in Figure 3.3(b). We see that the histogram obtained from nonuniform quantization has higher bins that are more populated than those in the histogram with uniform quantization. This is desirable since the ‘Football’ sequence of Figure 3.1 has high motion content. We note that although the non-uniform quantization in Figure 3.3 is illustrated for a single video sequence, the actual requantization is carried out on all the histograms obtained from the entire database.

The use of the accumulated histogram to determine the step sizes implies that we use all the available information in the database. However, addition of a new sequence to the database does not require the recomputation of quantization levels since we are already considering a wide variety of sequences whose properties are included in the accumulated histogram. The recomputation needs to be done only in the case where the accumulated histogram is seen to be altered significantly. The determination of the step sizes as described above can also be justified from an information theoretic point of view wherein we would like the requantized histogram to provide the maximum information about the dataset. Since uniform distribution is the one that maximizes the entropy of a source in the absence of constraints other than those imposed by laws of probability, we would like a uniform distribution of the *PCRM* values among the bins.

3.3 The Motion Correlogram

A correlogram C^k is a matrix whose element (i, j) is the probability of finding a pixel with intensity j at a distance k from a pixel with intensity i . For an $N \times N$ *PCRM*

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.3: (a) Histogram of Figure 3.1(b) before re-quantization, (b) Histogram of Figure 3.1(b) after 8-level nonuniform quantization.

image I , whose intensities are quantized into L levels, the correlogram element at (i, j) for distance k is given by

$$C^k(i, j) = P(I(p_2) = j | I(p_1) = i, d(p_1, p_2) = k) \quad (3.1)$$

where $I(p_2)$ ($I(p_1)$) is the intensity at pixel p_2 (p_1). We use the L_∞ norm as the distance measure $d(p_1, p_2)$, i.e., $d(p_1, p_2) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$, where pixel $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. The correlogram in equation (3.1) can be computed by

$$C^k(i, j) = \frac{A(i, j)}{n_i \cdot 8k} \quad (3.2)$$

where $A(i, j)$ is the cardinality of the set $\{I(p_1) = i \wedge I(p_2) = j \wedge d(p_1, p_2) = k\}$, n_i is the number of pixels of intensity i in I and $8k$ is the number of pixels in the neighborhood at distance k from p_1 . A set of correlograms is found for each distance in the set $s \in \{1, \dots, k\}$. The computational complexity of finding a correlogram for an $N \times N$ image and distance k is $O(N^2k)$. To compute an entire set of correlograms, one for every distance in $s \in \{1, \dots, k\}$, the computational complexity is $O(N^2q^2)$, where q is the cardinality of s . This can be reduced to $O(N^2q)$ using dynamic programming[95]. However, since we do not propose to use every possible distance in s , and moreover, since $q \ll N$, the computational burden is not a major concern.

A subset of the correlogram C^k is the autocorrelogram which contains only the diagonal elements of C^k . It gives the probability of finding the same intensity at a particular distance. The computations can be further simplified by considering only autocorrelograms. We choose to use autocorrelogram to index video sequences on the basis of motion content. The same quantization level on *PCRM* described in the previous section is applied for extraction of motion correlogram.

3.4 Applications of Motion Histogram and Correlogram

In this section, we illustrate three applications of the two newly proposed motion features, i.e. motion histogram and motion correlogram. We utilize the motion features for video retrieval, clustering and classification. The video database is the same as the one used in chapter 2. It consists of 643 video sequences taken from the MPEG-7 test set. The videos include sports videos (e.g., soccer, basketball, golf, cycling), news videos, sitcoms, concert videos, documentaries etc. The duration of each video segment varies from 7 to 30 seconds and there are a total of 299,110 frames. We expect various types of motion content in the different videos and therefore, they form a suitable data set in which to test the proposed motion features.

3.4.1 Video Retrieval

The objective here is to retrieve video sequences with motion content similar to that in a query video. The motion content is represented by the motion histogram and motion correlogram features separately. We consider three instances of the number of quantization levels viz., 8, 16 and 32. While the choice of these numbers is merely illustrative, it may be pointed out that they can be represented in binary format with full usage of bits and that they enable the histograms to be feature scalable, i.e, a histogram with a high

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.4: Examples of video retrieval using motion histogram (showing the key frame of each sequence).

number of quantization levels can be converted to one with a lower number by simply merging the adjacent bins. Moreover, the MPEG-7 Scalable Color Descriptor which is based on histograms also uses these quantization levels. The Minkowski distance between two histograms h_q and h_d is defined as

$$d(h_q, h_d) = \left(\sum_{m=1}^N |h_q(m) - h_d(m)|^u \right)^{1/u}, \quad (3.3)$$

where N is the number of quantization levels (or bins). In our experiment, we choose Euclidean distances ($u = 2$ in equation (3.3)) to compute the similarity between motion histograms.

Figure 3.4 shows examples of video retrieval experiments based on 8D motion histogram. The first column consists of the key frames of the query video shots. In this case, we simply consider the middle frame in the sequence as the key frame. The second, third and fourth columns show the key frames of the first, second and third retrieved video sequences. It is evident that sequences belonging to the same category are present

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

among the top returns for different types of videos. We also observe that the motion histogram is able to distinguish between large and small moving objects. For example, the soccer sequence (first row in Figure 3.4) consists of moving regions that are small while the sequence containing walking people (second row in Figure 3.4) consists of moving regions that are large.

To quantify the performance of the motion histogram for video retrieval, we consider the precision of retrieval. However, first, we would like to emphasize the difficulty of obtaining the ground truth when motion histogram is used as a feature for retrieval. Intuitively, it would seem that the ground truth could be obtained by categorizing the database according to motion content, e.g., low, mid and high motion sequences. But this is a very subjective task and it is not easy to define these categories exactly. The alternative is to categorize the database on the basis of semantics in which case construction of the ground truth is not hard. But the problem here is that semantics does not always relate to motion content in a direct manner, e.g. in a golf sequence, the swing of the golfer is high motion while the ball slowly rolling towards the hole is low motion. Even though ‘golf’ is the semantic class, the motion is not coherent throughout. However, we observe that in soccer, basketball and talking head videos, the motion content is coherent in that there is significant motion throughout the sequence, e.g., in soccer sequence, even when the ball goes out of play and there is not much motion among the players, the camera will be zooming into the area from where the ball is thrown in. Thus, we calculate precision of retrieval on two semantic classes - the first one is called ‘soccer&basketball’ and the second one is called ‘talking head’. Note that soccer and basketball sequences are categorized into one class - we could as well consider them as separate classes. Figure 3.5(a) shows the average precision for different values of top n retrievals for ‘soccer&basketball’. The average precision for a particular n is obtained by querying the database with each video sequence belonging to this class and averaging the

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

precision over all the queries. The 32-level motion histogram performs better than the 8-level and the 16-level histograms because the variation in pixel intensities in the *PCRM* image for these sequences is very large and hence, a finer quantization is required. Note that our measure of precision is stricter than the one based only on semantic ground truth because the retrieved video sequences are considered relevant only when they are not only similar in motion content to the query video sequence, but also share the same semantics. Figure 3.5(b) shows the average precision for top n retrievals for the ‘talking head’ class of videos. Here, the 8-level histogram performs better since the sequences in this class contain very low motion resulting in the intensity values of the *PCRM* image to be concentrated near zero. Since the variation of the pixel values in the *PCRM* is very small, the 32 level motion histogram may over quantize the *PCRM* values. The lower average precision in this case compared to ‘soccer&basketball’ is a consequence of our definition of precision. In the database, there are many sequences from sitcoms which have similar motion content to talking head videos, e.g. conversation between two people. Although they are similar in motion content, they are considered as false positives since they do not belong to the same semantic class. We also compare the performance of the proposed feature with motion feature (*PCA* on motion vectors field) used in [82]. To make the comparison fair, we kept 32 principle components with the largest 32 eigen values so that the dimension of the *PCA*+*MV*(32D) feature is equal to the maximum number of dimensions of the motion histogram features we used. The precision for the *PCA*+*MV*(32D) motion feature on the ‘soccer&basketball’ and ‘talking head’ video clips is illustrated by the *MV*+*PCA*(32D) curves in Figure 3.5(a) and Figure 3.5(b). Clearly, the proposed feature is superior.

In figure 3.6, we compare the motion histogram with the *TM*/*CM*/*OM* features (described in chapter 2) for the retrieval of (a) ‘soccer&basketball’ and (b) ‘talking head’ sequences. Recall that in chapter 2 the *TM*/*CM*/*OM* is shown to perform better than

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.5: Average precision of retrieval for (a) ‘soccer&basketball’ and (b) ‘talking head’ sequences using motion histogram with varying quantization levels and MV+PCA(32D) features.

the motion histogram to describe the motion content in terms of the amount of camera motion, object motion and total motion. However, for this experiment, the motion histogram shows the best performance among other features. Although the TM/CM/OM motion feature captures the amount of camera motion, object motion and total motion best, it is not the best feature that distinguishes different video class. The amount of camera motion, object motion and total motion might not be directly correlated to different video class. Different video class may have similar amount of camera motion, object motion and total motion. Since motion histogram includes not only the amount, but also the distribution of the motion content, it is shown as the best feature to distinguish different video class.

We also compare the performance of the motion histogram with the motion correlogram. The number of quantization levels for the *PCRM* is set to 32. The set of distances to compute the correlogram is $\{1, 3, 5, 7\}$. As described before, we use the motion autocorrelogram, though refer to it as motion correlogram. Figure 3.7 shows examples of video retrieval experiments based on the motion correlogram and motion histogram

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.6: Average precision of retrieval for (a) 'soccer&basketball' and (b) 'talking head' sequences using motion histogram for motion histogram(8D), MV+PCA and TM+CM+OM features.

described earlier. Figure 3.7(a) shows 3 example video retrieval results using motion correlogram, while the corresponding retrieval results with motion histogram are shown in Figure 3.7(b). It is evident that both features are effective to differentiating the sequences which belong to the same category. We also observe that the motion correlogram is able to distinguish between large and small moving objects better than motion histogram (third row in Figure 3.7(a) and (b)). Figure 3.8(a) shows the average precision for different values of top n retrievals for 'soccer&basketball'. It is clear that the proposed motion correlogram yields very high precision and outperforms the motion histogram and the MV+PCA features. The high performance of motion histogram and motion correlogram over MV features demonstrates that the proposed *PCRM* is able to represent the motion content in more detail than the MPEG motion vectors. The reason for slightly better retrieval performance of motion correlogram over motion histogram is that the motion correlogram includes spatial as well as distribution information while the histogram is only the approximation of the distribution. Generally, correlogram has better indexing capability than histogram.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.7: Examples of video retrieval with (a) motion correlogram and (b) motion histogram features (showing the key frame of each sequence).

3.4.2 Video Classification via Probabilistic Modeling

We wish to group together video sequences with similar motion content using the proposed motion histogram and motion correlogram features. However, the clusters thus formed should also be semantically meaningful. If so, the clusters can be initialized to be

Figure 3.8: Comparison of average precision of retrieval for (a) 'soccer&basketball' and (b) 'talking head' videos with different features.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

the starting point for content browsing by the user. We assume that the video sequences in the database are generated by a Gaussian Mixture Model (*GMM*). Since each sequence is represented by the motion histogram or motion correlogram (N dimensional feature vector $\mathbf{x}$), the *GMM* for the video database is denoted by

$$f(\mathbf{x}|\theta) = \sum_{k=1}^M \alpha_k f_k(\mathbf{x}|\theta_k), \quad (3.4)$$

where M is the number of mixtures, α is the mixing co-efficient and θ is the parameter set. The assumption of Gaussian Mixture Models *GMM* allows us to write $f_k(\mathbf{x}|\theta_k)$ as

$$f_k(\mathbf{x}|\theta_k) = \frac{1}{\sqrt{(2\pi)^N |\Sigma_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right\}. \quad (3.5)$$

The model is specified in terms of the parameter set $\theta_k = \{\alpha_k, \mu_k, \Sigma_k\}$, for $k = 1$ to M , where $\sum_{k=1}^M \alpha_k = 1$ and Σ_k is a $N \times N$ positive definite covariance matrix. The clustering problem is then the estimation of these parameters followed by determining those video sequences that have the most similar sets of parameters. Here, M is the number of clusters. A maximum likelihood (*ML*) approach is followed to carry out this task. If there are Q observations of the random vector $\mathbf{x}$, then the *ML* estimate of the parameters, θ_{ML_k} , is obtained by maximizing the likelihood function $f(\mathbf{x}|\theta_k)$, i.e.,

$$\theta_{ML_k} = \arg \max_{\theta_k} f(\mathbf{x}_1, \dots, \mathbf{x}_Q|\theta_k). \quad (3.6)$$

Here, $\mathbf{x}_1, \dots, \mathbf{x}_Q$ are the feature vectors corresponding to motion features of the Q video sequences in the database. We use the Expectation-Maximization (*EM*) algorithm [96] to estimate the parameter θ_k .

The parameters estimated by EM algorithm maximize the likelihood function given the observations. EM algorithm alternates between an E-step and an M-step. If we consider the observations to be incomplete and assume an unobserved random variable

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

$\omega_{i,k}$ such that $\omega_{i,k} = 1$ if the i^{th} sample is generated by the k^{th} mixture component, then the E-step of the *EM* algorithm is the computation of $\omega_{i,k}$ as

$$\omega_{i,k} = \frac{\alpha_k \mathcal{N}(\mathbf{x}_i | \mu_k, \Sigma_k)}{\sum_{j=1}^M \alpha_j \mathcal{N}(\mathbf{x}_i | \mu_j, \Sigma_j)}, \quad k = 1, \dots, M \quad (3.7)$$

for all $i = 1, \dots, Q$. The M-step results in the following update equations [97]:

$$\begin{aligned} \alpha_k &= \frac{1}{Q} \sum_{i=1}^L \omega_{i,k}, \\ \mu_k &= \frac{\sum_{i=1}^Q \omega_{i,k} \mathbf{x}_i}{Q \alpha_k}, \\ \Sigma_k &= \frac{1}{Q \alpha_k} \sum_{i=1}^Q \omega_{i,k} (\mathbf{x}_i - \mu_k)^T (\mathbf{x}_i - \mu_k) \end{aligned}$$

After the *EM* algorithm converges, the feature vector x_i is assigned to cluster j such that $\omega_{i,j} = \max_k \{\omega_{i,k}\}_{k=1}^M$.

It is important that the *EM* algorithm is initialized with parameters so that it does not get trapped in a local maxima of the likelihood function. Therefore, we utilize the K-means algorithm [98] to perform data-driven initialization instead of a random one. In the experiment, we perform K-means clustering 5 times (here the initialization is random). The K-means clustering result with the smallest *SSD* (Sum of Squared Distance) is chosen as the initialization for the *EM* algorithm. An optimal criterion for the number of clusters, M , is based on a tradeoff between the performance and number of parameters used for describing the mixture distribution. We choose the Bayes Information Criterion (*BIC*) [99] to determine the optimum number of clusters,

$$BIC = -2 \times \log Lik + q \times \log L, \quad (3.8)$$

where $\log Lik$ is the log-likelihood evaluated at the maximum likelihood estimates of the model parameters, and q is the number of parameters in the model and L is the sample size. Figure 3.9(a) shows the *BIC* for different number of clusters with motion

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.9: (a) Bayes Information Criterion (BIC) for different number of clusters with motion histogram. (b) Scatter plot of feature vectors reduced from 8D motion histogram to 2D using PCA .

histogram. We see that 7 is the optimal number of clusters as chosen by the BIC for the video database.

Although we would like to show the discriminatory capability of the motion histogram visually, it is not possible to plot the feature vectors because of their high dimensionality. However, we use the Principal Component Analysis to reduce the dimension of the 8D motion histogram feature to two and plot them in Figure 3.9(b). We observe that the features are separated into two clusters. One of the clusters represents ‘high’ motion and the other represents ‘low’ motion as perceived by the human visual system. Next, we apply the EM based clustering algorithm on the 643 video sequences in the database to cluster them into 7 groups using the motion histogram. Figure 3.10 shows the key frames from 3 representative video sequences for each of the clusters. It is interesting to note that each cluster represents a particular type of motion. For example, cluster 2 represents sequences in which the camera is stationary while there is little object motion. Such sequences include news videos, interviews etc. However, in cluster 3, there is indeed a smooth camera motion although motion of objects is very little or non-existent, for example as in the camera pan of a landscape. The descriptions of motion in the video sequences in each cluster and their examples are listed in table 3.1.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

The use of correlogram for video clustering is hindered by its high dimensionality. Since we use a 32D correlogram (eight dimensions for each of four distances), the *EM* based clustering algorithm tends to get trapped in a local maximum. As a result, the clustering algorithm becomes highly dependent on the initialization parameters and the *BIC* no longer is an effective way to decide on the number of clusters. In the video retrieval experiment, motion correlogram is shown to be able to index the video content better than the motion histogram. However, in the case of clustering, the generality of features is more desirable. Hence, the motion histogram is more suitable for video clustering. However, in order to arrive at a rough estimate of the performance of the correlogram for clustering, we initialize the process using the clustering result obtained from the motion histogram and then cluster with motion correlogram by *EM* algorithm. It is observed that there is no significant difference from Figure 3.10 in the final clusters thus obtained.

Table 3.1: Description of clusters using motion histogram.

Cluster	Description
Cluster 1	Irregular camera motion, large object motion with small moving object size (e.g., soccer(long shot), basketball(long shot), etc.)
Cluster 2	Still camera and little object motion (e.g., talking head, dialogue, interview, etc.)
Cluster 3	Smooth camera motion with little or no object motion (e.g., scenery, etc.)
Cluster 4	Little camera motion and little object motion (e.g., outdoor interview, out door news shots, etc.)
Cluster 5	Little camera motion and large object motion (marathon, racing cars, animal shows, etc.)
Cluster 6	Irregular camera motion and large moving object (e.g., soccer(close-up), basketball(close-up), cycling, etc.)
Cluster 7	Little or no motion (scenery, etc)

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.10: Video clustering using 8-level motion histogram (the key frames are shown).

3.4.3 Video Classification via Pattern Recognition Techniques

In this section, we classify video sequences into predefined categories according to their motion content with motion histogram and motion correlogram features respectively. We choose two categories - ‘high motion’ and ‘low motion’. As before, we use the 643 sequences in the database and label them as belonging to one of the two classes. There are 337 sequences labelled as ‘high motion’ and the rest 306 are labelled as ‘low motion’. Figure 3.11(a) shows some of the representative video sequences from the two classes: the top row shows sequences containing high motion and the bottom row those containing low motion. Support Vector Machines (*SVM*) [100] is used for classification because of their discriminating power and simplicity. In this case, the *SVM* performs a binary classification. Radial basis functions are chosen as the kernel for training.

In order to train the *SVM*, we randomly pick n samples from the database. We consider the effect of n on the classification rate by performing the training/classification 30 times for each n and then finding the average of the classification rate. The performance of the classifier for various values of n are shown in Figure 3.11(b). We note that the classifier needs very less training samples to converge. In our experiment, only 20 samples are enough to train the *SVM*, which is just 3% of the entire data. The small size of the training samples indicates that the motion histogram and motion correlogram are good features to separate the two classes. Considering that labelling of the video clips into ‘high motion’ and ‘low motion’ is a highly subjective task, the high classification rate is encouraging. Comparing the *PCRM* based features with the *PCA*+*MV*(32D) features proposed in [82], we find the former outperforms that latter by a significant margin as shown in Figure 3.11(b). Between motion histogram and motion correlogram, we find the motion correlogram performs slightly better than the motion histogram feature when the training sample are above 5% of the total data. Motion correlogram has poorer performance than motion histogram for small sample size because of its relative high

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.11: (a) Examples of video sequences (key frames) from two classes - 'high motion' (*Top Row*) and 'low motion' (*Bottom Row*), (b) Average classification rate with different number of training samples.

dimensionality (4 times the dimension of motion histogram). This result demonstrates that the proposed motion representation (*PCRM*) is more effective for video motion content indexing than motion vectors.

3.5 Key Frame Extraction

Key frames provide an abridged representation of the original video sequence and has a wide range of applications such as video indexing, retrieval, browsing and summarization. In this section, we introduce a new motion feature called the *Pixel Change Map (PCM)*, which is used to develop an adaptive algorithm for key frame extraction from compressed video. The proposed methods also includes compensation of camera motion inherent in most videos. To process the video directly in compressed domain not only offers

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

savings in computational resources but also provides additional information like motion vector to be utilized for global motion compensation. The motivation for global motion compensation is justified as follows. Large changes in pixel intensities between adjacent frames indicate a potential candidate for key frame. However, even a smooth motion of the camera can result in such large changes, especially in textured regions, although there is no significant change in the content and appearance of the scene. This results in redundant detection of key frames. Hence, there is a need to undo the effect of camera motion. At the same time, an irregular global motion component will indeed translate into large pixel changes pointing towards the existence of potentially valid key frames. Our method starts with computing a simple *pixel change map (PCM)* between adjacent frames. The key frame algorithm involves two passes over the *PCM*. The first pass is an adaptive filtering of the *PCM* that enables the prediction errors to be used as a cue for key frame extraction. The second pass involves the actual selection of the key frames. The proposed key frame extraction algorithm consists of 3 components: (1) computing Pixel Change Maps (*PCM*), (2) camera motion compensation by adaptive filtering and (3) key frame selection.

3.5.1 Pixel Change Map

The objective of the *pixel change map (PCM)* is to capture the presence of additional information vis-a-vis the current frame. This additional information could be due to the presence of a shot boundary or due to changes in the foreground and/or background. If the current frame occurs between two shot boundaries, it is similar to its neighboring frames and little additional information is contained in it. On the other hand, changes in background/foreground result in large changes in pixel values indicating significant additional information. Thus, the *PCM* is obtained by thresholding the difference between the current frame and the adjacent frames. However, we would like to achieve this in

the *compressed domain*. Hence, we use DC images to compute the *PCM* instead of fully decoding the frames. The DC images for *I* and *P* frames are extracted using the method in [101]. The DC images are spatially reduced versions of the original frames. They are less prone to the noise because the DC images are in essence the result of applying a low-pass filter on the original image. Note that although *PCM* is very similar to *PCRM* but there is one important difference between *PCM* and *PCRM*: *PCRM* is shot-based while *PCM* is frame-based.

3.5.2 Global Motion Compensation by Adaptive Filter

As mentioned earlier, even a little camera motion can result in large pixel changes if the scene is textured. However, such pixel changes are undesirable because the subsequent key frame extraction algorithm will be falsely triggered at these points. Hence, there is a need to compensate for camera motion. While there are several methods that determine global motion which can be used to warp the frame using estimated motion parameters, there are shortcomings to this approach. In the context of compressed domain processing, most global motion estimation approaches relies on motion vectors. Therefore, the motion vector information must be readily available from the video stream. However, not all video streams have motion vector information. For some video formats such as avi and motion jpeg, there is no motion estimation at all and for most video codecs the motion estimation mode can be disabled or carried out at different level of accuracy. Therefore, the quality of the motion vectors is codec dependent and may not be accurate for fast coded videos¹. Moreover, the warping of the frames using estimated motion parameters (e.g. using an affine model) requires significant computation, e.g., for each $M \times N$ frame, the computation for warping one frame using bilinear interpolation requires $4 \times M \times N$ multiplications and $4 \times M \times N$ additions.

¹In chapter 2, the video sequences are encoded using a high quality MPEG encoding software (TM-PEG) with the highest accurate motion estimation. It takes significant time to encode those videos.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Based on the observation that camera motion always lasts for a few frames and that the camera motion is much smoother than object motion, we ascertain that the pixel changes due to camera motion will have a strong local correlation. This allows us to model the pixel changes as an auto-regressive (*AR*) process

$$c(n) = \sum_{i=1}^h w(n-i) \cdot c(n-i), \quad (3.9)$$

where $c(n)$ is the pixel change at location n (assuming lexicographic ordering of the frame), w is the coefficient and h is number of the taps of the *AR* model. The errors from the local regression with the *AR* model are deemed as the true pixel changes, i.e., additional information carried by each frame and used as input for the algorithm described in section 3.5.3 to extract the key frames. It is these prediction errors that serve as a cue to the correct extraction of key frames in that large errors occur due to the difficulty in predicting the next frame, which in turn is attributed to significant motion.

We use a modified version of the Least Mean Squares (*LMS*) algorithm to determine the *AR* parameters[102]. The reason for modification is illustrated by the following scenario. Consider a shot boundary. The pixel changes corresponding to a shot boundary will be much higher relative to other frames. This will result in a large error in prediction. Note that the large error is *useful* for key frame selection. However, it will also update the *AR* coefficients drastically causing a drift in the weights. Hence we modify the *LMS* algorithm to the extent that the weights are updated only if the error is less than a threshold. We call this process the *Switched – LMS(SLMS)* algorithm.

Consider an *AR* process with input $x(n) = \sin(\frac{n\pi}{8}) + \sigma(n)$, the desired signal (the pixel change sequence in this case) $d(n) = 2 \cos(\frac{n\pi}{8}) + T \cdot \delta(\text{mod}(T, 100))$, where $\sigma(n)$ is the white Gaussian noise process with variance equal to 0.1 and $T \cdot \delta(\text{mod}(T, 100))$ simulates (periodic) shot changes. Figure 3.12 (a) shows the simulated random process $d(n)$ and Figures 3.12(b) and (c) show the convergence of the adaptive weights for *LMS* and

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.12: (a) The simulated pixel change sequence. Convergence of weights using (b) *LMS* algorithm and (c) *SLMS* algorithm.

SLMS algorithms, respectively. Clearly, the *SLMS* converges more smoothly and faster than the *LMS* algorithm. The guarded weight updates in *SLMS* prevent the weights to be distracted by the abrupt shot changes and result in smooth estimates of the weights, which corresponds to the effect of camera motion on pixel changes. This is desirable because the camera generally follows a smooth movement during the production of the video and it is unlikely that the resulted pixel changes vary drastically.

3.5.3 Key Frame Selection

As we have seen in section 3.5.2, modelling the *PCM* as an *AR* process yields prediction errors, PCM_e , that captures the additional information at a particular frame. To this extent, the total amount of information contained in an entire video sequence is deemed to be captured by the accumulation of the PCM_e s for the sequence. Note that the *PCM* values for the first frame are set to 1, the *AR* coefficients are initialized to 0 and, therefore, the prediction errors PCM_e are equal to 1 for the first frame. Given the total information in a sequence (corresponding to accumulated PCM_e s), the objective is to partition the sequence with key frames such that each segment contains equal amount of information. Hence the key frames act as pegs in the cumulative PCM_e s such that their locations force equal information in every interval.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Let PCM_e^i be the prediction error of the pixel change map at frame i and N be the number of key frames to be extracted. We compute the cumulative PCM_e for the sequence as $CPCM_e = \sum_{j=1}^M PCM_e^j$, where M is the number of frames. Next, we traverse the cumulative PCM_e marking off points $l = \frac{l}{N} - \frac{1}{2N}$, $l = 1, \dots, N$. The abscissa of each of these points marks the location of the key frame. Note that this method does not need any information about the location of shot boundaries. Moreover, a single pass through the cumulative PCM_e is enough to select the location of the key frames.

Figure 3.13 illustrates the proposed key frame extraction on synthetic data. From the prediction error of pixel change in Figure 3.13 (a), we see that the data has two distinct segments: the PCM_e per frame is 0.5 in the first half while it is 1.5 in the second half. Suppose 4 key frames are to be extracted. The cumulative PCM_e , shown in Fig 3.13 (b) is marked off at $1/8, 3/8, 5/8, 7/8$ and the corresponding abscissa that locate the key frames are extracted at $1/4, 7/12, 3/4$ and $11/12$. The PCM_e in the second half segment is 3 times that in the first half segment. Thus, the number of key frames extracted from the former should be three times that of the key frames extracted from the latter. The proposed method has successfully extracted one key frame and three key frames from the first and second half segments, respectively. This is consistent with our assertion that larger prediction errors indicate the presence of more key frames. Note that no information about shot boundaries was used to locate the key frames.

3.5.4 Experimental Results

In this section, we describe two sets of experiments. The first set evaluates the effectiveness of the *SLMS* adaptive filtering algorithm to compensate for the effect of camera motion on pixel changes and to generate perceptually consistent prediction errors. The second set evaluates the proposed key frame extraction algorithm.

Filtering for Global Motion Compensation: We choose two sequences to illustrate adaptive filtering on the *PCMs*. The first sequence contains only smooth camera

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.13: (a) Prediction error in pixel change, PCM_e , per frame (synthetic data), (b) Cumulative PCM_e with location of extracted key frames.

motion consisting of a pan from left to right. An example frame from the sequence is shown in Figure 3.14(a). The actual pixel changes, predicted pixel changes, and the prediction error curves are shown in Figure 3.14(b). We see that the error drops very quickly after a few frames and remains so till the end of the shot. Since the error is low, we conclude that there is no significant activity and hence the number of key frames extracted should be less. The second sequence consists of the camera tracking a soccer player on the field. Since the player moves quite irregularly, the motion of the camera is not smooth and the values of the pixels change rapidly. An example frame for the video clip is shown in Figure 3.14 (c). The actual pixel changes, predicted pixel changes, and the prediction error curves are shown in Figure 3.14(d). The jerky motion of the camera results in little local correlation in the PCM and hence a higher prediction error. However, such high errors are desirable for key frame selection. The content in this video interval is of high motion and carries more information than still or smooth camera motion video intervals. Therefore, more emphasis should be put on such intervals for key frame selection, i.e., more key frames should be selected.

Key Frame Extraction: We consider a sequence from the MPEG-7 test set which consists of four shots in which the first two shots track a group of people walking, the third shot shows a person answering questions while the fourth shot is that of an anchor

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.14: Frame from video clip with (a) smooth and (c) irregular camera motion. (b) and (d) original, predicted and prediction errors of pixel changes for video clip of (a) and (c), respectively (Horizontal axis shows the frame number.)

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.15: DC images of (a) 4 keyframes (frame #1, #67, #139, #374) and (b) 5 keyframes (frame #1, #45, #96, #192, #374) from an MPEG-7 test sequence.

person. The 4 keyframes extracted from this sequence are shown in Figure 3.15(a) (as DC images) where each key frame is extracted from each shot. Note, however, that our method does not presume the presence of a shot boundary or use any information about the location of shot boundaries. Figure 3.15(b) shows 5 key frames extracted from the same sequence. The additional key frame is that of a turning person; thus, the key frames are essentially able to capture the important events in the sequence.

Next, we compare the proposed algorithm with two other methods. The first one (abbreviated *CIK*) selects key frames at constant time intervals and the second (abbreviated *CTK*) is based on clustering the colour histograms of the frames so that the frame closest to the centroid of the cluster is chosen as the key frame. Due to its subjective nature of the key frame selection problem, it is very hard to compare different approaches quantitatively. Most approaches use subjective evaluations [103, 104] for the measurement of the key frame extraction performance, e.g. subjective score [103], visual effect [104] etc. Another popular performance measurement for key frame extraction is to see whether the density of selected key frames is correlated to the motion content of the video [103, 105, 44]. Intuitively, more key frames should be selected from high motion

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.16: DC images of key frames extracted by (a) *CTK* method (frame #37, #112, #211, #256) and (b) the proposed method (frame #106, #214, #241, #277).

segment. In this experiment, we use the clustering objective function as the performance measure for key frame selection. The performance of each method is compared by the average histogram error defined as $H_{err} = \frac{1}{N} \sum_{i=1}^N \min_{j \in K} (dist(H_i, H_j))$, where H_i is the histogram for frame i , N is the total number of frames, K is the set of key frames and $dist$ is the distance function between two histograms chosen as the summation of the absolute difference between the two histograms. The H_{err} represent the summation of deviation of all the frames from their representative key frames. A good set of key frames should be able to closely represent all the frames in the video, which will result in a small H_{err} . Table 3.2 shows the average histogram errors on 7 video sequences each containing 300 frames. 4 key frames are selected. The proposed method outperforms *CIK*. Although *CTK* is shown to achieve the lowest average histogram error, it does not take temporal information into consideration leading to incorrect selection of key frames from a perceptual point of view. This is illustrated in Figure 3.16(a) where 2 key frames are extracted from the first court-view shot, but none from the second court-view shot. However, the proposed method exhibits proper evolution of the video content as shown in Figure 3.16(b). The clustering method is also much more computationally intensive than the proposed method.

Table 3.2: Average histogram errors for *CIK*, *CTK* and the proposed method.

Videos	CIK	CTK	Proposed
Basketball 1	0.045	0.036	0.038
Basketball 2	0.082	0.058	0.071
News 1	0.087	0.058	0.067
Soccer 1	0.078	0.045	0.070
Soccer 2	0.088	0.053	0.075
TV 1	0.027	0.015	0.025
TV 2	0.031	0.020	0.023

In the third experiment, we perform the proposed key frame selection method on 7 long video clips. The test video clips are shown in table 3.3. Figure 3.17 shows the key frames selected by the proposed method compared with *CIK*. In each part, the top line of dots show the key frames selected by the proposed method and the bottom line of dots show the key frames selected by the *CIK* method. We can see that the proposed method selects key frames more effectively. More key frames are selected during high pixel change interval, viz., high motion intervals. For example, we notice regular gaps in the selected key frames by the proposed method in Figure 3.17(b) for a ‘news’ video. Actually, the gap interval corresponds the anchor person scenes, which are almost no content change and few key frames are selected during those intervals. The difference of proposed and *CIK* methods between ‘News’ and ‘TV’ are more obvious than the difference between ‘Basketball’ and ‘Soccer’. This is because that ‘TV’ and ‘News’ videos have varying pace, while ‘soccer’ and ‘basketball’ videos have almost constant pace.

3.6 Conclusion

In this chapter, we have introduced these new motion features useful for video indexing. We propose the formation of a *Pixel Change Ratio Map* based on motion content in a video sequence and extract the motion histogram and motion correlogram from it. The bins in the motion histogram and motion correlogram are adaptively quantized so

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Figure 3.17: Key frame selection result: (a) 'News' video, (b) part of the 'News' video, (c) 'Basketball' video 1, (d) 'Basketball' video 2, (e) 'Soccer' video 1, (f) 'Soccer' video 2, (g) 'TV program' video 1, (h) 'TV program' video 2.

CHAPTER 3. NEW MOTION FEATURES AND THEIR APPLICATIONS

Table 3.3: Test video clips and number of extracted key frames.

Videos	Duration (min:sec)	# of key frames
News 1	27:58	200
Soccer 1	9:48	60
Soccer 2	11:4	60
Basketball 1	10:00	60
Basketball 2	10:49	60
TV 1	12:25	60
TV 2	13:26	60

as to have a high discriminating power among classes with different motion content. We illustrate the efficacy of the proposed motion features through applications in video retrieval, clustering and classification. Not only do they retrieve sequences having similar motion as the query sequence, but they are also able to provide an indication of the size of the moving objects. The clusters formed by using the motion features are very similar in motion content to what is perceived by the human visual system. Classification of video sequences using the proposed motion histogram and motion correlogram results in a high classification rate. Moreover, there is need for only a few training samples indicating that it has a high discriminating power.

With the proposed *Pixel Change Map*, we present a novel approach for key frame extraction for compressed videos. The total information of the video is represented by accumulation of prediction errors of *PCMs*. These errors are obtained after adaptive filtering of the *PCM* to compensate for the effect of global motion. The prediction errors are then used as a cue for key frame extraction. The key frames are selected such that the video is partitioned into segments with equal amount of prediction error of pixel changes. All the computation is carried out with DC images extracted from compressed video directly. The proposed method is computationally simple and flexible. It can rapidly generate key frame based video summary of any desired length. The experimental results and comparison with other methods demonstrate those advantages of the proposed method.

Chapter 4

Automatic Motion Trajectory Extraction

Motion trajectory is a useful descriptor for video motion content indexing. It records the successive positions of moving object. The specification of object movement using a series of points is called *trajectory*. Since it is relevant to human's perception of object motion at high level, it has a wide range of applications, e.g. motion based query and video hyperlinking [106]. Motion trajectory provides direct and independent access to positions, speeds and accelerations, in each spatial dimension (x,y,z). Many high-level queries can be carried out with motion trajectory. For example, 'retrieve objects passing near a given area' (useful in indoor surveillance context) is simply a test on the key point spatial position value; 'retrieve object moving faster than a give speed' (useful for road monitoring) is a comparison between the speed values, which are straightforward to be computed from the key points position and time. Another example application is video hyperlinking: link some predefined objects in videos to other multimedia contents, to enable rich interactive viewing. Using several trajectory to represent the evolution of a few contour points of an object in time is very efficient to achieve this. Because of its rich indexing capability, motion trajectory has been adopted into the MPEG-7 standard as a motion descriptor. However, it is a challenging task to extract the motion trajectories from videos. In this chapter, we describe an approach for automatic motion trajectory

extraction in compressed sports video. Our method is able to extract arbitrary number of the motion trajectories in both stationary and moving camera for compressed video and it can be used to generate MPEG-7 *Motion Trajectory* Descriptors.

4.1 Tracking and Trajectory Extraction

The problems of trajectory extraction and object tracking are closely related. However, the objectives of extracting the trajectory and of tracking moving objects are different, although one may argue that tracking a particular object implies availability of its trajectory. The goal of tracking is to follow video objects in the scene and to update their 2D shapes from frame to frame [107]. Here, the object to be tracked is predefined by segmenting a frame of the image sequence. Most successful tracking algorithms are either region based [108, 109] or contour based [110]. [111] gives a review of the tracking techniques. In the former, tracking relies on information provided by an entire region while the latter tracks only the contour of an object. Trajectory extraction, on the other hand, deals with determining the spatiotemporal positions of one or more moving objects in a sequence in terms of one representative point of the object(s). Unlike in tracking, where some property of a bounding box is searched for in subsequent frames, we view trajectory extraction as a process of assigning moving blobs to a dynamic list of trajectories that evolves as the video progresses. While object tracking is an end in itself, the trajectory extracted from a video sequence is used for content retrieval, video classification, video hyperlinking for rich interactive viewing etc [106]. This is possible through the high level descriptors which form part of the MPEG-7 standard.

Trajectory extraction in sports video is a challenging problem because of the presence of global motion, multitude of trajectories and occlusion. In [112], object motion and trajectory are used for soccer event detection where the trajectory is marked manually. In [113], the authors track the referee and players in a soccer video using the colour of

their shirts; however, the position of the object to be tracked should be manually entered in the first frame in the form of a bounding box. Pingali et al. develop a real time tracking system for a tennis match [114]. They use two cameras, each camera capturing half of the court, and tracking a predefined player. Their algorithm tracks local features, like maximum curvature points, in a colour segmented region. Gueziec [115] develops a system for tracking the pitch in a baseball video. However, they do not consider the case of global motion since the camera is stationary. In [116], the authors use information from manually entered trajectories of the players to infer some semantic events in the game. In [117], the authors developed algorithms for segmentation and analysis of structures in soccer video.

The proposed method works for both stationary and moving camera and can extract one or more trajectories of the multiple number of moving objects. Shot segmentation and classification is used to locate the meaningful segment in which trajectory extraction is applied. The central idea for trajectory extraction is to dynamically initialize motion trajectory and evolve the existing trajectory based on the adaptive trajectory description and detected motion blobs.

4.2 System Overview

We propose a method for automatic extraction of trajectory in sports video. As mentioned earlier, our method works for both stationary and moving cameras. Furthermore, it can extract one or more trajectories of the multiple number of moving objects. The proposed method, illustrated in Figure 4.1, consists of six major steps. In the first step, the video is segmented into shot units using a *shot detection* algorithm. We use the algorithm developed in chapter 2, which works directly on MPEG compressed video. Our shot boundary detection algorithm utilize the macroblocks' encoding types to calculate the similarity between neighboring frames (chapter 2). The next step is to *classify the*

Figure 4.1: Overview of the proposed trajectory extraction approach

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

shots into long shots and close-up shots. A long shot includes relevant motion information which is useful for extraction of trajectories. Cameras are usually set to a long shot when the game is ‘in play’. A close-up shot involves a particular referee or player in the midst of a special event, such as showing a red card in soccer or immediately after scoring a point in badminton. Close-up shots do not contain any information related to the trajectory of the object. Hence, we process only the wide-angle shots for trajectory extraction. We develop a new *global motion estimation* algorithm for global motion compensation and subsequently warp each frame to its previous frame. This completes the pre-processing for the actual extraction of the trajectory. The trajectory extraction consists of *detecting motion blobs* followed by an *evolution of the trajectories* of the individual blobs. A further *refinement* of the trajectory is carried out as a post processing step. Statistical hypothesis test is applied within the Block Rejection Map (explained in section 2.4), which is the by-product of the global motion estimation, to detect the motion blobs. The trajectory evolution is the process in which the newly detected blobs are either appended to an existing trajectory or are considered as the beginning of a new trajectory based on its distance to the Adaptive Trajectory Description (*ATD*). The *ATD* is updated when new blobs are appended to the trajectory.

4.3 Shot Classification

The first step in the proposed trajectory extraction method involves shot boundary detection in which we use the algorithm described in chapter 2. Next, the shots are to be classified into either long shots or close-ups. As noted in the previous section, information about the trajectory is contained in the long shots rather than in the close-ups. In [113], the authors propose a shot classification method to classify soccer video based on the amount of green colour (representing the turf) within a frame. However, this approach

is highly domain specific. The shot classification method that we propose is applicable to all types of sports videos, both outdoor or indoor sports.

4.3.1 Feature Extraction

Our shot classification approach is based on the observation that the close-up shots will contain at least one homogeneous region that is neither too large nor too small. In the close-up shots, the camera zooms into the player. Therefore, the homogenous regions of the player (i.e. face or body) are of medium size and take up a reasonable portion of the frame area. On the other hand, in a long shot the size of homogeneous regions will be either large or small. For example, a long shot in a soccer game will have a large homogeneous region for the soccer pitch and small regions for the individual players. Similarly, in a badminton game the camera is set to take long shots while the game is ‘in play’; close-ups occur when a player wins a point. We use texture-based features as the discriminant to characterize the homogeneous regions within a frame. The co-occurrence matrix is a classical texture descriptor that represents the spatial distribution of gray levels within a local area [118]. We choose the following three statistical features of the co-occurrence matrix:

Co-occurrence Contrast:

$$c1 = \sum_i \sum_j (i - j)^2 p_{ij} \quad (4.1)$$

Co-occurrence Energy:

$$c2 = \sum_i \sum_j p_{ij}^2 \quad (4.2)$$

Co-occurrence Entropy:

$$c3 = - \sum_i \sum_j p_{ij} \log p_{ij} \quad (4.3)$$

where p_{ij} is the $(i, j)^{th}$ entry in the co-occurrence matrix. In order to reduce the variance of the intensity levels in a frame, we quantize them to 64 levels. Figure 4.2 (a) shows

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

the distribution of the extracted features from 150 key frames of three sports videos viz., basketball, soccer and badminton (the first two are part of the MPEG-7 data set). The separability of the features for long shots and close-up shots are clearly seen in Figures 4.2 (b) and (c), which are the 2D feature spaces containing energy and contrast in the first instance and, entropy and contrast in the second instance.

We demonstrate another set of texture based features which are useful for differentiating between long shots and close-up shots. This is based on the run length matrix in which the $(i, j)^{th}$ element represents the frequency that j pixels with gray level i continue in the direction q where $q = 0^\circ, 45^\circ, 90^\circ$ or 135° [119]. Three features are calculated from the run length matrix as follows:

Long Run Emphasis:

$$r1 = \frac{\sum_i \sum_j j^2 p_{ij}}{\sum_i \sum_j p_{ij}} \quad (4.4)$$

Short Run Emphasis:

$$r2 = \frac{\sum_i \sum_j \frac{p_{ij}}{j^2}}{\sum_i \sum_j p_{ij}} \quad (4.5)$$

Run Length Entropy:

$$r3 = \frac{\sum_i \sum_j (-p_{ij}) \log p_{ij}}{\sum_i \sum_j p_{ij}} \quad (4.6)$$

As before, we plot the three features in 3D space and illustrate the separability of the features in 2D space for long shots and close-ups for the same 150 frames as those used to extract co-occurrence features (see Figures 4.2(d), (e) and (f)).

4.3.2 Support Vector Machine

Support vector machines are core machine learning technology. They have both strong theoretical and excellent empirical success. They have been applied to tasks such as handwritten digit recognition [120], object recognition [121], and text classification [122]. We use Support Vector Machines (*SVM*) as the classifier because of its discriminating

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.2: (a) Distribution of co-occurrence matrix based texture features extracted from 150 frames of three sports videos, (b) 2D plot of Energy v/s Contrast of (a) and (c) 2D plot of Entropy v/s Contrast of (a); (d) Distribution of run-length matrix based texture features, (e) 2D plot of Run Length Entropy v/s Long Run Emphasis of (d) and (f) 2D plot of Run Length Entropy and Shot Run Emphasis of (d).

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

power and simplicity. In our classification task, the *SVM* performs a binary classification. We are given training data $\{x_1 \cdots x_n\}$ that are vectors in some space $x \subseteq R^d$. We are also given the labels $\{y_1 \cdots y_n\}$, where $y \subseteq \{-1, +1\}$. In the simplest form, *SVMs* are hyperplanes that separate the training data by a maximal margin. All vectors lying on one side of the hyperplane are labelled as -1, and all the other vectors lying on the other side are labelled as +1. The training instances that lie closest to the hyperplane are called support vectors. More generally, *SVMs* allow one to project the original training data in space X to a higher dimensional feature space F via a Mercer kernel operator K . In other words, we consider another set of classifier of the form:

$$f(x) = \sum_{i=1}^n \alpha_i K(x_i, x). \quad (4.7)$$

When $f(x) \geq 0$, we classify x as +1, otherwise we classify x as -1.

When K satisfies Mercer's condition [123], we can write: $K(u, v) = \phi(u) \cdot \phi(v)$, where $\phi : X \rightarrow F$ and “ $\cdot$ ” denotes an inner product. We can then write f as:

$$f(x) = \hat{w} \cdot \phi(x), \quad \text{where } \hat{w} = \sum_{i=1}^n \alpha_i \phi(x_i). \quad (4.8)$$

Thus, by using K we are implicitly projecting the training data into a different (often higher dimensional) feature space F . The *SVM* then computes the α_i s that correspond to the maximal margin hyperplane in F . By choosing different kernel functions we can implicitly project the training data from X into spaces F for which hyperplanes in F corresponds to very complex decision boundaries in the original space X . Two commonly used kernels are the polynomial function kernel $K(u, v) = (u \cdot v + 1)^p$ and radial basis function kernel $K(u, v) = e^{-\gamma(u-v) \cdot (u-v)}$. The former function kernel introduces polynomial boundaries of degree p in the original space X . The second function kernel introduce boundaries by placing weighted Gaussians upon key training instances. We choose the radial basis kernel function for training and classification.

Figure 4.3: The transformation matrix between frames F_n and F_1 given by $M_1 \times M_2 \times \dots \times M_{n-1}$

4.4 Global Motion Estimation

Most sports videos, especially outdoor sports, contain significant global motion associated with movement of the camera. In order to extract the motion trajectory of objects while the camera is in motion, we need to estimate the global motion and compensate for it. Model based motion estimation has been reported extensively in literature [124]. In [125], the affine parameter estimation problem is formulated as a nonlinear minimization problem which is solved using an iterative algorithm such as the Levenberg-Marquart algorithm. This method is semi-automatic because the user needs to identify at least 3 corresponding feature points in two frames. The author suggests the use of the phase correlation algorithm to fully automate the process of parameter estimation. Phase correlation uses the shifting property of the Fourier transform.

In [125], the affine parameters are chosen to minimize the sum of squared difference between all common pixels in both the images. Here, we adopt the camera motion estimation algorithm presented in chapter 2. The algorithm uses 6 parameter affine motion model to fit the block based motion vector between two frames. These motion vectors are available from the MPEG compressed video stream. We first get a frame by frame optical flow field. Then a six affine parameter motion model is fitted to the frame by frame optical flow field with robust outlier rejection.

The motion vector affine parameters obtained from global motion estimation algo-

rithm are converted back to the classical affine parameters through equation (4.9):

$$\begin{pmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \end{pmatrix} = \begin{pmatrix} 1 + p_1 & p_2 & p_3 \\ p_4 & 1 + p_5 & p_6 \end{pmatrix} \quad (4.9)$$

where p_i s are the estimated affine parameters. The transformation matrix $M(a_m, a_{m+n})$ between two non-consecutive frames a_m and a_{m+n} in the sequence is computed as

$$M(a_m, a_{m+n}) = \prod_{i=m}^{i=m+n} M_i \quad (4.10)$$

where the M_i 's are the affine transformation matrix between frame i and $i - 1$. This is illustrated in Figure 4.3.

The estimated optical flow field for zooming and panning in the table tennis sequence is shown in Figure 4.4. Figure 4.4 (a) and (b) show the motion vector fields constructed from the MPEG motion vectors and the estimated camera motion field using our algorithm for zooming. Figure 4.4 (c) and (d) shows the motion vector field and estimated camera motion field for panning.

4.5 Trajectory Extraction

Having determined the frames that contain long shots in a video sequence and estimated the global motion between the frames, we now describe our algorithm for extracting the trajectory. It consists of three steps: (1) motion blobs detection, (2) trajectory evolution and (3) trajectory refinement. The algorithm detects the motion blobs first. Then the blobs are used to initialize and evolve the trajectories based on the Adaptive Trajectory Description (*ATD*). Whenever the players move, the trajectories will evolve with the newly detected motion blobs. However, if the players keep still, the trajectories remain unchanged. The trajectory detection based on motion avoids the computational expensive search for finding the object and is much more efficient. In the following sections, the above three stages of trajectory extraction are described in details.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.4: Motion vector field and estimated camera motion optical flow field for zooming and panning of table tennis shot (a) motion vector field for zooming and (b) estimated camera motion field (zooming), (c) motion vector field for panning and (d) estimated camera motion field (panning)

4.5.1 Motion Blob Detection

Motion blob detection is to find moving regions in the frame. We use a statistical model based approach to blob detection which is motivated by a similar model used by Aach *et al.* for detecting changes in video [126].

The observed background image pixel is modelled to be affected by the addition of Gaussian noise, with zero mean and standard deviation σ , i.e.,

$$\hat{I}(i, j) = I(i, j) + \eta \quad (4.11)$$

where $I(i, j)$ is the intensity at pixel position (i, j) , $\hat{I}(i, j)$ is the observed intensity and η is the Gaussian noise.

Given the hypothesis H_0 that the observed pixel belongs to the background, the temporal adjacent frame difference $d_n(i, j) = \hat{I}_n(i, j) - \hat{I}_{n-1}^{warp}(i, j)$ is a random variable with Gaussian distribution $N(0, \sqrt{2}\sigma)$, so that

$$\begin{aligned} d_n(i, j) &= \hat{I}_n(i, j) - \hat{I}_{n-1}^{warp}(i, j) \\ &= I(i, j) + x_n - (I(i, j) + x_{n-1}) \\ &= x_n - x_{n-1} \end{aligned} \quad (4.12)$$

Thus, the following conditional *pdf* holds:

$$f(d_n(i, j)|H_0) = \frac{1}{2\sigma\sqrt{\pi}} \exp\left\{-\frac{d_n^2(i, j)}{4\sigma^2}\right\} \quad (4.13)$$

For a more robust detection, we consider a group of pixels belonging to the spatio-temporal window $\Omega \equiv w \times w \times l$, where w is the spatial width and l is the temporal length of the window. We define Δ as

$$\Delta^2(i, j) = \sum_{(k, l, n) \in \Omega} d_n^2(k, l), \quad (4.14)$$

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

where Ω is the spatio-temporal window centered on (i, j, n) and containing w^2l pixels. If H_0 is valid for all the pixels in the window Ω , the corresponding random variable Δ is the summation of w^2l random variables whose conditional density with respect to hypothesis H_0 follows a χ^2 distribution[127],

$$f\left(\frac{\Delta}{\sqrt{2}\sigma} | H_0\right) = \chi^2(w^2l - 1) \quad (4.15)$$

The moving pixel is detected by evaluating the probability

$$P(\Delta^2 \leq \Delta^2(i, j) | H_0) = \frac{\Gamma(w^2l/2, \frac{\Delta^2(i, j)}{2\sigma^2})}{\Gamma(w^2l/2)} \quad (4.16)$$

If the probability $P(\Delta^2 \leq \Delta^2(i, j) | H_0) < T$, we declare that the pixel belongs to the background, else the pixel is set to be the moving pixel.

Recall that in the camera motion estimation algorithm (described in chapter 2), the iterative process successively rejected those MBs whose residual motion vectors (the absolute difference between the actual motion vector and the estimated motion vector) were greater than an adaptive threshold resulting in a Block Rejection Map (*BRM*). An example of a *BRM* was earlier shown in Figure 2.4. The statistical hypothesis testing algorithm described above is applied only on the rejected MBs in the *BRM* since it is observed that the moving objects lie predominantly in the rejected MBs. This also contributes to making the algorithm faster and to further improving its robustness. Instead of fully decoding all the MBs of the frame, only the rejected MBs are decoded. This saves both the processing and decoding time for each frame. From our experiment, on the average less than half of the MBs for each frame are decoded. It may be mentioned here that σ in equation (4.16) is estimated by using those pixels that do not lie in the rejected MBs using

$$\sigma^2 = \frac{\sum_{(i,j) \notin \{\text{rejected regions}\}} d^2(i, j)}{2N}$$

where $d(i, j)$ is the difference image between the first two frames and N is the number of pixels lying in the MBs that are not rejected. Finally, for the detected motion blobs,

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

we use morphological closing and opening operation to form connected regions. Those regions that are smaller than a threshold are removed. At the same time, two regions are merged if their Euclidean distance is less than a threshold. Lastly, the centers of mass of the regions are transformed back to the co-ordinate system of the reference frame. The algorithm is outlined in Algorithm 3.

Algorithm 3 Motion Blob Detection**Input:** Video frames: F_m **Output:** Center of mass of motion blob

-
- 1: **for** $i = m - l$ to $i = m + l$ {frame F_i lying in the temporal window $2l + 1$ } **do**
 - 2: **if** $px \neq (1, 0, 0), py \neq (0, 1, 0)$ {there is camera motion} **then**
 - 3: Warp F_i to F_m using the affine parameters computed with global motion estimation algorithm to get $\hat{F}_i$.
 - 4: **else**
 - 5: $\hat{F}_i = F_i$
 - 6: **end if**
 - 7: **end for**
 - 8: Compute $P(\Delta^2 \leq \Delta^2(i, j) | H_0)$ using Eq. 4.16 for every pixel (i, j) lying in the *BRM* of F_m . //statistical test to detect moving pixels.
 - 9: Threshold P to get a binary image.
 - 10: Apply morphological close and open on the binary image to form connected regions.
 - 11: Merge two regions if the distance between them is less than a threshold and remove regions whose sizes are less than a threshold.
 - 12: Find the centers of mass for each region.
 - 13: Transform the centers of mass to coordinate system of the reference frame.
-

4.5.2 Trajectory Evolution

The detected motion blobs are used to develop the trajectory which evolves with each successive frame. We now describe the proposed method for trajectory evolution.

Let T_u be a trajectory and Bl_v be a motion blob having center of mass at (x_v, y_v) . The color of blob Bl_v is represented by the 3-tuple (R_v, G_v, B_v) calculated as the mean of the R, G and B components of each pixel belonging to the blob. For each trajectory, we formulate an adaptive trajectory description D_u . The description consists of the mean

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

color component (R, G, B) and the center of mass component (x, y) of the moving blobs. When the trajectory is constructed, the description is initialized using the first blobs's mean color and center of mass.

Let D_t^u represent the current description for trajectory T_u . The center of mass and color component of D_t^u are represented by (x_t^u, y_t^u) and (R_t^u, G_t^u, B_t^u) respectively. We define the distance between blob Bl_v and trajectory T_u as the distance between Bl_v and D_t^u given by

$$d(Bl_v, D_t^u) = \sqrt{(x_v - x_t^u)^2 + (y_v - y_t^u)^2} + \beta \sqrt{(R_v - R_t^u)^2 + (G_v - G_t^u)^2 + (B_v - B_t^u)^2} \quad (4.17)$$

where β is a weight for the colour coherence part of the distance function and is chosen to be 0.1. The objective is to assign Bl_v to the trajectory that is closest to it in terms of the distance given by equation (4.17). The algorithm is initialized by assigning each blob as the beginning of a trajectory. For each successive frame, the distances of the blobs from the last blob in each trajectory is computed and stored in a matrix M of order $U \times V$, where U is the number of trajectories and V is the number of blobs. For each row, if the blob with the minimum distance is less than a threshold, then that blob is appended to the trajectory represented by that row. If the minimum distance is greater than the threshold, it means that the corresponding blob belongs to a new trajectory. If there are unassigned blobs remaining, they are marked as the beginning of a new trajectory. In this way, with each frame the trajectory evolves by comparison of the blobs in that frame with the last blob in each trajectory.

When new blobs are appended to the trajectories, the trajectory description is updated using the adaptive weighting equation:

$$M_t^u = (1 - w) \cdot M_{t-1}^u + w \cdot M_v, \quad (4.18)$$

Figure 4.5: Occlusion illustration

where M^u is the trajectory description component (R, G, B, x, y) , M_v is the corresponding component for the newly appended blob and w is the weight calculated as

$$w = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(M_{t-1}^u - M_v)^2}{2\sigma^2}\right\} \quad (4.19)$$

Occlusion is a very important problem in tracking. In sports video, there are many occlusions. Figure 4.5 illustrates that how the Adaptive Trajectory Description (ATD) helps the trajectories to evolve correctly under occlusion. Initially, two motion blobs A and B are detected at frame #1. The motion trajectory descriptions for the two trajectories are initialized with the blobs A and B . In the next frame (#2), A and B move into each other and part of A is occluded by B . Only one motion blob is detected in this frame. Since B takes a large portion of the detected blob, the detected blob is closer to trajectory description of B than to that of A . Thus, the detected blob is appended to trajectory B . However, the trajectory description of B is updated very little due to low value of the weighting factor w . In frame #3, A and B separated and two motion blobs A and B are detected. At this stage, the detected motion blobs A and B will be assigned

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

to the correct corresponding trajectories based on its distance to $ATDs$. The algorithm for the proposed trajectory evolution technique is summarized in Algorithm 4.

Algorithm 4 Trajectory Evolution**Input:** Motion Blobs Bl and Trajectories**Output:** Trajectories

```

1: Initialization : Assign each blob to a trajectory.
2: Calculate the distance matrix  $D$  using equation (4.17)
3: for each row  $u$  of  $D$  do
4:    $m = \min(D(u, v))$ 
5:   if  $m \leq Threshold$  then
6:     Append motion blob  $Bl_v$  to trajectory  $T_u$ .
7:     Update the trajectory description  $D_t^u$  with Eq. 4.18
8:   end if
9: end for
10: for all the remaining unassigned blobs do
11:   create a new trajectory
12: end for

```

4.5.3 Trajectory Refinement

Trajectory refinement consists of removing spurious trajectories and smoothening the remaining trajectories. If the length of the trajectories extracted is less than a threshold, they are considered to be spurious and hence removed. The remaining trajectories are deemed to be corrupted by white noise, and hence the refinement entails removal of noise. We propose to use a Kalman filter for noise removal since it provides optimal estimates of the system parameters in least square sense, given measurements and knowledge of a system's behaviour. The task then is to determine the state vector and the measurement vector. The state vector is chosen to consist of position and the first derivative of the position (that is the velocity) of a blob while the measurement vector is also chosen to

be the velocity. Thus, the system can be defined by the following equations :

$$\begin{pmatrix} x_i \\ y_i \\ dx_i \\ dy_i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ y_{i-1} \\ dx_{i-1} \\ dy_{i-1} \end{pmatrix} + \eta_i \quad (4.20)$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ dx_i \\ dy_i \end{pmatrix} + \nu_i$$

where η_i and ν_i are white noise. The Kalman filter yields an optimal estimate of the trajectory while removing noise.

4.6 XML Document for Motion Trajectory Descriptor

The extracted motion trajectory information consists of several lists of points. If this information is stored in different ways, it would not allow access across different repositories to enable content exchange using various description methods. These are interoperability issues, which can be solved by creating a standard description. As described in chapter 1, MPEG-7 provides a standardized description interface for multimedia content. It uses an extended XML and XML schema as the description definition language (DDL). There is a motion descriptor, called ‘Motion Trajectory’, which defines the syntax and structure to store the trajectory information.

The motion trajectory records the spatio-temporal locations of a moving object. In order to describe the spatial location of the points, the spatial coordinate system must be specified. MPEG-7 supports two kinds of 2D spatial coordinate systems: local and integrated. In a local coordinate system, the coordinates in the descriptor coordinate system are mapped to the current frame. In an integrated system, each frame of a video sequence is mapped to the coordinate system specified by the first frame. *SpatialRef*

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

attribute of the coordinate system element *CoordRef* indicates which type of spatial coordinate system is used; *SpatialRef* set to 0 implies the integrated system while *SpatialRef* set to 1 implies the local system. The other element *Params* specifies the spatio-temporal key-points and interpolation function used to express the trajectory. *KeyTimePoint* and *KeyValue* define the temporal and spatial locations of the key points. The *type* attribute of the *InterpolationFunctions* element specifies the type of interpolation function used between key points. The *CameraFollows* attribute indicates whether the camera follows the moving object. Further details of the schema for MotionTrajectory descriptor can be found in [91].

An example of the Motion trajectory descriptor is shown as follows:

```
<MotionTrajectory CameraFollows=1>
  <CoordRef ref="ID1" spatialRef=0/>
  <Params>
    <WholeInterval>
      <MediaIncrDuration mediaTimeUnit="PT1N25F">369</MediaIncrDuration>
    </WholeInterval>
    <KeyTimePoint>
      <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
        mediaTimeBase=" ../MediaLocator[1]">18</MediaRelIncrTimePoint>
        .....
      <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
        mediaTimeBase=" ../MediaLocator[1]">369</MediaRelIncrTimePoint>
    </KeyTimePoint>
    <InterpolationFunctions type="firstorder">
      <KeyValue> 166  146 </KeyValue>
      .....
      <KeyValue> 196  126 </KeyValue>
    </InterpolationFunctions>
  </Params>
</MotionTrajectory>
```

The camera follows the moving object (*CameraFollows=1*). The descriptor uses a coordinated system name as ID1, which is defined elsewhere. An integrated coordinated system is indicated by *SpatialRef=0*. The total length of the video is 369 frames with frame rate of 25 frames per second. The *KeyTimePoint* defines the temporal location of the keypoints in the trajectory. The *KeyValue* contains the spatial location of the

keypoints in the trajectory. The fact that first order interpolation is used between the key points is shown by *type* attribute of the *InterpolationFunctions*.

4.7 Experimental Results

In this section, we demonstrate the results of our approach for automatic trajectory extraction in sports video. It may be recalled that the overall framework of the algorithm as illustrated in Figure 4.1 begins with shot boundary detection followed by shot classification into long shots and close-up shots and global motion estimation before trajectory extraction. Therefore, our experiments are designed to assess the performance of the proposed approach for shot classification, global motion estimation and trajectory extraction.

4.7.1 Shot Classification

SVM is based on the structural risk minimization principle from statistical learning theory and can be used to find the optimal separating hyperplane in feature space [123]. Radial basis functions are chosen as the kernel for training our *SVM*. The experimental data consist of 700 key frames of long shots and closeup shots from various sports videos (i.e. basketball, soccer, badminton, tennis etc.). 50% of the data is used as the training samples, the remaining 50% is used as the test data.

Table 4.1: Shot classification using co-occurrence and run-length features.

Quantization level	128	64	32	16
Co-occurrence feature	0.814	0.971	0.925	0.835
Run length feature	0.852	0.914	0.871	0.643

The classification rate with different quantization level of intensity is shown in Table 4.1. If the quantization levels of intensity is chosen to be 64, we achieve a classification rate of 97.1% for co-occurrence based features and 91.4% for run-length based features.

4.7.2 Global Motion Estimation

We test our global motion estimation algorithm by creating an image mosaic from 30 frames of video. Figure 4.6 (a) and (b) are two frames from a soccer sequence, where both global and local motions exist. Figure 4.7 (a) and (b) are the mosaic images generated using the gradient based algorithm of [125] and the block motion vector based algorithm presented here, respectively. Note the white circle in the middle of the field is sharper using our method. Thus, the motion vectors estimated using the block based method are more accurate. Furthermore, our outlier rejection method introduces a degree of robustness which removes erroneous motion vectors caused either by local motion or mismatch. Our method converges faster than the gradient based method because of the known limitations of the latter. The added advantage of our method is the readily available motion vectors from compressed video streams.

Figure 4.6: (a) Frame 153 and (b) frame 183 of the soccer sequence

We use the proposed global motion estimation method to generate a mosaic from all the 300 frames of the soccer sequence as shown in Figure 4.8. Here, the motion of the camera consists of a pan from right to left.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.7: Mosaic of frames 153 and 183 of the soccer sequence using (a) gradient based algorithm and (b) the proposed motion vector based algorithm

Figure 4.8: Mosaic generated from 300 frames of soccer video

4.7.3 Motion Trajectory Extraction

We demonstrate the results of our algorithm for trajectory extraction in three sports videos, viz., badminton, soccer and tennis. The frame structure of the MPEG compressed videos is $I_1B_2B_3P_4B_5B_6P_7\dots$, out of which we process only the I and P frames. This will reduce the computational load by two thirds. Note that in order to visualize the trajectories and motion blobs, we show the fully decoded frames with the detected bounding boxes in the following figures, which does not mean we have to decompress all the macroblocks to extract the trajectory. As described earlier, only partial MBs lying in the *BRM* are fully decoded. We elaborate this point with reference to Table 4.2, later.

Figure 4.9: Motion blob detection in sample frames of the badminton sequence.

The badminton sequence here has no camera motion. As noted earlier, the trajectory extraction part consists of motion blob detection, trajectory evolution and trajectory refinement. Figure 4.9 shows the motion blobs detected from a sample of the frames in the badminton sequence. The threshold for probability in equation (4.16) to decide whether a pixel belongs to the background or the foreground was set to 0.1. We consider only those blobs which have more than 150 pixels. If the Euclidean distance between two blobs is less than 10 pixels, they are merged. Figures 4.10 (a) and (b) show the evolution of the trajectories from frames 1 to 123 and from frames 1 to 246 respectively.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.10: Trajectories extracted from (a) frames 1 to 123 and (b) frames 1 to 246 and (c) final trajectory extracted after refinement from 369 frames in the badminton sequence.

We see some spurious trajectories due to some amount of motion among the spectators. These spurious trajectories are removed and a smoothened trajectory for 369 frames is generated after refinement by the Kalman filter, as shown in Figure 4.10 (c). The output MPEG-7 compliant XML document of the trajectory is shown in Figure 4.11.

Next, we consider 300 frames of the soccer sequence which is part of the MPEG-7 test data set. In this sequence, the camera pans from left to right, while the players move from right to left. The detected motion blobs are shown for a sample of the frames in Figure 4.12. Here, we use the same value as that for the badminton sequence. Generally the threshold lies in the range $[0.0001 \ 0.1]$. The evolution of the trajectories is illustrated in Figure 4.13 where (a) and (b) are the trajectories extracted from frames 1 to 100 and from frames 1 to 200, respectively. If the number of blobs in a trajectory is less than 10, it is rejected. After refinement by Kalman filter, the final trajectory is shown in Figure 4.13 (c). The output MPEG-7 compliant XML document of the trajectory is shown in Figure 4.14. This document consists of 21 *MotionTrajectory* descriptors for the players and the referees and the *KeyTimePoint* and *KeyValue* of those *MotionTrajectory* descriptors are filled by the extracted 21 trajectories.

Finally, we extract the trajectories from a tennis sequence of 312 frames. Here too,

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

```

<?xml version="1.0" encoding="utf-8"?>
<!--Mpeg-7 Motion Trajectory Descriptor for "badmintion" sequence. -->
<MPEG7 xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 file:///c:/Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>file:///c:/badmintion.mpg</MediaUri>
        </MediaLocator>
      </Video>
    </MultimediaContent>
  </Description>
  <DescriptionUnit xsi:type="DescriptorCollectionType">
    <Descriptor xsi:type="MotionTrajectoryType" CameraFollows=0>
      <CoordRef ref="ID1" spatialRef=1/>
      <Params>
        <WholeInterval>
          <MediaIncrDuration mediaTimeUnit="PT1N25F">369</MediaIncrDuration>
        </WholeInterval>
        <KeyTimePoint>
          <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
            mediaTimeBase="../../MediaLocator[1]">18</MediaRelIncrTimePoint>
          .....
          <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
            mediaTimeBase="../../MediaLocator[1]">369</MediaRelIncrTimePoint>
        </KeyTimePoint>
        <InterpolationFunctions type="firstorder">
          <KeyValue> 251 75 </KeyValue>
          .....
          <KeyValue> 176 54 </KeyValue>
        </InterpolationFunctions>
      </Params>
    </Descriptor>
    .....
  </DescriptionUnit>
</MPEG7>

```

Figure 4.11: Motion trajectory descriptors for "Badminton video" in MPEG-7 compliant XML format.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.12: Motion blob detection in sample frames of the soccer sequence.

Figure 4.13: Trajectories extracted from (a) frames 1 to 100 and (b) frames 1 to 200 and (c) final trajectory extracted after refinement from 300 frames in the soccer sequence.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

```

<?xml version="1.0" encoding="utf-8"?>
<!--Mpeg-7 Motion Trajectory Descriptor for "Soccer" sequence-->
<MPEG7 xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 file:///c:/Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaLocator>
          <MediaUri>file:///c:/soccer.mpg</MediaUri>
        </MediaLocator>
      </Video>
    </MultimediaContent>
  </Description>
  <DescriptionUnit xsi:type="DescriptorCollectionType">
    <Descriptor xsi:type="MotionTrajectoryType" CameraFollows=1>
      <CoordRef ref="ID1" spatialRef=0/>
      <Params>
        <WholeInterval>
          <MediaIncrDuration mediaTimeUnit="PT1N25F">300</MediaIncrDuration>
        </WholeInterval>
        <KeyTimePoint>
          <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
            mediaTimeBase="../MediaLocator[1]">3</MediaRelIncrTimePoint>
          .....
          <MediaRelIncrTimePoint mediaTimeUnit="PT1N25F"
            mediaTimeBase="../MediaLocator[1]">300</MediaRelIncrTimePoint>
        </KeyTimePoint>
        <InterpolationFunctions type="firstorder">
          <KeyValue> 91 127 </KeyValue>
          .....
          <KeyValue> -360 110 </KeyValue>
        </InterpolationFunctions>
      </Params>
    </Descriptor>

    <Descriptor xsi:type="MotionTrajectoryType" CameraFollows=1>
      ...
    </Descriptor>
    .....
    <Descriptor xsi:type="MotionTrajectoryType" CameraFollows=1>
      ...
    </Descriptor>
  </DescriptionUnit>
</MPEG7>

```

Figure 4.14: Motion trajectory descriptors for 'Soccer video' in MPEG-7 compliant XML format.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.15: Motion blob detection in sample frames of the ‘tennis’ sequence.

Figure 4.16: Trajectories extracted from (a) frames 1 to 104 and (b) frames 1 to 208 and (c) final trajectory extracted after refinement from 312 frames in the tennis sequence.

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

there is global motion due to the camera. Figure 4.15 shows the motion blobs detected from the sample frames. The threshold for the probability in equation (4.16) is set to 0.01. Figure 4.16 shows the evolution process of the trajectories. Figure 4.16 (a) and (b) show the trajectories extracted from frame 1 to frame 104 and frame 1 to frame 208 respectively. We can see that the spurious trajectories fall into two categories. The first category is the short noisy trajectories which are separated from the dominant trajectories. These kinds of spurious trajectories are caused by false motion blob detection. Generally, they are short and can be eliminated by setting a threshold on the length of the trajectory. The second category of spurious trajectories are those that are entangled with the true trajectories. These kinds of trajectories are caused by oversegmenting an object into more than one motion blob. This would result in some false trajectories near the dominant one. Such trajectories can be eliminated if we only choose the dominant trajectories. Figure 4.16 (c) shows the final dominant trajectories after smoothing with Kalman filter. Since the structure of the XML document for the motion trajectories is very similar to the badminton sequence (there would be two trajectories corresponding to the two players), we do not include the XML file here. All the result video clips are available from the website (<http://www.ntu.edu.sg/home5/pg03763623/video/badminton.mpg>, [soccer.mpg](#), [tennis.mpg](#)).

To demonstrate the efficiency of the proposed approach, the decoded MB ratios (*DMBRs*), defined as the ratio of the number of MBs lying in the *BRM* to the total number of MBs in the frame, are calculated for the three experimental video sequences above. Figure 4.17 shows the *DMBRs* for the three video sequences used in the above experiment. Table 4.2 shows the average decoded MB ratio for the three sequences. The average *DMBRs* for ‘badminton’, ‘soccer’, ‘tennis’ sequences are 0.158, 0.404 and 0.481 respectively. The average *DMBRs* for ‘soccer’ and ‘tennis’ sequences are larger than that for the ‘badminton’ sequence. This is because the ‘soccer’ and ‘tennis’ sequence

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.17: Decoded MB ratio for the experiment sequence: ‘soccer’, ‘tennis’, ‘badminton’.

Table 4.2: Average decoded MB ratio.

Sequence	Soccer	Tennis	Badminton
$DMBR_{average}$	0.404	0.481	0.158

contain significant camera motion and more iterations are required during the global motion estimation, resulting in more blocks being rejected. On the other hand the camera remains stationary for ‘badminton’ sequence, requiring less iterations for computation of the camera motion and less blocks being rejected. In all the cases, the average $DMBR$ is less than 0.5, which means that less than half of the MBs are fully decoded.

4.7.3.1 Failure modes and Discussion

Our trajectory extraction method fails for ‘basketball’ games. Figure 4.18 shows an example frame from the ‘basketball’ game (one of the MPEG-7 test sequences) and its corresponding motion blob detection results. This ‘basketball’ video sequence presents a great challenge for trajectory extraction since the camera is moving fast and the players are cluttered around each other. In figure 4.18, we see that the motion blob detection algorithm is able to detect the moving players, but since the players are very close together, the detected motion blobs are not separated for individual players very well. Moreover

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

Figure 4.18: An example frame and the motion blob detection result for the ‘basketball’ video.

there is a lot of occlusion. These facts cause the trajectory evolution to fail.

As mentioned before, generally, trajectory extraction is a very difficult task. Our algorithm works well with certain kinds of games such as badminton, tennis, where the scene is simple and the context is fixed. Our algorithm is also capable of extracting motion trajectories under both stationary and moving cameras as long as the scene is not too complicated and the players are well separated like in the ‘soccer’ video. If the scene is very complicated and the players are cluttered like in the ‘basketball’ game, our algorithm would fail. For such games, semi-automatic tracking method might be a better choice.

4.8 Conclusion

A framework for automatic extraction of trajectories in compressed sports videos has been presented in this chapter. The pre-processing stages consist of shot detection and classification. Motion information to generate trajectories is available mostly in long shots rather than in close-ups. Hence, the former is used for further processing. The proposed trajectory extraction approach works irrespective of whether the camera is still or is moving. The trajectories are extracted through a four stage process of (1) global motion estimation, (2) motion blob detection, (3) trajectory evolution and (4) trajectory

CHAPTER 4. AUTOMATIC MOTION TRAJECTORY EXTRACTION

refinement. Experimental results have demonstrated the effectiveness and efficiency of the proposed method. The method can extract an arbitrary number of trajectories for multiple objects. Moreover, the method is applied directly on the MPEG compressed video without fully decoding of each frame. The extracted motion trajectories are output as MPEG-7 descriptors in an XML document. This XML document can then be put into use for browsing and sports video retrieval applications. Extraction of trajectories may not be an end in itself. The trajectory is very useful in many applications, for example, video retrieval by matching the trajectories [106].

Chapter 5

Towards Semantic Video Analysis

Although low-level features provide useful cues for video indexing, it is more appealing to describe the video content by its semantic meaning or concept (i.e. video category, key words etc) rather than low-level features. For example, a description of the video content by ‘the bride and the groom are getting married’ is more meaningful than the description ‘the video contains 50% white color, 30% black color, and 20% yellow color’. However, automatic extraction of high-level semantic information from video is not an easy task. Generally, there is a huge gap between the low-level feature to high-level semantic. Low-level feature similarity may not correspond to high level semantic similarity. In literature, many approaches have been proposed for semantic video analysis in specific domains, such as sports video analysis, medical video analysis, surveillance video analysis etc [64]. By applying the domain specific constraint, the gap between low-level features and high-level semantics are narrowed. Hence, it becomes easier to extract the semantics. However, daily broadcasting video has a wide domain. It contains news, sports, entertainment sitcoms etc. In this chapter, we present two approaches to deal with extraction of semantics from ‘broadcasting videos’. Our first approach is a hidden Markov model (*HMM*) video classification system. We utilize *HMM* to classify the videos into different themes broadly (i.e news, sports, and entertainment sitcoms) with multiple low level features. Our second approach builds up a direct and more detailed semantic index of the

Figure 5.1: Framework for pattern classification

video content by analyzing the scripts. The extracted semantic index is demonstrated for semantic video retrieval by key words. The semantic video index can be also used to generate both visual and keyword based video summaries. These two approaches can be combined to analyze the semantic content of ‘broadcasting videos’. First, we can use the video *HMM* to classify the video into different themes. Then, domain specific video analysis techniques can be applied on the videos in each category, for instances, story board can be extracted from the ‘news’ videos; semantic events (i.e. foul, shoot, penalty etc.) can be detected from ‘sports’ videos; our script analysis approach can be used for extraction semantic index for the ‘sitcom’ videos. In the following sections, we present the *HMM* video classification system and the script mining approach.

5.1 *HMM* based Video Classification

Video content classification is useful for efficient access and retrieval of videos. It is also useful for extracting semantics from videos. Not only does the classification itself provide semantic labels, but also it facilitates other semantic video analysis approaches, which serve to analyze the semantics in more details. Different methods have been proposed to classify videos into predefined categories [128]. The video classification approach follows the classical pattern classification algorithm depicted in Figure 5.1.

In [47], the authors use nearest neighbor clustering to do video classification. This classification method is simple, but it does not take the temporal features into consideration. Face and text trajectories are used to classify videos into predefined types by *HMM* in [48]. The authors in [49] use the chromaticity of keyframes in video summarization

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

to do video classification. The authors build an *HMM* framework using audio and visual features for video classification in [50]. The use of both audio and visual features can improve classification accuracy, but the system becomes complicated and computational unmanageable.

Video has both spatial and temporal dimensions. We take advantage of the temporal characteristics by incorporating them into *HMM*. *HMMs* are widely used in pattern recognition and have a good capability to grasp temporal statistical properties of stochastic processes. The essence of the *HMM* process is to construct a model that explains the occurrence of observations in a time sequence and uses it to identify other observation sequences.

In this section, we describe a rapid *HMM* based video classification approach to classify broadcasting videos into different classes like ‘news’, ‘sports’, ‘movie’ etc. We consider such a classification as a step towards creating semantic indices for video sequences. We use both visual and motion features. The visual feature is the compressed illumination invariant color histogram, which is extracted from DC sequence. The motion feature is the measurement of the motion amount, which is also computed from the compressed videos. After extracting the features, we use *HMM* to do video classification. In our experiment, we apply the method to classify broadcasting videos into four categories: ‘news’, ‘basketball’, ‘soccer’ and ‘sitcom’.

5.1.1 Hidden Markov Models

In an *HMM*, there are a finite number of states, each of which is associated with a transition probability to other state. At each time, the *HMM* stays in one specific state. The state at time t is directly influenced by the state at time $t - 1$. After each transition from one state to another, an output observation is generated based on an observation probability distribution associated with the current state [129]. We can only observe the output, but the state variables are “hidden” and unobservable.

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

The *HMM* model consists of the following parameters:

- (i) A set of N states, $s = \{s_1, s_2, \dots, s_N\}$, with the state at time t denoted by $q_t \in s$.
- (ii) The initial state probability distribution, $\Pi = \{\pi_i\}$, where $\pi_i = P(q_1 = s_i)$, $1 \leq i \leq N$.
- (iii) The state transition probability matrix, $A = \{a_{ij}\}$, where $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$, $1 \leq i, j \leq N$, with $0 \leq a_{ij} \leq 1$ and the constraint that $\sum_{j=1}^N a_{ij} = 1$, $1 \leq i \leq N$.
- (iv) The observation symbol probability for the observation, $B = \{b_j(O_t)\}$, where $b_j(O_t)$ is the probability of observation O_t at time t given that the state is $q_t = s_j$, $b_j(O_t) = P(O_t | q_t = s_j)$. In discrete *HMM*, the probability of observation symbols is defined as: $b_j(O_t) = P(O_t = \nu_k | q_t = s_j)$, $1 \leq j \leq N$, where $V = \{\nu_1, \nu_2, \dots, \nu_M\}$ is the set of all possible observation symbols, and M is the number of different observation symbols. Therefore, an *HMM* is defined as the triplet, $\lambda = (A, B, \Pi)$.

In structured video sequence, certain patterns appear regularly. For example, the newsreader will appear at regular intervals in news videos. In a basketball game, the left court, right court, middle court and close up will appear regularly. In soccer video, the penalty area, the midfield and close up will appear periodically. We attempt to model this process in time using a set of hidden states. The Markov assumption forces the current state of the model to depend only on the previous states. A Markov chain is a process that moves from one state to another state depending on the previous n states. It is of order n when n previous states influence the choice of the current state. If n is equal to 1, it means the current state only depends on the preceding state. In the video classification problem, the observations are feature vectors extracted from the observed frames. The hidden states of the *HMM* are modelled as a first order Markov chain, depicting the hidden semantic reasons for the frames; e.g. news reader shot, story board shot in news

videos. For a first order process with M states, there are M^2 transitions between states since it is possible for any state to transfer to any state (including itself). Each possible transition is associated with a transition probability. These M^2 probabilities are collected together to form a *state transition matrix*.

We model each video class as a stochastic process over the feature vectors extracted from video frames. An *HMM* is constructed for each class of videos. In the following, we describe the features extraction, *HMM* training and classification.

5.1.2 Feature Extraction

In MPEG videos, the frames are coded in Group of Picture (*GOP*) unit, e.g. I, B, B, P, B, B, P, B, B, P, B, B. Every *GOP* can be decoded independent of the others. We extract color and motion features from each *GOP*; thus the features are extracted in the compressed domain.

5.1.2.1 Color Feature

The first frame of every *GOP*, called *I* frame, is intra coded using Discrete Cosine Transform (*DCT*). The color feature is extracted from the DC image of the *I* frame. Since the DC image is constructed by the DC coefficients of all the 8×8 blocks over the *I* frame, its size is 64^{th} of the original frame. Compared with the original frames, DC image is a blurred representation, but they are less influenced by camera and object motion. The colors in DC images are more robust than those in the whole frame.

We analyze the DC image colors in RGB color space. The colors in the DC image is converted to RGB color space. It is known that the R , G , B components of the colors are increased proportionally to the increase of illumination. In broadcasting videos, the illumination varies drastically. If the videos are captured under sunshine or strong light, the illumination is high, while if the videos are captured under dim light, the illumination

is low. In order to make the feature invariant to illuminations, the colors (R, G, B) are normalized by the following equation:

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \frac{1}{\sqrt{R^2 + G^2 + B^2}} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (5.1)$$

Since the degree of freedom of color vector $[r, g, b]^T$ is two (i.e., $r^2 + g^2 + b^2 = 1$), any two components capture all the color information. We use only r and g to extract the color histograms. The computed histogram is two dimensional and can be regarded as a image, called ‘histogram image’, where each bin (r, g) corresponds to the number of pixels in the DC images whose colors are (r, g) . In our experiment, the colors r and g are quantized into 16 levels, which yields 16 by 16 histogram images. In order to further compress the feature vectors, a 16 by 16 Discrete Cosine Transform (*DCT*) is used to compress the ‘histogram images’. *DCT* is widely used in image and video compression because of its near optimal performance. The low frequency *DCT* coefficients captures a large percentage of the total energy. In [130], the authors demonstrate that *DCT* is suitable to compress image histogram as well. After compressing the histogram using *DCT*, the first 21 low-frequency coefficients resulting from zigzag scan are extracted as the color feature vectors.

There is still redundance in the color feature vectors, i.e. the low-frequency coefficients. For instance, the first coefficient in the feature vector, DC component, is equal to the average value of the ‘histogram image’ and the average of the histogram images is determined by the frame size. Since the frame size for video broadcasting is standardized by CCIR 601 specification, the DC values are the same for all feature vectors. Hence, this dimension is trivial, which will result in the feature covariance matrix being singular. This is not desirable during the training and classification with *HMM* as shown in the following sections. Therefore, *PCA* is used to further reduce the dimension of the features. First, we collect the feature vectors from all the training sequence into a feature

matrix, F_c ($21 \times N$). Each column in F_c corresponds to one *GOP* color feature and N is the total number of *GOP*s. The computation of the principle component P from the feature matrix F_c is shown by the following equation:

$$C_F = F_c^T F_c = P^T D P \quad (5.2)$$

where C_F is the feature covariance matrix and D is the diagonal matrix whose diagonal entries are eigen values. In our experiment, we choose 10 eigen vectors, which correspond to the 10 largest eigen values as the principle components. We emphasize that although the basis are computed during the training stage, they can be used for any new incoming videos at the classification stage as well.

5.1.2.2 Motion Features

Motion is useful to describe the video content. For compressed videos, motion information is captured in motion vectors. Hence, we use motion vectors of P frame to compute the amount of motion F_m , which is the motion feature. First, the average magnitude of the motion vectors in each P frame is calculated. Then, the median among all average magnitudes of the P frames within the *GOP* is chosen as the motion feature, F_m . It measures the amount of motion in the *GOP*. The larger the amount of motion, the higher the value of F_m . The final *GOP* feature vector is constructed by concatenating the 10 dimensional color feature vector and the scalar motion feature:

$$F = (F_c \ F_t)^T$$

These feature vectors are used as the observations for training and classification with video *HMM*.

5.1.3 Video classification

As described before, *HMM*s are discrete-time stochastic signal models. They can be completely defined by the number of hidden states, the static state transition probability

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

matrix, the observation symbol probability distribution and the initial state distribution. The hidden state sequence observe the Markov property. Current state depends only on the preceding state if a first-order HMM is used, which is the case in our method.

In our experiment, a continuous observation first-order HMM template was created for each category of videos with random initial parameters. Each state's observation probability distribution is modeled as a single 11 dimensional Gaussian with diagonal covariance matrix. The number of hidden state varies from 2 to 4.

To determine the class, we have to estimate the *HMM* parameters, namely state transition probability matrix, the means and the diagonal covariance matrix of Gaussian distribution for each state. Because of the assumption that the observation distribution is a 11 dimensional Gaussian, the covariance matrix should be nonsingular. When the determinant of the covariance matrix becomes too small, the performance of *HMM* will deteriorate drastically. To avoid this problem, *PCA* has to be applied to reduce the feature dimension so that the eigen value of the covariance matrix is not too small. As described before, we reduce the dimension of the color feature vectors from 21 to 10. The reduced feature vectors capture the most discriminative information. By ignoring the trivial information, the reduced features avoid overtraining and achieve faster convergence during training. Furthermore, because general video conditions are more or less the same between the training set and the test set, the basis computed with the eigen vectors at the training stage are applicable for the test set as well.

The estimation of the *HMM* parameters is also called the “decoding process” [129]. Baum-Welch algorithm is a well-known algorithm to estimate *HMM* parameters. In essence, it is a generalized Expectation-Maximization algorithm. The approach iteratively updates the *HMM* parameters in order to better explain the observed training sequences. The *HMM* parameter estimation algorithm is summarized in Algorithm 5.

After estimating the *HMM* parameters, the next problem is to evaluate the probability of a sample series generated from a given *HMM*. This probability corresponds to the

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Algorithm 5 Parameter Estimation of *HMM*

- 1: Give a rough estimate of a_{ij}, μ_j, C_j
- 2: Compute probability among all the state paths generate the observation and sum the probability P_s .
- 3: Find all the probability of $s_i \rightarrow s_j$ among all the state paths and sum them $P_{a_{ij}}$.
- 4: Update a_{ij} by $a_{ij} = \frac{P_{a_{ij}}}{P_s}$.
- 5: Find all the probability of $s_j \rightarrow \nu_k P_{b_j}(O_t)$.
- 6: Use P_{b_j} to update the mean and the variance of the gaussian distribution by

$$\bar{\mu}_j = \frac{\sum_{t=1}^T P_{b_j}(O_t) \cdot O_t}{\sum_{t=1}^T P_{b_j}(O_t)}$$

$$\bar{C}_j = \frac{\sum_{t=1}^T P_{b_j}(O_t) \cdot (O_t - \mu_j)(O_t - \mu_j)^T}{\sum_{t=1}^T P_{b_j}(O_t)}$$

- 7: Go to step 2, until the change of a_{ij}, μ_j and C_j are less than a threshold (convergence achieved).

summation of all possible state paths in the *HMM* to generate the observation series.

It can be calculated using the ‘forward’ algorithm which is a recursive procedure [129].

The ‘forward’ algorithm is summarized in Algorithm 6.

Algorithm 6 Evaluation of *HMM*

- 1: Initialization:

$$\alpha_1(i) = P_{b_j}(O_1), 1 \leq i \leq N.$$
- 2: **for** $t = 1$ to $T - 1$ **do**
- 3: $\alpha_{t+1}(j) = [\sum_{i=1}^N \alpha_t(i) a_{ij}] P_{b_j}(O_{t+1}).$
- 4: **end for**
- 5: return $P(O^T | \lambda) = \sum_{i=1}^N \alpha_T(i).$

For video classification, we build up an *HMM* for each type of video and train the *HMMs* using the training set with Algorithm 5. Algorithm 6 is used to calculate the log likelihood for each test video sequence associated with each *HMM*. We classify the video sequence to the video category whose *HMM* gives the highest probability. This is known as maximum likelihood (*ML*) classifier, which is optimal in Bayes sense. Equation (5.3)

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Figure 5.2: A diagram of training and classification of video clips using *HMM*

shows the *ML* classification scheme.

$$P(\lambda|O) = \frac{P(O|\lambda) \cdot P(\lambda)}{P(O)} \propto P(O|\lambda) \quad (5.3)$$

where λ denotes the video class, O are the observation features, and $P(\lambda)$ and $P(O)$ are assumed to be equally distributed. The training and classification diagram is summarized in Figure 5.2.

5.1.4 Experimental Results

We evaluate our *HMM* video classification approach by classifying the videos into 4 types of programs: ‘news’, ‘basketball game’, ‘soccer game’ and ‘sitcom’. The whole data set

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Table 5.1: Training and test set

	Basketball	Soccer	News	Sitcom
Training set	4	4	4	4
Test set	5	5	6	6

Table 5.2: Classification result with HMM (2-4 hidden states) HMM and 3-component GMM

States Num	Precision/Recall%			
	Basketball	Soccer	News	Sitcom
2	100/60	83/100	85.7/100	100/100
3	100/100	100/100	100/100	100/100
4	100/80	83/100	100/100	100/100
GMM	80/80	80/80	83.3/83.3	100/100

contains 38 broadcasting videos. Those videos are taken from the MPEG-7 test set. They are compressed with MPEG-1 standard. Each video sequence lasts for one minute. The videos are randomly divided into training (40%) and test sets(60%). The numbers of training and test videos are shown in table 5.1.

The features are extracted efficiently from the compressed videos with no need for full decoding. It takes less than 3 seconds to process 1 minutes video (1500 frames) without any optimization of the source code (c++) on Pentium IV 1.6G CPU. We use *precision* and *recall* to measure the performance of the classification. The *precision* and *recall* is defined as follows:

Precision: For a video category, *precision* is the ratio of the number of test videos correctly classified to this category over the total number of test videos classified to this category.

Recall: For a video category, *recall* is the ratio of the number of test videos correctly classified to this category over the number of all the test videos in this category.

Table 5.3: Log likelihood distance measure with different combinations of color spaces and compression methods

HSV/Wavelet	HSV/DCT	RGB/DCT
0.727132	1.360332	0.746141

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

The classification results with various number of states are shown in table 5.2. From the table, we can see that 3-state *HMM* gives perfect classification result. We compare the *HMM* classification scheme with Gaussian Mixture Model (*GMM*). In *GMM* based video classification, each video class is modeled as a mixture of Gaussians. Since 3 states give the best performance for *HMM* with a single Gaussian observation model, the number of Gaussian components in the *GMM* is set to be 3. *EM* algorithm is used to estimate the set of parameters (for details on the *GMM*, please refer to chapter 3). Both models use the same set of features. We also use the maximal likelihood classifier to classify the videos as shown in equation (5.3). The last row in table 5.2 shows the classification results with *GMM*. It is clear that *HMM* outperforms *GMM* model for the same set of features and in the same setting. This is because *HMM* encapsulates the temporal dependency into the model while *GMM* considers each frame individually. This temporal dependency information is shown to be valuable for the classification task.

In the following experiment, we compare the performance of color features extracted from other color spaces and with other compression methods. We choose two color spaces, i.e. *RGB* and *HSV* and two compression methods, i.e. *DCT* and *wavelet*. The number of the hidden states is fixed to 3. Since all *GOP* features yield perfect classification result, we use the mean difference between the largest and second largest log likelihood as the criterion. The larger the difference, the more effective the features are. The results are shown in table 5.3. From the table, we observe that the *HSV + DCT* features yield the best result. This is expected since *HSV* color space is closer to human's perception of colors than *RGB* color space and *DCT* has better energy compacting capability in colors than wavelet transform.

Figure 5.3: The proposed approach to build a semantic index.

5.2 Semantic Video Indexing Using Subtitles

The *HMM* video classification system can be used to categorize the video into broad classes (i.e. news, sports, sitcoms etc). In order to extract more detailed semantic index, further domain specific analyzing approaches are proposed, such as news board extraction, semantic event detection in sports video etc [64]. The general theme underlying those approaches is to combine domain rules and visual features to infer semantics. However, such approaches are too domain-specific. The approaches focus on the visual or auditory information and are not able to convey the high level semantics very well. In this section, we present a different approach to analyze the video semantics in more details. Our approach is based on analyzing the scripts of the videos. Scripts of videos provide direct access to the semantic aspect of the video content because the semantic information is captured very well in the scripts. Thus, it seems to be prudent to exploit this fact to extract semantic information in videos, instead of developing complex video processing algorithms. Our approach is illustrated in Figure 5.3.

Here, we are using the subtitle files from the DVD videos as the script source. Note if such script source files are not available, we can still extract the script information by other means, such as the speech recognition techniques. In the following, we assume that the script information is available in DVD subtitle files. First, the scripts with time stamps are extracted from the subtitles associated with the video. The second step is

to partition the scripts into segments. Each segment of the script is then converted into a vector based representation which is used as the semantic index. The vector based indexes can be used for retrieval and summarization.

5.2.1 Semantic Video Indexing

In this section, we describe in detail the 3 stages of the proposed technique to index video sequences based on semantic information extracted from scripts in a video. The stages are script extraction, script partition and script to vector mapping.

5.2.1.1 Script Extraction

DVD videos come with separate subtitle or script files for each frame in the video sequence. There are always separate subtitle files on the disc. There are two types of subtitle files - one in which the scripts are recorded as bitmap pictures which are drawn directly on the screen when the video plays and the other in which the scripts are recorded as strings in a text file. The text based subtitle files are much smaller and more flexible than those based on bitmaps. The advantage of using a text based subtitle file is that it is not only human-readable, but the user can easily change the appearance of the displayed text. However, bitmap pictures can be converted to text using readily available software such as 'VOBSUB' [131]. VOBSUB uses Optical Character Recognition (OCR) technique which is able to translate pictures of characters into a standard machine-editable text (ASCII or Unicode). The OCR techniques involve a training and a recognition phrases. First, a set of known samples for each character is provided in the training phrase. Then, after being trained, the system is able to recognize most of the character at almost perfect accuracy. Hence, we focus on script extraction from the text base subtitle file. An example of a text based subtitle file is shown in Figure 5.4. Each script in the file consists of an index for the script, the time of appearance and disappearance of the script with respect to the beginning of the video and the text of the

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

```
. . . . .
. . . . .

14
00:00:26,465 --> 00:00:28,368
You didn't tell anybody I was, did you?

15
00:00:28,368 --> 00:00:29,682
No.

16
00:00:30,810 --> 00:00:32,510
I'll be right back.

17
00:00:33,347 --> 00:00:35,824
Now, why don't we get a shot of just Monica and the
bloody soldier.

. . . . .
. . . . .
```

Figure 5.4: Script example file

script. The subtitle file is parsed into *ScriptElements*, where each *ScriptElement* has the following three attributes: ‘Start Time’, ‘End Time’ and ‘Text’. We use the information in the *ScriptElements* in order to partition them in the next step.

5.2.1.2 Script Partitioning

Following the script extraction process that results in *ScriptElements*, we now discuss the partitioning of the *ScriptElements*. The objective here is to group together those *ScriptElements* that have a common semantic thread running through them. Clearly, it is the temporally adjacent *ScriptElements* that are grouped together because they tend to convey a semantic notion when read together. At the same time, some *ScriptElements* may contain only a few words, which by themselves do not convey any semantic meaning. This leads us to the question of how to determine which *ScriptElements* should be grouped together to create a partitioning of the entire script.

We use the time gap between *ScriptElements* as the cue for script partition. This time gap, which we call the *ScriptElement* gap is defined as the time gap between the ‘EndTime’ of the previous *ScriptElement* and the *StartTime* of the current *ScriptElement*. In a video, when there is a dialogue or a long narration that extends to several frames,

the *ScriptElement* gap is very small. Concurrently, it is evident that *ScriptElements* that constitute an extended narration will also have a high "semantic correlation" among themselves. Hence, it is seen that the *ScriptElement* gap is a useful parameter by which to group together semantically relevant *ScriptElements*, thereby creating a partition of the scripts. In the proposed method, the *ScriptElements* are partitioned by thresholding the *ScriptElement* gap. We call each partition as a *ScriptSegment*.

5.2.1.3 Script Vector Representation

After partitioning the scripts into *segments*, we build an index for each script segment. We adopt the *term-frequency inverse document frequency* (*tfidf*) vector space model [132], which is widely used for information retrieval, as the semantic index for the segments.

The first step involves removal of stop words, e.g. "about, I". The Potter Stemming algorithm [133], is used to obtain the stem of each word. The stems are collected into a dictionary, which are then used to construct the script vector (i.e., the index) for each segment. To illustrate, the stem for the word "families" is "famili". Just as the vector space model represents a document with a single column vector, we represent the script segment using the *tfidf* function [134] given by

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|S_s|}{\#S_s(t_k)} \quad (5.4)$$

where $\#(t_k, d_j)$ denotes the number of times that a word t_k occurs in segment d_j , $|S_s|$ is the cardinality of the set S_s of all segments, and $\#S_s(t_k)$ denotes the number of segments in which the word t_k occurs. This function states that (a) the more often a term occurs in a segment, the more it is representative of its content, and (2) the more segments a term occurs in, the less discriminating it is. The *tfidf* function for a particular segment is converted to a set of normalized weights for each word belonging to the segment according to

$$w_{k,j} = \frac{tfidf(t_k, d_j)}{\sqrt{(\sum_{i=1}^T (tfidf(t_i, d_j))^2)}}. \quad (5.5)$$

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Here, $w_{k,j}$ is the weight of the word t_k in segment d_j and T is the total number of words in the dictionary. This is done to ensure that every segment extracted from the subtitle file has equal length and that the weights are in $[0,1]$. It is these weights that are collected together into a vector for a particular segment such that the vector acts as a semantic index to that segment. We call this vector as the “*tfidf* vector” in the following discussion.

5.2.2 Applications

We now illustrate two applications of the semantic index that has been extracted from the script files of DVD videos using the proposed method. In the first instance, we retrieve a video sequence using a keyword or a sentence as the query. The second application is video summarization wherein the semantic index is used to create a summary of the entire video - the summary can be expressed as a set of keywords or as a video.

5.2.2.1 Video Retrieval

In this subsection, we illustrate video retrieval with script vector based representation which acts as a semantic index. As described in the previous section, each script segment is represented as a *tfidf* vector. We collect all the column script vectors together into a matrix of order $T \times |S_s|$, called the script matrix.

The query can be in the form of a single word in which case the query vector (which has the same dimensions as the *tfidf* vector) will consist of a single non-zero element. For example, a query with the word “bride” will result in a query vector like $[0 \cdots 1 \cdots 0]$, where only the entry of the vector correspond to word “bride” are set to 1, others are all set to zeros. The query can also take the form of a sentence like “The bride and groom are dancing”; here the query vector will look like $[0 \cdots 1/\sqrt{3} \cdots 1/\sqrt{3} \cdots]$. As we see, the word(s) that are present in the query will have higher values in the query vector. The result of the querying process is the return of script segments which are geometrically close to the query vector. This can be accomplished in many ways, but the measure we

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

will use is the cosine of the angle between the query vector and the columns of the script matrix using,

$$\cos \theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^T a_{ij} q_i}{\sqrt{\sum_{i=1}^T a_{ij}^2} \sqrt{\sum_{i=1}^T q_i^2}} \quad (5.6)$$

for $j = 1 \cdots |S_s|$, where a_j is a column vector from the script matrix, q is the query vector and T is the number of words. Those script vectors for which equation (5.6) exceed a certain threshold are considered relevant. Alternatively, we could sort the values of $\cos \theta_j$ to present the top n results.

We could also use other similarity/distance measures, such as the norm of the difference between the query vector and script vector. Since both the computations are monotonic, they will achieve the same result. In both cases, we have to normalize the vectors. However, the sparsity of the vectors, especially the query vector, is a key feature in the model. Consider what happens when we take the similarity of a very sparse query vector with a dense script vector. In order to compute the distance between such vectors, we would have to subtract each entry of the query from each entry in the script vector, and then square and add each of them. Even precomputing the norms of the script vectors is not feasible since it is computationally expensive and, furthermore, large storage will be required to store the values when dealing with videos with thousands of script segments. However, using cosines, we can take advantage of the sparsity of the query vector, and only compute those multiplications (to get the numerator in the equation) in which the query entry is non-zero. The number of additions is also then limited. The time saved by taking advantage of sparsity would be significant when searching through long videos.

Another observation about the script matrix is that it is very sparse because many of its elements are zero. On the other hand, the dimensionality of the script vectors is very high. Hence, it is desirable to reduce the rank of the matrix. This is viable because if we

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

assume that the most represented words in the script matrix are in many basis vectors, then deleting a few basis vectors will only delete the extraneous and least important information in the script matrix. This will then result in a more effective index and search.

We use the Singular Value Decomposition (*SVD*) to reduce the rank of the script matrix. *SVD* factors a $T \times |S_s|$ script matrix A into three matrices: (i) a $T \times T$ orthogonal matrix U with the left singular vectors of A in its columns, (ii) a $|S_s| \times |S_s|$ orthogonal matrix V with the right singular vectors of A as its columns, and (iii) a $T \times |S_s|$ diagonal matrix E having the singular values in descending order along its diagonal, i.e.

$$A = UEV^T. \quad (5.7)$$

If we retain only the largest k singular values in the diagonal matrix E , we get the k_{th} rank approximation matrix A_k of the original matrix A . The k_{th} rank approximation of A by *SVD* is the mathematically best approximation (in terms of the Frobenius norm) [132]. That is:

$$\|A - A_k\|_F = \min_{\text{rank}(X) \leq k} \|A - X\| = \sqrt{\sigma_{k+1}^2 + \dots + \sigma_{r_A}^2} \quad (5.8)$$

Here $A_k = U_k E_k V_k^T$, where U_k is $T \times k$ and consists of the first k columns of U , V_k is $|S_s| \times k$ matrix whose columns are the first k columns of V , and E_k is the $k \times k$ diagonal matrix whose diagonal elements are the ordered k largest singular values of A . The σ s in the equation are the singular values, or the diagonal entries in E .

Using the approximate k_{th} rank script matrix made by *SVD*, we can then compute the cosines of the script vectors with the query vectors of same as the script vector:

$$\cos \theta_j = \frac{(A_k e_j)^T q}{\|(A_k e_j)\|_2 \|q\|_2} = \frac{(U_k \Sigma_k V_k^T e_j)^T q}{\|(U_k \Sigma_k V_k^T e_j)\|_2 \|q\|_2} = \frac{e_j^T V_k \Sigma_k (U_k^T q)}{\|\Sigma_k V_k^T e_j\|_2 \|q\|_2} \quad (5.9)$$

$$\cos \theta_j = \frac{s_j^T (U_k^T q)}{\|s_j\|_2 \|q\|_2}, \quad j = 1, \dots, |S_s| \quad (5.10)$$

where $s_j = E_k V_k^T e_j$, and e_j is the j th canonical vector of dimension d . The *SVD* factorization of the script matrix will not only help to reduce the ‘noise’ of script matrix, but also improve the recall rate of retrieval.

5.2.2.2 Summarization

In this subsection, we propose a new video summarization method by clustering the script matrix. Recall that the columns of the script matrix are the script vectors for the script segments. The script vectors can be viewed as points in a high dimension vector space. As described before, the script vectors are of very high dimension and sparse. It is desirable to reduce the dimension of the feature space. Principle Component Analysis (*PCA*) is used to reduce the dimensions of the script vector. The *PCA* used here has the same effect of the *SVD* used in the retrieval application on reducing the script vector representation.

After the dimension reduction of the script vectors with *PCA*, the script vectors are clustered in the high dimension space. K-means algorithm is used to cluster those script vector points. After clustering, those script segments whose script vectors are geometrically closest to the centroids of the clusters are picked out and concatenated the form the video summary. Besides, the script text of the picked out segments can be used as the text abstract of the videos. The number of clusters can be determined by the desired length of the video summary, e.g, if the desired length of the video summary is 5% of the original video, then the number of clusters should be one twentieth of the total number of the script segments.

5.2.3 Experimental Results

In this section, we present the experimental results to demonstrate the efficacy of the proposed semantic indexing method. Our test data consists of a single episode from the popular TV sitcom ‘Friends’(season 8 episode 1).

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Figure 5.5: (a) Time gap between the script element of ‘friends’ video. (b) Energy ratio VS feature rank (*PCA*).

Figure 5.5(a) shows the distribution of the time gap between *ScriptElements* for a total of 450 *ScriptElements*. In our implementation, we use 2 seconds as the threshold to partition the scripts into segments. With this threshold, the 450 *ScriptElements* are partitioned into 71 script segments. For each script segment, a script vector is extracted from the text as described in subsection 5.2.1.3.

Several queries by keywords are performed on the script matrix to retrieve the corresponding script segment as described in subsection 5.2.2.1. The retrieval results using the keywords ‘bride’, ‘dance’, ‘groom’ and ‘wedding’ are shown in Figures 5.6 (a), (b), (c) and (d), respectively. As we can see, the proposed method has successfully retrieved the relevant scripts as well as the associated video sequences. Thus, a high level semantic notion like ‘bride’ can be easily modelled using the technique described in this chapter.

In order to illustrate the results for video summarization, we use Principal Component Analysis (*PCA*) to reduce the script vector from 454 to 50 dimensions. Figure 5.5(b) shows the plot of the percentage of the total variance in the observations explained by the reduced rank of feature with *PCA* (Energy Ratio). The first 50 dimensions capture more than 98% of the total energy. We extract 10 keywords from 5 principle components with the largest 5 eigenvalues. We examine the absolute value of each entry in those principle component vectors and pick out the largest two entries for each principle component

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

Figure 5.6: Example retrieved results: (a) ‘bride’ query, (b) ‘dance’ query, (c) ‘groom’ query, (d) ‘wedding’ query

vector as the key words. The extracted ten key words are ‘happy’, ‘Chandler’, ‘marry’, ‘Joey’, ‘Monica’, ‘bride’, ‘husband’, ‘pregnant’, ‘baby’ ‘dance’. This episode talks about two friends ‘Chandler’ and ‘Monica’ getting ‘married’, Rachel getting ‘pregnant’ (with a ‘baby’) and Rose dancing with bridesmaids at the wedding party. We see that the extracted key words give a good summary of the content of this video.

We also extracted video summaries from the original video with the lengths of 5%, 10%, and 20% of the original video. We find that the video summaries capture most of the content of the video. We observe that the 10% video summary is the optimal one. While the 5% summary is too concise and a little difficult to understand, the 20% summary is quite a few redundancies (The result summary videos are available at <ftp://user:123456@155.69.103.62/>).

5.2.4 Discussion

The approach presented in this section points to a trend for future video indexing. Efficient and effective management of the video content depends on the suitable indices. Since video itself contains different modalities, e.g. visual, auditory, and textual, different modalities should be explored collaboratively. The open question is how to utilize and integrate different modalities for video index seamlessly. Previous researches mainly focus on the visual and auditory modalities, while the semantic rich textual modality is largely ignored. Visual and auditory information can be indexed by feature vectors quantitatively. In this chapter, we proposed to use the *tfidf* vector to index the textual modality quantitatively, which can be seamlessly integrated into the visual and auditory modalities. The script information is a bridge for the information retrieval and natural language processing research communities to make contribution to video indexing. Besides our approach, there are some recent works on how to integrate different modalities for video index [135]. Most of the integration methods report improvement of performance over using single modality. Multimodality increases the context for video indexing and yields a richer semantic index.

5.3 Conclusion

In this chapter, we present two approaches to extract the semantics for ‘broadcasting’ videos. The first one is an *HMM* video classification system. We classify the videos into different classes. We use color and motion features as the observation of the video *HMM*. The feature extraction method is quite efficient since it is carried out in *compressed* domain with minimal decoding requirement. Experimental result demonstrate that the *HMM* video classification scheme is effective for classifying the video into broad and strongly different categories (news, sports, sitcoms etc). After classifying the broadcasting videos into different classes, we propose the second approach to analyze the video

CHAPTER 5. TOWARDS SEMANTIC VIDEO ANALYSIS

semantics in more detail. We apply modern information retrieval techniques to analyze the scripts of the videos to get the semantics directly for video indexing. The semantic video index is extracted by analyzing the script and represented by the vector based model. Experimental results on video retrieval and summarization demonstrate the effectiveness of the proposed approach, especially for sitcoms. Other Information Retrieval models can be used to analyze script and the extracted semantic index (tfidf vector) can be combined with the low level features to give a more comprehensive index to the content. The extracted video summary can be easily integrated into MPEG-7 standard video summary representation using MDS (Multimedia Description Schema).

These two techniques provides a combined approach to extract semantics for daily 'broadcasting' videos. First, the *HMM* video classification system is applied to classify the videos into different categories, such as news, sports, sitcoms and others. Then, different domain-specific techniques are applied to extract the detailed semantics for the videos in different categories. Our script mining approach can be used to build up a detailed semantic index for the entertainment sitcoms.

Chapter 6

Conclusion

Multimedia indexing and description is a critical step in the creation of a multimedia information system. The focus of this thesis is a problem of finding an appropriate representation and indexing scheme for video content, with special emphasis on the use of motion as an index. In traditional relational databases, the indices for each item are the key fields since the data are organized in a well structured manner, i.e. table, attributes etc. However, the notion of index with respect to images and videos undergoes a paradigm shift in that an image is indexed using low-level pictorial features (e.g. color, texture, and shape etc.) and a video is also indexed using low-level features extracted from individual frames or a collection of frames. MPEG-7 provides a set of comprehensive and standardized descriptors for multimedia content indexing. However, the extraction and consumption of this index information in the most accurate, efficient and effective way is open for research.

6.1 Summary of Contributions

The main contributions of the thesis are summarized below:

- (i) We identify three fundamental tasks in building up an efficient video database management system: 1) The entire video sequence must be segmented into shots,

CHAPTER 6. CONCLUSION

where a shot is defined as a collection of frames recorded from a single camera operation. We develop a rapid shot boundary detection algorithm that is capable of detecting both the abrupt and gradual shot changes like dissolve, fade-in, fade-out. 2) Shots that are related to each other must be grouped together into a scene, which is the semantic unit for the video content. We present an adaptive hierarchical structure called the ‘Scene Tree’ for video content organization. The scene tree is shown to capture some semantic information as well as to provide a construct for hierarchical browsing of *compressed* videos. 3) Finally, a robust retrieval method for a video information system depends on a model that captures similarity in the semantics of video content. We build a new model for video similarity based on global as well as local motion associated with each node in the scene tree.

- (ii) We introduce a new representation for video motion content based on *pixel changes*. It is called the Pixel Change Ratio Map (*PCRM*). From the *PCRM*, two motion features for video motion content indexing are extracted: 1) motion histogram and 2) motion correlogram. The *PCRM* captures the *intensity* and *duration* of the motion in a video, which are the two most important factors for human’s perception of motion content. The *PCRM* also indicates the spatial location and size of the moving object(s). We demonstrate the usefulness of the motion feature in video retrieval, video clustering and video classification. We also propose a new key frame extraction method with the pixel change based video motion representation. The key frame extraction approach is based on change detection of DC images extracted from compressed video. The pixel change map captures additional information in a frame with respect to its adjacent frames. Since global motion contributes to pixel changes, falsely indicating the presence of key frames, we have to compensate the global motion. An adaptive filtering scheme is applied on the pixel change map

CHAPTER 6. CONCLUSION

to compensate the global motion. The prediction errors thus obtained are used to subsequently select the key frames. The entire procedure is computationally simple and flexible.

- (iii) We investigate the extraction of the motion trajectory descriptors. Motion trajectory is a fairly high level descriptors in MPEG-7 for video content description and indexing. The presented method is capable of extracting arbitrary number of the motion trajectories under both stationary camera and moving camera. It consists of the following stages: (1) shot boundary detection, (2) shot classification, (3) global motion estimation, (4) motion trajectory extraction. The shot classification step classifies the shot into close-up and long shots. Since motion trajectory information is mostly contained in long shots, we use only long shots for further trajectory extraction. The global motion is estimated from the MPEG motion vectors in compressed video using our iterative algorithm with robust outlier rejection. Statistical hypothesis test is applied within the Block Rejection Map, which is the by-product of the global motion estimation, to detect the motion blobs. The usage of Block Rejection Map avoids full frame decoding, makes the algorithm faster and further improve its robustness. The second stage of trajectory extraction consists of trajectory evolution based on Adaptive Trajectory Description (*ATD*). Trajectory evolution is the process in which the newly detected blobs are either appended to an existing trajectory or are considered as the beginning of a new trajectory. In the final stage, the extracted motion trajectories are smoothed with Kalman filter. The method of automatically extracting trajectories from video sequences is used to generate MPEG-7 *Motion Trajectory* Descriptors.

- (iv) We explore the problem of extracting semantic description and indexing of the video content. Two approaches in different themes are presented. The first approach belongs to the popular classification based approaches. We provide a video

CHAPTER 6. CONCLUSION

classification system with *HMM* using color and motion features as the observation of the *HMM*. Both features are extracted from the compressed domain and give a good characterization of the video in both the spatial and temporal dimension. The *HMM* video classification has been proved to be a viable method for video topic classification. The feature extraction method in compressed domain is proved to be effective and efficient. The second approach follows a different theme. The semantics are extracted from the scripts directly. The proposed approach for building semantic video index consists of three stages, viz., script extraction, script partition and script vector representation. First, the scripts are extracted from the subtitle files that are available in the DVD/DivX videos. Then, the extracted scripts are partitioned into segments. Finally, the partitioned script segments are converted into a *tfidf* vector based representation, which acts as the semantic index. The efficiency of the semantic index is demonstrated through video retrieval and summarization applications.

6.2 Future Directions

The works presented in this thesis include our researches on various topics of video content description and indexing problem. We have developed methods for segmenting, indexing, searching and summarizing videos at both low level visual/motion features as well as semantic levels. But there are a number of issues for future exploration:

- **Hierarchical structure model for video content representation**

Because of video's length and unstructured format, building a content-based video indexing system is not an easy task. The common approach for video structure analysis is to first divide the video into shot segment, then group the shot segment into scene. Since scene is a group of semantically related shot, it can be regarded

as a semantic segmentation of the video. However, different videos have different semantic structures. A semantically meaningful structure for video content description should be tailored for different videos. For example, the tennis game can be organized into hierarchy as : set $\rightarrow$ game $\rightarrow$ point. Besides, rather than describing the video content with fixed structure, it is desired to have such a structure that could describe the video at different semantic granularity. Ontology has been widely used for conceptualization and description of semantic relationships, i.e. parent-child relation, sibling relation etc. in semantic webs. It is quite promising to utilize specially designed ontology for video structure description (video ontology).

- **Automatic extraction of high level semantic information**

Automatic extraction of high level semantic for the video content is essential for building up a video information system since human uses high-level concepts in everyday life and tends to describe the video content by semantics rather than low level features. But it is a very difficult task to extract meaningful semantics from the video automatically. In general the low level features do not have close link to high level concepts and the low level feature similarity doesn't correspond to high level semantic similarity. Efficient and effective techniques are needed to bridge the gap between the semantics and low level features. These algorithms rely on two important factors: 1) the features and 2) the intelligent usage of the features. The feature must be capable of representing the semantics content of the video. We have proposed several effective features for video content indexing. But to find better features remains an open research problem. The competition for better features will never end. The second factor is how to combine the features intelligently to yield the semantic information. One way is to use the supervised classification to extract the semantic information by classification like the *HMM* video classification described in chapter 5. The other method is to use unsupervised

CHAPTER 6. CONCLUSION

clustering algorithm to propagate the semantic label among the videos. But the research on better modelling of the semantics is a further way to go. For example, the domain knowledge may help to design a better model to represent the semantics with low level features, i.e. graphical models.

- **Multicue and multimodel video content analysis**

We are aware that video is not just a visual medium. It contains text and audio information in addition to visual information, thus, “true” multimedia. Multicue analysis and multimodel processing are always more reliable and comprehensive than a single medium. Integrating audio and text cues into the video content analysis is a promising way to go. In chapter 5, we have described a novel method for video content analysis by mining the script files, which points out a direction that modern information retrieval techniques can be used to understand the video semantics directly from the script files.

In this thesis, we have developed several new tools for multimedia indexing problem. The recent advances, in addition to our works, held our promise for the future of multimedia information system. In particular, the algorithm proposed to analyze the *compressed video* directly makes the application for large video database realizable. The approaches for effective and efficient video indexing and retrieval pave the way for building up a multimedia information system, although there are still many other important issues to be addressed. The open questions are primarily concerned with the semantics indexing of the video content. In the gap between the human awareness of the semantic concept and low level feature lies the essential conundrum of the video indexing. Detection and representing the semantics is a fundamental problem and will allow us to understand the video world. We are immersed in a “sea” of multimedia information and we wish to have a guide of those information.

References

- [1] "MPEG-7: Context, objectives and technical roadmap," *ISO/IEC JTC1/SC29/WG11/N2861*, July 1999.
- [2] S. Jun, K. Yoon, and H. Lee, "Dissolve transition detection algorithm using spatio-temporal distribution of mpeg macro-block types," in *Proc. ACM Multimedia*, 2000, pp. 391 – 394.
- [3] "MPEG-7: Visual part of the XM 4.0," *ISO/IEC JTC1/SC29/WG11/N3068*, December 1999.
- [4] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QIBC project: Querying images by content using color, texture and shape," in *Proceedings of IST/SPIE. on Storage and Retrieval for image and video databases I*, San Jose, CA, February 1993.
- [5] A.P. Pentland, R.W. Picard, and S. Scarloff, "Photobook: tools for content-based manipulation of image databases," in *Proceedings of IST/SPIE. on Storage and Retrieval for image and video databases II*, San Jose, CA, February 1994, pp. 34–47.
- [6] J. R. Smith and S.-F. Chang, "VisualSEEK: a fully automated content-based image query system," in *Proceedings of ACM Multimedia*, Boston, MA San Jose, CA, November 1996.
- [7] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Sharad Mehrotra, and Thomas Huang, "Supporting similarity queries in MARS," in *Proceedings of ACM Multimedia*, Seattle, WA, November 1997.

REFERENCES

- [8] S.F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "Videoq: an automated content-based video search system using visual cues," in *Proceedings of ACM Multimedia*, Seattle, WA, November 1997.
- [9] H. Zhang, A. Kankanhalli, and S. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Systems*, vol. 1, pp. 10–28, 1993.
- [10] J. Oh and K. A. Hua, "An efficient and cost-effective technique for browsing and indexing large video databases," in *Proceedings of 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, TX, May 2000, pp. 415–426.
- [11] "Introduction to MPEG-7," *ISO/IEC JTC1/SC29/WG11/N3545*, July 2000.
- [12] H. Zhang and H. Liu, "A hierarchical organization scheme for video data," *Pattern Recognition*, vol. 35, pp. 2381–2387, 2002.
- [13] Alan Hanjalic, "Shot-boundary detection: Unraveled and resolved?," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 2, pp. 90–105, Feb. 2002.
- [14] R. M. Ford, C. Robson, D. Temple, and M. Gerlach, "Metrics for shot boundary detection in digital video sequences," *Multimedia Systems*, vol. 8, no. 1, pp. 37–46, Jan 2000.
- [15] R. Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Proceedings of SPIE conference on Storage and Retrieval for Image and Video Databases VII*, San Jose, CA, Jan. 1999, vol. 3656, pp. 290–301.
- [16] I. K. Sethi and N. V. Patel, "Statistical approach to scene change detection," in *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*, 1995, pp. 329–338.
- [17] C.-C. Lo and S.-J. Wang, "A histogram-based moment-preserving clustering algorithm for video segmentation," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2209–2218, October 2003.

REFERENCES

- [18] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proceedings of ACM Multimedia 95*, "San Francisco, CA", Nov. 1995, vol. 1, pp. 10–28.
- [19] W. J. Heng and K. N. Ngan, "An object-based shot boundary detection using edge tracing and tracking," *Journal of Visual Communications and Image Representation*, vol. 12, no. 3, pp. 217–239, sep 2001.
- [20] J. Feng, K.-T. Lo, and H. Mehrpour, "Scene change detection algorithm for MPEG video sequence," in *Proceedings of IEEE International Conference on Image Processing*, 1996, vol. 1, pp. 821–824.
- [21] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Transactions On Circuit and System for Video Technology*, vol. 5, pp. 553–544, Dec. 1995.
- [22] Jianhao Meng, Yujen Juan, and Shih-Fu Chang, "Scene change detection in a MPEG video sequence," in *Proceedings of SPIE Conference on Multimedia Computing and Networking*, "San Jose, CA", Feb. 1995, vol. 2417, pp. 180–191.
- [23] M. M. Yeung, B.-L. Yeo, and B. Liu, "Segmentation of video by clustering and graph analysis," *Computer Vision and Image Understanding*, vol. 71, no. 1, 1998.
- [24] Alan Hanjalic, Reginald L. Lagendijk, and Jan Biemond, "Automated segmentation of movies into logical story units," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 580–588, 1999.
- [25] Y. Rui, T. S. Huang, and S. Mehrotra, "Constructing table-of-content for videos," *ACM Multimedia Systems Journal, Special Issue Multimedia Systems on Video Libraries*, sep 1999.
- [26] J. Zhou and W. Tavanapong, "Shotweave: A shot clustering technique for story browsing for large video databases," in *International Workshop on Multimedia Data Document Engineering*, Mar. 2002.

REFERENCES

- [27] B. Adams, C. Dorai, and S. Venkatesh, "Towards automatic extraction of expressive elements from motion pictures: tempo," in *IEEE International Conference on Multimedia and Expo*, 2000, pp. 641–644.
- [28] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing on MC-DCT compressed video," *IEEE Journal on Selected Areas in Communication, Special Issue on Intelligent Signal Processing*, pp. 1–11, January 1995.
- [29] B.-L. Yeo and B. Liu, "On the extraction of dc sequence from MPEG video," in *Proceedings of IEEE International Conference on Image Processing*, 1995, vol. 2, pp. 260–263.
- [30] K. Shen and E. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proceedings of IEEE International Conference on Image Processing*, 1995, vol. 2, pp. 252–255.
- [31] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, 1995.
- [32] C. S. Won, D. K. Park, I. Y. Na, and S.-J. Yoo, "Efficient color feature extraction in compressed video," in *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, San Jose, CA, January 1999, pp. 677–686.
- [33] Y.-P. Tan, S.R. Kulkarni, and P.J. Ramage, "A framework for measuring video similarity and its application to video query by example," in *Proceedings of IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.
- [34] Y. S. Ho and A. Gersho, "Classified transform coding of images using vector quantization," in *Proceedings of IEEE ICASSP 89*, 1989, pp. 1890–1893.
- [35] O. Bao, J. Lay, and L. Guan, "Compressed domain video parsing using energy histogram of the lower frequency dct coefficients," in *Proceedings of SPIE Conference on Storage and Retrieval for Multimedia Databases 2000*, San Jose, CA, January 2000, pp. 293–300.

REFERENCES

- [36] Y. Zhong, H.-J. Zhang and A.-K. Jain, "Automatic caption localization in compressed video," in *Proceedings of IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.
- [37] B. Shen and I. K. Sethi, "Direct feature extraction from compressed images," in *Proceedings of SPIE Conference on Storage and Retrieval for Multimedia Databases IV*, January 1996.
- [38] B. Shen and I. K. Sethi, "Convolution-based edge detection for image/video in block dct domain," *Journal of Visual Communications and Image Representation*, vol. 4, no. 7, pp. 441–423, December 1996.
- [39] B. C. Song and J. B. Ra, "Fast edge map extraction from MPEG compressed video data for video parsing," in *Proceedings of SPIE Conference on Storage and Retrieval for Multimedia Databases VII*, San Jose, CA, January 1999, pp. 710–721.
- [40] F. Arman, A. Hsu, and M.-Y. Chiu, "Image processing on compressed data for large video databases," in *Proceedings of International Conference on Pattern Recognition*, 1996.
- [41] E. Ardizzone, M. L. Cascia, A. Avanzato, and A. Bruna, "Video indexing using MPEG motion compensation vectors," in *Proceedings of SPIE International Conference on Multimedia Computing and Systems*, 1999.
- [42] N. Domitrova and F. Golshani, "Motion recovery for video content analysis," *ACM Transactions on Information Systems*, vol. 4, no. 13, pp. 408–439, October 1995.
- [43] A. Divakaran and H. Sun, "A descriptor for spatial distribution of motion activity," in *Proceedings of SPIE Conference on Storage and Retrieval for Multimedia Databases 2000*, San Jose, CA, January 2000, pp. 392–398.
- [44] W. Wolf, "Key frame selection by motion analysis," in *Proceedings of IEEE ICASSP 97*, Hong Kong, June 1997, vol. 2, pp. 1488–1491.
- [45] D. Swanberg, C.-F. Shu, and R. Jain, "Knowledge guided parsing in video databases," in *Electronic Imaging: Science and Technology*, Feb. 1993.

REFERENCES

- [46] C.-Y. Lin, B. L. Tseng, and J. R. Smith, "VideoAnnEx: IBM MPEG-7 annotation tool for multimedia indexing and concept learning," in *IEEE International Conference on Multimedia & Expo*, Baltimore, USA, July 2003.
- [47] W.-S. Zhou, A. Vellaikal, and C. Kuo, "Rule-based video classification system for basketball video indexing," in *Proceedings of ACM multimedia workshop*, 2000, pp. 213–216.
- [48] G. Wei, L. Agnihotri, and N. Dimitrova, "TV program classification based on face and text processing," in *Proceedings of IEEE multimedia and Expo*, New York, 2000.
- [49] L. Cheng, M. Drew, and J. Au, "An automatic video classification system based on a combination of hmm and video summarization," *International Journal of Smart Engineering System Design*, vol. 5, no. 1, pp. 33–45, 2003.
- [50] J. Huang, Z. Liu, and Y. Wang, "Joint video scene segmentation and classification based on hidden markov model," in *Proceedings of Advances in Digital Libraries Conference*, Santa Barbara, CA, 1998.
- [51] Ahmet Ekin, A. Murat Tekalp, and Rajiv Mehrotra, "Automatic soccer video analysis and summarization," *IEEE Trans. on Image Processing*, vol. 12, no. 8, pp. 796–807, July 2003.
- [52] Hong Lu and Yap-Peng Tan, "Content-based sports video analysis and modeling," in *Proceedings of Seventh International Conference on Control, Automation, Robotics And Vision*, December 2002, pp. 1198–1203.
- [53] K. E. Maleh, M. Klein, G. Petrucci, and P. Kabal, "Speech/ music discrimination for multimedia applications," in *IEEE ICASSP*, 2000.
- [54] D. Yow, B-L Yeo, M. Yeung, and B. Liu, "Analysis and presentation of soccer highlights from digital video," in *Proc. ACCV*, 1995.
- [55] C-W. Chang and S-Y. Lee, "A video information system for sport motion analysis," *Journal of Visual Languages and Computing*, vol. 8, pp. 265–287, 1998.

REFERENCES

- [56] E. Katz, *The film encyclopedia*, 2nd ed., New York: Harper Collins, 1994.
- [57] J. R. Kender and B.-L. Yeo, "Video scene segmentation via continuous video coherence," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, jun 1998, pp. 367–373.
- [58] B.-T. Truong, S. Vekatesh, and C. Dorai, "Scene extraction in motion picture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 5–15, jan 2003.
- [59] M. A. Robertson and R. L. Stevenson, "Restoration of compressed video using temporal information," in *Proceedings of SPIE conference on Visual Communications and Image Processing*, San Jose, CA, 2001, vol. 4310, pp. 21–29.
- [60] B. Gunturk, Y. Altunbasak, and R. Mersereau, "Super-resolution reconstruction of compressed video using transform-domain statistics," *IEEE Transactions on Image Processing*, vol. 13, no. 1, pp. 33–43, jan 2004.
- [61] D. Schonfeld and D. Lelescu, "VORTEX: Video retrieval and tracking from compressed multimedia databases - multiple object tracking from MPEG-2 bitstream," *Journal of Visual Communications and Image Representation, Special Issue on Multimedia Database Management*, vol. 11, pp. 154–182, 2000.
- [62] H. Wang, A. Divakaran, A. Vetro, S.-F. Chang, and H. Sun, "Survey of compressed-domain features used in audio-visual indexing and analysis," *Journal of Visual Communication and Image Representation*, vol. 14, no. 2, pp. 50–183, June 2003.
- [63] A. Yilmaz and M. Shah, "Shot detection using principal coordinate system," in *Proc. ASTED Int. Conf. Internet and Multimedia Systems and Applications*, Nov. 2000.
- [64] R. Brunelli, O. Mich, and C. M. Modena, "A survey on the automatic indexing of video data," *Journal of Visual Communications and Image Representation*, vol. 10, no. 2, pp. 78–112, jun 1999.

REFERENCES

- [65] A. Hampapur, R. Jain, and T. Weymouth, "Production model based digital video segmentation," *Journal of Multimedia Tools and Applications*, vol. 1, pp. 255–258, 1995.
- [66] W. A. C. Fernando, C. N. Cannagarajah, and D. R. Bull, "Automatic detection of fade-in and fade-out in video sequences," in *Proc. IEEE Int. Symp. Circuits systems*, 1999, vol. 4, pp. 255–258.
- [67] S. J. Golin, "New metric to detect wipes and other gradual transitions in video," in *Proc. SPIE conf. on visual communication and image processing*, 1999, pp. 1464–1474.
- [68] J. Boreczky and L. Wilcox, "A hidden markov model framework for video segmentation using audio and image features," in *Proc. ICASSP*, 1998, pp. 3741–3744.
- [69] W. J. Heng and K. N. Ngan, "Post shot boundary detection technique: Transition type-independent shot boundary refinement," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 434–445, December 2002.
- [70] Jianhao Meng, Yujen Juan, and Shih-Fu Chang, "Scene change detection in a MPEG video sequence," in *Proceedings of SPIE Conference on Multimedia Computing and Networking*, San Jose, CA, Feb 1995, vol. 2417, pp. 180–191.
- [71] B. Haskell, A. Puri, and A. N. Netravali, *Digital video : an introduction to MPEG-2*, New York : Chapman I& Hall, 1997.
- [72] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Transactions On Circuits and Systems for Video Technology.*, vol. 5, pp. 553–544, Dec. 1995.
- [73] Arijon Daniel, *Grammar of the Film Language*, London: Focal Press, 1976.
- [74] C. Saraceno and R. Leonardi, "Identification of story units in audio-visual sequences by joint audio and video processing," in *Proceeding of International Conference on Image Processing*, Chicago, IL, USA, 1998, pp. 358–362.

REFERENCES

- [75] Z. Rasheed and M. Shah, "Scene detection in hollywood movies and TV shows," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 343–348.
- [76] J. Nam and A. H. Tewfik, "Combined audio and visual streams analysis for video sequence segmentation," in *Proceedings of IEEE ICASSP-97*, Munich, Germany, Apr 1997, vol. 4, pp. 2665–2668.
- [77] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7 : multimedia content description interface*, New York : Chichester : Wiley, New York, NY, USA, 2002.
- [78] G. Robertson, J. Machinlay, and S. Card, "Cone trees: Animated 3d visualization of hierachical information," in *Proceedings of ACM SIGCHI conference on Human Factors in Computing System'91*, 1991.
- [79] J. Heuer and A. Kaup, "Global motion estimation in image sequences using robust motion vector field segmentation," in *Proc. ACM Multimedia*, 1999, pp. 261–264.
- [80] Y.-P. Tan, D. Saur, S. Kulkarni, and P. Ramadge, "Rapid estimation of camera motion from compressed video with application to video annotation," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 10, no. 1, pp. 133–146, Feb. 2000.
- [81] Y. Su, M.-T. Sun, and V. Hsu, "Global motion estimation from coarsely sampled motion vector field and the applications," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 2, pp. 232–242, Feb. 2005.
- [82] E. Sahouria and A. Zakhori, "Content analysis of video using principle components," *IEEE Transactions on Circuit and System of Video Technology*, vol. 9, pp. 1290–1298, Dec. 1999.
- [83] S. Smoliar and H. Zhang, "Content-based video indexing and retrieval," *IEEE Transactions on Multimedia*, vol. 1, pp. 62–72, 1994.
- [84] Y. Deng, B. S. Manjunath, C. Kenney, M. S. Moore, and H. Shin, "An efficient color representation for image retrieval," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 140–147, 2001.

REFERENCES

- [85] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.
- [86] Y. Rui, A. C. She, and T. S. Huang, "Modified fourier descriptors for shape representation a practical approach," in *Proceedings of First International Workshop on Image Databases and Multimedia search*, 1996.
- [87] D. K. Park, Y. S. Jeon, and C. S. Won, "Efficient use of local edge histogram descriptor," in *Proceedings of the ACM workshops on Multimedia*, Los Angeles, California, USA, 2000, pp. 51–54.
- [88] C.-W. Ngo, T.-C. Pong, and H.-J. Zhang, "On clustering and retrieval of video shots through temporal slices analysis," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 446–458, Dec. 2002.
- [89] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A fully automatic content-based video search engine supporting multiple object spatio-temporal queries," *IEEE Transactions on Circuit and System of Video Technology*, vol. 8, pp. 602–615, Sept. 1998.
- [90] R. Fablet, P. Bouthemy, and P. Perez, "Statistical motion-based video indexing and retrieval," in *International Conference on Content Based Multimedia Information Access*, 2000, pp. 602–619.
- [91] MPEG, *Multimedia content description interface-part 8: extraction and use of MPEG-7 descriptors*, ISO/IEC 15938-8:2002, 2002.
- [92] W. Wolf, "Key frame selection by motion analysis," in *Proceedings of the ICASSP*, Atlanta, USA, Apr. 1996, pp. 1228–1231.
- [93] A. Divakaran, R. Regunathan, and K. A. Peker, "Video summarization using descriptors of motion activity: A motion activity based approach to key-frame extraction from video shots," *Journal of Electronic Imaging*, vol. 10, pp. 909–916, Oct. 2001.

REFERENCES

- [94] H. C. Lee and S. D. Kim, "Rate-driven key frame selection using temporal variation of visual content," *IEE Electronics Letters*, vol. 38, no. 5, pp. 217–218, Feb. 2002.
- [95] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih, "Image indexing using color correlograms," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 762–768.
- [96] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.
- [97] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, pp. 195–202, 1984.
- [98] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons Inc., New York, USA, 1973.
- [99] K. P. Burnham, D. R. Anderson, and G. C. White, "Evaluation of the Kullback-Leibler discrepancy for model selection in open population capture-recapture models," *Biometrical Journal*, vol. 36, pp. 299–315, 1994.
- [100] N. Christianini and J. Shawe-Taylor, *An introduction to Support Vector Machines*, Cambridge University Press, 2000.
- [101] B.-L. Yeo and B. Liu, "On the extraction of DC sequence from MPEG compressed video," in *Proceedings of the IEEE ICIP*, Washington, DC, USA, 1995.
- [102] S. Haykin, *Adaptive Filter Theory*, NJ: PrenticeHall, 2002.
- [103] T. Liu, H. Zhang, and F. Qi, "A novel video key-frame-extraction algorithm based on perceived motion energy model," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, pp. 1006–1013, Oct. 2003.
- [104] M. Cooper and J. Foote, "Discriminative techniques for keyframe selection," in *IEEE International Conference on Multimedia & Expo*, July 2005.

REFERENCES

- [105] Y. Zhuang, Y. Rui, T. Huang, and S. Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *IEEE International conference on Image Processing*, Oct. 1998, vol. 1, pp. 866–870.
- [106] C.-B. Shim and J.-W. Chang, "Efficient similar trajectory-based retrieval for moving objects in video databases," in *Proceedings of the International Conference on Image and Video Retrieval(CVIR)*, 2003, pp. 163–173.
- [107] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Multiple video object tracking in complex scenes," in *Proceedings of ACM Multimedia 2002*, December 2002, pp. 523–532.
- [108] B. Bascle and R. Deriche, "Region tracking through image sequences," in *Proceedings of 5th International Conference on Computer Vision*, Cambridge, MA, 1995.
- [109] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of nonrigid objects using mean shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head, SC, 2000, vol. 2, pp. 142–149.
- [110] Y. Chen, Y. Rui, and T.-S. Huang, "JPDAF based HMM for real-time contour tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, 2001, vol. 1, pp. 543–550.
- [111] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Analysis Machine Intell*, vol. 25, no. 5, pp. 564–575, 2003.
- [112] V. Tovinkere and R. J. Qian, "Detection of semantic events in soccer games: towards a complete solution," in *Proceedings of IEEE International Conference on Multimedia and Expo*, August 2001.
- [113] A. Ekin and A. M. Tekalp, "A framework for analysis and tracking of soccer video," *Visual Communication and Image Processing(VCIP)*, Jan 2002.
- [114] G. S. Pingali, Y. Jean, and I. Carlbom, "Real time tracking for enhanced tennis broadcasts," in *Proceedings of the IEEE CVPR*, 1998, pp. 260–265.

REFERENCES

- [115] A. Gueziec, "Tracking pitches for broadcast television," *IEEE Computer*, vol. 35, no. 3, pp. 38–43, March 2002.
- [116] S. Intille and A. Bobick, "Recognizing planned, multi-person action," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 414–445, March 2001.
- [117] P. Xu, L. Xie, S.-F. Chang, A. Divakaran, A. Vetro, and H. Sun, "Algorithms and system for segmentation and structure analysis in soccer video," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Tokyo, August 2001.
- [118] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610–621, November 1973.
- [119] L. H. Siew, R. M. Hodgson, and E. J. Wood, "Texture measures for carpet wear assessment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 1, pp. 92–105, January 1988.
- [120] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.
- [121] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of International Conference on Computer Vision*, Bombay, India, Jan. 1998.
- [122] T. Joachims, "Text categorization with support vector machine," in *Proceedings of the European Conference on Machine Learning*, 1998.
- [123] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, February 1998.
- [124] Christoph Stiller and Janusz Konrad, "Estimating motion in image sequences," *IEEE Signal Processing Magazine*, vol. 16, no. 4, pp. 70–91, July 1999.
- [125] Richard Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 22–30, March 1996.

REFERENCES

- [126] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," *Signal Processing*, vol. 31, pp. 165–180, 1993.
- [127] S. G. Wilson, *Digital Modulation and Coding*, Prentice Hall, 1996.
- [128] A. Hauptmann and M. Witbrock, "Story segmentation and detection of commercials in broadcast news video," in *Proceedings of Advances in Digital Libraries Conference*, Santa Barbara, CA, 1998.
- [129] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of IEEE*, vol. 77, no. 2, February 1989.
- [130] M. Drew, J. Wei, and Z.-N. Li, "Illumination-invariant image retrieval and video segmentaion," *Pattern Recognition*, vol. 32, pp. 1369–1388, 1999.
- [131] "<http://www.doom9.org/dvobsb.htm>," .
- [132] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 301–328, 1995.
- [133] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, July 1980.
- [134] M. Berry, Z. Drmavc, and E. Jessup, "Matrices, vector spaces, and information retrieval," *SIAM Review*, vol. 41, no. 2, pp. 335–362, June 1999.
- [135] C. Snoek and M. Worring, "Multimodal video indexing: A review of the state-of-the-art," *Journal of Multimedia Tools and Applications*, vol. 25, no. 1, pp. 5–35, 2005.

List of Publications

Journals

1. H. Yi, D. Rajan and L.-T. Chia, "A Motion based Scene Tree for Browsing and Retrieval of Compressed Videos", *Information Systems (accepted)*.
2. S. Liu, M. Xu, H. Yi, L.-T. Chia and D. Rajan, "Multi-modal Semantic Analysis and Annotation for Basketball Video", *EURASIP Journal on Applied Signal Processing (accepted)*.
3. H. Yi, D. Rajan and L.-T. Chia, "A New Motion Histogram to Index Motion Content in Video Segments", *Pattern Recognition Letters*, Vol. 26, No. 9, pp. 1221-1231, Jul, 2005.
4. H. Yi, D. Rajan and L.-T. Chia, "Automatic Generation of MPEG-7 Compliant XML Document for Motion Trajectory Descriptor In Sports Video", *Multimedia Tools and Application Journal*, Vol. 26, No. 5, pp. 189-204, Apr, 2005.
5. H. Yi, D. Rajan and L.-T. Chia, "A Motion based Scene Tree for Compressed Video Content Management", *Image and Vision Computing Journal (accepted)*.

Conferences

1. S. Liu, H. Yi, L.-T. Chia and D. Rajan, "Adaptive Hierarchical Multi-class SVM Classifier for Texture-based Image Classification", In *Proc. of IEEE ICME*, Amsterdam, Netherlands, Jul, 2005.
2. H. Yi, D. Rajan and L.-T. Chia, "Global Motion Compensated Key Frame Extraction from Compressed Videos", In *Proc. of IEEE ICASSP*, Philadelphia, PA, USA, Mar,

PUBLICATIONS

2005.

3. H. Yi, D. Rajan and L.-T. Chia, "ARIRS: Association Rule Based Image Retrieval System", In *Proc. of International Workshop on Advanced Image Technology(IWAIT'05)*, Jeju Island, Korea, Jan, 2005.
4. H. Yi, D. Rajan and L.-T. Chia, "A Motion based Scene Tree for Browsing and Retrieval of Compressed Videos", In *Proc. of ACM Multimedia Database04*, Arlington, VA, USA, Nov, 2004.
5. H. Yi, D. Rajan and L.-T. Chia, "Automatic Extraction of Motion Trajectories in Compressed Sports Videos", In *Proc. of ACM Multimedia 2004*, New York, NY, USA, Oct, 2004.
6. H. Yi, D. Rajan and L.-T. Chia, "Semantic Video Indexing and Summarization using Subtitles", In *Proc. of IEEE Pacific-Rim Conference on Multimedia*, Tokyo, Japan, Nov 30-Dec 3, 2004.
7. S. Liu, H. Yi, L.-T. Chia and D. Rajan, "Semantic Analysis of Basketball Video Using Motion Information", In *Proc. of IEEE Pacific-Rim Conference on Multimedia*, Tokyo, Japan, Nov 30-Dec 3, 2004.
8. H. Yi, D. Rajan and L.-T. Chia, "Motion Histogram: A New Motion Feature to Index Motion Content in Video Segment", In *Proc. of International Conference on Knowledge and Information Engineering*, Las Vegas, USA, Jun, 2004.
9. H. Yi, D. Rajan and L.-T. Chia, "An Efficient Video Classification System Based On HMM In Compressed Domain", In *Proc. of IEEE Pacific-Rim Conference on Multimedia*, Singapore, Dec, 2003.
10. H. Yi, D. Rajan and L.-T. Chia, "Automatic Generation of MPEG-7 Compliant XML Document for Motion Trajectory Descriptor In Sports Video", In *Proc. of ACM Multimedia Database'03*, New Orleans, Louisiana, USA, Nov, 2003.
11. H. Yi, D. Rajan and L.-T. Chia, "A Unified Approach to Detection of Shot Boundaries and Subshots In Compressed Video", In *Proc. of IEEE ICIP*, Barcelona, Spain, Sep, 2003.