Τεχνικές Μηχανικής Μάθησης για Διαχείριση Γνώσης σε Πολυμεσικά Δεδομένα

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Βασίλειο Χασάνη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

Ιούνιος 2009

Τριμελής Συμβουλευτική Επιτροπή

- Αριστείδης Λύκας, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων).
- Νικόλαος Γαλατσάνος, Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.

Επταμελής Εξεταστική Επιτροπή

- Αριστείδης Λύκας, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων).
- Νικόλαος Γαλατσάνος, Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Στέφανος Κόλλιας, Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.
- Ανδρέας Σταφυλοπάτης, Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.
- Ισαάκ Λαγαρής, Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Λυσίμαχος Κόντης, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.

This thesis is part of the 03ED375 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (20% from the Greek Ministry of Development-General Secretariat of Research and Technology and 80% from E.U.-European Social Fund).

Acknowledgments

I would like to sincerely thank my advisor Dr. Aristidis Likas, Associate Professor at the Department of Computer Science, University of Ioannina, and the members of the advisory committee Dr. Nikolaos Galatsanos, Professor at the Department of Electrical and Computer Engineering, University of Patras and Dr. Konstantinos Blekas, Assistant Professor at the Department of Computer Science, University of Ioannina, for their full support and excellent cooperation.

I would also like to thank Dr. Isaac Lagaris, Professor and Dr. Christophoros Nikou, Lecturer, both at the Department of Computer Science, University of Ioannina, for their valuable comments and suggestions.

I would like especially to thank Dr. A. Likas and Dr. N. Galatsanos for the guidance and assistance they provided during this dissertation, their valuable counseling during my studies and their actions for the funding of my research from the research program PENED 2003 of the Greek General Secretariat of Research and Technology.

I would also like to thank Dr. D. Tzikas and the PhD candidates C. Voglis, A. Kalogeratos and G. Tzortzis from the Department of Computer Science of University of Ioannina for their contribution to the creation of an exceptional research environment. Finally, I would like to thank my parents Thomas and Froso, my brother Constantinos, Maria and Stefanos for their support and understanding during my work for this thesis.

Contents

List of Figures

List of Tables

Εκτενης Περιληψη Στα Ελληνικα

Χασάνης Βασίλειος του Θωμά και της Φρόσως.

Διδακτορικό Δίπλωμα, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ελλάδα. Ιούνιος, 2009.

Τίτλος: Τεχνικές Μηχανικής Μάθησης για Διαχείριση Γνώσης σε Πολυμεσικά Δεδομένα.

Επιβλέποντας: Αριστείδης Λύκας.

Η Διατριβή εστιάζεται στα ζητήματα της *κατάτμησης* (video segmentation) και *αναπαράστασης βίντεο* (video representation) με τη χρήση *τεχνικών μηχανικής μάθησης* (machine learning techniques), καθώς και στην εφαρμογή των προτεινόμενων μεθόδων στα προβλήματα της *περίληψης αμοντάριστου βίντεο* (video rushes summarization) και *παρακολούθησης μέσω βίντεο* (video surveillance).

Καταρχήν εξετάζεται το χαμηλότερο επίπεδο κατάτμησης βίντεο που σχετίζεται με το *πρόβλημα της ανίχνευσης των ορίων των πλάνων* (shot boundary detection). Προτείνεται μια μεθοδολογία μάθησης με επίβλεψη που χρησιμοποιεί ένα σύνολο χαρακτηριστικών που έχουν σχεδιαστεί ειδικά για να εκφράσουν τη συνάφεια των εικονοπλαισίων σε μια γειτονιά. Ένα *σύστημα ταξινόμησης* SVM (Support Vector Machines) εκπαιδεύεται τόσο για τον εντοπισμό των ορίων των πλάνων όσο και για το χαρακτηρισμό των μεταβάσεων μεταξύ των πλάνων.

Στη συνέχεια προτείνεται ένας αλγόριθμος *εξαγωγής χαρακτηριστικών εικονοπλαισίων* (key-frame extraction) ενός πλάνου που βασίζεται σε τεχνικές *φασματικής ομαδοποίησης* (spectral clustering). Η προτεινόμενη μέθοδος παρέχει επίσης μια εκτίμηση του αριθμού των χαρακτηριστικών εικονοπλαισίων εξετάζοντας τις ιδιοτιμές του πίνακα ομοιότητας μεταξύ των εικονοπλαισίων ενός πλάνου. Τα εξαγόμενα χαρακτηριστικά εικονοπλαίσια είναι μοναδικά,

μη επαναλαμβανόμενα και συνοψίζουν ικανοποιητικά το περιεχόμενο του κάθε πλάνου.

Κατόπιν προτείνεται ένας αλγόριθμος για την *κατάτμηση ενός βίντεο σε σκηνές* (video scene segmentation). Τα πλάνα ενός βίντεο αρχικά ομαδοποιούνται σε ομάδες χρησιμοποιώντας τεχνικές φασματικής ομαδοποίησης. Στη συνέχεια, σε κάθε πλάνο αντιστοιχίζεται μία ετικέτα (label) με βάση την ομάδα στην οποία ανήκει. Για την ανίχνευση των ορίων των σκηνών αναζητούμε αλλαγές στο μοτίβο των ετικετών των πλάνων. Για αυτό το λόγο χρησιμοποιείται ένας αλγόριθμος *ευθυγράμμισης ακολουθιών* (sequence alignment algorithm), ο οποίος συγκρίνει ακολουθίες από συμβολικές ετικέτες πλάνων. Τα όρια των σκηνών εντοπίζονται στα χρονικά σημεία για τα οποία ο αλγόριθμος ευθυγράμμισης ακολουθιών παρέχει χαμηλές τιμές.

Στη συνέχεια προτείνεται μία μέθοδος για την *υψηλού επιπέδου κατάτμηση μίας ταινίας* (high-level movie segmentation), δηλ. την κατάτμηση σε σκηνές και κεφάλαια (chapters). Η μέθοδος χρησιμοποιεί *περιγραφείς τοπικών χαρακτηριστικών* (local invariant descriptors) για την αναπαράσταση των πλάνων με μια σημασιολογική περιγραφή. Πιο συγκεκριμένα, ένα *λεξιλόγιο οπτικών λέξεων* (visual word vocabulary) παράγεται από τους περιγραφείς και ένα πλάνο αναπαρίσταται από ένα ιστόγραμμα που εκφράζει τη συχνότητα εμφάνισης των περιγραφέων του πλάνου σε κάθε λέξη του λεξιλογίου. Υιοθετώντας μία προσέγγιση από το πεδίο της κατάτμησης κειμένων, αυτά τα σημασιολογικά ιστογράμματα ομαλοποιούνται σε σχέση με γειτονικά ιστογράμματα χρησιμοποιώντας μία γκαουσσιανή συνάρτηση πυρήνα (gaussian kernel). Τα τοπικά μέγιστα της διαφοράς των ομαλοποιημένων ιστογραμμάτων σε διάφορες χρονικές κλίμακες αντιστοιχούν στα όρια των σκηνών/κεφαλαίων.

Ένα άλλο ζήτημα που μελετάται στη διατριβή είναι το πρόβλημα της *περίληψης αμοντάριστου βίντεο* (video rushes summarization). Το αμοντάριστο βίντεο περιέχει αρκετή περιττή πληροφορία, όπως μονόχρωμα εικονοπλαίσια, αλλά και επαναλαμβανόμενα πλάνα. Ο στόχος είναι η απομάκρυνση των ανεπιθύμητων εικονοπλαισίων αλλά και της επαναλαμβανόμενης πληροφορίας. Περιγραφείς τοπικών χαρακτηριστικών και *ιστογράμματα διεύθυνσης ακμών* (edge direction histograms) χρησιμοποιούνται για την απομάκρυνση των ανεπιθύμητων εικονοπλαισίων. Στη συνέχεια, ορίζεται ένα μέτρο ομοιότητας μεταξύ πλάνων βασισμένο στην οπτική ομοιότητα των χαρακτηριστικών εικονοπλαισίων τους. Με βάση αυτό το μέτρο

παρόμοια πλάνα ομαδοποιούνται σε ομάδες και ένα μόνο πλάνο από κάθε ομάδα κρατείται ως αντιπροσωπευτικό, απαλείφοντας έτσι επαναλαμβανόμενα πλάνα. Τέλος, επιλέγοντας ένα αριθμό εικονοπλαισίων πριν και μετά από τα χαρακτηριστικά εικονοπλαίσια κάθε αντιπροσωπευτικού πλάνου, καταλήγουμε στην τελική περίληψη του αρχικού ακατέργαστου βίντεο.

Τέλος στη διατριβή προτείνεται ένας αλγόριθμος για *ανίχνευση και χαρακτηρισμό γεγονότων* (event detection and classification) σε ακολουθίες βίντεο παρακολούθησης (video surveillance). Χρησιμοποιούνται περιγραφείς τοπικών χαρακτηριστικών για την κατάτμηση του βίντεο σε τμήματα/γεγονότα, καθώς και για τον χαρακτηρισμό των εικονοπλαισίων κάθε γεγονότος με ιστογράμματα οπτικών λέξεων. Με βάση τα παραγόμενα ιστογράμματα ορίζονται δύο μέτρα ανομοιότητας μεταξύ γεγονότων και εφαρμόζονται τεχνικές μηχανικής μάθησης για την κατηγοριοποίηση γεγονότων σε προκαθορισμένες κατηγορίες.

Abstract

Chasanis Vasileios, T.

PhD, Computer Science Department, University of Ioannina, Greece. June, 2009.

Title: Machine Learning Techniques for Multimedia Knowledge Management.

Supervisor: Aristidis Likas.

In this thesis, we study the *video segmentation* and *representation* problems using *machine learning techniques*. We also consider two video analysis applications such as *video rushes summarization* and *video surveillance*.

At first we examine the first level of video segmentation which is the *shot boundary detection problem*. We propose a supervised learning methodology that uses a set of features that are specifically designed to capture the variation between adjacent frames and the contextual information in a neighborhood of frames. A *support vector machine* (SVM) classifier is trained both to locate shot boundaries and characterize transition types in videos with different characteristics.

Next we present a *key-frame extraction* algorithm that is based on *spectral clustering* and the *fast global k-means* algorithm. The proposed method also provides an estimation of the number of clusters using elements from *spectral graph theory*. The extracted key-frames are unique, non-repetitive and summarize the video shot content. This is also indicated from numerical experiments, where appropriate quality measures are computed.

Then a *video scene segmentation* algorithm is proposed. Shots are clustered into groups using the improved spectral clustering algorithm and a label is assigned to each shot according to the group that it belongs to. In order to detect scene boundaries, we search for changes in the patterns of shot labels. Therefore, a *sequence alignment* algorithm is

applied to compare sequences of shot labels. Scene boundaries correspond to low score values of the sequence alignment algorithm. Numerical experimental results on several videos indicate that the proposed method accurately detects most scene boundaries, while providing a good trade off between recall and precision.

Next we present a *high-level movie segmentation* algorithm. *Local invariant descriptors* are employed to provide a semantical shot representation through visual word histograms. The visual word histograms of shots are temporally smoothed using a *gaussian kernel* with respect to neighboring histograms to preserve useful contextual information. The semantic smoothing process at different time scales results in efficient movie segmentation at different high-levels, such as scenes and chapters.

Based on the above methods for video segmentation and key-frame extraction, a *video rushes summarization* algorithm has been proposed. *Edge direction histograms* and local invariant descriptors are first employed to remove useless frames from the initial video. Next in order to remove repetitive shots, a shot similarity metric is computed based on a sequence alignment algorithm on the key-frames of the shots under comparison. Finally, by selecting a number of frames around each key-frame, the final video summary is generated constituting an efficient representation of the initial video.

Finally an *event detection* and *classification* algorithm is proposed for video surveillance sequences. The method employs local invariant descriptors to segment the video sequence in segments/events and describe each video segment with a visual word histogram. Two different dissimilarity metrics between events are defined based on the computed visual word histograms and machine learning techniques are employed to classify events into predefined categories.

Chapter 1

Introduction

1.1 Video Indexing and Retrieval

In recent years, there has been a significant increase in the availability of high quality digital video as a result of the expansion of broadband services and the availability of large volume digital storage devices. Due to the extended use of videos in several applications such as distance learning, video surveillance, internet-TV, digital libraries and video on demand, as well as the the thousands of produced movies and documentaries, a large amount of video information is added to the repositories every year. Consequently, there has been an increase in the need to access this huge amount of information and a great demand for techniques that will provide efficient indexing, browsing and retrieving of video data.

Content-based video database modeling, representation, summarization, indexing, retrieval, navigation and browsing have emerged as challenging and important problems in computer vision and database management. *Video structure parsing* is an initial step to organize the content of videos. Video data are typically organized in an hierarchical structure [44, 32]. Such structure is obtained through *video segmentation* into meaningful pieces of information, either *physical* (*shots*) or *semantical* (*scenes* or *chapters*).

Another important problem is *video browsing* and *summarization*, i.e. the representation of the whole content of a video in a form as concise and accessible as possible [92, 59]. The need behind such representation is that users are often interested to inspect only part of the video, and it should be possible to locate such segment without watching the whole video. The most common approach is to represent the shots or the scenes of each video with a sequence of *key-frames*, which are the most representative frames of the video content. In this way, few images can summarize several minutes of a video. Moreover, a user can access the video and search its content by referring to the key-frames without watching the whole content.

A successful segmentation and representation of a video is important for *video classification*, *indexing* and *retrieval* [25]. Automatic or semi-automatic methods have been proposed to categorize the video into predefined categories. However, this is a quite challenging task due to the *semantic gap*, defined as the problem of accurately classifying multimedia content from automatically extracted low-level features. Alternatively, videos can be indexed in video databases using selected *keywords*. The last stage of video analysis is *content-based video retrieval*, where a user can search and retrieve selected video segments from a video database [70]. There are three widely accepted approaches to access a video database. The first one is *query by example*, where users provide an example video clip and retrieve a set of similar videos. In the second approach, users prefer to query the video database via high-level semantic visual concepts. Finally, in the *query by keyword* approach, the users query the video database via keywords that are used for indexing the videos in the databases.

In this thesis, we focus on the problems of efficient segmentation and representation of

a video sequence. More specifically, the first step towards automatic annotation of digital video sequences is to divide the video stream into a set of meaningful and manageable segments (*shots*) that are used as basic elements for indexing [44]. The *shot* is defined as an unbroken sequence of frames recorded from one camera. Proceeding further towards the goal of video indexing and retrieval requires the grouping of shots into high level video units such as *scenes* and *chapters* (*stories*) [62]. A *scene* refers to a group of shots that take place in a fixed setting or describe an action or event. A *chapter* or *story* is a more compact representation of a video corresponding to a group of semantically correlated scenes. Concerning the video representation problem, each video segment (shot or scene) is represented with still frames (*key-frames*) or a short sequence of frames (*video summary*).

1.2 Video Segmentation and Representation

Under the visual perspective, a video is a three-dimensional signal, in which two dimensions reveal the visual content in the horizontal and vertical frame direction, and the third one reveals the variations of the visual content over the time axes. *Shot boundary detection* aims to temporally segment the video into consecutive shots (low-level segmentation). The basic idea is to identify the discontinuities of visual content [85].

There are three major challenges concerning the shot boundary detection problem [85]. The first one is the *representation* of the visual content of each video frame. The most common approach is to extract different features from each frame and obtain a compact content representation (i.e. histogram). Video segmentation techniques can be applied on a variety of features extracted from the video frames such as *visual*, *audio*, *motion* and *text* features. Visual features such as *color*, *texture* and *shape* provide important information to recognize the content of the video. Audio features are not commonly used, but sometimes they provide additional and useful information for video. Motion features are proposed to exploit the spatio-temporal relation of video frames and are

usually employed to describe the relation of moving objects across time. Text features are extracted from embedded objects in videos and contain rich semantic information related to the multimedia content. The extracted features must be invariant to several content variations such as illumination changes and object/camera movement. Furthermore, the more details feature captures, the more sensitive it is, since the feature can even reflect the tiny changes of visual content. Thus, the feature within shots should remain relatively stable, whereas between different shots should exhibit considerable change.

The second challenge concerns the construction of a *discontinuity signal* to identify the shot transitions. The most common practice is to calculate the discontinuity (distance) values of adjacent frames. In this way, the visual content flow is transformed into a 1-D temporal signal. This signal should keep low magnitudes within shots, while should increase to high values surrounding the positions of shot transitions. However, the temporal signal obtained by inter-frame comparison of features is not always stable enough to various disturbances such as abrupt illumination variation and large object/camera movement. Thus, contextual information expressing the variations in a neighborhood of a particular frame should also be considered.

Finally, *classification* of discontinuity values is a critical issue. The most common approaches compare discontinuity values with predefined thresholds or employ machine learning techniques to classify discontinuity values into shot transitions or normal transitions.

Video segmentation into high level units such as *scenes* and *chapters* (high-level segmentation) is a very difficult but also challenging task [32, 62]. The difficulty stems from the fact that high level units do not have *physical boundaries* like shots, but their boundaries correspond to changes in the *semantic content* of the movie. A common approach to detect scene boundaries is to compare adjacent shots and construct a discontinuity signal similar to the shot boundary detection problem. The constant change of the video content and the fact that adjacent shots do not always describe the same *setting*, necessitate the consideration of contextual information in a neighborhood of shots. Furthermore, shot representation should be enriched with *semantic* features, extracted directly from

the video content or by classifying shots using a predefined lexicon of *semantic concepts*.

Shot representation is also an important issue, since it is a pre-processing step for scene and chapter segmentation [92]. Shots are usually represented as a set of key-frames. The extracted representative frames (key-frames) should fulfil some requirements. Firstly, the key-frames should represent the whole video content without missing important information and secondly, these key-frames should not be similar, in terms of video content information, in order not to contain redundant information.

Apart from typical video database applications, video segmentation and representation can also be extended and provide solution to other video-based applications. Two such applications that are examined in this thesis are *video rushes summarization* [59] and *video surveillance* [34]. The goal of video rushes summarization is to create a condensed version of the initial video, so that judgements about the video content can be made in less time and effort than using the initial video. Video rushes contain many repetitive information and junk frames that should be removed from the final video summary. In video surveillance systems, the goal is to detect and characterize the major events of the video surveillance sequence into predefined categories.

1.3 Machine Learning Problems

Machine learning is the area of artificial intelligence that attempts to provide machines with the ability to learn from examples [3, 43]. More specifically, in machine learning problems we make use a set of observations (examples), which we call *training set*, in order to make predictions for unseen events. In the area of machine learning there are two major categories of problems; *supervised learning* and *unsupervised learning*. In supervised learning, every training example of the training set has the form of a pair (input, target), where input contains the features of the examples we want to characterize and target, the desired output result. The aim is to build a model that can be used to make predictions for the outputs of previously unseen inputs. On the other hand, unsupervised learning

methods assume a training set that only consists of observed inputs. The objective is to learn a model of these inputs, which can later be used for example to predict missing values of some of the observations, or to group similar observations into clusters. *Semi-supervised* machine learning methods combine characteristics of both supervised and unsupervised methods. These methods, require that some of the input observations are associated with the corresponding desired output, but they can also take advantage of available input observations whose corresponding desired output is unknown.

Supervised methods are further divided in two categories depending on the type of the outputs. In *classification* problems the outputs are labels that distinguish in which category the input belongs to. In contrast, if the outputs are continuous variables, the problem is known as *regression*. A popular classification method used in this thesis is the Support Vector Machines (SVM) [17].

Unsupervised methods consider among others the problems of *density estimation* and *clustering*. In density estimation problems, we wish to find the distribution that could have generated a set of observations with high probability. In clustering problems a set of examples is given that we wish to group them into clusters such that the examples in a cluster are similar and different from the examples in other clusters. To solve these problems we can use probability density estimation methods, such as *mixture models* and assign one cluster to each mixture component. A similar approach is followed in the *k-means* algorithm, where each example is assigned to the cluster whose center (also called *centroid*) is nearest. A quite different method is the *hierarchical clustering algorithm*, which is based on the gradual formation of clusters. The agglomerative algorithm starts with one cluster for each example and builds the hierarchy by progressively merging clusters with minimal distance. Another clustering approach based on graph theory is *spectral clustering* [56]. A graph can be constructed where the distances between prototypes correspond to the weights of edges of the graph. Clusters are obtained by analyzing the *spectrum* of the *similarity matrix*. Several clustering methods have been considered in this thesis as it will be described in the following chapters.

1.4 Thesis Contribution

The contribution of the thesis is twofold. On one hand, we focus on the efficient segmentation of a video into shots, scenes and chapters and we also provide an efficient shot representation scheme. On the other hand, using the proposed algorithms for video segmentation and representation, we provide efficient methods for video rushes summarization and event detection in video surveillance sequences. Next, we summarize the contributions of this thesis.

In Chapter 2, we present a supervised learning methodology for video shot detection [11, 15]. The main novelty of this approach is that shot transitions are detected without using any thresholds, which is the main drawback of the majority of shot detection algorithms. In the proposed approach, novel features that describe the variation between adjacent frames and the contextual information in a neighborhood of frames become inputs to a SVM classifier which categorizes transitions to three classes: normal, abrupt and gradual. Another novelty of out approach is that all types of shot transitions are detected using a single classifier. Numerical experiments are presented that compare our algorithm with threshold dependent methods and another supervised learning methodology.

In Chapter 3, we consider the key-frame extraction problem [10, 14]. In order to find unique and non repetitive frames that summarize the shot content, frames are clustered into groups using an enhanced spectral clustering algorithm. In the clustering stage after the eigenvector computation we employ the very efficient global k-means algorithm and the medoids of the clusters are characterized as key-frames. A novelty of the proposed approach is that the number of key-frames is estimated using results from spectral graph theory, by examining the eigenvalues of the similarity matrix corresponding to pairs of shot frames. Appropriate quality measures indicate that the proposed approach outperforms traditional techniques and provides efficient summarization and reconstruction of the video sequence from the extracted key-frames.

The efficient shot detection and representation methods are the first steps towards the definition of shot similarity metrics and the segmentation of videos into high-level

units. In Chapter 4, we present a scene detection algorithm [9, 14] that is based on the improved spectral clustering of Chapter 3 and on sequence alignment methods. In the method we propose, to overcome the difficulty of having prior knowledge of the scene duration, the shots are clustered into groups based only on their visual similarity and a label is assigned to each shot according to the group that it belongs to. Next, a sequence alignment algorithm is applied to detect when the pattern of shot labels changes, providing the final scene segmentation result. Experiments on TV-series and movies indicate that the proposed scene detection method accurately detects most of the scene boundaries while preserving a good tradeoff between recall and precision.

In Chapter 5, we present a high-level movie segmentation algorithm [13]. The main novelty of this approach is that movie shots are represented with local invariant descriptors instead of color histograms, resulting into a visual words histogram representation. Using a technique from text document segmentation, the visual words histograms of shots are temporally smoothed (using a gaussian kernel) with respect to neighboring histograms to preserve valuable contextual information. As indicated from numerical experiments, the semantic smoothing process at different time scales provides the efficient movie segmentation into different high-levels, such as scenes and chapters.

In Chapter 6, we propose a system for video rushes summarization [12]. A video sequence is segmented into shots and key-frames are extracted for each shot. Then, the edge direction histogram of each key-frame is computed in order to determine if it is a monochrome frame or a colorbar. In order to remove redundant information, we compare shots using a sequence alignment metric between the sets of their key-frames. The SIFT descriptors of the key-frames of the remaining representative shots are compared with a database of descriptors of frames containing clapboards. In that way, frames with clapboards are identified and removed from the video summary. Finally, the video summary is generated by concatenating frames around the key-frames of the remaining shots. Experimental results indicate that our system exhibited good performance in the Rushes Summarization task of TRECVID 2008.

In Chapter 7, we describe a system for event detection and classification in video rushes

surveillance sequences. First, the video is segmented into events using the local invariant descriptors of video frames. Next, we compute the visual words histograms for each video event and we employ machine learning techniques to classify events into predefined categories. Numerical experiments indicate that the proposed approach provides high event detection and classification rates.

Finally, in Chapter 8 we provide a review of the results of this thesis and we suggest some interesting directions for further research.

Chapter 2

A Support Vector Machine Approach for Detection of Video Shot Transitions

2.1 Introduction

The first step towards indexing, browsing and retrieval of video data is the efficient segmentation of video into smaller *physical* units. The smallest physical segment of a video

is the *shot* and is defined as an unbroken sequence of frames recorded from one camera. After this segmentation has been accomplished, each shot can be summarized with one or more frames called *key-frames* which are selected using spatial and temporal features. Further analysis requires grouping of shots into scenes with similar content. In this Chapter, we will focus on the first stage of the video segmentation problem which is *shot boundary detection*. Shot transitions can be classified into two categories. The first one which is the most common is the *abrupt cut*. An abrupt or hard cut takes place between consecutive frames due to camera switch. In other words, a different or the same camera is used to record a different aspect of the scene. The second category concerns *gradual transitions* such *dissolves*, *fade-outs* followed by *fade-ins*, *wipes* and a variety of video effects which stretch over several frames. A dissolve takes place when the initial frames of the second shot are superimposed on the last frames of the first shot. A fade-out is a gradual decrease in the intensity of a frame resulting to a black frame, while fade-in is the opposite i.e., starting from a black image the intensity of the frame gradually increases. In Fig. 2.1 and Fig. 2.2, we present examples of a hard cut and a dissolve, respectively.

Figure 2.1: Visual example of a hard cut.

Figure 2.2: Visual example of a dissolve.

A formal study of the shot boundary detection problem is presented in [85]. In [31], the major issues to be considered for the effective solution of the shot-boundary detection problem are identified. A comparison of existing methods is presented in [6, 19, 29, 47]. There are several approaches to the shot-boundary detection task most of which involve the determination of a predefined or adaptive threshold. A simple way to declare a hard cut is pair-wise pixel comparison [90]. This method is very sensitive to object and camera

motions, thus many researchers propose the use of a motion independent characteristic, which is the intensity or color, global or local histogram [54, 90]. The use of second order statistical characteristics of frames, in a likelihood ratio test, is also suggested [40, 90]. More specifically, the likelihood ratio test is used to compare corresponding blocks of successive frames. Shot transitions are identified when the number of changed blocks is above a predefined threshold. To overcome the difficulties that arise from the use of global thresholds several adaptive thresholding methods are reported [77, 83, 86]. In [87], an algorithm is presented based on the analysis of entering and exiting edges between consecutive frames. This approach works well on abrupt changes, but fails in the detection of gradual changes. In [7], mutual information and joint-entropy between frames are used for the detection of cuts, fade-ins and fade-outs. An original approach to partitioning of a video into shots based on a foveated representation of the video is proposed in [5].

A quite interesting approach is presented in [85], where the detection of shot boundaries is based on a graph partitioning problem. More specifically a weighted graph is constructed where each frame is treated as a node and the edges represent the similarity between corresponding frames. Then, the min-max criterion is used to partition this graph and the scores for all feasible cuts are calculated. As it concerns the gradual transitions, multi-resolution graphs are constructed which are further partitioned using the same criterion. Finally, support vector machines with active learning are implemented to declare boundaries and non-boundaries. A support vector machine classifier with color and motion features is also employed in [20]. In that work, the first minutes of a video have been used for training and the rest for testing. In [27], the authors propose as inputs to SVMs, wavelet coefficient vectors within sliding windows.

A variety of methods have been proposed for gradual transitions detection, but still are inadequate to solve this problem due to the complicated nature of such transitions. In [90], a twin-comparison technique is proposed for hard cuts and gradual transitions detection by applying different thresholds based on differences in color histograms between successive frames. In [57], a spatio-temporal approach was presented for the detection of a variety of transitions. There is also research specifically aimed towards the dissolve

detection problem. In [48], the problem of dissolve detection is treated as a pattern recognition problem. Another direction, which is followed in [28, 31, 46], is to model the transitions types by presupposing probability distributions for the feature difference metrics and perform a posteriori shot change estimation. It is worth mentioning that the organization of the TREC video shot detection task [68] provides a standard performance evaluation and comparison benchmark.

In summary, the main drawback of most previous algorithms is that they are threshold dependent. As a result, if there is no prior knowledge about the visual content of the video that we wish to segment into shots, it is rather difficult to select an appropriate threshold.

In order to overcome this difficulty we propose in this Chapter a supervised learning methodology for the shot detection problem [11, 15]. The herein proposed approach does not use thresholds and can actually detect shot boundaries of videos with different visual characteristics. Another advantage of the proposed approach, apart from the fact that we do not use any thresholds, is that we can detect hard cuts and gradual transitions at the same time in contrast with existing approaches. For example, in [20], a support vector machine classifier only for abrupt cut detection is proposed. In [85], features for abrupt cuts and dissolves are constructed separately and two different SVM models are trained. In our approach, we define a set of features designed to discriminate hard cuts from gradual transitions. These features are obtained from color histograms and describe the variation between adjacent frames and the contextual information at the same time. Due to the fact that the gradual transitions spread over several frames, the frame-to-frame differences are not sufficient to characterize them. Thus, we also use the differences between non-adjacent frames in the definition of the proposed features.

The rest of the Chapter is organized as follows: In Sections 2.2 and 2.3, the proposed features used for video shot classification are described. In Section 2.4 the SVM method employed for this application is briefly presented. In Section 2.5 we present numerical experiments and compare our method with four existing methods and finally, in Section 2.6 we provide some conclusions.

2.2 Feature Selection

2.2.1 Color Histogram and x^2 Value

Color histograms are the most commonly used features to detect shot boundaries. They are quite robust to object and camera motion, and provide a good trade-off between accuracy of detection and implementation speed. We have chosen to use normalized RGB histograms. Thus, for each frame a normalized histogram is computed, with 256 bins for each one of the RGB component defined as H_R, H_G and H_B, respectively. These three histograms are concatenated into a 768 dimension vector representing the final histogram of each frame:

$$H = [H_R H_G H_B]. \tag{2.1}$$

To determine whether two shots are separated with an abrupt cut or a gradual transition we have to define a difference measure between frames. The simplest method for shot detection is to compute the histograms of two adjacent frames, calculate the sum of their bin-wise differences and compare it to a threshold. In our approach, we use a variation of the x^2 value [54, 66] to compare the histograms of two frames. Finally, the difference between two images I_i, I_j based on their color histograms H_i, H_j is given from the following equation:

$$d(I_i, I_j) = \frac{1}{3} \sum_{k=1}^{768} \frac{(H_i(k) - H_j(k))^2}{H_i(k) + H_j(k)}, \tag{2.2}$$

where k denotes the bin index.

2.2.2 Inter-frame Distance

The dissimilarity value given in equation 2.2 can be computed for any pair of frames within the video sequence. We compute the value not only between adjacent frames, but also between frames with time distance l, where l is called the *inter-frame distance* as suggested in [1, 31]. We compute the dissimilarity value $d(I_i, I_{i+l})$ for three values of the inter-frame distance l:

1. $l = 1$. This is used to identify hard cuts between two consecutive frames. Thus, the dissimilarity values are computed for $l = 1$.

2. $l = 2$. Due to the fact that during a gradual transition two consecutive frames may be the same or very similar to each other, the dissimilarity value will tend to zero and, as a result, the sequence of the dissimilarity values could have the form shown in Fig. 2.3. The computation for $l = 2$ usually results in a smoother curve, which is more useful for our further analysis. A typical example of a sequence of dissimilarity values for $l = 2$ is shown in Fig. 2.4.

3. $l = 6$. A gradual transition stretches along several frames, while the difference value between consecutive frames is smaller, so we are interested not only in the difference between consecutive frames, but also between frames that are a specific distance apart from each other. As the inter-frame distance increases, the curve becomes smoother as it can be observed in the example of Fig. 2.5.

The maximum distance between frames for which the inter-frame distance is useful is rather small. This distance should be less than the minimum length of all transitions in the video set in order to capture the form of the transition. Thus, the choice of $l = 6$ was made due to the fact that most of the gradual transitions in our set of videos have length between 7 and 40 frames.

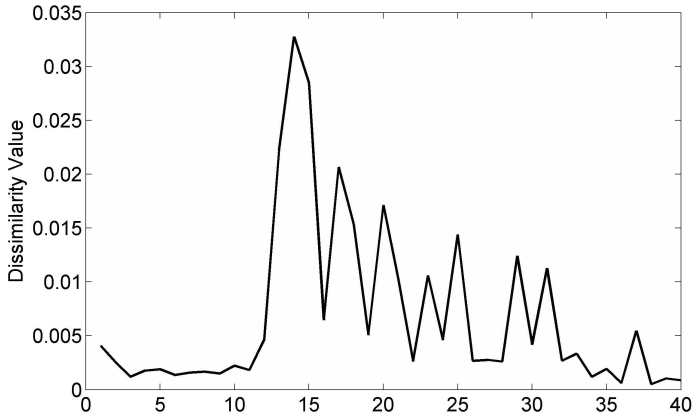


Figure 2.3: Dissimilarity pattern for $l = 1$.

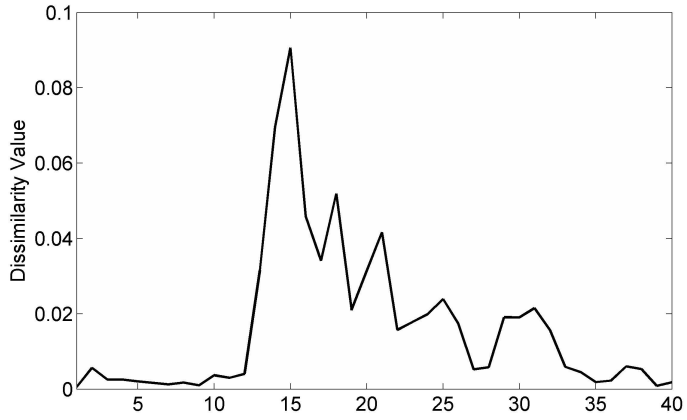


Figure 2.4: Dissimilarity pattern for $l = 2$.

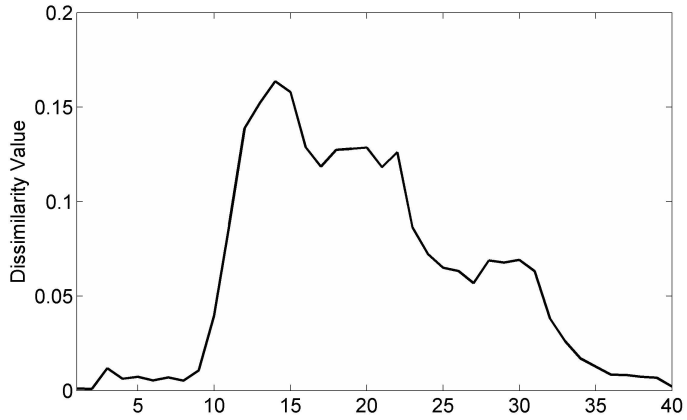


Figure 2.5: Dissimilarity pattern for $l = 6$.

2.3 Feature Vector Formulation for Shot Boundary Classification

The dissimilarity values defined in Section 2.2 will not be compared with any threshold, but they will be used to form the feature vectors based on which an SVM classifier will be constructed.

2.3.1 Definition of Feature Vectors

The selected feature vectors are the normalized dissimilarity values calculated in a temporal window centered at the frame of interest. More specifically, the dissimilarity values that have been defined in Section 2.2 form three vectors, one for each of the three inter-

frame distances l.

$$D^{l=1} = [d(I_1, I_2), \ldots, d(I_i, I_{i+1}), \ldots, d(I_{N-1}, I_N)], \tag{2.3}$$

$$D^{l=2} = [d(I_1, I_3), \ldots, d(I_i, I_{i+2}), \ldots, d(I_{N-2}, I_N)], \tag{2.4}$$

$$D^{l=6} = [d(I_1, I_6), \ldots, d(I_i, I_{i+6}), \ldots, d(I_{N-6}, I_N)]. \tag{2.5}$$

where N denotes the number of video frames. Moreover, for each frame i we define a window of length w that is centered at this frame and contains the dissimilarity values. As a result, for the i-th frame the following three vectors are composed:

$$W^{l=1}(i, 1:w) = [D^{l=1}(i - w/2), \ldots, D^{l=1}(i), \ldots, D^{l=1}(i + w/2 - 1)], \tag{2.6}$$

$$W^{l=2}(i, 1:w) = [D^{l=2}(i - w/2), \ldots, D^{l=2}(i), \ldots, D^{l=2}(i + w/2 - 1)], \tag{2.7}$$

$$W^{l=6}(i, 1:w) = [D^{l=6}(i - w/2), \ldots, D^{l=6}(i), \ldots, D^{l=6}(i + w/2 - 1)]. \tag{2.8}$$

To obtain the final features we normalize the dissimilarity values in equations (2.6), (2.7) and (2.8) by dividing each dissimilarity value by the sum of the values in the window. This provides the normalized "magnitude" independent features.

$$\tilde{W}^{l=k}(i, j) = \frac{W^{l=k}(i, j)}{\sum_{j=1}^{w} W^{l=k}(i, j)}, \; k = 1, 2, 6. \tag{2.9}$$

The size of the window used is $w = 40$. In our experiments, we also considered windows of length 50 and 60 in order to capture longer transitions. The 120-dimensional vector resulting from the concatenation of the normalized dissimilarities for the three windows given by equation (2.10), is the feature vector corresponding to frame i.

$$F(i) = [\tilde{W}^{l=1}(i) \; \tilde{W}^{l=2}(i) \; \tilde{W}^{l=6}(i)] \; . \tag{2.10}$$

In what follows, we show examples of the feature vectors for a hard cut, two dissolves and a "normal" sequence of frames in Fig. 2.6, Fig. 2.7, Fig. 2.8 and Fig. 2.9, respectively. By

observing these features, it is clear that the shape of the $l = 1$ normalized dissimilarity vectors for normal sequences and dissolves may be of similar shape. However, the inclusion of the $l = 2$ and $l = 6$ dissimilarity vectors discriminates the two categories.

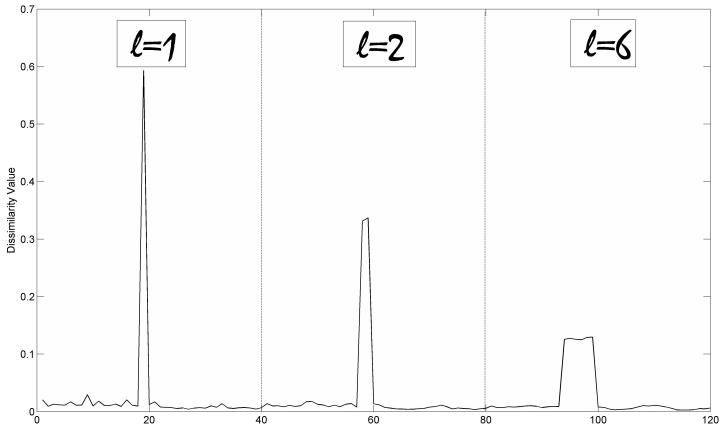


Figure 2.6: Feature vector for a hard cut.

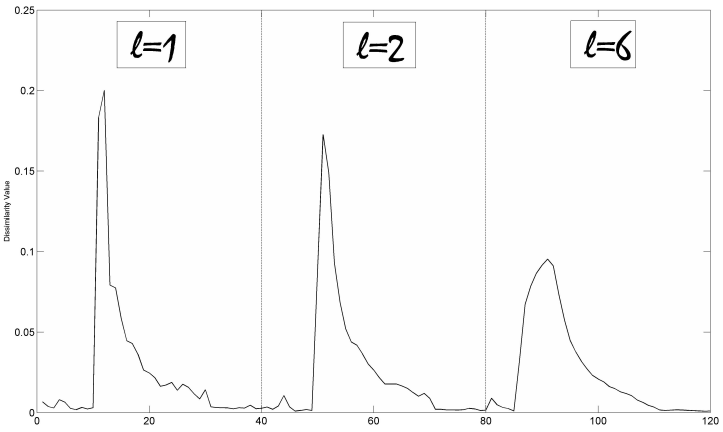


Figure 2.7: Feature vector for the first dissolve example.

2.4 Support Vector Machine Classifier

After the feature definition, an appropriate classifier has to be used in order to categorize each frame in three categories: normal sequences, abrupt cuts and gradual transitions. For

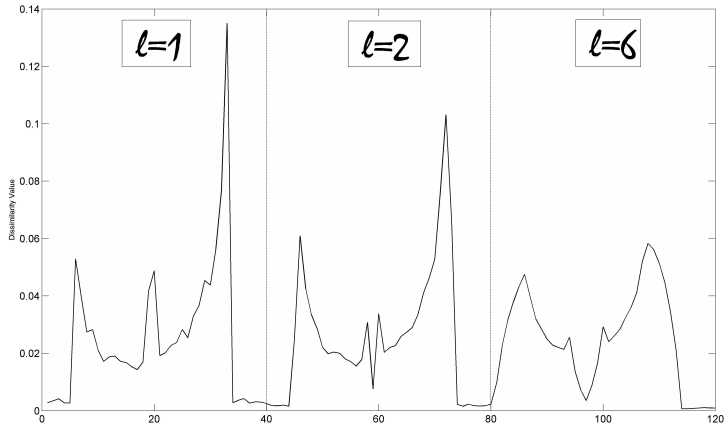


Figure 2.8: Feature vector for the second dissolve example.

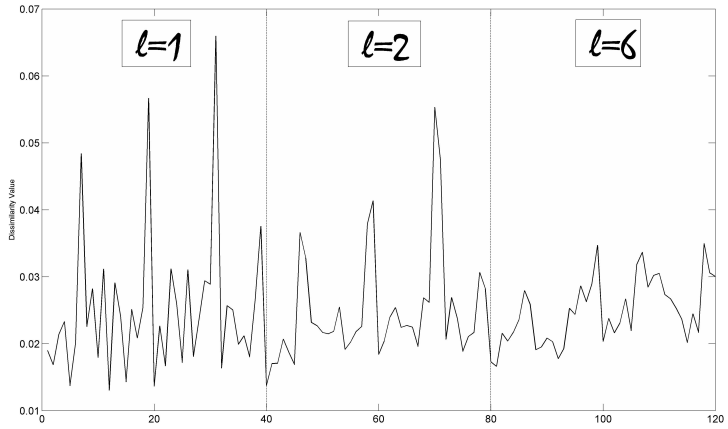


Figure 2.9: Feature vector for a normal sequence of frames.

this purpose we selected the Support Vector Machine (SVM) classifier [17] that provides state-of-the-art performance and scales well with the dimension of the feature vector which is relatively large (equal to 120) in our problem.

The classical SVM classifier finds an optimal hyperplane which separates data points of two classes. More specifically, suppose we are given a training set of m vectors $x_i \in \mathbb{R}^n$, i=1, . . . , m and a vector $y \in \mathbb{R}^m$ with $y_i \in \{1,\text{-}1\}$ denoting the class of vector x_i. We also assume a mapping function $\phi(x)$, that maps each training vector to a higher dimensional space, and the corresponding kernel function (equation (2.15)). Then, the SVM classifier

[17] is obtained by solving the following primal problem:

$$\min_{w,b,\xi} \quad \tfrac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i \tag{2.11}$$

$$\text{subject to} \quad y_i(w^T\phi(x_i)+b) \geq 1-\xi_i \tag{2.12}$$

$$\xi_i \geq 0,\ i=1,\ldots,m. \tag{2.13}$$

The decision function is:

$$sign(\sum_{i=1}^{m} w_i K(x_i,x)+b),\ \text{where } K(x_i,x_j)=\phi^T(x_i)\phi(x_j)\ . \tag{2.14}$$

A notable characteristic of SVMs is that after training, usually most of the training patterns x_i have $w_i = 0$ in equation (2.14), in other words they do not contribute to the decision function. Those x_i for which $w_i \neq 0$, are retained in the SVM model and called Support Vectors (SVs). In our approach the commonly used radial basis function (RBF) kernel is employed:

$$K(x_i,x_j) = exp(-\gamma\|x_i-x_j\|^2)\ , \tag{2.15}$$

where γ denotes the width of the kernel. It must be noted that in order to obtain an efficient SVM classifier the parameters C, γ in equations (2.11), (2.15) respectively, must be carefully selected, usually through cross-validation.

The above algorithm is suitable for binary classification. In our application, we have a three-class problem, thus we used the "one-against-one" approach [41] in which for a k-class problem, $k(k-1)/2$ binary classifiers are constructed and each one is trained to discriminate data from two classes. More specifically, if we assume that class label 0 corresponds to normal sequences, class label 1 to dissolves and class label 2 to hard cuts, three binary classifiers discriminating between pairs of classes (0,1), (1,2) and (0,2) are constructed. The final classification is based on a voting strategy where the decision of each binary classifier is considered as a vote for its proposed class and the class with the maximum number of votes is selected. In the case of a tie the class with the smallest index is selected. This tie braking strategy is well-justified in our case, since class 0 corresponds

Table 2.1: Characteristics of videos used for the shot detection problem.

Video ID	Frames	Cuts	Dissolves	Genre
T1	6318	36	23	Comedy
T2	9466	28	16	Action
T3	11807	4	6	Drama
T4	1535	14	8	Educational
T5	17982	146	7	Action
T6	1665	1	19	Comedy
T7	14993	105	11	Drama
T8	9840	12	41	Documentary
T9	6355	9	11	Documentary
Total	69334	355	142	-

Table 2.2: Training examples and support vectors.

Transition type	Positive examples	Negative examples	Support vectors
Cuts	315	-	152
Dissolves	126	-	101
Normal	-	2200	1276

to normals which is the most probable outcome.

2.5 Numerical Experiments

In this Section, we present numerical experiments of the proposed approach and compare our method with four other methods.

2.5.1 Video Data for Shot Detection Problem

The video sequences used for our data set were taken from TV-series, documentaries and educational films. Nine videos (70000 frames), manually annotated by a human observer, were used; containing 355 hard cuts and 142 dissolves (Table 2.1).

2.5.2 Performance Criteria

To evaluate the performance of our method we used the following commonly used criteria [2]:

$$\text{Recall} = \frac{N_c}{N_c + N_m}, \tag{2.16}$$

$$\text{Precision} = \frac{N_c}{N_c + N_f}, \tag{2.17}$$

$$F_1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{2.18}$$

where N_c stands for the number of correct detected shot boundaries, N_m for the number of missed ones and N_f the number of false detections. During our experiments we calculate the F_1 value for the cuts (F_{1C}) and the dissolves (F_{1D}) separately. Then, the final performance measure is given from the following equation:

$$F_1 = \frac{\alpha}{\alpha + b} F_{1C} + \frac{b}{\alpha + b} F_{1D} \ , \tag{2.19}$$

where α is the number of true hard cuts and b the number of true dissolves.

2.5.3 Results

In our experiments, 8 videos are used for training and the 9-th for testing, therefore, 9 "rounds" of testing were conducted. In order to obtain good values of the parameters C and γ (in terms of providing high F_1 values), in each "round" we applied 3-fold cross-validation using the 8 videos of the corresponding training set. A difficulty of the problem under consideration is the generation of an imbalanced training set that contains few positives examples and a huge number of negative ones. In [65], an active learning procedure is proposed to reduce the training time. Based on the assumption that the support vectors determine the decision boundary in equation (2.14), they suggest removing the training examples that are far from the SVM's decision hyperplane. In [85], the authors identify the positive examples while reducing the number of negative ones by

applying a predefined

Table 2.3: Performance results for $w = 40$, $l = 1$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	351	4	9	98.87	97.50	98.18
Dissolves	127	15	33	89.44	79.38	84.11
Average	-	-	-	96.18	92.32	94.21

Table 2.4: Performance results for $w = 50$, $l = 1$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	352	3	8	99.15	97.78	98.46
Dissolves	130	12	25	91.55	83.87	87.54
Average	-	-	-	96.98	93.80	95.37

Table 2.5: Performance results for $w = 60$, $l = 1$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	353	2	4	99.44	98.88	99.16
Dissolves	127	15	25	89.44	83.55	86.39
Average	-	-	-	96.58	94.50	95.53

threshold on their constructed features. In our approach, we sample negative examples uniformly, thus we reduce their number to 3% of the total number of examples. More specifically, in our training set there are 440 positive examples (transitions) and 2200 negative examples (no transitions) on average. Finally, each model of the training procedure generated on average 1276 support vectors for normal transitions, 101 support vectors for gradual transitions and 152 support vectors for abrupt transitions. The number of examples and support vectors (on average) of the support vector machines classification are summarized in Table 2.2.

We also tested our method by using larger windows of width $w = 50$ and $w = 60$. In what follows in Tables 2.3, 2.4 and 2.5, we provide the classification results using different selections of window lengths. It can be observed that the performance improves as the size of the window increases. False boundaries are reduced since larger windows contain

more information. The use of larger windows also helps the detection of dissolves that last longer.

In order to reduce the size of our feature vector, we have also considered as feature vectors used to train the SVM classifier, those obtained from the concatenation of features extracted for $l = 2$ and $l = 6$, only. It can be observed in Tables 2.6, 2.7 and 2.8 that even with the shorter feature vector the proposed algorithm gives very good results that are only slightly inferior to the ones obtained by the longer feature vector.

In order to test the importance of selecting the best values for parameters (C, γ), in another experiment we used the SVM classifier with constant values pair $(C, \gamma) = (6, 8)$ for all "rounds" of testing. The obtained results (Tables 2.9-2.14) indicate that even without the optimal "selection" of (C, γ) the performance of the SVM classifier remains very good.

In another additional experiment we carried out, an HSV normalized histogram is computed for each frame, with eight bins for hue and four bins for each of saturation and value, resulting $8 \times 4 \times 4$ bins. In Tables 2.15 and 2.16, we provide the classification results using the x^2 value defined in equation (2.2) and the Kullback-Liebler distance between two histograms. It can be observed that the method is not sensitive to the choice of the color space and the distance measure between the color.

Table 2.6: Performance results for $w = 40$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	351	4	9	98.87	97.50	98.18
Dissolves	127	16	30	88.73	80.77	84.56
Average	-	-	-	95.98	92.72	94.32

Table 2.7: Performance results for $w = 50$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	350	5	5	98.59	97.49	98.04
Dissolves	129	13	21	90.85	86.00	88.36
Average	-	-	-	96.38	94.21	95.28

Table 2.8: Performance results for $w = 60$, $l = 2$ and $l = 6$.

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	351	4	5	98.87	98.60	98.73
Dissolves	128	14	26	90.14	83.12	86.49
Average	-	-	-	96.38	94.17	95.26

Table 2.9: Performance results for $w = 40$, $l = 1$, $l = 2$ and $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	353	2	9	99.44	96.98	98.19
Dissolves	128	14	23	90.14	84.77	87.37
Average	-	-	-	96.78	93.49	95.11

Table 2.10: Performance results for $w = 50$, $l = 1$, $l = 2$ and $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	353	2	7	99.44	98.06	98.74
Dissolves	128	14	31	90.14	80.50	85.05
Average	-	-	-	96.78	93.04	94.87

Table 2.11: Performance results for $w = 60$, $l = 1$, $l = 2$ and $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	353	2	5	99.44	98.60	99.02
Dissolves	128	14	23	90.14	84.77	87.37
Average	-	-	-	96.78	94.65	95.70

Table 2.12: Performance results for $w = 40$, $l = 2$, $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	352	3	13	99.15	96.44	97.78
Dissolves	127	16	22	88.73	85.14	86.90
Average	-	-	-	96.18	93.21	94.67

Table 2.13: Performance results for $w = 50$, $l = 2$ and $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	352	3	10	99.15	96.44	98.19
Dissolves	129	13	20	90.85	86.58	88.66
Average	-	-	-	96.78	94.19	95.47

In order to gain more intuition of how the SVM classifier solves this problem, in Fig. 2.10 - 2.12, we provide correctly detected feature vectors from dissolves, hard cuts and

Table 2.14: Performance results for $w = 60$, $l = 2$ and $l = 6$ and constant (C, γ).

Transition type	N_c	N_m	N_f	Recall (%)	Precision (%)	F_1 (%)
Cuts	351	4	7	98.87	98.04	98.46
Dissolves	127	15	22	89.44	85.23	87.29
Average	-	-	-	96.18	94.38	95.27

normal sequences of frames. From these figures, it is clear that the selected features of the three classes of frame sequences exhibit distinguishable characteristics. More specifically, for hard cuts the feature vector contains three "impulses" with decaying height and increasing width. For dissolves it contains three replicas of a pattern that resembles to a rectangle from which a sinusoidal lobe has been subtracted. Furthermore, these patterns become smoother as we move from left to right. Finally, for "normal" sequences of frames the pattern resembles to "white noise" superimposed on a constant DC value.

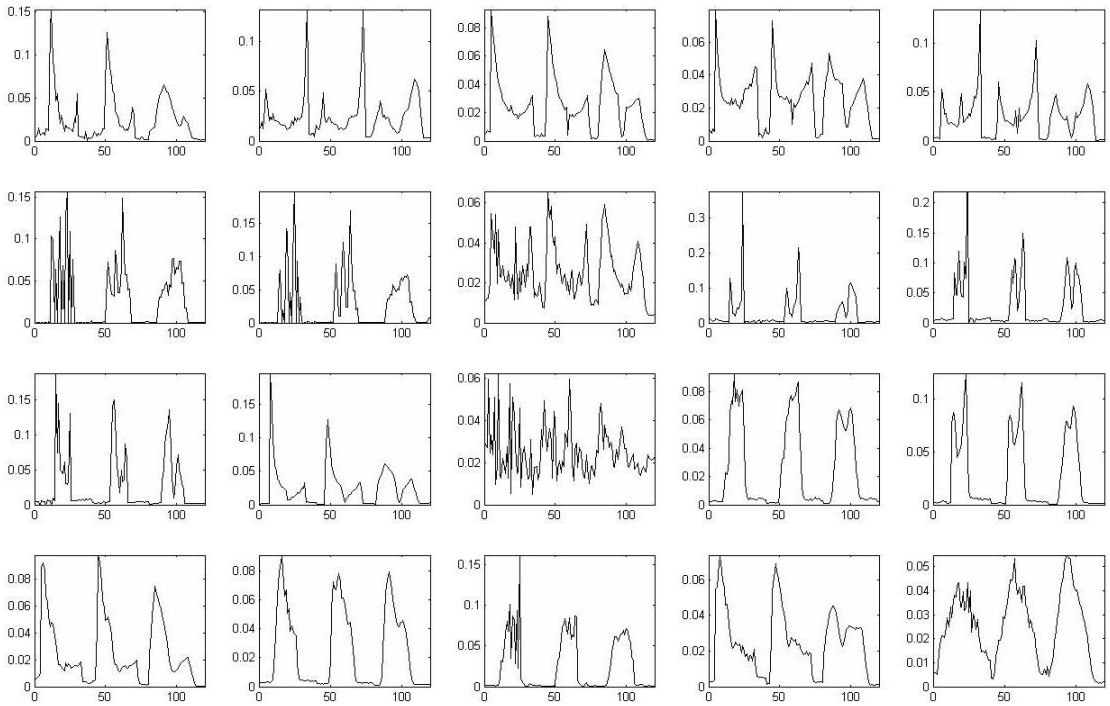


Figure 2.10: Correctly detected dissolve patterns.

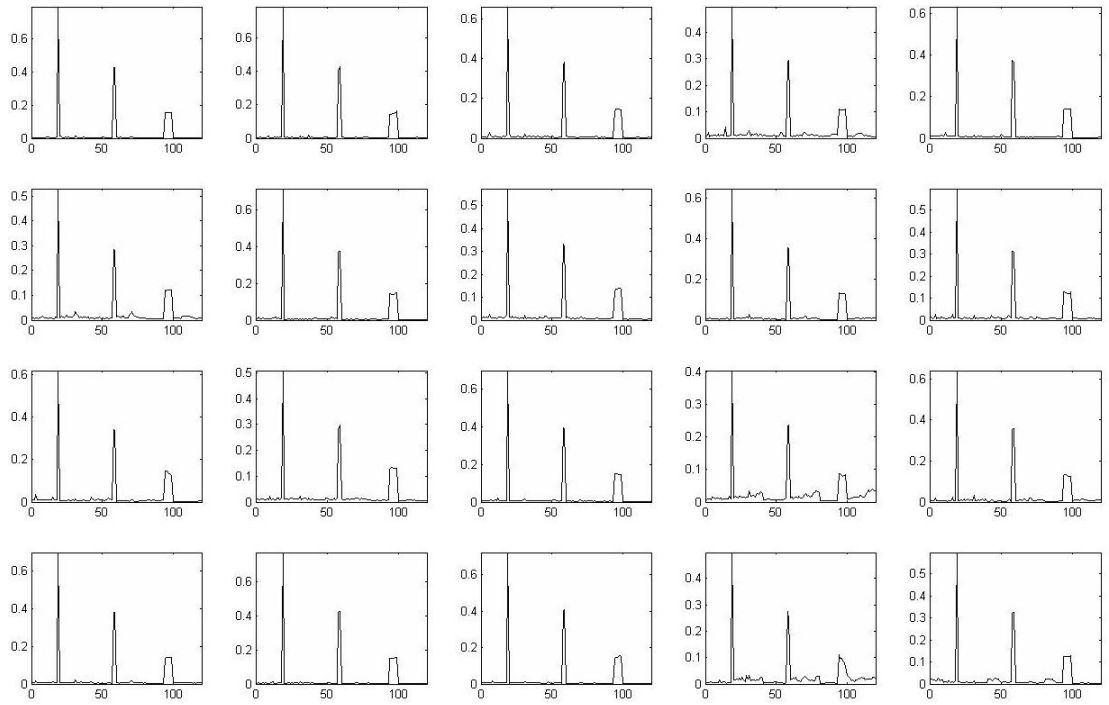


Figure 2.11: Correctly detected hard cut patterns.

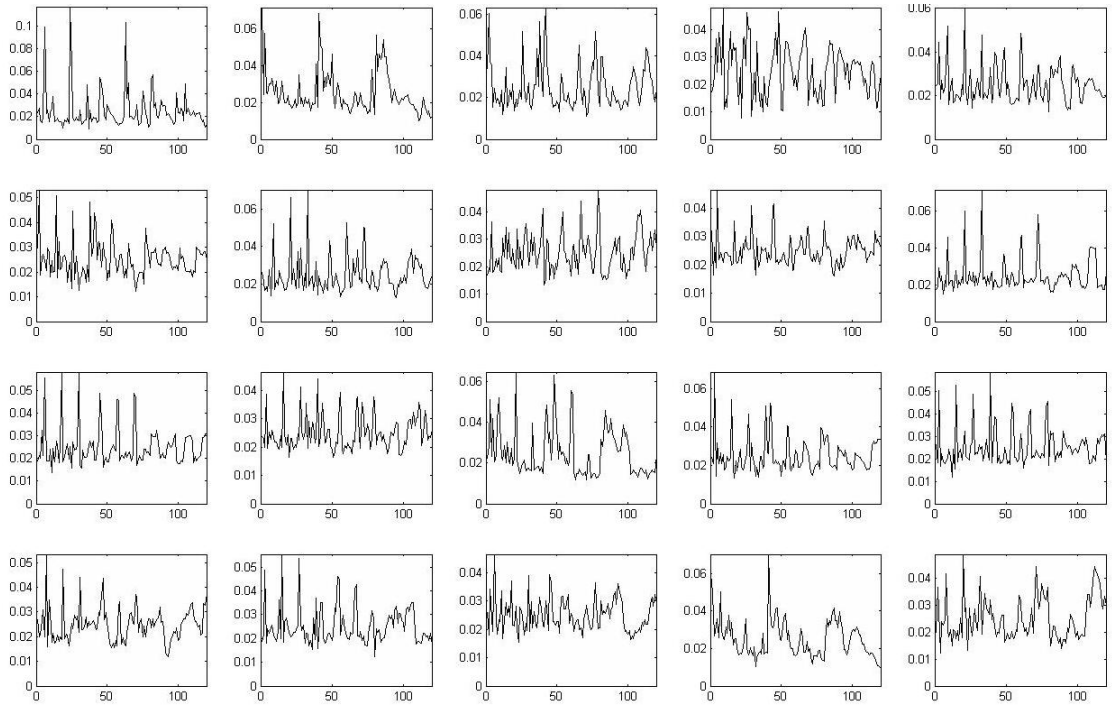


Figure 2.12: Correctly detected patterns of normal sequences.

In Fig. 2.13 - 2.15, we provide a portion of SVs for all the cases for the same "round". These examples are training patterns for which $w_i \neq 0$ in equation (2.14) and are retained in the SVM model. We immediately notice, as expected, that the SVs are the "borderline

examples" for all categories. The hardest cases to separate are normal from dissolves, since more SVs of the normal class have characteristics of correctly classified dissolves and fewer have characteristics of hard cuts. Similarly, more SVs for dissolves have characteristics of "normal" than hard cuts. Furthermore, the SVs for hard cuts are much fewer than their "normal" and dissolves counterparts. Also one cannot make an assessment whether most of them have characteristics of the normal or dissolve class.

2.5.4 Comparison

To demonstrate the effectiveness of our algorithm and its advantage over threshold depended methods, we implemented three methods that use thresholds in different ways. More specifically, we implemented pair-wise comparison of successive frames [90], likelihood ratio test [40, 90] and the twin-comparison method [90]. The first two methods

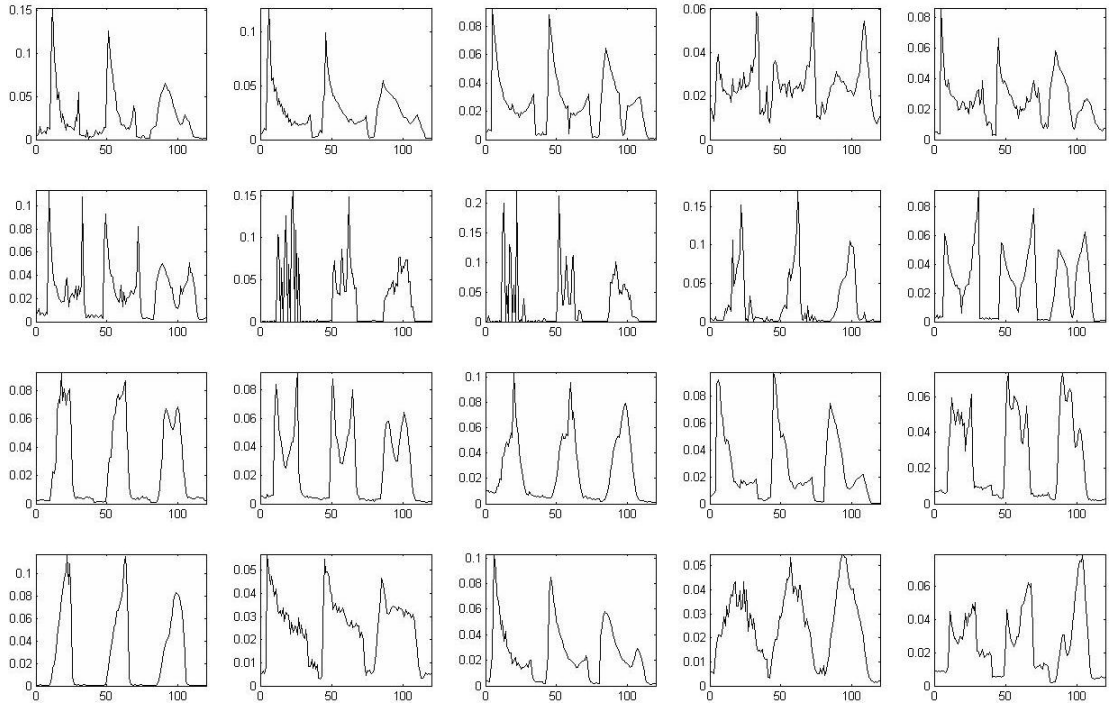


Figure 2.13: Support vectors for dissolves.

can only detect cuts, while the third can identify both abrupt and gradual transitions. We have also compared our method with the method proposed in [27].

The pair-wise comparison method [90] compares corresponding pixels of successive

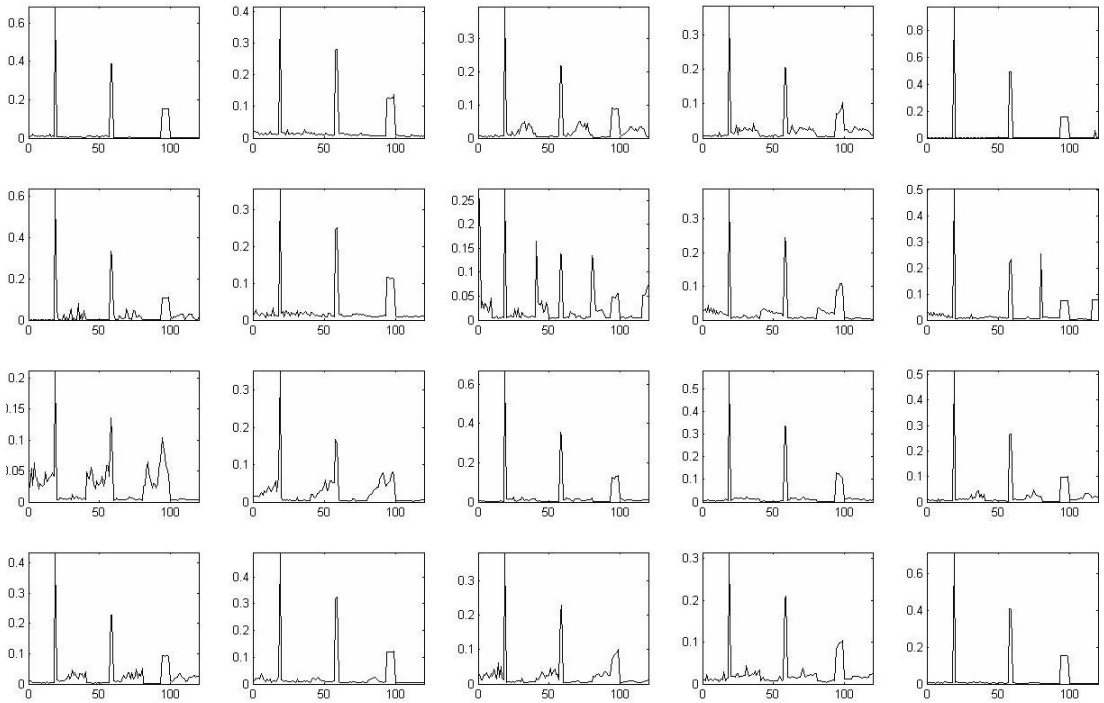


Figure 2.14: Support vectors for hard cuts.

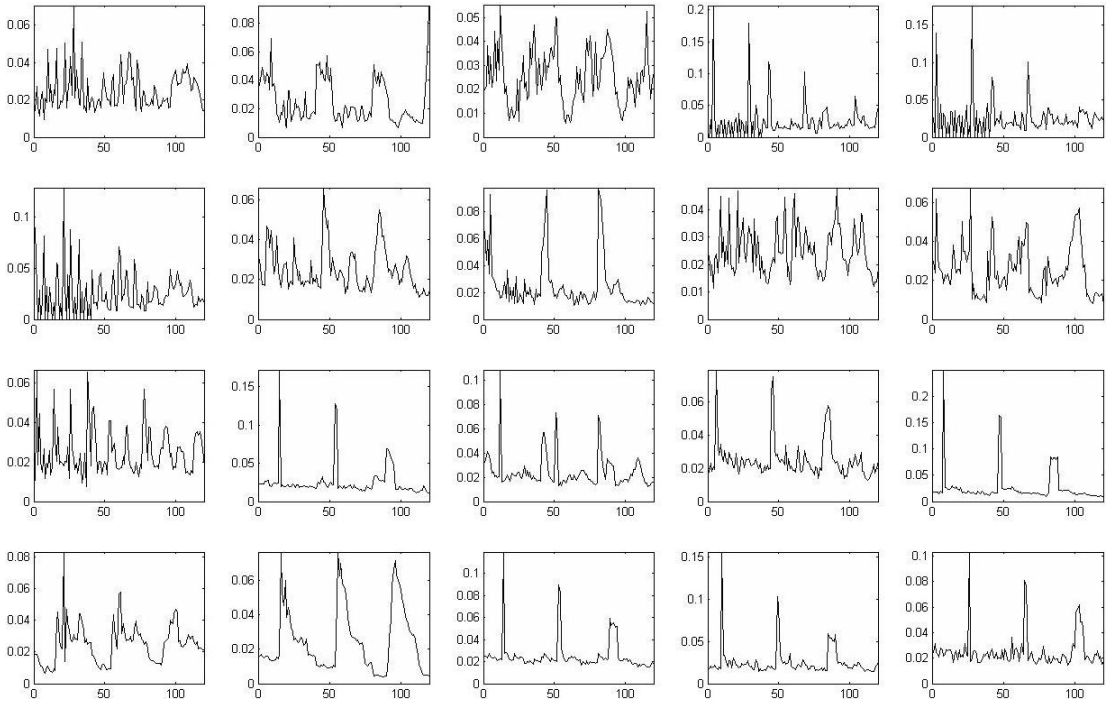


Figure 2.15: Support vectors for normal sequences.

frames to determine how many pixels have changed. More specifically we consider that a pixel changes if the difference of its corresponding pixel in the following frame is over a

predefined threshold:

$$DP_i(x,y) = \begin{cases} 1, & \text{if } \mid P_i(x,y) - P_{i+1}(x,y) \mid > \text{Th}; \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

where $P_i(x,y)$ is the intensity of pixel with coordinates (x,y) in frame i. A shot boundary is declared if the number of the pixels that have changed is over another predefined threshold. However the appropriate selection of such a threshold is a tedious task. Firstly, this threshold is different for videos that belong to different genres and secondly, even in the same video differences between frames may vary due to different states of illumination and content. Thus, it is difficult to define a global threshold. In [86], the authors propose the use of a sliding window over the differences. A shot boundary is detected if two conditions are fulfilled: the middle sample of the window (a) is the maximum in the window and (b) is greater than:

$$\max(\mu_{left} + a\sigma_{left}, \mu_{right} + a\sigma_{right}), \tag{2.21}$$

where μ_{left}, μ_{right} and $\sigma_{left}, \sigma_{right}$ are the means and standard deviations of the samples, left and right of the middle sample of the window, respectively. The length of the window and the parameter a are set to 21 and 5, respectively. In Tables 2.15 and 2.16, we provide the performance results of the pair-wise comparison method.

The second method we implemented uses the likelihood ratio [40, 90] as the metric to compute frame differences. More specifically, this metric compares the second order statistics of corresponding regions. Each frame is divided into blocks, which represent the regions, and the likelihood ratio between two consecutive frames $i, i+1$ for a specific block k is given from the following equation:

$$\lambda(i,i+1)_k = \frac{[(\frac{\sigma_i+\sigma_{i+1}}{2}) + (\frac{\mu_i+\mu_{i+1}}{2}))]^2}{\sigma_i \times \sigma_{i+1}}, \tag{2.22}$$

where μ_i, μ_{i+1} and σ_i, σ_{i+1} are the means and standard deviations of block k of frames

$i, i+1$, respectively. Then,the likelihood ratio between two consecutive frames $i, i+1$ is as follows:

$$L(i, i+1) = \frac{\sum_{k=1}^{K} \lambda(i, i+1)_k}{K}, \tag{2.23}$$

where K is the number of blocks of the frame. A shot boundary is detected when the likelihood ratio between two frames exceeds a predefined threshold. To improve the performance of the specific method, we do not use a global threshold, but we select the threshold via cross-validation, using the "leave-one-out" method. To identify the threshold and test it on a video of our dataset, we choose the threshold that achieves the best performance over the rest eight videos of our dataset. In Tables 2.15 and 2.16 we provide the performance of the likelihood ratio method.

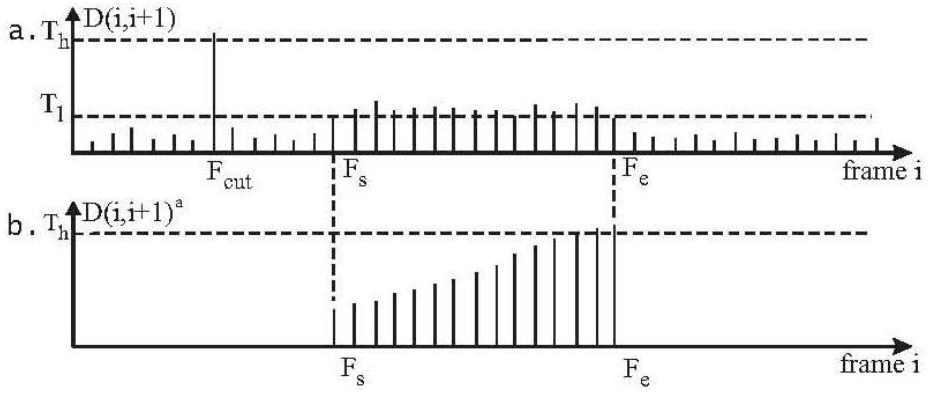


Figure 2.16: Twin-comparison algorithm [90].

The third method we implemented was the twin-comparison algorithm [90] which uses two thresholds for the detection of abrupt and gradual transitions. Each frame is represented with a histogram and the differences between histograms of consecutive frames are calculated. Histograms and their differences are computed using equations (2.1) and (2.2). If the difference between two successive frames exceeds a high threshold T_{high}, a cut is detected. A low threshold T_{low} is used for the detection of gradual transitions. If the difference is above T_{low} then this frame is characterized as a potential start F_s of the gradual transition. Then, F_s is compared with subsequent frames providing the accumulated differences metric. The end frame F_e of the transition is detected if two conditions are satisfied: (1) the consecutive difference falls below threshold T_{low} and (2)

Table 2.15: Comparative results using Recall, Precision and F_1 measures for cuts detection.

METHOD	CUTS		
	Recall (%)	Precision(%)	F_1(%)
$w = 40$, l=1, l=2 and l=6.	98.87	97.50	98.18
$w = 40$, l=2 and l=6.	98.87	97.50	98.18
$w = 40$, l=1, l=2,l=6 and constant (C, γ).	99.44	96.98	98.19
$w = 40$, l=2, l=6 and constant (C, γ).	99.15	96.44	97.78
$w = 40$, l=1, l=2 and l=6 (HSV, x^2).	99.44	98.89	99.16
$w = 40$, l=1, l=2 and l=6 (HSV, KL).	99.15	98.60	98.92
Pair-wise comparison [90]	85.07	84.83	84.95
Likelihood ratio [90]	94.37	86.12	90.05
Twin-comparison [90]	89.30	88.05	88.92
[27]	97.18	91.57	94.29

the accumulated difference exceeds threshold T_{high}. If consecutive difference falls below T_{low} before the accumulated difference exceeds T_{high}, then the potential start frame is discarded. In Fig. 2.16, we illustrate the twin-comparison algorithm.

Table 2.16: Comparative results using Recall, Precision and F_1 measures for dissolves detection.

METHOD	DISSOLVES		
	Recall (%)	Precision(%)	F_1(%)
$w = 40$, l=1, l=2 and l=6.	89.44	79.38	84.11
$w = 40$, l=2 and l=6.	88.73	80.77	84.56
$w = 40$, l=1, l=2,l=6 and constant (C, γ).	90.14	84.77	87.37
$w = 40$, l=2, l=6 and constant (C, γ).	88.73	85.14	86.90
$w = 40$, l=1, l=2 and l=6 (HSV, x^2).	88.03	81.17	84.46
$w = 40$, l=1, l=2 and l=6 (HSV, KL).	85.92	79.74	82.73
Pair-wise comparison [90]	-	-	-
Likelihood ratio [90]	-	-	-
Twin-comparison [90]	70.42	64.94	67.57
[27]	74.64	81.53	77.93

To compute the two thresholds we follow the method proposed in [42]. The threshold T_{low} is calculated from the following equation:

$$T_{low} = \mu + a\sigma, \tag{2.24}$$

where μ and σ are the mean and standard deviation of histogram differences respectively. The value of parameter a is set to 5. To calculate T_{high} we compute the histogram of the differences values. Threshold T_{high} is assigned to the index value that corresponds to half of the peak value on the right slope of the peak value of the histogram. T_{high} must be higher than mean value. In Tables 2.15 and 2.16, we provide the performance of the twin-comparison method.

Finally, in [27], Blocked Color Histogram is incorporated as feature vector and the temporal multi-resolution characteristics of shot presented by the wavelet transition coefficients are selected as video frame series patterns for a SVM classifier. In Tables 2.15 and 2.16, we present the classification results for this method.

The obtained results indicate that our algorithm outperforms the other three threshold dependent methods and the method proposed in [27]. In Tables 2.15 and 2.16, we provide the recall, precision and F_1 values for our algorithm and the four methods under consideration for cuts and dissolves detection, respectively. For our algorithm we present the results using $w = 40$, for best values (C, γ) and constant values pair (C, γ) = (6, 8), using all features (l =1, $l = 2$ and $l = 6$) and less features ($l = 2$ and $l = 6$). We also present the results using HSV histograms and two different distance metrics between histograms: (a) x^2 value and (b) Kullback-Liebler. The thresholds used in the three threshold dependent methods were calculated in different ways. We used adaptive thresholds in pair-wise comparison algorithm, cross-validation in likelihood ratio method and finally global adaptive threshold in the twin-comparison method. Especially for the dissolve detection our algorithm, provides far better results than the twin-comparison algorithm.

In summary, the proposed system is capable of identifying where a shot boundary occurs and whether the transition is abrupt or gradual. The main advantage of the method is that it can be trained using different types of video and then it can be used to locate shot boundaries in other videos without using any thresholds.

2.6 Conclusions

In this Chapter we have proposed a method for shot boundary detection and discrimination between a hard cut and a gradual transition. Traditionally, video shot segmentation approaches rely on thresholding methodologies which are sensitive to the content of the video being processed and do not generalize well when there is little prior knowledge about the video content. To ameliorate this shortcoming, we have proposed a learning based methodology using a set of features that are specifically designed to capture the differences between hard cuts, gradual transitions and normal sequences of frames at the same time. These features describe the variation between adjacent frames and the contextual information and are derived from color histograms using a temporal window. Next, they become inputs to a SVM classifier which categorizes transitions of the video sequence into normal transitions, hard cuts and gradual transitions. This categorization provides an effective segmentation of any video into shots, thus is a valuable aid to further analysis of the video for indexing and browsing. The main advantage is that throughout the whole procedure, no use of any thresholds is made.

Chapter 3

Key-Frame Extraction Using an Enhanced Spectral Clustering Approach

3.1 Introduction

A major issue with video retrieval is the efficient indexing of databases. The most popular indexing and summarization method is based on *key-frame extraction*. More specifically,

each video shot can be sufficiently summarized using its most representative frames, which are the *key-frames*. Any key-frame extraction algorithm should fulfil some requirements. Firstly, the key-frames should represent the whole video content without missing important information and secondly, these key-frames should not be similar, in terms of video content information, thus containing redundant information.

The simplest methods choose the first, last and median frames of a shot or a combination of the previous ones to describe the content of a shot [63]. A major category of key-frame extraction algorithms detect abrupt changes in the similarity between successive frames. In [78], the optical flow is computed and the local minima of a motion metric are selected as key-frames. In [22], it is proposed to form a trajectory from the feature vectors for all frames within a shot. The magnitude of the second derivative of this feature trajectory with respect to time is used as a curvature measure in this case. As key-frames the local minima and maxima of this magnitude are selected. In [30], the key frames are extracted by detecting curvature points within the curve of the cumulative frame differences.

Another category of key-frame extraction algorithms perform clustering of shot frames into groups and select a representative frame of each group as key-frame. In [92], multiple frames are detected using unsupervised clustering based on the visual variations in shots. A main drawback of this algorithm is the determination of the appropriate number of key-frames to represent each shot which depends on the threshold parameter that controls the density of the clusters. A variant of this algorithm is presented in [62], where the final number of key-frames depends on a threshold parameter which defines whether two frames are similar. In [7], the mutual information values of consecutive frames are clustered into groups using a split-merge approach. As key-frames are selected the representative frames of the clusters that maximize the interframe mutual information in each cluster. A different approach is presented in [38], where a video shot is segmented into homogeneous parts based on major types of camera motion and key-frames are extracted for each segment.

There are two major issues concerning key-frame extraction problem. The first one

is the extraction of key-frames that capture the whole content of the shot and do not contain redundant information. The second problem is the selection of the appropriate number of key-frames without any knowledge about the shot content. In this Chapter [10, 14], we propose a clustering of the frames of a video sequence into groups using an improved version of the typical spectral clustering algorithm [56] that employs the fast global k-means algorithm [49] in the clustering stage after the eigenvector computation.

The rest of this Chapter is organized as follows: In Section 3.2, we describe our key-frame extraction algorithm. In Section 3.3, we propose a method to estimate the number of key-frames using results from the spectral graph theory. In Section 3.4, we provide the criteria based on which the proposed key-frame extraction algorithm is evaluated. In Section 3.5, we provide numerical experiments and examples of video shot summarizations and finally, in Section 3.5 we conclude our work.

3.2 Key-Frame Extraction Algorithm

In this Section, we present the key-frame extraction algorithm that is based on the combination of the spectral clustering approach with the fast global k-means algorithm. Next, we estimate the number of key-frames using the eigenvalues of the similarity matrix corresponding to pairs of shot frames. For each frame, a 16-bin HSV normalized histogram is used, with 8 bins for hue and 4 bins for each saturation and value.

3.2.1 The Typical Spectral Clustering Algorithm

To perform key-frame extraction the video frames of a shot are clustered into groups using an improved spectral clustering algorithm. Then, the *medoid* of each group, defined as the frame of a group whose average similarity to all other frames of this group is maximal, is characterized as a key-frame. The main steps of the typical spectral clustering algorithm [56] are described next. Suppose there is a set of objects $S = s_1, s_2, \ldots, s_N$ to be partitioned into K groups.

1. Compute similarity matrix $A \in \mathbb{R}^{N \times N}$ for the pairs of objects of the data set S.

2. Define D to be the diagonal matrix whose (i, i) element is the sum of the A's i-th row and construct the Laplacian matrix $L = I - D^{-1/2} A D^{-1/2}$.

3. Compute the K principal eigenvectors $x_1, x_2, \ldots, x_K$ of matrix L to build an $N \times K$ matrix $X = [x_1\ x_2 \ldots\ x_K]$.

4. Renormalize each row of X to have unit length and form matrix Y so that:

$$y_{ij} = x_{ij} / (\sum_{l} x_{il}^2)^{1/2}. \tag{3.1}$$

5. Cluster the rows of Y into K groups using k-means.

6. Finally, assign object s_i to cluster j if and only if row i of the matrix Y has been assigned to cluster j.

In what concerns our key-frame extraction problem, suppose we are given a data set $H = \{H_1, \ldots, H_N\}$ where H_n is the feature vector (normalized color histogram) of the n-th frame. The distance function we consider is the Euclidean distance between the histograms of the frames. As a result, each element of the similarity matrix A is computed as follows:

$$a(i, j) = 1 - \frac{1}{\sqrt{2}} \sqrt{\sum_{h \in bins} (H_i(h) - H_j(h))^2}\ . \tag{3.2}$$

3.2.2 Fast Global k-means

In our method, in the fifth step of the spectral clustering algorithm, instead of using the typical k-means approach, we have used the fast version of the very efficient global k-means algorithm [49]. Global k-means in an incremental deterministic clustering algorithm that overcomes the important initialization problem of the typical k-means approach. This initialization problem has been found to be severe in the case of frame clustering, significantly affecting the quality of the key-frames. Using the global k-means, the obtained key frames usually provide a sensible representation of shot content.

Next, we briefly review the global k-means algorithm. Suppose we are given a data set $X = \{x_1, \ldots, x_N\}$, $x_n \in R^d$ to be partitioned into K disjoint clusters $C_1, C_2, \ldots, C_K$. This algorithm is incremental in nature. It is based on the idea that the optimal partition into K groups can be obtained through local search (using k-means) starting from an initial state with i) the K-1 centers placed at the optimal positions for the (K-1)-clustering problem and ii) the remaining K-th center placed at an appropriate position within the dataset. Based on this idea, the K-clustering problem is incrementally solved as follows. Starting with $k = 1$, find the optimal solution which is the centroid of the data set X. To solve the problem with two clusters, the k-means algorithm is executed N times (where N is the size of the data set) from the following initial positions of the cluster centers: the first cluster center is always placed at the optimal position for the problem with $k = 1$, whereas the second center at execution n is initially placed at the position of data x_n. The best solution obtained after the N executions of k-means is considered as the solution for $k = 2$. In general, if we want to solve the problem with k clusters, N runs of the k-means algorithm are performed, where each run n starts with the k-1 centers initially placed at the positions corresponding to solution obtained for the (k-1)-clustering problem, while the k-th center is initially placed at the position of data x_n. A great benefit of this algorithm is that it provides the solutions for all k-clustering problems with $k \leq K$.

The computational cost of the global k-means algorithm can be reduced without significant loss in the quality of the solution using the fast global k-means algorithm [49]. This method computes an upper bound E_n of the final clustering error obtained by initializing a new cluster center at position x_n. The initial position of the new cluster center is selected as the point x_i that minimizes E_n and k-means runs only once for each k. The application of fast global k-means requires a single execution of k-means for each value (m) of the number of clusters: $m = 1, \ldots, k$.

3.3 Estimation of Number of Key-Frames Using Spectral Graph Theory

As already mentioned in Section 3.1, the number of key-frames cannot be predetermined due to the different content of each shot. In our approach, we attempt to estimate the number of the key-frames using results from the spectral graph theory. Assume we wish to partition dataset S into K disjoint subsets $(S_1, \ldots, S_K)$, and let $X = [X_1, \ldots, X_K] \in \mathbb{R}^{N \times K}$ denote the partition matrix, where X_j is the binary indicator vector for set S_j such that:

$$X(i,j) = 1 \ : \ if \ i \in S_j, \tag{3.3}$$

$$X(i,j) = 0 \ : \ otherwise. \tag{3.4}$$

This clustering problem can be defined as [81]:

$$\max_X \ trace(X^T L X), \tag{3.5}$$

$$s.t. \ X^T X = I_K \text{ and } X(i,j) \in \{0,1\}. \tag{3.6}$$

where L is the Laplacian matrix defined in Section 3.2.1. The spectral clustering algorithm (for K clusters) provides solution to the following *relaxed* optimization problem:

$$\max_Y \ trace(Y^T L Y), \tag{3.7}$$

$$s.t. \ Y^T Y = I_K. \tag{3.8}$$

Relaxing Y into the continuous domain turns the discrete problem into a continuous optimization problem. The optimal solution is attained at $Y = U_K$, where the columns u_i of $U_k, i = 1, \ldots, K$, are the eigenvectors corresponding to the ordered top K largest

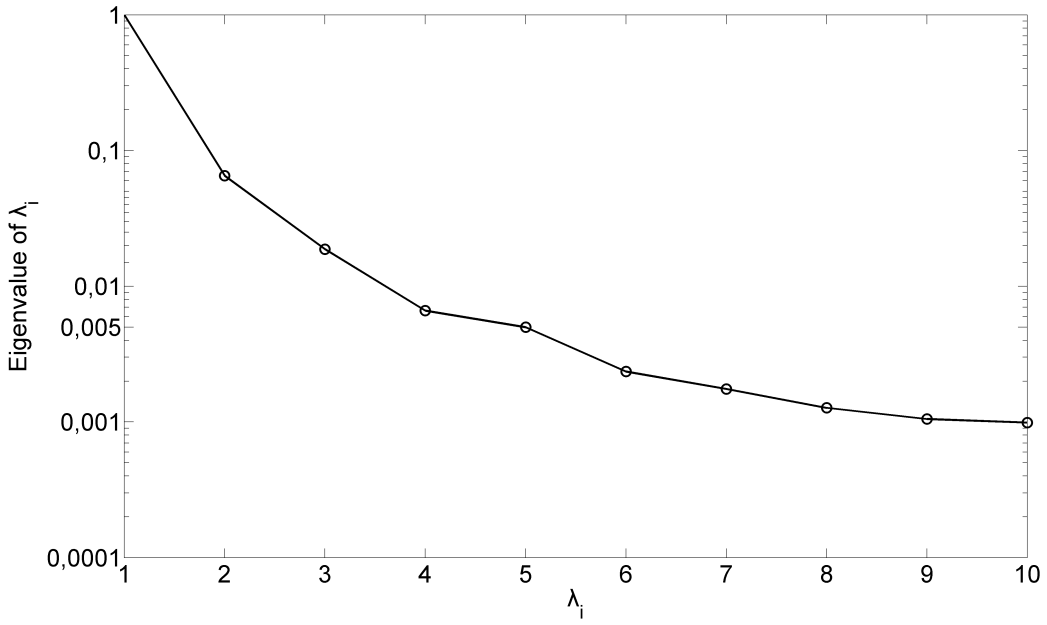


Figure 3.1: Eigenvalues and selection of k.

eigenvalues λ_i of L. Since it holds that [88]:

$$\lambda_1 + \lambda_2 + \ldots + \lambda_K = \max_{Y^T Y = I_K} trace(Y^T L Y) \ , \tag{3.9}$$

the optimization criterion that quantifies the quality of the solution for K clusters and its corresponding difference for successive values of K are respectively given by:

$$sol(K) = \lambda_1 + \lambda_2 + \ldots + \lambda_K, \tag{3.10}$$

$$sol(K+1) - sol(K) = \lambda_{K+1}. \tag{3.11}$$

When the improvement in this optimization criterion (i.e. the value of the λ_{K+1} eigenvalue) is below a threshold, improvement by the addition of cluster K+1 is considered negligible, thus the estimate of the number of clusters is assumed to be K. The threshold value that is used in all our experiments was fixed to Th=0.005 with very good results. In Fig. 3.1, we provide an example of the eigenvalues of a matrix L for a key-frame extraction problem with five clusters (key-frames).

Summarizing, to extract the appropriate key-frames for a shot, we compute the cor-

responding Laplacian matrix L and analyze its eigenvalues to select the number of key-frames k_f. After we have determined k_f, we proceed with the steps 4-6 of the spectral clustering algorithm employing the fast global k-means in step 5, instead of k-means.

3.4 Summary Evaluation

A difficult issue of the key-frame extraction problem is related to the evaluation of the extracted key-frames, since it is rather subjective which frames are the best representatives of the content of a shot. There are several quality measures that can be used to evaluate the efficiency of the algorithms. In [30], two quality measures are used. The first is the Fidelity measure proposed in [8] and the second is the Shot Reconstruction Degree measure proposed in [51].

3.4.1 Average Shot Fidelity

The Fidelity measure compares each key-frame with other frames in the shot. Given the frame sequence $F = \{F_1, F_2, \ldots, F_N\}$ and the set of key-frames $KF = \{KF_1, KF_2, \ldots, KF_{N_{kf}}\}$ the distance between the set of key-frames KF and a frame F_n is defined as:

$$d(F_n, KF) = \min_j \; Diff(F_n, KF_j), \; j = 1, \ldots, N_{kf} \; , \tag{3.12}$$

where N_{kf} is the number of key-frames and $Diff(F_i, F_j)$ is the histogram intersection [74] between two frames F_i and F_j, defined as:

$$Diff(F_i, F_j) = \sum_{h \in bins} \min(H_i(h), H_j(h)) \; , \tag{3.13}$$

where H_i and H_j are the feature vectors (normalized color histograms) of frames F_i and F_j.

However, as mentioned in [51], Fidelity cannot capture well the dynamics of a shot

since it focuses on global details. For that reason we compute the Average Shot Fidelity (ASF) measure which is computed using the average of the minimal distances between the key frame set and the video shot and is given from the following equation:

$$\mathrm{ASF}(F, KF) = 1 - \frac{1}{N}\sum_{n=1}^{N} d(F_n, KF) \ . \tag{3.14}$$

3.4.2 Shot Reconstruction Degree

The whole frame sequence of a shot can be reconstructed from the set of key-frames using an interpolation algorithm. The better the reconstructed video sequence approximates the original sequence, the better the set of key-frames summarizes the video content. More specifically, given the frame sequence F, the set of key-frames KF and a frame interpolation algorithm IA(), we can reconstruct any frame from a pair of key-frames in KF [51]:

$$\tilde{F}_n = \mathrm{IA}(KFn_j, KFn_{j+1}),\ n_j \leq n < n_{j+1} \ . \tag{3.15}$$

The Shot Reconstruction Degree (SRD) measure is defined as follows:

$$SRD(F, KF) = \sum_{n=0}^{N-1} Sim(F_n, \tilde{F}_n) \ , \tag{3.16}$$

where $Sim()$ is given from the following equation:

$$Sim(F_n, \tilde{F}_n) = \log(1/Diff(F_n, \tilde{F}_n)) \ , \tag{3.17}$$

where $Diff(F_i, F_j)$ is the the histogram intersection between frames F_i and F_j, defined in equation (3.13).

3.5 Numerical Experiments

Numerical experiments have been carried out in order to demonstrate the efficiency of the proposed key-frame extraction algorithm. We have also compared our method with existing approaches.

3.5.1 Data for Key-Frame Extraction

To evaluate the performance of our key-frame extraction algorithm we have used two datasets. The first one (Dataset A) consists of seven frame sequences (single-shot) taken from TV-series and sports (Table 3.1), which contain high camera and object motion. The first frame sequence describes an action of a comedy movie that takes place in an office. The next three sequences describe three attempts in a NBA Slam Dunk Contest and the other three a goal attempt in a football match taken from three individual cameras. The second dataset (Dataset B) consists of ten video sequences taken from TV-series and movies (Table 3.2).

Table 3.1: Dataset A characteristics.

Frame Sequence	No. Frames	Genre
F_1	633	Comedy
F_2	144	Basketball
F_3	145	Basketball
F_4	146	Basketball
F_5	225	Football
F_6	300	Football
F_7	172	Football

3.5.2 Comparison with other Key-Frame Extraction Algorithms

We have compared the proposed approach with three other methods. The first one is the simple k-means algorithm applied on the histogram vectors. For each shot, we performed 20 runs of the k-means algorithm keeping as final solution one with the minimum clustering

Table 3.2: Dataset B characteristics.

Video	Duration(min)	Shots	Genre
V_1	22	404	comedy
V_2	31	591	comedy
V_3	30	587	comedy
V_4	23	437	comedy
V_5	27	633	drama
V_6	26	454	drama
V_7	32	377	comedy
V_8	45	608	drama
V_9	31	714	action
V_{10}	26	246	action

error. The number of clusters in k-means algorithm is assumed to be the same as selected using the proposed estimation algorithm (Section 3.3). The second technique used for comparison is presented in [62], as a variant of the method proposed in [92]. Initially, the middle frame of the video sequence is selected as the first key-frame and added to the empty set of key-frames KF. Next, each frame in the video sequence is compared with the current set of key-frames. If it differs from every key-frame in the current set, then it is added into the set as a new key-frame. This algorithm uses a threshold to discriminate whether two frames are similar or not. In our experiments, this threshold parameter is set to such a value that the number of key-frames extracted is the same as in our algorithm. Finally, the third technique is the typical spectral clustering algorithm [56], described in Section 3.2.1 and employing the simple k-means algorithm (20 runs of the k-means algorithm are performed keeping as final solution one with the minimum clustering error).

To evaluate the results of the extracted key-frames we use the metrics mentioned in Section 3.4. More specifically in Tables 3.3 and 3.4, we present the performance results on dataset A, for the ASF and SRD measures respectively. To compute the SRD we use a simple linear interpolation algorithm on the frame's features [51]. The dataset A, which contains high camera and object motion, is used to show the effectiveness of our algorithm in cases where many key-frames are required to represent the shot. It is clear that our approach provides the best summarization of each shot compared to the other methods

Table 3.3: Comparative results of the tested key-frame extraction algorithms using Average Shot Fidelity measure on dataset A.

ASF	Algorithm			
Frame Seq.	Our method	K-means	[62]	Spectral
F_1	0.973	0.9549	0.9616	0.9619
F_2	0.9437	0.9278	0.8913	0.9235
F_3	0.9506	0.9344	0.9268	0.9253
F_4	0.9557	0.948	0.9405	0.9462
F_5	0.9673	0.9467	0.955	0.9625
F_6	0.9558	0.931	0.9424	0.9318
F_7	0.9782	0.9654	0.9672	0.9675

Table 3.4: Comparative results of the tested key-frame extraction algorithms using SRD measure on dataset A.

SRD	Algorithm			
Frame Seq.	Our method	K-means	[62]	Spectral
F_1	1859.66	1533.34	1693.1	1620.6
F_2	424.72	369.87	292.43	362.64
F_3	502.76	430.78	374.23	431.32
F_4	528.09	356.46	340.89	393.02
F_5	843.10	808.2	758.23	780.33
F_6	855.44	753.75	813.1	791.2
F_7	707.92	648.71	642.97	663.15

and the best reconstruction of the original video sequence from the extracted key-frames.

We have also tested our key-frame extraction algorithm and compared it with the other methods using dataset B (TV-series and movies). The measures we have used are : i) Average Video Fidelity, which is the mean of Average Shot Fidelities of each video and ii) Average SRD, which is the mean of the SRD of the shots of each video. In Fig. 3.2 and Fig. 3.3, we present the Average Video Fidelity and the Average SRD, respectively. It is obvious that our key-frame extraction algorithm provides better shot representation and reconstruction than the other three methods.

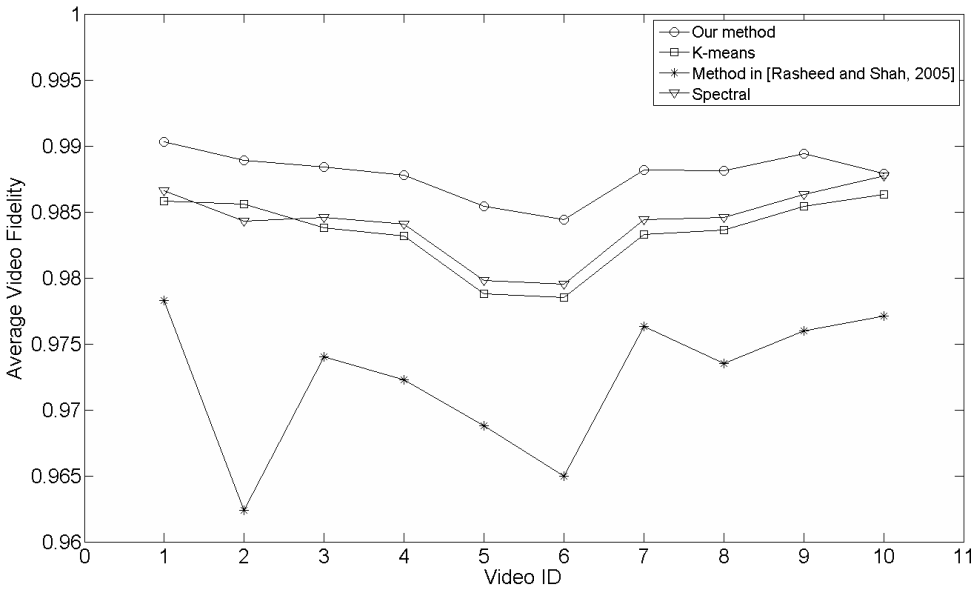


Figure 3.2: Comparative results of the tested key-frame extraction algorithms using Average Video Fidelity measure on dataset B.

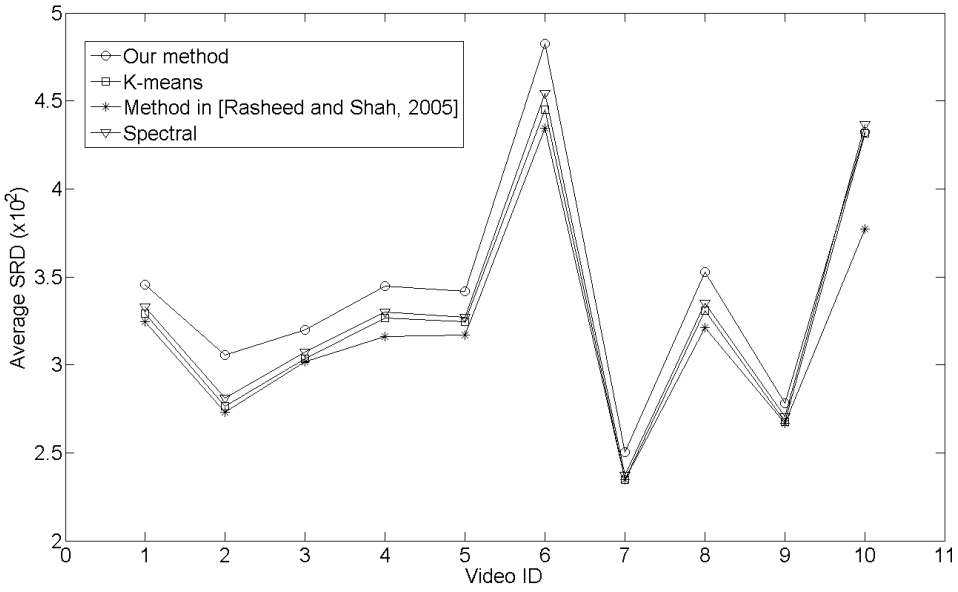


Figure 3.3: Comparative results of the tested key-frame extraction algorithms using Average SRD measure on dataset B.

3.5.3 Video Shot Representation

As already mentioned (Section 3.2.2), a great benefit of the fast global k-means algorithm is that it provides the solutions for all intermediate k-clustering problems with $k \leq K$. In Fig. 3.4 we give an example of the extracted key-frames of a video shot with object and camera motion. Moving from the top to the bottom of this figure we show all intermediate solutions until the selected number of key-frames $N_{kf} = 5$ is reached. The shot that we used contains 633 frames (frame sequence F_1). It shows a woman in an office set-up. This shot can be semantically divided into 5 sub-shots. a) The woman stands against a door eavesdropping and then rushes to her office to pick up the phone that is ringing; b) she talks on the phone, c) lays the receiver of the phone down with a visible effort not to make any noise, d) she rushes back to the door, and e) she continues eavesdropping.

In Fig. 3.5, we provide the key-frames extracted performing the simple k-means algorithm, the algorithm in [62] and the typical spectral clustering algorithm. All algorithms fail to provide a solution adequately describing the visual content of the shot, whereas our approach provides a sensible solution. More specifically, they do not produce any frames for sub-shots (c), (d) and (e) and instead produce multiple frames for sub-shot (a). In contrast, the proposed approach produces key frames for all sub-shots.

In Fig. 3.6, we provide the key-frames for these four algorithms for a video shot describing a slam dunk attempt (frame sequence F_3). It becomes clear that our algorithm summarizes the attempt from the beginning to the end, whereas the other three fail to describe the end of the action.

3.6 Conclusions

In this Chapter, we considered the key-frame extraction problem [10, 14]. In Section 3.2, we described how key-frames are extracted using a spectral clustering approach that employs the fast global k-means algorithm in the clustering procedure. In Section 3.3 we described the estimation of the number of key-frames using results from the spectral

Figure 3.4: Key-frame extraction using the proposed approach of a shot with object and camera motion ($N_{kf} = 5$).

Figure 3.5: Results for the key-frame extraction algorithms used for comparison with ($N_{kf} = 5$). 1st row: K-means. 2nd row: [62]. 3rd row: Spectral Clustering employing simple k-means.

Figure 3.6: Key-frame extraction algorithms in comparison in basketball sequence. 1st row: Our method. 2nd row: K-means. 3rd row: [62]. 4th row: Spectral Clustering employing simple k-means.

graph theory, by examining the eigenvalues of the similarity matrix corresponding to pairs of shot frames. Finally, we evaluated the performance of our algorithm using appropriate quality measures that indicate that our method outperforms traditional techniques and provides efficient summarization and reconstruction of a video sequence from the extracted key-frames.

In the next Chapters, we use the proposed key-frame extraction algorithm for shot representation. The efficient shot representation assists the definition of an effective shot similarity metric since only the most representative frames of the shots are compared and not all the frames. In Chapter 4, we also demonstrate the efficiency of our key-frame extraction algorithm in the scene detection problem. Numerical results show that, the more information we extract about the shot content, the better we can associate shots and detect scene boundaries.

Chapter 4

Segmentation of Videos into Scenes Using Spectral Clustering and Sequence Alignment

4.1 Introduction

Proceeding further towards the goal of video indexing and retrieval requires the grouping of shots into scenes. A *scene* can be regarded as a series of semantically correlated shots. Usually, a *scene* refers to a group of shots that take place in a fixed physical setting (e.g.

a dialogue detection in a room) or a group of shots that describe an action or event (e.g. a car chase by police cars). A more compact representation/segmentation of a video is the merging of scenes into logical story units. The latter, corresponds to the DVD *chapters* describing the different sub-themes of a movie.

Several approaches have been proposed for the scene segmentation problem. In [62], the authors transform this task into a graph partitioning problem. A shot similarity graph is constructed, where each node represents a shot and the edges between shots depict their similarity based on color and motion information. Then, the Normalized cuts [67] method is applied to partition the graph. In [32], a method is proposed for detecting boundaries of the logical story units by linking similar shots and connecting overlapping links. For each shot, all key frames are merged into a larger image and the similarity between shots is computed by comparing these shot images. A similar approach is presented in [84], where a scene transition graph is constructed to represent the video and the connectivity between shots. Then, this transition graph is divided into connected subgraphs representing the scenes. A different approach is presented in [61], where a two-pass algorithm is proposed. In the first pass, shots are clustered by computing backward shot coherence, a similarity measure of a given shot with respect to the previously seen shots, while in the second pass oversegmented scenes are merged based on the computation of motion content in scenes. Another method that uses Markov chain Monte Carlo to determine scene boundaries is proposed in [89]. Two processes, diffusions and jumps, are used to update the scene boundaries that are initialized at random positions. Diffusions are the operations that adjust the boundaries between adjacent scenes, while jump operations merge or split existing scenes.

Most of the above approaches, calculate shot similarity based on visual similarity. Furthermore, they consider the temporal distance of shots as an extra feature that is taken into account when computing the similarity between two shots for shot clustering into scenes. Due to the absence of prior knowledge concerning the video content and the duration of scenes, it is difficult to determine an appropriate weight parameter that will account for the contribution of the temporal distance in the computation of the overall

similarity between shots.

In this Chapter we present an approach [9, 14] where shots are clustered into groups using an improved version of the typical spectral clustering method [56] that uses the fast global k-means algorithm [49] in the clustering stage after the eigenvector computation [10, 14]. In addition, we employ a criterion for estimating the number of groups based on the magnitude of the eigenvalues of the similarity matrix as proposed in the previous Chapter. The resulted groups of shots are not the final scene boundaries, but this clustering procedure is a preprocessing step towards the final detection of scene boundaries.

Another novelty of our method is that shot similarity is computed based only on visual features, because incorporating time distance in a shot similarity metric requires a priori knowledge of the scene duration. Thus, it is a quite difficult task to determine a distance parameter that defines whether two shots are related or not. In our method, cluster labels are assigned to shots according to their visual content and then, sequences of shot labels are compared to identify changes in the patterns of successive labels. In that way, time distance between shots is not taken into account since our method locally searches for changes in patterns of shot labels ignoring the relation between shots with respect to time distance.

Typically, the sequence of shots in a video follows specific production rules. The most common is known as the 180° rule, where the director draws a line in the physical setting of a scene and all cameras are placed on the same side of this line [73]. This production rule produces repeating shots of one person, a group of persons or the same setting which is commonly seen in movies, documentaries and TV-series. The most common patterns of repetitive shots are two. The first one is a dialogue between two or more persons, where the camera switches from one person to another, thus producing a sequence of shots like $ABABCBCABABC$, where A, B and C are the shot labels for three different persons. Another common pattern is a sequence of shots like $A_1A_2A_1A_3A_3A_2$ where A_1, A_2 and A_3 are captions of three different cameras providing views of the same physical setting from different angles. When a scene changes it is expected that a change in such patterns will occur. For example, if two dialogues take place in different scenes, it is expected

that a sequence of shots like $ABABCBDEDFEDEF$ is produced where $ABABCB$ corresponds to the first scene and $DEDFEDEF$ corresponds to the second scene. To identify the change in pattern, a comparison of successive non-overlapping windows of shot labels is performed. Thus, we need to define a proper measure to define whether two sequences are related (share the same patterns of shots) or not. A very efficient category of algorithms that compare sequences in order to define whether two sequences are related or not are the sequence alignment algorithms that are successfully used in biological applications [39].

In our approach, to compare sequences we use the "Needleman - Wunsch" global sequence alignment algorithm [55], which performs global alignment on two sequences and is guaranteed to find the alignment with the maximum score. This algorithm requires the definition of a substitution matrix in order to implement the alignment. This matrix represents the rate at which one character in a sequence changes to another character over time. In our method, the substitution matrix is formulated based on criteria that are adapted to the problem of scene detection. Color similarity between clusters of shot labels and probability of existence of a pair of successive shot labels are the two components that contribute to the substitution matrix. The score of each alignment is given through a scoring function which takes into account matches, mismatches and gaps of shot labels. When an alignment gives a low score, a change in the patterns of shot labels is implied and suggests a scene boundary. The proposed two-stage approach (shot clustering, sequence alignment) achieves high correct detection rates while preserving a good trade off between the number of missed scenes and the number of falsely detected scenes.

In Fig. 4.1 we summarize the main steps of our approach and the algorithms employed in these steps. The video is segmented into shots and the spectral clustering algorithm of the previous Chapter is employed to extract the key-frames of the corresponding shots. Next, shots are grouped with respect to their visual similarity and labeled according to the group they are assigned. Finally, a sequence alignment algorithm is implemented to identify high dissimilarities between successive windows of shot labels. Scene boundaries are considered to be the points of high dissimilarity.

The rest of the Chapter is organized as follows: In Section 4.2, the procedure of video shot representation and similarity is described. In Section 4.3, the proposed scene detection algorithm is presented. In Section 4.4, we present numerical experiments and compare our method with two other methods proposed in [62] and [84]. Finally, in Section 4.5, we provide some conclusions.

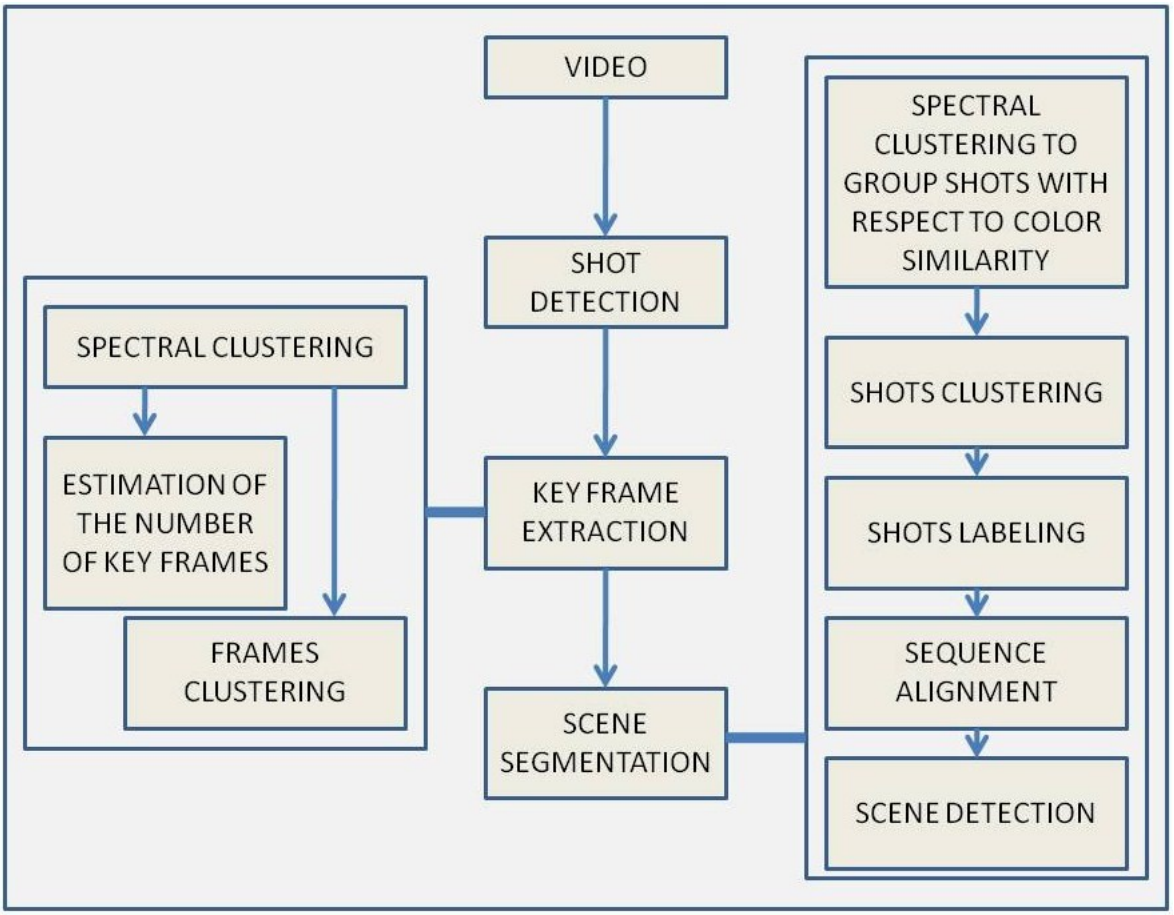


Figure 4.1: The main steps of our scene segmentation method.

4.2 Video Shots Representation

4.2.1 Key-Frame Extraction and Shot Detection

In order to proceed with video segmentation into scenes, the volume of video data to be processed must be reduced. It is required to start with the video segmentation into shots

and continue with the efficient shot representation. In this way, a video comprised of thousands of frames can be efficiently represented using only several hundreds of frames. In our approach [14], each video is manually segmented into shots and each shot is represented with key-frames extracted by applying the algorithm proposed in Section 3.2. Each frame is represented by an 16-bin HSV normalized histogram with 8 bins for hue and 4 bins for each of saturation and value. The frames of the shot are clustered into groups using the method of the previous Chapter which is based on an improved version of the typical spectral clustering method [56] that uses the fast global k-means algorithm [49] in the clustering stage after the eigenvector computation. Then, the medoid of each group, defined as the frame of the group whose average similarity to all other frames of this group is maximal, is characterized as a key-frame.

4.2.2 Shot Similarity

As explained earlier, shots that belong to the same scene often have similar color content. As suggested in [62], the visual similarity between a pair of shots i and j can be computed as the maximum color similarity ($ColSim$) among all possible pairs of their key-frames:

$$VisSim(i,j) = \max_{p \in K_i, q \in K_j} ColSim(p,q) \ , \tag{4.1}$$

where K_i and K_j are the sets of key-frames of shots i and j respectively, and the color similarity ($ColSim$) between two frames f_i, f_j is defined as the histogram intersection [74]:

$$ColSim(i,j) = \sum_{h \in bins} \min(H_i(h), H_j(h)) \ , \tag{4.2}$$

where H_i, H_j are the HSV normalized color histograms of frames f_i and f_j respectively.

4.3 Scene Detection Algorithm

Scene detection is a quite difficult task, because a scene is a group of shots that are i) semantically correlated and ii) continuous in time. The semantic correlation between two shots cannot actually be described with low-level features. However, low-level features such as color give useful information about the connection between shots and the physical setting where the scene takes place. On the other hand, taking into account the contribution of temporal distance in the computation of the overall similarity between shots is difficult, due to the absence of prior knowledge about the scene duration.

4.3.1 Shots Clustering

In order to perform scene detection, clustering of shots into groups is required, taking into account visual similarity ($VisSim$) and time adjacency. Suppose there is a set $V = \{v_1, v_2, \ldots, v_N\}$ of N shots, ordered in time, to be segmented. In order to implement shot grouping, an $N \times N$ similarity matrix A must be specified. In [58, 62], both visual similarity and time distance are combined in a single similarity metric (see Section 4.4.1). On the contrary, in our method we have considered only visual similarity (equation (4.1)) for shot clustering:

$$a(i,j) = VisSim(v_i, v_j),\ v_i, v_j \in V\ , \tag{4.3}$$

while ordering of shots is taken into account at a later processing stage.

After the similarity matrix A has been computed, the modified spectral clustering algorithm is used to group shots into clusters. The main steps of this algorithm have been presented in Section 3.2.1. The selection of the number of shot clusters is done in a way similar to the key-frame extraction problem in Section 3.3. However it is worth mentioning that the number of shot clusters is not equal to the number of scenes in the video. Our aim is to estimate the principal color distributions over the video shots and group all shots according to the color distribution that they fit most. Following the same approach proposed for key-frame extraction, the analysis of the eigenspectrum of the

Laplacian matrix L provides an estimate of the number of clusters K. Then, shots are clustered into K groups with respect to their visual content (color histogram similarity (equation (4.1)), while the final number of scenes will be extracted at a later step of our algorithm.

Once the spectral clustering algorithm has provided a partition of the shots into K clusters $\{C_1, C_2, \ldots, C_K\}$, a label is assigned to each shot according to the cluster it belongs, thus producing a symbolic sequence of labels. In this way, the sequence of shots is transformed into a new sequence of labels that illustrates the visual similarity between shots. An illustrative example is given in Fig. 4.2:

V_{01}	V_{02}	V_{03}	V_{04}	V_{05}	V_{06}	V_{07}	V_{08}	V_{09}	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}	V_{19}	V_{20}	V_{21}
C_1	C_1	C_1	C_1	C_1	C_2	C_2	C_2	C_2	C_3	C_5	C_3	C_5	C_3	C_5	C_3	C_4	C_4	C_2	C_4	C_4

Figure 4.2: Video sequence of labels.

To each shot V_t (the index t implies time) a label from the set $\{C_1, C_2, C_3, C_4, C_5\}$ is assigned to. Typically, during a scene there exists a sequence of similar shot labels (different captions of the same person/place) or a sequence of repetitive label patterns (rotation of different camera captions, eg. dialogue). We consider that a scene change occurs when the pattern of symbols changes. In our example, distinct scenes correspond to shots with time indices 1-5, 6-9, 10-16 (repetitive pattern C_3C_5) and 17-21. In practice, due to the presence of noise (shot V_{19} with label C_2), it is not trivial to discriminate patterns of symbols. To efficiently segment the symbolic sequence into segments-scenes, two methods are suggested in this Chapter. In the first one [9], a comparison of successive pairs of labels is employed. In the second one [14], a sequence alignment algorithm is used that provides the best performance.

4.3.2 Symbolic Sequence Segmentation

In order to account for noise in the sequence (shot C_2 in the last scene in Fig. 4.2) and for the case of repetitive patterns (scene V_{10}-V_{16} in Fig. 4.2) we form a new sequence

containing *pairs of successive labels* (this is analogous to *two-grams* in traditional string processing). In this sequence of pairs, successive similar pairs are considered to belong to the same scene. More specifically, in order to merge successive pairs in the same scene we check for the existence of at least one similar label in both pairs. If this is not the case, a scene boundary is identified and we consider that a new scene starts from the next pair in the sequence.

The algorithm [9] proceeds in two passes. In pass one, the first two shots are regarded as the first pair, whereas in pass two the second and the third shots are regarded as the first pair. As we can see in Fig. 4.3, during the first pass the scene boundary (denoted as $\downarrow$) between clusters 1 and 2 is not detected since shots that belong to different scenes form a pair. However, in the second pass this boundary is clearly detected. The final boundaries are the union of the boundaries detected from both passes.

Original sequence
$C_1C_1C_1C_1C_1C_2C_2C_2C_2C_3C_5C_3C_5C_3C_5C_3C_4C_4C_2C_4C_4$

Shot pairs sequence - First Pass
$(C_1C_1)(C_1C_1)(C_1C_2)(C_2C_2)(C_2C_3)(C_5C_3)(C_5C_3)(C_5C_3)\downarrow(C_4C_4)(C_2C_4)C_4$

Shot pairs sequence - Second Pass
$C_1(C_1C_1)(C_1C_1)\downarrow(C_2C_2)(C_2C_2)\downarrow(C_3C_5)(C_3C_5)(C_3C_5)(C_3C_4)(C_4C_2)(C_4C_4)$

Figure 4.3: Video sequence of labels.

4.3.3 Sequence Segmentation Through Sequence Alignment

However, there are some disadvantages in the method proposed in Section 4.3.2. The first one is that shots with different labels (for example shot i with label C_1 and shot j with label C_2) are considered as totally different. As already mentioned in Section 4.1 a common pattern in a video shot sequence is a sequence of shots like $A_1A_2A_1A_3A_3A_2$ where A_1, A_2 and A_3 are captions of three different cameras providing views of the same physical setting form different angles. Suppose that views A_1 and A_2 are assigned to cluster C_1 and view A_3 is assigned to cluster C_2. Then the shot sequence $A_1A_2A_1A_3A_3A_2$

is transformed to label sequence $C_1C_1C_1C_2C_2C_1$. If we apply the method described in Section 4.3.2 a scene boundary will be falsely detected between shots 3 (C_1) and 4 (C_2). Thus, we should take into consideration the similarity between clusters C_1 and C_2, which in the above example is quite high since they describe the same physical setting. In other words, we need to find a more robust "label similarity" metric between shots labels. Next, we propose the use of a sequence alignment algorithm that employs a substitution matrix and fits better to the problem of scene detection.

The second disadvantage of the method described in Section 4.3.2 is the existence of multiple irrelevant shots (noise) in the symbolic sequence of labels. Suppose the following sequence of shot labels describing a scene $C_4C_4C_2C_1C_4C_4$. The noisy shots have labels C_2 and C_1. If there has been only one such shot, the method proposed in Section 4.3.2 would successfully not detect a scene boundary. In this case however, a scene boundary would be detected regarding shot labels C_2C_1 as a separate scene. To avoid such occasions, we have to extend the window of shots compared and take into account the possible similarity of shot labels C_2, C_1 with shot label C_4. In the following, the sequence alignment algorithm employed for scene detection compares successive sequences of shots labels of length larger than 2.

As already mentioned in the introduction, videos such as movies, documentaries and TV-series, follow some production rules. These rules result in the generation of patterns of shots inside a scene. Different scenes share different patterns of shots (different sub-sequences of labels). Thus, it is expected to detect scene changes in cases where the pattern of shot labels changes. In order to find the points in the sequence of shot labels where the pattern of symbols changes, we compare successive non-overlapping windows of shot labels using a sequence alignment algorithm. More specifically, given the set V of N shots, the sub-sequences of the original video sequence to be compared at each iteration i are formulated as:

$$X_1^i = L_iL_{i+1}\dots L_{i+w-1} \text{ and } X_2^i = L_{i+w}L_{i+w+1}\dots L_{i+2w-1}, \tag{4.4}$$

where $i = 1, \ldots, N-2w$, w is the length of the window used and $L_j,\ j = 1, \ldots, N$ are the shot labels. In Fig. 4.4, the first three sub-sequences of the video sequence in Fig. 4.2 are shown, using a window of length 4. In iteration 1 the first sub-sequence containing shots $V_1 - V_4$ will be compared with sub-sequence containing shots $V_5 - V_8$. In next iteration the two sub-sequences under comparison are those containing shots $V_2 - V_5$ and $V_6 - V_9$ respectively.

$$\overbrace{\begin{array}{c} V_{01}V_{02}V_{03}V_{04}V_{05}V_{06}V_{07}V_{08}V_{09}V_{10}V_{11}V_{12}V_{13}V_{14}V_{15}V_{16}V_{17}V_{18}V_{19}V_{20}V_{21} \\ C_1\ C_1\ C_1\ C_1\ C_1\ C_2\ C_2\ C_2\ C_2\ C_3\ C_5\ C_3\ C_5\ C_3\ C_5\ C_3\ C_4\ C_4\ C_2\ C_4\ C_4 \end{array}}^{Original Sequence}$$

$$\text{Iteration 1} \rightarrow \underbrace{(C_1C_1C_1C_1)}_{X_1^1}\underbrace{(C_1C_2C_2C_2)}_{X_2^1}$$

$$\text{Iteration 2} \rightarrow \underbrace{(C_1C_1C_1C_1)}_{X_1^2}\underbrace{(C_2C_2C_2C_2)}_{X_2^2}$$

$$\text{Iteration 3} \rightarrow \underbrace{(C_1C_1C_1C_2)}_{X_1^3}\underbrace{(C_2C_2C_2C_3)}_{X_2^3}$$

Figure 4.4: Sub-sequences to be compared.

A well established approach to compare sequences of symbols is the sequence alignment algorithm. Significant similarity between sequences may imply that the sequences belong to the same scene. Our interest however, focuses on cases of high dissimilarity that is a strong indication of a scene boundary. The sequence alignment algorithm we used in our approach is the "Needleman-Wunsch" algorithm [55] which is commonly used in bioinformatics to align protein or nucleotide sequences. This algorithm performs global alignment on two sequences and is guaranteed to find the alignment with the maximum score. The input consists of two sequences of length w as described in equation (4.5). Let us denote

$$X_1 = L_1L_2 \ldots L_w \quad and \quad X_2 = M_1M_2 \ldots M_w \quad . \tag{4.5}$$

The labels $L_i, M_i,\ i = 1, \ldots, w$ belong to some alphabet of K symbols, where K is the number of cluster labels generated from the spectral clustering of shots. To align these sequences, a $w \times w$ matrix N is constructed where the value $N(i, j)$ is the score of the best alignment between the segment $X_1(1 \ldots i)$ and the segment $X_2(1 \ldots j)$ [39]. There

are three possible ways to obtain the best score of an alignment up to $X_1(i)$, $X_2(j)$: a) $X_1(i)$ could be aligned to $X_2(j)$, b) $X_1(i)$ could be aligned to a gap and c) $X_2(j)$ could be aligned to a gap. The best score will be the largest of these three options:

$$N(i,j) = \begin{cases} N(i-1,j-1) + S(X_1(i), X_2(j)) \\ N(i-1,j) - d \\ N(i,j-1) - d \end{cases}, \tag{4.6}$$

where S is a substitution matrix and d is a gap penalty. The definition and calculation of these quantities are given below. The traceback from $N(w,w)$ to $N(0,0)$ defines the optimal alignment of X_1 and X_2. The time complexity for aligning two sequences of length w is $O(w^2)$. A typical example of a sequence alignment over an alphabet $\{C_1, C_2, C_3, C_4\}$ is given in Fig. 4.5. The output of the alignment algorithm is an alignment matrix. The columns of this matrix that contain the same label in both rows are called matches (M), while columns containing different letters are called mismatches (m). The columns of the alignment containing one space are called gaps (G). A gap in an alignment is defined as a contiguous sequence of spaces in one of the rows of the alignment matrix [39]. By inserting one or more gaps, the algorithm succeeds in aligning symbols that occur in different positions.

Seq_1 : $C_1C_2C_1C_2C_3C_4C_1C_1C_3C_4C_4$
Seq_2 : $C_4C_1C_2C_1C_2C_2C_3C_4C_4C_1C_3C_4$

Output : (Alignment matrix)

Seq_1	-	C_1	C_2	C_1	C_2	-	C_3	C_4	C_1	C_1	C_3	C_4	C_4
Seq_2	C_4	C_1	C_2	C_1	C_2	C_2	C_3	C_4	C_4	C_1	C_3	C_4	-
Type	G	M	M	M	M	G	M	M	m	M	M	M	G

Figure 4.5: Alignment matrix of a sequence alignment example.

The sequence alignment algorithm requires a substitution matrix S and a gap cost function δ. In our problem, the elements $s(i,j)$ of the substitution matrix S express how similar are shot labels C_i and C_j in terms of color and position. The color similarity

between shot labels can be defined from the similarity of their respective clusters. In what concerns position, it can be observed that during a scene, repetitive patterns of labels frequently occur. This increases the possibility that a shot label i can be aligned with a shot label j and the opposite with high score, when shot labels i and j belong to the same pattern, thus the similarity between shot labels, as far as position is concerned, can be expressed through the possibility that a shot label i precedes or follows a shot label j. As a result, the substitution matrix S is defined as the combination of two different similarity metrics. Next, we define these similarity metrics, one for color similarity and one for position similarity, and how they are combined to formulate matrix S.

For color similarity, for each cluster i we compute the medoid m_i, defined as the shot of a cluster, whose average similarity to all the other shots of this cluster is maximal. Then, the visual similarity between shot clusters can be computed from the visual similarity between the corresponding medoids, thus producing a cluster similarity matrix (CSM):

$$CSM(i,j) = VisSim(m_i, m_j),\ m_i, m_j \in Med \tag{4.7}$$

where $VisSim$ is given from equation (4.1) and Med is the set of the medoids of the clusters. Next, we compute a pair probability matrix (PPM) which represents the probability (frequency) of existence of a pair of sequential labels in the video. There are N-1 pairs of successive labels in a video containing N shots and the PPM matrix is given from the following equation:

$$PPM(i,j) = \frac{1}{N-1}\{\#\text{ pairs}(L_1 = C_i, L_2 = C_j)\}\ , \tag{4.8}$$

where L_1, L_2 are the first and the second label of a pair respectively and $i, j = 1, \ldots, K$. The final substitution matrix S is computed as follows:

$$S(i,j) = \begin{cases} A(i,j) + B(i,j) & , i = j \\ -\alpha(1 - A(i,j)) - \beta(1 - B(i,j)) & , i \neq j \end{cases}\ , \tag{4.9}$$

where A and B are the CSM and PPM matrices respectively and α, β with $\alpha+\beta = 1$, are weights controlling the contribution of each matrix element. Each entry (i, j) of the matrix represents the score of alignment of the ith and jth symbols in the alphabet. The diagonal elements of matrix S account for match operations, while the non-diagonal elements account for the mismatch operations during the alignment procedure. To represent the cost of having a gap of length l we consider the linear gap model $\delta(l) = -ld$, where d is a nonnegative constant called the "linear gap penalty" and is set to 1.

After the formulation of the substitution matrix, the sequence alignment algorithm computes the score for the best alignment in each iteration (Fig. 4.4). The evaluation of the alignment is based on the number of matches, mismatches and gaps between the sequences. A scoring function [39] is defined as:

$$F = (\text{score of matches}) - (\text{score of mismatches}) - (\text{score of gaps}). \tag{4.10}$$

In Fig. 4.6 we illustrate the computation of this scoring function for the previous sequence alignment example using a similarity matrix with score +1 for matches (M), -1 for mismatches (m) and a linear gap (G) function with d =1.

Seq_1 : $C_1C_2C_1C_2C_3C_4C_1C_1C_3C_4C_4$
Seq_2 : $C_4C_1C_2C_1C_2C_2C_3C_4C_4C_1C_3C_4$

Seq_1	-	C_1	C_2	C_1	C_2	-	C_3	C_4	C_1	C_1	C_3	C_4	C_4
Seq_2	C_4	C_1	C_2	C_1	C_2	C_2	C_3	C_4	C_4	C_1	C_3	C_4	-
Type	G	M	M	M	M	G	M	M	m	M	M	M	G
Score	-1	1	1	1	1	-1	1	1	-1	1	1	1	-1

Figure 4.6: Scoring function of the sequence alignment example.

We apply the above sequence alignment procedure to all pairs of subsequences (X_1^i, X_2^i), $i = 1, \ldots, N - 2w$. The values of the scoring function are stored in a score sequence SC. In Fig. 4.7, an example of the score sequence values is shown. At the scene boundaries a change in the pattern of labels occurs, thus it is expected to observe a low score value. In other words, low score values are considered as indicators of the possibility for scene

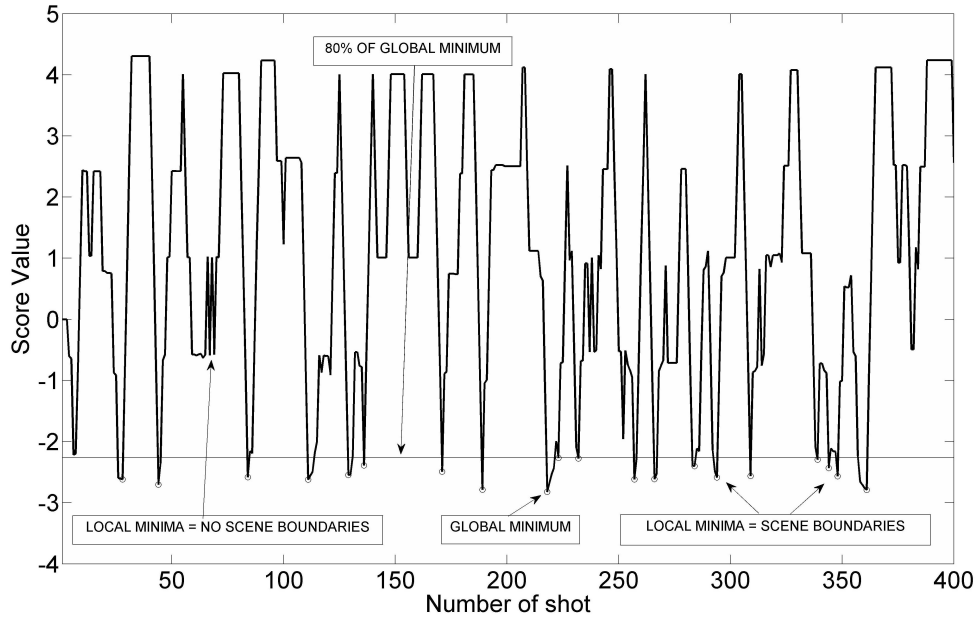


Figure 4.7: Scoring sequence of a sequence alignment example.

change. The global minimum of the score sequence corresponds to the most dissimilar sub-sequences in the video, thus to the most certain scene boundary. Since there are many local minima in the score sequence, it is expected that those with value close to the global minimum to correspond to the most probable scene boundaries. To locate these boundaries we first find the global minimum value in sequence SC. Then, the local minima of the sequence SC that are less than a percentage of the global minimum value are characterized as scene boundaries. In our experiments, a percentage equal to 80% was used providing very good results.

4.4 Numerical Experiments

4.4.1 Data and Performance criteria

To evaluate the performance of our scene detection algorithm, we used Dataset B (see Table 3.2), which contains video sequences taken from TV-series and movies. The commonly used criteria in equations (2.16), (2.17) and (2.18), where N_c stands for the number of correct detected scene boundaries, N_m for the number of missed ones and N_f the number of

false detections, were used to evaluate our method and the algorithms under comparison.

In Fig. 4.8, the average performance of our algorithm on all videos is presented, varying the length of the window w, (which defines the length of the sequences to be aligned) from 2 to 8. It can be observed that even for $w = 8$, the algorithm yields very good results. We believe that the choice of $w = 4$ is preferable because, apart from reducing the possibility of missing a scene with a small number of shots, it is sufficiently large for a reliable comparison during the sequence alignment algorithm.

To detect the final scene boundaries, as already mentioned in Section 4.3.3, we select the local minima of the SC sequence that are less than a percentage Th of its global minimum. In Fig. 4.9, the average F_1 values (for $w = 4$) for all videos are presented, for Th varying from 0.7 to 0.95. It can be observed that for any Th from 0.7 to 0.85 our algorithm provides very good result achieving the best performance for $Th = 0.8$. In Fig. 4.10, we present the F_1 values for $w = 4$ and $Th = 0.8$ varying the weight parameter α, which controls the contribution of the matrices CSM and PPM, from 0 to 1. The best performance is achieved for $\alpha = 0.5$. It can be observed that for $\alpha = 1$ the performance is very low, thus indicating that the use of the PPM matrix is beneficial. In Table 4.1 we present the recall, precision and F_1 values for $w = 4$, $Th = 0.8$ and $\alpha = 0.5$ for our algorithm [14]. It can be observed that our approach achieves high correct detection rate while keeping small the number of false detections.

To demonstrate the efficiency of the string comparison method, we also implemented another approach where subsequences are simply considered as *sets* of labels and their similarity is measured using the similarity of the corresponding histograms of labels (as an extension of the comparison of pairs of shot labels in Section 4.3.2). In Fig. 4.11 we present the F_1 values comparing the set comparison and our method (string comparison using sequence alignment). It is clear that the structure of the label sequence assists in the scene detection problem. In Table 4.1, we also present the performance of the algorithm presented in Section 4.3.2, [9].

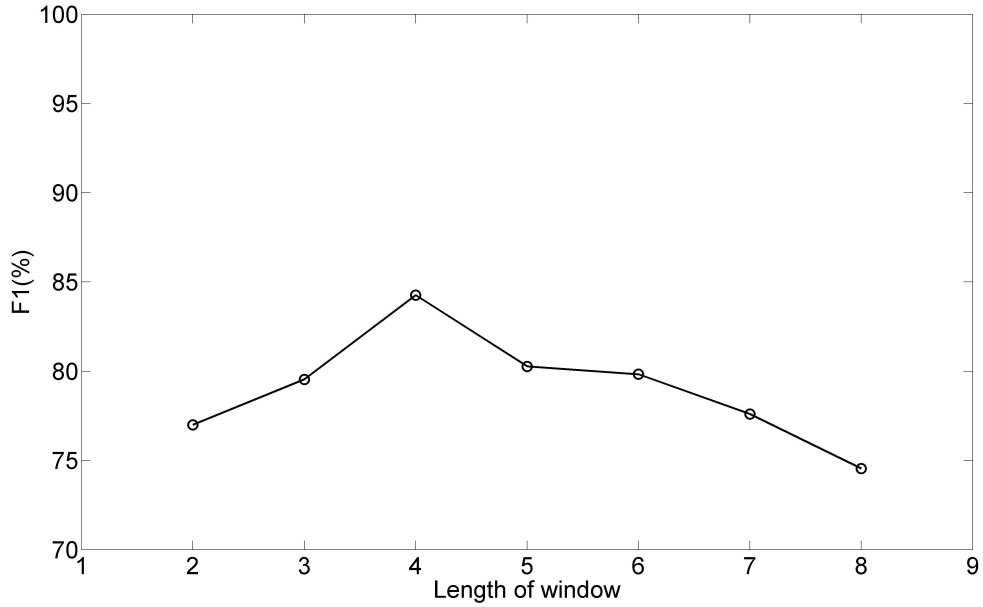


Figure 4.8: Average performance results for different values of the window parameter w.

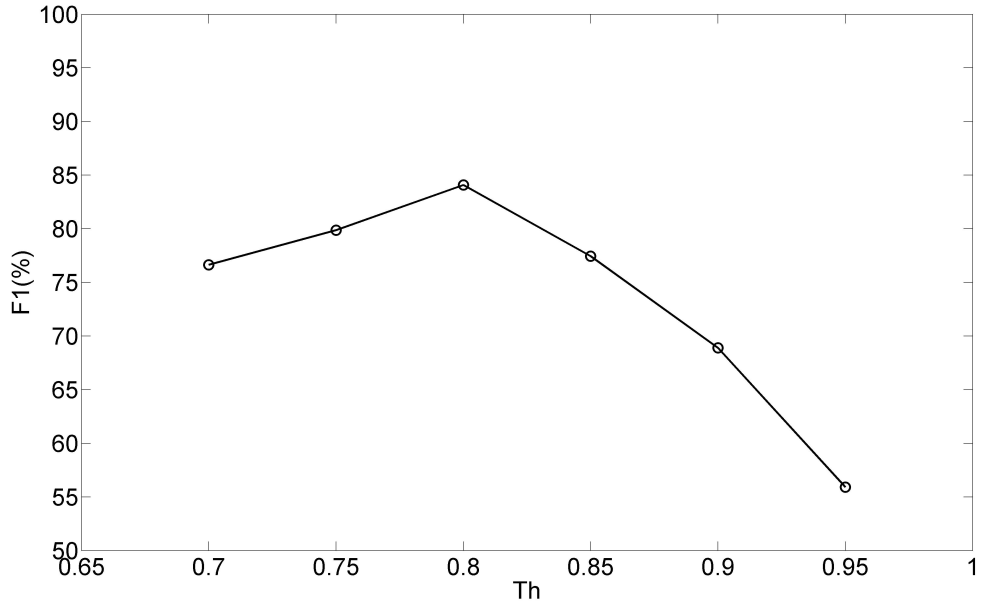


Figure 4.9: Average performance results for different values of the Th parameter and w = 4.

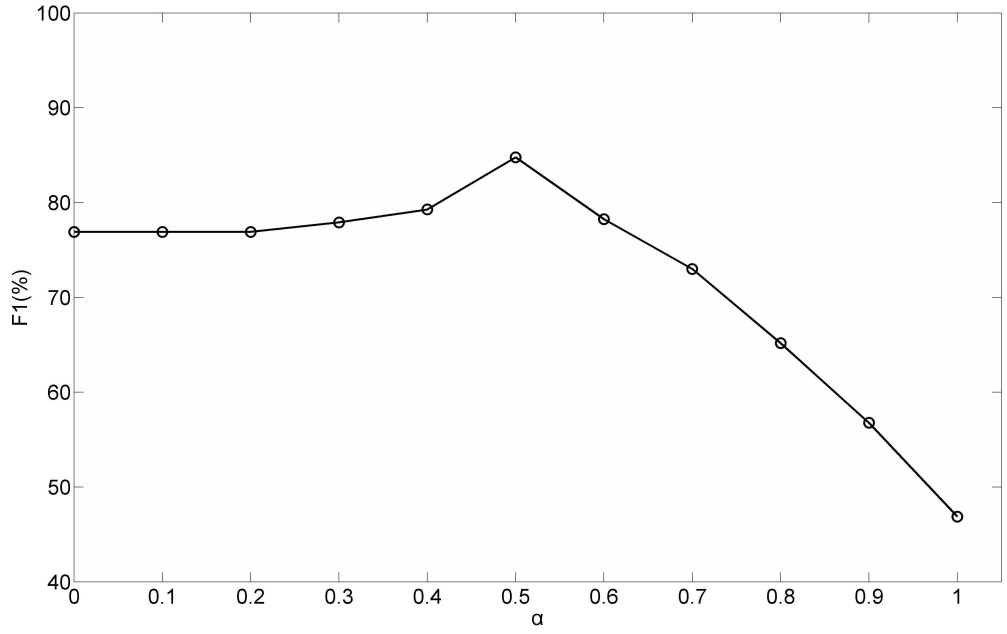


Figure 4.10: Average performance results for different values of the a parameter and $w = 4$, $Th = 0.8$.

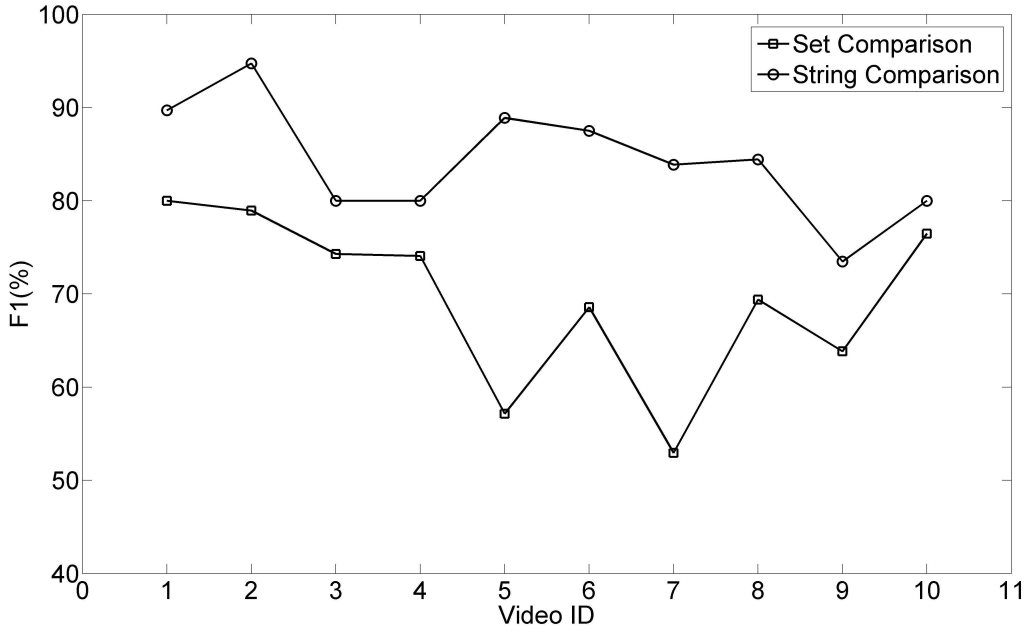


Figure 4.11: Scene detection results (using F_1 measure) when subsequences are considered as i) strings (compared using sequence alignment) and ii) sets of labels (compared using histogram similarity).

4.4.2 Comparison

To compare the effectiveness of our approach, we have also implemented two other methods. The first one is proposed in [62]. This method computes both color and motion similarity between shots and the final similarity value is weighted by a decreasing function of the temporal distance between shots given by the following equation:

$$w_t(i,j) = e^{-\frac{1}{d}|\frac{m_i - m_j}{\sigma}|^2} \ , \tag{4.11}$$

where m_i and m_j are the time indices of the middle frames of the two shots under consideration and σ the standard deviation of the shots duration in the entire video. The parameter d plays a critical role in the final number of scenes produced by the algorithm. The final shot similarity matrix defines a weighted undirected graph where each node represents a shot and the edges are the elements of the matrix. To partition the video into scenes, an iterative application of Normalized cuts method [67] was used that divides the graph into subgraphs. It must be noted that the implementation of the Normalized cuts method in this approach does not require the computation of eigenvectors, because scenes are composed of shots which are time continuous. Thus a cut can be made along the diagonal of the shot similarity matrix. The $Ncut$ algorithm is applied recursively as long as the $Ncut$ value is below some stopping threshold T. We have implemented and tested this method using the same video set for different values of the threshold parameter T and the parameter d (equation (4.11)). Determination of optimal values for these parameters is a tedious task. In our comparisons we found distinct values for each video that provide the best performance. The recall, precision and the F_1 values of the experiments are presented in Table 4.1, [62].

The second method has been proposed in [84]. This method clusters shots into groups taking into account the visual characteristics and temporal dynamics of video. Then, a *scene transition graph* which is a graphical representation of the video is constructed. The nodes of this graph represent the shots and the edges the transitions between the shots. To find the scenes, this graph is partitioned into connected subgraphs. The above algorithm

depends on two parameters. The first one is the parameter δ which defines the minimum separation between any two resulting clusters and controls the final number of clusters. The second parameter is T that defines two shots to belong in different clusters if they are not close to each other. After the initial segmentation, the segmented scenes are refined by adjusting the threshold parameter T to reflect the duration of scenes. Determination of optimal values for the parameters δ and T is a tedious task. To test the performance of this algorithm we executed multiple runs using different values for the parameters δ and T. In our comparisons, we used distinct values for each video that provide the best performance. The recall, precision and the F_1 values of the experiments are presented in Table 4.1, [84].

In Fig. 4.12, the F_1 values of the three examined methods are graphically presented. It is clear that our algorithm provides the best F_1 value for all videos, and in general our method outperforms the other approaches. Finally, to show that a sensible representation of a shot by its key-frames contributes to the scene detection problem, we carried out the following experiment. We implemented our scene detection algorithm using as key-frames for the shots those extracted the method proposed in Chapter 3 and the other three methods mentioned in Chapter 3. The F_1 values of the four examined methods are presented in Fig. 4.13. It is obvious that the better the shot is represented by its key-frames, the better our algorithm detects scene boundaries.

All three algorithms were implemented in Matlab. Considering the scene detection problem for the first video sequence, our algorithm and the method in [62] took approximately the same time to identify the scene boundaries, whereas the method in [84] took approximately five times more than the first two.

4.5 Conclusions

In this Chapter, we have proposed a new method for video scene segmentation. First, key-frames are extracted using a spectral clustering method employing the fast global

Table 4.1: Comparative results of the tested scene detection algorithms ([14] - C2009, [9]- C2007, [62] - R2005, [84] - Y1998) using Recall(R), Precision(P) and F_1 measures.

		C2009	C2007	R2005	Y1998
V_1	R(%)	86.67	86.60	86.67	60.00
	P(%)	92.85	72.20	61.90	81.82
	F_1(%)	89.70	78.70	72.22	69.23
V_2	R(%)	100.00	93.30	83.33	72.22
	P(%)	90.00	88.20	62.50	68.42
	F_1(%)	94.74	90.70	71.43	70.27
V_3	R(%)	87.50	81.30	81.25	87.50
	P(%)	73.68	65.00	52.00	70.00
	F_1(%)	80.00	72.20	63.41	77.78
V_4	R(%)	76.92	84.60	92.31	76.92
	P(%)	83.33	64.70	60.00	71.43
	F_1(%)	80.00	73.30	72.73	74.07
V_5	R(%)	85.71	92.50	92.86	78.57
	P(%)	92.31	76.40	63.16	64.71
	F_1(%)	88.89	83.70	75.18	70.97
V_6	R(%)	82.35	80.00	76.47	70.59
	P(%)	93.33	60.00	61.90	66.67
	F_1(%)	87.50	68.60	68.42	68.57
V_7	R(%)	86.67	73.30	86.67	80.00
	P(%)	81.25	91.60	61.90	75.00
	F_1(%)	83.87	81.40	68.42	77.42
V_8	R(%)	76.00	72.00	80.77	71.43
	P(%)	95.00	85.70	70.00	74.07
	F_1(%)	84.44	78.30	75.00	72.73
V_9	R(%)	72.00	68.00	80.77	64.00
	P(%)	75.00	58.60	55.26	59.26
	F_1(%)	73.47	63.00	65.63	61.54
V_{10}	R(%)	70.00	75.00	75.00	68.42
	P(%)	93.33	78.95	75.00	72.22
	F_1(%)	80.00	76.82	75.00	70.27

k-means algorithm in the clustering phase and also providing an estimate for the number of the key-frames. Then, shots are clustered into groups using only visual similarity as a feature and they are labeled according to the group they are assigned. Shot grouping is achieved using the same spectral clustering method proposed for key-frame extraction.

After shot grouping, shots are labeled according to the cluster they are assigned. Since

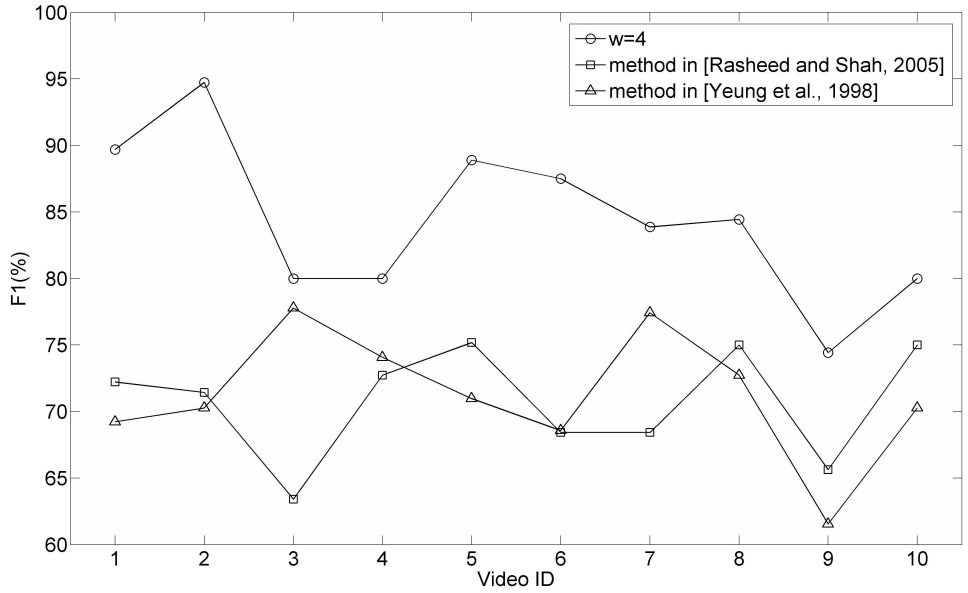


Figure 4.12: Scene detection results (using F_1 measure) comparing three scene detection algorithms.

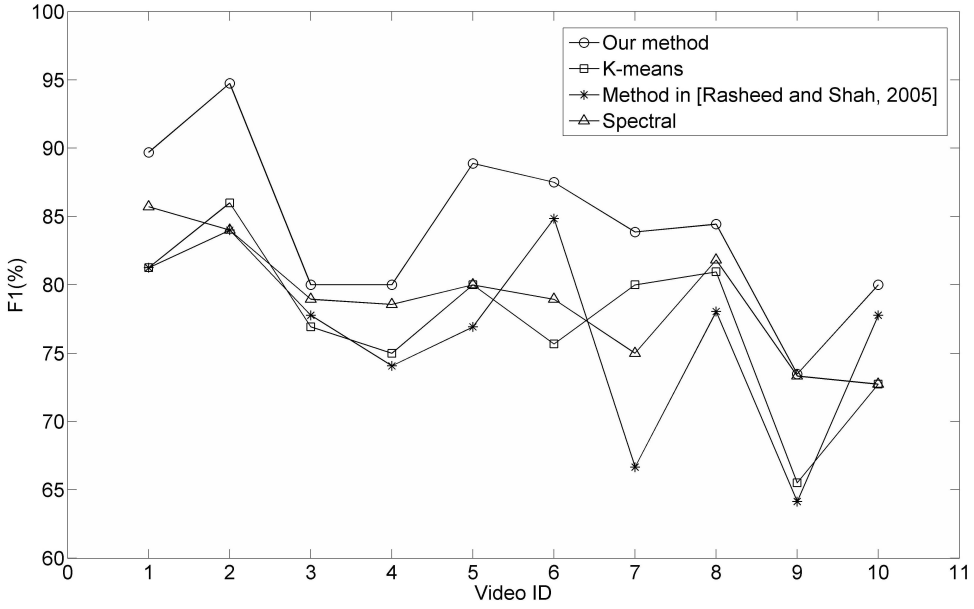


Figure 4.13: Scene detection results (using F_1 measure) comparing four key-frame extraction algorithms.

a typical scene contains a sequence of similar shot labels or a sequence of repetitive label patterns of two or more different groups of shots, when a change in the pattern occurs, we consider that a scene boundary also occurs. To identify such changes, we considered windows of shot sequences which are compared using the "Needleman - Wunsch" sequence

alignment algorithm [55]. Thus, our approach treats time adjacency in a distinct processing phase while existing methods use temporal distance between shots in the definition of the similarity matrix that is subsequently used as input to the clustering procedure. The presented experimental results on several videos indicate that the proposed method accurately detects most scene boundaries, while providing a good trade off between recall and precision.

Chapter 5

Movie Segmentation Using Temporally Weighted Histograms of Visual Words

5.1 Introduction

Movie is the genre of a video that provides the most well-defined structure so far. Typically, a movie is organized into low-level units (shots) and high-level units (scenes and

chapters). Movie segmentation into high-level units is a quite tedious task due to the "semantic gap". Low-level features do not provide useful information about the semantical correlation between shots and usually fail to detect scenes with constantly dynamic content. The scene and chapter boundaries of a movie *are not physical*, as shot boundaries, but correspond to changes in the *semantic content* of the movie.

There are two major problems concerning movie segmentation into scenes and chapters. The first problem concerns the nature of the video content. In dialogue scenes, where the content of video does not change dramatically (changes between cameras recording actors speaking), low-level features such as color histograms are quite efficient in detecting the scene boundaries. This is expected since the shots of a dialogue scene have similar color distribution due to the fixed physical setting (same background). Thus, to detect a scene boundary, we have to seek for a change in the color distribution of shots. However, there are scenes where content changes constantly.

For example, in a scene describing a car chase, there are different shots taken at different places during the course of the car resulting into a constant change in the color distribution of the shots. If we use the same approach described above for a dialogue scene, then the correspondence between scene boundaries and changes in color distributions of shots would lead to a large number of falsely detected scenes. Therefore, color histograms are inefficient to describe scenes with constant changing content. On the other hand, there are some objects or *distinctive points* that are repeated in consecutive shots during the progress of such an event (e.g. the thief and the stolen car). Locally invariant descriptors have been found to provide sufficient description of these interest points and their possible transformations (rotation, scale). These descriptors can be further grouped into a large number of clusters, where each cluster is treated as a *visual word* and represents a specific local pattern shared by all the descriptors in the cluster. By mapping the descriptors into visual words we can adopt the *bag of words* representation, that in the field of image and video processing is known as *bag of visual words* [82, 75]. Thus, each video shot can be represented as a vector containing the frequency of each visual word in the shot. This derived shot representation uses a set of "semantically" richer features, the visual words,

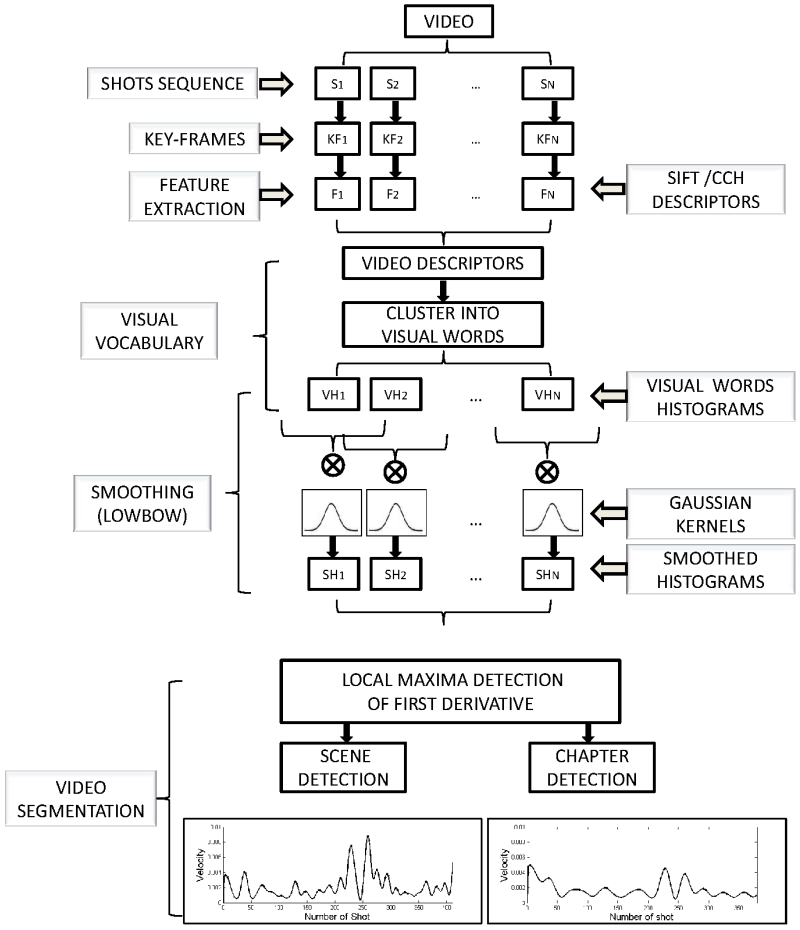


Figure 5.1: Main steps of our high-level movie segmentation method.

helping to correlate two shots and detect possible scene boundaries.

The second problem in movie segmentation concerns the detection of chapters (logical story units). Since a chapter is a group of scenes describing a sub-theme of the movie, it is expected that the color distributions of the corresponding shots will fail to describe the connectivity between them. For example, consider a chapter that comprises of the following two scenes. The first scene describes a thief stealing a car followed by the second scene that describes the chase of the stolen car from the police. Considering that these two events take place in different places, the color distribution of the shots will be very different. On the other hand, features describing the stolen and the police cars could provide useful information about the semantical connection of these two scenes and their corresponding shots.

Several approaches have been proposed for the scene segmentation problem as mentioned in Section 4.1. Most of these approaches calculate shot similarity based on color histograms. Thus, their major disadvantage is the erroneous detection of scenes when the visual content changes continuously. Segmentation of a movie into a more compact representation, such as chapters, has not received much attention yet. In [62], the scene segmentation results of the proposed algorithm are compared with the chapters provided in the DVD compilations of known movies. Considering that chapters are more compact representations, there are many false detections resulting into a very low precision, thus making the specific algorithm inefficient for the chapter detection problem.

In the method we propose herein [13], each video is first segmented into shots. To represent the content of each shot, key-frames are extracted using an improved version of spectral clustering [10, 14]. Then, local invariant descriptors are extracted from all key-frames of the shot. The descriptors of all shots are clustered into a predefined number of visual words (visual vocabulary) and a visual words histogram is constructed for each shot. The histograms of visual words corresponding to each shot are further smoothed temporally by taking into account the histograms of neighboring shots. This is a process that applies local semantic smoothing and enables preserving valuable contextual information. Smoothing is achieved using a Gaussian kernel whose variance can be adjusted to control the amount of smoothing [45]. The final scene and chapter boundaries are determined at the local maxima of the difference of successive smoothed histograms for low and high values of the smoothing parameter, respectively. Thus, by adjusting the smoothing parameter of the gaussian kernel, we can segment each video at different levels, i.e. scenes or chapters. In Fig. 5.1, we summarize the main steps of our approach.

The proposed approach exhibits several novel characteristics such as the use of local invariant descriptors instead of color histograms for movie shots representation. In this way, we provide a semantic representation of a movie that is more robust to visual content variations. Also, the visual words histograms of shots are temporally smoothed (using a gaussian kernel) with respect to neighboring histograms to preserve valuable contextual information. The semantic smoothing process at different time scales facilitates the

efficient segmentation of a movie at different high-levels, such as scenes and chapters. Finally, using the proposed method with different values of smoothing parameter of the gaussian kernel we can tackle both scene and chapter segmentation.

The rest of the Chapter is organized as follows: In Section 5.2, the key-frame extraction method, the feature extraction method and the construction of the visual vocabulary are described. In Section 5.3, the proposed scene and chapter detection algorithm is presented that is based on temporally smoothed shot histograms. In Section 5.4, we present numerical experiments and compare our method with the method proposed in previous chapter and two other methods proposed in [62] and [84]. Finally, in Section 5.5, we conclude our work and provide suggestions for further study.

5.2 Video Representation with Bag of Visual Words

Each video is first segmented into shots and each shot is represented with key-frames extracted using the algorithm presented in Chapter 3. Each frame is represented by a 3D HSV normalized histogram with 8 bins for hue and 4 bins for each of saturation and value, resulting to $8 \times 4 \times 4$ bins. It is worth mentioning that, we use color histograms for the key-frame extraction problem, whereas for the scene and chapter detection problem we propose the use of local invariant descriptions. The latter could also be used for the key-frame extraction problem. However, this is not examined in this Chapter since color histograms sufficiently work well for key-frame extraction as demonstrated in Chapter 3.

5.2.1 Feature Extraction

As already mentioned, color histograms fail to describe connectivity between shots in cases where the visual content constantly changes. The semantic content usually remains the same because objects or interest points are repeated during consecutive shots of a scene. A well-known method to describe objects in images are the invariant local descriptors. In our approach, we consider two kinds of descriptors that have been proposed in bibliography:

the SIFT descriptors proposed in [52] and the CCH descriptors proposed in [35].

SIFT Descriptors

In [52], scale-invariant feature transforms have been proposed that transform image data into scale-invariant coordinates relative to local features. These features are invariant to image scale and rotation. The method described in [52], consists of four major stages:

1. scale-space peak selection;
2. keypoint localization;
3. orientation assignment;
4. keypoint descriptor.

In the first stage, the image is scanned over scale and location to detect features (or interest) points. A Gaussian pyramid is constructed and local peaks (keypoints) in a series of difference-of-Gaussian (DoG) images are detected. In the second stage, unstable keypoints are eliminated. In the third stage, the dominant orientations for each keypoint based on its local image patch are identified. Finally, in the fourth stage, a local image descriptor is built for each keypoint, based upon the image gradient in its local neighborhood. In Fig. 5.2, we present the stages for the keypoint selection.

The standard keypoint descriptor used by SIFT is created by sampling the magnitudes and orientations of the image gradient in the patch around the keypoint, and building smoothed orientation histograms to capture the important aspects of the patch (see Fig. 5.3). A 4×4 array of histograms, each with 8 orientation bins, captures the rough spatial structure of the patch. This 128-element vector ($4 \times 4 \times 8$) is then normalized to unit length and thresholded to remove elements with small values. SIFT descriptors constitute a very popular approach successfully employed for several computer vision problems [60, 26].

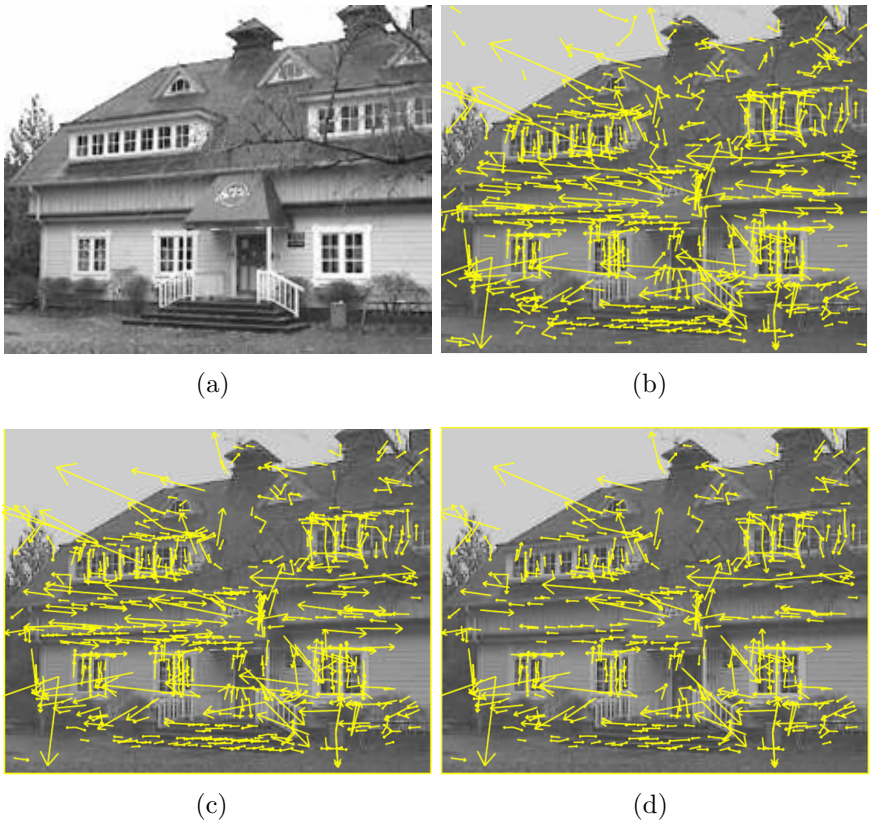


Figure 5.2: Stages of keypoint selection (Image taken from [52]). (a) The original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

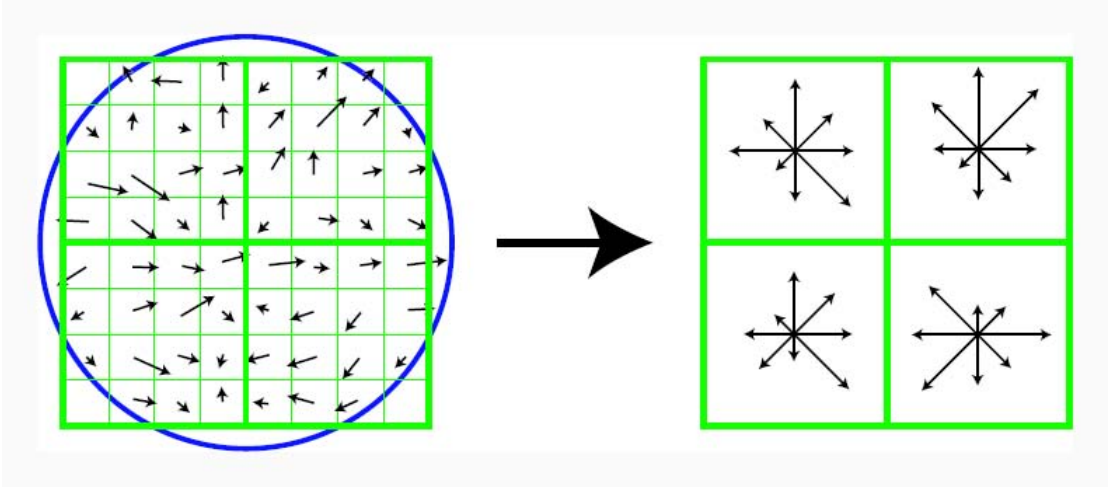


Figure 5.3: Computation of a keypoint descriptor (Image taken from [52]).

CCH Descriptors

A similar local invariant descriptor, called contrast context histogram (CCH) has been proposed in [35] and successfully employed for the shot detection problem [36]. It represents the contrast distributions of a local region around an interest point and serves as a local descriptor for this region. Given an image I, gaussian kernels are applied to smooth this image. Then, a multi-scale Laplacian pyramid is constructed and salient points are extracted by detecting Harris corners [33]. A region R around each salient point p_c and the contrast of a point p in this area is given from the following equation:

$$C(p) = I(p) - I(p_c). \tag{5.1}$$

Region R is defined in a quantized log-polar coordinate system (r, θ), where $r_i = 0, \ldots, r$, $r = \lfloor log(\sqrt{2n^2}) \rfloor$ and $\theta_j = \frac{2\pi}{l} m,\ m = 0, \ldots, l-1$. Parameters r, l define the distance and orientation quantization, respectively. For each sub-region $R_{ij} = (r_i \theta_j)$, a positive and a negative bin of the contrast values are computed. More specifically, given a salient point p_c and a sub-region R_{ij}, the positive and the negative histogram bins are defined from equations (5.2) and (5.3) respectively:

$$H_{R_{ij}^+}(p_c) = \frac{\sum\{C(p) | p \in R_{ij} \ and \ C(p) \geq 0\}}{\#R_{ij}^+}, \tag{5.2}$$

$$H_{R_{ij}^-}(p_c) = \frac{\sum\{C(p) | p \in R_{ij} \ and \ C(p) < 0\}}{\#R_{ij}^-}, \tag{5.3}$$

where $\#R_{ij}^+$ and $\#R_{ij}^-$ define the number of positive and negative positive contrast values in R_{ij}. By composing the contrast histograms of all the sub-regions into a single vector, the CCH descriptor of p_c with respect to its local region R is defined as follows:

$$CCHp_c = (H_{R_{00}^+}, H_{R_{00}^-}, \ldots, H_{R_{rl}^+}, H_{R_{rl}^-}). \tag{5.4}$$

In our approach, we have used $r = 3$ and $l = 8$, as proposed in [35], resulting to

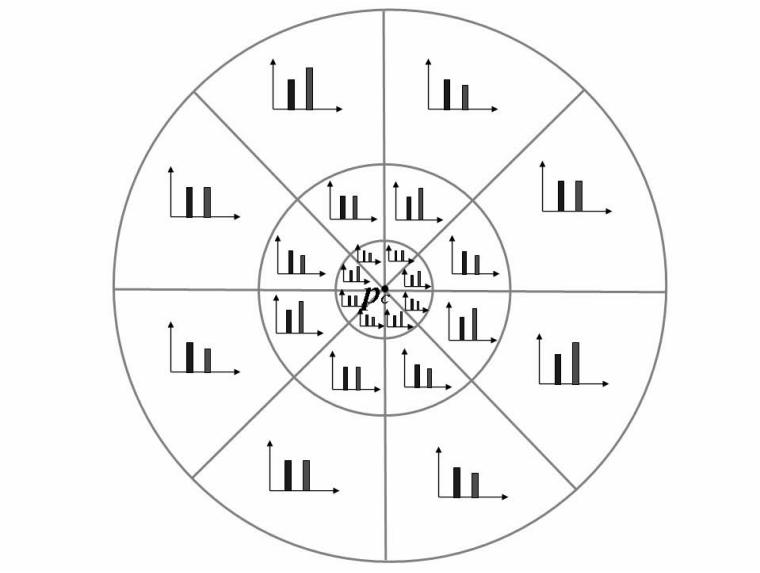


Figure 5.4: Contrast context histogram of a salient corner p_c under the log-polar coordinate system (Image taken from [35]).

$2 \times 4 \times 8 = 64$ dimensions for each CCH descriptor. In Fig. 5.4 we present the contrast context histogram of a salient corner p_c under the log-polar coordinate system.

5.2.2 Bag of Visual Words

For each shot a different number of descriptors is computed that describe certain objects or interest points in the shot. More specifically, suppose we are given a shot s_t and its corresponding set of n key-frames $KF = \{kf_1, \ldots, kf_n\}$. For each key-frame $kf_i,\ i = 1, \ldots, n$, a set of descriptors D_{kf_i} is extracted (SIFT or CCH) using the algorithms presented in [52] and [35], respectively. Then, all the sets of descriptors are concatenated to describe the whole shot

$$D_{s_t} = D_{kf_1} \bigcup \ldots \bigcup D_{kf_n}. \tag{5.5}$$

To speed up the implementation, the set D_{s_t} of the descriptors of each shot s_t is summarized to obtain a more compact representation. More specifically, the set D_{s_t} is clustered into M groups (set to 20 in our experiments) using the fast global k-means algorithm [49]. Thus, each shot is finally represented by the *centroids* of the M groups denoted as $D_{s_t}^M$.

To extract *visual words* from the descriptors, the set of representative centroids de-

scriptors for all N video shots $D_S = D_{s_1}^M \bigcup D_{s_2}^M \bigcup \ldots \bigcup D_{s_N}^M$ is clustered into k groups $\{C_1, C_2, \ldots, C_k\}$ using the k-means algorithm, where k denotes the total visual words vocabulary size. To construct the visual word histogram (bag of visual words) for shot s_t, each element of the set of descriptors D_{s_t} is assigned to one of the k visual words (clusters), thus resulting into a vector containing the frequency of each visual word in the shot. Thus, given that shot s_t has D descriptors $d_{t_1}, \ldots, d_{t_D}$, the visual word histogram VH_t for this shot is defined as:

$$VH_t(l) = \frac{\#\{d_{t_j} \in C_l,\ j = 1, \ldots, D\}}{D},\ l = 1, \ldots, k. \tag{5.6}$$

5.3 Scene and Chapter Detection Using Temporally Weighted Histograms of Visual Words

So far, each video shot s_t is described by a visual word histogram VH_t that corresponds to the probability that a specific *visual word* of the video is included in the specific shot. Next, the similarity between successive shots must be defined in order to detect the scene and chapter boundaries. This will be based on a technique that has been previously proposed for text documents.

5.3.1 Similarities Between Video and Text Documents

Video and text documents exhibit many analogies. Videos can be segmented into shots, scenes and logical story units (DVD chapters). In a similar way, text documents can be segmented into words, paragraphs and logical story units (book chapters). A key aspect of our method is that, based on an idea from text segmentation, the histograms of visual words corresponding to each shot are further smoothed temporally by taking into account the histograms of neighboring shots.

In [45], the Locally Weighted Bag of Words (Lowbow) framework has been proposed for text document representation and segmentation. The main idea of this approach is

to represent a text document by describing several locations in its word sequence using histograms, instead of using a typical bag of words histogram that models the word distribution in that document. To construct a local histogram representation, a *smoothing kernel* is utilized to smooth the semantics temporally around a given location in the original word sequence. Initially each location t in a word sequence is described using a *trivial* histogram H_t whose probability mass is concentrated only at the bin that corresponds to the word that occurs at that location in text. Formally, the Lowbow representation for a certain location t in a text sequence is computed by

$$L_t = \sum_{n=-\infty}^{\infty} H_n \cdot K_\sigma(t-n), \tag{5.7}$$

where K_σ is a normalized discretized gaussian kernel with zero mean and standard deviation σ. In this way, the presence of a word at a certain location in the document also contributes to a neighboring location but with *discounted contribution* depending on the temporal distance between the two locations. This representation captures the sequential content at a certain resolution determined by a given local smoothing parameter (σ).

We have adopted a similar approach for video documents. Video shots can be considered analogous to the words of a document that compose a paragraph (scene) that further compose a book chapter (movie chapter) that describes a specific theme. Initially, each location in the video shot sequence can be described using a visual word histogram computed as described in Section 5.2.2. Next, the temporal semantic smoothing takes place to compute the final shot representations, the temporally smoothed visual word histograms. It must be noted that video shots are complicated units of visual information, contrary to textual words. For this reason, we introduce the visual words histograms to initially summarize the semantics of the raw features extracted from video shots, while in the text case trivial histograms are used as described previously. As a result, in the case of video data, the temporal semantic smoothing can be considered to be at a higher semantic level than that of the original method for texts.

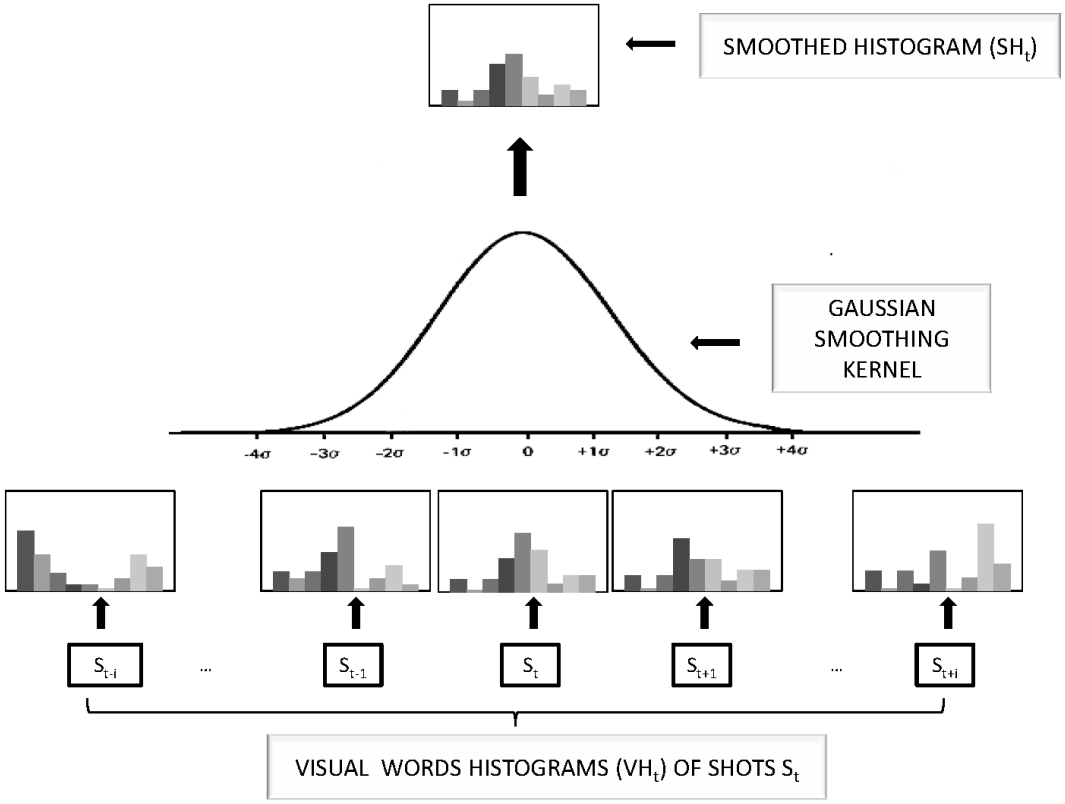


Figure 5.5: Temporal smoothing of visual word histograms representing the video shots, using a gaussian smoothing kernel.

5.3.2 Scene and Chapter Segmentation

In a similar way to Lowbow framework described in [45], a local smoothing kernel is used to smooth temporally the visual word histogram of a shot with respect to the histograms of neighboring shots. The smoothed histogram SH_t of a visual word histogram VH_t of a shot s_t (where t denotes the time index of the shot) is given from the following equation:

$$SH_t = \sum_{n=-\infty}^{\infty} VH_n \cdot K_\sigma(t-n), \tag{5.8}$$

where K_σ is a normalized discretized gaussian kernel with zero mean and standard deviation σ. In Fig. 5.5, a visual representation of the smoothing process is given. First, the visual words histogram (VH_t) for each shot is computed (bottom level). Then, the visual words histogram of shot s_t is *smoothed temporally* with the neighbor visual word

histograms using a gaussian kernel, resulting to the smoothed histogram (SH_t) of the shot (upper level). The number of neighboring histograms that contribute to smoothing is defined by the value of the smoothing parameter σ. By adjusting the value of σ, we can preserve contextual information at different time scales. A low value of σ results in small scale smoothing and can preserve contextual information within scenes, whereas a higher value of σ results in large scale smoothing and can preserve contextual information within chapters.

Our model associates each shot $s_t,\ t = 1, \ldots, N$ with a smoothed histogram SH_t, which can be considered to be a point in the multinomial simplex $\mathbb{P}_{k-1}$, where k is the vocabulary size. The multinomial simplex $\mathbb{P}_{k-1}$ for $k \geq 1$ is the k-dimensional subset of $\mathbb{R}^k$ of all histograms over k objects:

$$\mathbb{P}_{k-1} = \{\theta \in \mathbb{R}^k : \forall i\ \theta_i \geq 0,\ \sum_{j=1}^{k} \theta_j = 1\}. \tag{5.9}$$

The sequence $SH_t,\ t = 1, \ldots, N$ of smoothed histograms represents the video shot sequence with a curve in $\mathbb{P}_{k-1}$ called Temporally Smoothed Visual Words Histograms or TSVWH curve. Fig. 5.6 illustrates an example of a video shot sequence, whose TSVWH curve representation is projected from $\mathbb{P}_{k-1}$ to $\mathbb{P}_2$ using Principal Component Analysis (PCA). This semantic representation of a video shot sequence could be extended in many other applications, such as video retrieval and surveillance since it provides useful information about the semantic content of the video sequence and its progress in time. The boundaries between different video segments separate video parts containing different visual words distributions. In the context of the TSVWH curve produced by the smoothed histograms, a boundary point would correspond to sudden shifts in the curve location. Due to the continuity of TSVWH curve, such sudden shifts may be discovered by considering the local maxima of the Euclidean distance between successive smoothed histograms:

$$V_t = \sqrt{\sum_{h=1}^{k} (SH_t(h) - SH_{t+1}(h))^2},\ t = 1, \ldots, N-1, \tag{5.10}$$

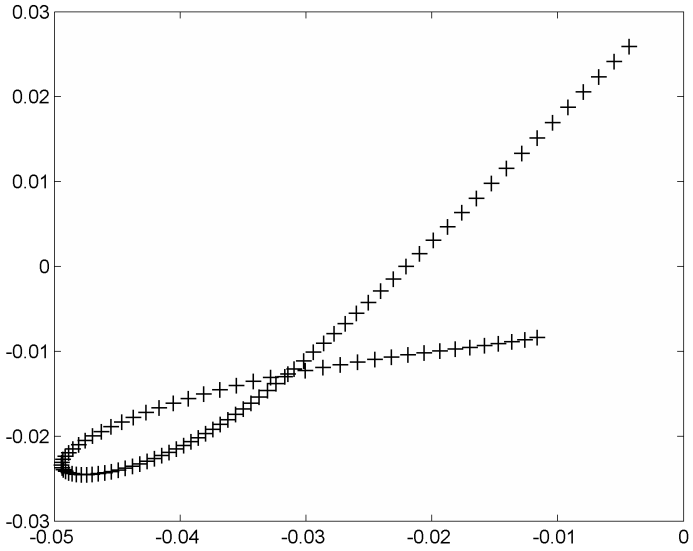


Figure 5.6: 2D embedding (using PCA) of the TSVWH curve representing a video shot sequence.

where k denotes the number of visual words (histogram bins).

In Fig. 5.7 and Fig. 5.8, we present the difference values between successive smoothed histograms for an example shot sequence using smoothing parameter $\sigma = 8$ (for scene detection) and $\sigma = 16$ (for chapter detection), respectively. In both cases, circles correspond to detected boundaries and stars correspond to true boundaries. It can be observed that using a high value of the smoothing parameter σ results into a smoother curve whose maxima are the boundaries of more compact representations (chapters). Therefore, using the same method with different values of σ we can tackle both scene and chapter segmentation.

5.4 Numerical Experiments

In this Section, we present numerical experiments for the scene and chapter detection problems, and we also compare our method with the method proposed in Chapter 4 and two other approaches: [62] and [84].

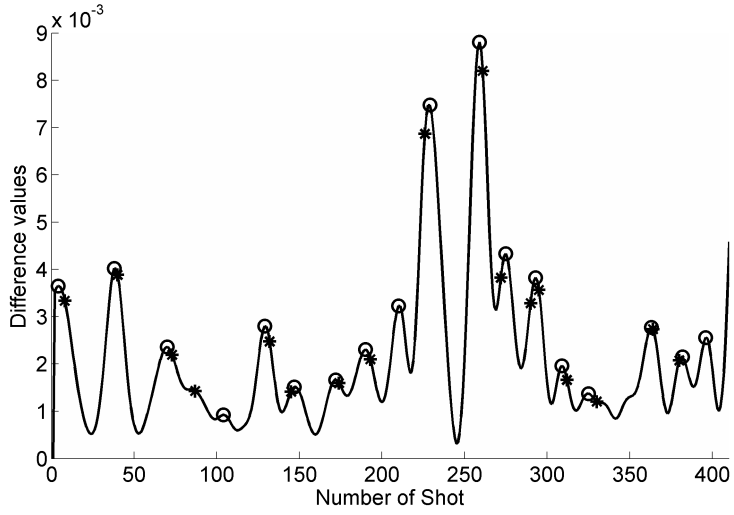


Figure 5.7: Difference values of the smoothed histograms using $\sigma = 8$ (scene detection).

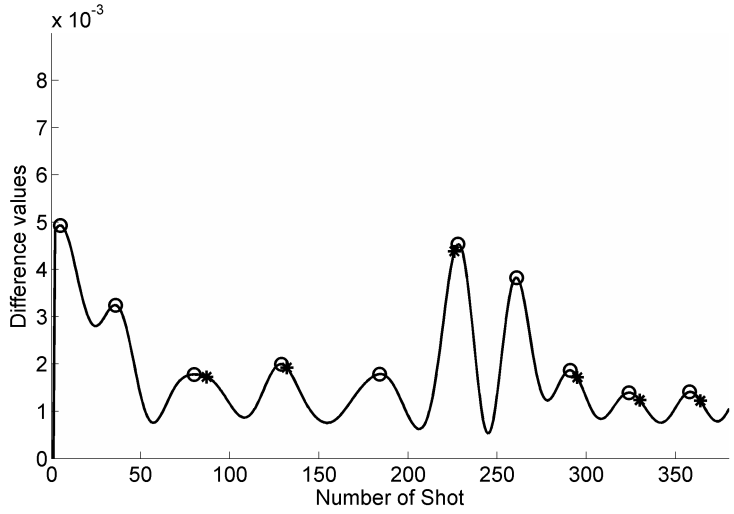


Figure 5.8: Difference values of the smoothed histograms using $\sigma = 16$ (chapter detection).

5.4.1 Data and Performance Criteria

To evaluate the performance of our detection algorithm we use five movies that belong to different genres. The characteristics of these movies are shown in Table 5.1.

To evaluate the performance of our method we have computed the commonly used criteria of equations (2.16), (2.17) and (2.18), where N_c stands for the number of correctly detected scene or chapter boundaries (true positive), N_m for the number of missed ones (false positive) and N_f the number of false detections (false negative). Two human ob-

Table 5.1: Movies characteristics.

Movie	ID	Duration(min)	Shots	Scenes	Chapters	Genre
A Beautiful Mind	M1	36	421	18	7	Biography | Drama
Sex and the City	M2	70	1217	45	19	Comedy | Romance
Gone in 60 seconds	M3	80	1788	74	23	Action | Crime | Thriller
Goldeneye	M4	74	1218	46	20	Action | Adventure
Top Gun	M5	74	1113	48	16	Action | Romance

servers identified the scene boundaries and the ground truth was defined as the cases for which there was agreement between the observers. The boundaries of the movie chapters were obtained from the menu of the corresponding DVD compilations.

5.4.2 Results

In Fig. 5.9, the average performance of our algorithm on all movies is presented, using SIFT descriptors and a visual vocabulary of 500 words, varying the smoothing parameter σ from 2 to 32. It can be observed that for $\sigma = 8$ the algorithm provides the best performance for the scene detection problem, whereas for $\sigma = 16$ the algorithm provides the best performance for the chapter detection problem. In all our experiments, we use $\sigma = 8$ and $\sigma = 16$ for the scene and chapter detection problems respectively. In Fig. 5.10, the average F_1 values for all movies are presented using SIFT and CCH descriptors, varying the vocabulary size (number of visual words) from 10 to 500. It can be observed that as the number of visual words increases, the performance of our algorithm improves. However, it is computational expensive to produce a good partition of the shot descriptors into a large number of visual words, thus we have only experimented with less or equal to 500 visual words. In Tables 5.2 and 5.3 (first two methods (SIFT, CCH)) we present the recall, precision and F_1 values using 500 visual words, $\sigma = 8$ for scene detection, $\sigma = 16$ for chapter detection, for both SIFT and CCH descriptors.

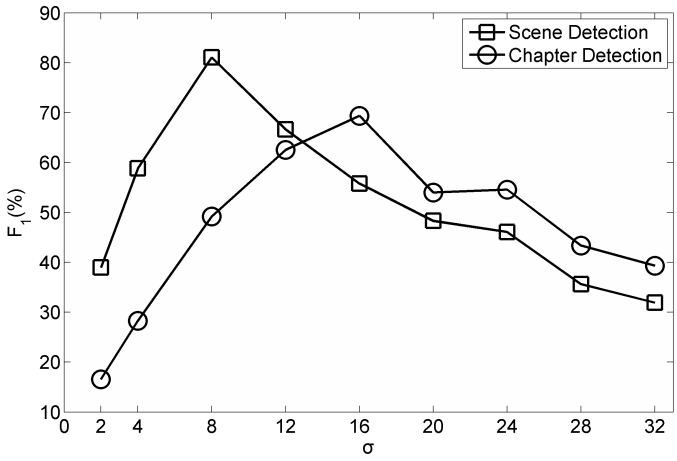


Figure 5.9: Average performance results (on all movies) for different values of the smoothing parameter σ for the scene and chapter detection problems, using SIFT descriptors and a vocabulary of 500 visual words.

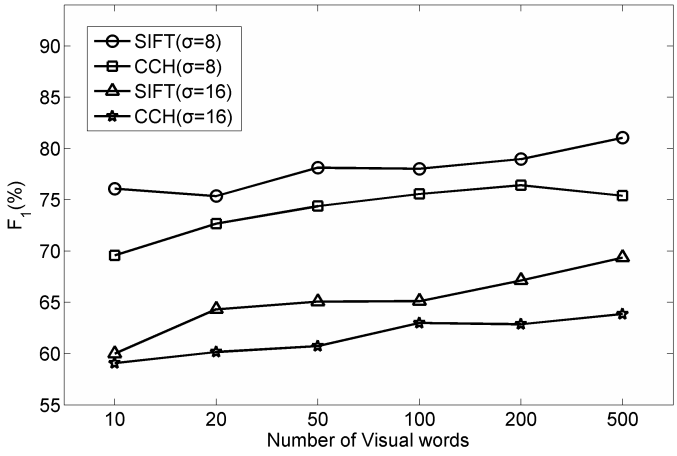


Figure 5.10: Average performance results (on all movies) for different number of visual words, using SIFT and CCH descriptors and $\sigma = 8$ and $\sigma = 16$ for the scene and chapter detection problems, respectively.

5.4.3 Comparison

To compare the effectiveness of our approach, we have also tested three other methods. In the method proposed in Chapter 4, shots are clustered into groups based on their visual similarity using an improved spectral clustering algorithm and a label is assigned to each shot according to the group that it belongs to. Then a sequence alignment algorithm is applied to detect when the pattern of shot changes providing the final segmentation result.

Table 5.2: Comparative results using Recall(R), Precision(P) and F_1 measures for the scene detection problem for movies M_1-M_5, (SEQAL - [14], NCUT - [62], GRAPH - [84]).

		SIFT	CCH	SEQAL	NCUT	GRAPH	HSV
M1	R(%)	88.89	83.33	77.78	83.33	77.78	72.22
	P(%)	88.89	88.24	82.35	60.00	60.87	72.22
	F_1(%)	88.89	85.71	80.00	69.77	68.29	72.22
M2	R(%)	91.11	80.00	80.00	71.71	64.44	73.33
	P(%)	83.67	73.47	67.92	46.24	56.86	63.46
	F_1(%)	87.23	76.60	73.47	56.22	60.42	68.04
M3	R(%)	82.43	77.03	79.73	74.32	62.16	71.62
	P(%)	72.62	69.51	68.60	55.56	54.12	67.09
	F_1(%)	77.22	73.08	73.75	63.58	57.86	69.28
M4	R(%)	88.89	77.78	88.89	80.00	68.89	68.89
	P(%)	68.97	63.94	64.52	53.73	48.44	58.49
	F_1(%)	77.67	70.00	74.77	64.29	56.88	63.27
M5	R(%)	75.00	70.83	75.00	72.92	70.83	52.08
	P(%)	73.47	72.34	70.59	53.85	45.33	58.14
	F_1(%)	74.23	71.58	72.73	61.95	55.28	54.95

In Table 5.2 (third method (SEQAL)) we present the recall, precision and F_1 values of the experiments with this method.

The second method we implemented has been proposed in [62]. It computes both color and motion similarity between shots and the final similarity value is weighted by a decreasing function of the temporal distance between shots. The shot similarity matrix defines a weighted undirected graph where each node represents a shot and the edges are the elements of the matrix. To partition the video into scenes, an iterative application of Normalized cuts method [67] is used that divides the graph into subgraphs. The $Ncut$ algorithm is applied recursively as long as the $Ncut$ value is below some stopping threshold T. We have implemented and tested this method using the same movies set for different values of the threshold parameter T. In our comparisons, we found distinct values for each video that provided the best performance. The recall, precision and the F_1 values of the experiments are presented in Table 5.2 (fourth method (NCUT)).

The third method we implemented has been proposed in [84]. It clusters shots into groups taking into account the visual characteristics and temporal dynamics of video.

Table 5.3: Comparative results using Recall(R), Precision(P) and F_1 measures for the chapter detection problem for movies M_1-M_5, (SEQAL - [14], NCUT - [62], GRAPH - [84]).

		SIFT	CCH	SEQAL	NCUT	GRAPH	HSV
M1	R(%)	100.00	85.71	42.86	71.43	71.43	71.43
	P(%)	63.64	60.00	33.33	41.67	31.25	50.00
	F_1(%)	77.78	70.59	37.50	52.63	43.49	58.82
M2	R(%)	89.47	84.21	68.42	57.89	63.18	73.68
	P(%)	68.00	59.26	41.94	45.83	37.50	50.00
	F_1(%)	77.27	69.57	52.00	41.16	47.06	59.57
M3	R(%)	78.26	82.61	65.22	60.87	60.87	65.22
	P(%)	45.00	43.18	30.62	40.00	40.00	30.61
	F_1(%)	57.14	56.72	41.67	48.28	48.28	41.67
M4	R(%)	80.00	75.00	95.00	40.00	45.00	60.00
	P(%)	61.54	53.57	30.65	42.10	33.33	42.86
	F_1(%)	69.57	62.50	46.34	41.02	38.29	50.00
M5	R(%)	87.50	75.00	68.75	56.25	81.25	56.25
	P(%)	51.85	50.00	32.33	36.00	26.00	39.13
	F_1(%)	65.12	60.00	44.00	43.90	39.39	46.15

Then, a *scene transition graph*, which is a graphical representation of the video, is constructed. The nodes of this graph represent the shots and the edges the transitions between the shots. To find the scenes, this graph is partitioned into connected subgraphs. The above algorithm depends on two parameters. The first one is the parameter δ which defines the minimum separation between any two resulting clusters and controls the final number of clusters. The second parameter is T that defines two shots to belong in different clusters if they are not close to each other. After the initial segmentation, the segmented scenes are refined by adjusting the threshold parameter T to reflect the duration of scenes. Determination of optimal values for the parameters δ and T is a tedious task. To test the performance of this algorithm we executed multiple runs using different values for the parameters δ and T. In our comparisons we used distinct values for each video that report the best performance. The recall, precision and the F_1 values of the experiments are presented in Table 5.2 (fifth method (GRAPH)).

To demonstrate the efficiency of using local invariant descriptors, instead of smooth-

ing the visual words histograms of shots, we smoothed the corresponding HSV color histograms. In Tables 5.2 and 5.3 (sixth method (HSV)), we present the recall, precision and the F_1 values of these experiments. It is clear that local invariant descriptors outperform color histograms. However, it is worth mentioning that the proposed technique that uses temporal smoothing of histograms with a gaussian kernel, provides better results than the approaches proposed in [62] and [84], even when color histograms are smoothed.

It must be noted that the three other algorithms SEQAL, NCUT and GRAPH have been proposed for scene detection. However, fewer boundaries (that correspond to more compact representations, such as chapter boundaries) can be detected if the thresholds that these algorithms employ are modified. Thus, the thresholds that these methods employ are adjusted to provide the best performance considering the chapter detection problem. More specifically, in the SEQAL method scene boundaries are identified from the local minima of a sequence alignment scoring function that are under a predefined threshold. To provide the best performance considering the chapter detection problem, we increase this threshold. In this way, fewer boundaries are detected that correspond to chapter boundaries. The recall, precision and the F_1 values of the experiments are presented in Table 5.3 (third method (SEQAL)). Similarly, in NCUT method the stopping threshold T controls the number of final scene boundaries. By decreasing this threshold we identified the threshold value providing the best performance for chapter detection and fewer scenes are detected corresponding to more compact representations (chapters). The recall, precision and the F_1 values of the experiments are presented in Table 5.3 (fourth method (NCUT). Finally, in GRAPH method the threshold parameter δ controls the final number of clusters. To find the best performance considering the chapter detection problem, we must increase this threshold. In this way, fewer number of clusters are obtained and fewer boundaries corresponding to chapters are detected. The recall, precision and the F_1 values corresponding to the best value of the threshold parameter are presented in Table 5.3 (fifth method (GRAPH)). It is clear that the use of local invariant descriptors in combination with the temporal smoothing of visual words histograms provides the best results outperforming the other methods both for scene and chapter detection. Also,

SIFT descriptors provide better performance compared to CCH descriptors.

5.5 Conclusions

In this Chapter, a new high-level movie segmentation method has been proposed based on the temporal smoothing of visual word histograms of video shots. For each movie shot a number of key-frames is extracted and local invariant descriptors are computed for each key-frame. All the descriptors are clustered to form the visual words vocabulary and for each shot a visual word histogram is computed. Next, to preserve contextual information, the Lowbow framework proposed for text segmentation is adopted and the histograms of visual words are smoothed temporally by taking into account the histograms of neighboring shots. A notable characteristic of the method is that by adjusting the smoothing parameter of the gaussian kernel we can detect both scene and chapter boundaries of each movie, determined at the local maxima of the difference of successive smoothed histograms. The presented experimental results on five movies indicate that the proposed method outperforms other methods and accurately detects most scene and chapter boundaries, while providing a good trade off between recall and precision. A direction for future work, is to incorporate the proposed movie representation with temporally smoothed shot histograms in other video-based applications, such as video retrieval and surveillance.

Chapter 6

Video Rushes Summarization

6.1 Introduction

In this Chapter, we present a system for *video rushes summarization* [12]. A video summary is a condensed version of the initial video where judgements about the video content can be made in less time and effort than using the initial video [59]. Video rushes are

unedited video footage and contain many repetitive information, since the same scene is taken many times until the desired result is produced. Moreover, since it is unedited video, it contains many "junk" frames such as *colorbars*, monochrome frames and frames that contain *clapboards*. Colorbars are an artificial electronic signal generated by the camera or by post production equipment. They are recorded at the head of a videotape to provide a consistent reference in post production. They are also used for matching the output of two cameras in a multi-camera shoot and to set up a video monitor. A clapboard is a device used to assist in the synchronizing of picture and sound. Additionally, the clapboard is used to designate and mark particular scenes and takes recorded during a production.

Three issues should be considered during the rushes summarization process. The first one is that useless frames such as colorbars, monochrome frames and frames that contain clapboards should be removed from the video. The second issue is that similar segments generated from multiple takes of the same scene should be removed keeping only one representative segment. The third issue is the efficient representation of the content of each of the selected representative shots and the creation of the final video summary.

In [16], a baseline video summarization system is described that presents the entire video in 25×, 50× and 100× speed, while removing junk frame using color and SIFT features. In [76], dynamic generation of binary trees is used, allowing realtime, on-the-fly summaries to be generated. In [21], shot similarity is computed based on color histograms of regions in so-called characteristic frames, and similar shots are then stacked. Then, an adaptive acceleration technique is used, changing playback speed based on the (visual) similarity of adjacent frames in the generated summary.

In the method we propose herein, each video is initially segmented into shots by comparing the normalized histograms of adjacent video frames. Then, for key-frame extraction we use the method proposed in Chapter 3 where an enhanced spectral clustering algorithm is employed that both estimates the number of clusters and uses the fast global k-means algorithm in the clustering stage after the eigenvector computation of the similarity matrix. Next, useless frames such as colorbars and monochrome frames are removed

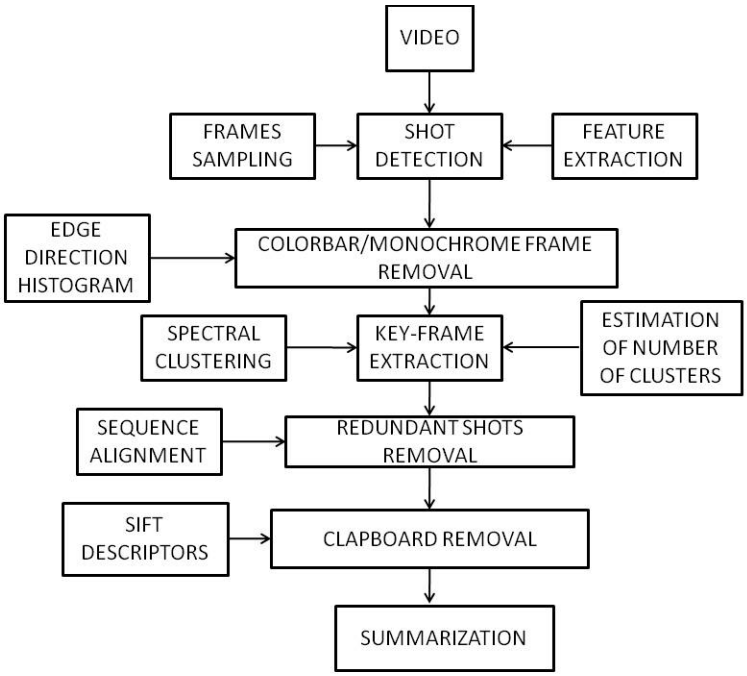


Figure 6.1: The main steps of our video rushes summarization method.

by checking their *edge direction* histogram. Rushes video contain redundant information, since the same scene is taken many times until the desired result is produced. To find similar segments (shots) in the rushes video, the key-frames of shots are compared using a sequence alignment algorithm. Those similar shots that describe the same scene are removed and only one of them is kept to contribute to the final video summary. Moreover, key-frames that contain clapboards should be removed from the final representative shots. Comparing the SIFT descriptors of the key-frames of each shot with the SIFT descriptors of a database of clapboards, we are able to determine if a key frame contains a clapboard and remove it. Finally, to produce the video summary with duration less than a percentage of the duration of the original video, a number of frames around each key frame of the selected shots are considered to contribute to the final video summary. In Fig. 6.1, we summarize the main steps of our approach and the algorithms employed in these steps.

The rest of the Chapter is organized as follows: In Section 6.2, we present the representation of the video by its shots key-frames. In Section 6.3, the removal of junk

frames is presented. In Section 6.4, the procedure for removing redundant information is described. In Section 6.5, the process of removing frames with clapboards is presented and in Section 6.6 the final summarization process is described. Finally, in Section 6.7, we present numerical experiments from the TRECVID Rushes Summarization task 2008, which indicate that our system exhibited good performance.

6.2 Video Representation

In this Section, we describe the shot detection and key-frame extraction methods. The first level of video processing is the extraction of features for each frame and the segmentation of the video into shots.

6.2.1 Feature Extraction and Shot Detection

Each video is sampled uniformly keeping only five frames per second. Then, for each frame an HSV normalized histogram is computed, with 8 bins for hue and 4 bins for each of saturation and value, resulting to $8 \times 4 \times 4$ bins. Since rushes are unedited video footage, they are always stand alone shots and do not contain transitions of any type. Thus, a simple shot detection algorithm is a very fast and efficient method. To detect shot boundaries we calculate the sum of the bin-wise differences of adjacent frames and compare them to a threshold. We use a variation of x^2 to compare the histograms of two frames in order to enhance the difference between the two histograms. Finally, the difference between two images I_i, I_j based on their color histograms H_i, H_j is computed:

$$d(I_i, I_j) = \frac{1}{2} \sum_{k=1}^{128} \frac{(H_i(k) - H_j(k))^2}{H_i(k) + H_j(k)} \ , \tag{6.1}$$

where k denotes the bin index. A shot boundary is defined at frame I_i if $d(I_i, I_j)$ is greater than a threshold T_{sh}, which in our experiments was set to 0.15. Shots shorter than 1 second were removed.

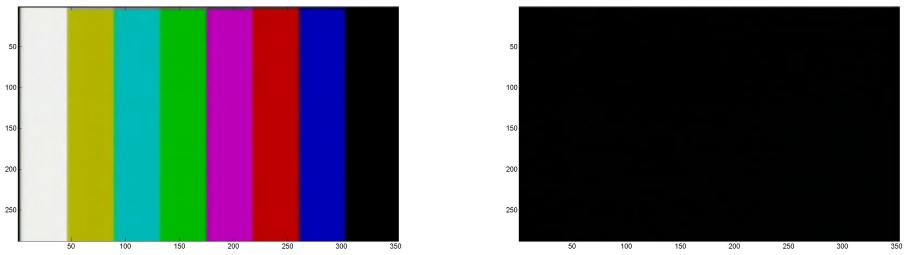
6.2.2 Key-Frame Extraction

To speed up the summarization process each shot must be represented by unique frames that will capture the whole content of the shot. In this way, to compare two shots, we don't use all the frames of each shot but a small number of key-frames that provide a sensible representation of the shot content. To perform key-frame extraction we use the method presented in Chapter 3. The video frames of each shot are clustered into groups using an enhanced spectral clustering algorithm [56] that employs the very efficient global k-means algorithm [49] in the clustering stage after the eigenvector computation. The medoids of the obtained groups are selected as the key-frames of the shot.

A key aspect of the summarization process is the number of key-frames that are necessary to capture the video content. In this Chapter, the number of key-frames is estimated using the method proposed in Section 3.3.

6.3 Useless Frames Detection

Video rushes contain many useless frames such as colorbars and monochrome frames (see Fig. 6.2), which are not necessary for the final summarization and should be removed. The shot detection algorithm usually isolates colorbars or monochrome frames into single shots, thus to speed up the implementation process the first key-frame of each shot is checked and if it is defined as useless frame, the corresponding shot is removed from the summarization process. To check whether a key-frame is useless or not we calculate its edge direction histogram [53]. The edge direction histogram captures the spatial distribution of edges. The key-frame is first sub-divided into 16 sub-images, and local edge histograms for each of these sub-images is computed. Edges are grouped into five categories: vertical, horizontal, 45 diagonal, 135 diagonal, and isotropic (nonorientation specific). Thus, each local histogram has five bins corresponding to the above five categories resulting in a 80-bin histogram for the whole frame. To compute the edge histograms, each of the 16 sub-images is further subdivided into image blocks (see Fig. 6.3). A simple edge detector



(a) Colorbar (b) Monochrome

Figure 6.2: Junk frames in rushes videos.

is then applied to each macro-block, which is treated as a 2×2 pixel image. The edge-detector operators include four directional selective detectors and one isotropic operator (Fig. 6.4). Thus, for an image block, we can compute five edge strengths, one for each of the five filters. Those image blocks whose edge strengths exceed a certain minimum threshold are used in computing the histogram.

In Fig. 6.5 we provide the edge direction histograms for (a) a colorbar and (b) a normal frame. The edge direction histogram of a colorbar produces peaks in vertical and horizontal bins whereas the other bins are close to zero. The bins of the edge direction histogram of a monochrome frame are all close to zero. Thus a colorbar or monochrome frame is detected if the difference between the sum of all bins of the edge histogram and the sum of the vertical and horizontal bins is lower that a threshold T_{edh}:

$$\sum_{k=1}^{128} E_i(k) - \sum_{m=0}^{15} E_i(5m+1) - \sum_{m=0}^{15} E_i(5m+2) < T_{edh} \ , \tag{6.2}$$

where E_i is the edge direction histogram of frame I_i and $E_i(5m+1)$, $E_i(5m+2)$, $m = 0, \ldots, 15$ are the vertical and horizontal bins of the histogram respectively. In our experiments T_{edh} was set to 10.

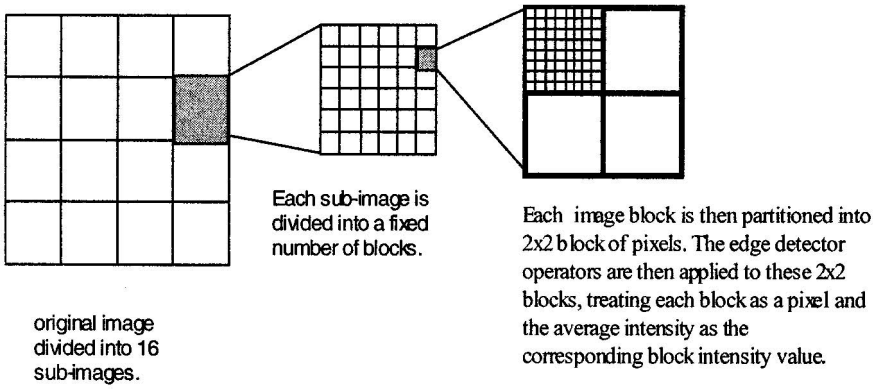


Figure 6.3: Edge direction histogram computation (Image taken from [53]).

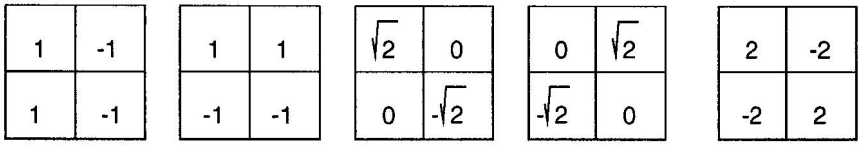
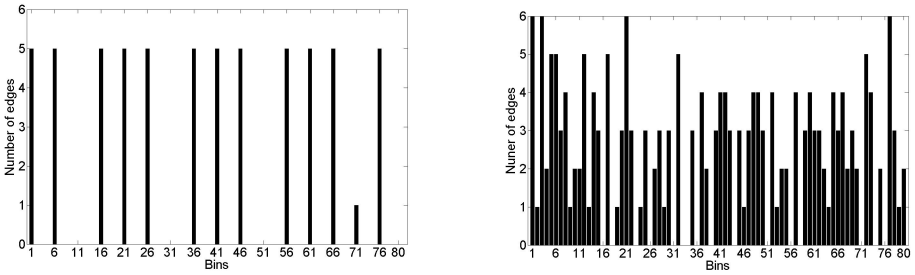


Figure 6.4: Edge detection filters (Image taken from [53]).



(a) Colorbar (b) Normal frame

Figure 6.5: Edge direction histograms.

6.4 Redundant Information Retrieval

Rushes often contain repetitive information, since the same scene is usually taken many times until the desired result is produced. Our goal is to detect similar segments which in our case are shots and keep only one representative for each group of similar shots that will be further analyzed to contribute to the final summary.

6.4.1 Visual Shot Similarity Metric

Once we have removed the shots that correspond to colorbars or monochrome frames we need to suggest a proper visual shot similarity metric. Suppose we are given two shots S_i and S_j and $KF_i = \{KF_i^1,\ KF_i^2, \ldots,\ KF_i^m\}$, $KF_j = \{KF_j^1,\ KF_j^2, \ldots,\ KF_j^n\}$ their corresponding key-frame sets. An $m \times n$ similarity matrix SM is constructed with elements:

$$SM(m,n) = VisSim(KF_i^m, KF_j^n)\ , \tag{6.3}$$

where $VisSim$ is the visual similarity between two frames I_i and I_j given by the following equation:

$$VisSim(I_i, I_j) = 1 - d(I_i, I_j)\ , \tag{6.4}$$

with $d(I_i, I_j)$ defined in equation (6.1) and $VisSim \in [0,1]$.

Taken into consideration that in rushes two shots that describe the same scene are similar, we expect that their key frames will follow the same order. Thus, it is expected that either a segment of one shot or the whole shot will also appear in the other shot. To find similar segments in two shots we use a sequence alignment algorithm between the sets of their key-frames. In this way, a key-frame is "matched" with the most similar (visually) key-frame of the the other set of key-frames, while also taking into consideration the temporal order of key-frames. Suppose that the first shot describes the following time ordered events $E_1, E_2, E_3, E_4, E_5, E_6$ and the second shot describes events E_2, E_3, E_5, E_6. An optimal alignment of the two shots is presented in Fig. 6.6.

$$\begin{array}{lcl} Seq_1 & : & E_1E_2E_3E_4E_5E_6 \\ Seq_2 & : & E_2E_3E_5E_6 \end{array}$$

Seq_1	E_1	E_2	E_3	E_4	E_5	E_6
Seq_2	-	E_2	E_3	-	E_5	E_6

Figure 6.6: Sequence alignment example

The score of the sequence alignment constitutes the final shot similarity metric. To align two sequences we use the "Smith-Waterman" algorithm [69]. This is a well-known

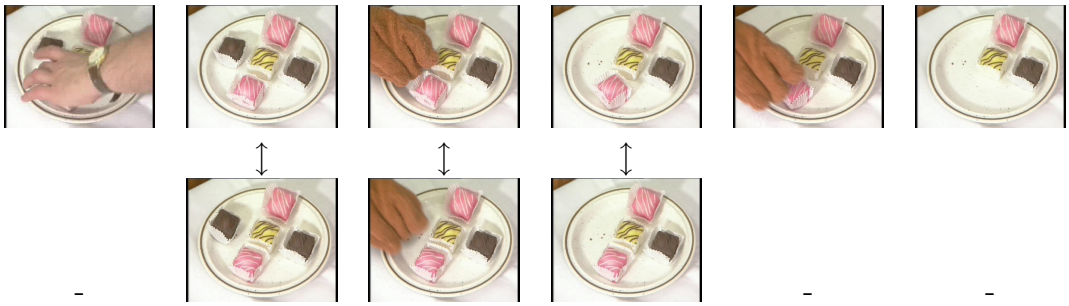


Figure 6.7: Sequence alignment of the key-frames of two shots describing the same scene.

algorithm for performing local sequence alignment in protein or nucleotide sequences and is guaranteed to find the optimal local alignment with respect to the scoring system being used, defined by a substitution matrix. The substitution matrix in our case is given by similarity matrix SM. The score of each alignment is normalized to be in range of 0-1:

$$Score_N = Score/min(n_{kf1}, n_{kf2}) \ , \tag{6.5}$$

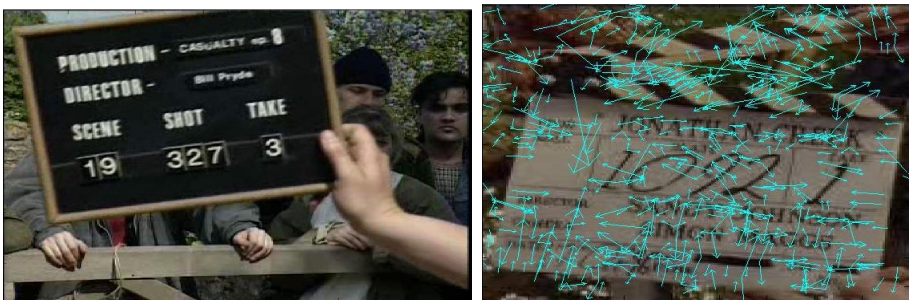
where n_{kf1}, n_{kf2} are the numbers of key-frames of the two shots under alignment, respectively.

In Fig. 6.7 we present the sequence alignment of the key-frames of two shots, where the three key-frame of the first shot are successfully aligned with the second, third and fourth key-frames of the second shot.

6.4.2 Repetitive shot detection

To find groups of repetitive and similar shots we compared each shot with the next three. If one of the three shots is similar with the shot under consideration then all the shots between these two shots and the shots under comparison, form a group. If none of the shots is similar then a new group of shots is considered and the algorithm continues until all shots are examined. Two shots are considered similar if the score of the sequence alignment of their sets of key-frames exceeds a predefined threshold which in our experiments was set to 0.9. Finally, the shot of each group with the largest duration

is selected as the representative for this group.



(a) Clapboard (b) Sift descriptors

Figure 6.8: Clapboard and its sift descriptors.

Figure 6.9: Comparison of the sift descriptors of two clapboards.

6.5 Clapboard Removal

So far, we have selected unique and non-repetitive shots which are represented by their key-frames. Rushes also contain clapboards to indicate the current number of the shot (see Fig. 6.8(a)). These frames should not be included in the final summarization, thus they have to be removed. Clapboards usually appear at the beginning of each shot.

To detect clapboards we compute for each key-frame the scale-invariant feature trans-

forms (SIFT) [52]. An example of the SIFT descriptors of a clapboard are shown in (see Fig. 6.8(b)). Using the TRECVID 2007 Development Data [59], a database of approximately 150 frames containing only clapboards was generated and their SIFT descriptors were calculated. In order to detect whether a key-frame contains a clapboard, we compute its SIFT descriptors and compare them with the SIFT descriptors of the database. In Fig. 6.9, we present the matching descriptors of two frames containing clapboards. If the number of matching descriptors is over a predefined threshold, this key-frame is characterized as clapboard and the cluster corresponding to this key-frame is removed from the shot. Having checked all the key-frames of a shot and having removed those key-frames characterized as clapboards and their corresponding clusters, we extract new key-frames for the shot using the method described in Section 3.2.

6.6 Summarization

The final stage of our summarization method involves the production of the final video summary. The summary of a video can be a set of key-frames or a video of a smaller duration than the original video. The method we described so far has produced unique, non-repetitive shots that are represented from their time-ordered key-frames. The goal of the rushes summarization process is to create a video summary with duration less than $p\%$ of the original video duration. Once the repetitive shots have been detected (Section 6.4), the shot with the largest duration is selected as their representative. The duration of a group of similar shots is referred as T_{all}. We want the duration of the summarized video T_{sum} for the specific group, to be $p\%$ of T_{all}. Suppose now that this shot is represented from k key-frames. A duration of $T_{kf} = T_{sum}/k$ is assigned to each key-frame. Finally, sampling every 3 frames, the $\lfloor T_{kf}/2 \rfloor$ preceding and $\lfloor T_{kf}/2 \rfloor$ following frames of each key-frame are selected to summarize the shot under consideration.

Table 6.1: Performance of our video rushes summarization method.

	Our method		All	
	Mean	Median	Avg.(Mean)	Avg.(Median)
IN (0-1)	0.53	0.56	0.44	0.44
JU (1-5)	3.31	3.33	3.17	3.21
TE (1-5)	2.50	2.33	2.76	2.75
RE (1-5)	3.16	3.33	3.3	3.36
DU (secs)	25.07	28.00	27.01	28.25
XD (secs)	6.64	5.17	4.69	3.93
TT (secs)	39.86	41.33	40.76	39.91
VT (secs)	27.57	30.33	29.31	30.47

6.7 Experiments

We have tested our method on TRECVID 2008 Test Data [59], under the Rushes Summarization task of TRECVID 2008 [59]. The goal of this task is to produce video summaries with duration less than or equal to $p = 2\%$ of the duration of the original video. The summary was to be constructed to maximize a viewer's efficiency in recognizing the main (primarily visual) objects and events from the original video as quickly as possible. The performance of our method was tested on 40 videos. The data are unedited video footage, shot mainly for five series of BBC drama programs, and was provided to TRECVid for research purposes by the BBC Archive [59]. The drama series included a historical drama set in London in the early 1900's, a series on ancient Greece, a contemporary detective program, a program on emergency services, a police drama, as well as miscellaneous scenes from other programs.

6.7.1 Evaluation metrics

Three humans at Dublin City University have judged each summary. The quality of each summary was evaluated directly by subjective and objective measures [59].

- Subjective measures

    1. The fraction of inclusions found in the summary (IN) ranging from 0 to 1.

2. Lack of junk (colorbars, clapboars and monochrome frames) (JU). The lack of junk score was an integer ranging from 1 (worst) to 5 (best).
3. Whether the summary had a pleasant tempo/rythm (TE). Score ranges from 1 (worst) to 5 (best).
4. Whether the summary contained lots of duplicate video (RE). Score ranges from 1 (worst) to 5 (best).

- Objective measures

1. Duration of summary in seconds (DU).
2. Different between target and actual summary size in seconds (XD).
3. Total time spent judging the inclusions in seconds (TT).
4. Total video play time (versus pause) judging the inclusions (VT).

6.7.2 Results and comparison

In Table 6.1 we present the scores of our method for the different measures given in Section 6.7.1. We also present the average mean and median for all 31 groups participated in the same task. It is worth mentioning that the proposed key-frame extraction algorithm efficiently summarizes the content of a shot, which is indicated from the high fraction of inclusions found in the summary (IN) that ranked our algorithm in the top ten in performance. In what concerns the removal of useless frames (JU), we observe that the results of our method are above average. Colorbars and monochrome were successfully removed from the summaries. However, clapboard removal could be further investigated and improved. Finally, the identification of repetitive information (RE) also needs improvement as indicated from the results.

6.8 Conclusions

In this Chapter, we have presented a method for video rushes summarization. There are three challenges concerning rushes summarization. The first one is the removal of junk frames (colorbars, clapboards and monochrome frames). The second one is the removal of repetitive shots generated from multiple takes of the same scene. The third one is the efficient summarization of the initial video in a summary video of much smaller duration describing most of the information provided in the initial video.

In the method we have proposed herein an improved spectral clustering algorithm was used to extract the key-frames of each shot as described in Section 3.2. Color-bars and monochrome frames were removed from each video using edge direction histograms, since they produce peaks in vertical and horizontal bins (colorbars) or empty bins (monochrome frames) and differ significantly form the edge direction histograms of normal frames. Frames containing clapboards were removed by comparing their SIFT descriptors with a set of descriptors of a database of clapboards that were computed in advance. To remove redundant video segments we have suggested a new shot similarity metric. A sequence alignment algorithm was employed to align the key-frames of the two shots under comparison with respect to their visual similarity and their temporal order. Next, similar segments were identified and only a single representative was kept. Finally, selecting a number of frames around each key-frame the final video summary was generated constituting an efficient representation of the initial video. Numerical results presented in this Chapter indicate that our system exhibited good performance in the Rushes Summarization task of TRECVID 2008.

Chapter 7

Event Detection and Classification in Video Surveillance Sequences

7.1 Introduction

In this Chapter, we present a system for event recognition and classification in video surveillance sequences. Video surveillance has received many attention over the last years

and is a major research topic in computer vision [34]. Typically, the framework of a video surveillance system involves the following stages: background substraction, environment modeling, object detection, classification and tracking of moving objects and descriptions of behaviors/events. The goal of video surveillance systems is to detect and characterize events as activities using unsupervised or supervised techniques.

In [18], a method is presented that integrates audio and visual information for scene analysis in a typical surveillance scenario, using only one camera and one monaural microphone. However, the proposed method is designed to discriminate specific events such as "making or receiving a call", "entering a room and switching on light". In [80], a video behavior modeling method is proposed for online normal behavior recognition and anomaly detection. For each video segment, blobs are detected that correspond to scene events. These scene events are clustered into groups using a gaussian mixture model producing a behavior representation for the video segment. Each behavior pattern is modeled using a Dynamic Bayesian Network and a spectral clustering algorithm is employed to cluster behavior patterns in groups. Then, a composite behavior model is constructed for the observed/expected behaviors. Finally, an anomaly measure is introduced to detect abnormal behavior, whereas normal behavior patterns are recognized using the Likelihood Ration Test method. In [4], a combination of Hidden Markov Model and stochastic grammar is proposed to recognize activities and identify different behaviors based on contextual information. In [91], an unsupervised technique for detecting unusual activity is proposed. A video sequence is segmented into equal length segments and for each frame a spatial histogram of detected objects is computed. The extracted spatial histograms are classified into prototypes and a prototype-segment co-occurrence matrix is computed from which unusual activity is detected.

The goal of video surveillance systems is first to detect those time intervals where a person performs an activity. Then, a crucial issue is to find the objects that correspond to this action by applying background substraction. Furthermore, each activity must be effectively represented in order to be classified in predefined categories. In our approach, local invariant descriptors are employed to remove background information.

Then, by analyzing the number of foreground descriptors, we automatically segment the video surveillance sequence into segments/events, which describe some activity taking place in the room under surveillance. Each video segment/event is represented either by a single (summary) visual word histogram or by multidimensional signal corresponding to the visual word histograms of its own frames. In the second case, Dynamic Time Warping distance [64] is employed to define a proper event dissimilarity metric. Finally, supervised and unsupervised techniques are implemented either to classify or to cluster events into categories.

The rest of the Chapter is organized as follows: In Section 7.2, the procedure of background substraction is described. In Section 7.3, the proposed event detection algorithm is presented. In Section 7.4, we define an event dissimilarity metric and in Section 7.5 we present numerical experiments for video event classification and clustering into categories. Finally, in Section 7.6, we provide some conclusions.

7.2 Background Substraction

For each frame of the video surveillance sequence we extract local invariant descriptors. In the proposed approach, we consider the SIFT descriptors proposed in [52]. Details of the extraction process are given in Section 5.2. In this work, we concentrate on different individual activities performed in an indoor environment, captured by using a standing camera. Thus, background remains the same and object/event detection relies on foreground detection modules. A popular idea for background substraction is proposed in [79, 72, 24], where background is characterized using Gaussian mixtures in a statistical framework. Alternative strategies for finding new objects in the scene involve motion based segmentation of some kind [50]. However, most flow computation methods are computationally heavy and very sensitive to noise. In our approach, for background extraction and event detection we use SIFT descriptors rather than image data, which offers several advantages, such as resistance to illumination changes, ability to cope with unusual

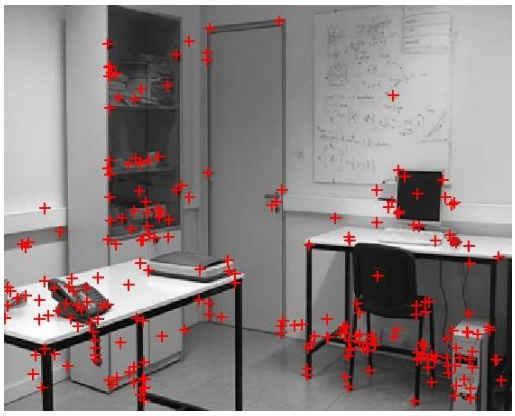


Figure 7.1: Video frame of the background and the location of the extracted descriptors.

motion activity, camouflaged foreground object detection, higher tolerance to background motion, and lower computation.

In order to remove descriptors that correspond to background objects, we compare the descriptors of each frame of the video surveillance sequence with a set of pre-computed descriptors corresponding to frames describing only the background. In Fig. 7.1, we present a video frame of the background and the location of the extracted descriptors.

Figure 7.2: Video frame with its descriptors.

Suppose now that D_b is the set of the descriptors extracted from video frames describing only the background and D_j is the set of descriptors of another video frame j. In order to find the descriptors of frame j that correspond to the background, each descriptor of set D_j is compared with the descriptors of set D_b to find a "match". According to a standard approach, the best candidate match for each descriptor is obtained by comparing

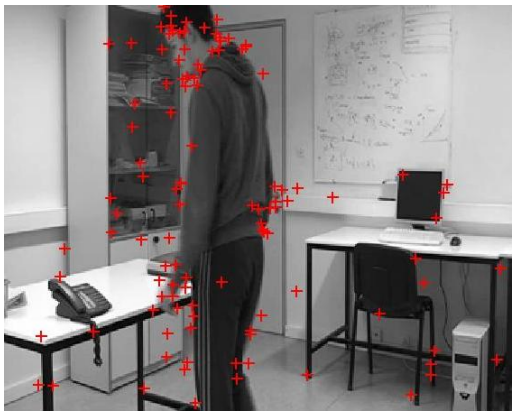


Figure 7.3: Video frame with unmatched descriptors.

the distance of its closest neighbor to the distance of its second-closest neighbor [52]. If it is lower than a threshold then a match is defined. The descriptors of set D_j that have a match in set D_b are removed, thus for frame j only the "unmatched" descriptors are kept. In Fig. 7.2, we present a video frame and the corresponding SIFT descriptors and in Fig. 7.3, we present the same video frame with the descriptors that do not match with those of set D_b.

7.3 Video Segmentation into Events

After we have subtracted the descriptors corresponding to background, we wish to identify unique events in the video sequence. In our surveillance problem a video event is defined as the time interval where a person performs an activity. Thus, it is expected that when someone enters the room under surveillance, new descriptors will appear that do not correspond to background. In [36] the authors propose a shot detection approach by analyzing a vector that corresponds to the number of matching descriptors between adjacent frames. In a way similar to that method we analyze a vector that corresponds to the number of "unmatched" descriptors between each frame and the background. In Fig. 7.4, we present the sequence of "unmatched" descriptors of a video surveillance sequence.

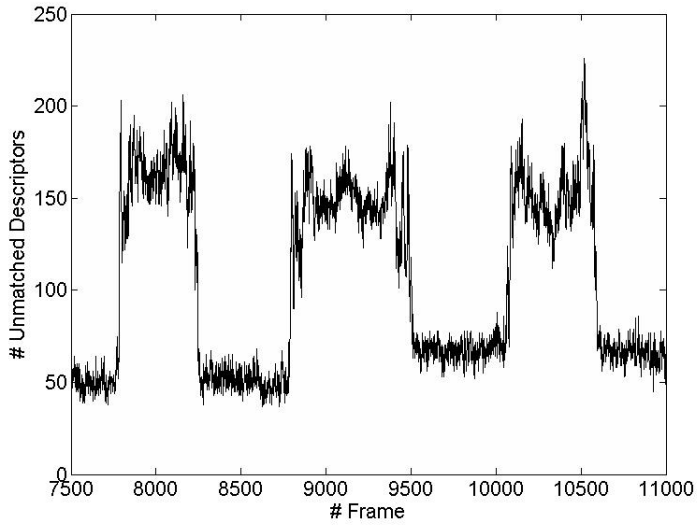


Figure 7.4: Number of unmatched descriptors of a video surveillance sequence.

In order to detect the beginning and the end of a video event, we first smooth this vector with the discretized gaussian kernel of equation (5.7). Furthermore, we discard low values of the smoothed signal to remove noise (background descriptors that have not been removed). In Fig. 7.5 we present the final smoothed signal for the sequence of Fig. 7.4.

Given the final smoothed signal S, we can detect a video event beginning at time instant t_b such that $S(t_b) > 0,\ S(t_{b-1}) = 0$ and ending at time t_e with $S(t_e) > 0,\ S(t_{e+1}) = 0$. In Fig. 7.6, we present the signal values for a video event. It can be observed that the event has at least two peaks, one close to the start of the event and the other close to its end. Typically, these two segments, i.e from the start to the first peak and from the last peak to the end, correspond to the entrance and exit from the room under surveillance of the person involved in the event, respectively. Thus, these two segments are removed and the remaining frames (their corresponding descriptors) constitute the event.

7.3.1 Event Representation

After we have segmented the video into events, we represent each video frame of the event with a visual word histogram following the method presented in Section 5.2.2. More specifically, the descriptors of all event frames are clustered into a predefined number of

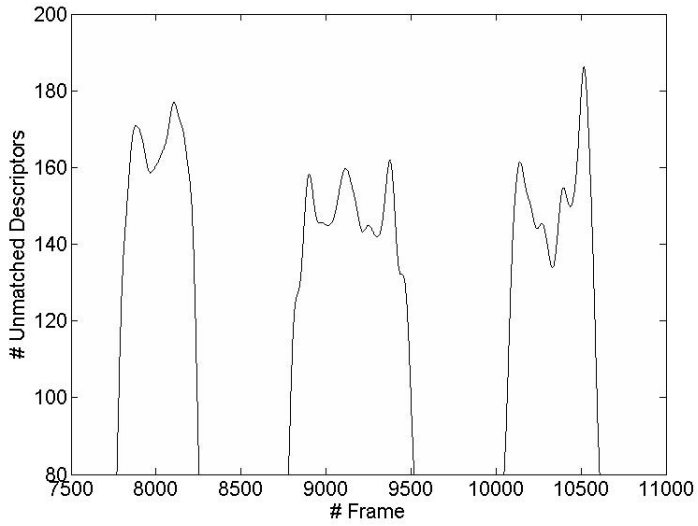


Figure 7.5: Smoothed signal of the number of unmatched descriptors.

clusters K using the k-means algorithm, where K denotes the size of the visual words vocabulary. A visual word histogram VHF_j^i of a frame j of event i is constructed by assigning the frame's descriptors to one of the K visual words (clusters). Similarly, a visual word histogram VHE_i of an event i is constructed by assigning each descriptor of all the event frames to one of the K visual words (clusters).

7.4 Event Dissimilarity

In order to proceed with video event classification an event dissimilarity metric must be defined. In our approach we consider two approaches. In the first one, to compute a distance value between two events E_i and E_l we compare their corresponding visual word histograms VHE_i and VHE_l. In the second approach, we compare the visual word histograms VHF of their frames. More specifically, suppose that we are given events $E_i = \{f_1^i, \ldots, f_{N_i}^i\}$ and $E_l = \{f_1^l, \ldots, f_{N_l}^l\}$. Since $N_i \neq N_l$, we have to define a proper dissimilarity metric to compare these two events. Next, we describe Dynamic Time Warping (DTW) distance, which is employed to compare two events with different number of frames.

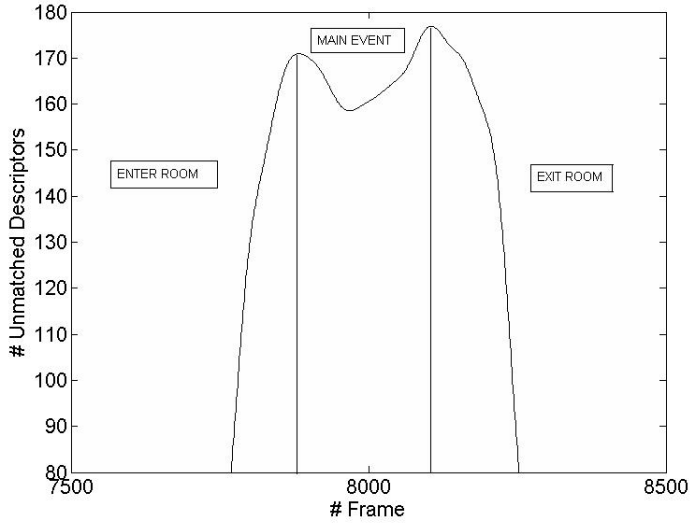


Figure 7.6: Descriptors selection of a video event.

7.4.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a well-known technique to find an optimal alignment between two given time-independent sequences. Originally, DTW has been used to compare different speech patterns in automatic speech recognition [64]. The objective of DTW is to compare two time-dependent sequences; $X = (x_1, x_2, \ldots, x_N)$ of length N and $Y = (y_1, y_2, \ldots, y_M)$ of length M, $x_i, y_j \in \mathbb{R}$. The first step in computing the DTW distance between time-series X and Y is to construct a $N \times M$ distance matrix D containing the pairwise distances between the samples, i.e $D_{ij} = (x_i - y_j)^2$, $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. The objective is to find a path through the matrix so that the cumulative distance between X and Y is minimized. This path called *warping path* $W = (w_1, w_2, \ldots, w_L)$, $max(N, M) \leq L < N + M - 1$ is a contiguous set of matrix elements that defines a mapping between X and Y. The warping path must satisfy the following criteria [64]:

1. *Boundary condition*: $w_1 = (1,1)$ and $w_L = (N, M)$. The starting and ending points of the warping path must be the first and the last points of the aligned sequences.

2. *Monotonicity condition*: This condition preserves the time-ordering of points.

3. *Step size condition*: This criterion limits the warping path from long jumps (shifts in time) while aligning sequences. The basic step size condition is formulated as $(w_{l+1} - w_l) \in \{(1,1),(1,0),(0,1)\}$.

The cost function associated with a warping path w is computed with respect to matrix D:

$$c_w(X,Y) = \sum_{l=1}^{L} D(x_{il}, y_{jl}). \tag{7.1}$$

The DTW distance minimizes this cost function [64]:

$$DTW(X,Y) = min\{c_w(X,Y), w \in W^{N\times M}\}, \tag{7.2}$$

where $W^{N\times M}$ is the set of all possible warping paths. Then, the accumulated cost matrix DA is computed as follows:

1. $DA(1,j) = \sum_{k=1}^{j} D(x_1, y_k),\ j \in [1,M]$.
2. $DA(i,1) = \sum_{k=1}^{i} D(x_k, y_1),\ i \in [1,N]$.
3. $DA(i,j) = D(x_i,y_i) + min \begin{cases} DA(i,j-1) \\ DA(i-1,j) \\ DA(i-1,j-1) \end{cases}$.

Once the accumulated cost matrix DA is built, the optimal warping path could be found by the simple backtracking from the point $w_{end} = (N,M)$ to the w_{start} = (1,1). The final DTW distance is given from the sum of the values of matrix DA following the optimal path.

7.4.2 Event Dissimilarity Metric

Each frame $f_j^i,\ j = 1,\ldots,N_i$ of event E_i is represented with a visual word histogram VHF_j^i as defined in equation (5.6). Thus, event E_i is represented by a K-dimensional

signal of length N_i:

$$VE_i = \begin{pmatrix} VHF_1^i(k=1) & \dots & VHF_{N_i}^i(k=1) \\ \vdots & \dots & \vdots \\ VHF_1^i(k=K) & \dots & VHF_{N_i}^i(k=K) \end{pmatrix}, \tag{7.3}$$

where K the size of the vocabulary size employed to create the visual word histograms in Section 5.2.2. Each row k of matrix VE_i represents the frequency of "visual word" k in the time interval of the event.

In order to compute the distance between two video segments/events E_i and E_l we compute the average DTW distance of their K-multidimensional signals. More specifically

$$D(E_i, E_l) = \frac{1}{K} \sum_{k=1}^{K} DTW(VHF^i(k), VHF^l(k)), \tag{7.4}$$

where $VHF^i(k), VHF^l(k)$ are the k-th rows of the K-dimensional signals VE_i, VE_l representing segments/events E_i and E_l, respectively.

7.5 Experimental Results

In this Section, we present numerical experiments for event classification and clustering using supervised and unsupervised learning methods.

7.5.1 Video surveillance sequences

As already mentioned in Section 7.2 we concentrate on different individual activities performed in an indoor environment captured by a standing camera. The performed activities are not overlapping, in the sense that a person enters the room performs a set of basic actions and leaves the room. The fist video sequence comprises of more than 25000 frames. In this video sequence, 20 activities/events are performed that are divided in five categories, as follows.

1. Phone:a person enters the room, goes to the phone and makes a conversation.

2. Scanner:a person enters the room, goes to the scanner and scans a document.

3. Library:a person enters the room, goes to the library and opens it.

4. Computer:a person enters the room, goes to the computer and works with it.

5. Board:a person enters the room, goes to the board and writes something.

In Fig. 7.7, we present sample frames of the background and the five categories of events.

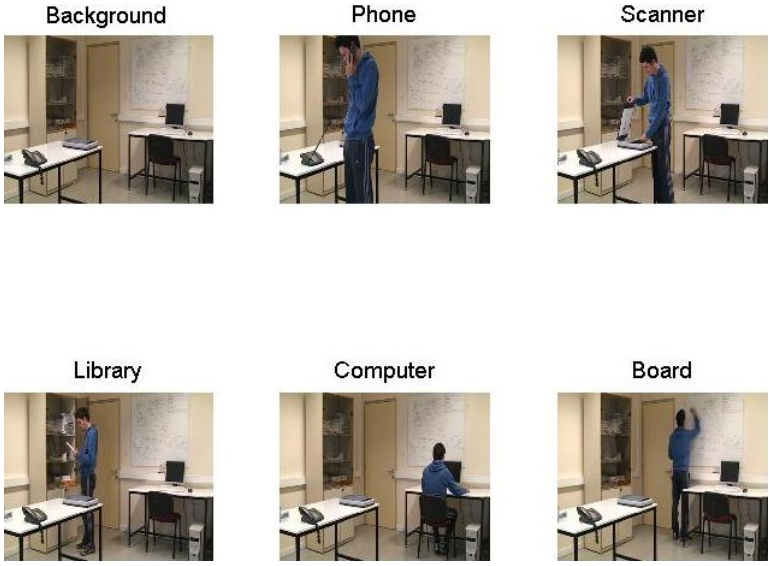


Figure 7.7: Sample frames of the background and the five categories of events.

The result of the automatic segmentation was optimal, since no over-segmentation or under-segmentation was performed and all 20 events were detected as unique.

7.5.2 Classification Results

To classify the 20 events into 5 categories we carried out two experiments. In the first one, we used the nearest neighbor classifier [23] and in the second one we used Support Vector Machines [17]. We implemented the nearest neighbor classifier with 1, 3, and 5 nearest neighbors for both dissimilarity measures defined in Section 7.4. Comparison between the

Table 7.1: Classification results for the first video sequence.

K	1-NN		3-NN		5-NN		SVM	
	DTW	EV	DTW	EV	DTW	EV	DTW	EV
10	80%	85%	80%	85%	65%	65%	75%	65%
20	90%	90%	95%	90%	90%	80%	95%	95%
50	95%	95%	95%	95%	95%	90%	100%	95%
100	95%	90%	100%	100%	10%	95%	100%	95%
200	95%	90%	100%	100%	100%	100%	100%	100%

visual word histograms of events is referred as EV and comparison between the visual word histograms of the frames of the events is referred as DTW. In Table 7.1 we present the numerical results of the experiments for different number of visual words K. The classification accuracy was estimated using the leave-one-out (LOO) approach [23].

In the second experiment, Support Vector Machine (SVM) classifiers [17] were employed using the leave one out (LOO) scheme again. In our approach, we employed the typical radial basis function (RBF) kernel (equation (2.15)) and the parameters C, γ of equations (2.11), (2.15) respectively, were selected through cross-validation. In Table 7.1, we present the numerical results for the two compared approaches of Section 7.4 and for different number of visual words K. It can be observed that DTW distance gives results slightly superior to the ones obtained by the other dissimilarity metric.

7.5.3 Clustering Results

We have also employed an unsupervised method for grouping the video events into categories. More specifically, we performed agglomerative hierarchical clustering [37], setting the number of cluster to five and using the Ward criterion to select the clusters to be merged at each iteration. In Fig. 7.8, we present the hierarchical clustering dendrogram using DTW distance to compare video events using a visual word vocabulary of 50 words. In this example all video events were successfully clustered into groups. The clustering accuracy was calculated by computing the number of errors: a clustering error occurs if an event is assigned to a cluster in which the majority of the events belongs to another category. In Table 7.2 we present the clustering accuracy for the two approaches of Section

7.4 using a different number of visual words K. It can be observed that DTW distance provides better results for a small number of visual words.

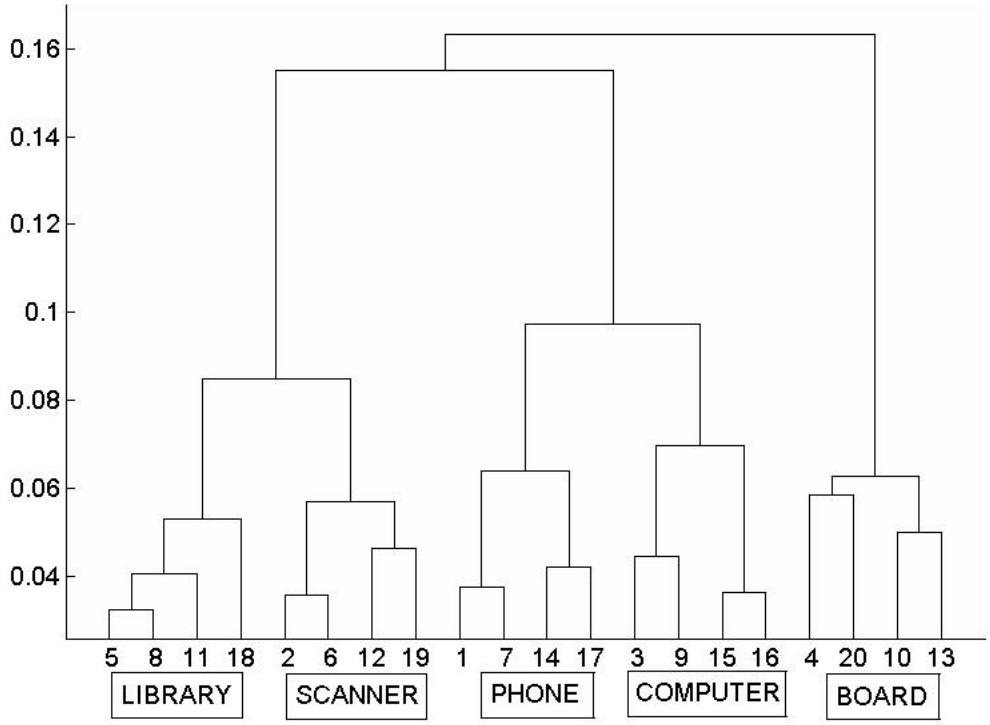


Figure 7.8: Hierarchical clustering dendrogram.

Table 7.2: Clustering results for the first video sequence.

K	Hierarchical Clustering	
	DTW	EV
10	80%	45%
20	95%	90%
50	100%	90%
100	100%	100%
200	100%	100%

7.6 Conclusions

In this Chapter, we have presented a method for video event detection and classification in video surveillance sequences. For each video frame, local invariant descriptors were computed and compared to a pre-computed set of descriptors from the background framer of the surveillance room. In this way, a number of "unmatched" descriptors was identified that describe foreground objects. By analyzing the number of "unmatched" descriptors, the video sequence was segmented into segments/events. Each video event was represented either by a single (summary) visual word histogram or by a K-dimensional signal corresponding to the visual word histograms of its frames. Thus, two different approaches were followed in order to compare video events. Unsupervised and supervised learning methods were employed to cluster and classify the events into certain categories. Numerical results presented in this Chapter indicate that our approach achieves high detection, classification and clustering rates.

Chapter 8

Conclusions

8.1 Concluding Remarks

In this thesis we have proposed novel methods for video segmentation and representation that are based on machine learning techniques (classification, clustering). First, we considered support vector machines for video shot detection. Then, an improved spectral clustering algorithm was employed for video shot representation. The same algorithm in combination with a sequence alignment algorithm was employed for video scene segmentation. Movie segmentation into scenes and chapters was also implemented using temporally smoothed visual words histograms. Furthermore, the proposed techniques were also employed for video rushes summarization and event detection in video surveillance sequences.

More specifically, in order to perform video shot detection, we proposed in Chapter 2 a supervised learning methodology [11, 15]. In this way, we have avoided the use of thresholds and we were able to detect shot boundaries of videos with totally different

visual characteristics. Novel features have been defined describing the variation between adjacent frames and the contextual information in a neighborhood of frames and became inputs to a SVM classifier which categorized transitions to normal, abrupt and gradual. In this way, all categories of video shot transitions were detected simultaneously. Numerical experiments that compare our algorithm with threshold dependent methods and another supervised learning methodology indicate that our algorithm provides superior results.

In Chapter 3 a key-frame extraction algorithm [10, 14] has been presented that is based on the combination of spectral clustering approach and fast global k-means algorithm. We have also proposed a technique to estimate the number of the extracted key-frames. The extracted key-frames are unique, non-repetitive and summarize the video shot content, which is also indicated from the numerical experiments where appropriate quality measures were defined and computed.

In Chapter 4 we presented a novel video scene segmentation algorithm [9, 14] that employs the improved spectral clustering algorithm of Chapter 3 and a sequence alignment algorithm. Shots were first clustered into groups based only on their visual similarity using the method presented in Chapter 3 and a label was assigned to each shot according to the group that it belonged to. Then, a sequence alignment algorithm was applied to detect when a change occurs to the pattern of shot labels, providing the final scene segmentation result. Numerical experiments on TV-series and movies have shown that the proposed scene detection method accurately detects most of the scene boundaries, while preserving good tradeoff between recall and precision.

In Chapter 5 we presented a high-level movie segmentation algorithm [13]. In this approach, the movie shots were represented with local invariant descriptors instead of color histograms, resulting into a visual words histogram representation. Next the visual words histograms of shots were temporally smoothed (using a gaussian kernel) with respect to histograms of neighboring shots in order to preserve valuable contextual information. This semantic smoothing process at different time scales results in efficient movie segmentation at different high-levels, such as scenes and chapters.

In Chapter 6, a system for video rushes summarization [12] has been presented. A

video sequence was first segmented into shots and key-frames were extracted for each shot using the method presented in Chapter 3. Then, the edge direction histogram of each key-frame was computed to check whether the frame is monochrome or a colorbar. In order to remove redundant information (repetitive shots), we compared shots using a sequence alignment score between the sets of their key-frames. The SIFT descriptors of the key-frames of the remaining representative shots were compared to a database of descriptors obtained from frames with clapboards. In that way, frames with clapboards were removed from the final video summary. Finally, the video summary was generated by concatenating frames around the key-frames of the remaining shots. Numerical results indicate that our system exhibited good performance in the Rushes Summarization task of TRECVID 2008.

In Chapter 7, a system for event detection in video rushes surveillance sequences has been presented. First, local invariant descriptors of video frames were employed to remove background information and segment the video into events. Next, visual word histograms were computed for each video event and used to define a distance measure between events. Finally, machine learning techniques were employed to classify events into predefined categories. Numerical experiments indicate that the proposed approach provids high event detection and classification rates.

8.2 Directions for Future Research

In future work it would be very interesting to consider the use of local invariant descriptors for the shot boundary detection problem of Chapter 2. The experiments of Chapter 5 using this type of descriptors are very promising and the proposed semantic smoothing process could also be applied to the shot boundary detection problem.

The temporally smoothed visual histograms of Chapter 5 could be used as well for the key-frame extraction algorithm. Using this method it would be interesting to examine in what degree the semantic concepts change inside the video shot. Another interesting

direction for future work would be to build a visual word vocabulary from different features such as motion and local invariant descriptors or even a combination of features.

Furthermore, in the high-level movie segmentation problem of Chapter 5, it would be interesting to build a visual word vocabulary by comparing video shots to existing semantic detectors [71]. An issue that worths to be examined is whether global semantic concepts, i.e. cars, people, animals and buildings, can provide better performance than semantic concepts extracted from local descriptors.

The high-level segmentation algorithm presented in Chapter 5 could also be tested for a variety of video genres, i.e sports and tv-news. In sport videos, such as a video sequence describing a basketball game, it would be very desirable to provide a video segmentation into play-time, replays, timeouts and other possible segments. In tv-news a possible segmentation could divide the video sequence into dialogues, reportage and tv-commercials.

Finally, in video surveillance there are several open problems that deserve further investigation. For example, a very challenging problem is the detection of multiple events in a video sequence. The majority of event detection algorithms consider independent events. The combination of motion trajectories with the semantic representation of video frames could provide a possible solution to this difficult problem.

Bibliography

[1] J. Bescos, G. Cisneros, J. M. Martinez, J. M. Menendez, and J. Cabrera. A unified model for techniques on video-shot transition detection. *IEEE Transactions on Multimedia*, 7(2):293–307, April 2005.

[2] A. Del Bimbo. *Visual information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[4] A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 196–202, 1998.

[5] G. Boccignone, A. Chianese, V. Moscato, and A. Picariello. Foveated shot detection for video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(3):365–377, March 2005.

[6] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–128, 1996.

[7] Z. Cernekova, I. Pitas, and C. Nikou. Information theory-based shot cut/fade detection and video summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(1):82–91, January 2006.

[8] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279, December 1999.

[9] V. Chasanis, A. Likas, and N. Galatsanos. Scene detection in videos using shot clustering and symbolic sequence segmentation. In *MMSP '07: Proceedings of the IEEE 9th Workshop on Multimedia Signal Processing*, pages 187–190, Chania, Greece, October 2007.

[10] V. Chasanis, A. Likas, and N. Galatsanos. Efficient video shot summarization using an enhanced spectral clustering approach. In *ICANN '08: Proceedings of the 18th International Conference on Artificial Neural Networks, Part I*, pages 847–856, Prague, Czech Republic, September 2008.

[11] V. Chasanis, A. Likas, and N. Galatsanos. A support vector machine approach for video shot detection. In *IIMSS '08: Proceedings of the 1st International Symposium on Intelligent Interactive Multimedia Systems and Services*, pages 45–54, Piraeus, Greece, July 2008.

[12] V. Chasanis, A. Likas, and N. Galatsanos. Video rushes summarization using spectral clustering and sequence alignment. In *TVS '08: Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 75–79, Vancouver, British Columbia, Canada, 2008.

[13] V. Chasanis, A. Likas, and N. Galatsanos. Movie segmentation into scenes and chapters using locally weighted bag of visual words. In *CIVR '09: Proceedings of the ACM International Conference on Image and Video Retrieval*, Santorini, Greece, 2009.

[14] V. Chasanis, A. Likas, and N. Galatsanos. Scene detection in videos using shot clustering and sequence alignment. *IEEE Transactions on Multimedia*, 11(1):89–100, January 2009.

[15] V. Chasanis, A. Likas, and N. Galatsanos. Simultaneous detection of abrupt cuts and dissolves in videos using support vector machines. *Pattern Recognition Letters*, 30(1):55 – 65, January 2009.

[16] M. Christel, A. Hauptmann, W. H. Lin, M. Y. Chen, J. Yang, B. Maher, R. Baron, and V. Robert. Exploring the utility of fast-forward surrogates for bbc rushes. In *TVS '08: Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 35–39, Vancouver, British Columbia, Canada, 2008.

[17] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[18] M. Cristani, M. Bicego, and V. Murino. Audio-visual event recognition in surveillance video sequences. *IEEE Transactions on Multimedia*, 9(2):257–267, 2007.

[19] A. Dailianas, R. B. Allen, and P. England. Comparison of automatic video segmentation algorithms. In *Proceedings SPIE on Integration Issues in Large Commercial Media Delivery Systems*, volume 16, pages 2–16, 1995.

[20] C. Dalatsi, S. Krinidis, S. Tsekeridou, and I. Pitas. Use of support vector machines based on color and motion features for shot boundary detection. In *IST '01:Proceedings of International Symposium on Telecommunications*, 2001.

[21] M. Detyniecki and C. Marsala. Adaptive acceleration and shot stacking for video rushes summarization. In *TVS '08: Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 109–113, Vancouver, British Columbia, Canada, 2008.

[22] A. D. Doulamis, N. Doulamis, and S. Kollias. Non-sequential video content representation using temporal variation of feature vectors. *IEEE Transactions on Consumer Electronics*, 46(3):758–768, August 2000.

[23] R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[24] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density for visual surveillance. In *Proceedings of the IEEE*, pages 1151–1163, 2002.

[25] J. Fan, A. K. Elmagarmid, X. Zhu, W. G. Aref, and L. Wu. Classview: Hierarchical video shot classification, indexing, and accessing. *IEEE Transactions on Multimedia*, 6:70–86, 2004.

[26] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR '05: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531, June 2005.

[27] H. Feng, W. Fang, S. Liu, and Y. Fang. A new general framework for shot boundary detection and key-frame extraction. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 121–126, Hilton, Singapore, 2005.

[28] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull. Fade and dissolve detection in uncompressed and compressed video sequences. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 299–303, 1999.

[29] U. Gargi, R. Kasturi, and S. H. Strayer. Performance characterization of video-shot-change detection methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):1–13, February 2000.

[30] C. Gianluigi and S. Raimondo. An innovative algorithm for key frame extraction in video summarization. *Journal of Real-Time Image Processing*, 1(1):69–88, 2006.

[31] A. Hanjalic. Shot-boundary detection: unraveled and resolved? *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):90–105, February 2002.

[32] A. Hanjalic, R. L. Lagendijk, and J. Biemond. Automated high-level movie segmentation for advanced video-retrieval systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):580–588, 1999.

[33] C. Harris and M. Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[34] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics*, 34:334–352, 2004.

[35] C.-R. Huang, C.-S. Chen, and P.-C. Chung. Contrast context histogram - a discriminating local descriptor for image matching. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, volume 4, pages 53–56, Los Alamitos, CA, USA, 2006.

[36] C.-R. Huang, H.-P. Lee, and C.-S. Chen. Shot change detection via local keypoint matching. *IEEE Transactions on Multimedia*, 10(6):1097–1108, October 2008.

[37] A. Jain and R. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[38] L. Jiebo, C. Papin, and K. Costello. Towards extracting semantically meaningful key frames from personal video clips: From humans to computers. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(2):289–301, February 2009.

[39] N. C. Jones and P. A. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, Cambridge, MA, 2004.

[40] R. Kasturi and R. C. Jain. *Computer Vision: Principles*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[41] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

[42] V. Kobla, D. DeMenthon, and D. Doermann. Special effect edit detection using VideoTrails: a comparison with existing techniques. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, pages 302–313, 1999.

[43] I. Kononenko and M. Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, 2007.

[44] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477–500, January 2001.

[45] G. Lebanon, Y. Mao, and J. Dillon. The locally weighted bag of words framework for document representation. *Journal of Machine Learning Research*, 8:2405–2441, 2007.

[46] D. Lelescu and D. Schonfeld. Statistical sequential analysis for real-time video scene change detection on compressed multimedia bitstream. *IEEE Transactions on Multimedia*, 5(1):106–117, March 2003.

[47] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *Proceedings of SPIE on Storage and Retrieval for Image and Video Databases*, volume 29, pages 290–301, January 1999.

[48] R. Lienhart. Reliable dissolve detection. In *Proceedings of SPIE on Storage and Retrieval for Still Image and Video Databases*, number 4315, pages 219–230, January 2001.

[49] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[50] A. Lipton. Local application of optic flow to analyse rigid versus non-rigid motion. In *In Proceedings of ICCV Workshop on Frame-Rate Vision*, September 1999.

[51] T. Liu, X. Zhang, J. Feng, and K.-T. Lo. Shot reconstruction degree: a novel criterion for key frame selection. *Pattern Recognition Letters*, 25(12):1451 – 1457, 2004.

[52] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[53] B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715, 2001.

[54] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In *Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II*, pages 113–127, Amsterdam, The Netherlands, 1992.

[55] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[56] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[57] C. W. Ngo, T. C. Pong, and R. T. Chin. Video partitioning by temporal slice coherency. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(8):941–953, Aug 2001.

[58] J. M. Odobez, D. Gatica-Perez, and M. Guillemot. Spectral structuring of home videos. In *CIVR '03: Proceedings of International Conference on Image and Video Retrieval*, Lecture Notes in Computer Science, Urbana-Champaign, USA, July 2003.

[59] P. Over, A. Smeaton, and G. Awad. The trecvid 2008 bbc rushes summarization evaluation. In *TVS '08: Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 1–20, New York, NY, USA, 2008.

[60] M.-H. Park, R.-H Park, and S.W. Lee. Shot boundary detection using scale invariant feature matching. In *SPIE Proceedings on Visual Communications and Image Processing*, volume 6077, pages 569–577, 2006.

[61] Z. Rasheed and M. Shah. Scene detection in hollywood movies and tv shows. In *CVPR '03: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, page 343, Los Alamitos, CA, USA, 2003.

[62] Z. Rasheed and M. Shah. Detection and representation of scenes in videos. *IEEE Transactions on Multimedia*, 7(6):1097–1105, December 2005.

[63] Y. Rui, T. S. Huang, and S. Mehrotra. Exploring video structure beyond the shots. In *Proceedings of IEEE Conference on Multimedia Computing and Systems*, pages 237–240, 1998.

[64] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[65] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of 17th International Conference on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[66] I. K. Sethi and N. Patel. A statistical approach to scene change detection. In *Proceedings of SPIE on Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 329–339, 1995.

[67] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[68] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006.

[69] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[70] S. Smoliar and H. Zhang. Content-based video indexing and retrieval. *IEEE MultiMedia*, 1(2):62–72, 1994.

[71] C. G. M. Snoek, B. Huurnink, L. Hollink, M. de Rijke, G. Schreiber, and M. Worring. Adding semantics to detectors for video retrieval. *IEEE Transactions on Multimedia*, 9(5):975–986, August 2007.

[72] C. Stauffer, W. Eric, and L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2000.

[73] H. Sundaram and C. Shih-Fu. Computable scenes and structures in films. *IEEE Transactions on Multimedia*, 4(4):482–491, Dec 2002.

[74] M. Swain and D. Ballard. Color indexing. *International Journal Computer Vision*, 7(1):11–32, November 1991.

[75] P. Tirilly, V. Claveau, and P. Gros. Language modeling for bag-of-visual words image categorization. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 249–258, Niagara Falls, Canada, 2008.

[76] V. Valdés and J. M. Martínez. Binary tree based on-line video summarization. In *TVS '08: Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 134–138, Vancouver, British Columbia, Canada, 2008.

[77] T. Volkmer, S. M. M. Tahaghoghi, and H. Williams. Rmit university at trecvid 2004. In *In Proceedings of the TREC Video Retrieval Evaluation (TRECVID) Workshop 2004*, Gaithersburg, MD, USA, November 2004.

[78] W. Wolf. Key frame selection by motion analysis. In *ICASSP '96: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1228–1231, May 1996.

[79] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[80] T. Xiang and S. Gong. Video behavior profiling for anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):893–908, 2008.

[81] E. P. Xing and M. I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report UCB/CSD-03-1265, EECS Department, University of California, Berkeley, June 2003.

[82] J. Yang, Y.-G. Jiang, A. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *MIR '07: Proceedings of the International Workshop on Multimedia Information Retrieval*, pages 197–206, Augsburg, Bavaria, Germany, 2007.

[83] B.-L. Yeo and B. Liu. Rapid scene analysis on compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):533–544, December 1995.

[84] M. Yeung, B.-L. Yeo, and B. Liu. Segmentation of video by clustering and graph analysis. *Computer Vision and Image Understanding*, 71(1):94–109, 1998.

[85] J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, and B. Zhang. A formal study of shot boundary detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(2):168–186, February 2007.

[86] Y. Yusoff, W. Chrismas, and J. Kittler. Video shot cut detection using adaptive threshold. In *BMVC '00: Proceedings of 11th British Machine Vision Conference*, 2000.

[87] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *Proceedings of ACM on Multimedia*, pages 189–200, San Francisco, California, United States, 1995.

[88] H. Zha, C. Ding, M. Gua, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14*, pages 1057–1064. MIT Press, 2001.

[89] Yun Zhai and M. Shah. Video scene segmentation using markov chain monte carlo. *IEEE Transactions on Multimedia*, 8(4):686–697, August 2006.

[90] H. Zhang, A. Kankanhalli, and S.Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28, 1993.

[91] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. volume 2, pages 819–826, 2004.

[92] Y. Zhuang, Y. Rui, T. Huang, and S. Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *ICIP '98:Proc of International Conference on Image Processing*, volume 1, pages 866–870, Chicago, IL, USA, 1998.

Author's Publications

Journal Publications

1. V. Chasanis, A. Likas, and N. Galatsanos. Simultaneous detection of abrupt cuts and dissolves in videos using support vector machines. Pattern Recognition Letters, 30(1):55-65, 2009.

2. V. Chasanis, A. Likas, and N. Galatsanos. Scene detection in videos using shot clustering and sequence alignment. IEEE Transactions on Multimedia, 11(1):89-100, January 2009.

Conference Publications

1. V. Chasanis, A. Likas, and N. Galatsanos. Scene detection in videos using shot clustering and symbolic sequence segmentation. In Proceedings of IEEE 9th Workshop on Multimedia Signal Processing, pp. 187-190, Chania, Greece, October 2007.

2. V. Chasanis, A. Likas, and N. Galatsanos. A support vector machine approach for video shot detection. In Proceedings of the 1st International Symposium on Intelligent Interactive Multimedia Systems and Services, pp. 45-54, Peraias, Greece, July 2008.

3. V. Chasanis, A. Likas, and N. Galatsanos. Efficient video shot summarization using an enhanced spectral clustering approach. In Proceedings of the 18th International

Conference on Artificial Neural Networks, Part I, pp. 847-856, Prague, Czech Republic, September 2008.

4. V. Chasanis, A. Likas, and N. Galatsanos. Video rushes summarization using spectral clustering and sequence alignment. In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop, Vancouver, Canada, October 2008.

5. V. Chasanis, A. Kalogeratos, and A. Likas. Movie segmentation into scenes and chapters using locally weighted bag of visual words. In Proceedings of ACM International Conference on Image and Video Retrieval, Santorini, Greece, July 2009.

Curriculum Vitae

Vasileios Chasanis was born in Ioannina, Greece in 1981. He received the Diploma Degree in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2004. Since 2005 he has been a Ph.D. candidate in the Department of Computer Science, University of Ioannina, Greece.

He has been involved in a research project and has published 2 papers in scientific journals and 5 papers in refereed conference proceedings. His research interests are in the areas of machine learning for video analysis and summarization, multimedia information retrieval and real time video surveillance.