# INTELLIGENT VIDEO PROCESSING USING DATA MINING TECHNIQUES

Shirahama, Kimiaki

Doctoral Dissertation

# INTELLIGENT VIDEO PROCESSING USING DATA MINING TECHNIQUES

January 2011

Graduate School of Engineering, Kobe University

**KIMIAKI SHIRAHAMA**

Doctoral Dissertation

# INTELLIGENT VIDEO PROCESSING USING DATA MINING TECHNIQUES

January 2011

Graduate School of Engineering, Kobe University

**KIMIAKI SHIRAHAMA**

# Abstract

Due to the rapidly increasing video data on the Web, much research effort has been devoted to develop video retrieval methods which can efficiently retrieve videos of interest. Considering the limited man-power, it is much expected to develop retrieval methods which use features automatically extracted from videos. However, since features only represent physical contents (e.g. color, edge, motion, etc.), retrieval methods require knowledge of how to use/integrate features for retrieving relevant videos to a query. To obtain such knowledge, this dissertation concentrates on *video data mining* where videos are analyzed using data mining techniques which extract previously unknown, interesting patterns in underlying data. Thereby, patterns for retrieving relevant videos are extracted as explicit knowledge.

Queries can be classified into three types. For the first type, a user can find keywords suitable for retrieving relevant videos. For the second type, the user cannot find such keywords due to the lexical ambiguity, but can provide some example videos. For the final type of queries, the user has neither keywords nor example videos. Thus, we develop a video retrieval system with 'multi-modal' interfaces by implementing three video data mining methods to support each of the above three query types. For the first query type, the system provides a *Query-By-Keyword* (QBK) interface where patterns which characterize videos relevant to certain keywords are extracted. For the second query type, a *Query-By-Example* (QBE) interface is provided where relevant videos are retrieved based on their similarities to example videos provided by the user. So, patterns for defining meaningful similarities are extracted using example videos. For the final query type, a *Query-By-Browsing* (QBB) interface is devised to characterize impressive video segments, so that the user can browse videos to find keywords or example videos. Finally, to improve retrieve performance, the integration of the QBK and QBE interfaces is explored where informations from text and image/video modalities are

interchanged using knowledge base.

The developed video data mining methods and the integration method are summarized as follows: In chapter 2, we develop a video data mining method for the QBK interface. This method focuses that a certain semantic content is presented by concatenating several shots taken by different cameras. Thus, the method extracts *sequential patterns* which relate adjacent shots relevant to certain keyword queries. Sequential patterns are extracted by connecting characteristic features in adjacent shots. However, the extraction of sequential patterns requires an expensive computation cost because a huge number of sequences of features have to be examined as candidates of patterns. Hence, time constraints are adopted to eliminate semantically irrelevant sequences of features.

In chapter 3, a method for the QBB interface is devised. We assume that impressive actions of a character are presented by abnormal video editing patterns. For example, thrilling actions of the character are presented by shots with very short durations while his/her romantic actions are presented by shots with very long durations. Based on this, the method detects *bursts* as patterns consisting of abnormally short or long durations of the character's appearance. The method firstly performs a probabilistic time-series segmentation to divide a video into segments characterized by distinct patterns of the character's appearance. It then examines whether each segment contains a burst or not.

In chapter 4, we develop a method for the QBE interface by focusing on a large variation of relevant shots. Specifically, even for the same query, relevant shots contain significantly different features due to varied camera techniques and settings. Thus, *rough set theory* is used to extract multiple patterns which characterize different subsets of example shots. Although this requires counter-example shots which are compared to example shots, they are not provided. Hence, *partially supervised learning* is used to collect counter-example shots from a large set of shots left behind in the database. In particular, counter-example shots which are as similar to example shots as possible, are collected as they are useful for characterizing the boundary between relevant and irrelevant shots.

In chapter 5, we address the integration of QBK and QBE. To achieve this, we construct a *video ontology* where concepts such as *Person*, *Car* and *Building* are organized into a hierarchical structure. The video ontology is constructed by considering the generalization/specialization relation among concepts and their co-occurrences in the same shots. Given the textual de-

scription of a query (i.e. QBK), concepts related to the query are selected by tracing the hierarchical structure of the video ontology. Shots where few of selected concepts are detected are filtered. After that, QBE is performed on the remaining shots to obtain a final retrieval result.

Experimental results validate the effectiveness of all the developed methods. In the future, the current multi-modal video retrieval system will be scaled-up to the internet scale, where the methods are parallelized using thousands of processors. In addition, the system will be extended by adopting another interface, *Query-By-Gesture* (QBG), where the user can create example shots to represent any arbitrary query.

# Acknowledgments

I would like to give my sincere thanks to my advisor Professor Kuniaki Uehara. I've been able to complete this thesis because of his guidance and open mind.

I would also like to express my grateful thanks to Professor Yasuo Ariki and Professor Hisashi Tamaki. They spared their precious time to review this dissertation.

I wish to give my special thanks to Assistant Professor Kazuhiro Seki. He often gave me helpful advance about my research.

I am grateful to Office Administrator Yoko Maruyama. She often took care of my paperwork and loosened up the atmosphere of my laboratory by her vitality.

I would like to thank many collaborators of my research, Yuya Matsuo, Koichi Ideno, Miho Shimizu, Kazuyuki Otaka, Akihito Mizui, Chieri Sugihara, Yuta Matsuoka, Kana Matsumura, Lin Yangpeng, Zhang Yu and Tomomi Komura.

Finally, I would like to express my deepest thanks to my family and friends, especially, my wife, Yuko.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Due to the recent advance of multimedia technologies, a large number of videos are distributed on the Internet or stored in hard disks. As such, there is a great demand to develop a *video retrieval* method which can efficiently retrieve videos of interest. It should be noted that users are usually interested in retrieving videos which match semantic contents, such as "people walk in the street", "an airplane flies" and so on. However, this is challenging because a raw video is just a sequence of video frames and audio samples. In other words, it is not associated with any semantic content. To overcome this, we focus on *data mining* which is a technique to discover previously unknown and interesting patterns in underlying data. Discovering patterns such as customer's purchase for marketing or network access for intrusion detection benefits users in various fields. In this context, we apply data mining to video data in order to extract patterns which characterize semantic contents in videos. Efficient video retrieval can be achieved using extracted patterns. We call this approach *video data mining.*

As the first step of video retrieval, a user issues a query to represent interesting videos. Queries can be classified into three types. For the first type of queries, the user can find keywords suitable for retrieving interesting videos. For the second type of queries, the user cannot find such keywords due to the lexical ambiguity, but can provide some example videos. For the final type of queries, the user has neither keywords nor example videos. Thus, we develop a video retrieval system with 'multi-modal' interfaces by implementing three video data mining methods to support each of the above three query types. For the first query type, the system provides a *Query-By-Keyword* (QBK) interface where patterns which characterize videos relevant to certain key-

words are extracted. For the second query type, a *Query-By-Example* (QBE) interface is provided where videos are retrieved based on their similarities to example videos provided by the user. So, patterns for defining meaningful similarities are extracted using example videos. For the final query type, a *Query-By-Browsing* (QBB) interface is devised to extract patterns which characterize impressive video segments, so that the user can browse videos to find keywords or example videos. Finally, to improve retrieve performance, the integration of QBK and QBE is explored where informations from text and image/video modalities are interchanged using knowledge base which represents relations among semantic contents. Below, we summarize the proposed video data mining methods for QBK, QBE and QBB interfaces, and the integration method of QBK and QBE interfaces.

## 1.1 Summary of the Proposed Video Data Mining Methods

Video data is crucially different from traditional alpha-numeric data. Traditional data is *structured* where its alpha-numeric representation directly describes semantic contents and relationship operators (e.g. equal, not equal, etc.) are well-defined [8]. On the other hand, video data is *unstructured* where its digitized representation (e.g. representation of RGB pixel values) does not directly describe semantic contents and relationship operators are ill-defined. Thus, it is difficult to perform video data mining directly on video data. Instead, we should first derive *features*, such as color, edge, motion and audio, which are related to semantic contents. The derivation of features is one of the most important tasks as it constructs the building blocks for video data mining [50]. Then, video data mining can be achieved by applying data mining techniques to features.

In terms of the kinds of patterns that can be extracted, data mining techniques can be roughly classified into three categories, *pattern discovery*, *classification* and *cluster/structure analysis*. In the following paragraphs, we relate the proposed video data mining methods to the above categories.

### Pattern Discovery

This category aims to extract patterns as high-level descriptions of underlying low-level data. For example, in transaction data, patterns are extracted as

sets of items that are frequently purchased at the same time [80]. An example of a pattern is that 80% of customers who purchase *bread* and *butter* also purchase *milk*. Such patterns are useful for retail marketing, shelf space management, catalog design and so on. In credit card transaction data, a fraud is detected as a pattern which indicates an abnormally high purchase of a customer in comparison to his/her regular purchase [92].

In chapter 2, we develop a pattern discovery method to implement a QBK interface. Specifically, we extract sequential patterns of features which characterize video segments relevant to certain keyword queries. Our method is motivated by the *video editing* process where various shots are concatenated to create a final video sequence. Each shot is a sequence of video frames continuously recorded by a single camera[1]. Considering the video editing process, a meaningful semantic content is not presented by a single shot, but presented by a shot sequence. For example, the conversation between two characters $A$ and $B$ is presented by a shot sequence, where $A$ repeatedly appears in one shot and then $B$ appears in the next shot. Also, the fight between $A$ and $B$ is presented by a shot sequence, where their violent actions are taken by cameras placed at different positions. Hence, we extract sequential patterns, each of which represents features in adjacent shots associated with a certain keyword query.

In chapter 3, we devise another pattern discovery method which is used for a QBB interface. We assume that impressive semantic contents are presented by abnormal editing patterns. For example, thrilling contents are presented by a fast transition of shots with very short duration, so that the thrilling mood is emphasized. In addition, romantic contents are presented by concatenating shots with very long durations, where a character's emotion and action are thoroughly shown. It should be note that only using shot durations is clearly insufficient for characterizing semantic contents. Hence, shot durations where a certain character appears are used as features, and impressive semantic contents are characterized by detecting abnormal patterns of his/her appearance, called *bursts*. Specifically, two types of bursts are detected: the one consists of abnormally short durations of the character's appearance, while the other consists of abnormally long appearance durations. We demonstrate that bursts characterize shot sequences where

---

[1]The video can be accurately divided into shots by using existing methods, where a camera transition is detected as a significant difference of features between two consecutive video frames [100].

the character performs interesting actions, such as fighting, chasing, kissing and so on.

## Classification

This category aims to extract a model (classifier) that classifies data into different classes (categories). The classification technique is used in various problem domains. For example, in Web page classification, a model which classifies Web pages into different subjects such as 'arts', 'business' and 'sport', is extracted to assist the development of Web directories [40]. Also, a model which classifies Electrocardiogram (ECG) records into normal or abnormal, is extracted for monitoring and diagnostic of heart diseases [66].

In chapter 4, a classification method is developed to implement a QBE interface. That is, when a user provides example shots for a query, we extract a retrieval model to classify shots into relevant or irrelevant to a query. Traditional classification tasks only consider a few classes and assume that a large number of training examples are available. On the other hand, in our classification task, the user can issue a great variety of queries. In addition, it is impractical for the user to provide a large number of example shots (training examples). Hence, we develop a method which extracts a retrieval model only using a small number of example shots.

For our method, one of the most important issues is a 'large variation of relevant shots'. Even for the same query, relevant shots contain significantly different features due to varied camera techniques and settings. A small number of example shots inevitably limit the range of relevant shots that can be retrieved. To overcome this, we construct many retrieval models using different sets of randomly selected example shots and feature dimensions. Since these models characterize significantly different shots depending on example shots and feature dimensions, they are useful for extending the range of relevant shots that can retrieved. However, this also results in many irrelevant shots potentially being retrieved. Thus, we use *rough set theory* to combine models into classification rules which provide greater retrieval accuracy.

## Cluster/Structure Analysis

This category aims to extract structures by grouping data into clusters or linking data based on a certain characteristic. For example, in biomedicine,

genes are grouped into clusters of similar gene expression measurements, in order to explore biological relationships among genes [68]. Also, in the field of social network, community structures which represent interpersonal relationships are extracted from e-mail corpora based on the intensity of message exchanges [54]. It should be noted that the cluster/structure analysis is generally used to produce a reduced data representation, while closely maintaining the integrity of the original data. Thus, cluster/structure analysis is often a pre-processing step to discover interesting patterns or to enhance the classification performance.

In chapter 5, to improve the retrieval performance, we develop a method which integrates the QBK and QBE interface based on cluster/structure analysis. We construct a *video ontology* which represents a hierarchical structure of concepts, such as *Person*, *Car*, *Building* and so on. These concepts are fundamental semantic contents in videos. Since features do not directly describe semantic contents, only using features inevitably leads to retrieve several irrelevant shots to a query. Hence, we use the video ontology as knowledge base for video retrieval. The video ontology is constructed by examining the generalization/specialization relation among concepts and the co-appearance of objects corresponding to concepts. Given the textual description of a query (i.e. QBK), the video ontology is used to select concepts related to the query. For example, for the query "buildings are shown", *Building*, *Factory*, *Window*, *Urban* etc. are selected as related concepts. Then, we filter irrelevant shots by referring to recognition results of objects corresponding to selected concepts. Finally, QBE is performed on the remaining shots to obtain a final retrieval result.

## 1.2   Dissertation Overview

The structure of this dissertation is as follows: In chapter 2, we describe a video data mining method for the QBK interface. This method extracts sequential patterns which characterize semantic contents. A video is firstly represented as a multistream by sequentially aggregating several types of features derived from each shot. Then, sequential patterns are extracted by connecting characteristic features in the multistream. In chapter 3, we introduce a method for the QBB interface. This method characterizes impressive semantic contents by detecting bursts as abnormal patterns of a character's appearance. A video is divided into shot sequences which are characterized

by specific patterns of the character's appearance and disappearance. Subsequently, it examines whether a burst occurs in each shot sequence. Chapter 4 describes a method for the QBE interface, where a variety of relevant shots are retrieved only using a small number of example shots provided as a query. We use rough set theory to extract multiple classification rules which can correctly identify different subsets of example shots. A variety of relevant shots can be retrieved by accumulating relevant shots retrieved by each rule. Chapter 5 presents a method which integrates the QBK and QBE interfaces to improve the retrieval performance. A video ontology is constructed to represent the lexical and visual relations among concepts. Given the textual description of a query, we select concepts related to the query based on the video ontology. Then, irrelevant shots are filtered by referring to recognition results of objects corresponding to selected concepts. QBE is performed on the remaining shots to obtain a final retrieval result. In chapter 6, we conclude this dissertation and suggest further research topics in video data mining.

# Chapter 2

# Time-constrained Sequential Pattern Mining for Extracting Semantic Events in Videos

## 2.1 Introduction

In this chapter, we extract *semantic patterns* as sequential patterns of features, which characterize semantically relevant events (*semantic events*) in videos. In general, a video involves two types of media: video and audio. These are known as *continuous media* [26]. As shown in Fig. 2.1, they are sequences of media quanta, i.e., video frames and audio samples, that convey their semantic contents only when continuously played over time. Therefore, the semantic information of the video is time-dependent.

In order to efficiently handle such time-dependent semantic information, it is useful to define two aspects of the semantic information, i.e. *spatial* and *temporal* aspects. A spatial aspect relates to the semantic content presented by a video frame, such as the location, characters and objects shown in the video frame. A temporal aspect relates to the semantic content presented by a sequence of video frames in terms of temporal order, such as a change of locations, character's action and object's movement presented in the sequence.

Since a shot is a segment of video frames recorded continuously by a single camera, it can be considered as a basic physical unit used to successfully capture the spatial and temporal aspects in a video [33]. But, as shown in Fig.
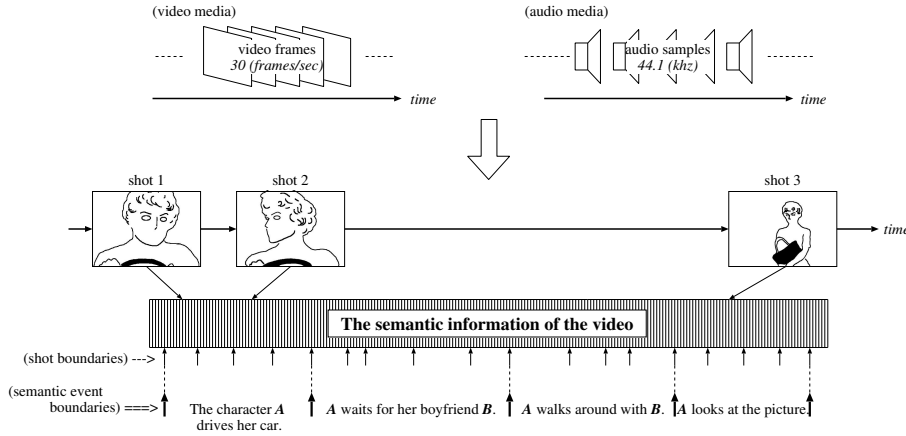
Figure 2.1: An example of the semantic information of a video, conveyed through video and audio media.

2.1, the shot cannot convey a semantic event by itself. Fig. 2.1 illustrates adjacent shots that form a continuous semantic event. For example, the leftmost semantic event consists of four separate shots. From this point of view, we define a semantic pattern as a sequential pattern that sequentially relates to the adjacent shots.

In order to extract the above semantic patterns from a video, we first construct a multistream of features derived from each shot sequentially. Next, we extract sequential patterns from the multistream. In this process, we consider the following two types of temporal characteristics of the video: *semantic event boundaries* and *temporal localities*. A semantic event boundary is defined as the boundary between two consecutive semantic events in which the viewer can recognize an important change in the semantic content, such as change of character's action or change of the location. For instance, in Fig. 2.1, the character $A$ drives her car in the leftmost event and waits for her boyfriend outside her car in the next event. Hence, a semantic event boundary can be found between the fourth and fifth shots from the left.

A temporal locality relates to shots in the same semantic event that are required to be temporally close to each other [103, 97, 98]. For example, in Fig. 2.1, the same character $A$ appears in the three shots, *shot 1*, *shot 2* and *shot 3*. We see that *shot 1* is temporally very close to *shot 2*, while *shot 3* is temporally far from both *shot 1* and *shot 2*. Here, *shot 1* and *shot*

*2* are more likely to be included in the same semantic event. On the other hand, the semantic event of *shot 3* should be different from that of *shot 1* and *shot 2* because several different semantic events may be shown between *shot 2* and *shot 3*. We utilize semantic event boundaries and temporal localities as time constraints to eliminate sequential patterns which are unlikely to be associated with semantic events.

## 2.2   Related Works

In this section, we review previous works in the research fields of video data mining and sequential pattern mining. We also describe our contributions of this chapter to both research fields.

### 2.2.1   Video Data Mining

For efficient video data management, such as video indexing, retrieval and browsing, various video data mining approaches have been proposed in recent years [52, 51, 50, 97, 98, 29, 101]. Pan et al. extracted some patterns of news and commercial video clips for video retrieval and browsing [52]. In particular, they extracted essential characteristics of news and commercial video clips by applying Independent Component Analysis (ICA) to an n-by-n-by-n cube, representing both spatial and temporal video (or audio) information in a video clip.

Pan et al. also extracted some patterns of plot evolution in news and commercial video clips [51]. Given a video clip, they group similar shots into shot-groups based on DCT coefficients of I-frames in a shot. This allows for the detection of basic shot-groups that contain shots that often appear in the video clip. The graph of basic shot-groups reveals the plot structure of the video clip. In [51], it was found that such graphs for news and commercial video clips are quite different from each other.

Oh et al. extracted some patterns of object's appearance and disappearance in a surveillance video [50]. They group incoming frames obtained from a fixed camera into segments of different categories. Furthermore, each segment is indexed by its motion feature which is defined as an accumulated difference of two consecutive frames in the segment. Finally, segments are clustered into groups of similar segments based on segment's category and motion.

Zhu et al. extracted sequential patterns of shots for addressing video semantic units (e.g. events, scenes, and scenario information) in news, medical and basketball videos [97, 98]. Initially, they cluster shots into groups of visually similar shots and construct a sequence consisting of the group names. Sequential patterns of shots with strong temporal correlations are then extracted from the sequence.

The above video data mining approaches extract semantic patterns only from *rule-dependent* videos, which have apparent rules associated with semantic events. For example, in the interview events in news videos, a shot where an interviewer appears is followed by a shot where an interviewee appears, and these are repeated one after the other [52]. Similarly, in goal events in ball game videos, the score shown on the telop changes after audience's cheering and applause [98]. In surveillance videos recorded with fixed cameras, if an object actively moves, the difference between two consecutive frames is clearly large [50]. In this way, these apparent rules indicate what kind of features should be used to extract semantic patterns in *rule-dependent* videos. Thus, the extracted semantic patterns are previously known.

In contrast to the above *rule-dependent* videos, we extract semantic patterns from *rule-independent* videos, such as movies, where there is no apparent rule which characterizes any kind of semantic events. For example, battle events in movies are presented in various ways depending on semantically arbitrary factors, such as characters, weapons, location, time and weather. So, neither what kind of features should be used nor what kind of semantic patterns are extracted can be predicted. Hence, we use several types of features which have been accepted as useful in the fields of image, video and audio processing researches. By examining numerous combinations of these features, we aim to extract previously unknown semantic patterns from *rule-independent* videos.

Some studies have also considered video data mining on *rule-independent* movies [29, 101, 62]. Wijesekera et al. proposed a video data mining framework on movies by using existing audio and video analysis techniques [29]. They also examined the suitability of applying existing both data mining concepts and algorithms to multimedia data, although no experimental result was reported. In [101], we extracted two types of editing patterns from movies: *cinematic rules* and *frequent events*. But, cinematic rules are not semantic patterns because they are just editing patterns for successfully conveying editor's idea to the viewers, and do not characterize any kind of semantic event. Frequent events are semantic patterns, but they are previously

known. The reason is that features used to extract these patterns (e.g. *Character's name*, *Direction of character's gaze*, *Character's motion* and *Sound type*) represent considerably high semantic contents by themselves.

In [62], we extracted a character's appearance and disappearance patterns for characterizing topics in a movie. Each topic corresponds to a semantic event where the character plays a particular action and role, for instance, he/she may talk to someone or make love to another character. This kind of topic can be detected as an interval where durations of the character's appearance and disappearance are roughly constant. However, semantic content in the topic, such as what action the character performs and what kind of situation the topic involves, cannot be identified by using only his/her appearance and disappearance.

## 2.2.2 Sequential Pattern Mining

Extracting sequential patterns from categorical streams is a research area of great interest in data mining. A categorical stream is defined as a sequence on a set of finite types of symbols. This task is challenging as a search space of possible sequential patterns is extremely large. As described in [75], even in the case of a one-dimensional stream defined on a set of $n$ types of symbols, there are $O(n^k)$ possible sequential patterns of time length $k$. In order to efficiently extract sequential patterns from categorical streams, many methods have been proposed [101, 81, 85, 39, 91, 32, 43, 56, 75, 25, 19, 104, 97, 98]. As shown in Table 2.1, these methods are classified into six categories, in terms of the number of dimensions of a categorical stream and a search technique for reducing the extremely large search space.

Table 2.1: The classification of sequential pattern mining methods in terms of the number of dimensions of a categorical stream and a search technique.

|  | one-dimension | multi-dimensions |
|---|---|---|
| window-based | [43] [56] [25] | [39] [91] [75] |
| apriori-based | [97] [98] | [81] [85] [32] [101] |
| two-step | [19] [104] | |

The number of dimensions of a categorical stream greatly affects the efficiency of the sequential pattern extraction. As a simple example, consider $m$-dimensional multistream where each component stream contains $n$ types of

symbols. In this multistream, there are $O(n^{mk})$ possible sequential patterns of time length $k$. Thus, extracting sequential patterns from a multistream requires much more expensive search than that of a one-dimensional stream. With respect to this, Tanaka et al. transformed an original multistream into a one-dimensional stream by using Principle Component Analysis (PCA) [104]. This kind of transformation approach was also proposed by Zhu et al. where they used a re-arrangement mechanism to maintain the original temporal order [98].

Search techniques can be classified into the following three types: *window-based*, *apriori-based* and *two-step* approaches. A window-based approach scans a given stream by sliding the window of a user-specified time length [88]. Sequential patterns are extracted as sets of symbols within the window, which are validated by the sliding window scan. The window-based approach limits a search space of possible sequential patterns as the time length of any extracted sequential pattern is up to window's length. The user needs to specify the maximum time length of extracted sequential patterns in advance.

Chudova et al. extracted fixed-length sequential patterns by using a Hidden Markov Model (HMM) [25]. The HMM has $k$ states for modeling symbols included in a pattern of time length $k$ and one state for modeling symbols which are not in the pattern. By learning the parameters of the HMM, the symbol which is most likely to appear in the $i$-th position in the pattern is determined. Ultimately, this and the window-based approach rely heavily on a priori knowledge of extracted sequential patterns. Thus, both of them may be more generally called *model-based* approaches.

An apriori-based approach iteratively extracts longer sequential patterns from shorter ones. Each iteration starts with a generation of *candidate patterns*, each of which is generated by concatenating a symbol or set of symbols to a pattern extracted in the previous iteration. This is based on the principle that if a sequence is not extracted as a pattern, no patterns can be extracted by adding symbols to this sequence. Candidate patterns are examined whether they are actually sequential patterns or not. This iteration terminates when no more sequential pattern is extracted. In this way, the apriori-based approach efficiently and dynamically reduces the search space of possible sequential patterns in the given stream.

A two-step approach consists of the following two steps. In the first step, some criteria are used to obtain the optimum time length of sequential patterns to be extracted. For example, Tanaka et al. computed such an optimum time length by using Minimum Description Length (MDL) princi-

ple [104]. Berberids et al. computed the optimum time length by using the autocorrelation function [19]. The optimum time length obtained in the first step is used to reduce the search space of possible time lengths of sequential patterns in the second step. Thus, the two-step approach can be considered as an extended window-based approach. Specifically, the time length of extracted sequential patterns is automatically obtained in the two-step approach in contrast to being user-specified in a window-based approach. As shown in Table 2.1, we could not find any two-steps approach for extracting sequential patterns from a multi-dimensional categorical stream.

Our proposed method extracts sequential patterns from a multi-dimensional categorical stream using an apriori-based approach. Hence, our method is classified into the same category as [81] [85] [32] [101] in Table 2.1. In order to eliminate sequential patterns which are unlikely to be semantic patterns, we incorporate two types of a priori information specific to video data: *semantic event boundaries* and *temporal localities*. Additionally, there are many possibilities to locate a sequential pattern in the stream. Thus, how to locate the sequential pattern in the stream is a very important issue. But, it has never been discussed in the previous works listed in Table 2.1. We propose a method for finding the location of the sequential pattern in the stream. Finally, we also propose a method for parallelizing the process of our mining method as it requires multiple scans over the stream to locate each candidate pattern.

## 2.3   Features

In this section, we provide a detailed explanation of features used to characterize the spatial and temporal aspects of a shot. It should be noted that each type of features is a categorical value. Consequently, deriving several types of features from the shot can be considered as a discretization of the spatially and temporally continuous semantic content into a set of categorical values. By sequentially aggregating features, we can obtain a multistream of features as shown in Fig. 2.2.

Since the spatial aspects presented by the video frames are continuous, we assume that the salient spatial aspect is represented by the middle video frame in a shot, denoted *keyframe*. Note that it is possible to change the definition of a keyframe [52, 103, 97]. We derive the following types of features from the keyframe:

| Shot No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| stream 1 CH: | CH4 | CH7 | CH3 | CH2 | CH7 | CH6 | CH6 | CH6 | CH6 | CH6 | CH6 | CH6 | CH2 | CH2 |
| stream 2 CS: | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS4 | CS0 |
| stream 3 CV: | CV2 | CV2 | CV3 | CV0 | CV1 | CV0 | CV0 | CV0 | CV0 | CV0 | CV0 | CV0 | CV0 | CV1 |
| stream 4 LN: | LN3 | LN3 | LN2 | LN1 | LN1 | LN2 | LN3 | LN1 | LN3 | LN3 | LN1 | LN2 | LN1 | LN1 |
| stream 5 LL: | LL0 | LL3 | LL1 | LL0 | LL1 | LL1 | LL3 | LL0 | LL0 | LL1 | LL0 | LL0 | LL0 | LL0 |
| stream 6 LB: | LB2 | LB0 | LB3 | LB2 | LB2 | LB0 | LB1 | LB0 | LB1 | LB1 | LB1 | LB3 | LB0 | LB2 |
| stream 7 SA: | SA2 | SA2 | SA1 | SA1 | SA2 | SA2 | SA2 | SA2 | SA2 | SA2 | SA2 | SA2 | SA1 | SA2 |
| stream 8 LA: | LA0 | LA2 | LA1 | LA0 | LA0 | LA0 | LA1 | LA0 | LA0 | LA0 | LA0 | LA0 | LA0 | LA1 |
| stream 9 SL: | SL4 | SL4 | SL3 | SL4 | SL4 | SL1 | SL1 | SL1 | SL1 | SL2 | SL1 | SL0 | SL0 | SL0 |
| stream 10 MS: | MS3 | MS3 | MS2 | MS4 | MS5 | MS4 | MS4 | MS4 | MS4 | MS4 | MS3 | MS3 | MS0 | MS0 |
| stream 11 MV: | MV1 | MV4 | MV0 | MV1 | MV0 | MV0 | MV1 | MV1 | MV0 | MV0 | MV0 | MV0 | MV3 | MV2 |
| stream 12 SM: | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM1 | SM2 | SM0 | SM0 |
| stream 13 AM: | AM1 | AM3 | AM2 | AM3 | AM3 | AM2 | AM3 | AM3 | AM2 | AM2 | AM2 | AM2 | AM2 | AM2 |

→ *Time*

Figure 2.2: An example of a multistream of features, where several types of features are derived from each shot in a video.

**CH:** *CH* reflects the semantic content of the background or dominant object in a keyframe, such as water, sky, snow, fire or human face [13]. *CH* represents the color composition in the keyframe on the H (Hue) axis of the HSV color space. We first compute the intensity histogram on the H axis for each keyframe. We then cluster keyframes into groups with similar histograms, where we use the *k*-means clustering algorithm [10] and the histogram intersection as a distance measure [53]. Finally, we assign the categorical value of *CH* to each keyframe by analyzing the cluster including the keyframe.

**CS:** Similarly to *CH*, *CS* reflects the semantic content of the background or dominant object in a keyframe. But, *CS* characterizes a keyframe which contains objects with saturated colors, such as fruits, flowers or man-made objects. *CS* represents the color composition in the keyframe on the S (Saturation) axis of the HSV color space. We assign the categorical value of *CS* to each keyframe by using the *k*-means clustering method.

**CV:** Unlike *CH* and *CS*, *CV* reflects the semantic content of the brightness in a keyframe. *CV* represents the color composition in the keyframe on the V (Value) axis of the HSV color space. Categorical values of *CV* are also assigned by the *k*-means clustering method.

**LN:** *LN* basically reflects the number of objects displayed in a keyframe. This means that as more objects are displayed in the keyframe, more straight

lines tend to be derived. Additionally, in a keyframe showing a night or foggy situation, objects' boundaries are obscure and few straight lines are derived. $LN$ represents the number of straight lines contained in the keyframe. We assign the categorical value of $LN$ to each keyframe by comparing the number of straight lines in the keyframe with some threshold values.

**LL:** $LL$ reflects shape features of man-made objects in a keyframe. For instance, buildings and windows have long straight lines which define these objects' boundaries. To be precise, $LL$ represents the distribution of straight line lengths in the keyframe. In order to assign such a categorical value of $LL$ to each keyframe, we compute the histogram of the lengths of straight lines. This histogram is then normalized so that it is independent of the number of straight lines. Finally, we assign the categorical value of $LL$ to each keyframe by using the $k$-means clustering method.

**LB:** $LB$ reflects the dominant direction of most straight lines contained in a keyframe. For example, buildings and windows have vertical straight lines, while a natural scene has different directions of straight lines [13]. Thus, $LB$ represents the distribution of straight line directions in the keyframe. As with $LL$, we compute the normalized histogram of straight line directions. We then assign the categorical value of $LB$ to the keyframe by using the $k$-means clustering method.

**SA:** $SA$ reflects the size of the main character displayed in a keyframe by representing the area of the largest skin colored region. We assign the categorical value of $SA$ to each keyframe by comparing the area of the largest skin colored region with some threshold values.

**LA:** $LA$ reflects the presence of weapons in a keyframe. It shoud be noted that keyframes where laser-beams from weapons are presented have some large light colored regions. On the other hand, keyframes where sunshines or lighting effects are presented, not only have some large light colored regions, but also have many small light colored blobs due to the light dispersions. Based on this observation, $LA$ represents the area of the largest light colored region, divided by the total number of light colored regions in a keyframe. By comparing this value with some threshold values, we assign the categorical value of $LA$ to the keyframe.

The above types of features are derived by using functions prepared in OpenCV library [5]. Apart from these types of features for characterizing the spatial aspect, we derive the following features for characterizing the temporal aspect.

**SL:** $SL$ represents the duration of a shot. In other words, $SL$ represents

the speed of the camera switching from a shot to the next shot. Typically, thrilling events, such as battles and chases, are presented by shots with short durations, while romantic events, such as hugging and kissing, are presented by shots with long durations. We assign the categorical value of $SL$ to each shot by comparing its duration with some threshold values.

**MS:** $MS$ represents the movement of some objects or the background in a shot. For example, in a shot where characters actively move or the background significantly changes, the movement is large, whereas the movement is small when characters are still and the background remains stable. To extract the movement, we select MPEG as the video format because MPEG compresses a video by predicting the color change between two consecutive video frames. The size and direction of the predicted color change are represented as a *motion vector*. Each motion vector is defined in a *macro block* which is a unit block consisting of $16 \times 16$ pixels. Based on this definition of motion vector, the movement is defined as a sum of motion vector sizes in all macro blocks. We assign the categorical value of $MS$ by comparing the movement with some threshold values.

**MV:** $MV$ represents the direction of movement of objects or the background in a shot. For example, $MV$ characterizes a direction of character's movement in a shot. It also characterizes no direction of movement in a shot where characters are still. The direction of movement is defined in the following way. Four direction counters are prepared: up, down, left and right. These are used to count the directions of motion vectors in all macro blocks. If the count in every direction is smaller than the threshold value, no direction is assigned to the shot. Otherwise, the direction with the largest count is assigned to the shot.

**SM:** $SM$ represents the sound type which frequently appears in a shot, such as *speech*, *music* or *no-sound*. For example, *speech* frequently appears in a shot where characters are talking. On the other hand, *music* frequently appears in a shot where BGM (BackGround Music) with large sound volume is used. We convert the sound stream into Mel-Frequency Cepstrum Coefficients (MFCCs). These are compared with a human voice model and music model constructed by using Gaussian Mixture Model (GMM) [65]. As a result, we can assign a categorical value of $SM$ to the shot.

**AM:** $AM$ represents the largest sound volume in a shot. For example, a shot which involves a scream, explosion or gunshot has a large sound volume, while a shot with a chat or spying activity has a small sound volume. We assign the categorical value of $AM$ to each shot by comparing the maximum

amplitude of the sound stream with some threshold values.

Finally, by deriving features from each shot, the video is transformed from a computationally-intractable raw material into a 13-dimensional categorical stream. An example of this categorical stream is presented in Fig. 2.2, where each component stream is constructed for one type of features. A symbol in the stream consists of two capital letters representing the type of features and the number representing the categorical value. For example, stream 1 is constructed for $CH$ and the symbol $CH4$ in *shot 1* represents that the categorical value 4 is assigned to *shot 1*.

## 2.4 Time-constrained Sequential Pattern Mining

In this section, we present our time-constrained sequential pattern mining method to extract sequential patterns from a multi-dimensional categorical stream. Firstly, we formally define sequential patterns together with time constraints. We then present our mining algorithm with time constraints. Finally, we extend our algorithm for parallelizing the mining process.

### 2.4.1 Formulation

We assume a multi-dimensional categorical stream $S$, where no symbol occurs in more than one component stream of $S$. Thus, a symbol $v$ that occurs in a component stream $s$ at a time point $t$ can be represented as $(v, t)$, because $v$ does not occur in the other component streams. An example of such a multi-dimensional categorical stream is shown in Fig. 2.3 where the capital letter on the left indicates the component stream name and the number on the right indicates the categorical value. For example, the symbol $A2$ occurring in *streamA* at the time point 1 is represented as $(A2, 1)$.

For any pair of two symbols in $S$, the relative temporal relationship is either *serial* or *parallel* [1]. For example, the relationship between $(A2, 1)$ and $(C3, 1)$ is parallel as these symbols occur at the same time point. On the other hand, the relationship between $(C3, 1)$ and $(C4, 2)$ is serial as these symbols occur at different time points. It should be noted that a serial relationship does not require two symbols to belong to the same component

---

[1]Our usage of serial and parallel relationships is different from that of [39].

| time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | ----- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| stream A: | A2 | A7 | A3 | A2 | A2 | A2 | A6 | A6 | A2 | A4 | A2 | A5 | A1 | A3 | A6 | ----- |
| stream B: | B4 | B4 | B3 | B3 | B4 | B4 | B4 | B3 | B4 | B4 | B4 | B4 | B2 | B4 | B4 | ----- |
| stream C: | C3 | C4 | C3 | C0 | C3 | C0 | C4 | C0 | C3 | C2 | C4 | C0 | C1 | C1 | C0 | ----- |
| stream D: | D3 | D3 | D2 | D1 | D1 | D2 | D2 | D2 | D0 | D1 | D2 | D1 | D2 | D3 | D2 | ----- |
| stream E: | E0 | E3 | E0 | E1 | E0 | E2 | E2 | E2 | E1 | E1 | E1 | E3 | E2 | E1 | E2 | ----- |

SEB1 is above time points 6. SEB2 is above time point 15.
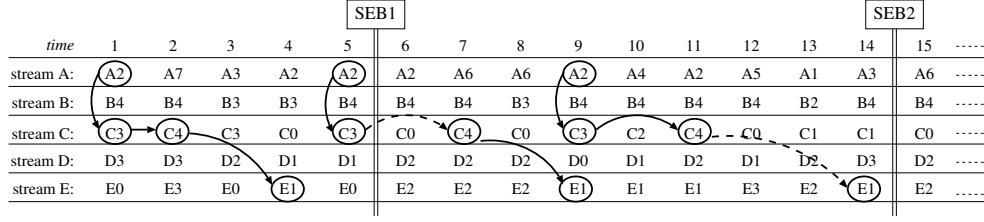
Figure 2.3: An example of a multi-dimensional categorical stream $S$ where only the occurrence of a 4-pattern $p_4 = (A2, nil), (C3, parallel), (C4, serial), (E1, serial)$ from the time point $t = 1$ to $t = 4$ satisfies both $SEB$ and $TDT$ time constraints.

stream, and thus the relationship between $(C4, 2)$ and $(E1, 4)$ is also serial. For two symbols $(v_1, t_1)$ and $(v_2, t_2)$, the serial and parallel relationships are formulated as follows:

$$t_1 \neq t_2 \longrightarrow serial,$$
$$t_1 = t_2 \longrightarrow parallel. \tag{2.1}$$

We define a sequential pattern $p_l$ as a temporally ordered set of $l$ symbols and refer to $p_l$ as *l-pattern*. In Fig. 2.3, the 4-pattern $p_4$ is presented by the temporally ordered set of 4 symbols surrounded by circles. For $p_l$, we represent the temporal order of $l$ symbols as a sequence of serial and parallel relationships between two consecutive symbols. Therefore, $p_l$ is formulated as follows:

$$p_l = (v_1, nil), (v_2, tr_2), (v_3, tr_3), \cdots, (v_l, tr_l), \tag{2.2}$$

where for all $i = 2, \cdots, l$, $(v_i, tr_i)$ represents a symbol $v_i$ whose relationship with $(v_{i-1}, tr_{i-1})$ is $tr_i$ (i.e. $tr_i = serial$ or $tr_i = parallel$). In Fig. 2.3, $p_4$ is denoted as $p_4 = (A2, nil), (C3, parallel), (C4, serial), (E1, serial)$. In equation (2.2), if $tr_i = serial$, the serial relationship between $(v_i, tr_i = serial)$ and $(v_{i-1}, tr_{i-1})$ is restricted by the following two types of time constraints:
**Semantic event boundaries:** A multi-dimensional categorical stream $S$ can be divided into some semantic events. For example, a stream of highway traffic sensor data can be divided into semantic events of different traffic conditions. An e-mail stream can be divided into semantic events where certain topics appear [45]. By restricting an occurrence of $p_l$ in one semantic event, $p_l$ becomes a useful sequential pattern associated with a certain semantic

content in $S$. Thus, the serial relationship between $(v_{i-1}, tr_{i-1})$ and $(v_i, tr_i)$ must not intersect any *Semantic Event Boundaries (SEBs)*. In Fig. 2.3, two $SEB$s ($SEB1$ and $SEB2$) are shown and it is not acceptable that $p_4$ occurs between the time point $t = 5$ and $t = 9$ as the serial relationship between $(C3, 5)$ and $(C4, 7)$ crosses over $SEB1$.

**Temporal localities:** Using only $SEB$ time constraint leads to the extraction of many sequential patterns which are not relevant to semantic patterns. Thus, we use *temporal localities* proposed in [97, 98]. A temporal locality means that two semantically related shots have to be temporally close to each other. By applying this to $p_l$, two consecutive symbols must be temporally close to each other [2]. Hence, in order for $p_l$ to be a semantic pattern, the relative temporal distance between $(v_{i-1}, tr_{i-1})$ and $(v_i, tr_i)$ must be less than *Temporal Distance Threshold* ($TDT$). In Fig. 2.3 where $TDT$ is set to 2, it is not acceptable that $p_4$ occurs between the time point $t = 9$ and $t = 14$ because the temporal distance between $(C4, 11)$ and $(E1, 14)$ is 3.

By using $SEB$ and $TDT$ time constraints, $p_l$'s occurrences which are semantically irrelevant, are not counted. Thereby, we can avoid extracting unnecessary sequential patterns which are unlikely to be semantic patterns. In Fig. 2.3, only the occurrence of $p_4$ between $t = 1$ and $t = 4$ satisfies both $SEB$ and $TDT$ time constraints. It should be noted that $p_l$ is just a template used to specify the temporal order of $l$ symbols. In contrast, an occurrence of $p_l$ is an actual instance where each symbol in $p_l$ is detected as $(v_i, t_i)$ in $S$.

## 2.4.2 Mining Algorithm

As described in section 2.2.2, our mining algorithm extracts sequential patterns from a multi-dimensional categorical stream $S$ by using an apriori-based approach. It is outlined below:

**Process 1:** Initialize $l = 1$ where $l$ is the number of symbols included in a pattern. Subsequently, from $S$, extract every 1-pattern $p_1$ which satisfies an interestingness measure $f$. As will be described in section 2.4.2, $f$ is used to measure the usefulness of each candidate pattern in order to determine whether it is regarded as a pattern or not.

**Process 2:** Increment $l$, and generate a set of *candidate l-patterns* from the

---

[2]Apart from video data, this time constraint has been applied to various kinds of data such as transactional data [85] and biological data [77]

set of $(l-1)$-patterns.

**Process 3:** Locate each candidate $l$-pattern $cp_l$ in $S$ by considering $SEB$ and $TDT$ time constraints and counting the number of $cp_l$'s occurrences in $S$. Then, regard $cp_l$ as an $l$-pattern $p_l$ only if $cp_l$ satisfies $f$.

**Process 4:** Terminates the mining process if no $p_l$ is extracted. Otherwise, go to Process 2.

In order to complete our mining algorithm, we need to discuss the following three issues in greater detail: how to generate candidate patterns at Process 2, how to locate $cp_l$ in $S$ at Process 3, and how to define $f$ in Processes 1 and 3.

### Generating Candidate Patterns

An efficient algorithm of candidate pattern generation is presented in [81], although we use a different definition of a sequential pattern. Hence, we revise the algorithm in [81] to generate a set of candidate $l$-patterns from the set of $(l-1)$-patterns extracted in the previous iteration. Fig. 2.4 illustrates the generation of a candidate $l$-pattern $cp_l$ from two $(l-1)$-patterns $p_{l-1}$ and $p'_{l-1}$. In Fig. 2.4, we select the following $p_{l-1}$ and $p'_{l-1}$: the temporally ordered set of $l-2$ symbols generated by removing the first symbol $(v_1, nil)$ from $p_{l-1}$, is the same to the one generated by removing the last symbol $(v'_{l-1}, tr'_{l-1})$ from $p'_{l-1}$. Note that the symbol $(v_2, tr_2)$ is replaced with $(v_2, nil)$, as it is now the starting symbol in the temporally ordered set of $l-2$ symbols. Subsequently, $cp_l$ is generated by concatenating $(v'_{l-1}, tr'_{l-1})$ and $p_{l-1}$.
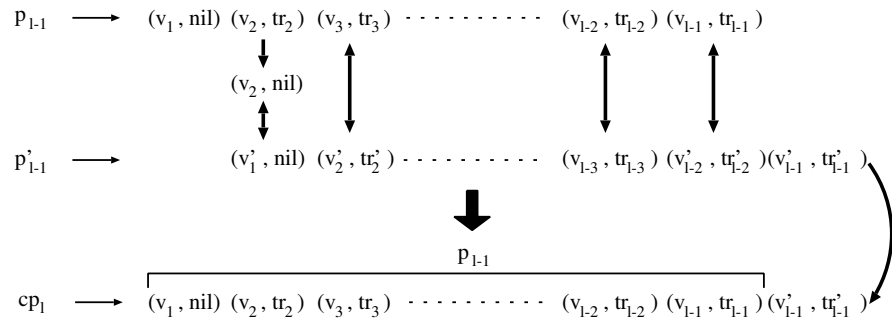


Figure 2.4: An illustration of the generation of a candidate $l$-pattern $cp_l$ from two $(l-1)$-patterns, $p_{l-1}$ and $p'_{l-1}$.

Locating all candidate $l$-patterns in $S$ requires a very expensive computational cost. Hence, we should delete candidate $l$-patterns which are unlikely to become $l$-patterns without searching for them in $S$. For $cp_l$, we remove any $(v_i, parallel)$ from $cp_l$ to form a temporally ordered set of $l-1$ symbols. Note that even if we delete $(v_i, parallel)$ from $cp_l$, the original temporal order of $cp_l$ is preserved. Therefore, all of the above temporally ordered sets of $l-1$ symbols have to be previously extracted as $(l-1)$-patterns. Otherwise, it is impossible for $cp_l$ to be $p_l$ and we thus delete $cp_l$ from the set of candidate $l$-patterns.

### Locating a Candidate Pattern in the stream

The outline of our approach for locating $p_l$ in $S$ is illustrated in Fig. 2.5. In the stream $A$, a symbol $(A1, t)$ $(t = 1, 2, 3, 5, 6, 8)$ represents 'the woman appears at a time point $t$ (i.e. in a shot $t$)', and a symbol $(A2, t)$ $(t = 4, 7)$ represents 'the woman does not appear at $t$'. In this condition, we locate 2-pattern $p_2 = (A1, nil), (A1, serial)$ by focusing on the three occurrences: $Occ_1$, $Occ_2$ and $Occ_3$. The semantic contents represented by these occurrences are as follows:

- $Occ_1 \cdots$ The woman meets the man and talks to him.

- $Occ_2 \cdots$ The woman meets the man and walks with him.

- $Occ_3 \cdots$ The woman meets the man and drives her car.

$Occ_3$ is clearly meaningless as it spans two different semantic events. Such an occurrence can be prevented by $SEB$ time constraint. $Occ_1$ is assumed to be more meaningful than $Occ_2$. This is due to the discussion of temporal localities in section 3.1, which noted that a serial relationship occurring in a short temporal distance between two symbols is assumed to represent a coherent semantic content. Thus, if there are some possible occurrences of the serial relationship within the temporal distance specified by $TDT$ time constraint, the serial relationship is located by using the shortest temporal distance, such as $Occ_1$ in Fig. 2.5.

Fig. 2.6 illustrates our approach for locating 3-pattern $p_3$ in $S$. In Fig. 2.6 where $TDT$ is set to 3, tracing along the solid arrows provides three occurrences of $p_3$. In addition to $SEB$ and $TDT$ time constraints, we further introduce the search constraint represented by three dashed arrows, such
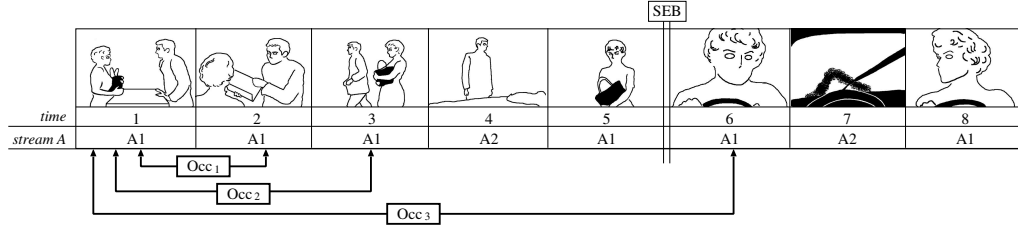
Figure 2.5: An example of occurrences of 2-pattern $p_2$ = $(A1, nil), (A1, serial)$, where $Occ_1$ represents the most coherent semantic content among three occurrences.

that, if $(v, t)$ is used to find an occurrence of $p_l$, $(v, t)$ cannot be used to find any later occurrences of $p_l$. For example, the leftmost dashed arrow represents that $(B1, 3)$ is used to find $p_3$'s occurrence starting from $t = 1$, thus it cannot be used again. So, $(B1, 4)$ is used to find $p_3$'s occurrence starting from $t = 2$. We regard a symbol $(v, t)$ as *unused* if it has not yet been used to find any $p_l$'s occurrence. Suppose that in Fig. 2.6, $(B1, 3)$ is used by $(A3, 1)$ and $(A3, 2)$ to find $p_3$'s occurrences. Consequently, at the time point $t = 5$, two $p_3$'s occurrences are counted, which is contradictory to that two different occurrences end at different time points. Hence, we only use *unused* symbols to detect $p_3$'s occurrences.
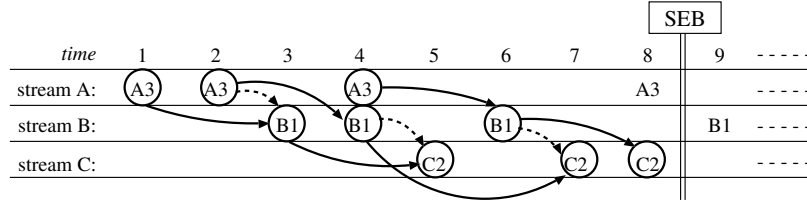


Figure 2.6: An example of our approach for locating $p_3$ = $(A3, nil), (B1, serial), (C2, serial)$ in $S$, where $TDT = 3$ and three occurrences of $p_3$ are located.

The following paragraphes describe our algorithm for locating $p_l$ in $S$. Our algorithm finds $p_l$'s occurrences one by one according to the temporal order. For example, in Fig. 2.6, the occurrence of $p_3$ starting from $t = 1$ is first detected, the occurrence from $t = 2$ is detected second and the

occurrence from $t = 4$ is detected third. In each detection of $p_l$'s occurrence, our algorithm switches *Forward* and *Backward* phases introduced in [85]. However, in order to deal with $SEB$ and $TDT$ time constraints and the above search constraint, we extend *Forward* and *Backward* phases. Suppose that we are now searching an occurrence of $p_l$ where the first $i$ symbols have already been detected as $(v_1, t_1), \cdots, (v_i, t_i)$.

**Forward phase:** Based on the already detected symbols, we select a symbol $(v_{i+1}, t_{i+1})$ for the $(i + 1)$-th symbol.

> **If the $(i + 1)$-th temporal relationship is serial,** we select the un-used symbol $(v_{i+1}, t_{i+1})$ where $t_{i+1}$ is larger than $t_i$. If the serial relationship between $(v_i, t_i)$ and $(v_{i+1}, t_{i+1})$ satisfies both $SEB$ and $TDT$ time constraints, we recursively perform *Forward* phase for selecting $(v_{i+2}, t_{i+2})$, otherwise, switch to *Backward* phase for backtracking.
>
> **If the $(i + 1)$-th temporal relationship is parallel,** we select the unused symbol $(v_{i+1}, t_{i+1})$ where $t_{i+1}$ is greater than or equal to $t_i$. If $t_i = t_{i+1}$, we recursively perform *Forward* phase, otherwise (i.e. $t_i \neq t_{i+1}$), switch to *Backward* phase for backtracking.
>
> In the case of $i + 1 = l$, if $(v_i, t_i)$ and $(v_{i+1}, t_{i+1})$ satisfy the serial (or parallel) relationship, one occurrence of $p_l$ is detected.

**Backward phase:** Suppose that two already selected symbols $(v_i, t_i)$ and $(v_{i+1}, t_{i+1})$ do not satisfy the $(i + 1)$-th temporal relationship. Hence, we select $(v_i, t'_i)$ as an alternative of $(v_i, t_i)$.

> **If the $(i + 1)$-th temporal relationship is serial,** we select the un-used symbol $(v_i, t'_i)$ where $t'_i$ is not only larger than the nearest $SEB$ before $t_{i+1}$, but also larger than $t_{i+1} - TDT$.
>
> **If the $(i + 1)$-th temporal relationship is parallel,** we select the unused symbol $(v_i, t'_i)$ where $t'_i$ is greater than or equal to $t_{i+1}$.

After modifying $(v_i, t_i)$ into $(v_i, t'_i)$, check whether $(v_i, t'_i)$ and $(v_{i-1}, t_{i-1})$ still satisfy the $i$-th temporal relationship in $p_l$. If so, switch to *Forward* phase in order to select a new $(v_{i+1}, t_{i+1})$ for the modified $(v_i, t'_i)$. Otherwise, recursively starts *Backward* phase for further backtracking. In the case of $i = 1$ where $(v_1, t_1)$ is modified into $(v_1, t'_1)$, immediately switch to *Forward* phase.

In both *Forward* and *Backward* phases, if no symbol can be selected, there is no more occurrence of $p_l$ in $S$ and our algorithm terminates. Finally, the number of $p_l$'s occurrences is defined as the frequency of $p_l$, and denoted as $freq(p_l)$.

**Using an Interestingness Measure**

Based on $cp_l$ $freq(cp_l)$, we determine whether $cp_l$ is actually an $l$-pattern $p_l$. At this point, many interestingness measures have been proposed to measure the usefulness of $cp_l$ [83]. In this chapter, we use the *support* and *confidence* of $cp_l$ as an interestingness measure $f$. Recall that $cp_l = (v_1, nil), \cdots, (v_{l-1}, tr_{l-1}), (v_l, tr_l)$ is generated by adding $(v_l, tr_l)$ to $(l-1)$-pattern $p_{l-1} = (v_1, nil), \cdots, (v_{l-1}, tr_{l-1})$. So, the first $(l-1)$ symbols in $cp_l$ denoted by $pre(cp_l) = (v_1, nil), \cdots, (v_{l-1}, tr_{l-1})$ are already validated as $p_{l-1}$. Therefore, based on the usefulness of connecting $pre(cp_l)$ and $(v_l, tr_l)$ measured by $f$, we determine whether $cp_l$ can be $p_l$ or not.

We denote the support and confidence of $cp_l$ by $sup(cp_l)$ and $conf(cp_l)$, respectively. They are defined as follows:

$$\begin{aligned} for\ cp_l\ (l \geq 1), \quad & sup(cp_l) = freq(cp_l), \\ for\ cp_l\ (l \geq 2), \quad & conf(cp_l) = P(cp_l \mid pre(cp_l)). \end{aligned} \tag{2.3}$$

In the above equation (2.3), $sup(cp_l)$ is the frequency of $cp_l$ in $S$, and represents the statistical significance of $cp_l$ in $S$. $conf(cp_l)$ is the conditional probability of $cp_l$ given that $pre(cp_l)$ occurs, and represents the strength of the association between $pre(cp_l)$ and $(v_l, tr_l)$. We then regard $cp_l$ as an $l$-pattern $p_l$ only when both $sup(cp_l)$ and $conf(cp_l)$ are larger than the minimum support and confidence thresholds.

## 2.4.3   Parallel Algorithm

In order to reduce the computational cost of our mining algorithm, we propose a parallel algorithm for parallelizing the mining process. Our parallel algorithm assumes a shared-nothing architecture where each of $p$ processors has a private memory and disk. These $p$ processors are connected to a communication network and can communicate only by passing messages. The communication primitives are prepared using MPI (Message Passing Interface) communication library [3].

The outline of our parallel algorithm is as follows: we begin with the extraction of 1-patterns by using a single processor. Suppose that the total number of 1-patterns is $k$. These $k$ 1-patterns are evenly distributed to $p$ processors. In other words, $k/p$ 1-patterns are distributed to each processor. Subsequently, by using our mining algorithm described in the previous section, each processor performs the extraction of sequential patterns where distributed $k/p$ 1-patterns are used as the initial symbols of these sequential patterns. After all $p$ processors have finished extracting sequential patterns, all of extracted sequential patterns are gathered in one processor, and output as the final mining result.

The above algorithm cannot be considered as an efficient parallel algorithm, because the number of sequential patterns extracted at one processor may be much fewer than those of the other processors. In this case, the processor has a long *idle time* in which it waits for the other processors to finish their mining tasks. Thus, we need to reduce such idle times on $p$ processors used in our parallel algorithm.

This is addressed by using a *load balancing* technique where mining tasks of $p$ processors are dynamically re-distributed. Specifically, when a processor $PE_i$ has an idle time, it sends *task requests* to the other processors. If a processor receives the task request from $PE_i$, the processor reports the current progress of its mining task to $PE_i$, allowing the selection of a *donor processor* which requires the longest time to finish its mining task. Consequently, a part of the mining task of the donor processor is re-distributed to $PE_i$.

Suppose that a processor $PE_j(j = 1, 2, \cdots, p \mid j \neq i)$ receives a task request from $PE_i$ after extracting $l_j$-patterns. Additionally, the number of different starting symbols in $l_j$-patterns is $m_j$, and the total number of sequential patterns extracted at $PE_j$ is $n_j$. After $PE_i$ receives $l_j$, $m_j$ and $n_j$ from another processor $PE_j$, $PE_i$ determines a donor processor in the following way. First, $PE_j$ with $m_j = 1$ cannot be a donor processor as it has no distributable task. Among the remaining processors, $PE_i$ selects $PE_j$ with the smallest $l_j$ as a donor processor. This is based on the assumption that, since $l_j$ represents the number of iterations which $PE_j$ has finished, a smaller $l_j$ indicates a slower progress of $PE_j$'s mining task. Nonetheless, if some processors have the same smallest $l_j$, $PE_j$ with the largest $n_j$ is selected as a donor processor. We assume that the more sequential patterns $PE_j$ extracts (i.e. the larger $n_j$ $PE_j$ has), the more candidate patterns are generated. After determining a donor processor $PE_{j'}$ where $l_{j'}$-patterns are classified into $m_{j'}$ groups of different starting symbols, $PE_{j'}$ sends $(m_{j'}/2)$

groups to $PE_i$. Subsequently, $PE_i$ extracts sequential patterns whose first $l_{j'}$ symbols are $l_{j'}$-patterns in the groups sent from $PE_{j'}$.

## 2.5   Experimental Results

We selected three movies, Star Wars Episode 2 (*SWE2*), Star Wars Episode 4 (*SWE4*) and Men In Black (*MIB*). For our experiment, our video data mining approach was tested on six fragments of about 30 minutes length from the above three movies. These six fragments are listed below:

- *video 1* is a fragment in *SWE2* and contains 444 shots.

- *video 2* is a fragment in *SWE2* and contains 725 shots.

- *video 3* is a fragment in *SWE4* and contains 578 shots.

- *video 4* is a fragment in *SWE4* and contains 439 shots.

- *video 5* is a fragment in *MIB* and contains 444 shots.

- *video 6* is a fragment in *MIB* and contains 453 shots.

The above videos are compressed in MPEG-1 format with a frame rate of 29.97 $frames/second$. Shot boundaries are detected using MP-Factory (MPEG Software Development Kit) library [2]. It should be noted that *video 2* contains a larger number of shots than the other videos because almost all semantic events are battle events presented with fast transitions of shots with short durations.

### 2.5.1   Evaluations of Assigning Categorical Values of $SM$

For the experimental videos, we evaluate the performances of our method for assigning categorical values of $SM$ to shots. These performances are presented in Table 2.2. The precision and recall of each video are computed by comparing our method with the human observer. If a shot contains both *speech* and *music*, the louder one is selected as the ground truth for the shot. As shown in Table 2.2, none of the performances in the experimental videos are satisfactory because these videos contain various kinds of sounds (e.g.

Table 2.2: The performance evaluations of assigning categorical values of *SM*.

|                    | *video 1* | *video 2* | *video 3* | *video 4* | *video 5* | *video 6* |
|--------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| precision (speech) | 52.9 (%)  | 57.29     | 72.11     | 74.94     | 97.2      | 96.2      |
| precision (music)  | 37.53     | 82.67     | 49.72     | 50.0      | 21.5      | 29.5      |
| recall (speech)    | 72.34     | 31.14     | 41.28     | 64.6      | 49.6      | 25.4      |
| recall (music)     | 20.68     | 92.8      | 78.44     | 61.2      | 90.5      | 96.8      |

sound effects and sounds of machine engines) apart from *speech* and *music*. Regarding these low performances, in order to extract reliable semantic patterns from the experimental videos, we manually assign a categorical value of $SM$, such as *speech*, *music*, *others* or *no-sound* to each shot.

## 2.5.2 Evaluation of Semantic Event Boundary Detections

In order to detect semantic event boundaries in a video, we use the method introduced in [103] that intelligently merges semantically related shots into a semantic event. In this method, semantic event boundaries are dynamically updated based on time-adaptive shot grouping, where the spatial and temporal information and temporal localities are used. However, since temporal localities rely on the average of shot durations in the whole video, the method in [103] does not work well when some segments have significantly different averages of shot durations. For example, shot durations are relatively long in segments where romantic events are presented, while shot durations are relatively short in segments where thrilling events are presented. The overall average of shot duration in these two segments is not suitable for capturing temporal localities in any of the above two kinds of segments. Hence, the method in [103] should not be applied to the whole video, but to individual segments separately. In each segment, shot durations are almost similar, thus the average of these shot durations is suitable for capturing temporal localities in the segment.

In order to divide a video into segments, we use the method introduced in [45]. This method models shot durations in the video by using an infinite-state automaton, where each state is associated with a probabilistic density function associated with an expected value of the shot duration. Furthermore, in order to make this modeling robust to many insignificant changes in

Table 2.3: The performance evaluations of our semantic event boundary detection method.

|  | *video 1* | *video 2* | *video 3* | *video 4* | *video 5* | *video 6* |
|---|---|---|---|---|---|---|
| # of segments | 4 | 3 | 4 | 3 | 3 | 3 |
| precision | 64.9 (%) | 55.8 | 63.2 | 54.3 | 25.0 | 62.1 |
| recall | 77.4 | 71.7 | 85.7 | 73.1 | 42.0 | 75.0 |

shot durations, costs are assigned to state transitions. As a result, the video is divided into segments in which shot durations are modeled by a single state in the infinite-state automaton. That is, shot durations in the segment are relatively constant towards the expected value of the state. Finally, by applying the method in [103] to each segment separately, the detection of semantic event boundaries have increased accuracy as compared to its application to the whole video.

Table 2.3 shows the performance evaluation of our semantic event boundary detection method. The row labeled '# of segments' shows that each experimental video is divided into three or four segments. The precision and recall in each video are computed by comparing our method with the human observer. The reasonably high recall values indicate that many true semantic event boundaries are detected by our method, although the relatively low precision values indicate that our method over-detects semantic event boundaries. The main reason is that the spatial and temporal information in a shot cannot be successfully obtained only from color information. For example, semantically related shots where a character performs similar actions in different background locations cannot belong to the same semantic event. Since we cannot extract reliable semantic patterns by using erroneous semantic event boundaries detected by our method, we manually correct these semantic event boundaries.

### 2.5.3   Evaluations of Extracted Semantic Patterns

Using our time-constrained sequential pattern mining method, we extracted semantic patterns from the experimental videos. Examples of extracted semantic patterns are shown in Table 2.4, where the three columns from the left represent the properties of extracted semantic patterns and the remaining four columns represent the results of retrieving semantic events using these patterns. In the third column, a serial relationship between two con-

secutive symbols $X$ and $Y$ is represented as $X - Y$ and a parallel relationship is represented as $XY$. Extracted semantic patterns can be classified into the following three groups. An *action* group includes the semantic events where characters perform specific actions (i.e. talk, move and violence). A *situation* group includes the specific situations (i.e. dark, close-up and thrilling). A *combination* group include the semantic events where characters perform specific actions in specific situations (i.e. talk in thrilling situation, talk in dark situation and so on).

For each semantic pattern in Table 2.4, the precision (P) and recall (R) are computed based on semantic events retrieved by using the semantic pattern. Additionally, a video annotated with * means that the semantic pattern cannot be extracted from the video. In accordance with these notations, we briefly explain extracted semantic patterns and evaluate their retrieval results.

Firstly, the semantic patterns associated with talk events $SM1 - SM1$, $SM1MV0$ and $MV0 - MV0$ are characterized by 'two continuous shots with human voice', 'a shot with human voice and no direction of movement' and 'two continuous shots with no direction of movement', respectively. The considerably high recall values for these patterns indicate that characters talk to each other and do not move noticeably in most talk events. The semantic pattern associated with move event $MV4SM2$ is characterized by 'a shot with a constant movement direction and music'. This pattern indicates that background music is frequently used when some vehicles or characters move. The semantic patterns associated with violence events $MS5SM2$, $SM2 - SM2$ and $SL0 - SL0$ are characterized by 'a shot with a large amount of movement and music', 'two continuous shots with music' and 'two continuous shots with short durations', respectively. In particular, $MS5SM2$ and $SM2 - SM2$ reveal the interesting tendency in violence events that background music is generally more emphasized than character's voice. But, we could not extract them from $MIB$ (i.e. *video 5* and *video 6*) where background music is hardly used. So, $SL0 - SL0$ is only the semantic pattern for characterizing violence events in $MIB$ and this pattern is also applicable to *SWE2* and *SWE4*. Finally, all of the above semantic patterns in the action group consist of features for temporal aspects.

Apart from the action group, the situation group includes semantic patterns consisting of features for spatial aspects. The semantic pattern associated with dark situation $CV2$ is characterized by 'a shot where brightnesses of most pixels are of low value'. Although all recall values are $100(\%)$, the

Table 2.4: Examples of extracted semantic patterns.

| group | event | pattern | *video 1* | *video 2* | *video 3* |
|---|---|---|---|---|---|
| action | talk | $SM1 - SM1$ | P:56.0 R:100 | P:66.7 R:88.9 | P:84.2 R:100 |
| | | $SM1MV0$ | P:65.0 R:92.9 | P:66.7 R:100 | P:72.3 R:100 |
| | | $MV0 - MV0$ | P:78.6 R:78.6 | * | P:70.4 R:100 |
| | move | $MV4SM2$ | P:73.3 R:91.7 | * | * |
| | violence | $MS5SM2$ | P:87.5 R:93.8 | P:88.9 R:80.0 | * |
| | | $SM2 - SM2$ | P:84.2 R:100 | P:76.7 R:62.2 | P:53.3 R:88.9 |
| | | $SL0 - SL0$ | * | P:81.8 R:90.0 | P:60.0 R:100 |
| situation | dark | $CV2$ | P:87.5 R:100 | * | P:35.5 R:100 |
| | close-up | $LN1$ | P:52.2 R:85.7 | P:42.2 R:86.4 | * |
| | | $SA3$ | P:48.1 R:92.9 | P:41.2 R:95.5 | P:100 R:61.1 |
| | thrilling | $CS4$ | P:56.2 R:100 | P:60.0 R:84.0 | P:71.4 R:100 |
| combination | talk + thrilling | $CS4SM1$ | P:30.0 R:100 | * | * |
| | talk + dark | $CV2SM2$ | P:61.1 R:84.6 | * | * |
| | | $CV2-$ $CV2LN1SA4$ | P:66.7 R:72.7 | * | * |
| | talk + close-up | $SA3SM1MV0$ | P:53.3 R:100 | * | P:100 R:47.4 |
| | | $LN1SM1MV0$ | * | * | P:92.6 R:65.0 |
| | talk + indoor | $LN2SM1MV0$ | P:78.6 R:84.6 | * | * |
| | dark + close-up | $CV2LN1SA3$ | P:71.4 R:83.3 | * | * |
| | dark + close-up + violence | $CV2LN1$ $SA3SL0$ | P:55.6 R:55.6 | * | * |

| group | event | pattern | *video 4* | *video 5* | *video 6* |
|---|---|---|---|---|---|
| action | talk | $SM1 - SM1$ | P:85.0 R:73.9 | P:100 R:91.7 | P:75.0 R:93.8 |
| | | $SM1MV0$ | P:90.5 R:89.5 | P:100 R:100 | P:65.2 R:93.8 |
| | | $MV0 - MV0$ | P:77.3 R:85.0 | P:100 R:100 | P:65.2 R:93.8 |
| | move | $MV4SM2$ | * | * | * |
| | violence | $MS5SM2$ | * | * | * |
| | | $SM2 - SM2$ | P:43.8 R:100 | * | * |
| | | $SL0 - SL0$ | P:46.7 R:100 | * | P:40.0 R:100 |
| situation | dark | $CV2$ | P:47.1 R:100 | * | P:35.3 R:100 |
| | close-up | $LN1$ | * | P:46.2 R:100 | P:89.5 R:85.0 |
| | | $SA3$ | P:66.7 R:82.4 | * | P:94.7 R:90.0 |
| | thrilling | $CS4$ | P:71.4 R:83.3 | * | * |
| combination | talk + thrilling | $CS4SM1$ | * | * | * |
| | talk + dark | $CV2SM2$ | * | * | P:14.3 R:66.7 |
| | | $CV2-$ $CV2LN1SA4$ | * | * | * |
| | talk + close-up | $SA3SM1MV0$ | P:90.9 R:62.5 | * | P:50.0 R:80.0 |
| | | $LN1SM1MV0$ | P:75.0 R:100 | P:72.7 R:80.0 | P:40.0 R:50.0 |
| | talk + indoor | $LN2SM1MV0$ | * | * | P:73.3 R:100 |
| | dark + close-up | $CV2LN1SA3$ | * | * | * |
| | dark + close-up + violence | $CV2LN1$ $SA3SL0$ | * | * | * |

low precision values of *video 3*, *video 4* and *video 6* represent that retrieval results by $CV2$ include many semantic events where black-costumed characters play important roles. Examples of such black-costumed characters are Darth Vader in *SWE4* and characters wearing black suits and sunglasses in *MIB*. Two semantic patterns associated with character's close-up $LN1$ and $SA3$ are characterized by 'a shot containing few straight lines' and 'a shot containing a large skin colored region', respectively. $LN1$ is confirmed by the fact that a human face is generally rounded and so few straight lines can be derived from its boundary. $SA3$ conform with our intuition that $SA$ reflects the size of the main character in a keyframe. However, the low precision values in *video 1* and *video 2* indicate that $SA3$ does not work well for videos where there are many skin colored objects (e.g. rock and desert). The semantic pattern associated with thrilling situation $CS4$ is characterized by 'a shot where saturations of most pixels are of low value'. This type of shot generally displays a blurred situation where an explosion or chase with high speed takes place. But, $CS4$ is not extracted from *MIB* as the dominant color in most shots is black.

For the semantic patterns in the combination group, $CS4SM1$, $CV2SM2$, $CV2 - CV2LN1SA4$, $SA3SM1MV0$, $LN1SM1MV0$, $LN2SM1MV0$, $CV2LN1SA3$ and $CV2LN1SA3SL0$ are the combinations of the action and situation groups. For example, $LN1SM1MV0$ is associated with talk events of character's close-up. This pattern is the combination of $LN1$ for character's close-up and $SM1MV0$ for talk events. Additionally, $CV2LN1SA3SL0$ which is associated with violence events of dark and character's close-up, is the combination of three types of semantic patterns $CV2$ for dark situation, $LN1$ and $SA3$ for character's close-up and $SL0$ for violence events. In this way, the combination group specifies more complex semantic events than those of the action and situation groups.

## 2.6  Summary

In this chapter, we introduced a method which extracts semantic patterns as sequential patterns in a multistream of features. For an efficient extraction of semantic patterns, we adopt an apriori-based approach to reduce the search space of possible sequential patterns. In addition, we use $SEB$ and $TDT$ time constraints in order to avoid extracting sequential patterns which are semantically irrelevant. Furthermore, we parallelize our method to reduce

its computation cost. Experimental results show that several interesting semantic patterns are extracted from commercial movies. Currently, we have developed a video retrieval system using extracted semantic patterns.

In future works, we will address the problem that transforming a raw material video into a multistream inevitably involves *semantic noises*. A semantic noise means that the same categorical value of a feature is assigned to semantically different shots. For example, the same categorical value of $SA$ can be assigned to two types of shots, one in which a character appears close to the camera and the other in which a skin colored background is shown. Such semantic noises prevent us from extracting some interesting semantic patterns. With respect to this, we plan to develop a video data mining approach which extracts semantic patterns without deriving features from a raw material video. To achieve this goal, a *data squashing* technique [94] may be useful as it scales down large original video data into smaller *pseudo data*, while preserving nearly the same structure as the original data. Since each *pseudo data* element has a weight for reflecting the distribution of the original data, we aim to develop a new data mining method which accepts weights to extract semantic patterns.

# Chapter 3

# Topic Extraction by Burst Detection in Video Streams

## 3.1  Introduction

In this chapter, we introduce a method for extracting *topics* as interesting events in a video. We assume that events presented by abnormal video editing patterns have much impacts on viewers. For example, thrilling events are presented by a fast transition of shots with very short duration, so that the thrilling mood is emphasized. In addition, romantic events are presented by connecting shots with very long durations, where character's emotion and action are thoroughly shown. The above thrilling and romantic events are extracted as topics. On the other hand, in conversation events, shot durations are neither short nor long, so they are not extracted as topics. In the following paragraphs, we will present a topic extraction method which detects abnormal editing patterns using data mining technique.

For the topic extraction, one of the most important tasks is *video segmentation* which is the process of dividing a video into events. Since some of events are extracted as topics, the accuracy of video segmentation is crucial for extracting semantically meaningful topics. Video segmentation methods adopt different approaches depending on the following video types: *structured* and *unstructured*. Structured videos such as news, sports and surveillance videos have explicit structures, while unstructured videos such as movies and dramas do not have such structures. Below, we briefly review existing video segmentation methods in structured and unstructured videos.

## Video Segmentation in structured videos

Events in structured videos are presented based on explicit structures which result from production or game rules. For example, in a news video, an event corresponding to one news story starts from a shot where an anchor person appears and ends at another shot where the anchor person appears again [105]. In a sports video, an event corresponding to one play starts with a specific shot [30]. For example, in a baseball video, an event starts with a shot taken by the camera behind the pitcher. And, if the batter hit out the ball, a camera follows it in the next shot. Also, in a surveillance video, an anomaly event where some action happens is preceded by shots where no motion is observed [41]. Thus, by using the above explicit structures as clues, structured videos can be easily divided into events.

## Video Segmentation in unstructured videos

Events in unstructured videos do not have such an explicit structure. That is, even if events have the same semantic content such as conversation and battle, different directors use different camera and editing techniques to present these events. Considering the lack of explicit structures, many researchers define events based on *similarities of features*, such as color, motion and audio [74, 103, 9, 86, 109, 106]. For example, in an event occurring in the mountains, most shots contain greenish vegetation in the background. In an action event where characters move actively, most shots contain large amounts of motion. Based on these observations, the researchers define an event as a set of shots that not only contain similar features but also are temporally close to each other.

However, features are not necessarily useful for defining semantically meaningful events, because they significantly differ depending on camera techniques. For example, in *Event 1* in Fig. 3.1, the female character $A$ walks around the city. We can see that backgrounds of all shots are different as $A$ walks at different locations. In addition, the shot sizes[1] of these shots are different. Consequently, all shots in *Event 1* contain different color features. Furthermore, their motion features are different. For example, small

---

[1]For a shot, according to the distance between the camera and objects, the shot size is classified into one of the following three types: *tight shot*, *medium shot* or *long shot* [101]. A tight shot is taken by a camera close to objects, while a long shot is taken by a camera distant from objects. A medium shot is taken by a camera, which is placed at an intermediate position between tight and long shots.

amounts of motion are derived from *shot 1* and *shot 2*. It is because these shots are long shots where $A$'s walking is shown in small parts of the screen. On the other hand, *shot 4* is taken by a tight shot where a large amount of motion is derived although $A$ just turns around. Like this, *Event 1* cannot be defined by similarities of features.
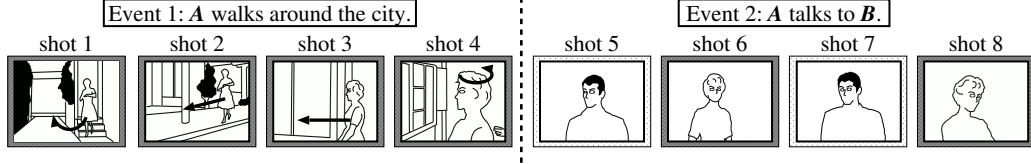


Figure 3.1: Examples of events where *Event 1* cannot be defined by similarities of features, while *Event 2* can be defined by these similarities.

In contrast to features, we develop a video segmentation method using objects like characters. Recently, objects can be recognized with high accuracy. In fact, NEC research group announced that about 80% of main characters' faces can be accurately recognized [4]. Considering such progress, we define an event based on a target character's appearance and disappearance. The idea behind this is that a semantically meaningful event is characterized by interaction between the target character and other characters. For example, in an event where only the target character performs an action such as *Event 1* in Fig. 3.1, he/she appears in most shots. On the other hand, in an event where the target character interacts with other characters such as *Event 2*, a sequence is repeated where he/she appears in one shot and other characters appear in the next shot. Based on this idea, we define an event by the pattern of the target character's appearance and disappearance.

However, it should be noted that patterns of the target character's appearance and disappearance are usually not clear for events. For instance, consider an event where the target character is in conversation with another character. Here, shot durations vary depending on their spoken lines. Furthermore, the repetition of shots where the target character appears and shots where the other character appears can be broken if a new character joins the conversation. Additionally, conversation between the target character and the other character may halt for a short period of time. In this way, even within an event, the pattern of the target character's appearance and disappearance is disturbed by various factors. To address this problem,

we incorporate a probabilistic function into our video segmentation method to divide the video into events characterized by probabilistically distinct patterns.

To extract topics from events, we take advantage of *film grammar* which consists of practical rules to concentrate viewer's attention on the story of a video [48]. In particular, we focus that a professional video editor tailors a rhythm of shot durations, so that the central action is not disturbed. For example, shot durations are very short in thrilling events while they are very long in romantic events. By extending this film grammar, we define a topic as an event which contains one of the following two abnormal patterns, called *bursts*. In the first one, the target character appears with very short durations, while in the second one he/she appears with very long durations. By detecting these bursts, we can extract topics where the target character performs interesting actions, such as fighting, chasing, kissing and so on.

## 3.2   Related Works

In this section, we describe the novelties of our topic extraction method from the perspectives of two different research fields, *video data mining* and *burst detection*.

### 3.2.1   Video Data Mining

Video segmentation only determines the boundary between two consecutive events. Thus, events matching a query need to be extracted from the set of events detected by video segmentation. For this purpose, the following methods have been proposed to extract events in unstructured videos. Nam et al. extracted violent events based on multiple features [49]. They heuristically find that violent events have a specific motion, a flame-color, a blood-color and a sudden increase in audio energy. Yoshitaka et al. extracted conversation, suspense and action events based on features such as shot durations, motions and repetition of visually similar shots [16]. For each event, the combination of features is heuristically determined based on film grammar. Zhai et al. extracted conversation, suspense and action events using Finite State Machines (FSMs) [107]. They heuristically construct these FSMs, where a state transition is determined based on features (motion and audio energy) and an object (human face). Like this, existing methods can only extract

limited kinds of events in unstructured videos. In other words, many events which may interest users are not extracted due to the lack of knowledge for these events.

To extract knowledge for events, much research has recently been conducted on video data mining. Generally, it consists of two steps, 'video representation' and 'knowledge extraction'. In the video representation, a raw video is firstly represented by several features which reflect semantic contents in the video. Here, the raw video only represents physical values such as RGB values of pixels in each video frame. But, this physical representation of the video is not directly related to semantic contents. So, in order to extract knowledge for characterizing semantic contents, we need to derive features from the raw video. Then, in the knowledge extraction, we apply a data mining technique to features and extract useful knowledge.

In terms of the knowledge extraction method, existing video data mining approaches can be classified into two categories, *pattern discovery* and *classification and clustering*[2]. Below, we briefly explain each category.

**Pattern discovery:** This category aims to extract special patterns from videos. For example, in [101, 98, 61], a video is represented as a multi-dimensional symbolic stream, where each symbol represents a feature value in a shot, such as color composition, amount of motion, camera work, sound type and so on. From this symbolic stream, the researchers extract frequent sequential patterns which characterize events like conversation, battle and goal events. In [17], a video is represented as a tempo. This indicates a speed at which a viewer perceives semantic contents, say, haste and calm. The researchers define such a tempo based on shot durations and camera works, and extract gradual and sharp tempo changes which characterize dramatic events. In [69], a video recorded by a fixed camera in a kitchen is represented as a multi-dimensional time series, where each dimension represents human movement in one region of interest, such as table, sink or refrigerator. From this multi-dimensional times series, the researchers extract temporal patterns of movements which characterize events. For example, an event of washing dishes is characterized by a movement near the sink, followed by a movement near the dishwasher.

**Classification and clustering:** This category aims to group videos into different classes. In [52], a video is represented as an n-by-n-by-n cube which

---

[2]This classification is originally introduced by X. Zhu et al. [98]. In this chapter, we slightly modify their classification to clarify the novelty of our topic extraction method.

represents both spatial and temporal information in the video. Then, the researchers apply Independent Component Analysis (ICA) to the n-by-n-by-n cube and classify videos into commercial and news. In [90], each shot in a soccer video is represented as a mosaic, which is a synthetic panoramic image obtained by combining video frames in the shot. Then, based on color histograms, camera motions and positions of players in mosaics, the researchers classify shots into free-kick, corner-kick and others. In [41], a human movement in a surveillance video is represented by parameters of a Gaussian Mixture Model (GMM). Based on this, the researchers group various human movements into clusters of similar movements. As a result, clusters including a small number of movements are detected as anomaly events, because such movements are irregular.

Considering the above two categories, our topic extraction method belongs to the category of pattern discovery because we extract abnormal patterns of target character's appearance (i.e. bursts), which characterize topics where the target character performs interesting actions. Also, in the video representation, we represent a video as one-dimensional time series, which represents intervals where the target character appears and intervals where he/she disappears. Such a video representation has not been proposed yet.

## 3.2.2 Burst Detection

Many researchers in the data mining community conduct burst detection to extract useful knowledge from various time series data. Generally, a time series is a sequence of feature pairs on the time axis, where bursts are detected as an abnormal activity of feature pairs. In what follows, we briefly explain bursts in various time series data using Fig. 3.2.

In financial data such as the daily trading volumes for the stock of a company as shown in Fig. 3.2 (a), a burst is detected as an abnormally large trading volume [72]. It characterizes a historical event which affects the company. For example, the 2001/9/11 attack affected various financial and travel related companies.

In a text stream such as an e-mail or a news stream, a burst is detected as an abnormally large number of arrivals of documents containing a specific keyword [45, 35]. It characterizes events related to the keyword. For example, in the e-mail stream in Fig. 3.2 (b), some researcher writes a paper and submits it to the conference sponsored by IEEE in *Event 1*. This event is characterized by many arrivals of messages containing the keyword 'IEEE'.
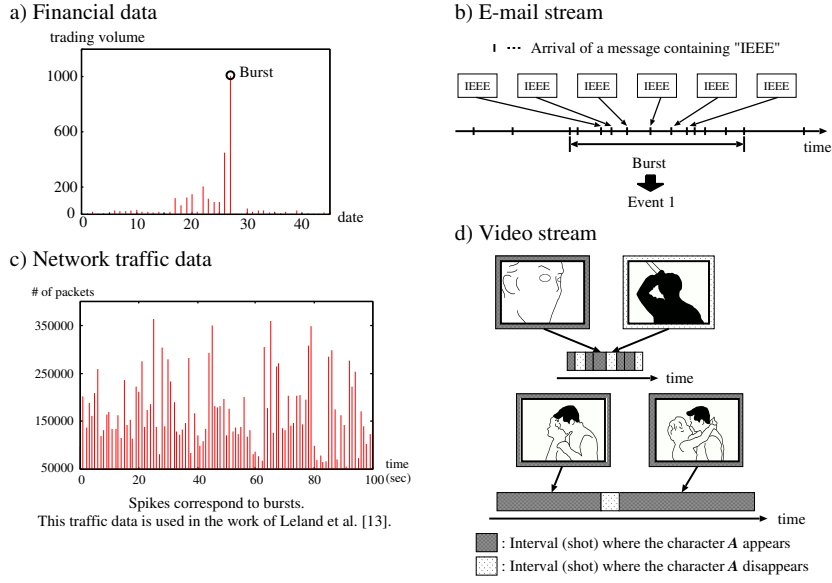
Figure 3.2: Examples of bursts in financial data (a), an e-mail stream (b), network traffic data (c) and a video stream (d).

These messages are mainly sent from co-authors.

In traffic data such as network traffic data or disk I/O traffic data, a burst is defined as an abnormally large number of packets or disk requests [95, 73]. In this research field, the main goal is to appropriately model and generate synthetic traffic data which contain bursts. This is useful for efficient network management and disk scheduling.

Note that each of the above bursts is defined on feature pairs which are associated with time stamps. In other words, as can be seen from Fig. 3.2, trading volumes in financial data, message arrivals in an e-mail stream and numbers of packets in network traffic data are represented by vertical lines on the time axis. On the other hand, a video is a *continuous media* which conveys semantic contents only when it is continuously played over time [26]. So, the video is a time series where the same feature continuously appears in time intervals. For example, Fig. 3.2 (d) shows that the female character $A$ continuously appears in dark-shaded intervals. Therefore, we define a burst in the video based on intervals of a target character's appearance. In particular, we define the following two bursts. In the first, intervals of

the target character's appearance have abnormally short durations, as in the upper part of Fig. 3.2 (d). In the second, intervals of his/her appearance have abnormally long durations, as in the bottom part of Fig. 3.2 (d). Based on the film grammar described in section 3.1, these bursts characterize topics where the target character performs interesting actions. Finally, we are not aware of the above kind of interval-based burst being proposed in any previous research.

## 3.3 Basic Concepts

This section describes the basic concepts underlying our proposed topic extraction method. In addition, to implement our video segmentation method, we describe an important statistical property in which durations of character's appearance and disappearance follow exponential distributions.

### 3.3.1 Intervals of a Target Character's Appearance and Disappearance

Fig. 3.3 is used to explain the proposed video representation from the viewpoint of a target character. Fig. 3.3 shows the editing process of making a conversation event with three characters, one woman $A$ and two men $B$ and $C$. Here, *shot 1* where only $A$ appears is recorded by *camera 1*, *shot 2* where only $B$ appears is recorded by *camera 2*, and *shot 3* where all characters $A$, $B$ and $C$ appear is recorded by *camera 3*. A video editor iterates selecting one of the above three types of shots and joining it to the previous shot. As a result, the conversation event has the following temporal order: *shot 3* → *shot 2* → *shot 1* → *shot 2* → *shot 3* → *shot 1* → *shot 2*.

Since $A$ appears in *shot 1* and *shot 3*, if she is a target character, we can construct the sequence shown in the lower part of Fig. 3.3. In this sequence, dark-shaded intervals represent shots where $A$ appears on the screen, while light-shaded ones represent shots where she disappears from the screen. Similarly, if a character $B$ or $C$ is a target character, we can construct a sequence of intervals of his appearance and disappearance. In this way, by targeting at a certain character, the video can be represented as a one-dimensional time series, i.e., a sequence of intervals of his/her appearance and disappearance.

In the above sequence, both of a character's appearance and disappearance are valuable in the characterization of semantic content of the video.
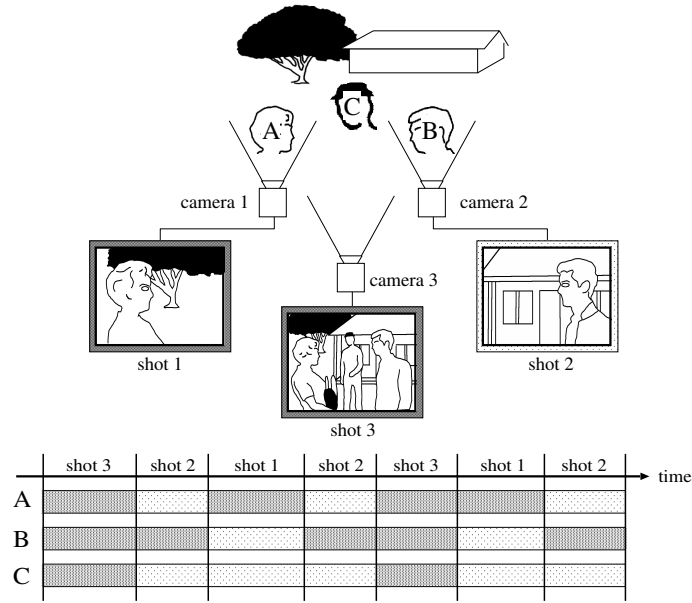
Figure 3.3: Intervals of a target character's appearance and disappearance.

Clearly, the character's appearance is essential to indicate his/her action on the screen. In addition, the character's disappearance may suggest whether or not he/she is of importance in the story. With respect to this, a video editor rarely uses redundant shots to avoid interrupting a viewer's interest [48]. This means that 'important' characters appear in many shots while 'unimportant' characters only appear in a few shots. For example, in the conversation event depicted in Fig. 3.3, the main conversation is between $A$ and $B$ while $C$ eavesdrops on their conversation. The editor designs this event so that it mostly consists of *shot 1* and *shot 2*, are of most obvious interest to the viewer, but also contains some instances of Shot 3 to ensure the presence of character C is not forgotten. As a result, the unimportant character $C$ appears in only a few shots and disappears in the rest, as depicted by long intervals of $C$'s disappearance in Fig. 3.3. Thus, invisible information, like a character's disappearance, is also useful for analyzing semantic content in the video.

### 3.3.2 Video Segmentation based on a Target Character's Appearance and Disappearance

Our proposed video segmentation can be explained using the sequence of intervals of $A$'s appearance and disappearance in Fig. 3.4. To clarify our concepts, the sequence is transformed into the bar graph representation, as shown in the lower section of Fig. 3.4.This is constructed by rotating each interval by 90 degrees to transform it into a vertical bar. In other words, the duration of the interval is represented as the height of the bar. Bars representing intervals of $A$'s appearance are directed upwards while those of $A$'s disappearance are directed downwards. Finally, bars are located on the horizontal axis in temporal order.
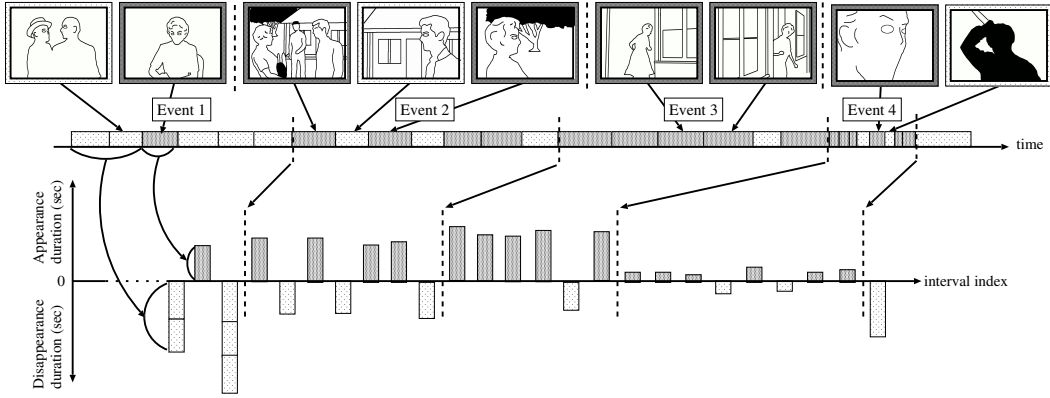


Figure 3.4: The proposed video segmentation based on a sequence of intervals of a target character's appearance and disappearance.

The bar graph representation in Fig. 3.4 clearly shows that each event is characterized by a specific pattern of $A$'s appearance and disappearance. In *Event 4*, where $A$ is murdered, all shot durations are very short, indicating that durations of $A$'s appearance and disappearance are also very short. Compared to *Event 4*, the other events are less thrilling and are characterized by much longer durations of $A$'s appearance and disappearance. Particularly in *Event 1*, where $A$ is an unimportant character, durations of her disappearance are comparatively long. In this way, a semantically meaningful event is characterized by the pattern of a target character's appearance and disappearance

However, events cannot be completely characterized only by appearance patterns. For example, in Fig. 3.4, durations of $A$'s appearance and disappearance in *Event 2* and *Event 3* are similar. But, it is clear that *Event 2* and *Event 3* are semantically different. In order to discriminate between such events, we consider an *occurrence ratio* between a target character's appearance and disappearance. Clearly, *Event 2* and *Event 3* have different occurrence ratios for $A$'s appearance and disappearance. Specifically, in *Event 2*, $A$'s appearance is frequently followed by $B$'s appearance. Thus, the number of $A$'s appearance, 4, is nearly equal to the number of her disappearance, 3. On the other hand, in *Event 3*, $A$ appears in most shots. Here, the number of $A$'s appearance, 5, is greater than her single disappearance. Therefore, we characterize a semantically meaningful event by using both the pattern of target character's appearance and disappearance, and the occurrence ratio.

### 3.3.3 Exponential Characteristics of Durations of a Target Character's Appearance and Disappearance

In order to implement a probabilistic video segmentation method, we need to know the statistical property of the durations of a target character's appearance and disappearance. In the following paragraphs, for the simplicity, we abbreviate duration of a target character's appearance and duration of his/her disappearance as *appearance duration* and *disappearance duration*, respectively. We assume that appearance and disappearance durations follow exponential distributions. An exponential distribution is generally used to model waiting times for events which occur at a constant rate in time, like waiting times for system failures and phone calls [1]. Such exponential distributions can be applied to appearance and disappearance durations. The reason is that an appearance duration can be defined as the waiting time associated with a target character's disappearance, while a disappearance duration corresponds to the waiting time associated with his/her appearance. In addition, if the target character is a main character, the switching rate between his/her appearance and disappearance is assumed to be constant and high throughout the video. Intuitively, the target character appears in many shots where he/she performs various actions, while he/she does not appear in many shots where various reactions from other characters are presented. Below, we examine whether appearance and disappearance durations

actually follow exponential distributions.

For a real variable $x$ ($\geq 0$), the probabilistic density function of an exponential distribution is $pdf(x) = \lambda e^{-\lambda x}$. If an appearance duration $X$ follows the exponential distribution, the cumulative distribution function is $cdf(X > x) = e^{-\lambda x}$ where $1/\lambda$ is the mean appearance duration. $cdf(X > x)$ represents the probability of $X$ which is larger than $x$. Thus, from appearance durations in an actual video, we can estimate $cdf(X > x)$ as $N(x)/N$, where $N$ is the number of appearance durations in the whole video and $N(x)$ is the number of appearance durations larger than $x$. Similarly, the exponential characteristic of disappearance durations can be estimated.

Fig. 3.5 shows the estimated exponential characteristic of appearance and disappearance durations in four experimental videos. In each video, one main character is regarded as a target character, and estimate $cdf(X > x)$ for his/her appearance (or disappearance) durations. The estimated $cdf(X > x)$ is depicted by the crossed line, where each cross is $N(x)/N$ for each appearance (or disappearance) duration. On the other hand, the theoretical $cdf(X > x) = e^{-\lambda x}$ is depicted by the solid line. As can be seen from Fig. 3.5, for the main character's appearance (or disappearance) durations, the estimated $cdf(X > x)$ well coincides with the theoretical $cdf(X > x)$. Hence, appearance and disappearance durations follow exponential distributions. Specifically, the exponential characteristic of appearance durations indicates that most of appearance durations are short, because each video is generally edited by connecting shots with short durations. In comparison, the exponential characteristic of disappearance durations indicates that most of disappearance durations are short, because it is rare that the main character continuously disappears in consecutive shots. In the next section, we utilize the above exponential characteristic of appearance and disappearance durations to extract topics.

## 3.4 Topic Extraction by Burst Detection in Videos

In this section, we describe the proposed topic extraction method. Firstly, the video segmentation method is explained, followed by a description of an evaluation measure, which is used to examine whether or not each event contains a burst.
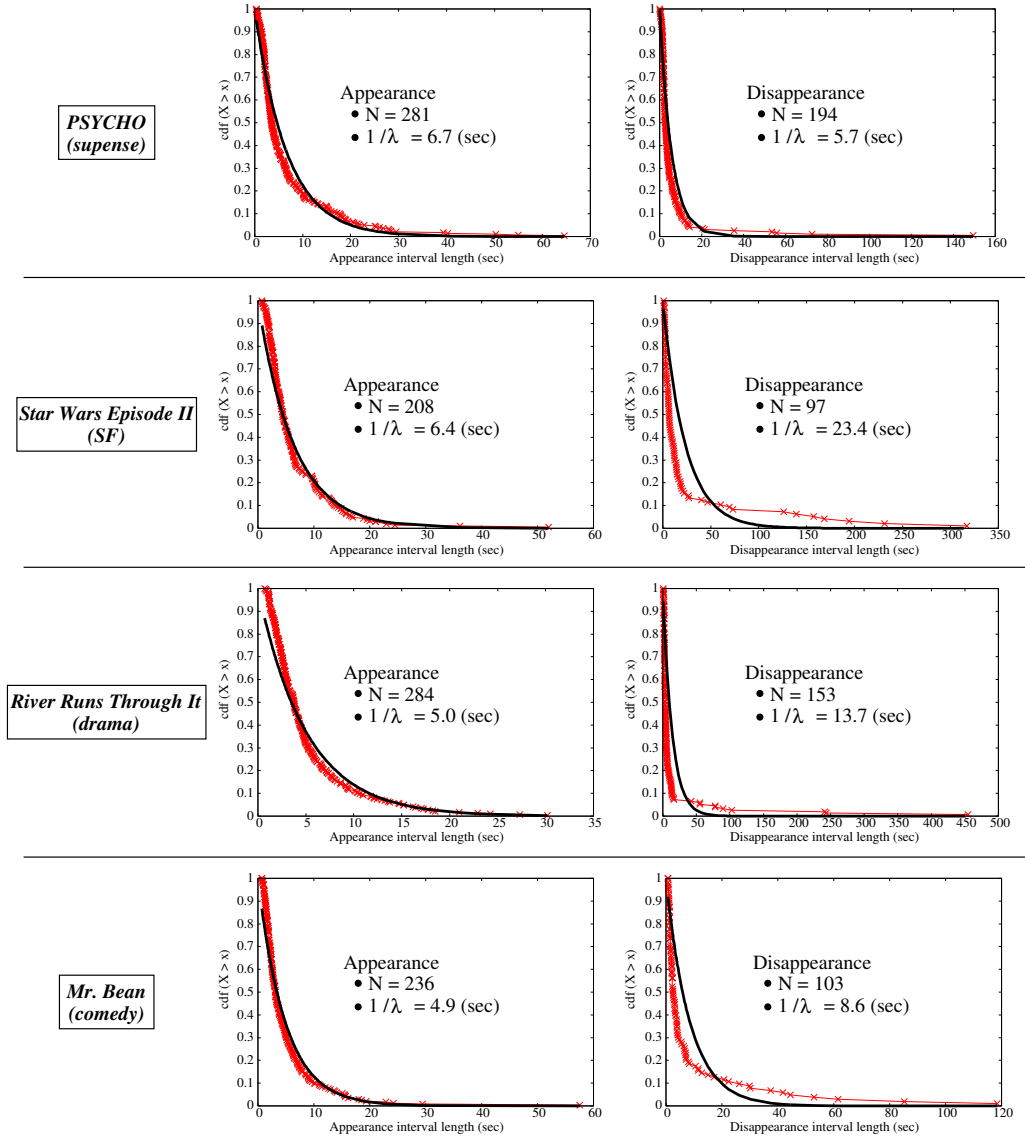
Figure 3.5: Results of examining the exponential characteristic of appearance and disappearance durations in four movies.

## 3.4.1 Video Segmentation Algorithm

A sequence of intervals of a target character's appearance and disappearance $X$ can be formulated as follows:

$$X = x_1, x_2, x_3, \cdots, x_N \qquad x_i = (a_i, d_i), \tag{3.1}$$

where $a_i \in \{A(ppearance), D(isappearance)\}$ represents the type of $i$-th interval, and $d_i \in \Re$ represents its duration. The time series segmentation technique is then used to divide $X$ into the following sequence $S$ consisting of non-overlapping $K$ events ($K \ll N$) [44]:

$$E = e_1, e_2, e_3, \cdots, e_K \qquad e_i = x_a, x_{a+1}, \cdots, x_b, \tag{3.2}$$

where $e_i$ is the subsequence from the $a$-th interval to the $b$-th interval in $X$.

Each event $e_i$ is evaluated based on whether it is be characterized by the pattern of the character's appearance and disappearance, and the occurrence ratio. To achieve this, the following probabilistic function is used:

$$
\begin{aligned}
p(e_i) &= \Pi_{j=a}^{b} \; p(x_j) \\
&= \Pi_{j=a}^{b} \begin{cases} p_A(d_j) \cdot p(a_j = A), & if \;\; a_j = A \\ p_D(d_j) \cdot p(a_j = D), & if \;\; a_j = D. \end{cases}
\end{aligned}
$$

For each interval $x_j = (a_j, d_j)$ in $e_i$, the following probabilistic distributions are used to compute the probability of $x_j$, denoted by $p(x_j)$. $p(a_j)$ represents the probability distribution of the type $a_j$. It consists of the probability of an appearance interval $p(A)$ and the one of a disappearance interval $p(D)$. If types of intervals in $e_i$ follow a single probability distribution $p(a_j)$ with a high probability, we regard the occurrence ratio as invariant in $e_i$. $p_A(d_j)$ and $p_D(d_j)$ represent the probability distribution of appearance durations and that of disappearance durations, respectively. In other words, $p_A(d_j)$ and $p_D(d_j)$ are used to evaluate the similarity of appearance durations and that of disappearance durations, respectively. Thus, given $p(a_j)$, $p_A(d_j)$ and $p_D(d_j)$, $p(e_i)$ represents a joint probability of all appearance and disappearance intervals in $e_i$. Consequently, $p(e_i)$ provides an overall evaluation value of whether appearance durations, disappearance durations and the occurrence ratio are invariant in $e_i$.

It is important to note that the above discussion assumes that $p(a_j)$, $p_A(d_j)$ and $p_D(d_j)$ are already known. However, these are generally unknown. In other words, for $e_i$, the pattern of the character's appearance

and disappearance and the occurrence ratio need to be determined. To this end, we estimate the optimal $p(a_j)$, $p_A(d_j)$ and $p_D(d_j)$ which maximize $p(e_i)$. For $p(a_j)$, the optimal $p(a_j = A)$ and $p(a_j = D)$ are estimated as $N_A^{e_i}/N^{e_i}$ and $N_D^{e_i}/N^{e_i}$, respectively.[3] Here, $N^{e_i}$ is the total number of appearance and disappearance intervals in $e_i$. $N_A^{e_i}$ and $N_D^{e_i}$ are the number of appearance intervals and that of disappearance intervals, respectively.

For $p_A(d_j)$ and $p_D(d_j)$, based on the exponential characteristics of appearance and disappearance durations, the following exponential distributions are employed:

$$
\begin{aligned}
p_A(d_j) &= \lambda_A^{e_i} e^{-\lambda_A^{e_i} d_j} \\
p_D(d_j) &= \lambda_D^{e_i} e^{-\lambda_D^{e_i} d_j}.
\end{aligned}
\tag{3.3}
$$

where the optimal $p_A(d_j)$ has the parameter $1/\lambda_A^{e_i}$ which is the mean appearance duration in $e_i$, while the optimal $p_D(d_j)$ has the parameter $1/\lambda_D^{e_i}$ which is the mean disappearance duration[3]. In this way, we estimate the optimal $p(a_j)$, $p_A(d_j)$ and $p_D(d_j)$, and then compute $p(e_i)$ by applying them to equation (5.1).

We aim to divide $X$ into $K$ events which are characterized by the corresponding $K$ patterns and $K$ occurrence ratios with the highest probability. To do so, the following joint probability of $K$ events in $X$ is maximized:

$$
P(X) = \Pi_{i=1}^{K} \ p(e_i).
\tag{3.4}
$$

In order to simplify $P(X)$, we maximize $P' = log\, P(X) = \sum_{i=1}^{K} log\, p(e_i)$. Note that $log()$ is a monotonically increasing function, and thus, the result of maximizing $P(X)$ is equivalent to that of maximizing $logP(X)$. The above summation maximization problem can be optimally solved using a dynamic programming technique [44]. Since it requires a very high computational cost $O(N^2 K)$, various techniques have been proposed to compute approximately optimal maximization with a significantly lower cost [44]. But, in this paper, the dynamic programming technique is used to obtain the optimal $K$ events, so that the anomaly of appearance durations in each event (i.e. burst) can be accurately examined.

---

[3]This can be proved by taking the logarithm of $p(e_i)$, substituting $p(a_j = D)$ with $1 - p(a_j = A)$ and differentiating $log\, p(e_i)$.

### 3.4.2   Burst Intensity Measure

An evaluation measure of *burst intensity* (BI) is devised to evaluate whether or not each event $e_i$ contains a burst. Here, the burst intensity of $e_i$ represents the degree of anomaly in appearance durations and is given by:

$$BI(e_i) = \frac{T_A^{e_i}}{T^{e_i}} \times \int_0^\infty |\lambda_A^{e_i} e^{-\lambda_A^{e_i} x} - \bar{\lambda}_A e^{-\bar{\lambda}_A x}| \, dx. \tag{3.5}$$

In this equation, $T^{e_i}$ is the total duration of $e_i$ and $T_A^{e_i}$ is the total duration of the character's appearance in $e_i$. The first term is a weighting factor which signifies that if the character appears for a longer duration, he/she plays a more important role. The second term represents the difference between the exponential distribution estimated from appearance durations in $e_i$ and the exponential distribution estimated from appearance durations in the entire video. A large difference indicates that $e_i$ contains either abnormally short or long appearance durations. Thus, if $BI(e_i)$ is larger than a pre-defined threshold, $e_i$ is regarded as a topic where a burst occurs.

## 3.5   Experimental Results

This section presents the experimental results of our topic extraction method. First, we introduce our automatic recognition method of a target character in a video. We then describe the results of our video segmentation method. Finally, we present several topics extracted by using the burst intensity measure.

### 3.5.1   Automatic Character Recognition Method

We summarize our method for recognizing a target character in a video. Our method consists of two phases: *detection phase* and *recognition phase*. In the detection phase, we detect face regions from the keyframe in each shot. Then, in the recognition phase, we recognize the target character based on detected face regions.

In the detection phase, faces with various directions are detected using the method proposed by Matsuyama et al. [47]. First, this method uses InfoBoost to learn a classifier of Haar-like features, as shown in Fig. 3.6 (a). Note that the learned classifier can only detect frontal face regions. So, in order to detect faces with various directions, the classifier is mapped into a

3D model shown in Fig. 3.6 (b). Then, as can be seen from Fig. 3.6 (c), the 3D model is rotated around the vertical axis. Finally, 3D models which are rotated with various angles are projected back to the 2D plain. In this way, classifiers which can not only detect frontal face regions but also side face regions, are generated.
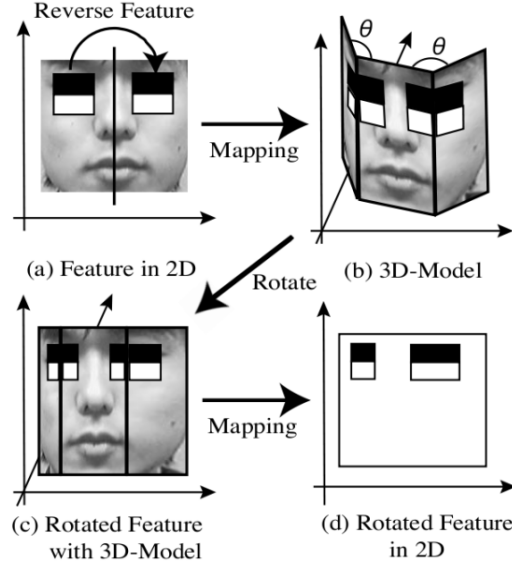


Figure 3.6: Illustration of generating classifiers for detecting face regions of various directions.

In the recognition phase, it should be noted that some facial parts like eyes, nose and mouth are occluded in side face regions. Hence, it is difficult to directly recognize the side face regions of the target character. Thus, we start by only recognizing the frontal face regions of the target character using the method proposed by Keysers et al. [27]. This method uses a two-dimensional Hidden Markov Model (2DHMM) to match frontal face regions with the reference frontal face regions of the target character. Thereby, we can flexibly match frontal face regions where the positions of facial parts are slightly different.

Recognized frontal face regions of the target character are associated with his/her side face regions. In addition, we correct wrong recognition of frontal face regions. To this end, we focus on the following *temporal locality* of a video: since the semantic content of the video are sequentially presented

shot by shot, it is likely that temporally close shots are semantically similar while temporally distant shots are semantically different. In particular, we assume that the target character wears the same cloth in temporally close shots. Based on this assumption, two cases shown in Fig. 3.7 are considered. In the first case, a shot containing a side face region is temporally close to a shot containing a frontal face region, such as *shot i* and *shot i+2* in Fig. 3.7 (a). We compare the clothes under the side and frontal face regions. If the clothes are the same, the side face region is classified into the target character. In the second case, the target character in temporally close shots are wrongly recognized as different characters, such as *shot j* and *shot j+2* in Fig. 3.7 (b), because his/her facial expressions are significantly different. Similar to the first case, the clothe under the wrongly recognized frontal face region is compared to the one under the correctly recognized frontal face region. In this way, by considering multiple shots based on the temporal locality, we can not only recognize side face regions of the target character but also improve the recognition accuracy.

Figure 3.7: Association of a frontal face region with a side face region, and correction of wrongly recognized frontal face regions.

## 3.5.2 Video Segmentation Results

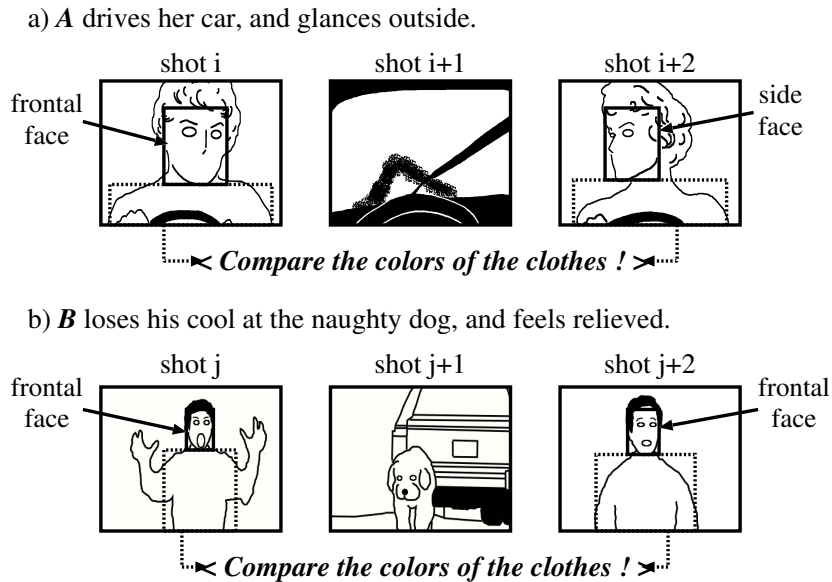The proposed topic extraction method is tested using four movies. These are summarized in Table 3.1. For each movie, the first, second and third columns represent its title, the number of shots and its total duration, respectively. The character depicted in the fourth column is selected as a target character. The fifth column shows the summary of the sequence of intervals of the target character's appearance and disappearance.

Table 3.1: The summary of each experimental video and the sequence of intervals of the target character's appearance and disappearance.

| Title | # of shots | Total duration | Target character | # of intervals (app/disapp) |
|---|---|---|---|---|
| *PSYCHO* | 480 | 2986 (sec) | Marion | 475 (281/194) |
| *Star Wars Episode II* | 688 | 3600 (sec) | Anakin | 305 (208/97) |
| *River Runs Through It* | 706 | 3533 (sec) | Paul | 437 (284/153) |
| *Mr. Bean* | 496 | 2038 (sec) | Bean | 339 (236/103) |

Table 3.2 shows the performance of our video segmentation method. The second column presents the number of events which is the parameter of our method. As shown in the third and fourth columns, we examine whether each event obtained by our method is meaningful or non-meaningful. On average, about 77% of events are semantically meaningful. Some main reasons for this good segmentation result is explained using Fig. 3.8. Fig. 3.8 (a) represents a part of the sequence of *Marion*'s appearance and disappearance intervals in *PSYCHO*. It ranges from the 175-th to the 235-th interval. Fig. 3.8 (b) represents a part of the sequence of *Paul*'s appearance and disappearance intervals in *River Runs Through It*. It ranges from the 375-th to the 435-th interval. Both of the above sequences are represented using the bar graph representation. The boundaries between two consecutive events are depicted using the vertical dashed lines. For each event, the solid horizontal line on the positive side represents the mean appearance duration (i.e. $1/\lambda_A^{e_i}$), while the line on the negative side represents the mean disappearance duration (i.e. $1/\lambda_D^{e_i}$).

First, our method can robustly detect events by ignoring insignificant changes in appearance and disappearance durations. For example, in the event from the 182-th to the 197-th interval in Fig. 3.8 (a), *Marion* worries about herself and most of appearance durations are long except for one short duration. Even for such an exceptional duration, the proposed method characterizes the event as generally consisting of long appearance durations.

Table 3.2: Results of our video segmentation.

| Title | # of events ($K$) | # of meaningful events | # of non-meaningful events |
|---|---|---|---|
| *PSYCHO* | 40 | 30 (75%) | 10 (25%) |
| *Star Wars Episode II* | 46 | 38 (83%) | 8 (17%) |
| *River Runs Through It* | 58 | 41 (71%) | 17 (29%) |
| *Mr. Bean* | 51 | 40 (78%) | 11 (22%) |
| Average performance | | 77% | 23% |
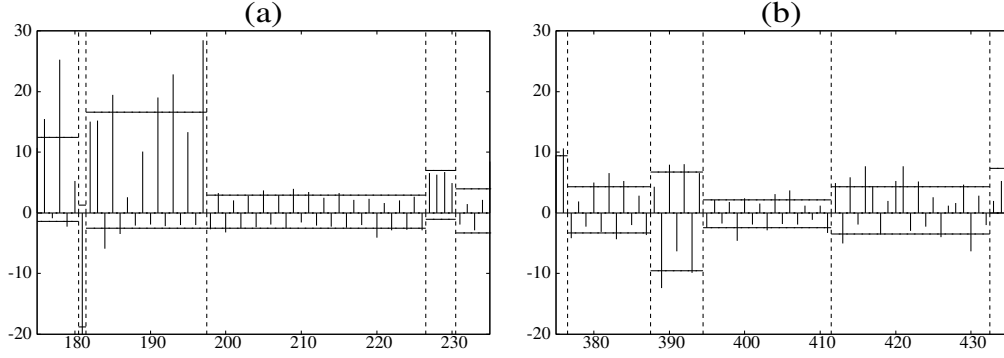


Figure 3.8: Partial results for *PSYCHO* and *River Runs Through It*.

Secondly, an occurrence ratio between target character's appearance and disappearance can appropriately capture an interaction between the target character and other characters. For example, in the event from the 227-th to the 230-th interval in Fig. 3.8 (a), only *Marion* appears and she walks around a motel. This event contains no disappearance of *Marion*. In contrast, in the next event where another character comes to *Marion*, her appearance is followed by her disappearance. Like this, as *Marion* gets to interact with the other character, the occurrence ratio is accordingly changed.

In addition, the use of appearance and disappearance durations is helpful in the extraction of semantically meaningful events. For example, in Fig. 3.8 (b), depending on *Paul*'s appearance and disappearance durations, the intervals from the 377-th to the 432-nd are divided into four events. In the first event from the 377-th to the 387-th interval, *Paul* talks to other characters. Compared to this, the second event from the 388-th to the 394-th interval is characterized by relatively longer disappearance durations, indicating that other characters become to mainly talk rather than *Paul*. The third event from the 395-th to the 411-th interval depicts an excited conver-

sation between *Paul* and other characters. In this event, *Paul*'s appearance and disappearance quickly switch, which relates to short appearance and disappearance durations. Finally, in the fourth event from the 412-th to the 432-th interval, *Paul* dances with his girlfriend. This event is characterized by relatively long appearance durations. In this way, each video can be divided into semantically meaningful events with a high level of accuracy.

### 3.5.3    Topic Extraction Results

Fig. 3.9 shows an overview of topic extraction results. In Fig. 3.9, the time-line of each movie is depicted where a vertical line represents a boundary between two consecutive events. Shaded events represent topics which are numbered sequentially. In particular, light-shaded topics are characterized by bursts of abnormally short appearance durations, while dark-shaded topics are characterized by bursts of abnormally long appearance durations. For the simplicity, the former and latter bursts are termed *short bursts* and *long bursts*, respectively. In the following paragraphs, by referring to Fig. 3.9, we list some representative topics extracted from each video.

**PSYCHO:** 13 topics are extracted from 47 events ($burst\_intensity\_threshold = 0.3$). In the 1-st topic in Fig. 3.9 (a), *Marion* makes love to her boyfriend. *Marion* drives her car in heavy rain in the 6-th topic. And, she is murdered from the 11-th to 13-th topics.

**Star Wars Episode II:** 26 topics are extracted from 46 events ($burst\_intensity\_threshold = 0.15$). In Fig. 3.9 (b), topics from the 4-th to the 11-th belong to one large event where *Anakin* chases and fights an enemy. In the 4-th, 10-th and 11-th topics, *Anakin* fights the enemy. From the 5-th to the 9-th topic, he chases the enemy. In the 20-th, 21-th and 26-th topics, *Anakin* talks with the woman he loves.

**River Runs Through It:** 20 topics are extracted from 58 events ($burst\_intensity\_threshold = 0.25$). In the 14-th topic in Fig. 3.9 (c), *Paul* drops down a river. In the 15-th topic, *Paul* fights his brother. In the 19-th topic, he excitedly talks with other characters.

**Mr. Bean:** 19 topics are extracted from 51 topics ($burst\_intensity\_threshold = 0.3$). In the 7-th and 8-th topics in Fig. 3.9 (d), *Bean* runs away from police men. In the 18-th topic, *Bean* rides on a roller coaster. He performs funny actions in the 1-st, 2-nd, 3-rd, 6-th, 12-th, 13-th, 14-th, 16-th and 19-th topics.
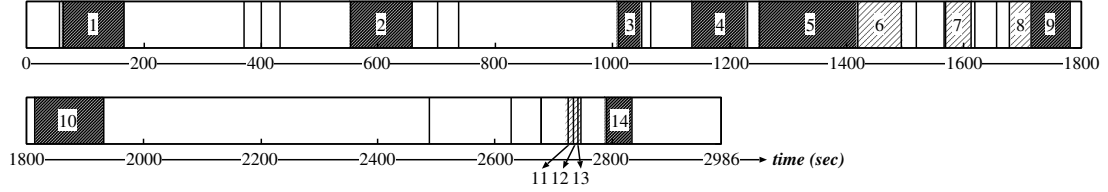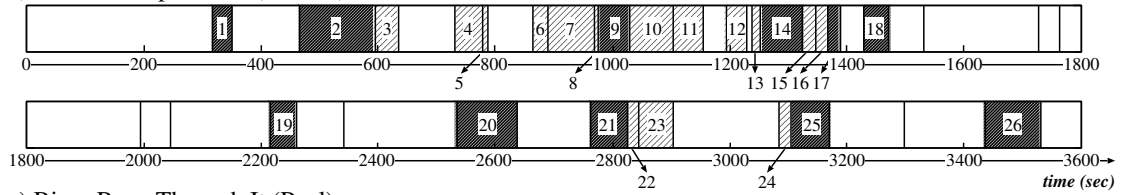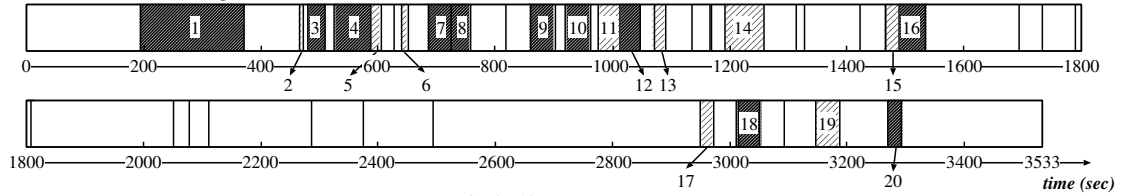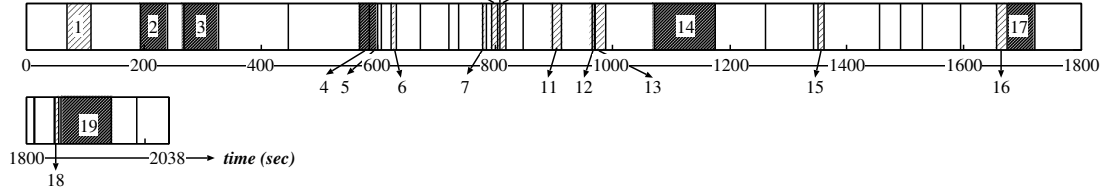
Figure 3.9: Temporal distributions of extracted topics in experimental movies.

We closely investigate topic extraction results using Fig. 3.10, where several extracted topics are depicted in the bar graph representation. Note that numbers assigned to topics in Fig. 3.10 correspond to the ones in Fig. 3.9. In addition, the bold horizontal line in each event represents its burst intensity[4]. Generally, short bursts characterize thrilling topics, such as fighting, chasing, running and so on. For example, the 11-th, 12-th and 13-th topics in Fig. 3.10 (a) are characterized by short bursts, where mean of *Marion*'s appearance durations are 0.8, 0.5 and 0.8 seconds, respectively. She is murdered in these topics.
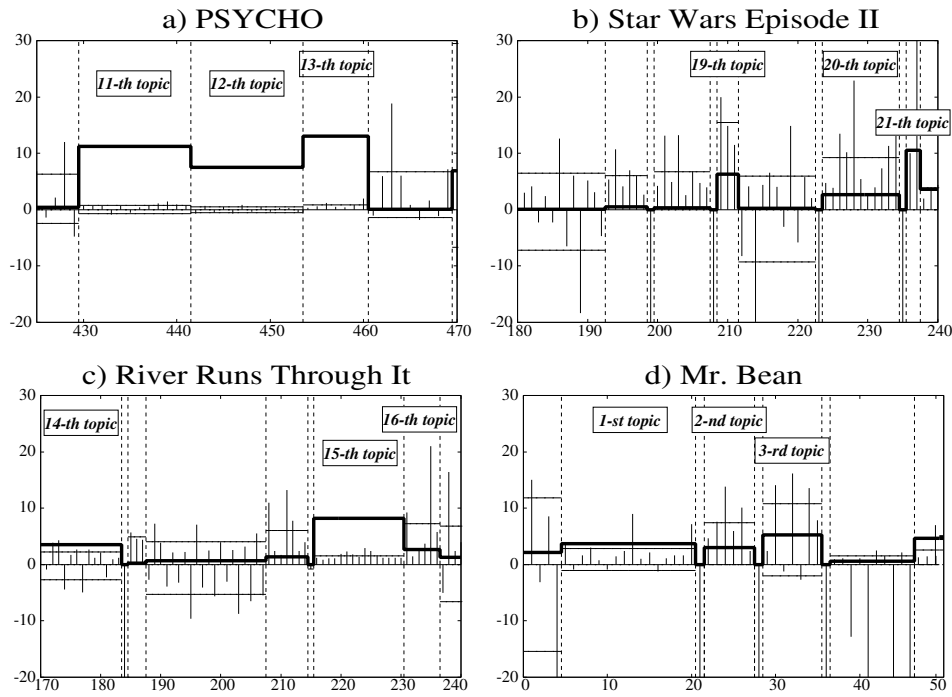


Figure 3.10: Examples of extracted topics.

On the other hand, long bursts characterize the following two types of topics. Typically, the first type includes romantic topics. For example, the 20-th and 21-st topics in Fig. 3.10 (b) are characterized by long bursts, where means of *Anakin*'s appearance durations are quite long, specifically, 9.2 and 31.0 seconds, respectively. In these topics, *Anakin* talks to the woman he

---

[4]For the ease of perception, the actual burst intensity value is multiplied by 10.

loves. In the second type of topics characterized by long bursts, a character's action is carefully presented. For instance, long bursts occur in the 2-nd and 3-rd topics in Fig. 3.10 (d), where means of *Bean*'s appearance durations are 7.4 and 10.8 seconds, respectively. In these topics, *Bean* performs funny actions which are taken by shots with long durations.

From the above discussion, the target character performs interesting actions in extracted topics. It is concluded that by viewing extracted topics, viewers can roughly understand what kind of actions the target character performs in a video. Here, only viewing topics requires less time than viewing the entire video. For example, the total duration of *PSYCHO* in Fig. 3.9 (a) is $2,986$ seconds while the total duration of 14 extracted topics is 915 seconds. Thus, we are currently developing a video browsing/summarization system based on extracted topics.

## 3.6   Summary

In this chapter, we introduced a novel topic extraction method based on a target character's appearance and disappearance in a video. First, we divide the video into events by applying time series segmentation technique to the sequence of intervals of the target character's appearance and disappearance. Each event is characterized by the pattern of the character's appearance and disappearance as well as the occurrence ratio. We then use the burst intensity measure to extract topics as events containing bursts. Two types of bursts are defined where the first one is a short burst characterized by abnormally short appearance durations, while the second type is a long burst characterized by abnormally long appearance durations. The experimental results validate the effectiveness of a character's appearance and disappearance for obtaining semantically meaningful events. Furthermore, burst detection is useful for extracting topics where a target character performs interesting actions. One important future work is the development of a method which can accurately recognize characters in a video.

# Chapter 4

# Video Retrieval by a Small Number of Examples Using Rough Set Theory and Partially Supervised Learning

## 4.1 Introduction

In this chapter, we develop a method which extracts retrieval models based on example shots provided by a user. The motivation behind this is that, the proper representation of a query is very important for video retrieval. Existing approaches can be roughly classified into two types, namely, *Query-By-Keyword* (QBK) and *Query-By-Example* (QBE) approaches. With QBK, a user represents a query by using keywords and shots are subsequently retrieved by matching with defined keywords. With QBE, a user provides example shots to represent a query and shots are then retrieved based on their similarity to example shots in terms of features. For example, consider the query 'people appear with computers' depicted in Fig. 4.1. Assume that a relevant shot, *Shot 1*, is annotated with the words 'people' and 'computer'. To retrieve *Shot 1* using QBK, a user needs to enter the keywords 'people' and 'computer'. However, the keywords 'programmer' or 'internet user' might be entered instead, which would not match the annotated words for *Shot 1*, despite matching the actual query. This ambiguity relating to semantic content makes it difficult for the user to appropriately represent queries

using keywords. Alternatively, with QBE, the query is defined using features contained in example shots, such as *Ex. 1* in Fig. 4.1. This eliminates the ambiguity associated with semantic content found in the QBK approach.
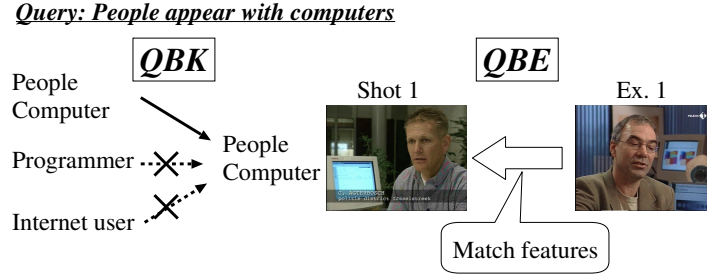


Figure 4.1: Comparison between the QBK and QBE approaches for the query 'people appear with computers'.

The QBE approach offers the added advantage of not requiring predefined retrieval models. In traditional QBK methods, a retrieval model needs to be prepared for each query [96, 23], and recent QBK methods prepare classifiers for assessing the relevance of keywords[1] defining each query [21, 24]. However, it is impractical to prepare retrieval models and classifiers for all possible queries. In comparison, a retrieval model is constructed on the fly from example shots in the QBE approach. In other words, as long as example shots are provided, QBE can perform retrieval for any query. In this chapter, we describe the development of a QBE method.

The following three problems in the QBE method have been addressed.

**1. Large variety of relevant shots:** Shots relevant to a query are taken using different camera techniques and settings. For example, in Fig. 4.2, *Shot 1* depicts the user's hands with the computer monitor in a tight shot. *Shot 2* shows the face of a person with the front of the computer monitor while *Shot 3* shows a computer monitor from the side. *Shot 4* captures the back of a person facing the computer screen. This illustrates that objects related to the same query can be depicted in several ways. Furthermore, relevant shots show numerous objects unrelated to the query. For example, among four shots in Fig. 4.2, the background and peoples' clothing are different and

---

[1]These keywords are frequently called *concepts*. However, some readers may confuse them with concepts which are hierarchically organized in an ontology. In light of this, we do not use the term *concept*.

a caption is visible in *Shot 2*, but not in any other shots. Thus, even for the same query, relevant shots not only contain significantly different features, but also many redundant features. Therefore, a single retrieval model is not capable of retrieving a large variety of relevant shots.

Shot 1      Shot 2      Shot 3      Shot 4



Figure 4.2: An example of a variety of event shots for the query 'people appear with computers'.

*Rough set theory* (RST) is a set-theoretic classification method for extracting rough descriptions of a class from imprecise (or noisy) data, and is used for retrieving a variety of relevant shots [46]. RST can be used to extract multiple classification rules, which can correctly identify different subsets of example shots. In other words, each classification rule is specialized for retrieving relevant shots characterized by certain features. Hence, by accumulating relevant shots retrieved with various classification rules, we can retrieve a variety of relevant shots.

**2. Small sample size:** In QBE, a user can only provide a small number of example shots for a query. Since QBE, by definition, retrieves shots similar to example shots, a small number of example shots will inevitably lead to a small range of relevant shots. We use *bagging* and the *random subspace method* to overcome this problem. Specifically, various classifiers are built using randomly selected example shots and feature dimensions. When only a small number of example shots are available, classifiers output significantly different classification results depending on example shots [28]. In addition, classification results differ depending on feature dimensions [89]. Thus, by building various classifiers using bagging and the random subspace method, we can extend the range of relevant shots that can be retrieved. However, this also results in many irrelevant shots potentially being retrieved. To overcome this, RST is used to extract classification rules as combinations of classifiers, which can accurately retrieve relevant shots.

**3. Lack of negative examples:** RST extracts classification rules to enable the discrimination of relevant shots from irrelevant shots. This requires

two types of example shots, *positive examples* (p-examples), provided by a user to serve as representatives of relevant shots, and *negative examples* (n-examples), which are not provided by the user and serve as representatives of irrelevant shots. To solve the lack of n-examples, we formulate the QBE approach as *partially supervised learning* (PSL). Classification rules are extracted using p-examples and *unlabeled examples* (u-examples), which refer to shots other than p-examples. N-examples are collected from these u-examples. It can be considered that n-examples similar to p-examples are informative, because they allow the characterization of the boundary between event and non-event shots. To collect such n-examples, we devise a PSL method based on the coarse-to-fine approach. Firstly, all u-examples are regarded as n-examples, because the number of relevant shots included in u-examples is usually very small. Subsequently, n-examples which are dissimilar to p-examples, are iteratively filtered using a classifier built on p-examples and the remaining n-examples.

Our proposed QBE method is summarized in Fig. 4.3. It should be noted that our main research objective is to develop a method which can retrieve a variety of relevant shots only by using a small number of p-examples. Given p-examples, n-examples are collected using the PSL method. Subsequently, various classifiers are built based on bagging and the random subspace method. Lastly, RST is used to extract combinations of classifiers as classification rules, and shots matching many rules are retrieved.

## 4.2 Related Works and the Innovation of Our Research

### 4.2.1 Rough Set Theory

Firstly, RST determines the indiscernibility relation, which relates to whether a p-example and n-example can be discerned with respect to available features. Thereafter, multiple classification rules are extracted by combining indiscernibility relations among examples based on set theory. RST can extract classification rules without any assumption or parameter as long as indiscernibility relations can be defined.

Although methods other than RST can be used to retrieve a variety of relevant shots, they have inappropriate limitations. For example, a Gaussian Mixture Model (GMM) can extract multiple feature distributions of relevant
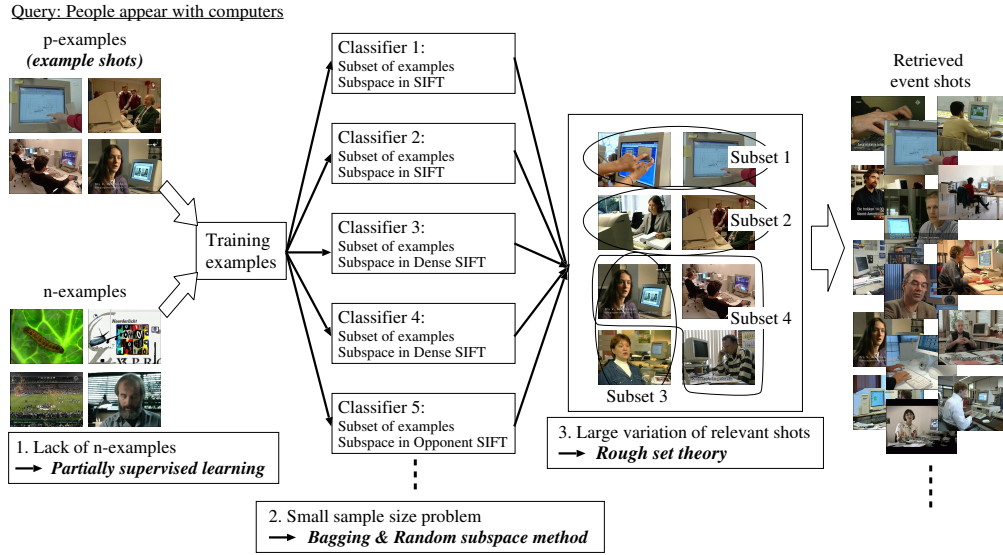
Figure 4.3: An overview of our video retrieval method based on the QBE approach.

shots [55]. However, such distributions cannot be appropriately estimated only from a small number of p-examples. In addition, without any a priori knowledge, the number of Gaussian distributions in the GMM cannot be effectively determined. The Genetic Algorithm (GA) can be used to extract multiple classification rules [42]. Each rule is encoded as a bit string (chromosome), where one bit indicates whether or not a feature is utilized. The GA combines bit strings based on the principles of biological evolution, such as crossover and mutation, to extract accurate rules. However, with no a priori knowledge, parameters in the GA, such as the number of bit strings, the probability of crossover and the probability of mutation, cannot be effectively determined.

Decision tree learning methods extract multiple classification rules in a tree-based approach [31]. Each rule is represented as a path in a tree, where p-examples and n-examples are recursively classified using a feature associated with each node. Sequential covering methods extract multiple rules in a sequential approach [31]. Each rule is sequentially extracted from p-examples, which are not characterized (covered) by already extracted rules. But, the tree-based and sequential approaches only extract one classification

rule for each p-example. As a result, the number of extracted rules is inevitably small, which is insufficient for retrieving a variety of relevant shots. In comparison to the above methods, without any parameter setting, RST can extract various rules as minimal sets of features, which can correctly identifying different subsets of p-examples.

Traditional RST can only deal with categorical features where the indiscernibility relation between two examples can be easily defined according to whether they have the same value. In contrast, in our case, examples are represented by non-categorical and high-dimensional features. For instance, the bag-of-visual-words representation involves thousands of dimensions, each of which indicates the frequency of a local edge shape (visual word). Thus, when applying RST to QBE, the most important issue becomes the definition of indiscernibility relations among examples, that is, the categorization of non-categorical high-dimensional features. With respect to this issue, existing approaches can be classified into the following three types:

**1. Clustering-based:** This approach groups examples into a small number of clusters. The indiscernibility relation between two examples is then defined by examining whether or not they belong to the same cluster [108, 12].

**2. Similarity-based:** This approach does not categorize a feature, but rather defines the indiscernibility relation between two examples by measuring their similarity for the feature [60, 79].

**3. Classifier-based:** This approach builds a classifier on a feature, and defines the indiscernibility relation by examining whether or not two examples are classified into the same class [87].

In our research, we have previously developed the clustering-based and similarity-based RSTs, however, they had performance limitations. In [12], we developed a clustering-based RST using k-means clustering and tested it on TRECVID 2008 video data. TRECVID is an annual international competition where a large video data is used to benchmark state-of-the-art video analysis methods [15]. However, for most queries, inferred average precisions of the clustering-based RST were nearly equal to zero. One main reason for this was the coarseness in categorizing a feature into a small number of clusters, which led to semantically different shots frequently being included in the same cluster.

We also developed a similarity-based RST in [60]. Although its performance was better than that of the clustering-based RST, it was far from the satisfactory. Table 4.1 provides a performance comparison between the similarity-based RST and classifier-based RST. We use the same examples,

and the same features as will be described in section 4.3. In the similarity-based RST, cosine similarity is used as a similarity measure, while the classifier-based RST categorizes features using Support Vector Machines (SVMs) with the Radial Basis Function (RBF) kernel. For each query, the retrieval performance is evaluated as the number of relevant shots within $1,000$ retrieved shots. As seen in Table 4.1, the similarity-based RST is significantly outperformed by the classifier-based RST. The reason for the low performance of similarity-based RST is that it is difficult to appropriately measure similarities among examples for high-dimensional features.

Table 4.1: Comparison of the similarity-based RST and classifier-based RST.

| Query | Tall building | Flame | Computer | Helicopter/plane | Talking |
|---|---|---|---|---|---|
| Similarity-based RST | 46 | 14 | 54 | 17 | 38 |
| Classifier-based RST | 144 | 100 | 150 | 34 | 67 |

Table 4.1 demonstrates the effectiveness of the classifier-based RST. In the following paragraphs, we discuss which classifiers are effective for the QBE approach, where high-dimensional features have to be categorized using only a small number of examples. The existing classifier-based RST uses different types of classifiers, such as the decision tree, nearest neighbor, naive Bayes and maximum entropy [87]. However, they are ineffective for the following two reasons. Firstly, a nearest neighbor is ineffective because the result of the similarity-based RST in Table 4.1 implies that similarity measures do not work well for high-dimensional features. Secondly, the other classifiers rely on probabilistic distributions, such as information gains in decision tree, conditional probabilities in naive Bayes, and entropy models in maximum entropy. These are ineffective because probabilistic distributions estimated from a small number of examples tend to deviate from the true distributions [36].

In contrast, SVMs are effective when only a small number of examples are available as the margin maximization does not require any probability estimation [36]. Additionally, Vapnik [93] theorized that if the number of feature dimensions is large, the generalization error of an SVM is defined by the margin size and properties of examples, such as the diameter of the sphere enclosing examples and the number of examples. That is, from the theoretical perspective, the generalization error of the SVM is independent of the number of feature dimensions if this number is sufficiently large. Furthermore, as

examples are generally not linearly separable, a kernel function is used to transform a high-dimensional feature into a higher-dimensional feature, with the above theory allowing a well generalized SVM to be built independent of the number of feature dimensions. Thus, we develop a classifier-based RST using SVMs, specifically using SVMs with the RBF kernel as it is known to be the most general kernel [20].

## 4.2.2 The Problem of Small Sample Size

As described in section 5.1, bagging and the random subspace method are useful for extending the range of relevant shots that can be retrieved. They are additionally useful for alleviating two important problems related to small sample sizes. The first is the *class imbalance problem*, which refers to the imbalance between the number of p-examples and n-examples, significantly degrading the classification performance [82]. In our technique, numerous n-examples can be collected using the partially supervised learning method. However, the SVM built using a small number of p-examples and all collected n-examples cannot appropriately classify shots into relevant and irrelevant shots. To address this, we use bagging which combines classifiers built on different sets of randomly selected examples [63]. In other words, for a small number of p-examples, an appropriate number of n-examples are randomly selected to build an SVM. However, due to the insufficiency of n-examples, the SVM wrongly classifies many irrelevant shots as relevant. Thus, we combine SVMs built on different sets of randomly selected n-examples in order to consider a variety of n-examples.

The second problem is *overfitting* that a classifier can perfectly classify p-examples and n-examples, but cannot appropriately classify unseen examples. Generally, as the number of feature dimensions increases, the number of examples required to construct a well generalized classifier exponentially increases [11]. This is due to the fact that a class needs to be determined for each combination of values along different dimensions. In our case, we can only use a small number of examples (at most, a hundred of p- and n-examples in total). On the other hand, based on the bag-of-visual-words model, we represent each example as a vector with more than $1,000$ dimensions. As a result, the SVM is overfit to feature dimensions which are specific to p-examples (or n-examples), but are ineffective for characterizing relevant (or irrelevant) shots. Thus, we use the random subspace method, which combines classifiers built on randomly selected feature dimensions [89]. The

original high-dimensional feature is transformed into a lower-dimensional feature, which alleviates building an overfit SVM. By combining such SVMs, a large number of feature dimensions can be considered.

Numerous methods have been proposed to overcome the class imbalance and overfitting. For the class imbalance problem, Japkowicz [76] tested various sampling approaches, such as over-sampling of examples belonging to the minority class, under-sampling of examples belonging to the majority class, and sampling based on a classifier for examining the association of an example to the minority (or majority) class. Akbani et al. [82] used the Synthetic Minority Oversampling TEchnique (SMOTE), which synthetically generates new examples for the minority class based on their similarities with existing examples. Liu et al. [37] presented various feature dimension reduction methods to overcome the problem of overfitting. For example, one method selects feature dimensions by measuring their relative importance based on an ensemble of decision trees, while a second method selects feature dimensions which are both maximally relevant and minimally redundant based on the mutual information between examples and classes. Guo et al. [36] proposed a method which simultaneously achieves dimension reduction and margin maximization in classifier training. As the above methods only select the best subset among examples or feature dimensions, they are not useful for extending the range of relevant shots that can be retrieved, unlike bagging and the random subspace method.

Our application of bagging and the random subspace method is crucially different to that in the previous research [28]. In particular, Tao et al. [28] performed simple majority voting using SVMs built by bagging and the random subspace method. We refer to this majority voting as *Simple_MV*. On the other hand, our method involves majority voting using classification rules, which are extracted as combinations of SVMs by RST. We refer to this as *RST_MV*. Table 4.2 shows a performance comparison between *Simple_MV* and *RST_MV*. For each query, 10 retrieval results are obtained using *Simple_MV* or *RST_MV*. To perform an appropriate comparison, retrieval results are obtained by using the same set of 60 SVMs for both *Simple_MV* and *RST_MV*. The second and fourth rows represent average numbers of relevant shots retrieved by *Simple_MV* and *RST_MV*, respectively. These rows demonstrate that *RST_MV* outperforms *Simple_MV* for all queries. A key reason for this is the difference between SVMs and classification rules. The third and fifth rows represent average numbers of relevant shots which are correctly classified by SVMs and classification rules, respectively, showing

that classification rules are much more accurate than SVMs. This allows *RST_MV* to achieve more accurate retrieval than *Simple_MV*.

Table 4.2: Performance of SVM combination by majority voting and RST.

| | Query | Tall building | Flame | Computer | Helicopter/plane | Talking |
|---|---|---|---|---|---|---|
| Majority voting | Retrieval Performance | 136.0 | 122.6 | 160.5 | 36.1 | 175.6 |
| | SVM Performance | 84.3 | 67.4 | 90.4 | 22.6 | 98.4 |
| RST | Retrieval Performance | 140.6 | 145.3 | 164.3 | 41.5 | 189.9 |
| | Rule Performance | 109.6 | 98.6 | 118.8 | 32.4 | 139.5 |

## 4.2.3 Negative Example Selection

Traditional QBE methods only use p-examples and retrieve shots similar to them [102, 58]. But, only considering similarities to p-examples cannot yield accurate retrieval. For example, consider the query 'a car moves'. If a p-example showing a moving red car is provided, it may seem more similar to a shot where a person wears a red piece of clothing than a shot showing a moving white car. Compared to this, if a QBE method uses n-examples and compares them to the p-example, it can be found that the edge feature characterizing a car shape is important while the color feature is not. Thus, n-examples are required to construct a retrieval model (in our case, classification rules), which characterizes features specific to p-examples.

Since relevant shots form only a small portion of all shots, several methods use an approach which considers randomly selected shots as n-examples [14, 24]. However, this approach is associated with the following problem. The random selection of n-examples does not consider the type of n-examples that are required to construct an accurate retrieval model. For example, if all of collected n-examples are significantly dissimilar to p-examples, we cannot identify features that are relevant to p-examples because p-examples and n-examples can be classified by using any features.

Numerous partially supervised learning (PSL) methods have been proposed to build a classifier using p-examples and unlabeled examples (u-examples). Most of the existing methods adopt the two-step approach,

wherein n-examples are first collected from u-examples and a classifier is subsequently built on p-examples and collected n-examples [34, 18, 40]. In such a PSL approach, one of main research issues is how to collect n-examples from u-examples. For instance, Liu et al. [18] proposed a method which selects some p-examples as 'spy' examples and adds them to u-examples. A naive Bayesian classifier is then built using p-examples and u-examples, where spy examples are used to set the probabilistic threshold for evaluating whether or not u-examples can be regarded as negative. Yu et al. [40] proposed a method which iteratively collects n-examples. In each iteration, the method builds an SVM from p-examples and already selected n-examples, and subsequently selects n-examples as u-examples, which are classified as negative by the SVM. Fung et al. [34] proposed an iterative method where u-examples which are more similar to already selected n-examples rather than p-examples are selected as n-examples.

However, most of existing PSL methods only collect a large number of n-examples. Due to the class imbalance problem, the use of all collected n-examples does not guarantee the construction of an accurate classifier. To overcome this, we propose a new PSL method which can select a small number of n-examples useful for building an accurate classifier like SVM. Our method is inspired by the method proposed in [57]. It was originally developed to solve the class imbalance problem by under-sampling examples of the majority class, i.e., n-examples. [57] contends that since n-examples similar to p-examples characterize the class boundary, they are useful for building an accurate SVM. N-examples are iteratively filtered to collect those n-examples that are similar to p-examples. In each iteration, the method filters out n-examples that are distant from the decision boundary of the SVM, which is built on p-examples and the remaining n-examples. As a result, the method selects a small number of n-examples that are as similar to p-examples as possible. We apply this method to PSL by regarding all u-examples as n-examples, because almost all of u-examples are irrelevant to a query.

## 4.3 Video Retrieval Method

In this section, the proposed QBE method is described. We assume that the representative semantic content in each shot is shown in the middle video frame, called *keyframe*. The following 6 types of features are extracted from

the keyframe in each shot using the color descriptor software [59]: *1. SIFT*, *2. Opponent SIFT*, *3. RBG SIFT*, *4. Hue SIFT*, *5. Color histogram* and *6. Dense SIFT*. The first three types of SIFT features are defined in different color spaces and have different invariance properties for lighting conditions. By using these three SIFT features, we aim to characterize local shapes of objects (e.g. corners of buildings, vehicles, human eyes etc.), regardless of changes in illumination. Since they only compute derivatives in color spaces, they do not consider absolute color values which are effective for characterizing certain objects, such as faces, flames, vegetation and so on. So, we use *Hue SIFT* and *Color histogram*. *Hue SIFT* represents a SIFT feature concatenated with a Hue histogram in HSV color space, and *Color histogram* represents a pure RGB color histogram. All of the above 5 types of features are extracted at interest points computed by Harris-Laplace detector, where pixel values largely change in multiple directions. But, small changes of pixel values are critical when considering certain objects such as sky, water, roads and so on. In addition, many interest points in the background may be missed due to low contrast. Thus, *Dense SIFT* is used, wherein SIFT features are extracted at interest points sampled with a fixed interval. In this way, interest points missed by Harris-Laplace detector can be compensated. Due to space limitations, the above features are not discussed any further and can be found in [59] in greater detail.

The above 6 types of features are represented using the bag-of-visual-words representation. For each type of feature, $1,000$ visual words are extracted by clustering features at $200,000$ interest points, sampled from keyframes in TRECVID 2009 development videos (219 videos) [15]. As depicted in Fig. 4.4, a shot is represented as a $6,000$-dimensional vector where each type of feature is represented as a $1,000$-dimensional vector. Based on this high-dimensional representation, we perform rough set theory extended by bagging and the random subspace method, and partially supervised learning. It should be noted that for the random subspace method, it is unreasonable to build an SVM using dimensions randomly sampled across different types of features. Thus, we build one SVM by randomly selecting dimensions in the same type of feature.
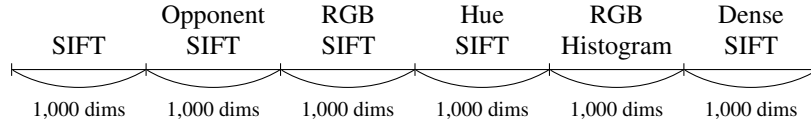
| SIFT | Opponent SIFT | RGB SIFT | Hue SIFT | RGB Histogram | Dense SIFT |
|---|---|---|---|---|---|
| 1,000 dims | 1,000 dims | 1,000 dims | 1,000 dims | 1,000 dims | 1,000 dims |

Figure 4.4: The representation of a shot as a 6,000-dimensional vector.

## 4.3.1 Rough Set Theory Extended by Bagging and the Random Subspace Method

Given p-examples and n-examples for a query, RST is used to extract classification rules, known as *decision rules*, for distinguishing relevant shots from irrelevant shots. Firstly, various SVMs are built using bagging and the random subspace method. Using each of these SVMs, the indiscernibility relation between a p-example and n-example is defined by determining whether or not the p-example and n-example are classified into the same class. Lastly, by combining such indiscernibility relations among examples, decision rules, which can discriminate subsets of p-examples from the entire set of n-examples, are extracted. The proposed RST is explained in detail in the following paragraphs.

First of all, the number of p-examples and that of n-examples are too small to characterize the distribution of relevant shots and that of irrelevant shots in a high-dimensional feature space, respectively. So, the decision boundary of an SVM tends to be inaccurate. Generally, an SVM determines the class of an example based on the binary criterion, i.e., whether the example is located on the positive or negative side of the decision boundary. However, this classification is erroneous since the decision boundary is inaccurate. To overcome this, a continuous-valued criterion is employed. Specifically, the *probabilistic output* of the SVM, which approximates the distance between an example and the decision boundary using a sigmoid function, is used [38]. Based on this, the class of an example is determined as follows. Firstly, examples are ranked in descending order of probabilistic outputs of the SVM. If an example is ranked within the top $M$ positions, where $M$ is the number of p-examples, its class is determined as positive, or otherwise as negative. Thus, although the decision boundary is inaccurate, examples can be robustly classified.

Classification results of SVMs can be summarized in a *decision table*, as

shown in Fig. 4.5. Each row represents the $i$-th p-example, $p_i$ $(1 \leq i \leq M)$ or $j$-th n-example, $n_j$ $(1 \leq j \leq N)$. Each column $a_k$ $(1 \leq k \leq K)$ represents the classification result of the $k$-th SVM, where $+1$ indicates that an example is classified as positive and $-1$ indicates that it is classified as negative. The classification result of the $k$-th SVM for $p_i$ and $n_j$ are represented by $a_k(p_i)$ and $a_k(n_j)$, respectively. That is, each classification result can be regarded as one feature in RST. Lastly, the rightmost column indicates whether an example is positive ($P$) or negative ($N$).

| 1-st SVM | 2-nd SVM | 3-rd SVM | 4-th SVM |
|---|---|---|---|
| Subset of examples | Subset of examples | Subset of examples | Subset of examples |
| $p_1$ $p_3$ ··· $n_2$ $n_5$ ··· | $p_3$ $p_4$ ··· $n_5$ $n_7$ ··· | $p_3$ $p_7$ ··· $n_1$ $n_8$ ··· | $p_1$ $p_2$ ··· $n_4$ $n_6$ ··· |
| Subset of dimensions in SIFT | Subset of dimensions in SIFT | Subset of dimensions in Opp. SIFT | Subset of dimensions in Opp. SIFT |
| 1-th, 8-th, ··· , 995-th | 5-th, 13-th, ··· , 990-th | 2-th, 5-th, ··· , 992-th | 4-th, 9-th, ··· , 987-th |

|  | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | ········· | Class |
|---|---|---|---|---|---|---|---|
| $p_1$ | +1 | +1 | -1 | +1 | -1 | ········· | P |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ |
| $p_M$ | -1 | +1 | +1 | +1 | -1 | ········· | P |
| $n_1$ | -1 | -1 | -1 | +1 | -1 | ········· | N |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ |
| $n_N$ | -1 | -1 | -1 | +1 | +1 | ········· | N |

Figure 4.5: Example of a decision table formed by applying RST extended by bagging and the random subspace method.

Before discussing the proposed decision rule extraction method, a conceptual explanation in relation to decision rules is provided using Fig. 4.6. In this figure, one p-example $p_1$ and two n-examples $n_1$ and $n_2$ are given for the query 'tall buildings are shown'. Let $a_{k_1}$ and $a_{k_2}$ represent the classification of the $k_1$-th SVM built on *SIFT* feature and that of the $k_2$-th SVM built on *Hue SIFT* feature, respectively. Since *SIFT* feature of $p_1$ is similar to the one of $n_1$, the $k_1$-th SVM incorrectly classifies both as positive. On the other hand, since *Hue SIFT* feature of $p_1$ is dissimilar to that of $n_1$, the $k_2$-th SVM correctly classifies $p_1$ and $n_1$ as positive and negative, respectively. The

dissimilarity between *SIFT* features of $p_1$'s and $n_2$'s enables them to be correctly classified by the $k_1$-th SVM. On the other hand, $p_1$ and $n_2$ cannot be correctly classified by the $k_2$-th SVM due to their similar *Hue SIFT* features. Thus, in order to discriminate $p_1$ from $n_1$ and $n_2$, a decision rule consisting of $a_{k_1}$ and $a_{k_2}$ is required. This rule indicates the combination of the $k_1$-th and $k_2$-th SVMs. In the following paragraphs, the extraction of such decision rules based on a logical operation is described.
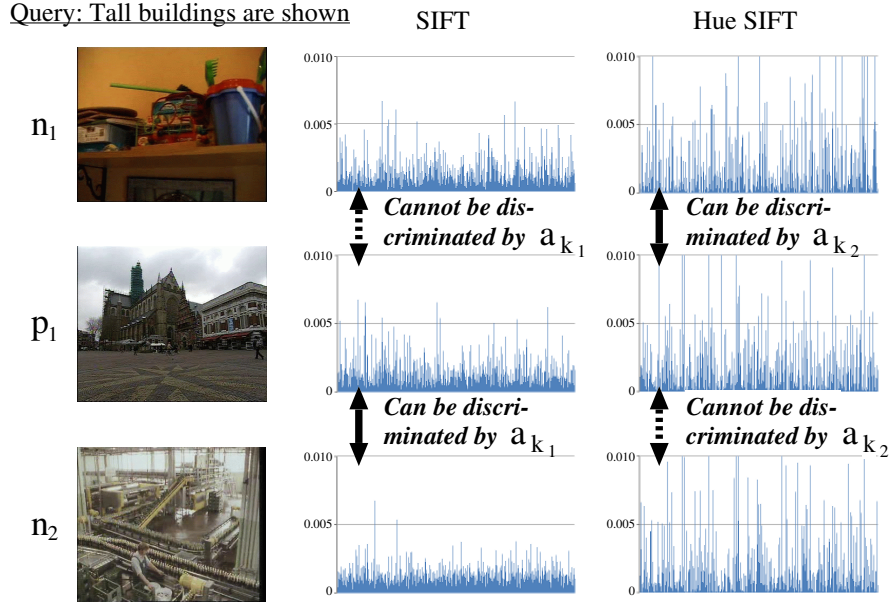


Figure 4.6: The concept of decision rules extracted by RST.

In order to extract decision rules, for each pair of $p_i$ and $n_j$, we first determine *discriminative features* which are useful for discriminating them. Specifically, the following set of discriminative features $f_{i,j}$ are extracted between $p_i$ and $n_j$.

$$f_{i,j} = \{a_k | a_k(p_i) = +1 \land a_k(p_i) \neq a_k(n_j)\} \tag{4.1}$$

This signifies that $a_k$ is included in $f_{i,j}$, if the $k$-th SVM correctly classifies $p_i$ and $n_j$ as positive and negative, respectively. Thus, $p_i$ can be discriminated from $n_j$ when at least one feature in $f_{i,j}$ is utilized.

Next, in order to discriminate $p_i$ from all n-examples, the discriminative features of $p_i$'s are combined. This is achieved by using at least one discriminative feature in $f_{i,j}$ for all n-examples. The *discernibility function $df_i$*, which takes a conjunction of $\vee f_{i,j}$, is computed as follows:

$$df_i = \wedge \{ \vee f_{i,j} | \ 1 \leq j \leq N \} \tag{4.2}$$

Let us consider the discernibility function $df_1$ for one p-example $p_1$ and two n-examples $n_1$ and $n_2$. Let the sets of discriminative features between $p_1$ and $n_1$ and between $p_1$ and $n_2$ be $f_{1,1} = \{a_1, a_3, a_5\}$ and $f_{1,2} = \{a_1, a_2\}$, respectively. Under this condition, $df_1$ can be computed as $(a_1 \vee a_3 \vee a_5) \wedge (a_1 \vee a_2)$. This is simplified as $df_1^* = (a_1) \vee (a_2 \wedge a_3) \vee (a_2 \wedge a_5)$ [2]. Thus, $p_1$ can be distinguished from $n_1$ and $n_2$, by using $a_1$, the set of $a_2$ and $a_3$, or the set of $a_2$ and $a_5$. Similarly, each conjunction term in $df_i^*$ represents a *reduct* which is the minimal set of features required to discriminate $p_i$ from all n-examples.

From each reduct, we construct a decision rule in the form of an *IF-THEN* rule. A reduct $r$, consisting of $L$ features, can be represented as follows.

$$r = a_{1*} \wedge a_{2*} \wedge \cdots \wedge a_{L*} \tag{4.3}$$

where $a_{l*}$ ($1 \leq l^* \leq L$) denotes a single feature among the total $K$ features $a_1, \cdots, a_K$. Recall that in equation (4.1), $a_k$ is selected as a discriminative feature only when $a_k(p_i) = +1$. Thus, the decision rule, *rule*, constructed from $r$ has a conditional part, where $a_{l*}(x)$ has to be $+1$ for a shot (i.e. unseen example) $x$. It is represented as follows.

$$rule : IF \ (a_{1*}(x) = +1) \wedge \cdots \wedge (a_{L*}(x) = +1), \ THEN \ Class = P \tag{4.4}$$

For example, from the reduct $(a_2 \wedge a_3)$, we can construct the decision rule $IF \ (a_2(x) = +1) \ \wedge \ (a_3(x) = +1), \ THEN \ Class = P$. This rule indicates that if both the 2-nd and 3-rd SVMs classify $x$ as positive, then $x$ is relevant to the query. Like this, a decision rule describes how to combine SVMs built by bagging and the random subspace method for retrieving relevant shots.

When matching $x$ with *rule*, we regard decision boundaries of SVMs as inaccurate, as described above. Decision rule matching is conducted based on

---

[2]This simplification is achieved by using the distributive law $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ and the absorption law $A \vee (A \wedge B) = A$. Although the simplification of a Boolean function is NP-hard, we can obtain an approximate solution by using the genetic algorithm [46].

probabilistic outputs of SVMs. To explain this, $SVM_{l^*}$ is used to represent the $l^*$-th SVM corresponding to each feature $a_{l^*}$ in $rule$ ($1 \leq l^* \leq L$). In addition, the probabilistic output of the $SVM_{l^*}$ for $x$ is denoted by $Prob_{SVM_{l^*}}(x)$. Based on the above notations, we calculate $Match(x, rule)$ which is an evaluation value of matching $x$ with $rule$.

$$Match(x, rule) = \prod_{l^*=1}^{L} Prob_{SVM_{l^*}}(x) \qquad (4.5)$$

$Match(x, rule)$ is the joint probability of $L$ SVMs in $rule$. It should be noted that a threshold is required to determine whether or not $x$ matches $rule$. In view of this, the following two considerations must be made. Firstly, feature numbers (i.e. $L$) differ depending on decision rules. Secondly, the distribution of SVM's probabilistic outputs differs depending on the sigmoid function estimated for the SVM. For example, the mean of some SVM's probabilistic outputs is 0.5, while the one of a different SVM's probabilistic outputs is 0.3. Thus, it is unreasonable to use the same threshold for all decision rules.

Instead of actual values of $Match(x, rule)$, the following ranking approach is used to determine whether or not $x$ matches $rule$. First, all shots are ranked in descending order of $Match(x, rule)$. A shot ranked within the top $T$-th position is considered to match $rule$ ($T$ is set to $1,000$ in all experiments). In this way, shots are matched with all decision rules based on the same criterion of the ranking position. Finally, our method outputs a retrieval result consisting of $T'$ shots which match the largest numbers of decision rules ($T' = 1,000$ in our experiments).

## 4.3.2 Effectiveness of Bagging and the Random Subspace Method

In this section, we discuss whether bagging and the random subspace method are effective in extending the range of relevant shots that can be retrieved. That is, we examine differences among classification results of SVMs, which are built using different examples and feature dimensions.

Table 4.3 shows experimental results obtained for five queries that will be used in section 4.4. In particular, the objective is to examine whether classification results of SVMs change when using different examples and dimensions, even for the same type of feature. Thus, SVMs are built only by

using *SIFT* feature. In the second to fourth rows of Table 4.3, we only use bagging where SVMs are built using the same number of randomly selected examples. On the other hand, in the fifth to seventh rows of Table 4.3, we use both bagging and the random subspace method where SVMs are built using the same number of randomly selected examples and *SIFT* dimensions. A comparison is drawn among classification results of 10 SVMs by examining $1,000$ shots with the highest probabilistic outputs among the total $97,150$ shots. We define the first classification result as the *baseline*, and examined the difference between the baseline and the remaining nine results, called *comparison results*. The second (or fifth) row represents the number of relevant shots included in the baseline. The third (or sixth) row represents the average number of shots that are only included in comparison results. Lastly, the fourth (or seventh) row represents the average number of relevant shots included only in comparison results.

Table 4.3: Differences among classification results built using bagging and the random subspace method.

| Query | | *Query 1* | *Query 2* | *Query 3* | *Query 4* | *Query 5* |
|---|---|---|---|---|---|---|
| | *Baseline:* # of relevant shots | 124 | 99 | 111 | 35 | 146 |
| Bagging | *Comparison:* # of different shots | 466.9 | 579.7 | 470.1 | 396.6 | 512.8 |
| | *Comparison:* # of different relevant shots | 39.4 | 16.7 | 34.3 | 7.3 | 42.2 |
| Bagging & Random subspace | *Baseline:* # of relevant shots | 129 | 97 | 115 | 22 | 150 |
| | *Comparison:* # of different shots | 461.3 | 695.4 | 656.4 | 428.4 | 693 |
| | *Comparison:* # of different relevant shots | 27.1 | 16.7 | 37.2 | 11.9 | 53.7 |

As can be seen from Table 4.3, by only changing examples based on bagging, comparison results include approximately 397 to 580 shots that differ from the baseline. In addition, compared to relevant shots in the baseline, about 16% to 32% of different relevant shots are included in comparison results. Furthermore, by changing both examples and feature dimensions with bagging and the random subspace method, comparison results include about 428 to 695 shots that are different from the baseline, and approximately 17% to 36% of relevant shots in comparison results are different from relevant shots in the baseline. This indicates that bagging and the random subspace

method are effective in extending the range of relevant shots that can be retrieved. However, Table 4.3 indicates that as the number of relevant shots which can be retrieved increases, the number of irrelevant shots also increases. It can be seen that the ratio of relevant shots to those shots included only in comparison results is less than 10%. Thus, in order to accurately retrieve relevant shots, decision rules need to be extracted as combinations of SVMs by using RST.

## 4.3.3 Partially Supervised Learning

Since the proposed RST relies on SVMs built using bagging and the random subspace method, it is necessary to collect n-examples that are useful for building accurate SVMs. In particular, considering the class imbalance problem, the proposed partially supervised learning (PSL) method should be able to collect a small number of *informative* n-examples from unlabeled examples (u-examples). N-examples similar to p-examples are considered as informative, because they are useful for characterizing the boundary between relevant and irrelevant shots. The procedure involved in selecting a small number of informative n-examples is described below.

The proposed PSL method is summarized in Algorithm 1. Firstly, since the number of relevant shots included in u-examples is very small, all u-examples are assumed to be n-examples. The set of p-examples and the set of n-examples are denoted by $P$ and $N$, respectively. The number of p-examples and the one of n-examples are represented by $|P|$ and $|N|$, respectively. Based on $P$ and $N$, an SVM, which examines whether or not n-examples are informative, is built. However, only a small number of n-examples can be used due to the class imbalance problem. If n-examples are randomly selected from $N$, n-examples located in certain regions of the feature space may not be selected. As a result, the decision boundary of the SVM is wrongly estimated, and it is not possible to appropriately evaluate the informativeness of n-examples. Thus, we need to collect a set of *representative* n-examples, which characterize the distribution of all n-examples. To this end, n-examples are grouped into clusters using the k-means clustering algorithm and the Euclidian distance measure. As shown in the line 1 of Algorithm 1, n-examples are grouped into $N/\beta$ clusters. $\beta$ is a pre-defined parameter used to control the number of clusters relative to the number of n-examples. Since various semantic contents are presented in n-examples, their features are very diverse. This necessitates many clusters of n-examples. In

---

**Algorithm 1** An overview of the proposed partially supervised learning.

INPUT: set of p-examples $P$, set of n-examples $N$, desirable number of n-examples $\alpha$, ratio between n-examples and clusters $\beta$

OUTPUT: shrunk set of n-examples $N$

**repeat**
   1. Cluster $N$ into $N/\beta$ clusters
   2. Obtain the set of representative n-examples $RN$, each of which is centrally located in a cluster
   3. Build an SVM using $P$ and $RN$
   4. From $N$, remove those n-examples distant from the decision boundary of the SVM
**until** $|N| \leq \alpha$ OR no n-example is removed from $N$
5. Output $N$

---

our experiment, $\beta$ is set to 10, so that when $|N| = 30,000$, one can obtain $3,000$ clusters. In addition, since it is difficult to appropriately measure similarities among n-examples in the $6,000$-dimensional feature space defined in Fig. 4.4, our PSL method is conducted on $1,000$-dimensional *SIFT* feature (the SVM in the line 3 of Algorithm 1 is also built on *SIFT* feature).

After clustering, for each cluster $c$, the most centrally located n-example is selected as the representative n-example $n_c$.

$$n_c = \min_{n_i \in N_c} \sum_{n_j \in N_c} dist(n_i, n_j) \tag{4.6}$$

where $N_c$ is the set of n-examples in $c$, and $n_i$ and $n_j$ are n-examples in $N_c$. $dist(n_i, n_j)$ represents the Euclidian distance between $n_i$ and $n_j$. Thus, $n_c$ is selected as the n-example having the minimum sum of Euclidian distances to the other n-examples in $c$. A set of representative n-examples for all clusters is denoted by $RN$.

By using an SVM built on $P$ and $RN$, it can be determined whether each n-example $n$ in $N$ is informative based on its distance from the decision boundary of the SVM. N-examples distant from the decision boundary are uninformative for defining the boundary between relevant and irrelevant shots. The above test is conducted using the following criterion:

$$|w^T n + b| > \gamma \tag{4.7}$$

Using $x$ as an arbitrary example, $w^T x + b = 0$ represents the decision boundary (hyperplane) of the SVM. Based on this, $|w^T n + b|$ can characterize the distance between $n$ and the decision boundary. Specifically, the distance is $|w^T n + b| / ||w||$, but since $||w||$ is constant, it can be omitted. If the distance between $n$ and the decision boundary is larger than the threshold $\gamma$, $n$ is regarded as uninformative and subsequently removed from $N$. This removal of uninformative n-examples is iterated until the number of n-examples is less than the pre-defined number of n-examples $\alpha$ or no further n-examples are removed from $N$.

An example illustrating the above iteration is shown in Fig. 4.7, with circles and crosses representing p-examples and n-examples, respectively. N-examples are initially collected as shown in Fig. 4.7 (a), and the first iteration is then performed as shown in Fig. 4.7 (b). N-examples are grouped into four clusters and an SVM is built using representative n-examples from these clusters. As a result, two p-examples and two representative n-examples are extracted as support vectors, as depicted by the solid lines. The dashed line represents the decision boundary of the SVM. Based on this, n-examples located on the left side of the bold line are regarded as uninformative and are subsequently discarded. The second iteration is then performed using the remaining n-examples, as shown in Fig. 4.7 (c). In this iteration, an SVM is built using three representative n-examples. Based on the decision boundary of this SVM, n-examples located on the left side of the bold line are discarded. In this way, a small number of n-examples, which are highly similar to p-examples, can be obtained. Such n-examples are useful for characterizing the boundary between relevant and irrelevant shots.
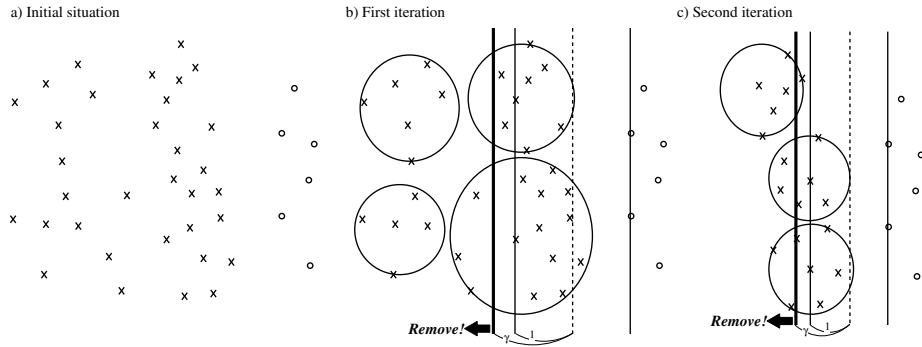


Figure 4.7: An example of the proposed partially supervised learning method.

# 4.4 Experimental Results

In this section, the proposed QBE method is tested on TRECVID 2009 video data [15]. This data consists of 219 development and 619 test videos in various genres, like cultural, news magazine, documentary and education programming. Each video is already divided into shots by using an automatic shot boundary detection method, where development and test videos include $36,106$ and $97,150$ shots, respectively (the detailed video data specification can be found on TRECVID 2009 Web site[3]). The proposed method is evaluated based on the following five queries:

*Query 1:* A view of one or more tall buildings and the top story visible

*Query 2:* Something burning with flames visible

*Query 3:* One or more people, each at a table or desk with a computer visible

*Query 4:* An airplane or helicopter on the ground, seen from outside

*Query 5:* One or more people, each sitting in a chair, talking

Each retrieval is conducted as follows. Firstly, p-examples are manually collected from development videos. Based on p-examples, n-examples are then collected using the proposed PSL method. Considering the class imbalance problem, the number of collected n-examples is set to be equal to five times the number of p-examples. Afterward, the proposed RST method extended by bagging and the random subspace method, is run where two libraries, LIBSVM [20] and ROSETTA [46], are used for SVM learning and for the reduct extraction in RST, respectively. SVM parameters are determined using 3-fold cross validation. Lastly, the retrieval performance is evaluated based on the number of relevant shots within $1,000$ retrieved shots.

## 4.4.1 Effectiveness of Rough Set Theory extended by Bagging and the Random Subspace Method

To examine the effectiveness of RST extended by bagging and the random subspace method, a comparison is drawn among the four types of retrieval described below.

---

[3]http://www-nlpir.nist.gov/projects/tv2009/tv2009.html

*Baseline:* For each of six features, an SVM is built using all examples and dimensions, and a search of test videos is conducted. The SVM which yields the best result is then manually selected. In other words, *Baseline* represents a favorable retrieval result under the ideal condition in which the best feature for the query can be selected.

*RST_only:* RST, which uses neither bagging nor the random subspace method, is executed. One SVM is built for each feature using all examples and dimensions.

*RST+BG:* RST is executed only by using bagging. For each feature, three SVMs are built using different subsets of examples and all dimensions. Each subset is constructed by randomly sampling 75% of examples.

*RST+BG+RS:* RST which incorporates both bagging and the random subspace method is executed. For each feature, 10 SVMs are built using different subsets of examples and dimensions. Each subset of examples is formed by randomly sampling 75% of examples, while each subset of dimensions is formed by randomly sampling 50% of dimensions.

Table 4.4 shows performances of the above four types of retrieval. For each query, the second row presents the number of p-examples. For the performance evaluation, we consider that retrieval results of the proposed method differ due to the following two random factors. The first is attributed to the fact that, the PSL method often terminates before the number of n-examples is reduced to the specified number, because there are no n-examples which can be filtered out (refer to the stopping criterion in Algorithm 1). In such a case, from the remaining n-examples, five times as many n-examples as p-examples are randomly selected. The second random factor is associated with bagging and the random subspace method, where examples and feature dimensions are randomly selected. Thus, in Table 4.4, each row labeled '# of relevant shots' indicates the mean number of relevant shots in 10 retrieval results. Similarly, rows labeled '# of decision rules' and '# of average precision' indicate the mean number of decision rules extracted by RST and the mean of average precisions, respectively.

In Table 4.4, the numbers in bold fonts indicate that *RST_only*, *RST+BG* or *RST+BG+RS* can retrieve more relevant shots than *Baseline*. Since the performance of *RST_only* for *Query 3*, *Query 4* and *Query 5* is lower than

Table 4.4: Performance comparison of *Baseline*, *RST_only*, *RST+BG* and *RST+BG+RS*.

| | Query | Query 1 | Query 2 | Query 3 | Query 4 | Query 5 |
|---|---|---|---|---|---|---|
| | # of p-examples | 100 | 46 | 61 | 40 | 124 |
| *Baseline* | # of relevant shots | 161.1 | 98.9 | 163.0 | 42.7 | 151.0 |
| *RST_only* | # of relevant shots | **165.1** | **111.0** | 161.5 | 34.1 | 146.9 |
| | # of decision rules | 5.5 | 5.2 | 5.4 | 4.5 | 3.8 |
| | Average precision | 0.0931 | 0.1449 | 0.0871 | 0.0094 | 0.0365 |
| *RST+BG* | # of relevant shots | **170.1** | **137.3** | **176.7** | 38.2 | **196.4** |
| | # of decision rules | 252.9 | 159.7 | 151.9 | 137.4 | 354.0 |
| | Average precision | 0.1148 | 0.1827 | 0.0877 | 0.0154 | 0.0571 |
| *RST+BG+RS* | # of relevant shots | **172.0** | **147.5** | **176.9** | 41.6 | **194.6** |
| | # of decision rules | 6159.2 | 1797.6 | 2286 | 1822.9 | 11455.6 |
| | Average precision | 0.1170 | 0.1942 | 0.0992 | 0.0159 | 0.0604 |

that of *Baseline*, it is not necessarily effective. One main reason for this ineffectiveness of *RST_only* is that the number of extracted decision rules is very small. On the other hand, except for *Query 4* (which will be discussed later), both *RST+BG* and *RST+BG+RS* outperform *Baseline* where a greater number of decision rules are extracted compared to *RST_only*. Thus, bagging and the random subspace method are useful for building various SVMs, which enables the extraction of decision rules covering a large variety of relevant shots. Lastly, when making a comparison between *RST+BG* and *RST+BG+RS*, numbers of retrieved relevant shots are not significantly different from each other. For all queries, average precisions of *RST+BG+RS* are higher than those of *RST+BG*, implying that relevant shots are ranked at higher positions in a retrieval result by *RST+BG+RS* than by *RST+BG*. Thus, *RST+BG+RS* is considered to be superior to *RST+BG*.

For *Query 4*, one main reason for the ineffectiveness of *RST+BG* and *RST+BG+RS* is the difficulty of accurately recognizing airplanes and helicopters. Specifically, SVMs built by bagging and the random subspace method wrongly classify many shots showing cars, trains, ships etc. as positive, because their shapes are relatively similar to those of airplanes and helicopters. Thus, combining such inaccurate SVMs into decision rules degrades the retrieval performance.

## 4.4.2   Effectiveness of Partially Supervised Learning

In this section, we examine the effectiveness of our PSL method. To this end, the performance using n-examples collected by our PSL method is compared to the one using n-examples which are randomly collected from all of u-examples. For the simplicity, we call the former and the latter types of n-examples *PSL* n-examples and *random* n-examples, respectively. In Table 4.5, we run *RST+BG* and *RST+BG+RS* using the same p-examples in Table 4.4. PSL n-examples are used in the second and third rows while random n-examples are used in the fourth and fifth rows. Each performance in Table 4.5 is evaluated as the average number of relevant shots in 10 retrieval results.

Table 4.5: Comparison between the retrieval performance using our PSL method and the one using the random n-example selection.

| Query | | *Query 1* | *Query 2* | *Query 3* | *Query 4* | *Query 5* |
|---|---|---|---|---|---|---|
| *PSL* | *RST+BG* | **170.1** | **137.3** | **176.7** | 38.2 | **196.4** |
| | *RST+BG+RS* | **165.1** | **147.5** | **176.9** | 41.6 | **194.6** |
| *Random* | *RST+BG* | 157.9 | 136.0 | 168.5 | 46.6 | 187.5 |
| | *RST+BG+RS* | 159.6 | 144.3 | 171.0 | 47.0 | 187.8 |

As can be seen from Table 4.5, except for *Query 4*, the retrieval using PSL n-examples is more accurate than the one using random n-examples. The reason for the ineffective performance in *Query 4* is as follows. Due to the difficulty of accurately recognizing helicopters and airplanes, few edges which characterize the sky and runways in the background, are important for *Query 4*. But, since PSL n-examples are highly similar to p-examples, most of them are characterized by few edges. So, several relevant shots with few edges are inevitably excluded from the retrieval result. Thus, it can be said that our PSL method works well for queries involving objects, which can be relatively accurately recognized. For example, for *Query 1*, the net of a soccer goal, iron bars, closet etc. are shown in PSL n-examples, because shapes of these objects are similar to buildings. By comparing such PSL n-examples to p-examples, a precise boundary between relevant and irrelevant shots can be extracted. Finally, since random n-examples are currently used in almost all of state-of-the-art methods [14, 21, 24], we believe that our PSL method is novel in the sense that it can outperform the random n-example selection.

### 4.4.3 Effectiveness for the Small Sample Size Problem

The performance of the proposed method is also tested in the case where only a small number of p-examples are available. Fig. 4.8 illustrates the difference in the retrieval performance of *Baseline*, *RST+BG* and *RST+BG+RS*, depending on p-example numbers. For a specified p-example number, we construct three different sets of examples in the following way. Firstly, the specified number of p-examples are randomly collected from all available p-examples, as shown in Table 4.4. Then, n-examples are collected using our PSL method. The performance is evaluated as the average number of relevant shots in retrieval results using the above three sets of examples.

In Fig. 4.8, for *Query 2* and *Query 3*, *RST+BG* and *RST+BG+RS* always outperform *Baseline*. For *Query 1* and *Query 5*, when only 10 p-examples are available, *RST+BG* and *RST+BG+RS* are outperformed by *Baseline*. In this case, most of SVMs are inaccurate and as a consequence, decision rules as combinations of these SVMs are also inaccurate. Finally, for *Query 4*, *RST+BG* and *RST+BG+RS* are always outperformed by *Baseline*, due to the difficulty of accurately recognizing airplanes and helicopters described in the previous section. To summarize the overall performance, except for *Query 4*, *RST+BG* and *RST+BG+RS* become more appropriate than *Baseline* when greater than 20 p-examples are available.

Finally, it is easy to collect more than 20 p-examples for *Query 1* and *Query 5* as relevant shots are often seen in videos. However, relevant shots to *Query 2* are rarely seen and there are only a limited number of videos containing these relevant shots. This increases the difficulty of collecting more than 20 p-examples for *Query 2*. A sufficient number of p-examples may be obtained by retrieving images and videos on the web using online image/video search engines such as Flickr and YouTube.

### 4.4.4 Performance Comparison

A comparison is made between the performance of the proposed method and those of state-of-the-art methods. *RST+BG+RS* is specially compared with methods developed in TRECVID 2009 search task [15]. This task consists of three categories, namely, the fully automatic, manually-assisted and interactive categories. Given the textual description of a query and some p-examples, methods in the fully automatic category retrieve relevant shots without any user intervention. In the manually-assisted category, a user in-
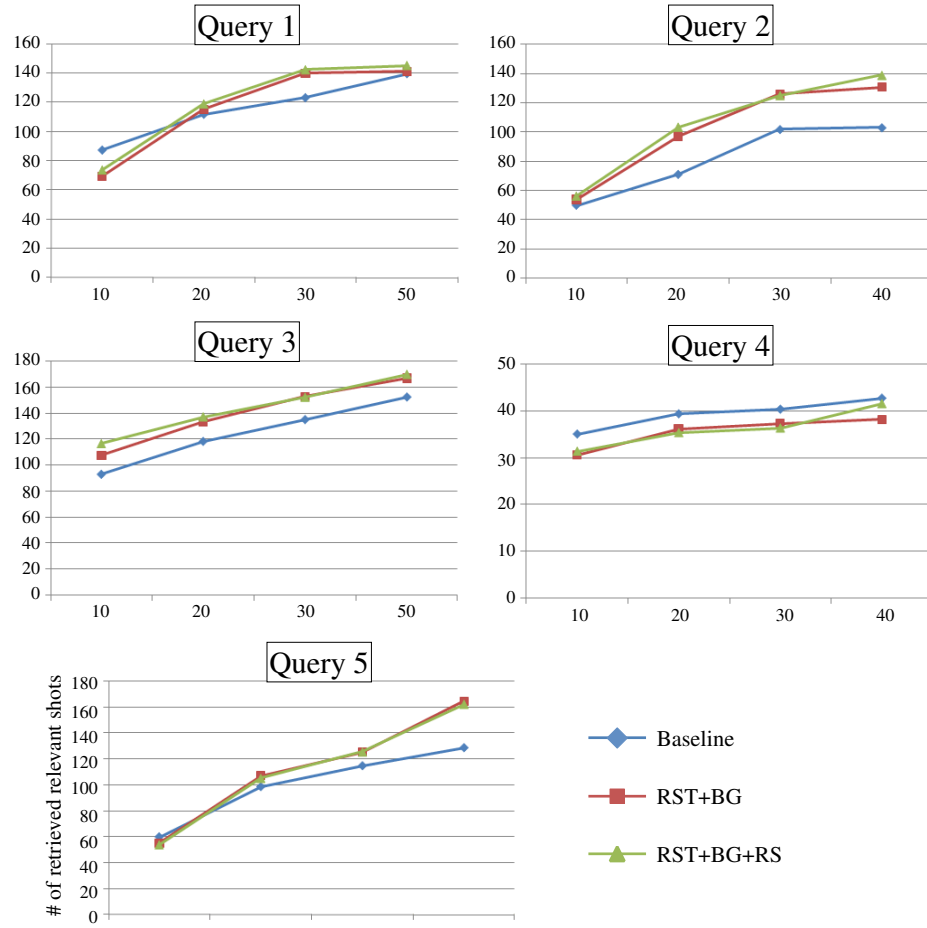
Figure 4.8: Retrieval performances for different available p-example numbers.

tervention is allowed only prior to the start of the test video search. The interactive category allows an interactive user intervention based on retrieval results. Although our proposed method belongs to the manually-assisted category, only three retrieval results are submitted to this category. This is clearly insufficient for achieving a meaningful comparison. Thus, the retrieval result of the proposed method is compared with 88 results in the fully automatic category.

Fig. 4.9 shows the maximum (solid arrow) and average (dashed arrow) numbers of relevant shots among 10 retrieval results, obtained using *RST+BG+RS* in Table 4.4. As can be seen from Fig. 4.9, the overall performance of the proposed method is ranked in the top quartile. It can be noted that almost all methods in the top quartile use classifiers to assess the relevance of each shot to keywords like *Person*, *Building*, *Cityspace* and so on. In this case, for each keyword, a classifier is built using a large number of training examples. For example, in the method proposed by researchers at City University of Hong Kong, classifiers for 374 keywords are built using 61, 901 manually annotated shots [21]. Also, in the method developed by researchers at University of Amsterdam, classifiers for 64 keywords are built using more than 10, 000 manually annotated shots, such as 39, 674 shots for *Bus*, 21, 532 shots for *Car* and so on [24]. Like this, methods in the top quartile require tremendous manual effort. Compared to these methods, our method only uses p-examples which are provided by a user in an off-the-cuff manner. Thus, it can be concluded that the proposed method is very effective in the sense that it requires neither manual shot annotation nor classifier preparation.

## 4.4.5   Reducing Computation Cost by Parallelization

In this section, we examine the computation cost of our QBE method and reduce it by parallelizing several processes. Our method consists of two main phases, PSL and RST. Fig. 4.10 illustrates processes in PSL (a) and RST (b) phases. Roughly speaking, the input of the PSL phase is a set of p-examples, and its output is a set of n-examples which are as similar to p-examples as possible. To this end, we first regard all of u-examples as n-examples and group them into clusters. Secondly, we find the representative n-example for each cluster. An SVM is then built by using p-examples and representative n-examples. Subsequently, n-examples which are distant from the decision boundary of the SVM are removed. Finally, the above processes are iterated
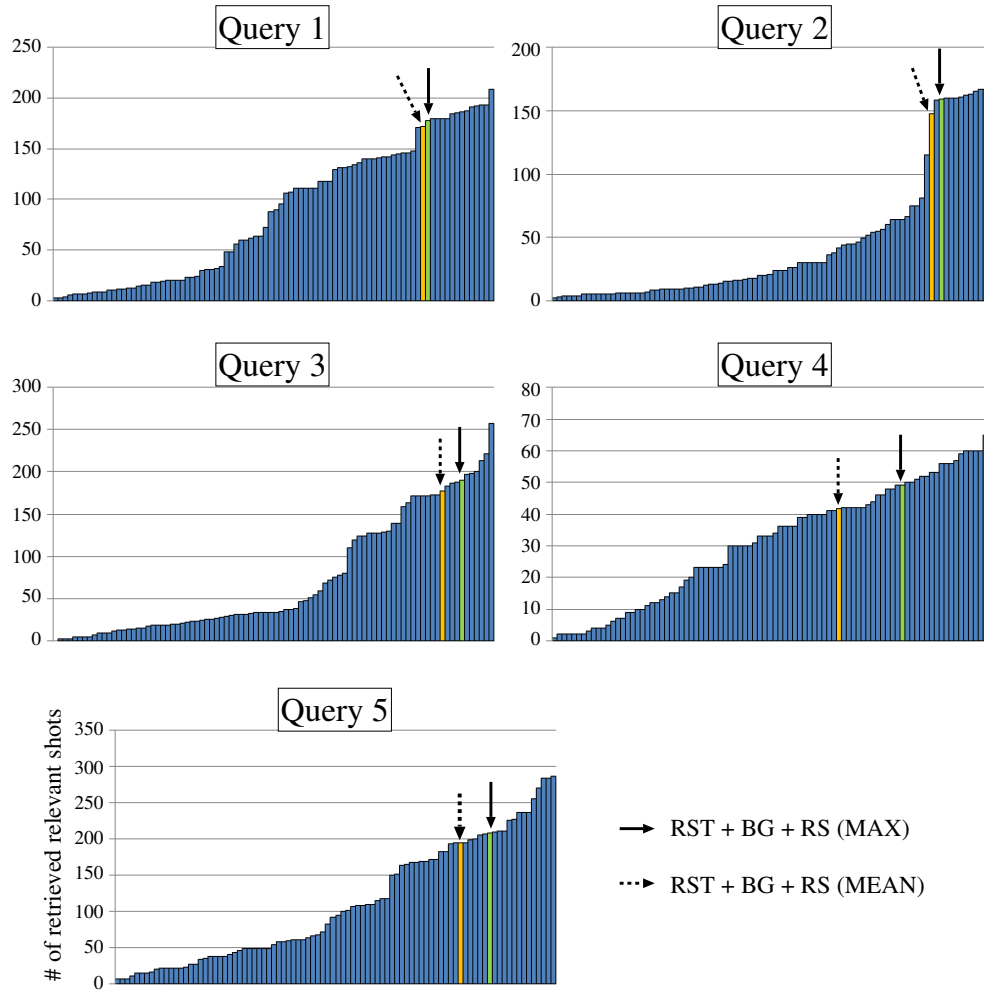
Figure 4.9: Performance comparison between our method and methods in TRECVID 2009.

until the number of n-examples is less than the pre-defined threshold or no n-example can be removed.
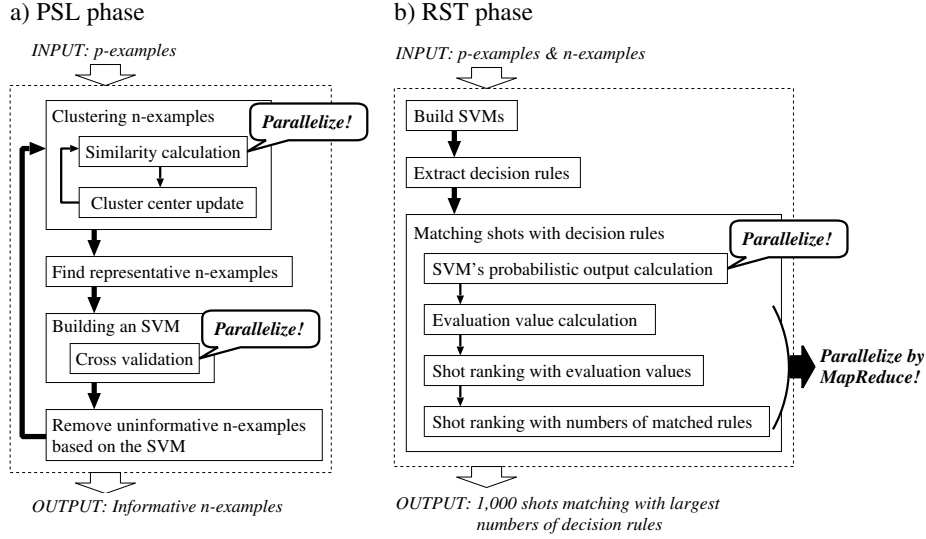


Figure 4.10: Illustrations of processes in PSL (a) and RST (b) phases.

When a large number of n-examples remain, the following two processes require expensive computation costs. The first is the similarity calculation for calculating the similarity between an n-example and each cluster center. The second computationally expensive process is cross validation where, for each parameter candidate, an SVM is built and evaluated by computing the error rate in classifying p-examples and n-examples. Thus, we parallelize the above two processes by using a multicore PC. For the similarity calculation process, each core is used to calculate the similarity between one cluster center and n-examples. For the cross validation process, each core is used to compute the error rate of an SVM with one parameter candidate. These are implemented using Matlab Parallel Computing Toolbox [6].

As shown in Fig. 4.10 (b), the input of the RST phase is a set of p-examples and n-examples, and its output is the set of $1,000$ shots which match the largest numbers of decision rules. The RST phase is summarized as follows. Firstly, various SVMs are built by using bagging and the random subspace method. Decision rules are then extracted based on SVMs' classification results of p-examples and n-examples. Subsequently, in order to

match shots with extracted decision rules, the following four processes are performed. The first one is SVM's probabilistic output calculation to compute the probabilistic output of an SVM for each shot. The second process is the evaluation value calculation which computes the evaluation value of matching a shot with each decision rule. The third is shot ranking with evaluation values where, for each decision rule, shots are ranked in the descending order of evaluation values. Based on this, $1,000$ shots with the largest evaluation values are regarded to match the decision rule. The final process is shot ranking with matched rules to determine $1,000$ shots matching the largest numbers of decision rules.

The RST phase has a high computation cost due to a large number of shots and decision rules. We address this by using two types of parallelizations on a multicore PC. The first one is applied to the process of SVM's probabilistic output calculation, where each core is used to compute SVM's probabilistic output for a distributed set of shots. The other three processes are implemented by using *MapReduce*, which is a parallel programming model that provides a simple and powerful interface [22]. We use 'Phoenix' which is a MapReduce library for multicore PCs, in order to save a significant amount of time on I/O operations [22]. In MapReduce, the basic data structure is a $(key, value)$ pair. Based on this, the *Map* function constructs input $(key, value)$ pairs from a distributed data, and produces intermediate $(key, value)$ pairs by conducting a user-defined task. Subsequently, the *Reduce* function conducts a user-defined merge operation on intermediate $(key, value)$ pairs with the same $key$ and outputs a final result. In this manner, MapReduce divides a large-scale data into small pieces of $(key, value)$ pairs, which can be efficiently processed in parallel by the Map and Reduce functions.

Three processes in Fig. 4.10 (b) are implemented by utilizing MapReduce twice. The first MapReduce performs two processes, the evaluation value calculation and shot ranking with evaluation values. Specifically, the objective is to determine $1,000$ shots which have the largest evaluation values for each rule. To do this, the following Map and Reduce functions are designed:

$$
\begin{aligned}
map_1 &: (x, [rule, Prob_{SVM}^{all}(x)]) \rightarrow List(rule, Match(x, rule)) \\
reduce_1 &: (rule, List(Match(x, rule))) \rightarrow (rule, SList_{rule}^{1,000}) \qquad (4.8)
\end{aligned}
$$

where $x$ and $rule$ are a shot and decision rule, respectively. $Prob_{SVM}^{all}(x)$ represents the set of probabilistic outputs of all SVMs for $x$. By using

$Prob_{SVM}^{all}(x)$, $map_1$ computes $Match(x, rule)$ which is the evaluation value of matching $x$ with *rule* as defined in equation (4.5). The output of $map_1$ is a list including evaluation values of distributed shots for all rules. Subsequently, $reduce_1$ merges such lists produced by $map_1$ on different cores so that for the same *rule*, evaluation values of all shots are combined into a list, namely, $List(Match(x, rule))$. By sorting this list, $reduce_1$ outputs $SList_{rule}^{1,000}$ which consists of $1,000$ shots with the largest evaluation values. That is, shots in $SList_{rule}^{1,000}$ are regarded to match *rule*.

The second MapReduce performs the process of shot ranking with numbers of matched rules. This process aims to obtain $1,000$ shots which match the largest numbers of decision rules. The following Map and Reduce functions are designed:

$$
\begin{aligned}
map_2 &: (x, 1) \rightarrow List(x, MRules(x)) \\
reduce_2 &: (x, List(MRules(x))) \rightarrow SList_{MRules}^{1,000}
\end{aligned}
\tag{4.9}
$$

where $(x, 1)$ is obtained by parsing $SList_{rule}^{1,000}$ and indicates that a shot $x$ matches one rule. It should be noted that as we have only to count the number of decision rules matched with $x$, we do not need to know which rules are matched with $x$. The function $map_2$ constructs a list of $(x, MRules(x))$s, where $MRules(x)$ represents the number of decision rules matched with $x$. Subsequently, $reduce_2$ merges $MRules(x)$ for the same shot into $List(MRules(x))$, which represents the total number of matched decision rules. Finally, the function sorts all shots based on $List(MRules(x))$ and outputs $SList_{MRules}^{1,000}$ which consists of $1,000$ shots matching the largest numbers of decision rules.

Fig. 4.11 illustrates retrieval times for our QBE method where relevant shots to *Query 1* are retrieved by applying *RST+BG* to 100 p-examples and 500 n-examples. Fig. 4.11 (a) and (b) show the change in computation times in the PSL and RST phases, respectively. In both figures, the four bars from the top to the bottom represent computation times by parallelizing our method with 1, 2, 4 and 8 cores, respectively. Bars named *Others* depict computation times of non-parallelized processes while the other bars depict computation times of parallelized processes. Specifically, in the PSL phase in Fig. 4.11 (a), *Clustering* and *SVM building* include the similarity calculation and cross validation processes in Fig. 4.10 (a), respectively. In the RST phase in Fig. 4.11 (b), *SVM prob.* corresponds to SVM's probabilistic output calculation process in Fig. 4.10 (b). *Rule matching* includes three

processes parallelized by MapReduce. In addition, numbers overlaid on bars present actual values of computation times. Both of Fig. 4.11 (a) and (b) indicate that, as the number of cores increases, the computation time can be significantly shortened. Specifically, when only one core is used, our QBE method takes a total of $26,312$ seconds to complete the retrieval with the PSL and RST phases requiring $24,796$ and $1,516$ seconds, respectively. In comparison, when 8 cores are used, the retrieval is completed in $6,194$ seconds with computation times of the PSL and RST phases being $5,531$ and $663$ seconds, respectively.
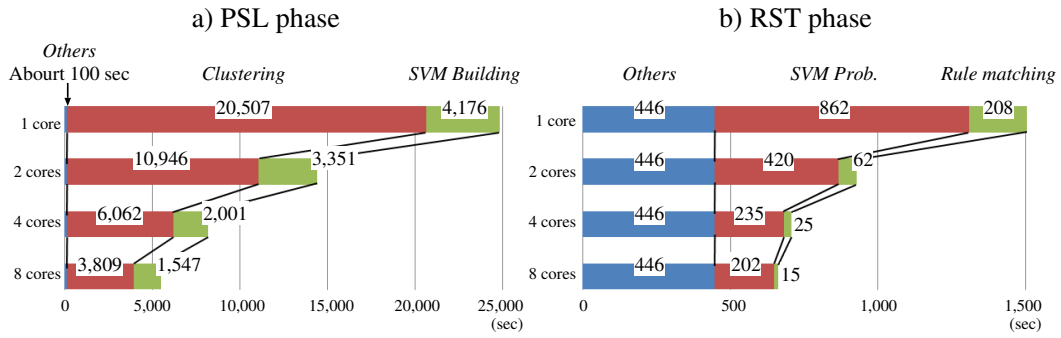


Figure 4.11: Comparison among retrieval times by parallelizing our QBE method with 1, 2, 4 and 8 cores.

From the perspective of computation time, our current QBE method is far from the satisfactory. So, the retrieval time will need to be further reduced by improving currently parallelized processes and parallelizing the other processes. In relation to this concern, Fig. 4.11 reveals an important issue. For *Clustering* and *Rule matching* processes, as the number of cores increases, computation times are nearly linearly reduced. However, compared to these reductions, reductions of computation times for *SVM building* and *SVM prob.* are significantly less effective. For example, for *SVM building* process in the PSL phase, even if the number of cores is doubled from 4 to 8, the ratio between the computation time of 4 cores and the one of 8 cores is only $1.29$ (i.e. $2,001$ seconds versus $1,547$ seconds). One consideration is that *SVM building* and *SVM prob.* are parallelized simply by distributing examples (or shots) to multiple cores. On each core, functions in LibSVM libraries [20] are called where memory allocations and releases are executed

many times. This degrades the effectiveness of parallelization. Therefore, in order to achieve effective parallelization, a process should be divided into sub-processes involving few memory allocations or releases.

## 4.5 Summary

In this chapter, we proposed a QBE method which can retrieve shots relevant to a query using only a small number of p-examples. Due to camera techniques and settings, relevant shots are characterized by significantly different features. As such, RST is used to extract multiple decision rules which characterize different subsets of p-examples. A variety of relevant shots can be retrieved where each decision rule is specialized to retrieve a part of relevant shots. Additionally, in order to extend the range of relevant shots that can be retrieved, bagging and the random subspace method are incorporated into RST. Classifiers built using different examples and feature dimensions, are useful for covering a variety of relevant shots while many irrelevant shots are potentially retrieved. Thus, RST is used to combine classifiers into decision rules in order to accurately retrieve relevant shots. Furthermore, to overcome the lack of n-examples, PSL is used to collect n-examples from u-examples. In particular, taking the class imbalance problem into account, a method which can collect a small number of n-examples useful for building an accurate classifier, is developed. Experimental results demonstrated that our method successfully covers various relevant shots when more than 20 p-examples are available. In addition, our method can achieve very effective retrieval without any shot annotation or classifier preparation.

The following issues will be addressed in future works. Firstly, we aim to use temporal features such as 3DSIFT [78] and acoustic features such as Mel-Frequency Cepstrum Coefficient (MFCC), as opposed to our current method which only makes use of image features. Secondly, although majority voting is currently conducted using all of extracted decision rules, some rules may be inaccurate or very similar to other rules. Thus, in order to obtain the optimal set of decision rules, a method which examines the accuracy of each decision rule based on cross validation will be developed. The relationship among decision rules will be investigated using the diversity measures proposed in [64]. Lastly, to improve the computation time of our current method, we plan to build a cluster consisting of tens or hundreds of PCs. On this cluster, in addition to the process of matching shots with decision rules, we parallelize

the other processes using Apache Hadoop [7], which implements MapReduce on a large-scale PC cluster.

# Chapter 5

# Knowledge-assisted Retrieval Using Video Ontology

## 5.1 Introduction

In this chapter, we construct a *video ontology* which is a formal explicit specification of concepts, concept properties and relations among concepts. Concepts are fundamental objects (semantic contents) such as *People*, *Car*, *Building* and so on. By using the video ontology, we aim to improve the retrieval performance of video retrieval methods which only uses features. Especially, we focus on the improvement of the Query-By-Example (QBE) method developed in the previous chapter. One of the biggest problems in QBE is that shots with similar features can have dissimilar semantic contents. For example, when *Ex. 1* in Fig. 5.1 is provided as an example shot for the query 'people walk in a street', *Shot 1* is correctly retrieved and *Shot 2* is wrongly retrieved. This is due to both *Ex. 1* and *Shot 2* having red-colored and ocher-colored regions. In addition, rectangular buildings and windows in *Ex. 1* are associated with rectangular posters on the wall in *Shot 2*. Like this, only features are considered in QBE, but semantic contents are not considered. Thus, we incorporate into QBE a video ontology as knowledge base. For example, in Fig. 5.1, *Shot 2* is not retrieved if we know that *People*, *Walking*, *Building* and *Road* should be observed in relevant shots.

A video ontology is especially important for managing a lack of example shots in QBE. Generally, as the number of feature dimensions increases, the number of example shots required to construct a well-generalized retrieval
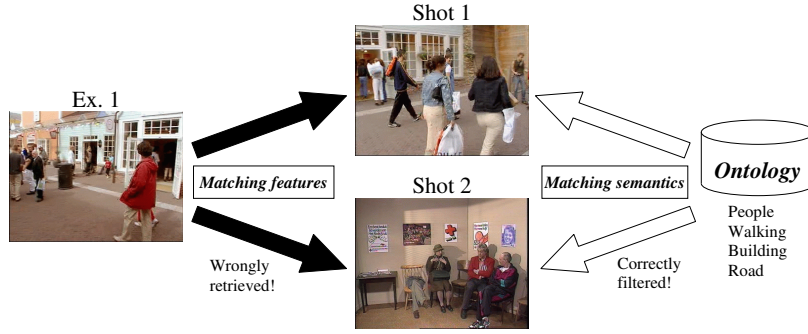
Figure 5.1: Example of QBE using a video ontology for the query 'people walk in a street'.

model exponentially increases [11]. It is because the relevance or irrelevance to a query needs to be determined for each combination of values along different dimensions. However, in QBE, a user can only provide a small number of example shots (e.g. 10 example shots), while each shot is represented using high-dimensional features (e.g. bag-of-visual-words representation with more than 1,000 dimensions). In this case, the statistical information of feature dimensions obtained from a small number of example shots, is not reliable. So, a retrieval model tends to be overfit to feature dimensions which are very specific to example shots, but are ineffective for characterizing relevant shots. For example, in Fig. 5.2, if *Ex. 1*, *Ex. 2* and *Ex. 3* are provided as example shots, the retrieval model is overfit to feature dimensions which characterize few edges in sky regions. As a result, *Shot 1*, *Shot 2* and *Shot 3* are retrieved, which are clearly irrelevant to the query.

In order to filter irrelevant shots, we develop a video ontology for utilizing object recognition results. Fig. 5.2 shows recognition results for three objects, *Building*, *Cityspace* and *Person*. One shot is represented as a vector of recognition scores, each of which represents the presence or absence of an object. For example, in Fig. 5.2, *Building* and *Cityspace* are likely to appear in example shots, although unlikely to appear in the other shots. Recently, researchers have used object recognition results in video retrieval. For example, researchers at City University of Hong Kong [21] and University of Amsterdam [23] built classifiers for recognizing 374 and 64 objects, respectively. These classifiers were built by using a large amount of training data, 61,901 shots in [21] and more than 10,000 shots in [23]. In this manner,
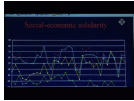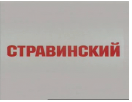
| | Ex. 1 | Ex. 2 | Ex. 3 | *Overfitting!* → *Filtered by ontology* Shot 1 | Shot 2 | Shot 3 |
|---|---|---|---|---|---|---|
| Building: | 2.5 | 2.2 | 1.2 | -0.3 | -1.0 | -2.3 |
| Cityspace: | 1.1 | 2.6 | 1.5 | -0.5 | -1.5 | -1.2 |
| Person: | -1.0 | -0.5 | 0.2 | 2.0 | -1.3 | -0.8 |

Figure 5.2: An example of an overfit retrieval result for the event 'tall buildings are shown'.

objects can be robustly recognized independently of sizes, positions and directions on the screen. The effectiveness of using object recognition results is validated with TRECVID which is a famous annual international workshop on video retrieval [15].

To utilize object recognition results, we construct a video ontology where a hierarchical structure of concepts and concept properties are defined. We can thereby select concepts related to a query and examine recognition scores of objects corresponding to selected concepts. For example, in Fig. 5.2, if *Building* and *Cityspace* are selected, *Shot 1*, *2* and *3* can be filtered due to low recognition scores for *Building* and *Cityspace*. It should be noted that this shot filtering discards shots which can be certainly regarded as irrelevant, several irrelevant shots remain because their recognition scores are not so small. Thus, in order to obtain the final retrieval result, we apply the QBE method developed in the previous chapter to the remaining shots. It should be noted that filtering irrelevant shots are useful for reducing the computation cost of the QBE method, because it reduces the number of shots examined by the method.

Finally, recall that in the QBE method, many SVMs are built using bagging and the random subspace method. Since the performance of an SVM is sensitive to parameters like kernel parameter and penalty parameter of misclassification, the SVM parameter estimation is important. We introduce a method for estimating the parameter of an SVM using the video ontology. Note that we are not provided with the relevance/irrelevance labeling of a shot, but are provided with object recognition scores. We assume that if the SVM is tuned with a good parameter, shots classified as relevant (positive) tend to have high object recognition scores for selected concepts. Thus, we

estimate the optimal parameter based on the correlation between selected concepts and shots classified by the SVM with each parameter candidate.

## 5.2 Related Works

The most popular video ontology is *Large-Scale Concept Ontology for Multimedia* (LSCOM) [71]. This targets broadcast news videos and defines a standardized set of $1,000$ concepts. However, LSCOM merely provides a list of concepts where no concept relation or structure is defined. Hence, there has been a lot of researches relating to the appropriate selection of LSCOM concepts for a query.

Existing concept selection approaches can be roughly classified into three types: manual, text-based and visual-based selections. In manual concept selection, users manually select concepts related to a query [67]. However, different users select significantly different concepts for the same query. For example, [67] conducted an experiment where 12 subjects were asked to judge whether a concept was related to a query. The results demonstrated only 13% of total $7,656$ judgements were the same among all subjects. In text-based concept selection, WordNet is frequently used where words in the text description of a query are expanded based on synonyms, hypernyms and hyponyms [24, 21]. Then, concepts corresponding to expanded words are selected. However, WordNet only defines lexical relations among concepts, and not spatial and temporal relations. For example, WordNet is unable to identify *Building* and *Road* as being frequently shown in the same shots. Finally, in visual-based concept selection, concepts are selected as objects which are recognized in example shots with high recognition scores [24, 21]. This approach relies on accuracies of object recognition. LSCOM includes concepts corresponding to objects, which are difficult to recognize, such as *Dogs*, *Telephone*, *Supermarket*, and so on. Thus, visual-based concept selection may wrongly select concepts which are unrelated to the query.

To overcome the above problems, we manually organize LSCOM concepts into a video ontology, which can capture both lexical relations among concepts and their spatial and temporal relations. To do so, we define several new concepts which are not addressed in LSCOM. For example, we define the new concept *Air_Vehicle* as a superconcept of *Airplane* and *Helicopter* in order to explicitly represent both *Airplane* and *Helicopter* as objects flying in the air or moving in airports. We also introduce a method which can

appropriately estimate parameters of a retrieval model based on concepts selected by the video ontology. We are not aware of any existing parameter estimation methods based on ontologies in the previous research.

## 5.3 Video Ontology Construction and Concept Selection

We assume that each video is already divided into shots. This can be accurately conducted by using existing methods, where a shot boundary is detected as a significant change of features (e.g. color, edge, motion etc.) between two consecutive video frames [100]. In addition, recognition scores of various objects are already assigned to each shot. Particularly, we use recognition results of 374 objects, provided by Video Retrieval Group (VIREO) in City University of Hong Kong [99]. Since our objective is the video ontology construction and the concept selection based on this ontology, we do not describe how to obtain object recognition scores, please refer to [99] in more detail. In the following discussion, we explain how to organize LSCOM concepts corresponding to 374 objects into a video ontology, and how to select concepts related to a query.

First of all, as a guideline of organizing concepts, we define four top-level concepts shown in Fig. 5.3. LSCOM concepts are represented by capital letters followed by lower-case letters, while concepts that we define are represented only by capital letters. Also, concept properties are represented by starting their names with lower-case letters. Top-level concepts are useful for classifying LSCOM concepts into broad categories. For example, since *Building* and *Construction_Site* frequently appear in the same shots, one may wrongly regard that these concepts belong to the same category. But, by carefully examining their meanings, we can find that *Building* and *Construction_Site* should be separately defined as subconcepts of *NON-PERSON_OBJECT* and *LOCATION*, respectively. And, *Building* should be defined as a part of *Construction_Site* (i.e. *hasPartOf* property). Like this, we can flexibly represent a concept by referring to concepts belonging to different top-level concepts.

Under top-level concepts, we organize LSCOM concepts by considering the *disjoint partition* requirement. This is a well-known ontology design pattern for making ontologies easily interpretable by both humans and machines
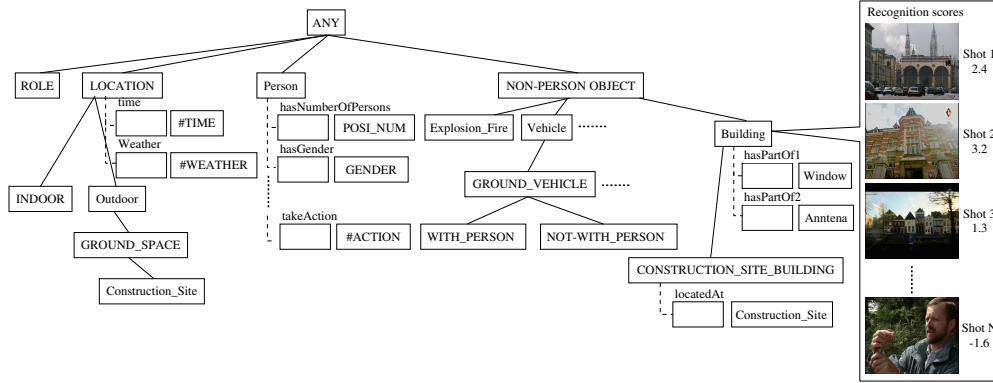
Figure 5.3: A part of our video ontology.

[70]. The disjoint partition requirement indicates that a concept $C1$ should be decomposed into disjoint subconcepts $C2, C3, \cdots$. That is, for $i, j \geq 2$ and $i \neq j$, $C_i \cap C_j = \phi$. So, an instance of $C1$ cannot be an instance of more than one subconcept $C2, C3, \cdots$. For example, *Vehicle* and *Car* should not be defined in the same level of the concept hierarchy, because an instance of *Car* is an instance of *Vehicle*. Instead, *Car* should be defined as a subconcept of *Vehicle*. Hence, to satisfy the disjoint partition requirement, we have to carefully examine whether a concept is a generalization (or specialization) of another concept.

Furthermore, visual characteristics are used to define the concept hierarchy. For example, as can be seen in Fig. 5.3, we define two subconcepts of *GROUND_VEHICLE*, namely, *WITH_PERSON* and *NOT-WITH_PERSON*. While we can infer that *Person* probably appears in shots containing subconcepts of *WITH_PERSON*, such as *Bicycle* and *Motorcycle*, it is uncertain whether *Person* appears in shots containing subconcepts of *NOT-WITH_PERSON*, such as *Car* and *Bus*. In this way, based on visual characteristics, we examine whether objects corresponding to different concepts are frequently shown in the same shots.

We now explain the selection of concepts related to a query. Roughly speaking, we first select concepts which match words in the text description of the query. Subsequently, for each selected concept, we select its subconcepts and concepts which are specified as properties. For example, for the query 'buildings are shown', *Buildings* and all of its subcocepts, such

as *Office_Buildings*, *Hotel* and *Power_Plant*, are initially selected. As shown in Fig. 5.3, *Windows* and *Antenna* are selected from *hasPartOf1* and *has-PartOf2* properties of *Building*. After that, from the *locatedAt* property of *CONSTRUCTION_SITE_BUILDING*, which is a subconcept of *Building*, we select *Construction_Site* and all of its subconcepts, such as *City-space*, *Urban* and *Suburban*. At this point, by tracing concept properties many times, we may select concepts which are unrelated to the query. For example, from the above *Construction_Site*, we can trace *ARTIFICIAL_ROAD*, *Sidewalk* and *Person*. However, these concepts are not related to the query. To avoid selecting unrelated concepts, we restrict the number of tracing concept properties to only once. That is, for the above example, the concept selection terminates after selecting *Construction_Site* and all of its subconcepts.

In Fig. 5.3, some concept properties are characterized by slots with #, called *# operator*. This represents a concept property which is used only when it is specified in the textual description of a query. For example, let us consider the query 'people are indoors'. For this query, we select *Person* and all of its subconcepts, and trace *Person*'s concept properties. But, for the *takeAction* property of *Person*, the current LSCOM only defines 12 concepts, such as *Singing*, *People_Crying*, *Talking* and so on. If these concepts are selected, shots containing them may be preferred. As a result, we may overlook shots where people perform different actions in indoor situations, such as eating and watching TV. Thus, only for queries like 'people talking indoors', we use the concept property *takeAction* to select concepts.

Since the textual description of a query is usually simple, we cannot select concepts which are definitely related to the query. For example, for the query 'buildings are shown', we select 55 concepts including *White_House*, *Military_Base*, *Ruins*, and so on, but only a portion of these concepts are actually related to the query. Hence, we validate selected concepts using example shots. Recall that all shots are associated with recognition scores of objects corresponding to LSCOM concepts, as shown in *Building* in Fig. 5.3. Based on such recognition scores in example shots, we validate concepts selected by our video ontology. In particular, for each object corresponding to a concept, we compute the average recognition score among example shots, which allows the ranking of concepts in descending order. Next, we select concepts which are not only selected by our video ontology, but also ranked in the top $T$ positions (we use $T = 20$). In this manner, selected concepts are validated from both semantic and statistical perspectives.

Finally, we filter irrelevant shots to a query by using selected concepts.

For each shot, among objects corresponding to selected concepts, we count the number of objects shown in the shot. An object is regarded to be shown in the shot, if its recognition score is larger than $R$. For any object, $R$ is set to 1.0 where recognition scores of all shots are normalized to have the mean of zero and the variance of one. If the number of objects shown in the shot is less than $C$ (we use $C = 1$ or 2), it is filtered as irrelevant, otherwise it is retained. In this way, we filter shots which can be certainly regarded as irrelevant to the query. Then, in order to obtain the final retrieval result, we apply the QBE method in the capter 4 to the remaining shots.

In the following paragraphs, we explain the estimation of an SVM parameter based on object recognition scores. Suppose that, for a query, we have a set of selected concepts $C$, where each concept is represented as $c_i$ ($1 \leq i \leq |C|$). Further, we have $P$ parameter candidates for an SVM $M$, where the $j$-th parameter is $p_j$ and the SVM with $p_j$ is $M_{p_j}$ ($1 \leq j \leq P$). To estimate the best parameter, we temporarily retrieve $S$ shots by using $M_{p_j}$ (we use $S = 1,000$). We then compute the correlation between $C$ and $M_{p_j}$ as follows:

$$Correlation(C, M_{p_j}) = \sum_{i=1}^{C} \gamma(rank(M_{p_j}),\ rank(c_i)) \tag{5.1}$$

where $rank(M_{p_j})$ represents a ranking list of $S$ shots according to their evaluation values by $M_{p_j}$. We obtain these evaluation values as SVM's probabilistic outputs [38]. $rank(c_i)$ represents a ranking list of $S$ shots according to recognition scores of the object corresponding to $c_i$. Using $rank(M_{p_j})$ and $rank(c_i)$, we compute $\gamma(rank(M_{p_j}), rank(c_i))$ as the Spearman's rank correlation coefficient [84]. This represents the correlation between two ranking lists. If these are highly correlated, $\gamma(rank(M_{p_j}), rank(c_i))$ is close to 1, or otherwise close to $-1$. Hence, a larger $\gamma(M_{p_j}, c_i)$ indicates that $M_{p_j}$ is more correlated with $c_i$. $Correlation(C, M_{p_j})$ represents the overall correlation over all concepts in $C$. Thus, we select the best parameter $p_j^*$ where $Correlation(C, M_{p_j})$ is the largest among $P$ parameter candidates. In this way, we can estimate an SVM parameter which is semantically validated based on selected concepts.

## 5.4 Experimental Results

We examine the effectiveness of our video ontology using TRECVID 2009 video data [15]. This data consists of 219 development and 619 test videos. Each video is already divided into shots where development and test videos include $36,106$ and $97,150$ shots, respectively. In addition, for all shots, recognition scores of 374 objects are provided by City University of Hong Kong [99]. We target the following four queries: *Query 1:* A view of one or more tall buildings and the top story visible; *Query 2:* Something burning with flames visible; *Query 3:* One or more people, each at a table or desk with a computer visible; *Query 4:* One or more people, each sitting in a chair, talking. For each query, the retrieval is conducted nine times using different sets of 10 example shots. We evaluate the retrieval performance as the average number of relevant shots within $1,000$ retrieved shots.

In Fig. 5.4 (a), we compare the following three types of retrieval in order to evaluate the effectiveness of our video ontology for filtering out irrelevant shots and estimating an SVM parameter. The first is *Baseline* without using our video ontology. The second is termed *Ontology1* and uses our video ontology only for filtering out irrelevant shots. The final type of retrieval is *Ontology2*, which uses our video ontology for both irrelevant shot filtering and SVM parameter estimation. For each query, performances of *Baseline*, *Ontology1* and *Ontology2* are represented by the leftmost red bar, the middle green bar and the rightmost blue bar, respectively.
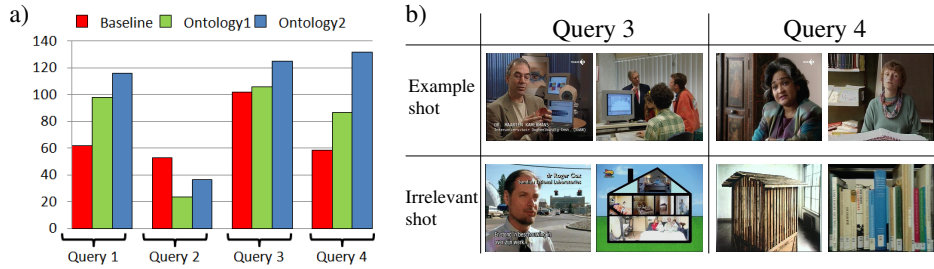


Figure 5.4: (a) Performance comparison among *Baseline*, *Ontology1* and *Ontology2*, (b) Examples of shots filtered by our video ontology.

Fig. 5.4 (a) shows that, with the exception of *Query 2*, it is very effective to filter irrelevant shots based on concepts selected by our video ontology.

The retrieval performance is further improved by estimating SVM parameters based on selected concepts. The reason for the low performance for *Query 2* is that, we can only select two concepts, *Explosion_Fire* and *Earthquake*, which are related to the term 'flame' in the text description of *Query 2*. Here, LSCOM does not contain any other concepts related to the flame, such as candle flame, bonfire, fire blasted from rockets. Thus, such concepts should be incorporated to improve the performance for *Query 2*. Furthermore, *Explosion_Fire* and *Earthquake* are not very effective for characterizing relevant shots. For example, $1,000$ shots with the highest recognition scores of *Explosion_Fire* and those of *Earthquake*, only characterize 37 and 12 relevant shots, respectively. Hence, to achieve accurate retrieval for *Query 2*, it is required to improve the recognition accuracy for *Explosion_Fire* and *Earthquake* (as well as newly incorporated concepts).

In Fig. 5.4 (b), the second row depicts two example shots for *Query 3* and *Query 4*. The third row shows two irrelevant shots which are wrongly retrieved by *Baseline*, but are appropriately filtered by our video ontology (for *Query 1*, see Fig. 5.2). More specifically, for *Query 3*, *Baseline* wrongly retrieves shots where people just appear and shots which contains straight lines corresponding to computer shapes, and shapes of pillars and blinds in a background. For *Query 4*, shots which contain straight lines corresponding to shapes of background objects, are wrongly retrieved. By filtering out the above types of shots, *Ontology1* and *Ontology2* can significantly outperform *Baseline*.

Finally, we examine the additional effectiveness of our video ontology, that is, what amount of computation time of the QBE method is reduced by filtering of irrelevant shots based our video ontology. In Fig. 5.5, for each query, the left red bar represents the computation time of *Baseline* where all shots are examined by the QBE method. On the other hand, the right green bar represents the computationt time of *Ontology1* where the method only examines the remaining shots after filtering of irrelevant shots. As seen in Fig. 5.5, filtering of irrelevant shots is useful for reducing computation times. Nonetheless, the QBE method requires about 500 seconds to complete the retrieval, which is far from the satisfactory. One main reason is that the QBE method requires building of multiple SVMs and matching of many decision rules. Thus, we are currently parallelizing processes of SVM building and decision rule matching by using multiple processors.
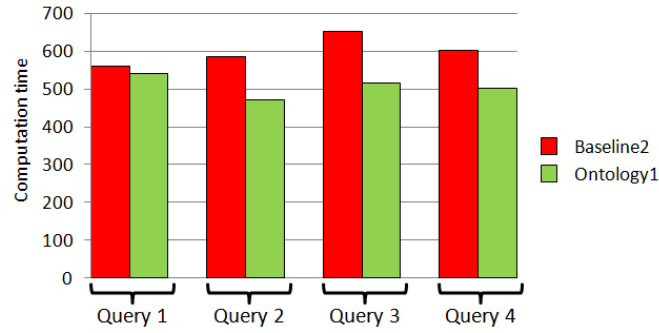
Figure 5.5: Comparison between the computation time of *Baseline* and that of *Ontology1*.

## 5.5 Summary

In this chapter, we constructed a video ontology to incorporate object recognition results into QBE. Specifically, it is constructed by defining a hierarchical structure of concepts based mainly on the disjoint partition requirement and visual characteristics. The video ontology is used to select concepts related to a query. Subsequently, irrelevant shots are filtered by referring to recognition results of objects corresponding to selected concepts. In addition, the video ontology is used to estimate the optimal SVM parameter based on the correlation between selected concepts and shots classified by the SVM with each parameter candidate. Experimental results validate two effectivenesses of filtering irrelevant shots based on our video ontology. The first is that the retrieval performance of the QBE method in chapter 4 can be significantly improved. The second is that its computation time can be reduced. The above two issues will be further explored in our future works. Specifically, to improve the retrieval performance, we plan to extend the current video ontology to deal with temporal relations among concepts. For the improvement of the computation time, we aim to parallelize the QBE method using multiple processors.

# Chapter 6

# Conclusion and Future Works

In this dissertation, we proposed three video data mining methods to develop a video retrieval system with multi-modal interfaces. The first method developed in chapter 2 is used for a QBK interface where queries are represented by keywords. The method extracts semantic patterns as sequential patterns of features. Each semantic pattern relates shot sequences associated with a certain keyword. We incorporate two time constraints, semantic event boundaries and temporal localities. Based on these, we can not only avoid extracting meaningless patterns, but also effectively reduce the search space of possible patterns. Experimental results show that extracted semantic patterns characterize character's actions, situations and combinations of actions and situations.

The method in chapter 3 is developed for a QBB interface, which provides a video browsing functionality for finding keywords or example shots to represent a query. The method extracts topics which have much impacts on viewers based on the anomaly of video editing patterns. Topics are extracted as shot sequences characterized by bursts, which are abnormal patterns of a character's appearance and disappearance in a video. First, the method performs a probabilistic time-series segmentation to divide the video into shot sequences. Each shot sequence is characterized by a distinct pattern of the character's appearance and disappearance. Then, the burst intensity measure is used to evaluate whether or not the pattern in each shot sequence is abnormal. Experimental results demonstrate that bursts characterize topics where a character performs interesting actions, such as fighting, chasing, making love and so on.

The third method in chapter 4 is used for a QBE interface where a query

is represented by providing example shots. We address that shots relevant to a query are characterized by significantly different features, due to varied camera techniques and settings. Thus, rough set theory is used to extract multiple patterns which can correctly identify different subsets of example shots. By accumulating relevant shots retrieved by such patterns, we can retrieve a variety of relevant shots. In addition, to extend the range of relevant shots that can be retrieved, rough set theory is extended by adopting bagging and the random subspace method, where many classifiers are built using randomly selected example shots and feature dimensions. Although such classifiers characterize significantly different relevant shots, they potentially retrieve many irrelevant shots. Hence, rough set theory is used to extract classification rules (patterns) as combinations of classifiers, which provide greater retrieval accuracy. Furthermore, counter example shots, which are a necessity of rough set theory, are collected using partially supervised learning. We collect counter example shots which are as similar to example shots as possible, because they are useful for defining the boundary between relevant and irrelevant shots. Experimental results on TRECVID 2009 video data demonstrate that the proposed method can achieve effective retrieval only using example shots, where no manual shot annotation is required.

Finally, in chapter 5, we present a method which integrates the QBK and QBE interfaces to improve the retrieval performance. This method uses a video ontology as knowledge base in QBE. The video ontology is constructed by organizing 374 LSCOM concepts based on the taxonomic (subconcept-superconcept) relation among concepts and their visual characteristics. Given the text description of a query (QBK), we trace the video ontology to select concepts related to the query. Then, irrelevant shots are filtered by referring to recognition results of objects corresponding to selected concepts. Lastly, QBE is performed on the remaining shots to obtain a final retrieval result. Experimental results show that the video ontology is useful for not only improving the retrieval performance, but also reducing the computation time.

In conclusion, video data mining is verified as effective for extracting patterns which characterize semantic contents. We have developed a multimodal video retrieval system using extracted patterns. In the future, we will extend this research in the following two directions. The first one is to extend the developed methods to the internet scale. For example, if the method for the QBK interface will be applied to a video hosting site where videos are annotated with text tags, we may extract much more semantic patterns

compared to the ones in chapter 2. In addition, since there are uncountable number of videos on the internet, the computation cost of a method is one of the most important issues. We plan to parallelize the developed methods on a PC cluster which consists of thousands of processors. The second direction is to extend the current multi-modal system by adopting another interface. We plan to develop a *Query-By-Gesture* (QBG) interface which complements the QBE interface. One crucial problem in the QBE interface is that the retrieval cannot be performed if a user does not have example shots for a query. Thus, we aim to develop a QBG interface where example shots for any query are created by the user. Especially, in order to facilitate the creation of example shots, we aim to use 'virtual reality' techniques where example shots are created by synthesizing user's gesture in front of the video camera, 3DCGs and background images. We will examine whether or not such example shots can substitute for example shots selected from actual videos.

# Bibliography

[1] *ENGINEERING STATISTICS HANDBOOK (1.3.6.6.7. Exponential Distr ibution).* `http://www.itl.nist.gov/div898/handbook/eda/section3/eda3667.htm`.

[2] *MPFactory: MPEG Software Development Kit.* `http://w3-mcgav.kddilabs.jp/mpeg/mpfs40/indexe.html`.

[3] *MPI: A Message-Passing Interface Standard.* `http://www.mpi-forum.org`.

[4] *NEC Press Release on January 10, 2008.* `http://www.nec.co.jp/press/ja/0801/1004.html`.

[5] *OpenCV: Open Source Computer Vision Library.* `http://www.intel.com/research/mrl/research/opencv/`.

[6] *Parallel Computing Toolbox - MATLAB.* `http://www.mathworks.com/products/parallel-computing/`.

[7] *Welcome to Apache Hadoop!* `http://hadoop.apache.org/`.

[8] A. Hampapur. *Designing Video Data Management Systems.* Ph.D. dissertation, University of Michigan, 1995.

[9] A. Hanjalic, R. 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.

[10] A. Jain, M. Murty and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[11] A. Jain, R. Duin and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[12] A. Mizui, K. Shirahama and K. Uehara. TRECVID 2008 NOTEBOOK PAPER: Interactive search using multiple queries and rough set theory. In *Proc. of TRECVID 2008*, pages 123–132, 2008.

[13] A. Mojsilovic and B. Rogowitz. Capturing image semantics with low-level descriptors. In *Proc. of ICIP 2001*, pages 18–21, 2001.

[14] A. Natsev, M. Naphade and J. Tešić. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proc. of ACM MM 2005*, pages 598–607, 2005.

[15] A. Smeaton, P. Over and W. Kraaij. Evaluation campaigns and TRECVid. In *Proc. of MIR 2006*, pages 321–330, 2006.

[16] A. Yoshitaka, T. Ishii and M. Hirakawa. Content-based retrieval of video data by the grammar of film. In *Proc. of VL 1997*, pages 314–321, 1997.

[17] B. Adams, C. Dorai and S. Venkatesh. Novel approach to determining tempo and dramatic story sections in motion pictures. In *Proc. of ICIP 2000*, pages 283–286, 2000.

[18] B. Liu, W. Lee, P. Yu and X. Li. Partially supervised classification of text documents. In *Proc. of ICML 2002*, pages 387–394, 2002.

[19] C. Berberidis, I. Vlahavas, W. Aref, M. Atallah and A. Elmagarmid. On the discovery of weak periodicities in large time series. In *Proc. of PKDD 2002*, pages 51–61, 2002.

[20] C. Hsu, C. Chang and C. Lin. *A Practical Guide to Support Vector Classification*, 2003. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[21] C. Ngo et al. VIREO/DVM at TRECVID 2009: High-level feature extraction, automatic video search and content-based copy detection. In *Proc. of TRECVID2009*, pages 415–432, 2009.

[22] C. Ranger et al. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proc. of HPCA 2007*, pages 13–24, 2007.

[23] C. Snoek and M. Worring. Multimedia event-based video indexing using time intervals. *IEEE Transactions on Multimedia*, 7(4):638–647, 2005.

[24] C. Snoek et al. The mediamill TRECVID 2009 semantic video search engine. In *Proc. of TRECVID2009*, pages 226–238, 2009.

[25] D. Chudova and P. Smyth. Pattern discovery in sequences under a markov assumption. In *Proc. of KDD 2002*, pages 153–162, 2002.

[26] D. Gemmell, H. Vin, D. Kandlur, P. Rangan and L. Rowe. Multimedia storage servers: A tutorial. *IEEE Computer*, 28(5):40–49, 1995.

[27] D. Keysers, C. Gollan and H. Ney. Local context in non-linear deformation models for handwriting character recognition. In *Proc. of ICPR 2004*, pages 511–514, 2004.

[28] D. Tao, X. Tang, X. Li and X. Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1088–1099, 2006.

[29] D. Wijesekera and D. Barbara. Mining cinematic knowledge: Work in progress. In *Proc. of MDM/KDD 2000*, pages 98–103, 2000.

[30] D. Zhong and S. Chang. Structure analysis of sports video using domain models. In *Proc. of ICME 2001*, pages 920–923, 2001.

[31] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[32] G. Das, K. Lin, H. Mannila, G. Renganathan and P. Smyth. Rule discovery from time series. In *Proc. of KDD 1998*, pages 16–22, 1998.

[33] G. Davenport, T. Smith and N. Pincever. Cinematic primitives for multimedia. *IEEE Computer Graphics and Applications*, 11(4):67–74, 1991.

[34] G. Fung, J. Yu, H. Ku and P. Yu. Text classification without negative examples revisit. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):6–20, 2006.

[35] G. Fung, J. Yu, P. Yu and H. Lu. Parameter free bursty events detection in text streams. In *Proc. of VLDB 2005*, pages 181–192, 2005.

[36] G. Guo and C. Dyer. Learning from examples in the small sample case: Face expression recognition. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 35(3):477–488, 2005.

[37] H. Liu. Evolving feature selection. *IEEE Intelligent Systems*, 20(6):64–76, 2005.

[38] H. Liu, C. Lin and R. Weng. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, 2007.

[39] H. Mannila, H. Toivonen and A. Verkamo. Discovering frequent episodes in sequences. In *Proc. of KDD 1995*, pages 210–215, 1995.

[40] H. Yu, J. Han and K. Chang. PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[41] H. Zhou and D. Kimber. Unusual event detection via multi-camera video mining. In *Proc. of ICPR 2006*, pages 1161–1166, 2006.

[42] J. Han and M. Kamber. *Data Mining: Concepts and Techniques (Second Edition)*. Morgan Kaufmann Publishers, 2006.

[43] J. Han, G. Dong and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proc. of ICDE 1999*, pages 106–115, 1999.

[44] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki and H. Toivonen. Unusual event detection via multi-camera video mining. In *Proc. of ICDM 2001*, pages 203–210, 2001.

[45] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. of KDD 2002*, pages 91–101, 2002.

[46] J. Komorowski, A. Øhrn and A. Skowron. The rosetta rough set software system. In *Handbook of Data Mining and Knowledge Discovery, W. Klösgen and J. Zytkow (eds.)*, chapter D.2.3. Oxford University Press, 2002.

[47] J. Matsuyama and K. Uehara. Multidirectional face tracking with 3d face model and learning half-face template. In *Proc. of VISAPP 2006*, pages 77–84, 2006.

[48] J. Monaco. *How to Read a Film*. Oxford University Press, 1981.

[49] J. Nam, M. Alghoniemy and A. Tewfik. Audio-visual content-based violent scene characterization. In *Proc. of ICIP 1998*, pages 353–357, 1998.

[50] J. Oh and B. Bandi. Multimedia data mining framework for raw video sequences. In *Proc. of MDM/KDD 2002*, pages 1–10, 2002.

[51] J. Pan and C. Faloutsos. Videograph: A new tool for video mining and classification. In *Proc. of JCDL 2001*, pages 116–117, 2001.

[52] J. Pan and C. Faloutsos. Videocube: A novel tool for video mining and classification. In *Proc. of ICADL 2002*, pages 194–205, 2002.

[53] J. Smith and S. Chang. Tools and techniques for color image retrieval. In *Proc. of SPIE Storage and Retrieval for Image and Video Databases*, pages 426–437, 1996.

[54] J. Tyler, D. Wilkinson and B. Huberman. Email as spectroscopy: Automated discovery of community structure within organization. In *Communities and Technologies, M. Huysman, E. Wenger, V. Wulf (eds.)*, pages 81–96. Kluwer Academic Publishers, 2003.

[55] J. Verbeek, N. Vlassis and B. Kröse. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.

[56] J. Yang, W. Wang and P. Yu. Mining asynchronous periodic patterns in time series data. In *Proc. of KDD 2000*, pages 275–279, 2000.

[57] J. Yuan, J. Li and B. Zhang. Learning concepts from large scale imbalanced data sets using support cluster machines. In *Proc. of ACM MM 2006*, pages 441–450, 2006.

[58] K. Kashino, T. Kurozumi and H. Murase. A quick search method for audio and video signals based on histogram pruning. *IEEE Transactions on Multimedia*, 5(3):348–357, 2003.

[59] K. Sande, T. Gevers and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[60] K. Shirahama, C. Sugihara, Y. Matsuoka, K. Matsumura and K. Uehara. Kobe university at TRECVID 2009 search task. In *Proc. of TRECVID 2009*, pages 76–84, 2009.

[61] K. Shirahama, K. Ideno and K. Uehara. A time-constrained sequential pattern mining for extracting semantic events in videos. In *Multimedia Data Mining and Knowledge Discovery, V. Petrushin and L. Khan (eds.)*, pages 404–426. Springer, 2006.

[62] K. Shirahama, Y. Matsuo and K. Uehara. Mining semantic structures in movies. *Lecture Notes in Computer Science*, 3392:116–133, 2005.

[63] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[64] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

[65] L. Lamel and J. Gauvain. Speaker recognition with the switchboard corpus. In *Proc. of ICASSP 1997*, pages 1067–1070, 1997.

[66] L. Wei and E. Keogh. Semi-supervised time series classification. In *Proc. of KDD 2006*, pages 748–753, 2006.

[67] M. Christel and A. Hauptmann. The use and utility of high-level semantic features in video retrieval. In *Proc. of CIVR 2005*, pages 134–144, 2005.

[68] M. Eisen, P. Spellman, P. Brown and D. Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proc. of the National Academy of Science of the United States of America (Vol. 95, No. 2)*, pages 14863–14868, 1998.

[69] M. Fleischman, P. Decamp and D. Roy. Mining temporal patterns of movement for video content classification. In *Proc. of MIR 2006*, pages 183–192, 2003.

[70] M. Horridge et al. *A Practical Guide to Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*, 2004. `http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf`.

[71] M. Naphade, J. Smith, J. Tešić, S. Chang, W. Hsu, L. Kennedy, A. Hauptmann and J. Curtis. Large-scale concept ontology for multimedia. *IEEE Multimedia*, 13(3):86–91, 2006.

[72] M. Vlachos, K. Wu, S. Chen and P. Yu. Fast burst correlation of financial data. In *Proc. of PKDD 2005*, pages 368–379, 2005.

[73] M. Wang, T. Madhyastha, N. Chan, S. Papadimitriou and C. Faloutsos. Data mining meets performance evaluation: Fast algorithm for modeling bursty traffic. In *Proc. of ICDE 2002*, pages 507–516, 2002.

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

[75] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

[76] N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of IC-AI 2000*, pages 111–117, 2000.

[77] P. Ferreira and P. Azevedo. Protein sequence pattern mining with constraints. *Lecture Notes in Artificial Intelligence*, 3721:96–107, 2005.

[78] P. Scovanner, S. Ali and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *Proc. of ACM MM 2007*, pages 357–360, 2007.

[79] P. Yao. Hybrid classifier using neighborhood rough set and svm for credit scoring. In *Proc. of BIFE 2009*, pages 138–142, 2009.

[80] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of VLDB 1994*, pages 487–499, 1994.

[81] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of ICDE 1995*, pages 3–14, 1995.

[82] R. Akbani, S. Kwek and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Proc. of ECML 2004*, pages 39–50, 2004.

[83] R. Hilderman and H. Hamilton. *Knowledge Discovery and Measures of Interest*. Kluwer Academic Publishers, 2001.

[84] R. Hogg and A. Crig. *Introduction to Mathematical Statistics (5th edition)*. Prentice Hall, 1994.

[85] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of EDBT 1996*, pages 3–17, 1996.

[86] S. Pfeiffer, R. Lienhart and W. Effelsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15(1):59–81, 2001.

[87] S. Saha, C. Murthy and S. Pal. Rough set based ensemble classifier for web page classification. *Fundamenta Informaticae*, 76(1-2):171–187, 2007.

[88] T. Dietterich. Machine learning for sequential data: A review. *Lecture Notes in Computer Science*, 2396:15–30, 2002.

[89] T. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[90] T. Mei, Y. Ma, H. Zhou, W. Ma and J. Zhang. Sports video mining with mosaic. In *Proc. of MMM 2005*, pages 107–114, 2005.

[91] T. Oates and P. Cohen. Searching for structure in multiple streams. In *Proc. of ICML 1996*, pages 346–354, 1996.

[92] V. Chandola, A. Banerjee and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):Article 15, 2009.

[93] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[94] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes and D. Pregibon. Squashing flat files flatter. In *Proc. of KDD 1999*, pages 6–15, 1999.

[95] W. Leland, M. Taqqu, W. Willinger and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[96] W. Lin and A. Hauptmann. Modeling timing features in broadcast news video classification. In *Proc. of ICME 2004*, pages 27–30, 2004.

[97] X. Zhu and X. Wu. Mining video associations for efficient database management. In *Proc. of IJCAI 2003*, pages 1422–1424, 2003.

[98] X. Zhu, X. Wu, A. Elmagarmid, Z. Feng and L. Wu. Video data mining: Semantic indexing and event detection from the association perspective. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):665–677, 2005.

[99] Y. Jiang, C. Ngo and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proc. of CIVR 2007*, pages 494–501, 2007.

[100] Y. Li and C. Kuo. *Video Content Analysis Using Multimodal Information: For Movie Content Extraction, Indexing and Representation.* Kluwer Academic Publishers, 2003.

[101] Y. Matsuo, K. Shirahama and K. Uehara. Video data mining: Extracting cinematic rules from movie. In *Proc. of MDM/KDD 2003*, pages 18–27, 2003.

[102] Y. Peng and C. Ngo. EMD-based video clip retrieval by many-to-many matching. In *Proc. of CIVR 2005*, pages 71–81, 2005.

[103] Y. Rui, T. Huang and S. Mehrotra. Constructing table-of-content for videos. *Multimedia Systems*, 7(5):359–368, 1999.

[104] Y. Tanaka, K. Iwamoto and K. Uehara. Discovery of time-series motif from multi-dimensional data based on MDL principle. *Machine Learning*, 58(2/3):269–300, 2005.

[105] Y. Zhai, A. Yilmaz and M. Shah. Story segmentation in news videos using visual and text cues. In *Proc. of CIVR 2005*, pages 92–102, 2005.

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

[107] Y. Zhai, Z. Rasheed and M. Shah. A framework for semantic classification of scenes using finite state machines. In *Proc. of CIVR 2004*, pages 279–288, 2004.

[108] Z. Gang, A. Kobayashi and Y. Sakai. Group-based relevance feedback using reduct of rough set theory for interactive image retrieval. *Journal of the Institute of Image Information and Television Engineers*, 56(6):884–893, 2005.

[109] Z. Rasheed and M. Shah. Scene detection in hollywood movies and tv shows. In *Proc. of CVPR 2003*, pages 343–348, 2003.

# Publication List

---

## Book Chapters

---

[1]   Kimiaki Shirahama, Koichi Ideno and Kuniaki Uehara, A Time-constrained Sequential Pattern Mining for Extracting Semantic Events in Videos, Multimedia Data Mining and Knowledge Discovery (Valery A. Petrushin and Latifur Khan Eds.), Springer, pp. 423 – 446, 2006.

---

## Journal Papers

---

[2]   Kimiaki Shirahama, Yuta Matsuoka and Kuniaki Uehara, Event Retrieval in Video Archive Using Rough Set Theory and Partially Supervised Learning, *Multimedia Tools and Applications*, (Accepted).

[3]   Kimiaki Shirahama and Kuniaki Uehara, A Novel Topic Extraction Method based on Bursts in Video Streams, *International Journal of Hybrid Information Technology (IJHIT)*, Vol. 1, No. 3, pp. 21 – 32, 2008.

[4]   Kimiaki Shirahama, Koichi Ideno and Kuniaki Uehara, Extracting Semantic Patterns in Video Using Time-constrained Sequential Pattern Mining, *The Journal of the Insitute of Image Information and Television Engineers (ITEJ)*, Vol. 60, No. 9, pp. 1473 – 1482, 2006 (In Japanese).

# International Conferences and Workshops

[5] Kimiaki Shirahama, Yuta Matsuoka and Kuniaki Uheara, Video Retrieval from a Small Number of Examples Using Rough Set Theory, In *Proc. of the 17th International Conference on MultiMedia Modeling (MMM 2011)*, pp. 96 – 106. 2011.

[6] Kimiaki Shirahama, Lin Yanpeng, Yuta Matsuoka and Kunaki Uehara, Query by Example for Large-Scale Video Data By Parallelizing Rough Set Theory Based on MapReduce, In *Proc. of 2010 International Conference on Scienece and Social Research (CSSR 2010)*, (to appear).

[7] Kimiaki Shirahama, Yuta Matsuoka and Kuniaki Uehara, Query by Few Video Examples Using Rough Set Theory and Partially Supervised Learning, In *Proc. of the Fifth International Conference on Semantic and Digital Media Technologies (SAMT 2010) (Poster)*, (to appear).

[8] Kimiaki Shiraham and Kuniaki Uehara, Video Retrieval from Few Examples Using Ontology and Rough Set Theory, In *Proc. of the Second Workshop on Semantic Multimedia Database Technologies (SMDT 2010)*, (to appear).

[9] Kimiaki Shirahama and Kuniaki Uehara, Example-based Event Retrieval in Video Archive Using Rough Set Theory and Video Ontology, In *Proc. of the 10th International workshop on Multimedia Data Mining (MDM/KDD 2010)*, Article No. 6, 2010.

[10] Kimiaki Shirahama, Chieri Sugihara, Yuta Matsuoka and Kuniaki Uehara, Query-based Video Event Definition Using Rough Set Theory and Video Prototypes, In *Proc. of IS&T/SPIE Electronic Imaging Multimedia Content Access: Algorithms and Systems IV*, 7540B-41, 2010.

[11] Kimiaki Shirahama, Chieri Sugihara and Kuniaki Uehara, Query-based Video Event Definition Using Rough Set Theory and High-dimensional Representation, In *Proc. of the 16th International Conference on Multimedia Modeling (MMM 2010)*, pp. 358 – 369, 2010.

[12] Kimiaki Shirahama, Chieri Sugihara, Kana Matsumura, Yuta Matsuoka and Kuniaki Uehara, Mining Event Definitions from Queries for Video Retrieval on the Internet, In *Proc. of the First International Workshop on Internet Multimedia Mining (IMM 2009)*, pp. 176 – 183, 2009.

[13] Kimiaki Shirahama, Chieri Sugihara, Yuta Matsuoka, Kana Matsumura and Kuniaki Uehara, Kobe University at TRECVID 2009 Search Task, In *Proc. of TREC Video Retrieval Evaluation (TRECVID) 2009 Workshop*, pp. 76 – 84, 2009.

[14] Kimiaki Shirahama, Chieri Sugihara, Yuta Matsuoka and Kuniaki Uehara, Query-based Video Event Definition Using Rough Set Theory, In *Proc. of the First ACM International Workshop on Events in Multimedia (EiMM 2009)*, pp. 9 – 15, 2009.

[15] Akihito Mizui, Kimiaki Shirahama and Kuniaki Uehara, TRECVID 2008 NOTEBOOK PAPER: Interactive Search Using Multiple Queries and Rough Set Theory, In *Proc. of TREC Video Retrieval Evaluation (TRECVID) 2008 Workshop*, pp. 123 – 132, 2008.

[16] Kimiaki Shirahama, Akihito Mizui and Kuniaki Uehara, Characteristics of Textual Information in Video Data from the Perspective of Natural Language Processing, In *Proc. of NSF Sponsored Symposium on Semantic Knowledge Discovery, Organization and Use*, P-24, 2008.

[17] Kimiaki Shirahama and Kuniaki Uehara, Query by Shots: Retrieving Meaningful Events Using Multiple Queries and Rough Set Theory, In *Proc. of the Ninth International Workshop on Multimedia Data Mining (MDM/KDD 2008)*, pp. 43 – 52, 2008.

[18] Kimiaki Shirahama and Kuniaki Uehara, A Novel Topic Extraction Method based on Bursts in Video Streams, In *Proc. of the Second International Conference on Multimedia and Ubiquitous Engineering (MUE2008)*, pp. 249 – 252, 2008.

[19] Kimiaki Shirahama, Kazuyuki Otaka and Kuniaki Uehara, Content-Based Video Retrieval Using Video Ontology, In *Proc. of the Third IEEE International Workshop on Multimedia Information Processing and Retrieval (MIPR 2007)*, pp. 283 – 288, 2007.

[20] Kimiaki Shirahama and Kuniaki Uehara, Video Data Mining: Discovering Topics by Burst Detection in Video Streams, In *Proc. of ICDM 2007 Workshop on Knowledge Discovery and Data Mining from Multimedia Data and Multimedia Applications (KDM 2007)*, pp. 57 – 62, 2007.

[21] Kimiaki Shirahama, Koichi Ideno and Kuniaki Uehara, Video Data Mining: Mining Semantic Patterns with temporal constraints from Movies, In *Proc. of the First IEEE International Workshop on Multimedia Information Processing and Retrieval (MIPR 2005)*, pp. 598 – 604, 2005.

[22] Kimiaki Shirahama, Yuya Matsuo and Kuniaki Uehara, Mining Semantic Structures in Movies, *Applications of Declarative Programming and Knowledge Management (Lecture Notes in Artificial Intelligence)*, Vol. 3392, pp.116 – 133, Springer, 2005.

[23] Kimiaki Shirahama, Kazuhisa Iwamoto, and Kuniaki Uehara, Video Data Mining: Rhythms in a Movie, In *Proc of the 2004 IEEE International Conference on Multimedia and Expo (ICME 2004)*, pp. 1463 – 1466, 2004.

[24] Yuya Matsuo, Kimiaki Shirahama and Kuniaki Uehara, Mining Semantic Structures in Movies, In *Proc. of the 15th International Conference on Applications of Declarative Programming and Knowledge Management*, pp. 229 – 240, 2004.

[25] Kimiaki Shirahama, Yuya Matsuo and Kuniaki Uehara, Extracting Alfred Hitchcock's Know-How by Applying Data Mining Technique, In *Proc. of the First International Workshop on Objects Models and Multimedia Technologies (OMMT 2003)*, pp. 43 – 54, 2003.

[26] Yuya Matsuo, Kimiaki Shirahama and Kuniaki Uehara, Video Data Mining: Extracting Cinematic Rules from Movie, In *Proc. of the Fourth International Workshop on Multimedia Data Mining (MDM/KDD 2003)*, pp.18 – 27, 2003.

# Domestic Conferences and Workshops

[27] Kimiaki Shirahama, Yuta Matsuoka and Kuniaki Uehara, Retrieving a large variety of videos from a small number of example videos based on rough set theory, *Technical Report of the IEICE Pattern Recognition and Media Understanding Workshop* (to appear) (In Japanese).

[28] Yuta Matsuoka, KImiaki Shirahama and Kuniaki Uehara K, Negative Selection for High Dimensional Feature in Video Retrieval with Query-by-Example Method, *Technical Report of the IEICE Pattern Recognition and Media Understanding Workshop* (to appear) (In Japanese).

[29] Chieri Sugihara, Kimiaki Shirahama and Kuniaki Uehara, Model Definitions of Video Event Retrieval from Queries and Rough Set Theory, In *Proc. of the 2009 Annual Conference of Information Processing Society of Japan Kansai Branch*, C-06, 2009 (In Japanese).

[30] Chieri Sugihara, Kimiaki Shirahama and Kuniaki Uehara, Classification of Video Keyframes with SIFT Algorithm, In *Proc. of the IEICE General Conference 2009*, D-12-61, 2009 (In Japanese).

[31] Akihito Mizui, Kimiaki Shirahama and Kuniaki Uehara, Improvement of Video Retrieval based on Feature Selection using Multiple Correspondence Analysis, In *Proc. of the IEICE General Conference 2009*, D-12-37, 2009 (In Japanese).

[32] Kana Matsumura, Kimiaki Shirahama and Kuniaki Uehara, Video Retrieval using Partially Supervised Learning, In *Proc. of the IEICE General Conference 2009*, D-12-4, 2009 (In Japanese).

[33] Yuta Matsuoka, Kimiaki Shirahama and Kuniaki Uehara, Improvement of Video Retrieval Using MPEG-7 Compliant Features, In *Proc. of the IEICE General Conference 2009*, D-12-3, 2009 (In Japanese).

[34] Kimiaki Shirahama and Kuniaki Uehara, Topic Extraction in Videos Based on Burst Detection, *Technical Report of the IPSJ Mathematical Modeling and Problem Solving Workshop (2009-MPS-73)*, pp. 85 – 88, 2009 (In Japanese).

[35] Kimiaki Shirahama and Kuniaki Uehara, Topic Extraction based on Burst Detection in Video Streams, In *Proc. of the 70th National Convention of IPSJ*, pp. 21 – 22, 2008 (In Japanese).

[36] Kimiaki Shirahama and Kuniaki Uehara, Dicoverying Topics in Video by Contious-time Burst Detection, In *Proc. of the ITE Annual Symposium 2008*, 6-4, 2007 (In Japanese).

[37] Kazuyuki O, Kimiaki Shirahama and Kuniaki Uehara, Content-based Video Archive Retrieval Using Video Ontology, In *Proc. of the ITE Annual Symposium 2008*, 6-6, 2007 (In Japanese).

[38] Koichi Ideno, Kimiaki Shirahama and Kuniaki Uehara, Ontology-based automatic annotation for semantic video retrieval, In *Proc. of the ITE Winter Symposium 2006*, 2006 (In Japanese).

[39] Kazuyuki Otaka, Kimiaki Shirahama and Kuniaki Uehara, Evaluation of Content-based Video Retrieval Using Semantic Patterns. In *Proc. of the ITE Winter Symposium 2006*, 2006 (In Japanese).