ADVISE: Advanced Digital Video Information Segmentation Engine

By

Chung-Wing NG

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Department of Computer Science & Engineering

Supervised by:

Prof. Michael R. LYU and Prof. Irwin King

July 2002

ADVISE: Advanced Digital Video Information Segmentation Engine

submitted by

Chung-Wing NG

for the degree of Master of Philosophy
at The Chinese University of Hong Kong

Abstract

Video over Internet is getting more popular in education, entertainment, and information sharing. Since it is time-consuming to download and browse most part of a video before we know the video contents, it is difficult to find out a piece of video that we want among the vast video sources in the Internet repositories. Therefore, in this thesis, we propose a web-based video retrieval and browsing enhancement system called ADVISE, Advanced Digital Video Information Segmentation Engine, to solve the above problem.

There are three major modules in ADVISE: the first one is the video table-of-contents (V-ToC) construction, the second one is the video summarization, and the third one is the video matching.

A V-ToC is an image-based video description, which provides users an abstract of the video contents. Since there are numerous images contained in a video, it is a difficult task to select and organize key images to show the video contents, unless we first understand the video structure. We use regional color histogram feature with the adaptive threshold to build up a 4-level tree structure for a video. According to this structure, key images are selected and organized into the V-ToC for describing the video contents. We present the V-ToC in a web-based format using XML with XSL. The V-ToC structure generated is evaluated in a set of experiment. We found that our current approach generate

the most accurate structure among four different settings.

The video summarization module extracts the major contents of a video. This module is important because different users have their different needs on extracting the video contents. We have proposed a video summarization algorithm, which accepts user's inputs to tailor the video contents extraction. The algorithm employs a statistical approach in order to select video features favored by the user. Furthermore, we have applied the clustering technique to refine the video segments selection, such that a smoother video summary can be generated. The resulting summary is delivered to the user in a SMIL presentation format. We have evaluated the video summarization algorithm using two experiments. The first one is used to measure the quality of the resulting video summary. We found that our algorithm is able to adjust the contents selection according to the user's preferences. For the second experiment, we evaluate the clustering results in the refinement. We found that the refinement process can reduce the quality of the video summary while improving the smoothness.

The video matching module is used to measure the similarity between two videos. There are many possible algorithms to match the video features in different temporal ordering, however seldom of them concern on the structure of videos. Now, based on the V-ToC tree structure mentioned above, we have proposed two video tree matching algorithms. The first one is the non-ordered tree matching algorithm and the second one is the ordered tree matching algorithm. They are different because the ordered tree matching algorithm considers the temporal ordering of video features while the non-ordered one does not. Our experiments on a set of various videos demonstrate that the proposed tree matching algorithms produce similar ranking results to what human will produce.

We have implemented the proposed ADVISE system to demonstrate our work. It provides all the features, which are mentioned above, to the Internet users. Hence, users can browse and retrieve videos efficiently using ADVISE.

ADVISE: Advanced Digital Video Information Segmentation Engine

submitted by

Chung-Wing NG

for the degree of Master of Philosophy

at The Chinese University of Hong Kong

論文摘要

現今的互聯網視頻越見普及。它可應用於教育、娛樂及資訊分享等方面。由於要了解視頻的內容，必需先把它下載，然後瀏覽大部份片段。這確是一件相當費時的工作。而在互聯網上的大量視頻來源中，我們更難逐一去根據內容來找到所想要的視頻。因此，在這一篇論文中，我們提出了一個可在網絡上應用的視頻提取及瀏覽系統，名叫 ADVISE，Advanced Digital Video Information Segmentation Engine(高階數碼視頻資訊分割引擎)。我們以此系統來解決上述所提及的問題。

ADVISE 包含了三個模組。其中，第一個模組負責提供視頻目錄(Video Table-of-Contents)，而第二個則負責提供視頻撮要(Video Summarization)，最後，第三個則負責視頻相似度的配對(Video Matching)。

視頻目錄是一個影像形式的視頻描述。它的作用是給我們了解視頻的大概內容。因爲視頻是集合了大量影像，所以從中抽取及整理較爲重要的影像來表現視頻內容是很困難的。我們必須先了解視頻的結構。我們運用了分區色譜柱狀圖的方法及可適應的分界值來建立一個視頻專用的四層樹狀結構。根據這個結構，我們可選取重要的影像及排列它們在視頻目錄中，用以描述視頻的內容。我們使用了 XML 及 XSL 技術將視頻目錄以網頁形式存放，令使用者

可以更方便地通過網絡到 ADVISE 瀏覽。爲了實驗上列視頻目錄結構的準確性，我們比較了四種不同的方法。由此發現，在 ADVISE 中使用了的方法是比較準確的。

另一個 ADVISE 提供的是負責視頻概要的模組。其主要目的是抽取比較重要的片段作爲視頻撮要，以減省使用者所須的瀏覽時間。由於使用者們各有所需，因此，我們便難於使用單一的方法去製作視頻撮要予不同的使用者。而在 ADVISE 中，我們的視頻撮要演算法是因應使用者所輸入的設定而提取視頻內容的。那演算法運用統計的方法來撮選使用者所喜愛的視頻特徵。除此之外，我們運用了尋找叢集技術來改善所得撮要的順暢度。在得出了視頻撮要後，ADVISE 把它製作成 SMIL 的格式。這樣使用者便可輕易地透過網絡接收到所需的視頻撮要。我們用了兩個實驗來檢討 ADVISE 的視頻撮要演算法。第一個實驗是用作了解所得出視頻撮要的質素。結果顯示出我們的演算法能有效地根據使用者的需要而選取視頻內容，在第二個實驗中，我們測試尋找叢集技術對所得撮要的效果。我們發現該尋找叢集技術能有效地改善順暢度。

最後是視頻配對模組，它是用作計算兩段視頻之間的相似度。雖然現有不少方法能配對視頻中的特徵，但甚少會重視視頻的結構。由於我們認爲視頻結構對計算相似度有著相當的重要性，因此，就製作視頻目錄的模組中所提及的樹狀視頻結構，ADVISE 提供了兩項演算法，用以配對視頻的結構。第一項演算法是非序列式的樹結構配對，而第二項演算法是序列式的演算的樹結構配對。兩項演算法的方別是在於序列式的演算法會依據視頻特徵出現的次序來配對，但非序列式的並不會重視該出現次序。我們以一系列的實驗，比較了所提出的演算法及人工的結果。這顯示出 ADVISE 所提供的演算法能有效地作出相似的配對。

我們集合了上述的三個模組，建立了 ADVISE 系統。它確能幫助使用者有效地在網絡上提取及瀏覽視頻。

Acknowledgment

I would like to take this opportunity to express my gratitude to my supervisors, Prof. Michael R. Lyu and Prof. Irwin King, for their generous guidance and patience to me during the research time. The inspiring advice from Prof. Lyu and Prof. King are extremely essential and valuable in my research papers, which is published in CISST2001 and WWW2002, as well as my thesis.

I am also grateful for the time and valuable suggestions that Prof. M.C. Lee and Prof. T.T. Wong have given in marking my term papers and term presentations.

I would also like to show my gratitude to the Department of Computer Science and Engineering, CUHK, for the provision of the best equipment and pleasant office environment for high quality research.

I want to give my thanks to my fellow colleagues, Edward Yau, Sam Sze, Kenny Kwok, Vincent Cheung, Jacky Ma, Joe Lau, Keith Wong, Ocean Cheung, K.H. Tsoi, Harvest Jang, Sunny Tang, and M.L. Ho. They have helped me in solving technical problems, enlightened me with new research ideas, and given me encouragement and supports. They have given me a joyful and wonderful time in my research.

Finally, my special thanks must go to my family who has given me the greatest support and encouragement, so that I can keep concentrated on my postgraduate study.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, since the rapid development of the Internet technologies, information sharing on the Internet is not limited in textual format. With the higher network bandwidth, people can retrieve information in the form of multimedia including images, audio, and particularly, video. According to the Home PC Portrait survey by PC Data, an estimated 57.2% of Internet users watched video clips, and 7.3% edited video clips on their personal computers, in the year 2000 [12]. Video is getting popular in education, entertainment and other multimedia applications [2][3]. It is because video enriches the content delivery by combining visual, audio, and textual information in multiple data streams. Now many companies provide video sharing services, which further speeds up the growth of the volume of Internet videos [25][41].

Under this evident growth, users may find it difficult while to search for some video contents they want from vast available sources. Hence, the management of video data on the Internet is an urgent need. It can help users to retrieve their favorite videos efficiently. Although many researchers have investigated this video content retrieval problem [21][52] and the designs of digital video libraries [9][16][22], there are still many interesting problems to be solved.

In this thesis, we propose a web-based system, called **ADVISE**, **Advanced Digital Video Information Segmentation Engine** [28]. The system can enhance video browsing and retrieval by providing a set of services through the

Internet. We propose three different solutions, which facilitates browsing and retrieval of video over the Internet. They are:

Image-based Video Description gives a brief description for users to know the video contents at once.

Video Summary reduces the time to browse a video by abstracting the important parts of video contents.

Video Matching measures the similarity between videos, such that videos with similar contents can be found efficiently.

We explain our work in these three areas in later sections.

1.1 Image-based Video Description

Without any descriptions about a video, we need to spend time to download and browse it before we know the contents. This is a time-consuming process when there is a large amount of videos available. It is impossible to know all the video contents, if we do not have any descriptions about them. A solution to solve this problem is hiring a person to annotate each video available. However, the man-power involved is too huge. We prefer to solve the problem through an automatic way. By preprocessing the raw video, we can automatically generate a video description, which can definitely help users in understanding the video contents while saving the time on downloading and browsing the whole video.

As we have mentioned above, video is a combination of images, audio and text, thus, it is a convenient way to generate the description by selecting video information from only one of these three aspects.

For text-based description, the most common generation method is the extraction of caption text from the video [11][23]. This kind of description with video captions is convenient for users to understand the video contents at once by reading through. It can be used on videos which always have captions, likes news broadcasts and movies. However, we find that there are many other videos

without enough captions for description, especially for personal video productions.

For audio-based description, the most popular method is the generation of textual transcripts using speech recognition [16][45]. This method is as easy to read as the text-based description above, and it can also distinguish the speakers through the speech. Since this approach can be applied only on videos with people's speech. Besides, most current speech recognitions cannot give very confident results except for those models with excellent training set [16].

For image-based description, the desirable way to present to video contents is to extract key video frame images [8][15][29][35][43]. With those extracted images well organized according to the video structure, we can concretely know what have been shown in the video, and estimate the whole video story by the organization. We find that videos may not always have caption and speech, but they seldom contain meaningless images only. For example, to describe the contents of a scenery video, an image-based description is the most appropriate.

Table 1.1 summarizes the pros and cons of the three different types of video descriptions.

Table 1.1 Summary of Video Descriptions

	Strengths	**Weakness**
Text-based description	• Easy to understand the meaning • Compact and small in size • Textual searchable	• Not applicable to videos without captions
Audio-based description	• Easy to understand the meaning • Compact and small in size • Textual searchable • Able to distinguish the speakers	• Not applicable to video with no human speech • Worrying accuracy about speech recognitions
Image-based description	• Able to see the actual video images • Applicable to most videos • Image-based searchable	• Larger in size while comparing with the text-based one

In this thesis, our proposed ADVISE system provides the image-based video description. It automates the generation of the visual descriptions for videos. As we mentioned above, video key frame extraction is needed for generating the description. Those key frame images are always extracted through the video segmentation process [8][15][29][35][43]. We will discuss various video segmentation processes in Section 2.2.

Instead of barely listing all the extracted key frame images, Unhihashi [43] prepares his image-based video description in the way similar to the representation in comic books. He calculates the importance score for each key frame images by the rarity and duration of its corresponding video segment. Then those key images are packed together according to the importance score, where the larger area for higher score. As a result, the resulting video description is in form of a comic book style.

While Unhihashi's idea, which calculates the importance scores in a quite simple way, gives only a few concerns on the video structure, Rui [35] proposes another image-based description, which can present the whole video structure. He calls his work as a Table-of-Content (ToC) for videos. It is because he finds that a video ToC, which functions similarly to the one in a book, can facilitate browsing of a video by capturing the structure in it. The video structure defined by Rui will be discussed in Section 2.2.

Now, we discuss briefly our image-based video description in our research work. We first employ the Rui's video structure and generate Video Table-of-Contents (V-ToC) in a modified approach [13][21][35][51]. Then we store the V-Toc in the eXtensible Markup Language (XML) format [46]. An XML Document Type Definition (DTD) is defined to provide grammars for the components of the video structure, and to maintain the consistency of XML. Hence, by using the eXtensible Stylesheet Language (XSL) [47], we further transform the XML video structure into a well readable web interface, which is used as the image-based video description on the Internet. We will discuss the XML technologies in Section 2.3.

1.2 Video Summary

Apart from the video description discussed above, the second goal for ADVISE is to generate video summary for the users. We notice that users may not be easily satisfied by the video description mentioned above with image-based information only. The video description can just help them to remove most unwanted videos, such that they can further investigate those remaining. Instead of attempting to guess the video contents at once through the image-based video description, users are now willing to spend some more time on getting more information from the selected video. Therefore, a video summary, which is an abstract of the important parts of the video, is well suitable for user's need. Table 1.2 summarizes the differences between the image-based video description and the video summary.

Table 1.2 Differences between Video Description and Video Summary

	Image-based Video Description	**Video Summary**
Time	Can be read at once	Takes certain duration to play
Format	Key frame images	Video
Features captured	Only image-based features	All kinds of features including image-based, audio-based, and text-based
Usage	To give the rough idea about the contents	To show selected important contents

The key problem in video summarization is the selection of important segments from the source video. Since we find that quality of summarization depends greatly on the interest of each target user, it is almost impossible to design a single approach to fit into every user's appetite. However, by accepting settings from each user before summarization, it is easier to determine which kinds of contents are more important to the user. This kind of user input model is widely used in other video retrieval applications [30][39][40][50], but it has not yet been applied on video summarization.

Once the contents for the video summary are confirmed, the next question is about the format of video summary, which can be generated and returned efficiently through the Internet. The Synchronized Multimedia Integration Language (SMIL) [48] is a good choice for us to solve this problem. SMIL is often applied to various online video personalization systems [17][18][30]. Through similar mechanisms, we can then easily define an individual SMIL presentation, which customizes contents from the source video. The resulting SMIL presentation can be returned back to the user through a streaming protocol on the Internet [32]. We will detail the SMIL technology in Section 2.4.

1.3 Video Matching

Instead of getting information from each video using video description and video summary, ADVISE also enables the matching of similarity between two videos. There are numerous solutions worked out by different researchers [1][24][26]. However, this problem is still challenging since it is difficult to extract video features that represent the content for matching, and there are many possible algorithms to match the temporal ordering of video features.

Mohan suggests an efficient video matching method [26]. He first extracts the DC coefficients of frames from MPEG videos. These DC coefficients of frames are used to formulate a sequence of feature vectors called fingerprints, in order to represent the video actions in the corresponding video sequence. Then, he applies sequential matching to find out similar fingerprints for videos in the database.

Instead of using general sequential matching, Adjeroh et al. [1] design a different scheme for matching video sequences. He models the video matching problem as a pattern matching problem. Various video features can be extracted from a video sequence, and are formulated into a video string (vString), which is a sequence of symbols, to represent the video sequence. He proposes a new approach of measuring string edit distance specialized for vString, so that such resulting distance is used to represent the similarity between video sequences.

Shearer et al. [38] suggest focusing more on the similarity of video frame images. He expresses the image similarity as the graph isomorphism detection problem. The graph is encoded according to the relationships between video objects over time [53]. He then designs a decision tree algorithm to detect the isomorphic sub-graphs between two video sequences. Thus, the videos are similar if their graphs are isomorphic.

Lienhart's method [24] pays more attention to the structure of videos, comparing with the above approaches. He suggests that video should be examined in different levels, where these levels are frame, shot, scene and video. A recursively approach is applied over those levels of temporal resolution, so that reordered sequences can be detected through a re-sequencing measure proposed.

By regarding Rui's video structure [35] as a hierarchical tree structure, we can apply our own structural matching algorithm to measure the video similarity. We propose two tree matching algorithms [27], they are the non-ordered tree matching algorithm and the ordered tree matching algorithm. They are different because the ordered tree matching algorithm is constrained by the temporal ordering but the non-ordered tree matching algorithm is not. Therefore, the ordered tree matching algorithm can be applied for matching video with similarities in both structure and video features, while the non-ordered tree matching algorithm can be applied for matching video with similar video features only.

1.4 Contributions

Our research work makes the following contributions:

1. We have proposed the whole framework of a video browsing and retrieval system called **ADVISE, Advanced Digital Video Information Segmentation Engine** [28].
2. We have proposed the automated generation of an image-based video description in ADVISE. The video description is called **Video**

Table-of-Contents (V-ToC). It provides users a convenient way to estimate the video contents at once. The video description in XML format is transformed into a web-based format using XSL.

3. We have proposed the **summarization of video** in ADVISE. It generates a video summary containing the most important parts of the source video. With the video summary, a user can know more about the video contents than browsing the video description. A user input model is used to tailor video summary with contents according to user's preferences. Video summary are delivered in a SMIL presentation, so that it can be generated efficiently and the user can easily access it through the Internet.

4. We have proposed two **video tree matching algorithms** [27] in ADVISE. They measure the similarity between videos. The first algorithm is the non-ordered tree matching algorithm, and the second one is the ordered tree matching algorithm. These similarity measures enable users to search videos with similar contents efficiently.

1.5 Outline of Thesis

In Chapter 2, we describe several technologies related to our research work. First, we describe the video pre-processing in digital video libraries. Also, we review the video structuring techniques carried out on top of the video segmentation results. Besides, we discuss the XML and SMIL technologies, which are used in our proposed system.

In Chapter 3, we describe the design of the proposed system, ADVISE [28]. It provides a set of services, which assists users in browsing and retrieval of videos over the Internet. We detail the application interface of each component in ADVISE and describe the relations between those components.

In Chapter 4, we detail the construction of the Video Table-of-Contents (V-ToC). We illustrate the extraction of video features and we use these

features in the structuring process of the video. This structuring process results in the V-ToC of a video, which is formatted into a XML structure, and is presented in a web-based format using XSL. At the end of this chapter, we carry out experiments to evaluate the video structuring results.

In Chapter 5, we discuss our video summarization algorithm. We illustrate how to select contents into the video summary by considering user's inputs. Also, we talk about the presentation of the resulting video summary in a SMIL format. At the end of chapter, we evaluate our video summarization.

In Chapter 6, we describe the algorithms for matching the video similarity. We detail first the two proposed video tree matching algorithms [27] and make comparison between them. Besides, a set of experiments is designed to demonstrate the results of our video matching algorithms.

Finally, we conclude this thesis in Chapter 7.

Chapter 2

Literature Review

In this chapter, we discuss several technologies related to our research work.

In Section 2.1, we describe how digital video libraries process videos before they are ready for retrieval. We take the VISION project [22] and the INFORMEDIA Project [9][15][44][45] as examples, so that we know how they currently work without a better structuring of videos. We expect that the structuring techniques in the following section can further assist the retrieval of video in digital video libraries.

In Section 2.2, we have a review on building video structures. It includes various video segmentation and structuring techniques. Since our video description, V-ToC, is constructed based on the video structure, so these techniques greatly affect the quality of the V-ToC.

In Section 2.3, we give an overview of XML technologies that are used in our work. It includes the introduction of XML syntax, DTD, and XSL. These technologies help us in storing and presenting the V-ToC properly.

In Section 2.4, we briefly introduce the SMIL technologies. SMIL is a markup language designed for performing multimedia presentations. More and more video personalization applications use SMIL presentations to deliver their customized videos. The major reason is the easy generation of SMIL. We present the common model of those applications in this section. In our research, we apply SMIL on the presentation of our video summary.

2.1 Video Retrieval in Digital Video Libraries

Digital Video Library is a comprehensive system, which integrates a variety of video analyzing techniques, including speech recognition, face recognition, and video caption extraction, to provide content-based indexing and retrieval of video to users [15][22]. To enable these services, raw videos will undergo two key pre-processing steps. The first step is the video features extraction and the second step is the video segmentation. In the following paragraphs, we will discuss these steps in two well-defined digital video libraries, the VISION project proposed by Li et al. [22] and the INFORMEDIA Digital Video Library project at Carnegie Mellon University [9][15][44][45].

2.1.1 The VISION Project

In the VISION project, three different video features are extracted. The first one is to construct the color distributions for video images through a histogram-based image analysis. Although it tries to have a rough matching of the objects appearing on video, this approach provides a very good balance between an efficient extraction and acceptable image similarity metrics [22]. The second feature is the video captions. Captions are extracted from video frames and divided into tokens (words). These tokens are reduced to their word stems, and stop words are removed. So, they become the keywords to represent the video. The third feature extracted is the audio energy level from the audio track of the raw video.

By using those features extracted, VISION carries out segmentation of the videos. With the color histograms, VISION identifies shots by dividing the video at the sharp histogram changes. The resulting video shots are examined for possible merging of related shots into scenes using the extracted audio energy level and the keywords. If there are people talking at the shot transitions, it results in a high audio energy level. VISION expects shots with high audio

energy level at the transition can be merged for presenting a series of related topics. With the caption keywords, VISION evaluates the contents relevancy for shots by counting the number of same keywords appearing in them. The shots relevant in textual contents are further merged to video scenes as the final segmentation result.

2.1.2 The INFORMEDIA Project

The INFORMEDIA project employs a more complex video features extraction model. The video features extraction can be classified into three categories, the audio analysis, the image analysis, and the natural-language processing [15][45]. For the audio analysis, in addition to extracting the audio energy level similar to VISION, INFORMEDIA generates the full transcript, which is more informative, by automatically using speech recognition techniques. In the image analysis, INFORMEDIA also extract color histograms and caption text from video frames. In addition, it detects two more video features. First, human faces appearing on video are detected as one video feature. Second, video motions are extracted as another video feature. INFORMEDIA uses the camera motion approach, which tracks changes of individual regions in frames, and creates a vector representation of motions. In the natural-language processing, INFORMEDIA investigates the content relevancy based on the transcript results from the speech recognition and the caption text extraction. Similar to VISION, INFORMEDIA performs keywords stemming to produce a textual description for video. Probabilistic matching is also applied on those keywords to return an ordered ranking on video content relevancy.

INFORMEDIA proposes three video segmentation approaches using different video features extracted. The first approach is a simple color histogram difference measure, which is equivalent to the shots detection in VISION. It is efficient to give an initial segmentation. The second approach improves the first approach by considering both image features and audio features. In addition to the audio energy level used in the scene formation of VISION, the speech recognition result is used to determine the contents changes, and consequently, the

approach becomes content-based and more reliable. The third approach is to consider the camera motions such as zooming, panning, and forward camera motions. This method can demonstrate the image flow, but it does not promise a content-based segmentation.

2.1.3 Discussion

We now discuss the video segmentation process of both the above projects and compare with our proposed system, ADVISE.

Both VISION and INFORMEDIA projects use simple histogram-based video shots detection as the basic video structure. However, they discover that using the histogram approach is not sufficient for a content-based segmentation. Therefore, other than the image-based feature, they employ also textual and audio features of video in their segmentation process. They do not give much concern on the structure of the video. Both of the projects use the shot-based structure. It is only a sequential segmentation according to the different video features collected. As a result, they cannot give any information about the organization of the video contents.

In our ADVISE system, we focus on the structure of videos. We generate the video structure using the histogram-based approach, without the assistant of textual and audio features. We expect that a hierarchical video structure, which we will discuss in Section 2.2, can provide a more organized image-based analysis mechanism for the video contents. Thus, compared with the above projects, ADVISE can efficiently generate the video structure with only the histogram-based method. Besides, it can give a good organization while integrating with other types of video features. Therefore, the video structure can enhance the video indexing and retrieval of the digital video libraries in the latter stages.

We summarize the difference of video segmentation between VISION, INFORMEDIA and ADVISE in Table 2.1.

Table 2.1 Comparison between VISION, INFORMEDIA and ADVISE

	VISION	INFORMEDIA	ADVISE
Feature Used	Color histogram, caption text and audio energy level	Color histogram, caption text, human faces, audio energy level and speech in video	Color histogram
Structure of Video	Shot-based structure	Shot-based structure	A video structure with 4 levels of video components
Pros	• Multi-modal for using various video features	• Multi-modal for using various video features	• Structure generated efficiently • Show the organization of contents
Cons	• Moderate complex for using 4 features • Give no information about the contents organization	• More complex for using too many features • Give no information about the contents organization	• Use only the color histogram feature

In the following section, we discuss about the video structure employed in ADVISE.

2.2 Video Structuring

According Rui's definition, a video can be decomposed into a well-defined structure consisting of five levels [35].

1. Video shot is an unbroken sequence of frames recorded from a single camera. It is the building block of a video.
2. Key frame is the frame, which can represent the salient content of a shot.
3. Video scene is defined as a collection of shots related to the video content, and the temporally adjacent ones. It depicts and conveys the concept or story of a video.

4. Video group is an intermediate entity between the physical shots and the video scenes. The shots in a video group are visually similar and temporally close to each other.
5. Video is at the root level and it contains all the components defined above.

The hierarchy of these video components is demonstrated in Figure 2.1. We can transform the hierarchy into a structured format as shown in Figure 2.2. This structure can be regarded as a specialized tree whose tree depth equals to four. In the following sections, we review the video segmentation and structuring techniques using the video components defined.

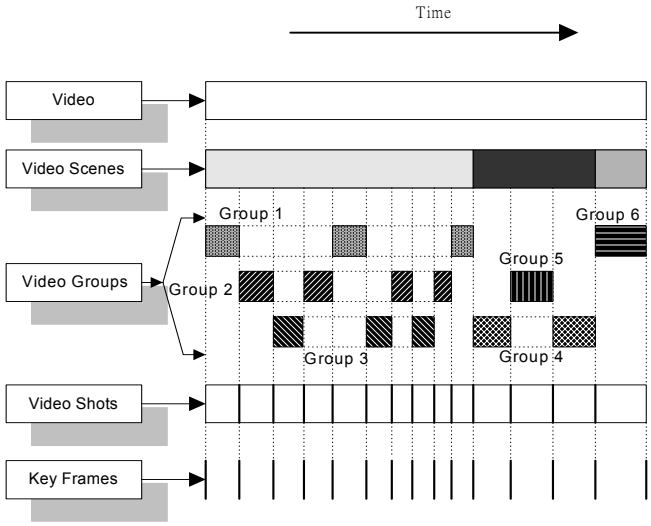


Figure 2.1 Hierarchy of Video Components

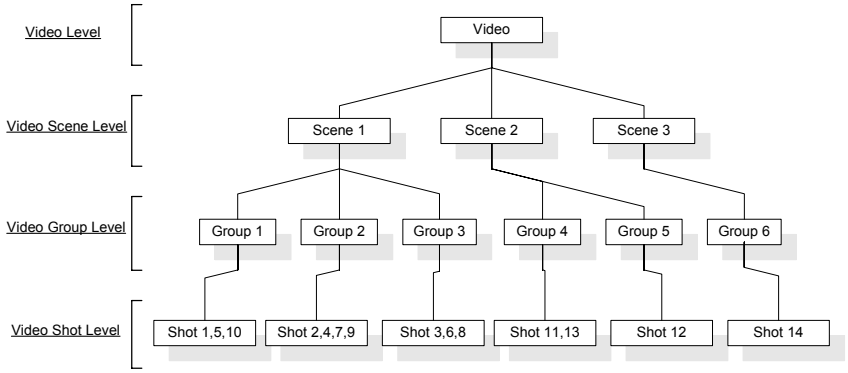


Figure 2.2 Video Tree Structure

2.2.1 Video Segmentation

The first step to structure a video is the segmentation process. There are various kinds of video segmentations. However, most of them try to segments the video at certain points of discontinuity. Hanjalic et al. [15] illustrate the general idea of video segmentation as shown in Figure 2.3. Two video frames are selected from the sequence and used as input. In the segmentation, required features are first extracted from the input frames. Then, a metric is used to quantify the feature variation from the two selected frames. The resulting discontinuity value is the magnitude of the variation. Hence, we can say that there is a discontinuous point while the discontinuity value is greater than certain threshold values or functions in the detector.

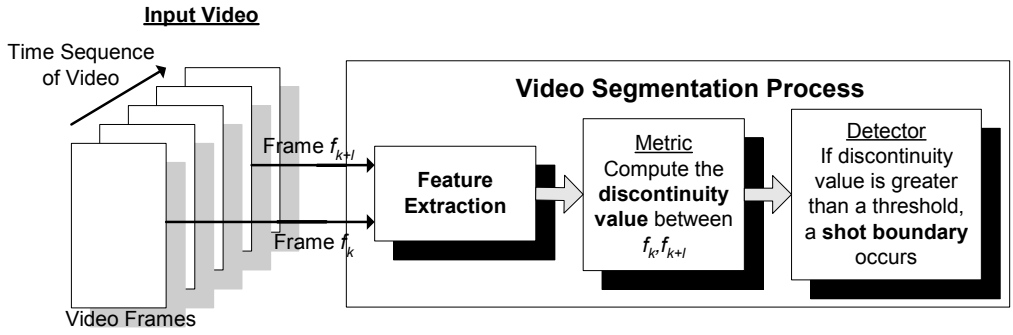


Figure 2.3 Illustration of Video Segmentation Process

The major difference between various video segmentations is the extraction of different features. Most of the video segmentation methods can be categorized into image-based, audio-based and text-based. Some other methods are always the mixtures of those categories. A simple audio-based method segments a video according the sharp changes in audio energy level [22][45], while a simple text-based segments a video by the detection of updates of caption text [23]. For the image-based segmentation, most of the methods attempt to detect the discontinuity of camera shots in the video. Different image-based features are used for shot boundary detection [6][7][13][15][51]. Among those features, we find that intensity level of pixels [5][6][15][51], color histograms

[6][7][13][15], and motion vectors [4][6][15] are the most efficient.

Our research work focuses mainly on the approach based on color histograms. It is because the histograms approach [42], which collects the global color information in each frame, is considerably accurate on passing object motions in video shots. The intensity level and motion vectors approaches are always over sensitive to that case.

2.2.2 Color histogram Extraction

For each input video frame, we build up a color histogram. A color histogram can be collected using various color models, RGB, YIQ, CMY, HSV, etc. A color model is divides into a number of bins, where each bin contains a range of color values. Then those bins collects the color pixels on frame with have color values fall into the same range. As a result, each value in the histogram is the number of collected pixel in a bin. Here we give an example of color histogram using a 64-bin RGB model and the corresponding video frame image on Figure 2.4.

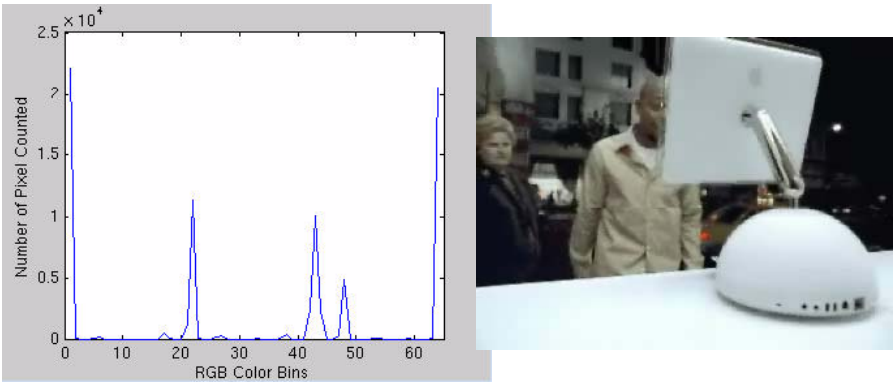


Figure 2.4 An Example Color Histogram

Once color histogram is constructed for each frame, we can use different distance functions [13] to obtain the value of discontinuity value described in Section 2.2.1. The simplest distance function is the sum of absolute differences of corresponding bins for both histograms. Then we can determine the shot

boundary using a threshold. A shot boundary occurs in between two video frames if the discontinuity value is greater than the threshold value; otherwise, no shot boundary occurs. After the shot boundary detection, a video is divided into shots, which consist of a sequence of similar frames.

2.2.3 Further Structuring

In order to provide an effective grouped analysis, some researchers suggest reducing the number of video fragments by grouping similar shots into a video group [2][15][21][24][29][35][52]. A cluster-based approach [35][52] can be applied on the video group formation. With the resulting video groups, we can go through some higher level analysis of the video contents [15][29]. First, we can know the organization of video materials along the video sequence. Besides, we can figure out which group of materials is rather important according to the number of member shots and shots' durations. Therefore, in the video group level, we start to explore more characteristics of the video contents.

Rui [35] further defines the video scene level by composing of content related groups. He points out that the video group formation, which takes only the visual content into account, leads to the lost of the temporal factor. The resulting video groups are discontinuous in time sequence; however, it always happens that adjacent video shots are related in the video contents, although they do not belong to the same video group. Therefore, Rui applies a time constrained clustering technique to further collect the video groups into video scenes. These resulting scenes are continuous in time sequence, and should be more understandable than the video groups. We illustrate the relation between video groups and video scenes in Figure 2.5.

The use of video groups and video scenes are important in video analysis. It is because there is no exact method to understand the contents in a video. However, with video groups and video scenes, we can be easier to examine the video contents according to the video structure. The video groups and video scenes provide the organization of video shots. It is directly related to how the

video contents, which are shown in video shots, are linked up together. Thus, it is more effective to understand a video from scene to scene instead of from shot to shot.

In our ADVISE system, we extend the use of Rui's video structure in two ways. First, we apply the video structure on building an image-based video description. The video description can clearly illustrate the organization of the video using a set of well-arranged video key frames. It enables users to understand to the video contents immediately. Second, we apply video matching technique on the video structure. We measure the similarity of video features appeared on the video along the video structure. It results in a structural matching for the videos. We will discuss these techniques in the later Chapters.

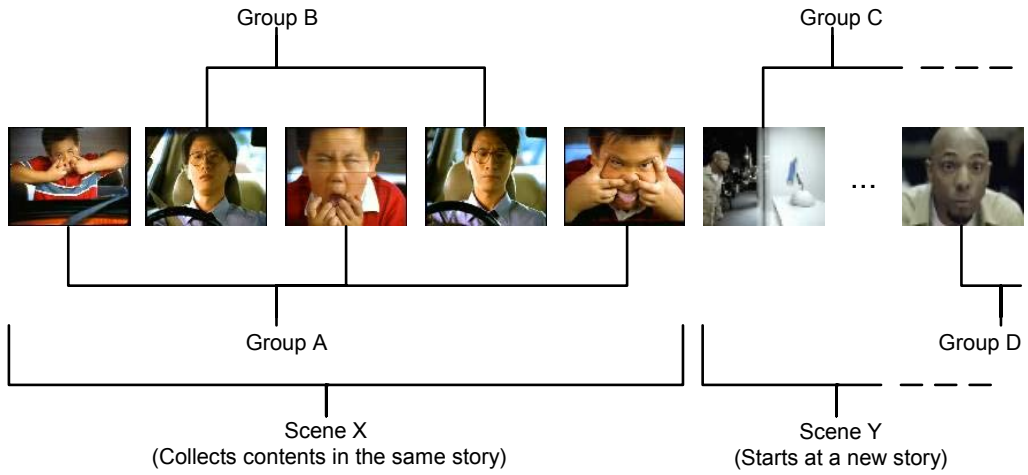


Figure 2.5 Illustration of Video Groups and Video Scenes

2.3 XML Technologies

Extensible Markup Language (XML) [46] is a standard adopted by the World Wide Web Consortium (W3C) in 1998. It is a restricted form of the standard generalized markup language. The design of XML makes documents flexible, easily accessible, and independent of platform. These features bring XML in advanced to our document description languages, and thus, lead to the widely uses

of XML in various kinds of applications. XML gradually becomes the standard for exchange and representation of data and information.

In our research work, we apply XML on the storage of the V-ToC. XML is used because it can provide a flexible nested hierarchical structure, which is well suitable for the V-ToC structure described in Section 2.2. Also, XML can be transformed easily into a web-based format using XSL, and hence, the V-ToC structure is accessible on the Internet.

2.3.1 XML Syntax

XML [46] represents data in a plain-text format. A pair of markup tags is used to encapsulate a piece of data, which is then called an XML element. Other than textual data, XML elements can contain other elements, such that we can build up a nested hierarchical structure. XML also allows us to associate attributes with an element. These attributes act like the properties for the data element. Example 2.1 illustrates the basic syntax of XML.

Example 2.1 *Refer to Figure 2.6, we use XML to represent details of a book. The pair of tags* `<book>` *and* `</book>` *encapsulates other elements like,* `<title>`*,* `<year>`*, etc. These elements build up a hierarchy as shown in Figure 2.7. Attribute* `unit` *with value* `cm` *defines a property for the* `<height>` *element.*

```
<book id="7">
  <title>ASP, ADO, and XML complete</title>
  <year>2001</year>
  <subjects>
    <topic>Internet programming</topic>
    <topic>Web sites design</topic>
  </subjects>
  <descriptions>
    <pages>1012</pages>
    <height unit="cm">23</height>
  </descriptions>
</book>
```

Figure 2.6 An Example of XML Document

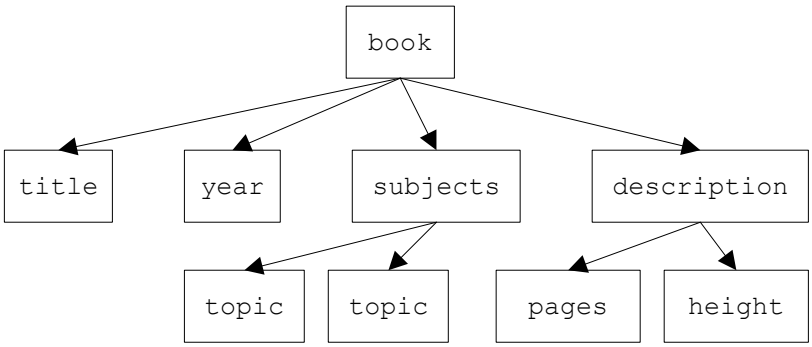


Figure 2.7 Tree Hierarchy of the Example XML Document

2.3.2 Document Type Definition, DTD

To construct a new structure using XML, we always need a DTD to maintain the consistency of the structure [49]. We define all the elements, attributes and relative association of elements inside a DTD, which serves as a grammar book to check for any exception in the XML document. Here we define the DTD for the above example XML document at Figure 2.8.

```
<!ELEMENT book (title, year, subjects, description)>
  <!ATTLIST book id CDATA #REQUIRED>
  <!ELEMENT title (#CDATA)>
  <!ELEMENT year (#CDATA)>
  <!ELEMENT subjects (topic+)>
    <!ELEMEMT topics (#CDATA)>
  <!ELEMENT description (pages, height)>
    <!ELEMENT pages (#CDATA)>
    <!ELEMENT height (#CDATA)>
      <!ATTLIST height unit CDATA #REQUIRED>
```

Figure 2.8 DTD for the Example XML Document

2.3.3 Extensible Stylesheet Language, XSL

Since XML does not provide any presentable interface by itself, then XSL [47] plays an important role in transforming XML documents to a neat presentation. XSL provides filtering and sorting functions such that we can select and well order data into the output presentation. HTML is a common output format while we use XSL transformation. Thus, the XML data can be presented using a

web-based interface. Here we give an example XSL transformation and its output at Figure 2.9 and Figure 2.10 respectively.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html><body>
   <xsl:for-each select="book" order-by="number(@id)">
    <hr/>Title: <xsl:value-of select="title"/><br/>
    Year: <xsl:value-of select="year"/><br/>
    Pages: <xsl:value-of select="pages"/><br/><hr/>
   </xsl:for-each>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```

Figure 2.9 XSL for the Example XML Document

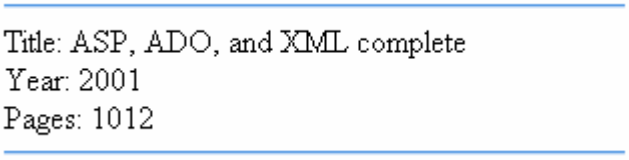


Figure 2.10 Web-based Output Presentation for the Example

2.4 SMIL Technology

Synchronized Multimedia Integration Language (SMIL), which is first announced on 1998, is another standard recommended by W3C [48]. It is designed for performing synchronized multimedia presentation on the Internet. We can use SMIL to specify the temporal behavior of the presentation, design the layout on the screen, and associate media objects with hyperlinks. SMIL documents are actually in XML format with well-defined DTD, such that SMIL browsers can interpret the tags and make proper layouts or operations. This mechanism is similar to opening HTML on common web browsers. There are several SMIL browsers available on the Internet, e.g., GRiNS [31] and RealPlayer [32] [33]. RealPlayer allows SMIL presentation to load streamed media clips through the Real Time Streaming Protocol (RTSP), so that the playback always keep synchronized according to the timeline [32].

2.4.1 SMIL Syntax

Since SMIL [48] is in form of a specific and well-defined XML, it uses markup tag pairs to define media objects in the presentation. We demonstrate the use of SMIL in Example 2.2.

Example 2.2 *Figure 2.11 shows the source of a sample SMIL presentation. The presentation consists of two video clips,* `1.rm` *and* `2.rm`*, which are streamed from different hosts using the RTSP protocol. We use* `clip-begin` *and* `clip-end` *attributes at the* `video` *element to define the relative position at the source videos. For example, we have taken the video segment from the* `0`*-th second to the* `8`*-th second from* `1.rm`*. Besides, we can define the layout of the presentation using* `layout`*,* `root-layout` *and* `region` *tags. We can play the SMIL with a RealPlayer [33] as shown in Figure 2.12.*

```
<?xml version="1.0"?>
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language">
<head>
  <layout type="text/smil-basic-layout">
    <root-layout width="550" height="300"/>
    <region id="r1"/>
  </layout>
</head>
<body>
  <seq>
    <video src="rtsp://host1/1.rm" clip-begin="0s" clip-end="8s" region="r1"/>
    <video src="rtsp://host2/2.rm" clip-begin="5s" clip-end="15s" region="r1"/>
  </seq>
</body>
</smil>
```

Figure 2.11 An Example for SMIL Source

2.4.2 Model of SMIL Applications

SMIL is often applied to online video personalization systems [17][18][30]. Two main reasons make SMIL favorable. First, SMIL benefits from the XML plain-text property. Web server can receive selections of preferred video clips from users through a web interface, and instantly generate the corresponding SMIL presentation with server-side scripting languages such as PERL, PHP, ASP, etc. The selected video clips are wrapped by SMIL and played according to

users' preference [17]. This model is demonstrated in Figure 2.13. The second reason is the network and client adaptability of SMIL [18]. It can dynamically configure the most appropriate media object for streaming, which depends on client display capabilities and connection speed. It would be convenient and safe for the SMIL browser on the client side to handle these limitations, instead of including additional considerations while generating the SMIL presentation.

Figure 2.12 An Example for SMIL Presentation

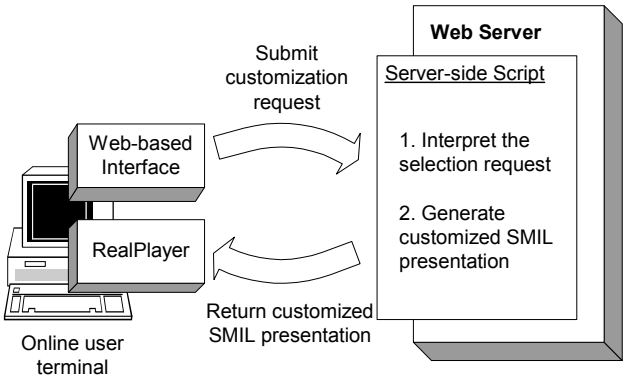


Figure 2.13 Model for Video Personalization Systems using SMIL

Chapter 3

Overview of ADVISE

In this chapter, we are going to describe the design of the ADVISE system [28]. ADVISE is a web-based video browsing and retrieval system. It provides a set of services to help users in understanding the video contents and the retrieve similar videos efficiently.

In Section 3.1, we first discuss the objectives for the design of ADVISE. There are four major objectives that ADVISE can achieve. We aim at enhancing the browsing and retrieval of video on the Internet through these objectives.

In Section 3.2, we describe the detail system architecture of ADVISE. There are three major modules in ADVISE. They are the video processing module, the web-based video retrieval module, and the video streaming server.

The web-based video retrieval module is, in fact, the interface for Internet users to access the video browsing and retrieval services provided by ADVISE. These services are V-ToC presentation, generation of SMIL video summary, and the query of similar videos. We briefly describe how they perform in Section 3.2.2. However the detail mechanism under these services will be discussed in later chapters.

We demonstrate the ADVISE system using the application interface that we have implemented. A set of screenshots is shown to illustrate how the ADVISE system works.

In Section 3.3, we summarize the major parts of this chapter.

3.1 Objectives

As we have mentioned in Chapter 1, there are not enough information provided for users to select a piece of video that they want from the abundant video sources in the Internet repositories. Therefore, it is an urgent need to design a system which provides a better management of those videos on the Internet.

In this thesis, the system we designed for the above purpose is called ADVISE. There are four major objectives for the design of ADVISE:

1. to provide a efficient way to describe the video contents;
2. to save the time for browsing the whole video to know the contents;
3. to search videos with similarity in certain video features; and
4. to provide services through the Internet.

According to these objectives, ADVISE is designed to provide a set of services on the Internet. These services enable users to have a better understanding of the video contents and an efficient retrieval of the videos.

3.2 System Architecture

The system architecture of ADVISE is shown in Figure 3.1, it consists of a video preprocessing module, a web-based video retrieval module, and a video streaming server. We discuss these three modules in the following sections.

3.2.1 Video Preprocessing Module

The video preprocessing module is used for generating a structure of Video Table-of-Contents (V-ToC) for a new input video and performing the similarity matching of the new video with other videos in the database.

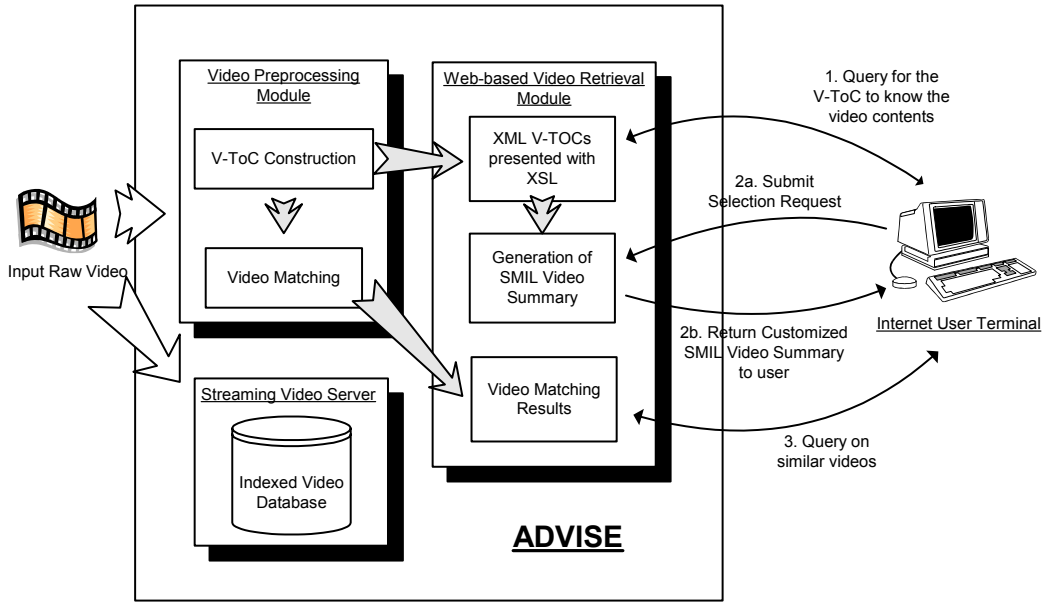


Figure 3.1 System Architecture of ADVISE

This video preprocessing module is an offline module operated by the administrator of the ADVISE system. We have implemented it using Visual C++ with DirectX. The interface is shown in Figure 3.2. We can input different types of raw videos into this module, for example, AVI, MPEG, Quick Time Movie, and Windows Media Video (WMV), which is provided by DirectX. The administrator preprocesses any new video source and prepares the video information for user to retrieve through the Internet.

There are two steps in this module; V-ToC construction and video matching.

Step 1: V-ToC Construction

We first structure the input video into a four level video tree structure called V-ToC. It extracts video features from the video as the ground information in video segmentation. Videos are segmented into four types of video components, video shots, video groups, video scenes and the video itself. We organize these video components hierarchically into a tree-like structure. Since this tree structure can show the organization of the video contents, so it acts as the table-of-contents for a video.

We can select several video segmentation options on the setting panel of this preprocessing module as shown in Figure 3.3. We can select whether to use an adaptive threshold instead of a fixed threshold, as well as the weighted regional histograms instead of a global color histogram for each video frame. Once the V-ToC structure is generated, we can save it into an XML format as shown in Figure 3.4. The XML V-ToC is then ready for presentation in the web-based video retrieval module in the later section. The detail mechanism of the V-ToC construction will be described in Chapter 4.

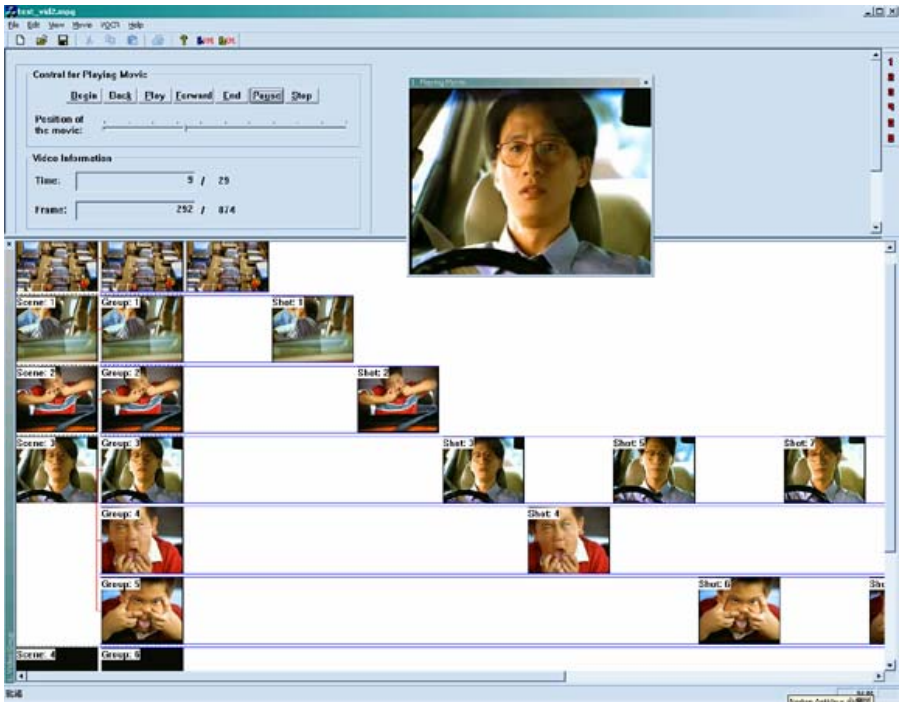


Figure 3.2 The Implementation of the Video Preprocessing Module

Step 2: Video Matching

Other than V-ToC generation, we proceed the video matching on this preprocessing module. We compare videos based their corresponding V-ToCs. Occurrences of video features in both videos are the key factors to determine the similarity between them. Since the V-ToCs are in the form of a specialized tree structure, we employ two tree matching algorithms to measure the similarity between them. The two matching algorithms are (i) the non-ordered tree

matching algorithm and (ii) the ordered tree matching algorithm. These algorithms are different because the ordered tree matching algorithm concerns on the temporal ordering of video features while the non-ordered tree matching algorithm does not. As a result, the ordered approach can show the result of similarity in video structure while the non-ordered approach can show the result of similarity in video contents. Therefore, both results are useful references for a user to look for a similar video that he may want. We will discuss these video matching algorithms in detail in Chapter 6.

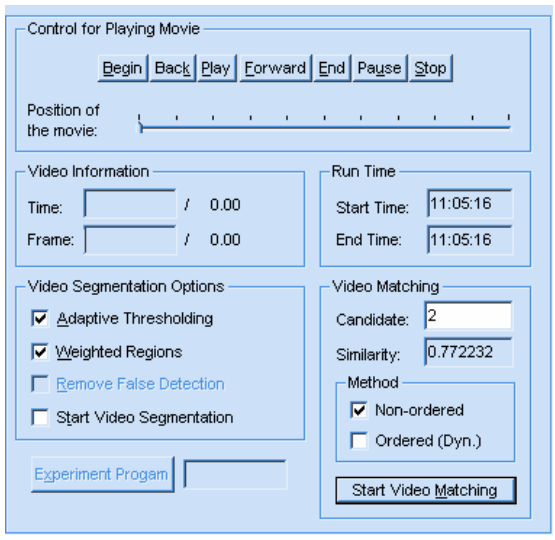


Figure 3.3 Setting Panel for the Video Preprocessing Module

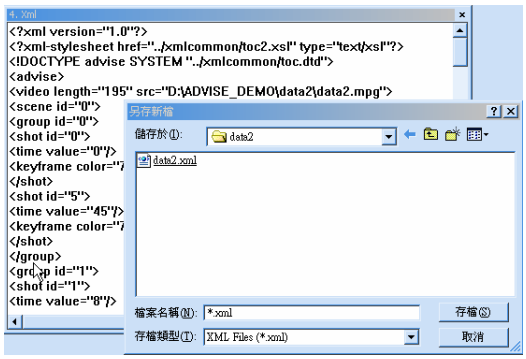


Figure 3.4 Save V-ToC Structure into XML

As it is shown on the setting panel in Figure 3.3, we can perform both the non-ordered video tree matching and the ordered video tree matching algorithms to get the similarity scores. The ADVISE administrator measures the similarity score between the new input video with other videos stored in the video database. The resulting scores are recorded and saved in a database. These scores are prepared for users to query for similar videos, through the web-based video retrieval module in the next section.

3.2.2 Web-based Video Retrieval Module

The web-based video retrieval module is in fact the interface for users to access the services provided ADVISE.

On the server side, all the contents of this module are prepared in a web-based format. A web server, which is capable to execute PHP scripts, is used to transmit our contents to the user. The PHP scripts will be used in some of the services below.

At the client side, a user is suggested to use the media browser of RealOne Player [33] to browse the contents of ADVISE. It is convenient for the user to retrieval video at the same time using the player.

There are three web services provided by ADVISE. They are the V-ToC presentation, the generation of SMIL video summary, and the query of similar videos.

Service 1: V-ToC Presentation

As we mentioned in the step 1 of Section 3.2.1, an XML V-ToC is generated for each input video. We can further present it in a web-based format using XSL as shown in Figure 3.5. This presentation is an image-based description for the video.

A user can know the contents for a list of video at once, by having a quick look on the V-ToCs. Then, he can easily select his interested pieces efficiently as they know the video contents. It saves a lot of time from downloading and browsing the video one by one.

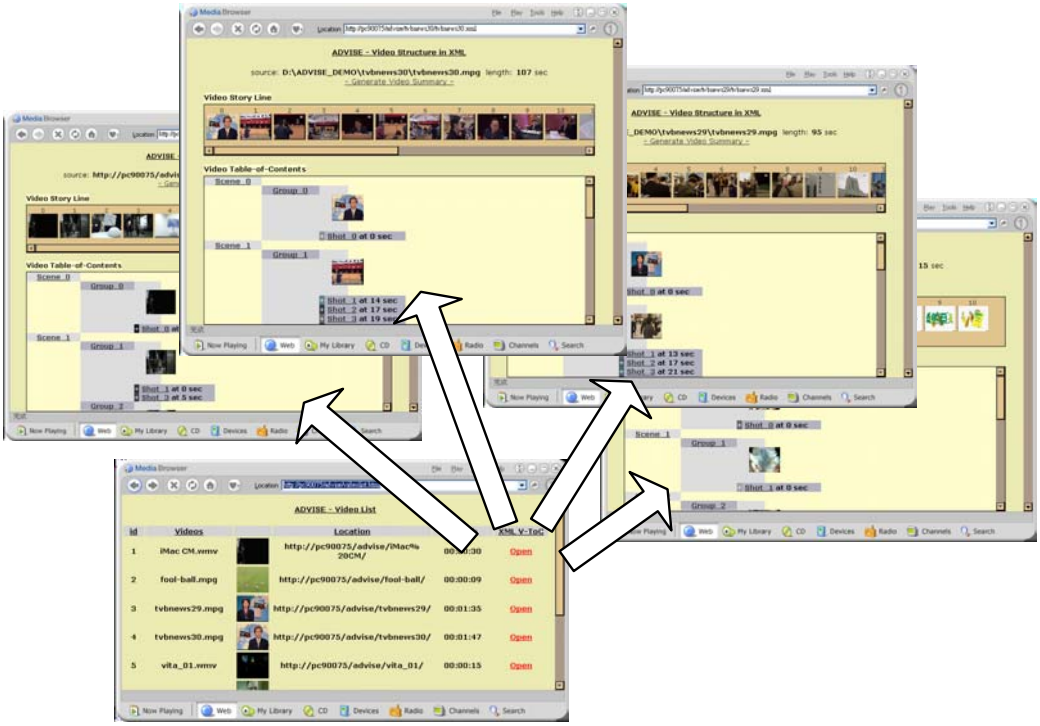


Figure 3.5 A List of V-ToCs

Service 2: Generation of SMIL Video Summary

Once a piece of video is selected according to the V-ToC, the user may want to extract only the important contents from the video for browsing. Then, ADVISE enables the user to generate a video summary, which is customized for his needs.

We employ a user input model to customize a video summary. A set of video features is defined, so that we calculate the score for each video segment according the ratings provided by the user. Then, we use a clustering method to adjust the video segments selection in order to generate a smooth summary of video. We present the video summary back to the user with a SMIL format, and thus a user can easily retrieve the summary through the Internet. The details of video summarization will be discussed in Chapter 5.

The user input panel to adjust the weights of features and other settings is shown in Figure 3.6. Once the summarization settings are received on the web server, a PHP script is invoked to interpret the request and generate the corresponding SMIL presentation.

We can further include some more details of each video segment by using the V-ToC. Since the XML and SMIL are both in plain text format, we can apply the PHP script, which consists of an XML parser, to transform the information in V-ToC structure into the SMIL presentation. Then the resulting SMIL is divided into three regions. In the first region, those selected video shots form a video sequence. The second region contains a text stream, which is aligned with the video sequence to show the scene, group and shot numbers of the playing video segment. In the third region, the corresponding key frame for the video segment is shown. The user is then able to play his customized SMIL video summary through the Internet with the RealOne player as shown in Figure 3.7.

Figure 3.6 User Input Panel for Video Summarization

Figure 3.7 Playing Video Summary

Service 3: Querying Similar Videos

A user can also query for similar videos in ADVISE. We use a PHP script to retrieve the stored video matching results described in the step 2 of Section 3.2.1.

We enable user to use the video name as the query. He can also select matching either the color histogram feature or the shot style feature. A number of similar videos are listed in descending order of the similarity score. We hide those results with similarity score lower than 30%. It is because those results are not likely to be similar to the querying video.

After searching the similar videos, a user can browse the V-ToC in order to see whether the resulting video is the one that he looks for. The web-based video matching interface is demonstrated in Figure 3.8.

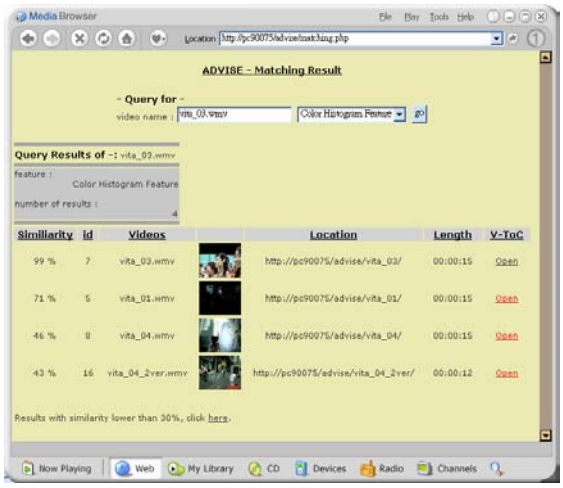


Figure 3.8 Interface for Video Matching

3.2.3 Video Streaming Server

The third module in ADVISE is the video streaming server. It is responsible for delivering videos on the Internet.

We use a Real System Server [33] as the streaming server in ADVISE. Figure 3.9 shows the interface for the server configurations. A streaming protocol, RTSP, is used in this server, such that videos can be sent in streams. Then, we can start playing the video immediately before the full video is downloaded. With the help of the streaming service, we can also select to play some specific video segments from a video, which resides on the server, using a SMIL presentation. This mechanism is used in delivering our SMIL video summary to the users.

As we have mentioned in Section 3.2.1, ADVISE accepts different video formats, some of them may not be compatible with RTSP. Therefore, we need to convert all the source videos into RealMedia such that they can be streamed using RTSP [33]. We use the RealProducer provided by the Real Networks [34] to convert the videos. The resulting RealMedia videos are then stored in the video database before they are retrieved by the users of ADVISE.

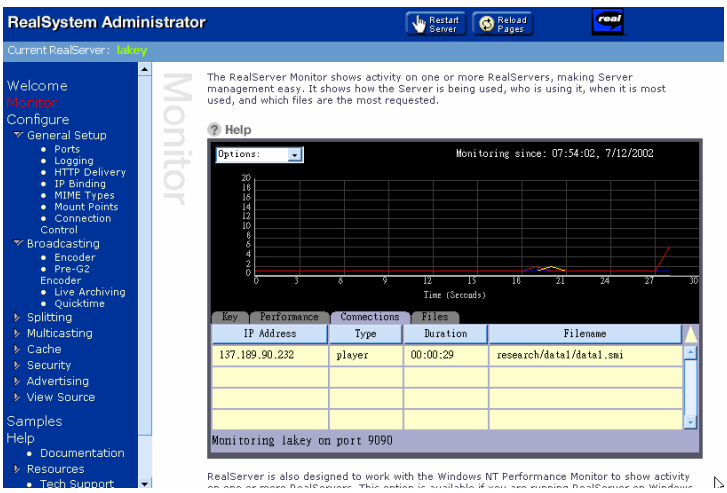


Figure 3.9 RealSystem Administator Page

3.3 Summary

We have described the overall architecture of the ADVISE system. There are three major modules in ADVISE. The first one is the video processing module, the second one is the web-based video retrieval module, and the third one is the video streaming server.

The web-based video retrieval module is, in fact, the interface for Internet users to access the ADVISE system. It provides three services to the users. They are the V-ToC presentation, the generation of SMIL video summary, and the query of similar videos.

We have briefly described how those services perform in Section 3.2.2. However the detail mechanism of these services will be discussed in later chapters. Therefore, we will discuss the construction of V-ToC in Chapter 4, the video summarization in Chapter 5, and the video matching in Chapter 6.

Chapter 4

Construction of Video Table-of-Contents (V-ToC)

In this chapter, we describe our approach on the construction of V-ToC, which is an image-based video description mentioned in Chapter 1. We organize the construction process into two major parts. The first one is about the video structuring and the second one is about the structure storage and presentation. Figure 4.1 stretches the workflow of the V-ToC construction.

In Section 4.1, we structure a video into a 4-level tree structure as described in Section 2.2. A color histogram approach with the additional regions setting [15] is used in the feature extraction step. Then, in the video shot boundaries detection and video groups formation, we employ an entropic threshold function [51], in order to make our approach adaptive. Next, we figure out the video scenes formation step, which concerns on the temporal factor from the video sequence. These are the steps to build up the V-ToC structure.

XML is used to store the V-ToC structure. In Section 4.2, we define a set of XML elements to document the structure in a nested hierarchy. Also, we define a DTD to maintain the consistency of the XML structure. To make a presentation of the XML structure, we use XSL to transform it into a web-based format. The web-based presentation is designed to provide users a concise and neat description of the video.

In the Section 4.3, we detail the experiment of the generated video structure and evaluate the results.

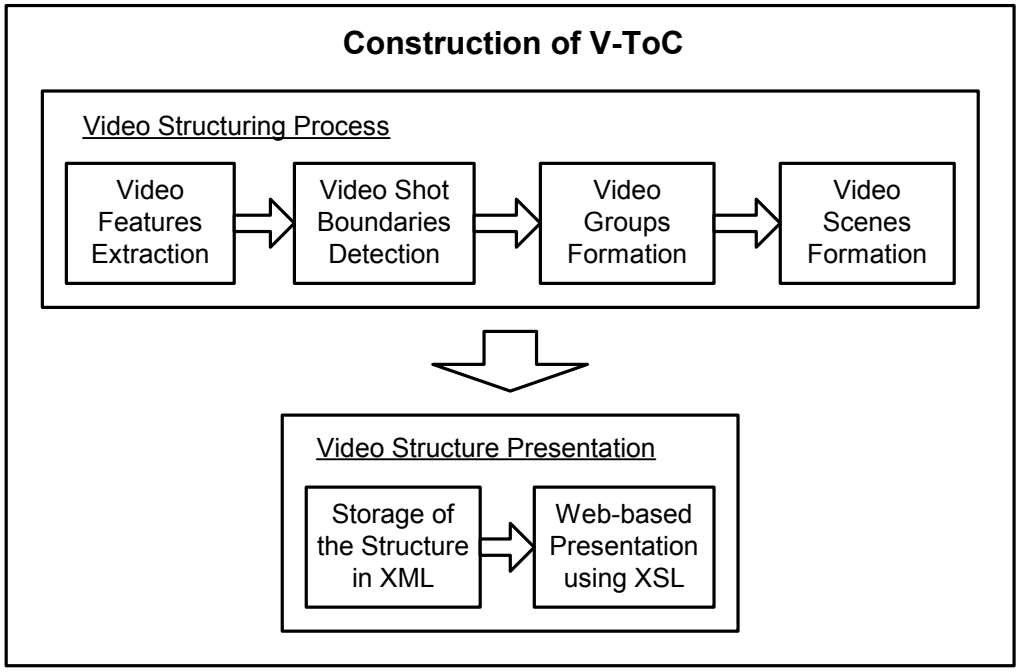


Figure 4.1 Workflow for V-ToC construction

4.1 Video Structuring

4.1.1 Terms and Definitions

Before we start to discuss the algorithm for video structuring, we define several key concepts as follows. We extend Rui's definition, which is introduced in Section 2.2, in a mathematical format.

Definition 4.1 *The structure of a **V-ToC** is defined as a 4-level tree structure, which is shown in Figure 2.2. It consists of the video level, the video scene level, the video group level, and the video shot level.*

Definition 4.2 *A **video** is defined as a sequence of frame images. It is represented by* $V = (f_1, f_2, \cdots, f_n)$, *where* f_i *is the* i-th *indexed frame of the video*

and n is the number of total frames in the video, V.

Definition 4.3 *A **video shot boundary** occurs when the frame-to-frame distance, $z(f_x, f_{x+1})$, between two consecutive video frames, f_x and f_{x+1}, is greater than a certain threshold, T. We use the frame index, x+1, to locate the video shot boundary.*

Definition 4.4 *To compare the similarity between two video frames, f_x and f_y, we define the **frame-to-frame distance** as $z(f_x, f_y)$. This distance is the difference of the quantified video features between two video frames. This is symmetrical between any two video frames.*

Definition 4.5 *At the video shot level, a **video shot**, s, is defined as a video sub-sequence, which has video shot boundaries occur at the beginning and the end only. A shot can be written as $s = (f_i, f_{i+1}, f_{i+2}, \cdots, f_{i+j})$, or in term of an ordered pair $s = [f_i, f_{i+j}]$, where $1 \le i \le (i+j) \le n$ and $i, j \ge 1$. We illustrate the properties of a video shot in Equation (4.1).*

$$\begin{gathered} \forall_{x,y \in (i,i+1,i+2,\cdots,i+j)} z(f_x, f_y) \le T \\ \text{and} \\ z(f_{i-1}, f_i) > T \text{ and } z(f_{i+j}, f_{i+j+1}) > T \end{gathered} \tag{4.1}$$

Definition 4.6 *We define a **key frame** be the first frame in the video shot. For the shot, s_i, we denote its corresponding key frame as k_i.*

Definition 4.7 *A **video group** is defined as a set of video shots. It can be written as $g = \{s_{a_1}, s_{a_2}, s_{a_3}, \cdots\}$, where $1 \le a_1, a_2, a_3, \cdots \le m$, and m is the number of video shots in the video. There are two properties for the member shots.*

i) The frame-to-frame distance between the key frames of any two member shots is not greater than the threshold, $T_{keyframe}$. It is formulated in Equation (4.2).

$$\forall_{i,j \in \{a_1, a_2, a_3, \ldots\}} z(k_i, k_j) \le T_{keyframe} \tag{4.2}$$

ii) The time separation between any two member shots on the video

sequence is not greater than the threshold, $T_{temporal}$, for this temporal factor. It is formulated in Equation (4.3).

$$\forall_{a_i \in \{a_2, a_3, \ldots\}} \text{ if } s_{a_i} = [f_{x_i}, f_{y_i}], s_{a_{i-1}} = [f_{x_{i-1}}, f_{y_{i-1}}] \quad x_i - y_{i-1} < T_{temporal} \tag{4.3}$$

Definition 4.8 *A **video scene** is defined as a set of video groups. It can be written as $c = \{g_{b_1}, g_{b_2}, g_{b_3}, \cdots\}$, where $1 \le b_1, b_2, b_3, \cdots \le p$, and p is the number of video groups in the video. Those video groups in a scene are mutually exclusive and they can be combined together to form a continuous video shot sequence, that is $c = (s_i, s_{i+1}, s_{i+2}, \cdots, s_{i+j})$, where j is the total number of members shots in all groups. Besides, no subset of a scene can form another continuous shot sequence.*

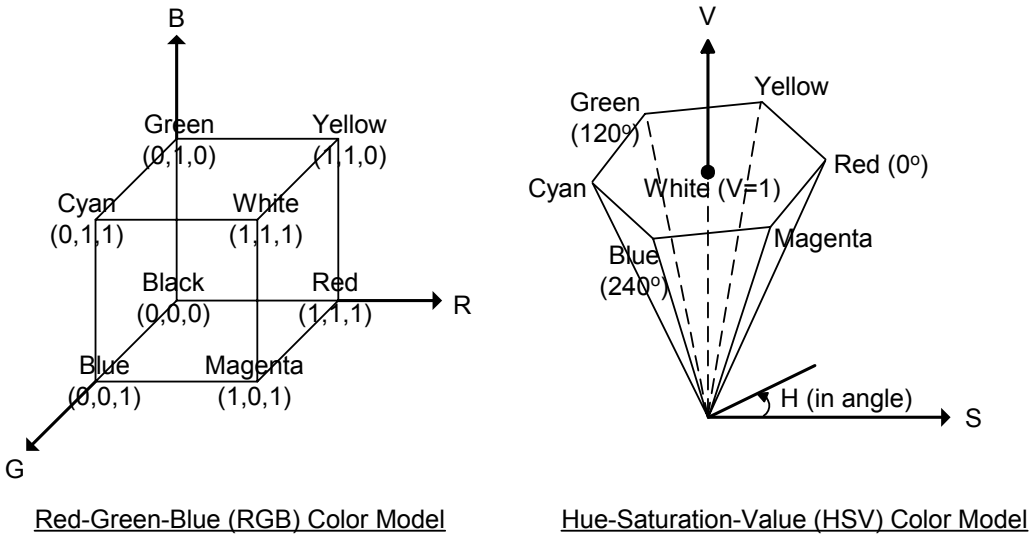
By using the above definitions, we detail the formation of V-ToC structure in the following sections. The first step is the extraction of regional color histograms as the video feature. Based on the extracted color feature, we carry out video shot boundary detection algorithm in the second step. Then, the third and forth steps are formations of video groups and video scenes respectively.

4.1.2 Regional Color Histograms

There are two major settings for our color histograms. The first one is the use of Hue-Saturation-Value (HSV) color model, and the second one the division of video frames into several regions.

Instead of a RGB color model, we use a HSV model for our color histograms. These two color models can be graphically presented as shown in Figure 4.2. A HSV model is more suitable for building color histograms of video frames [15][35]. It is because a RGB model is always too sensitive to the color changes due to light intensity, while in a HSV model, we can easily avoid this problem by detecting the color value which varies along the V-axis. We find that it is rather essential in shot boundary detection, since there are always instabilities of lighting effect in videos, for example, flash light, and sunshine.

In our setting, we use a 64-bin color histogram. It consists of a 2 dimensional color space for H and S values. The H values are divided into 16 intervals and the S values are divided into 4 intervals. It results into 64 different ranges of color value. If we further divide the color space into more bins, for example 256 or 512 bins, it will be too sensitive to color changes. Therefore 64 color bins is an appropriate configuration.



Red-Green-Blue (RGB) Color Model Hue-Saturation-Value (HSV) Color Model

Figure 4.2 RGB and HSV Color Models

Since the traditional color histogram method can only detect the global color changes between video frames, we improve it by using regional color histograms. By using histograms for regions, we are able to catch a more localized color distribution in a video frame. As a result, we can avoid missing the boundary between two frames having different objects but with the same combinations on them. Besides, we can see that the major object is always shot at the center region of the camera, while the regions around are the background. When there is a rapid change in background color between two consecutive video frames, it causes great difference in color histograms of those frames. Then, a shot boundary detection algorithm will wrongly expect that a shot boundary occurs between the two frames. If we divide the video frame into several weighted regions, we can avoid the false detection by focusing mainly on the most important part located at the center.

Though there are benefits to localize the color histograms, we cannot create too many regions on the video frames. It is because a huge computation time is needed to calculate the difference between frames with many regional color histograms. It demands large memory storage for all the histograms. Other than these problems, we find that adding regions increases the sensitivity to objects across regions. That means objects shifting from region to region can causes great differences in regional color histograms, and it leads to the false detection of shot boundary. Figure 4.3 demonstrates those problems for a region color histogram to tackle.

We also summarize the pros and cons of the above settings in Table 4.1.

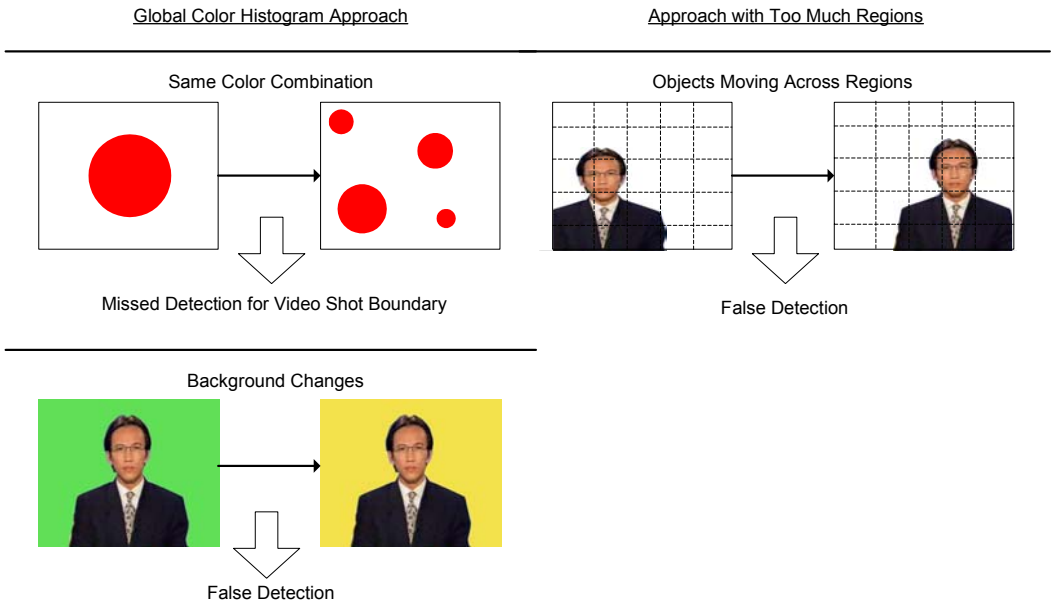


Figure 4.3 Problems for Global Approaches and Regional Approaches

In our configuration, we divide a video frame to five regions as shown in Figure 4.4. They include a rectangular region at the center and 4 corner regions. We also build the color histogram for the whole video frame, which we number it as region *6*. It is because we want to avoid the false detection caused by objects moving across regions. The color histogram for the whole video frame is the global histogram, which help us to keep a balance between a global approach and a regional approach as we mentioned above.

Table 4.1 Summary of Color Histogram Settings

	Pros	Cons
Use of HSV Model	• Can detect color changes caused by lighting effects	• Require additional transformation
Use of Regional Histograms	• Can detect certain local color changes • Can be set insensitive to background changes	• Increase the computations and memory usage • Sensitive to objects across regions

As a result, we collect six histograms for each video frame. When we calculate the total difference between two video frames, a weight can be applied to each regional color histogram. We expect that the global histogram should be the most weighted, the one for center region should be the second important, and the corner regions are the least important. Our current setting is listed in Table 4.2.

Table 4.2 Our Current Setting for Weight of Regional Color Histograms

Region	Weight
1. Center	0.2
2. Left Upper	0.1
3. Left Lower	0.1
4. Right Upper	0.1
5. Right Lower	0.1
6. Global	0.4

In order to calculate the frame-to-frame distance defined in **Definition 4.4**, we now use the color histograms as the quantified video feature. For two video frames, f_x and f_y, we calculate the frame-to-frame distance, $z(f_x,f_y)$, using a simple absolute distance of their color histograms as in Equation (4.4).

$$z_i(f_x,f_y) = \sum_{j=1}^{N} \left| Hist_{x,i}(j) - Hist_{y,i}(j) \right|$$
$$z(f_x,f_y) = \sum_{\forall \text{ histograms}, i} \left(z_i(f_x,f_y) \times w_i \right) \quad (4.4)$$

$Hist_{x,i}(j)$ denotes the j-th color bin in the histogram for region i in frame f_x. $z_i(f_x,f_y)$ is the difference in region i between two frames. w_i is the weight of a region.

By using Equation (4.4), we can find out all the frame-to-frame distances, $z(f_x,f_{x+1})$, between any two consecutive video frames, f_x and f_{x+1}. According to **Definition 4.3**, these results are used to determine the occurrences of video shot boundaries.

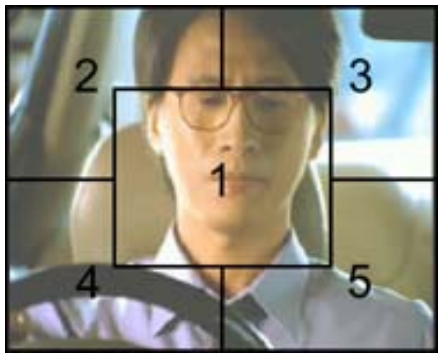


Figure 4.4 Five Regions in a Video Frame

4.1.3 Video Shot Boundaries Detection

Once we gather all the frame-to-frame distances between consecutive video frames, we need a threshold, T, to determine whether a video shot boundary occurs as defined in **Definition 4.3**. Since the choice of threshold can greatly affect the result of video shot boundaries detection, we need to adjust the value using adaptive approach according to each video. We employ the algorithm suggested by Yu [51], which uses entropies to adjust the threshold. It is an efficient algorithm to calculate an adaptive threshold.

The first step is building the statistics for the calculated frame-to-frame distances. Let the largest frame-to-frame distance, $\max(z(f_x,f_{x+1}))$, among n video frames to be z_{max}. We divide the range of values from 0 to z_{max} into q intervals. The size of each interval is z_{max} divided by q. Then for each interval, we can count the number of the frame-to-frame distances which values fall into the range. The counted number, $count_i$, for an interval, i, is calculate as in Equation (4.5).

$$count_i = \sum_{x=1}^{n-1} \delta\left(\left\lceil \frac{z(f_x,f_{x+1})}{z_{\max}/q} \right\rceil - i \right),\ 1 \le i \le q$$
$$\text{where} \begin{cases} \delta(a-i)=1, & \text{when } a=i \\ \delta(a-i)=0, & \text{otherwise} \end{cases} \tag{4.5}$$

The second step is the calculation of probability distributions and entropies. If we pick a threshold value, t, in between 1 and q, we can divide those q intervals into two classes at t, one for the video shot boundaries and the other for the non-boundaries. These two classes have their corresponding probability distributions for the frame-to-frame distances. By assuming the threshold value to be t, the probabilities for the non-boundaries, $P_{ns}(i,t)$, and the boundaries, $P_s(i,t)$, at interval i are calculated using Equation (4.6). Also, the entropies for the non-boundaries, $H_{ns}(t)$ and boundaries, $H_s(t)$, and the sum of those entropies, $H(t)$, are then calculated according to Equation (4.7).

$$P_{ns}(i,t) = \frac{count_i}{\sum_{j=1}^{t} count_j},\ 1 \le i \le t$$
$$P_s(i,t) = \frac{count_j}{\sum_{j=t+1}^{q} count_j},\ t+1 \le i \le q \tag{4.6}$$

$$H_{ns}(t) = -\sum_{i=1}^{t} P_{ns}(i)\log P_{ns}(i)$$
$$H_s(t) = -\sum_{i=t+1}^{q} P_s(i)\log P_s(i) \tag{4.7}$$
$$H(t) = H_{ns}(t) + H_s(t)$$

After we have calculated all the entropies, *H(t)*, for each value of *t* in between 1 and *q*, we can find an optimal threshold, t_{opt}, at which *H(t)* is the largest. It is formulated in Equation (4.8). The optimal threshold, t_{opt}, is in fact the most informative point to divide the intervals into two classes. Figure 4.5 illustrates, in a graphical format, the approach to find an optimal threshold, t_{opt}.

$$H(t_{opt}) = \max_{t=1,2,\ldots,q} \{H(t)\} \tag{4.8}$$

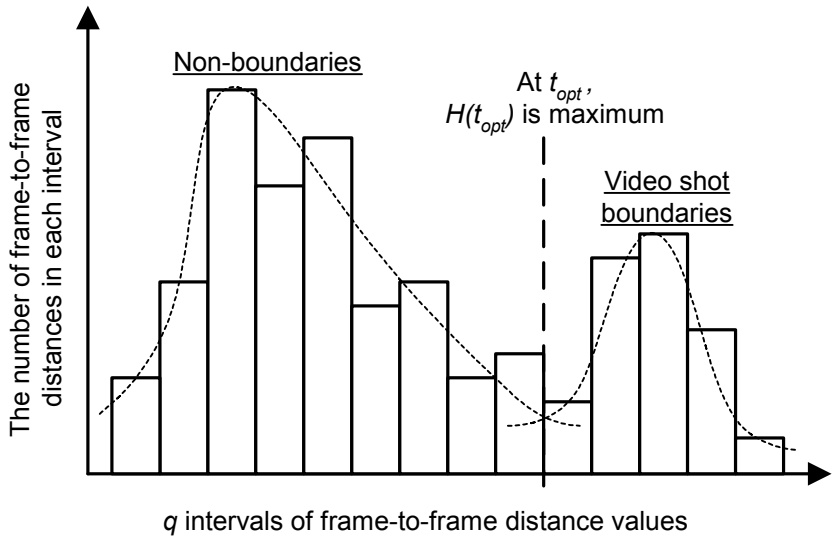


Figure 4.5 Finding the Optimal Threshold

Finally, we can calculate the required adaptive threshold, *T*, which equals t_{opt} times the size of an interval, as shown in Equation (4.9).

$$T = t_{opt} \times \frac{z_{\max}}{q} \tag{4.9}$$

We can then determine the video shot boundaries using the resulting adaptive threshold value as shown Equation (4.10).

$$z(f_x, f_{x+l}) \begin{cases} > T \Rightarrow \text{Shot boundary occurs} \\ \leq T \Rightarrow \text{Not a shot boundary} \end{cases} \tag{4.10}$$

After finding all the video shot boundaries, we can divide the whole video sequence into a number of video shots. We can use the calculated frame-to-frame distances and the adaptive threshold to formulate the video shots according to **Definition 4.5**. We also illustrate the shots formation in **Example 4.1**. We regard the set of resulting video shots as the video shot level of the V-ToC. Figure 4.6 shows the formation of the video shot level for V-ToC.

Example 4.1 *Given a video,* $V = (f_1, f_2, \cdots, f_{1500})$*, having 1500 frames. By using the above calculations, we found that the following frame-to-frame distances are greater than the adaptive threshold, T.*

$$z(f_{210}, f_{211}) > T,\ z(f_{677}, f_{678}) > T,\ z(f_{1295}, f_{1296}) > T,\ z(f_{1445}, f_{1446}) > T$$

As a result, we can conclude that the video shot boundaries are located right after frames f_{210}*,* f_{677}*,* f_{1295} *and* f_{1445}*. Then we divide the video sequence at those frames, such that the video shots,* s_1 *to* s_5*, are represented as follows.*

$$s_1 = [f_1, f_{210}], s_2 = [f_{211}, f_{677}], s_3 = [f_{678}, f_{1295}], s_4 = [f_{1296}, f_{1445}], s_5 = [f_{1446}, f_{1500}]$$

According to **Definition 4.6**, we take the first frame of each video shot to be the key frame of the shot. A key frame represents the video shot in the video groups and video scenes formation. We detail those formations in the following sections.

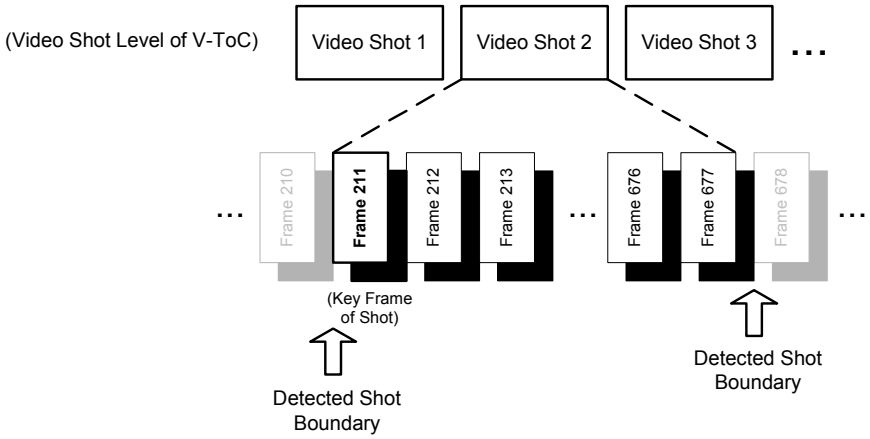


Figure 4.6 Formation of Video Shot Level for V-ToC

4.1.4 Video Groups Formation

After dividing a video into video shots, our next step is collecting similar shots into video groups. A video group is defined in **Definition 4.7**. There are two problems to be solved in the video groups' formation. The first one is the finding the threshold, $T_{keyframe}$, to test the feature similarities between key frames of member shots. The second problem is finding the threshold, $T_{temporal}$, to control the time separation between member shots.

For the first problem, we can directly use the threshold, T, which obtained in the video shot boundaries detection, as $T_{keyframe}$. It is shown in Equation (4.11). Since T can determine similar video frames in a sequence, we use the same value for $T_{keyframe}$, such that we can find similar key frames from different video shots. There are other methods to find the inter-shot dissimilarity in [15], we can use them to find out a more accurate value for $T_{keyframe}$. However, it is more convenient to use the value of T because most of those methods involved a lot of computations, which may not really worth doing.

$$T_{keyframe} = T. \tag{4.11}$$

For the second problem, we are going to find the value of $T_{temporal}$, which can avoid grouping shots separated far apart. Rui suggests that video shots separated far apart are not likely to be related in the video contents [35]. Figure 4.7 is an example for this case. Therefore we need to choose the value of $T_{temporal}$ carefully.

Assume there are m video shots in the video, and then $T_{temporal}$ is calculated by the average length of video shots, s_i, times a predefined constant factor, K. Equation (4.12) shows the calculation of $T_{temporal}$. According to the result of Rui, a video shot, which is ten times the average shot length apart from other video shot, is probably not related [35]. As a result, in our current setting, K is equal to 10.

$$T_{temporal} = \frac{\sum_{i=1}^{m} length(s_i)}{m} \times K$$
$$\text{where} \begin{cases} s_i = [f_x, f_y] \\ length(s_i) = y - x \end{cases} \tag{4.12}$$

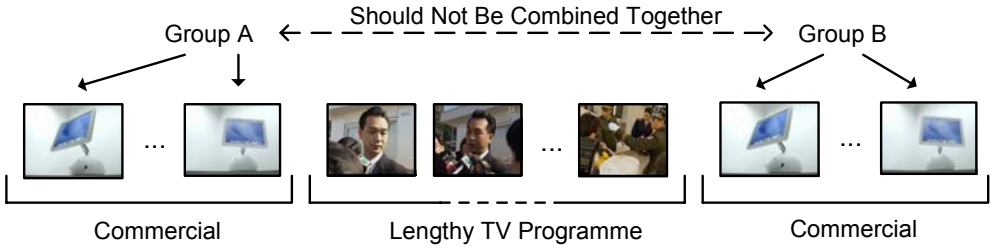


Figure 4.7 Temporal Factor for Video Groups Formation

Once we have set the thresholds, we can use an efficient algorithm to group the video shots.

Since it is not efficient to ensure the first property of a video group in **Definition 4.7**, we modify this property in order to fit into our algorithm. Instead of finding the frame-to-frame distances between all key frames in the video group, we calculate only two of them. The first one is the distance between the key frame, k_x, of a video shot, s_x, and the key frame, k_y, of the first member shot, s_y, in the group. The second one is the distance between k_x and the key frame, k_z, of the last member shot, s_z, in the group. Finally, we take the average of the two distances to be the shot-to-group distance, $z_g()$. We formulate these calculations in Equation (4.13). In our algorithm, we assign a video shot to the video group by comparing the shot-to-group distance with the threshold, $T_{keyframe}$.

$$\text{Given a video shot, } s_x, \text{ and a video group, } g_i = (s_y, s_w, \cdots, s_z):$$
$$\text{shot-to-group distance, } z_g(s_x, g_i) = \frac{z(k_x, k_y) + z(k_x, k_z)}{2} \tag{4.13}$$

The video groups formation algorithm is summarized in Algorithm 4.1. Each video shot from the input shot sequence is examined in turn. By finding the smallest shot-to-group distance, we can take an existing group which is the most similar to the current shot. In order to fulfill the two properties of a video group, we assign the current shot to the group when (i) the shot-to-group distance is smaller than $T_{keyframe}$, and (ii) the time separation between the current shot and the last member of the group is smaller than $T_{temporal}$. Otherwise, we need to create a new video group for the current shot.

Input: A sequence of m video shots, $(s_1, s_2, \cdots, s_m)$.

Output: A set of p video groups, $G = (g_1, g_2, \cdots, g_p)$, where g_i is the i-th video group.

1. Add a group g_1 to G, and assign s_1 to g_1.
2. **for each** s_x in $(s_2, s_3, \cdots, s_m)$
3. **for each** g_i in G,
4. Calculate the shot-to-group distance, $z_g(s_x, g_i)$.
5. **end for**
6. Find $z_g(s_x, g_a) = \min_{\forall g_i}(z_g(s_x, g_i))$, among all the existing groups.
7. Test:
   i) $z_g(s_x, g_a) \le T_{keyframe}$
   ii) $w - v \le T_{temporal}$, where w is the index of first frame in s_x and v is the index of last frame in the last member shot of g_a.
8. **if** i) and ii),
9. **then** assign s_x to g_a.
10. **else** add a group g_b to G, and assign s_x to g_b.
11. **end if**
12. **end for**

Algorithm 4.1 Video Groups Formation

By using the above algorithm, we can formulate a set of video groups, which collect similar video shots. We can then know the number of times that a video shot repeats and the number of different shot groups appeared on the video.

Figure 4.8 demonstrate the formation the video group level in V-ToC using an example.

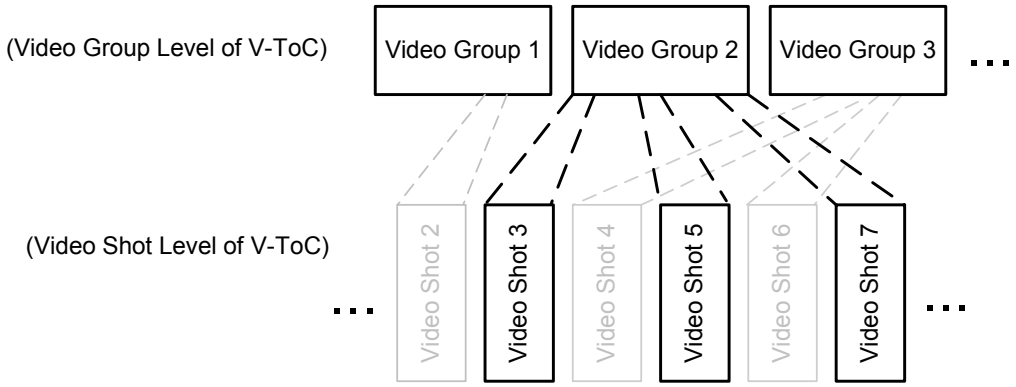


Figure 4.8 Formation of Video Group Level of V-ToC

4.1.5 Video Scenes Formation

Based on the video groups prepared, we can construct video scenes in this step. According to **Definition 4.8**, we combine a set of adjacent video groups into a continuous video shot sequence. Since member shots of those video groups are appearing in turns, it is reasonable that they are quite related in video contents [29][35]. As a result, we can combine them into a scene which extracts a complete segment of the video story. **Example 4.2** is a common example to illustrate this idea.

Example 4.2 *In an interview video as shown in Figure 4.9, there are shots taken on the interviewer and interviewee at different view angles. Then video groups, which contain those different kinds of shots, always appear across each other throughout the interview. Therefore, we can expect that if we combine those concurrent video groups to a video scene, the resulting segment of video will have its own substantive content.*

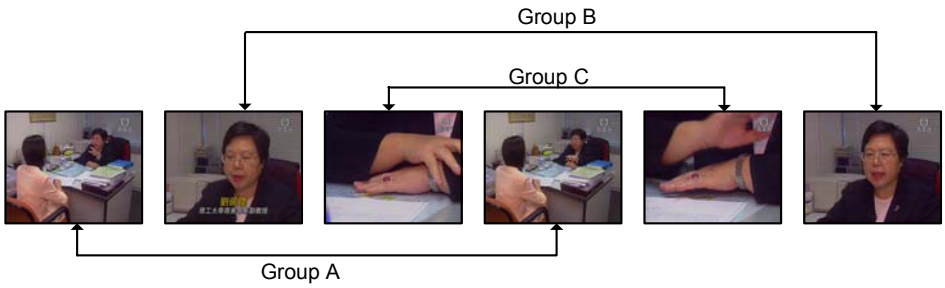


Figure 4.9 An Interview Video

There are two steps to form video scenes. First we need to sort video groups according to the temporal order of their first member, which is the most preceding video shot in the group. Second, we compare the time slots for the first and the last member in the group with the time slot for each scene.

After the groups are sorted, there are only three cases we need to handle in the second step. (1) We assign a group to a scene if it is overlapped with the time interval of the scene. (2) If a group includes the first member within the scene time and the last member outside the scene time, we also assign the group to the scene. However, in this case, we need to expand the time interval of the scene to cover this group. (3) If a group is not overlapped with any scene, we create a new scene for the group. The examples for these three cases are shown in Figure 4.10. All video scenes are formed after every video group is examined in turns.

We summarize this algorithm in Algorithm 4.2.

```
Input:  A sequence of p video groups, G = (g_1, g_2, ..., g_p), where g_i's are
        ordered by the temporal sequence of their first member shots.
Output: A set of r video scenes, C = (c_1, c_2, ..., c_r), where c_j is the j-th
        video scene.
1.  Add a scene c_1 to C, and assign g_1 to c_1, such that c_1 can also
    represented by a sequence of member shots of g_1.
2.  Set the number of scenes, r = 1
3.  for each g_i = (s_y, ..., s_z) in G
4.    if w < y < x < z, //case(2)
5.      then
6.        i)    assign g_i to c_r = (s_w, ..., s_x)
7.        ii)   update c_r to (s_w, ..., s_z).
8.      else
9.        if w < y < x < z, //case(3)
10.         then
11.           i)    add a new scene c_{r+1} to C, and assign g_1 to c_{r+1}.
12.           ii)   c_{r+1} can be represented by (s_y, ..., s_z).
13.           iii)  r = r + 1
14.         else //case(1): w < y < x < z
15.           i)    assign g_i to c_r = (s_w, ..., s_x)
16.     end if
17.   end if
18. end for
```

Algorithm 4.2 Video Scenes Formation

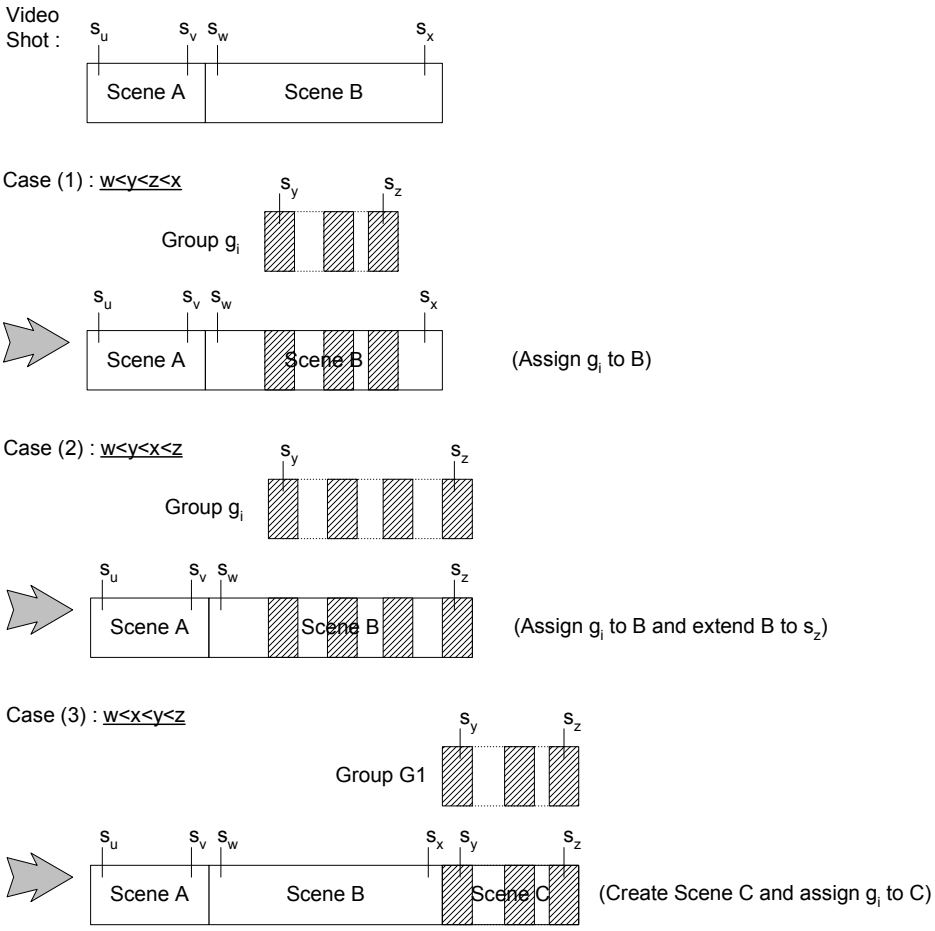


Figure 4.10 Different Cases of Video Scene Formation

After the video scenes formation, we need to build up the tree structure as defined in **Definition 4.2**. We have extracted all the required components under the video scene level for V-ToC. According to those sets of video scenes, video groups and video shots, we can organize the video components into nodes of the tree structure as shown in Figure 4.11. Finally, at the video level, there is only one single node, which collects all video scenes as child nodes, represents the whole video.

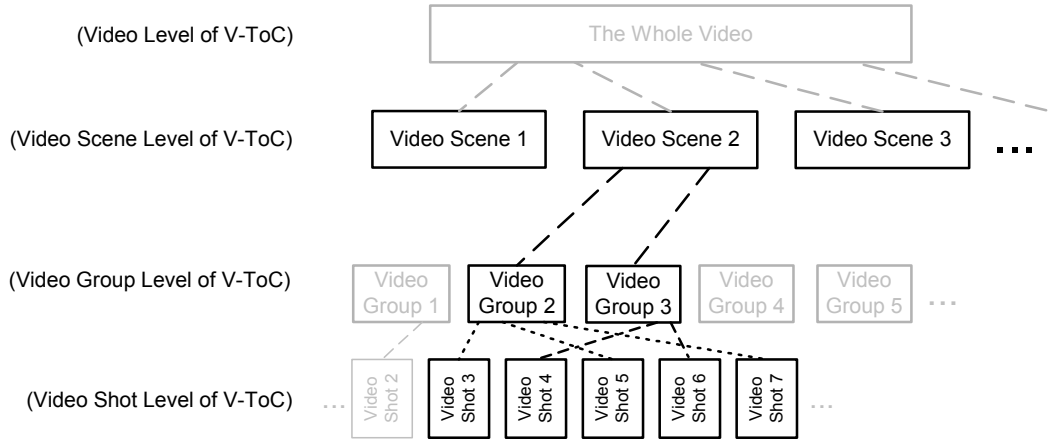


Figure 4.11 Formation of the Whole V-ToC Structure

4.2 Storage and Presentation

Once the V-ToC structure is built, we store it in an XML format. We use XML for V-ToC storage because of four major benefits. First, we can build up an organized and compact data structure for using the nested hierarchy of XML [46]. It will be efficient to identify each video component by the corresponding XML element defined. Second, with the plain-text property of XML, we are able to modify any items, reorganize the structure, or query the stored information comfortably. Third, by the extensibility of XML, we can be flexible to include additional information in the video structure. Defining a new set of elements can extend the video tree structure, and carry other video features, including caption text from video caption extraction, transcript from speech recognition, or the presence of face detection. For the fourth major benefit, due to the growing importance of XML as a standard data exchange protocol on the Internet, we can widely spread the V-ToC structure to other multimedia applications with the XML format.

We design an XML data structure in the following to store the V-ToC. A DTD is defined to maintain the consistency of the V-ToC data structure. Besides, we detail the presentation of the V-ToC structure on a web-based format using XSL transformation.

4.2.1 Definition of XML Video Structure

We define seven XML elements to store the 4-level tree structure of V-ToC. Those elements are in the following.

- **<advise>** is used to encapsulate all elements for V-ToC structure, such that it is convenient for exchanging data structure between our XML applications.
- **<video>** is the root level component of the V-ToC structure. It contains multiple scenes. It has two attributes, **length** and **src**. **length** states the video duration and **src** point to the file location of the video source.
- **<scene>** represents a video scene component in the V-ToC structure. It contains video groups. An attribute, **id**, is associated with **<scene>**, and represents the scene number.
- **<group>** is a video group in the V-ToC structure. It consists of multiple video shots. Similar to scene, it contains an attribute, **id**, which is the group number.
- **<shot>** is a video shot in the V-ToC structure. It also has an attribute, **id**, to represent the shot number. It can carry different video information, including the time and the key frame.
- **<keyframe>** is an element to store the key frame for the corresponding video shot. The attribute, **img**, points to the location of the stored key frame image. The attribute, **id**, is used to represent the video frame index.
- **<time>** contains an attribute, **value**, which is used to record the beginning time, in seconds, of a shot in the video sequence. This value is a positive integer. For example, a value, 11, means that the shot starts at 11 seconds in the video sequence. The time value is important because most video browsing applications use time to locate a video shot instead of video frame index. The value is calculated by Equation (4.14).

$$value = \text{video frame index} \times \text{video frame rate} \tag{4.14}$$

We use these elements to construct a nested XML hierarchy for the V-ToC structure. We use the elements **<video>**, **<scene>**, **<group>** and **<shot>** to represent the four level of video components, and organize into a tree-like structure as shown in Figure 2.2. Also, we associate each **<shot>** with the **<keyframe>** and **<time>** elements, so that more video features and other information about a video shot can be included in the XML V-ToC. The associations between the above elements are summarized in Table 4.3.

Table 4.3 Associations between XML elements for V-ToC

XML Elements	Child Nodes	Attributes
<advise> … </advise>	video	-
<video> … </video>	scene	length, src
<scene> … </scene>	group	id
<group> … </group>	shot	id
<shot> … </shot>	time, keyframe	id
<keyframe/>	-	img, id
<time/>	-	value

In order to maintain the consistency, we define a DTD [49] for the V-ToC structure. We define the DTD with all those elements, their relations and associated attributes. The DTD we employed is defined in Figure 4.12. An example XML V-ToC structure according to the defined DTD is shown in Figure 4.13.

4.2.2 V-ToC Presentation Using XSL

In this section, we transform the XML into a web-based presentation by using the XSL transformation [10][47]. Although the data structure of the V-ToC is well-defined, it does not place any limitations to the design of the web-based interface to present the V-ToC. Therefore, apart from simply showing the V-ToC structure, we aim at providing users for the best way to understand the video contents at once.

```
<?xml version="1.0"?>
<!ELEMENT advise (video+)>
<!ELEMENT video (scene+)>
  <!ATTLIST video length CDATA #REQUIRED>
  <!ATTLIST video src CDATA #REQUIRED>
<!ELEMENT scene (group+)>
  <!ATTLIST scene id CDATA #REQUIRED>
<!ELEMENT group (shot+)>
  <!ATTLIST group id CDATA #REQUIRED>
<!ELEMENT shot (keyframe+, time+)>
  <!ATTLIST shot id CDATA #REQUIRED>
<!ELEMENT keyframe EMPTY>
  <!ATTLIST keyframe img CDATA #REQUIRED>
  <!ATTLIST keyframe id CDATA #REQUIRED>
<!ELEMENT time EMPTY>
  <!ATTLIST time value CDATA #REQUIRED>
```

Figure 4.12 DTD for XML V-ToC Structure

```
<?xml version="1.0"?>
<!DOCTYPE advise SYSTEM "./toc.dtd">
<advise>
<video length="25" src="rstp://localhost/video1.rm">
<scene id="1">
  <group id="1">
    <shot id="1">
      <keyframe img="./sh_1.jpg" id="1"/>
      <time value="0"/>
    </shot>
    <shot id="3">
      <keyframe img="./sh_3.jpg" id="359"/>
      <time value="11"/>
    </shot>
  </group>
  <group id="2">
    <shot id="2">
      <keyframe img="./sh_2.jpg" id="217"/>
      <time value="7"/>
    </shot>
  </group>
</scene>
<scene id="2">
  <group id="3">
    <shot id="4">
      <keyframe img="./sh_4.jpg" id="611"/>
      <time value="20"/>
    </shot>
  </group>
</scene>
</video>
</advise>
```

Figure 4.13 XML V-ToC Structure

We make use of the XSL filtering and sorting techniques alternatively to extract the required XML data from the V-ToC. A simplified segment of the XSL is quoted in Figure 4.14. In this XSL segment, we order the video shot components according to the attribute **id** in each **shot**. Then, in an HTML table row, we print out the **scene id**, the **group id**, and the **shot id**. After that, we show the key frame image using an HTML image tag with the source location stored at the **keyframe img** attribute. Besides, we print the corresponding time instance recorded at the **value** attribute of the element **time**.

```
<xsl:for-each select="advise/video/scene/group/shot"
order-by="../@id">
<tr class="nfont">
  <th><xsl:value-of select="../../@id"/></th>
  <th><xsl:value-of select="../@id"/></th>
  <th><xsl:value-of select="@id"/></th>
  <th align="left">
    <img width="55" height="45">
      <xsl:attribute name="src">
          <xsl:value-of select="keyframe/@img"/>
      </xsl:attribute>
    </img> at <xsl:value-of select="time/@value"/> sec
  </th>
</tr>
</xsl:for-each>
```

Figure 4.14 XSL Segment for Transforming XML V-ToC Structure

A sample web-based presentation of the V-ToC structure is shown in Figure 4.15. There are four major features in our design.

The first one is the clear display of the basic video information. It consists of the video source location and the length of video duration. A user can easily identify which video that the V-ToC is describing.

Secondly, we provide a sequential video story line, which is actually the set of video shots sorted according to the temporal ordering. We find that user can follow the video story along the shots sequence.

Third, we allow user to fold or unfold any video component inside the V-ToC region. We put our video components on HTML layers and use JavaScript to enable the control of the showing and hiding functions. By folding and

unfolding video components, a user can have a better understanding of the contents organization. They can also customize the outlook of the V-ToC structure, such that key contents are shown while those less important sections are hidden.

The last major feature is the enlargement of key frame images. Since it allocates a very large area if we use images of their original sizes, we need to minimize the size of those images on the V-ToC. However, the smaller images may not be enough to provide a clear view for a user to see the detail of the contents. Therefore, we enable a user to enlarge each key frame image by a simple mouse click, when he wants to view it in detail. We use JavaScript to catch to mouse click and resize the selected image on the display. We also recover the resized image, such that it would never stay allocating the space on the page.

With the design of these features, we make the web-based presentation of the V-ToC concise and convenient to users. They can easily understand the contents of the video in a short time using this image-based video description, V-ToC.

4.3 Evaluation of Video Structure

In this section, we evaluate the V-ToC structure generated. Since a more accurate video structure can better describe the video content to the users, an experiment is carried out to evaluate the accuracy of the video segmentation process in our system. We have taken four commercials as a simple test set for our system. The number of shots, groups and scenes in the set of videos are first examined by human. We then use these results to compare different segmentation approaches. We then perform video segmentation on the experimental video set with four segmentation approaches. The results are compared with the human judgments and shown in Table 4.4.

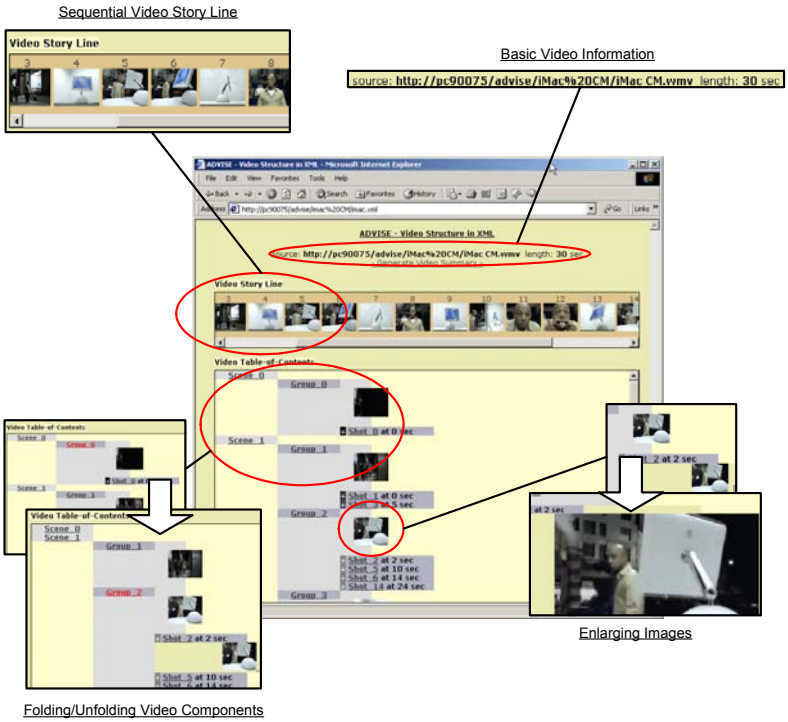


Figure 4.15 Web-based Presentation of V-ToC using XSL

In the first approach, video segmentation is based on single color histogram on each frame and a fixed threshold to determine the video shot boundaries. This is the most basic approach, which detects only the global color changes between video frames with one color histogram for each of them. It is convenient to set a common threshold value for video shots detection. However, through this approach, we find that the results are not quite accurate because there are always misdetections of video shots. Especially for video 2, the number of detected video shots is 2 times of the result judged by human. However, for video 3, the number of over detected video shots is much smaller. That means, the predefined threshold is suitable for certain videos but it causes a great error rate for other videos.

In the second approach, we try to improve the first approach by using an adaptive threshold for video shots detection. As we described in Section 4.1.3, the adaptive threshold is calculated using entropies. The results of this approach are more accurate then the first approach. It is because most of the misdetections are removed by using an appropriate threshold value for each video. The over

detection problem is reduced greatly, particularly for video 2 and video 4. It is because the adaptive threshold is effectively adjusted according to each video such that it can classify the video shot boundaries and non-boundaries.

Both the first and the second approach use single color histogram for each video frame. In the third approach, weight regional color histograms are used stead. This approach attempts to catch the local color differences between video frames. The segmentation result of this approach is similar to that of the first approach. However, we find that this approach can refine the segmentation results by overcoming the deficiency of applying global color differences. There are decreases in number of detected video shots for video 1 and 2 while increases for video 3 and 4. It shows that this approach can overcome the over detection problem for video 1 and 2 and the misdetection problem for video 3 and 4. We have illustrated these problems for using global color histogram in Figure 4.3

The fourth approach is the implementation of our system. We employ the weighted regional color histograms and the adaptive threshold in our system. Comparing with the previous approaches, we find that the results of our system are the closest to the human judgments. In fact, the video segmentation process of our system is improved over the other approaches and it is the most accurate one among them. As this approach combines both the second approach and the third approach, it takes those advantages from regional color histograms and the adaptive threshold. Although the result for detecting video shots is just similar to that of the second approach, the video groups and video scenes formation are improved with the help of regional color histograms.

Finally, we compare our results with the segmentation of the INFORMEDIA project [9][15][44][45]. We found that INFORMEDIA give better results in video shot detection. Therefore, we should further improve the video shot detection in ADVISE. Although the INFORMEDIA is superior to ADVISE in video shot detection, it does not give any further information about the organization of those shots. So, we can also conclude that using the shot-based structure only in INFORMEDIA is the deficiency compared with ADVISE.

Table 4.4 Comparing Video Segmentation Results with the Human Judgments

	Video	Frames	Shots	Group	Scene
Human Judgments	Video 1	874	12	4	3
	Video 2	1571	15	8	2
	Video 3	901	18	5	2
	Video 4	894	14	5	2
First Approach - using single color histogram - with fixed threshold	Video 1	874	19	7	5
	Video 2	1571	29	12	3
	Video 3	901	21	8	4
	Video 4	894	25	10	5
	Accuracy		0.594	0.591	0.542
Second Approach - using single color histogram - with adaptive threshold	Video 1	874	16	6	4
	Video 2	1571	16	9	3
	Video 3	901	21	8	3
	Video 4	894	13	6	3
	Accuracy		0.858	0.753	0.688
Third Approach - using weighted regional color histograms - with fixed threshold	Video 1	874	18	6	4
	Video 2	1571	19	11	3
	Video 3	901	25	10	3
	Video 4	894	27	12	5
	Accuracy		0.634	0.578	0.621
ADVISE - using weighted regional color histograms - with adaptive threshold	Video 1	874	16	5	4
	Video 2	1571	16	8	3
	Video 3	901	19	7	2
	Video 4	894	17	7	3
	Accuracy		0.812	0.807	0.771
INFORMEDIA - using a shot-based structure only	Video 1	874	12	-	-
	Video 2	1571	15	-	-
	Video 3	901	17	-	-
	Video 4	894	14	-	-
	Accuracy		0.985	-	-

Chapter 5

Video Summarization

In this chapter, we present an algorithm to automate the video summarization. Based on the V-ToC tree structure generated in Chapter 4, we can retrieve video features and then select video segments into a video summary.

We expect the video summary can provide users more video information than the V-ToC we discussed in Chapter 4. Since the V-ToC describes the video contents using the video key frame images, a user is still not able to know exactly all the contents because a video delivers information also in form of audio and text apart from image. Therefore, a video summary, which is a shortened form of the source video, can give the user all types of information, and hence, he can know the video contents exactly.

There are two major objectives for a video summary. First, we want to browse only the major contents of the whole video from the summary. Second, we want to shorten the duration of the summary in order to browse it efficiently. According to these two objectives, our video summary is designed as follows.

A video summary is combined by a set of video segments, which contain the important video features of the source video. These important features are in fact the most valuable contents of the video. The summary with more important features is better in quality, as it collects the major video contents. However, our first problem is the different users' preferences about the importance of video features. We find that each user may have different opinions on whether a video feature is valuable in a video. As a result, the

quality of a video summary really depends on each user's preferences. Besides the quality of a video summary, the duration depends also on the need of each user. Since a longer video summary contains more video contents while a shorter one can be browsed efficiently, then a user needs to make a decision on either getting more information or spending less time on the summary. Therefore our second problem is to customize the video summary according the time constraint provided by the user.

We propose a statistical approach to select the contents for the video summary. In our system, we accept user's input about their preferences on the set of video features that we provided. We can then calculate a score for each video segment based on the user's preferences, such that if the score is high, the video segment contains more preferred video features; otherwise, it contains less preferred video features. Under a user defined time constraint, we can only select those segments with higher scores into our video summary, such that the summary contains more preferred video features. The generated video summary is therefore able to fit into user's appetite. Since there may be too many discontinuous and short segments selected into the video summary, it is difficult for a user to browse it comfortably. Hence, we refine the selection of segment with a clustering method, in order to reduce the discontinuity and make the video summary smoother for browsing.

We detail the video summarization algorithm in the following sections. In Section 5.1, we first define the key terms used in our video summarization algorithm. Then in Section 5.2, we describe those video features provided for user to select. In Section 5.3, we detail the video summarization algorithm based on the provided video features. Moreover, in Section 5.4, we describe the presentation of video summarization result using SMIL presentation. Finally in Section 5.5, we evaluate our video summarization algorithms using a set of experiments.

5.1 Terms and Definitions

In this section, we define a set of key terms for the video summarization algorithm as follows.

Definition 5.1 *An **extracted video segment**, e, is defined as a subsequence of the source video, $V = (f_1, f_2, \cdots, f_n)$. The segment is formulated in Equation (5.1).*

$$e = \begin{cases} (f_x, f_{x+1}, \cdots, f_y) \\ [f_x, f_y] \end{cases}, \quad 1 \le x \le y \le n \tag{5.1}$$

Definition 5.2 *The **score** of an extracted video segment, score(e), is defined as the summation of the weights, w_j, for the video features, $feature_j$, existing in the segment. It is shown in Equation (5.2).*

$$\begin{aligned} & score(e) = \sum_{\forall feature_j} \delta(feature_j) \times w_j \\ & \text{where} \begin{cases} \delta(feature_j) = 1, & \text{when } feature_j \text{ exists in } e \\ \delta(feature_j) = 0, & \text{otherwise} \end{cases} \end{aligned} \tag{5.2}$$

Definition 5.3 *A **video summary**, V', is defined as a subset of the source video, V. It consists of a set of extracted video segments, e_i. As shown in Equation (5.3) .*

$$\begin{gathered} V' \subseteq V \\ V' = \{e_1, e_2, \cdots, e_n\} \\ \text{where any } e_i \text{ , } i = 1, \cdots, n \end{gathered} \tag{5.3}$$

Definition 5.4 *The **duration of video summary**, d(V'), is defined as the summation of all the member video segment lengths. It is shown in Equation (5.4).*

$$d(V') = \sum_{i=1}^{n} length(e_i)$$
$$\text{where} \begin{cases} e_i = [f_x f_y] \\ length(e_i) = y - x \end{cases} \tag{5.4}$$

Definition 5.5 *The **time constraint**, T_{time}, of a video summary is defined as the maximum video summary duration in seconds. It should not be smaller than by the video summary duration dividing the video frame rate. We use the time constraint to limit the summary duration as shown in Equation (5.5).*

$$\frac{d(V')}{\text{video frame rate}} \le T_{time} \tag{5.5}$$

Definition 5.6 *The **score** for the video summary is defined as the mean score of the member video segments. We illustrate the calculation using Equation (5.6).*

$$score(V') = \frac{\sum_{i=1}^{n} \left(score(e_i) \times length(e_i) \right)}{d(V')} \tag{5.6}$$

5.2 Video Features Used for Summarization

Before we start our video summarization algorithm, we need to extract a set of features from the video in order to calculate the score for each video segment. Different video features, which appear on the video sequence, can be used in our video summarization algorithm. With more video features employed in our algorithm, a user can have a more flexible selection of his interested video segments. The resulting video summary can then be customized for the user more accurately. In our system, we employ five video features. They are: human face detection, male voice recognition, female voice recognition, volume level, and caption text detection. These features bring us most important

information about the video content. We describe the extractions of these video features in the following paragraphs.

Human face detection is used to locate the presences of human face in video segments. We use this video feature because human faces are always important to the video contents [44], especially for interview videos and news broadcasts. Human faces, which appear at the middle of the screen and occupy large areas, are more important to the video. In our algorithm, we detect the human face which appeared on the video manually. We locate the video segments with human faces, which cover at least half of the area for the center region (refer to the five regions defined in Section 4.1.2).

Apart from the human face, we extract the human voice feature in our summarization. We use the application created by a group of undergraduate student to classify human voice in video [14]. Since human voice gives a special waveform pattern and falls into certain frequency range, it can always be recognized from the audio channel of a video. We can then use a voice classification method, which considers parameters like voice level, pitch and frequency, to determine the male and female voice. By using this classification method, we can extract video segments with male voice or video segments with female voice.

The volume level measures the loudness of video segments in decibel, dB. This level is calculated by the amplitude of the sound wave in a video segment. A higher volume level means a loud sound, while a lower one means a quiet sound or even silence. As quiet sounds and silence are like to appear at the video shot boundaries, so they can almost give no information about the video contents. Thus, we can use the average volume level to cut off the video segments with quiet sounds or silence. In our algorithm, since normal human conversation ranged from about 40dB to 60dB, we extract video segments with volume level higher than 35dB.

Caption text on videos provides a lot of important information for the video contents. For example, the video script on screen shows the audio contents

directly. We can apply caption text localization methods to index the captions in the video sequence [23][36]. In our algorithm, we use the application created by our fellow student, who implemented the algorithm suggested by Sato et al. [36]. It extracts all the video segments with caption text on them.

We can extract the video segments for each video feature as shown in Figure 5.1. Based on those video features defined, we apply our video summarization algorithm to generate the video summary.

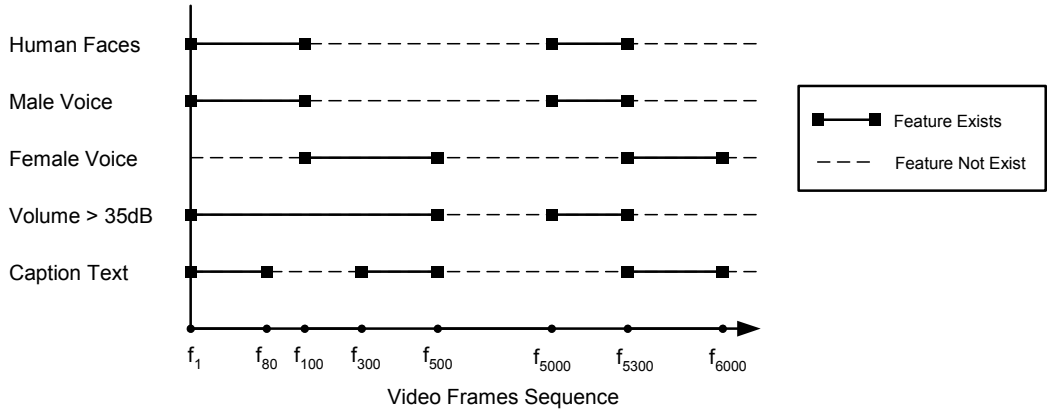


Figure 5.1 Video Features for Summarization

5.3 Video Summarization Algorithm

Since we notice that the quality of summarization depends on the interest of the target user, therefore, it is almost impossible to design a single approach, which can fit into every user's appetite. As a result, we need to accept inputs from the user, such that it is easier to determine which kinds of contents are more valuable to the user. Inputs from the user are summarized in Table 5.1.

Table 5.1 User Inputs for Video Summarization

Input	Variable name	Range of value
Weight for human faces	w_{face}	From 1 to 10
Weight for male voice	w_{male}	From 1 to 10
Weight for female voice	w_{female}	From 1 to 10
Weight for volume level	w_{volume}	From 1 to 10

Weight for caption text	$w_{caption}$	From 1 to 10
Time constraint for video summary	T_{time}	Any integer in sec.
Clustering control constant	K	Any integer

There are four steps to summarize a video. The first step is the reordering of the extracted video segments in order to avoid overlapping. The second step is calculation the score for each video segment. Then, the third step is the selection of video segments according to both the scores and the time constraint. Finally, the fourth step is the refinement of the selections using a clustering technique.

5.3.1 Combining Extracted Video Segments

In the first step, we generate a single set of non-overlapping video segments. Since we have five sets of extracted video segments, we cannot combine them together directly because of the partially overlapping regions as shown in Figure 5.1. Therefore, for each member segment, $[f_x, f_y]$, in all the five sets, we need to sort the frames, f_x and f_{y+1}, at the boundaries. We can then create a single set of video segments, e_i's, as shown in Equation (5.7). We also illustrate the formation of the new set of video segments in

Example 5.1.

$$\begin{gathered}\text{Given a set of sorted video frames } \{f_{a_1}, f_{a_2}, \cdots, f_{a_n}\}, \\ e_i = [f_{a_i}, f_{a_{i+1}-1}], \text{ where } 1 \le i \le n-1\end{gathered} \tag{5.7}$$

Example 5.1 *Given the sets of extracted video segments for different video features as shown in Figure 5.1.*

Human faces: $[f_1, f_{99}], [f_{5000}, f_{5299}]$
Male voice: $[f_1, f_{99}], [f_{5000}, f_{5299}]$
Female voice: $[f_{100}, f_{499}], [f_{5300}, f_{5999}]$
Volume > 35dB: $[f_1, f_{500}], [f_{5000}, f_{5299}]$
Caption text: $[f_1, f_{79}], [f_{300}, f_{499}], [f_{5300}, f_{5999}]$

We can sort the indexing frames and remove those duplicated, in order to get the

following set.

$$\{f_1, f_{80}, f_{100}, f_{300}, f_{500}, f_{5000}, f_{5300}, f_{6000}\}$$

The resulting set of extracted video segments is formulated as follows.

$$e_1 = [f_1, f_{79}], e_2 = [f_{80}, f_{99}], e_3 = [f_{100}, f_{299}], e_4 = [f_{300}, f_{499}],$$
$$e_5 = [f_{500}, f_{4999}], e_6 = [f_{5000}, f_{5299}], e_7 = [f_{5300}, f_{5999}]$$

5.3.2 Scoring the Extracted Video Segments

After formulating a set of extracted video segments, we calculate the score for each of the member segments in the second step. The calculation of the score for each segment is defined in **Definition 5.2**. We can illustrate the scores calculation in **Example 5.2**.

Example 5.2 *Given the combined set of extracted video segments,* $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$*, which are formulated in* ***Example 5.1****. Assume the user input weights are given, and then we can calculate the score for each segment,* e_i*, as follows. The scores can be represented in a graphical format as shown in Figure 5.2.*

Given $w_{face} = 10$, $w_{male} = 8$, $w_{female} = 2$, $w_{volume} = 3$, $w_{caption} = 1$,

$$\begin{aligned} score(e_1) &= \delta(feature_{face}) \times w_{face} + \delta(feature_{male}) \times w_{male} + \delta(feature_{female}) \times w_{female} \\ &\quad + \delta(feature_{volume}) \times w_{face} + \delta(feature_{caption}) \times w_{caption} \\ &= 1 \times 10 + 1 \times 8 + 0 \times 2 + 1 \times 3 + 1 \times 1 \\ &= \underline{\underline{22}} \end{aligned}$$

$$\text{and} \begin{cases} score(e_2) = 21 \\ score(e_3) = 5 \\ score(e_4) = 6 \\ score(e_5) = 0 \\ score(e_6) = 21 \\ score(e_7) = 3 \end{cases}$$

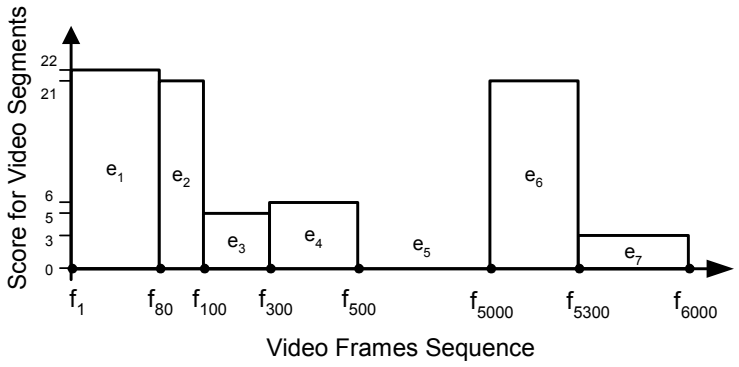


Figure 5.2 Scoring the Extracted Video Segments

5.3.3 Selecting Extracted Video Segments

In the third step, we select the extracted video segments into the summary according to the scores. Since there is time constraint, T_{time}, defined in **Definition 5.5**, only a limited number of segments are selected. We first sort the extracted video segments in descending order of the calculated score. Then, we pick up the segments from the highest score one into the video summary, *V'*. Until the duration of the summary, *d(V')* divided the video frame rate, reaches the time constraint, we stop picking more video segments. The selection process is summarized in Algorithm 5.1.

```
Input:  Given a set of extracted video segments, {e1,e2,⋯,en}, the
        scores for the segments, and the video frame rate.
Output: A set of segments selected into the video summary, V'
1.  Sort the segments in descending order of their scores.
                        {ea1,ea2,⋯,ean}
              such that score(ea1) ≥ score(ea2) ≥ ⋯ ≥ score(ean)
2.  for each i in 1,2,⋯,n
3.    if d(V') ÷ framerate ≤ Ttime
4.      then Add eai to V'
5.      else break
6.    end if
7.  end for
```

Algorithm 5.1 Selection of Extracted Video Segments

Example 5.3 *Given the time constraint, T_{time}, we try to make selection of extracted video segments according to the segments and the scores that we calculated in* ***Example 5.2****. The selection result is shown in Figure 5.3.*

$$\text{Given } T_{time} = 15\,\text{sec},\ framerate = 30\,fps$$

$$score(e_1) \geq score(e_2) \geq score(e_6) \geq score(e_4) \geq score(e_3) \geq score(e_7) \geq score(e_5)$$

therefore,

$$V' = \{e_1, e_2, e_6\}$$

and

$$\left.\begin{array}{l} d(V') = 78 + 18 + 298 = 394 \\ T_{time} \times framerate = 15 \times 30 = 450 \end{array}\right\} \Rightarrow d(V') \leq T_{time} \times framerate$$

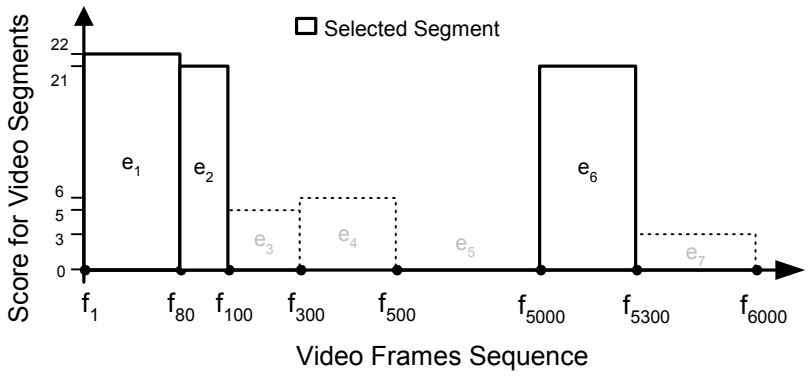


Figure 5.3 Selecting Extracted Video Segments

5.3.4 Refining the Selection Result

In the last step for the video summarization algorithm, we refine the selection result above, in order to generate a smoother video summary. Since we find that the selected video segments are always short and disjointed, then the resulting summary cannot be browsed smoothly as shown in Figure 5.4. As a result, we propose this refinement process to solve the above problem. Based on the time sequence of the selected segments, we collect adjacent segments together using the clustering technique. Thus, we can pick a cluster of segments, which is long enough and continuous, instead of those disjointed short segments in order to make the video summary smoother.

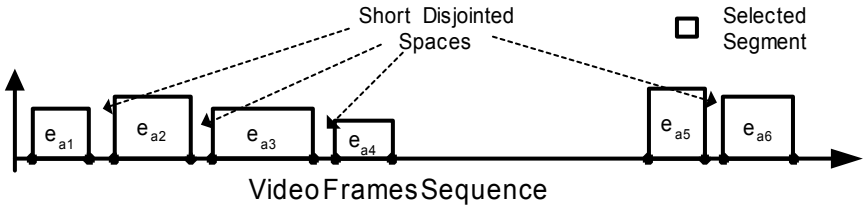


Figure 5.4 Problem for Disjointed Video Segments

We apply the K-mean clustering technique [19] in the refinement process. The number of resulting clusters is set by the clustering control constant, *K*, defined by the user. The clustering process classifies the selected segments into *K* clusters along the video frames sequence. Then the number of disjointed segments in the video summary is limited by the *K* clusters.

We model our refinement process into the K-mean clustering algorithm in Algorithm 5.2. For a selected video segments, e_i, and a mean point, m_k, the distance between, $dist(e_i,m_k)$, is defined as in Equation (5.8). Then we can calculate the mean point, m_k, for a cluster, u_k, using Equation (5.9). An example for K-mean clustering is shown in Figure 5.5.

$$\begin{aligned} &\text{Given } e_i = [f_x, f_y], m_k = f_z, \\ &dist(e_i, m_k) = \min(|x - z|, |y - z|) \end{aligned} \tag{5.8}$$

$$\begin{aligned} &m_k = f_z, \\ &\text{where } z = \frac{\sum_{\forall e_i = [f_x, f_y] \text{ in } u_k} \frac{x + y}{2}}{\text{number of members in } u_k} \quad , \quad k \in 1,2,\cdots,K \end{aligned} \tag{5.9}$$

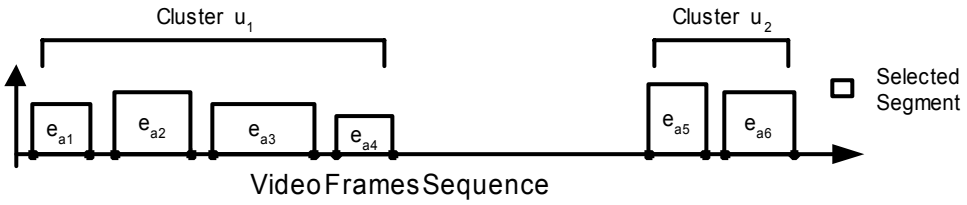


Figure 5.5 Result of K-mean Clustering

```
Input:  Given a set of selected video segments, {e1,e2,⋯,en}.
Output: K segment clusters, {u1,u2,⋯,uK}
1.  Make an initial guess of the means, m1,m2,⋯,mK, for clusters
    u1,u2,⋯,uK respectively.
2.  while changes in m1,m2,⋯,mK
3.    i) Assign each ei in {e1,e2,⋯,en} to the nearest cluster.
4.    ii) Calculate the new mean points using Equation (5.9).
5.  end while
```

Algorithm 5.2 K-mean Clustering for Selection Refinement

Now, we finalize our selection by picking clusters into the video summary. Once we find out the clusters, we connect all the member segments into a whole block of video frames sequence as shown in Figure 5.6. The block is so called a clustered video segment. Then we calculate the score for each clustered segment as the summation of the member scores times the corresponding segment length. It is shown in Equation (5.10).

$$\begin{aligned} &\text{Given a cluster } u = [f_x, f_y], \\ &score(u) = \sum_{\forall e_i \in u} score(e_i) \times length(e_i) \end{aligned} \tag{5.10}$$

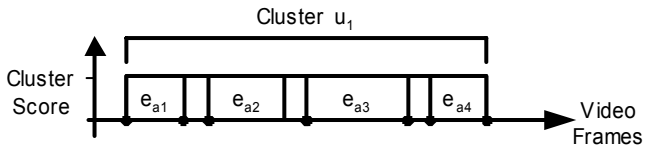


Figure 5.6 Transforming Selected Segments into a Clustered Segment

Since the disjointed spaces between the selected segments are now included, we need to remove certain clustered segments in order to satisfy the time constraint, T_{time}. Similar to the approach mentioned in Section 5.3.3, we first sort the clustered segments in descending order of their scores. Then, we select the clustered segments into the video summary, V', from one with the highest score,

until the time constraint is reached. The duration can be checked by Equation (5.11). The selection of clustered segments is illustrated in Figure 5.7.

$$d(V') = \sum length(u_i) \quad \text{such that } d(V') \le T_{time} \tag{5.11}$$

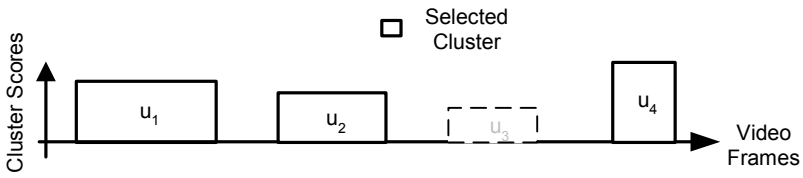


Figure 5.7 Selecting Clustered Segments in Video Summary

By using the refinement process, we pick clustered segments into the video summary instead of short and disjointed segments selected in Section 5.3.3. As a result, the video summary becomes a sequence of clustered segments as shown in Equation (5.12). It will be smooth enough to provide the user the major video contents that he wants. The quality for the video summary can be measured by the score defined in Equation (5.13), which is modified from **Definition 5.6**.

$$\text{Given selected clusters, } u_1, u_2, \cdots, u_n, \quad V' = (u_1, u_2, \cdots, u_n) \tag{5.12}$$

$$score(V') = \frac{\sum_{i=1}^{n} score(u_i)}{d(V')} \tag{5.13}$$

5.4 Video Summary in SMIL

In order to demonstrate the video summary, we transform our resulting set of clustered segments into SMIL format. As we have mentioned in Section 2.4, we can specify the temporal behavior of video clips in a SMIL presentation. Therefore, in our video summary, we make the required clustered segments in form of video clip objects for SMIL, and order them into a video sequence. In **Example 5.4**, we demonstrate the transformation into SMIL.

Example 5.4 *Given a video summary, V'=(u_1,u_2,u_3), where the clustered segments are as follows.*

$$u_1 = [f_{1500}, f_{2100}], u = [f_{4200}, f_{4500}], u = [f_{8100}, f_{8250}]$$

In Table 5.2, we transform the clustered segments into the time sequences of video according to the video frame rate. Given frame rate equals 30 fps.

Table 5.2 Clustered Segments in Form of Time

Clustered Segments	Time in Seconds	
	Begin	End
u_1	50	70
u_2	140	150
u_3	270	275

Each clustered segments is written as a video clip object in SMIL. For example, u_1 is shown in Figure 5.8. All the clustered segments are organized in a sequential order in SMIL as shown in Figure 5.9. In Figure 5.10, we play the resulting SMIL video summary.

```
<video id="u1" src="rtsp://host1/1.rm"
  clip-begin="50s" clip-end="70s" region="video"
  fill="freeze"/>
```

Figure 5.8 A Clustered Segment in SMIL

```
<?xml version="1.0"?>
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language">
<head>
 <layout type="text/smil-basic-layout">
  <root-layout width="362" height="298" background-color="black"/>
  <region id="video" left="5" top="5" width="352" height="288"
    fit="fill"/>
 </layout>
</head>
<body>
 <seq>
  <video id="u1" src="rtsp://host/data.rm" clip-begin="50s"
     clip-end="70s" region="video" fill="freeze"/>
  <video id="u2" src="rtsp://host/data.rm" clip-begin="140s"
     clip-end="150s" region="video" fill="freeze"/>
  <video id="u3" src="rtsp://host/data.rm" clip-begin="270s"
     clip-end="275s" region="video" fill="freeze"/>
 </seq>
</body>
</smil>
```

Figure 5.9 An Example Source for SMIL

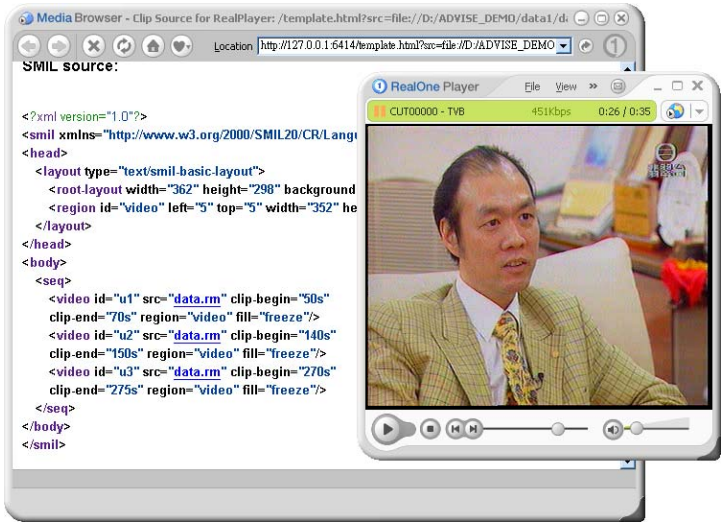


Figure 5.10 SMIL Video Summary

5.5 Evaluations

In this section, we design two experiments to evaluate our video summary. First we evaluate the quality of the video summary by the percentage of each feature extracted. Second, we evaluate the performance of the refinement process.

5.5.1 Experiment 1: Percentages of Features Extracted

As we have mentioned that the video summarization result depends greatly on the user's preferences, there is no standard method to evaluate the results. However, we can use several extreme cases of the user's inputs, such that we can easily determine whether the summarization algorithm works.

For different sets of user inputs, we set up our experiments as follows. We apply the inputs on a set of videos with duration around 200 seconds. If we set the time constraint, T_{time}, to be 60 seconds and all the weights for video features are the same, we would expect that our video summary includes around 30% of each feature from the source video. However, when we try to increase the weight for a specific feature, we would expect that a higher percentage of that

feature can be extracted from the source video. The percentage extracted for a feature, P_{total}, can be calculated using Equation (5.14). We also calculate the percentage of the extracted feature in a video summary, $P_{summary}$, as shown in Equation (5.15).

$$P_{total}(feature) = \frac{\text{duration containing that feature in video summary}}{\text{total duration containing that feature in the source video}} \quad (5.14)$$

$$P_{summary}(feature) = \frac{\text{duration containing that feature in video summary}}{\text{total duration of the video summary}} \quad (5.15)$$

We present six cases in this experiment. They are: **Case (1) Same Weight for Each Feature**, **Case (2) Human Face Favoring**, **Case (3) Male Voice Favoring**, **Case (4) Female Voice Favoring, Case (5) Volume Level Favoring**, and **Case (6) Caption Text Favoring**. The inputs are tabulated in Table 5.3.

Table 5.3 Inputs for Experiment 1

	w_{face}	w_{male}	w_{female}	w_{volume}	$w_{caption}$	T_{time}	K
Case (1)	5	5	5	5	5	60	10
Case (2)	10	1	1	1	1	60	10
Case (3)	1	10	1	1	1	60	10
Case (4)	1	1	10	1	1	60	10
Case (5)	1	1	1	10	1	60	10
Case (6)	1	1	1	1	10	60	10

The average values for all the results are tabulated in Table 5.4. According to the results, we find that the percentages, P_{total} and $P_{summary}$, vary with the input weights. A higher weight for the feature results in higher percentages of that feature in the source video and the video summary. Therefore, we can conclude that our video summary can be successfully customized to fit the user's interest.

Table 5.4 Average Values for All Results in Experiment 1

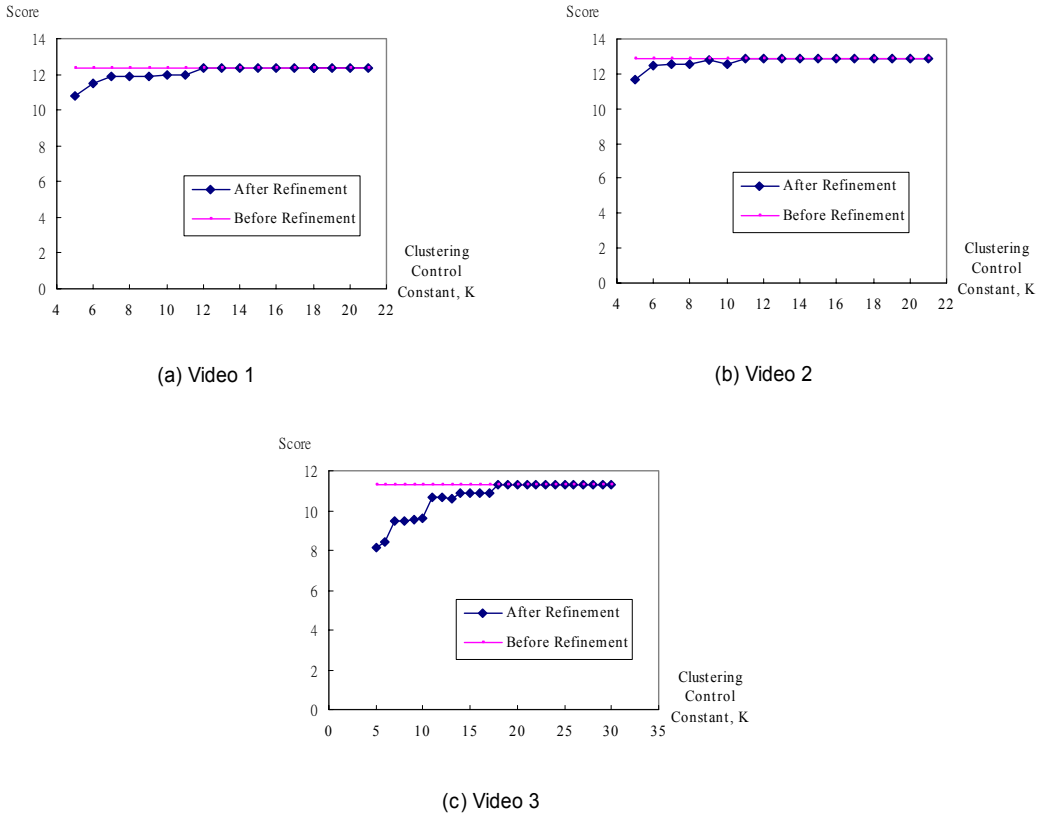
	Human Face	*Male voice*	*Female voice*	*Volume level*	*Caption text*
Case (1) :					
P_{total}	0.45	0.28	0.30	0.29	0.64
$P_{summary}$	0.77	0.25	0.62	0.45	0.14
Case (2) :					
P_{total}	0.50	0.20	0.34	0.06	0.23
$P_{summary}$	0.96	0.19	0.78	0.10	0.06
Case (3) :					
P_{total}	0.41	0.84	0.08	0.07	0.46
$P_{summary}$	0.76	0.80	0.17	0.11	0.11
Case (4) :					
P_{total}	0.35	0.15	0.46	0.30	0.26
$P_{summary}$	0.61	0.18	0.97	0.47	0.06
Case (5) :					
P_{total}	0.13	0.03	0.43	0.64	0.30
$P_{summary}$	0.22	0.03	0.90	0.97	0.07
Case (6) :					
P_{total}	0.43	0.32	0.22	0.15	0.89
$P_{summary}$	0.83	0.31	0.52	0.26	0.22

5.5.2 Experiment 2: Evaluation of the Refinement Process

In this experiment, we evaluate the effect of the refinement process in making a smoother video summary. Since clustered segments are selected instead of all the short and disjoint segments with highest scores, we sacrifice the quality of the video summary in certain extent. Therefore in this experiment, we examine the effect on the score of the video summary while increasing the number of clusters.

Figure 5.11 shows the results for three videos. According the graphs, we find that the score of video summary for using clustered segments (after refinement) approaches the score of video summary for using short and disjoint segments (before refinement), as the number of clusters increases. There are penalty in scores while we are reducing the number of number of clusters wanted for refinement.

However, from the above results, we can also find that the penalty decreases zero until certain value for number of clusters. In Table 5.5, we can see that the number of fragments in video summary, at which there are no penalty, is greatly reduced after the refinement process; hence, the resulting summary is more continuous and smoother.



(a) Video 1

(b) Video 2

(c) Video 3

Figure 5.11 Results in Graphs for Experiment 2

Table 5.5 Results for Experiment 2

	Before Refinement	After Refinement
Video 1	21	13 (at K=12)
Video 2	21	10 (at K=11)
Video 3	30	18 (at K=18)

Chapter 6

Video Matching Using V-ToC

In this chapter, we describe our video matching algorithms for finding similar videos. Based on the V-ToC tree structure generated in Chapter 4, we can apply a tree matching algorithm to measure the similarity between two V-ToC trees.

We propose two tree matching algorithms in this chapter [27]. The first algorithm is the non-ordered tree matching algorithm, and the second one is the ordered-tree matching algorithm. Our tree matching algorithms are different a general tree matching algorithm because the V-ToC tree we generated is well structured and with tree depth always equal to four. We need to measure the similarity according the four levels of video components. In both of our algorithms, the matching processes start from the top of the tree and proceeds to the next sub-level in an orderly manner, i.e., scene to scene, group to group, and shots to shots. Similarity measure is calculated at each corresponding level between the two video trees. There is a major different between the two algorithms. The non-ordered tree matching algorithm does not consider the temporal ordering of video features, while the ordered tree matching algorithm considers the features ordering as a key factor to determine the similarity between videos.

We detail the two video tree matching algorithms in the following sections. In Section 6.1, we first define several key terms for video tree matching. Then in Section 6.2, we talk about the video features used for video matching. In Section 6.4 and 6.4, we introduce the non-ordered tree matching algorithm and the

ordered tree matching algorithm respectively. Finally, we evaluate our video tree matching algorithms with a set of experiments in Section 6.5.

6.1 Terms and Definitions

In this section, we define several key terms for the video tree matching algorithms. Since our video matching algorithm is applied on top the V-ToC tree structure, those definitions in Section 4.1.1 also hold. Apart from those definitions, we define a new set of concepts in the following.

Definition 6.1 *The **feature similarity** between two video tree node, $node_x$ and $node_y$, is defined as $sim(node_x, node_y)$. This is a normalized score ranged from 0 to 1, where 0 means dissimilar and 1 means similar. The input nodes must be at the same tree level.*

Definition 6.2 *A **child similarity matrix**,* **ChildSim***, of two video tree node, $node_x$ and $node_y$, is the table, which stores all the feature similarities between child nodes of $node_x$ and child nodes of $node_y$. The value for i-th column and j-th row in the matrix is ChildSim(i,j). It is defined in Equation (6.1). Besides, we give an example child similarity matrix at Example 6.1.*

$$\begin{aligned} &\text{Given } node_x = (child_{a1}, child_{a2}, \cdots) \text{ and } node_y = (child_{b1}, child_{b2}, \cdots): \\ &\quad ChildSim(i,j) = sim(child_{ai}, child_{bj}) \qquad \text{for all } child_{ai} \text{ in } node_x, child_{bj} \text{ in } node_y \end{aligned} \tag{6.1}$$

Example 6.1 *Given two video groups $g_1 = (s_1, s_3, s_8, s_{13})$ and $g_2 = (s_5, s_9, s_{10})$. If we have the following feature similarities, then we can tabulate the values into a child similarity matrix, as shown in Figure 6.1.*

$$\begin{aligned} &sim(s_1, s_5) = 0.5,\ sim(s_1, s_9) = 0.9,\ sim(s_1, s_{10}) = 0.1, \\ &sim(s_3, s_5) = 0.8,\ sim(s_3, s_9) = 0.4,\ sim(s_3, s_{10}) = 0.4, \\ &sim(s_8, s_5) = 0.1,\ sim(s_8, s_9) = 0.6,\ sim(s_8, s_{10}) = 0.9, \\ &sim(s_{13}, s_5) = 0.2,\ sim(s_{13}, s_9) = 0.0,\ sim(s_{13}, s_{10}) = 1.0 \end{aligned}$$

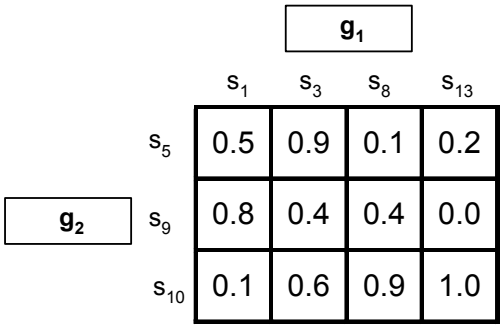


Figure 6.1 Child Similarity Matrix

Definition 6.3 *The **video similarity** between, $video_x$ and $video_y$, is defined as $sim(video_x, video_y)$. We apply a video tree matching algorithm, which make use of the child similarity matrix to generate a one-to-one mapping of the most similar child nodes, such that a similarity score can be propagated from the feature similarities of the child nodes.*

6.2 Video Features Used for Matching

There are two video features that are used in our video matching algorithms. The first one is the color histogram and the second one is the shot style.

The color histogram feature is useful for matching the global color content of frames in the video. As we have described in Section 4.1.2, we make use of the histogram difference between two frames to determine the visual similarity. If the difference is small, the frames are similar; otherwise, these frames are different. In our algorithm, since a video shot is a sequence of frames with similar content, the key frame is used a representative in the matching process. Then the frame-to-frame distance of the key frames in two video shots is used to calculate the similarity between two video shots.

The shot style feature is composed of the camera motion and the length of the shot. The camera motion [4] consists of zooming, horizontal movements, vertical movements, and still which means that there is no camera motion. In a shot, there could be many camera motion segments. For examples, a shot of a

person may consist of zooming in and zooming out camera segments. In our algorithm, we use the first camera motion segments to represent the camera motion for the shot. The length of the shot is the summation of all the camera motion segment durations in the shot. The camera movement and the length of the shot can reflect the pace of the video. For example, if a shot is short and the camera moves in different directions, we would expect that the video has a fast pace. The pace of video can help us to determine the type of video since we know that action videos are faster and artistic videos are typically slower.

6.3 Non-ordered Tree Matching Algorithm

In the non-ordered tree matching, video features are matched without any constraint of the temporal sequence. In other words, this method is able to match video features in any order, as shown in Figure 6.2.

Figure 6.2 Matching Video Using a Non-ordered Approach

The algorithm examines the structural V-ToC trees of two videos in a top-down manner, i.e., from the video level to the video shot level. However the scoring of similarity of the video is propagated from bottom-up.

At the video level, the video scene level and the video group level

Before we get down to the video shot level, there are three steps to work out the feature similarities for each level. First, the algorithm needs to retrieve the feature similarities of all child nodes by traversing down the tree. For example, when we need to find the similarities between two videos, we need to know how similar their scenes are. In the second step, we tabulate all the feature similarities of the child nodes in a child similarity matrix defined in **Definition 6.2**. Then in the third step, we calculate the feature similarity of the current level with a scoring function called *MaxSum()*, and propagate the resulting score up to the parent level.

The *MaxSum()* function is used to sum up the similarities of the best match of the child nodes and then return the normalized value of the sum. We use the feature similarities at **Example 6.1** to show an example of the best match in Figure 6.3. Figure 6.4 demonstrates the matching in tree format. To calculate the sum, we can add up the maximum score at each column. However, in most of the cases, the number of scenes, groups and shots in videos are not the same. Then, the tabulated matrix of child feature similarities is not in square shape, and there is one to multiple mappings. For example, the third row in Figure 6.3, s_{10} from g_2 matches both s_8 and s_{13} of g_2. Then, the sum calculated is different if we take the summation of row maximum instead of column maximum. Therefore, in our algorithm, we want to set the feature similarity to respect to the dimension with a smaller value, such that we would not penalize matching of video segments to its full version. We explain this penalty in Figure 6.5. Hence, when the number of rows is smaller than the number of columns, the feature similarity is calculated by dividing the row maximum sum with the number of rows; otherwise, the similarity is calculated by dividing column maximum sum with the number of columns. Equation (6.2) shows the calculation of the feature similarity between $node_x$ and $node_y$. Let the number of child nodes for $node_x$ and the number of child nodes for $node_x$ be u and v respectively. Then, the number of columns and the number of rows in the child similarity matrix are also u and v respectively.

$$sim(node_x, node_y) = MaxSum(\textbf{ChildSim}) = \begin{cases} \dfrac{\sum_{i=1}^{u} \max_{\forall 0 \le j \le v} (ChildSim(i,j))}{u}, & \text{if } u \le v \\ \dfrac{\sum_{j=1}^{v} \max_{\forall 0 \le i \le u} (ChildSim(i,j))}{v}, & \text{if } v < u \end{cases} \quad (6.2)$$

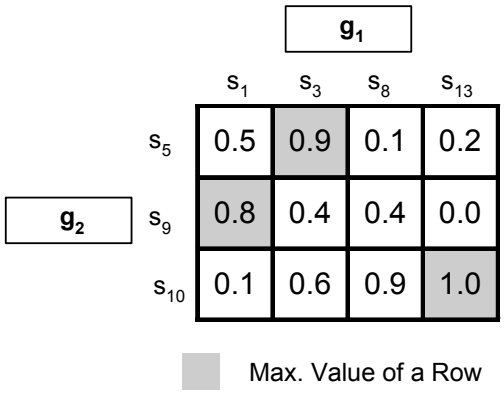


Figure 6.3 The Best Matched Nodes in **ChildSim**

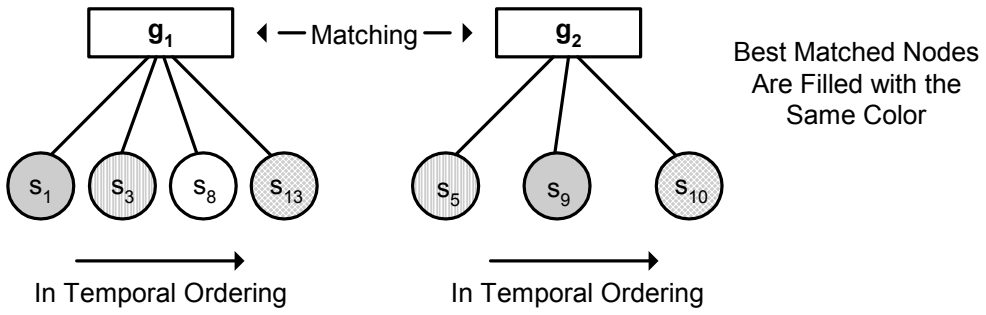


Figure 6.4 Non-ordered Tree Matching

Example 6.2 *Given two video groups* $g_1 = (s_1, s_3, s_8, s_{13})$ *and* $g_2 = (s_5, s_9, s_{10})$. *We get the feature similarities of child nodes as shown in Figure 6.3. Since the number of rows is fewer than the number of columns, therefore we select the best matched nodes according to the rows. The feature similarity for* g_1 *and* g_2 *is calculated as follows.*

$$sim(g_1, g_2) = \frac{0.9 + 0.8 + 1.0}{3} = \underline{\underline{0.9}}$$

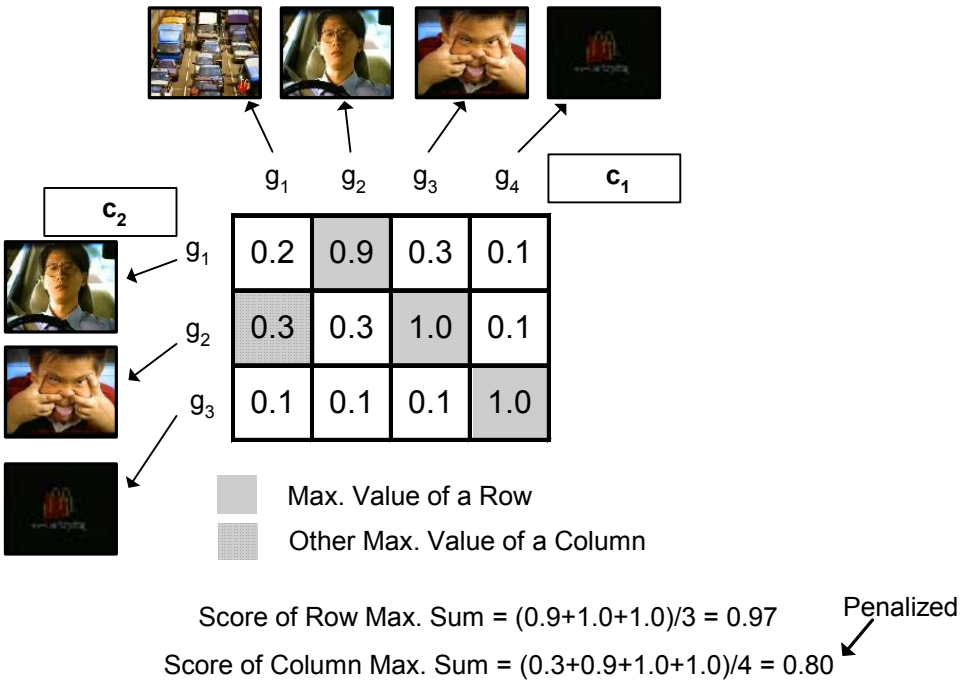


Figure 6.5 Penalty of Matching Video Segments

At the video shot level

The algorithm calculates the feature similarities based on the shot feature, which is the color histogram and the shot style.

The color histogram similarity is calculated using the key frames of shots. We use the frame-to-frame distance of color histogram defined in Equation (4.4). The result, which is the difference of the color histograms, is normalized. Then the color histogram similarity, $ColorSim(s_x,s_y)$, between video shots s_x with key frame k_x and s_y with key frame k_y is defined in Equation (6.3). We apply a simple normalizing function, *normalize()*, to normalize the frame-to-frame distance. As a result, the resulting color histogram similarity is a value ranged from 0 to 1.

$$ColorSim(s_x,s_y) = 1 - normalize(z(k_x,k_y)) \quad (6.3)$$

The shot style feature similarity is set to be the ratio of lengths of video shots when the representative camera motions are the same. For example, the camera motions of two shots, s_x and s_y, are the same, and length of s_x is smaller than that of s_y, then the shot style feature similarity is equal to length of s_x divided by length of s_y, as shown in Equation (6.4). The length of a video shot is mentioned in Equation (4.12).

$$StyleSim(s_x,s_y) = \begin{cases} \dfrac{length(s_x)}{length(s_y)}, & \text{if } length(s_x) < length(s_y) \\ \dfrac{length(s_y)}{length(s_x)}, & \text{otherwise} \end{cases} \tag{6.4}$$

After calculating both feature similarities, the algorithm then propagates the shot level feature similarity to the upper level.

6.4 Ordered Tree Matching Algorithms

The ordered tree matching algorithm is different from the non-ordered matching in the previous section because it considers the temporal ordering of the shot features. It allows only matching of feature similarities with temporal constraint. Therefore, the condition on Figure 6.2 is not considered as very similar any more. The score of similarities propagated up is the summation of feature similarities for the best-ordered child nodes matching.

An ordered tree matching is significant because it can capture the difference in video similarity due to the changes of features ordering. The reordering of features can form a different tree structure. The non-ordered algorithm cannot detect these kinds of structural differences. An ordered matching algorithm is designed to tighten the similarity measurement by the temporal sequences constraints, so that we give more concern on the video structure.

In this algorithm, we traverse the V-ToC tree in the same manner as in the non-ordered tree matching algorithm.

At the video level, the video scene level and the video group level

At each tree level, we use a *MaxOrderedSum()* function instead of the *MaxSum()* function in the non-ordered matching. The *MaxOrderedSum()* function considers the ordering while finding out the sum of feature similarities for the best matched child nodes. Figure 6.6 shows an example best ordered and matched child nodes on the child similarity matrix used in **Example 6.1**. The selected set matches a sequence of video shots from both videos. Figure 6.7 demonstrates this matching function in a tree format.

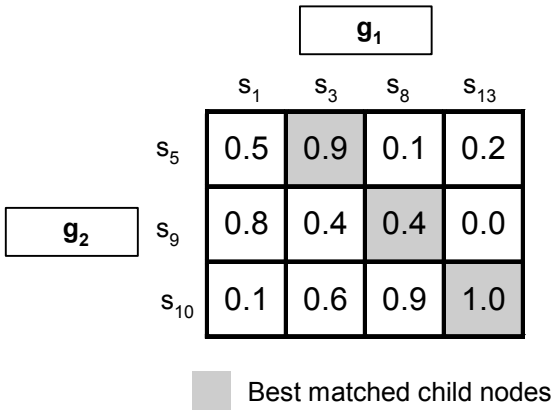


Figure 6.6 The Best Ordered and Matched Nodes in **ChildSim**

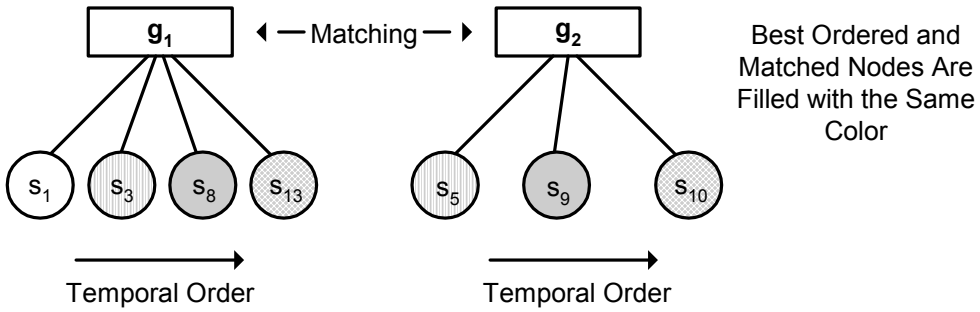


Figure 6.7 Ordered Tree Matching

We use the dynamic programming technique to make the calculations more efficient [37]. There are four steps for our algorithm to find out the feature similarity of the current level. In the first step, we initialize a matrix *D* with all values equal to zero. Then in the second step, we fill up the matrix according to Equation (6.5), such that we look for a maximum sum of the similarity scores

along the child nodes sequence. Let **ChildSim** contains *u* columns and *v* rows. The third step is getting the maximum sum at *D(u+1,v+1)* once *D* is filled. Finally, the feature similarity is calculated by the normalized sum as shown in Equation (6.6). Now, we use the data from **Example 6.1** to show the feature similarity calculation in **Example 6.3**.

$$D(i+1, j+1) = \max(D(i, j) + ChildSim(i, j), D(i, j+1)) \tag{6.5}$$

$$sim(node_x, node_y) = MaxOrdered\ Sum(\textbf{ChildSim}) = \max(\frac{sum}{u}, \frac{sum}{v}) \tag{6.6}$$

Example 6.3 *For the data from **Example 6.1**, we need to initialize a 5 by 4 matrix, D, with each value equals to zero. We fill in the values in D, according to Equation* (6.5). *The resulting D is shown in Figure 6.8. Then, we can get the maximum sum at D(5,4) = 2.3. The feature similarity between g_1 and g_2 is calculated by:*

$$sim(g_1, g_2) = \max(\frac{2.3}{4}, \frac{2.3}{3}) = \underline{\underline{0.77}}$$

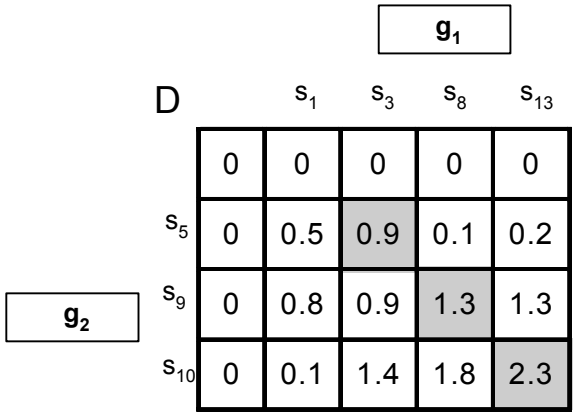


Figure 6.8 An Example for Dynamic Programming

At the video shot level

We extract the two video features using the same techniques as mentioned in the non-ordered tree matching algorithm, Section 6.3. We propagate the calculated feature similarities to the upper levels until we get the final result.

We implement the whole ordered tree matching algorithm using a recursive function. The algorithm is summarized in Algorithm 6.1. We start the matching process at the root level by $OrderedMatching(video_a, video_b)$. It traverses all the tree nodes along each tree level to propagate the feature similarities up recursively. We use Figure 6.9 to illustrate to this recursive process.

Input: Given two tree nodes, $node_x = (child_{a1}, child_{a2}, \cdots, child_{au})$ and $node_y = (child_{b1}, child_{b2}, \cdots, child_{bv})$.

Output: Score of feature similarity ($ColorSim$ or $StyleSim$)

1. **function OrderedMatching(** $node_x$, $node_y$ **)**
2. **if** current node level is video shot level
3. **then**
4. **return** either $ColorSim$ or $StyleSim$ as feature similarity
5. **else** //at upper levels
6. i) Initialize D with elements equal to zero.
7. ii) Fill in D for each $i \in 1, \cdots, u$ and $j \in 1, \cdots, v$ with:

$$D(i+1, j+1) = \max(D(i, j) + OrderedMatching(child_{ai}, child_{bj}), D(i, j+1))$$

8. iii) **return** $\max(\frac{D(u+1, v+1)}{u}, \frac{D(u+1, v+1)}{v})$
9. **end if**
10. **end function**

Algorithm 6.1 Recursive Dynamic Programming for Ordered Tree Matching

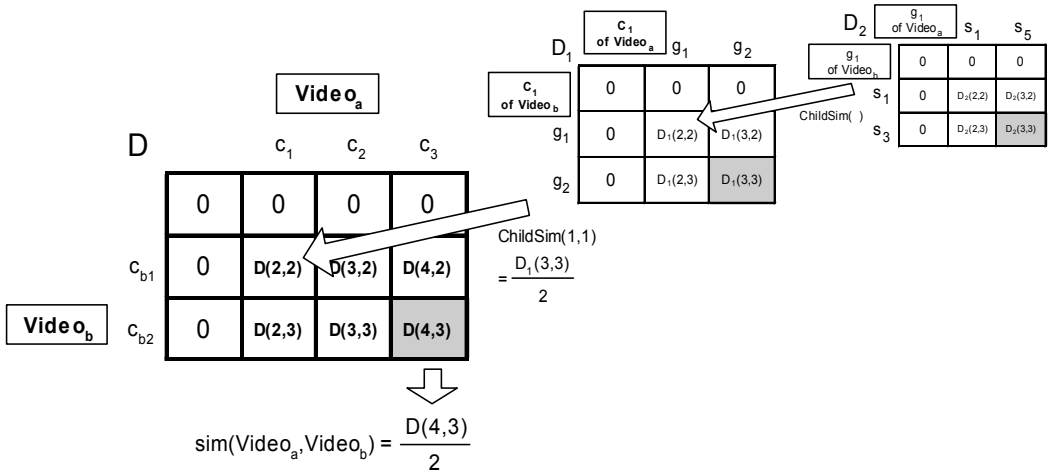


Figure 6.9 An Example for Recursive Dynamic Programming

6.5 Evaluation of Video Matching

In this section, the proposed tree matching algorithms will be evaluated by comparing the results of a small set of videos with the human's ranking results; one example is in Figure 6.10. Some information of the videos is shown in Table 6.1. The human's ranking results of the videos are shown in Table 6.2 and Table 6.3. There are 5 videos matching with each others using the proposed algorithms.

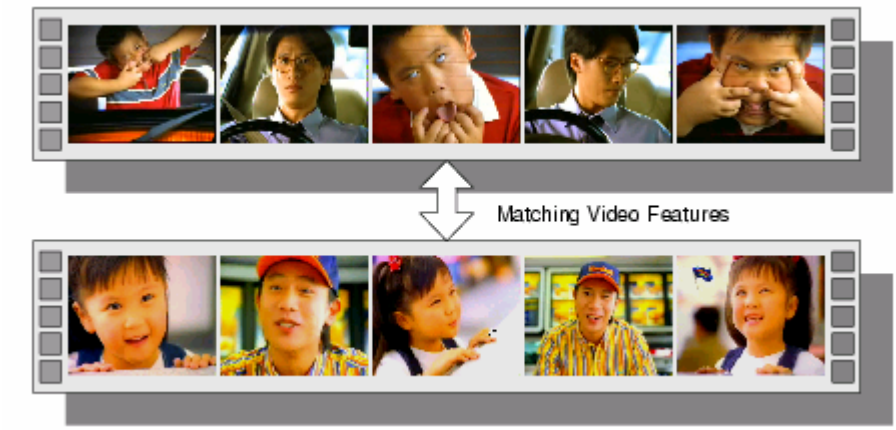


Figure 6.10 Matching Video Features

Table 6.1 Video Tree Structure Information

Videos	Number of shots	Number of Groups	Number of scenes
Video 1	12	4	3
Video 2	14	5	2
Video 3	16	6	3
Video 4	18	6	2
Video 5	27	9	6

Table 6.2 Human's Ranking for Color Histogram Feature

Ranking of Videos	Most Similar 1	2	3	Least Similar 4
Video 1	Video 2	Video 3	Video 4	Video 5
Video 2	Video 1	Video 4	Video 3	Video 5
Video 3	Video 1	Video 2	Video 4	Video 5
Video 4	Video 2	Video 1	Video 3	Video 5
Video 5	Video 2	Video 1	Video 3	Video 4

Table 6.3 Human's Ranking for Shot Style Feature

Ranking of Videos	Most Similar 1	2	3	Least Similar 4
Video 1	Video 2	Video 3	Video 4	Video 5
Video 2	Video 1	Video 3	Video 4	Video 5
Video 3	Video 1	Video 2	Video 4	Video 5
Video 4	Video 2	Video 1	Video 3	Video 5
Video 5	Video 2	Video 3	Video 1	Video 4

6.5.1 Applying Non-ordered Tree Matching

According to the feature similarity scores calculated by the non-ordered tree

matching algorithm, we rank the similarities between each video and the others. For example, when we match video 1 with the other 4 videos, if we find that video 2 have the highest similarity score, video 2 is the most similar one to video 1. The ranking results from non-ordered tree matching are shown on Table 6.4 and Table 6.5.

For color histogram features, we can compare Table 6.2 and Table 6.4. We find that the results are quite similar. The last column in Table 6.4 is the same as the last column in Table 6.2. That means the non-ordered algorithm can successfully find out the least similar video according to each source video. Also, the rows for video 3 and video 4 in Table 6.4 are the same as their corresponding rows in Table 6.2. So, our algorithm can determine exactly the same similarities with the human judgments for these two videos. Since there are only few interchanges in the resulting orders, our algorithm can measure the similarity for the color histogram feature quite accurately.

For shot style feature, we compare Table 6.3 and Table 6.5. The results in both tables are again quite similar. Our algorithm can find out all the least similar videos for the shot style feature. Moreover, the rows for video 3 and video 5 in both result tables are the same. Although there are again some interchanges in other rows, our algorithm is quite accurate in finding the similarity for the shot style feature.

Table 6.4 Ranking Results for Non-ordered Tree Matching for Color Histogram Feature

Ranking of Videos	**Most Similar 1**	**2**	**3**	**Least Similar 4**
Video 1	Video 3	Video 2	Video 4	Video 5
Video 2	Video 4	Video 1	Video 3	Video 5
Video 3	Video 1	Video 2	Video 4	Video 5
Video 4	Video 2	Video 1	Video 3	Video 5
Video 5	Video 3	Video 2	Video 1	Video 4

Table 6.5 Ranking Results for Non-ordered Tree Matching for Shot Style Feature

Ranking of Videos	Most Similar 1	2	3	Least Similar 4
Video 1	Video 3	Video 2	Video 4	Video 5
Video 2	Video 4	Video 3	Video 1	Video 5
Video 3	Video 1	Video 2	Video 4	Video 5
Video 4	Video 2	Video 3	Video 1	Video 5
Video 5	Video 2	Video 3	Video 1	Video 4

6.5.2 Applying Ordered Tree Matching

Similar to the ranking in non-ordered tree matching, we rank the videos according to the result of the ordered tree matching. The ranks are shown in Table 6.6 and Table 6.7.

We compare the results in Table 6.2 and Table 6.6 for color histogram feature. In these two tables, the results in the least similar columns are the same. Thus, the ordered algorithm can find out the least similar video according each source video. Also, refer to Table 6.6, the algorithm can identify three most similar videos in column 1 and the row for video 5 is the same as the corresponding row in Table 6.2. Therefore, our algorithm is able to give similar result as the human judgment for color histogram feature.

For the shot style feature, we look up the results in Table 6.3 and Table 6.7. Our algorithm can find out that video 5 is the least similar to video 1, video 2 and video 4. Also, for video 4 and video 5, the video 2 is most similar candidate among the video set. These results in Table 6.7 are the same in Table 6.3. The result for shot style feature is less accurate because it is quite difficult for human to remember and judge the ordered similarity for video shot motions. Therefore, our computed results cannot well satisfy the human judgment for shot style feature.

Table 6.6 Ranking Results of Ordered Tree Matching for Color Histogram Feature

Ranking of Videos	Most Similar 1	2	3	Least Similar 4
Video 1	Video 3	Video 2	Video 4	Video 5
Video 2	Video 4	Video 5	Video 1	Video 3
Video 3	Video 1	Video 4	Video 2	Video 5
Video 4	Video 2	Video 3	Video 1	Video 5
Video 5	Video 2	Video 1	Video 3	Video 4

Table 6.7 Ranking Results of Ordered Tree Matching for Shot Style Feature

Ranking of Videos	Most Similar 1	2	3	Least Similar 4
Video 1	Video 3	Video 4	Video 2	Video 5
Video 2	Video 4	Video 5	Video 3	Video 1
Video 3	Video 4	Video 1	Video 2	Video 5
Video 4	Video 2	Video 3	Video 1	Video 5
Video 5	Video 2	Video 4	Video 3	Video 1

Chapter 7

Conclusion

Video over Internet is getting more popular now than ever before, due to the rapid growth of the Internet bandwidth and the growing use of video in education, entertainment, and information sharing. Among the vast video sources, it is difficult for users to search for their desired pieces. We address two problems about video retrieval. First, we do not know the contents of video before we download and browse it. Second, it is difficult to find videos with similar video contents. In our research, we have designed the web-based video retrieval system called ADVISE, Advanced Digital Video Information Segmentation Engine, to solve the above problems.

For the first problem mentioned above, we find that it is always more efficient for a user to search for his desired videos if some descriptions are provided. A meaningful video description can help us to know the contents at once, so that we do not need to waste the time on downloading a huge video clip, which in fact, we are not interested. As a result, video descriptions can enhance efficient browsing and retrieval of video contents. Textual description extracted from video caption text is a commonly used solution; however, text may not always well describe the video as the contents are delivered by combining visual, audio, and textual information. ADVISE proposes two kinds of video descriptions to solve the problem. It generates the video table-of-contents, V-ToC, and the video summary for user to know the video contents in a short period of time.

The V-ToC is an image-based video description. It describes the video contents to the users on the Internet. We propose the automated generation of the V-ToC structure. A color histogram based approach has been employed in our system. We have improved a general color histogram based method using regional color histograms and the adaptive threshold. The resulting structure is in form of a four levels tree structure. We have designed an XML structure to store the V-ToC. A DTD is defined in order to maintain the consistence of XML. The XML V-ToC is further presented on a web-based interface using the XSL transformation. In addition, we have performed an experiment to evaluate the V-ToC structure generated

The video summary collects the essential features of a video. It is used to provide more video information then the V-ToC. We proposed the video summarization algorithm, which accepts user's inputs. The user can set the weights for video features, and the time constraint he wanted. Therefore, the resulting video summary would be suitable for the need of the user. We proposed a clustering approach to refine the selection of video segments into the video summary. It can increase the smoothness of the summary by reducing the number of fragments. We have designed two experiments to evaluate the video summarization algorithm. The first one has evaluated the flexibility for customizing the video content. We have found that the algorithm can generate video summary according the user's inputs. In the second experiment, we evaluate the effect of the refining process. We have found that the refinement process reduces the score of the video summary in some extents; however, it helps greatly on improving the smoothness of the summary.

In the second problem, we focus on finding similar videos contents. There are various video matching algorithms developed, but seldom of them consider the structure of video. We find that using different algorithms and different video features can result in a different matching. Since we have built up the V-ToC structure in a tree format, therefore, we proposed two tree matching algorithms in our ADVISE system to match the video contents.

We have proposed two tree matching algorithms. The first one is the non-ordered tree matching algorithm and the second one is the ordered tree matching algorithm. The major difference between them is the concern on temporal ordering of the video features. The ordered matching algorithm considers the temporal ordering while the non-ordered one not. We applied two video features on video matching. They are the color histogram feature and the shot style feature. We have evaluated the video matching algorithms using a set of experiment. We found that the video matching algorithms are able to determine the video similarity.

To demonstrate all the proposed works, we have implemented ADVISE into a practical system. We have constructed the backend video processing engine to generate the V-ToC and to perform video matching. Then, we have built the web-based video retrieval system, which organizes the V-ToC, provides an interface for video summarization, and also is used for querying the video matching results.

Finally, we summarize our contributions.

- The ADVISE system, which enhanced video browsing and retrieval system on the Internet, has been proposed.
- The generation and presentation of the image-based video description, V-ToC, have been proposed. The V-ToC structure has been evaluated using a set of experiments.
- The automated summarization of video into SMIL format has been proposed. Experiments have been performed to evaluate the video summarization results.
- Two video tree matching algorithms for measuring the similarity between videos have been proposed. The results have been evaluated by a set of experiments.

We find the proposed ADVISE system brings us convenience in searching videos. The V-ToC, video summary, and video matching work well in ADVISE to assist the video browsing and retrieval for users on the Internet.

There are three major directions that we can further enhance our research work. First, we need to improve the accuracy of the V-ToC construction process. We can employ more video features other than color histograms to assists the video shot detections and groupings. A better construction of V-ToC can also improve the video matching results. Second, we can further research on the video summarization algorithm in order to provide a more optimized content selection. A constraint satisfaction problem is one possible way to model the video summarization. As a result, we can optimize summarization process using constraint satisfaction programming. Third, we can further extend the data management techniques for our XML V-ToC. We can make use of the XML hierarchical structure to design some video information searching schemes. Then the extracted video features can be well-organized into an XML-based database system. The querying of video features will be made more efficient. Our ADVISE system can be extended by further research on these three directions.

Bibliography

[1] D.A. Adjeroh, M.C. Lee, I. King. A Distance Measure for Video Sequence Similarity Matching. In *International Workshop on Multi-Media Database Management Systems*, pages 72-79, 1998.

[2] Adobe Systems Incorporated, Sharing Photos On the Web, Sept 1999. http://www.adobe.com/aboutadobe/pressroom/pressmaterials/pdfs/activeshare/activeshare.pdf

[3] Apple Computer Inc., Digital Video: Putting Ideas in Motion with Apple Digital Video Technologies. http://www.apple.com/scitech/appletech/dv/

[4] E. Ardizzone, M. La Cascia, A. Avanzato, A. Bruna. Video Indexing Using MPEG Motion Compensation Vectors. In *Proceedings of IEEE Conference on Multimedia Computing and Systems* (ICMCS), volume 2, pages725-729, June, 1999.

[5] M. Bertini, A.D. Bimbo, P. Pala. Content-based indexing and retrieval of TV news. In *Elsevier Pattern Recognition Letters*, volume 22, pages 503-516, 2001.

[6] J.S. Boreczky, L.A. Rowe. Comparison of video shot boundary detection techniques. In *Proceedings of SPIE Conference: Storage and Retrieval for Image and Video Databases*, volume 2670, pages 170-179, 1996.

[7] P. Browne, A.F. Smeaton, N. Murphy, N. O'Connor, S. Marlow, C. Berrut. Evaluating and Combining Digital Video Shot Boundary Detection Algorithms. In *Proceedings of the Irish Machine Vision and Image Processing Conference (IMVIP 2000)*, August 2000.

[8] P. Chiu, A. Girgensohn, W. Polak, E. Rieffel, L. Wilcox. A genetic algorithm for video segmentation and summarization. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1329-1332, 2000.

[9] M. Christel, D. Martin. Information Visualization within a Digital Video Library. *Journal of Intelligent Information Systems*, volume 11, no. 3, pages 235-257, 1998.

[10] M.G. Christel, B. Maher, A. Begun. XSLT for Tailored Access to Digital Video Library. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*, page 290-299, June 2001.

[11] Byung Tae Chun, Younglae Bae, Tai-Yun Kim. Text Extraction in Videos using Topographical Features of Characters. In *1999 IEEE International Fuzzy Systems Conference Proceedings*, August, 1999.

[12] Creativepro.com, POPcast Selected by MGI to Offer Personal Broadcasting Services to MGI VideoWave 4 Users, November 14, 2000. http://www.creativepro.com/story/news/10207.html

[13] A. Dailianas. Comparison of automatic video segmentation algorithms. In Proceedings of SPIE Photonics East'95: Integration Issues in Large Commercial Media Delivery Systems, October 1995.

[14] Z. H. Gao, L. Mo. *Final Year Project LYU0103*: *Speech Recognition Techniques for Digital Video Library*, The Chinese University of Hong Kong, 2001.

[15] A. Hanjalic, G.C. Langelaar, P.M.B. van Roosmalen, J. Biemond, R.L. Lagendijk. *Image and Video Databases: Restoration, Watermarking and Retrieval*, Elsevier Science, ISBN 0-444-50502-4, Amsterdam (NL), 2000.

[16] A.G. Hauptmann, M.J. Witbrock, M.G. Christel. Artificial Intelligence Techniques in the Interface to a Digital Video Library. In *Proceedings of the CHI-97 Computer-Human Interface Conference New Orleans LA*, March 1997.

[17] R. Hjelsvold, S. Vdaygiri, Y. Léauté. Web-based Personalization and Management of Interactive Video. In *Proceedings of the Tenth International World Wide Web Conference*, page 129-139, May 2001.

[18] J. Hunter, S. Little. Building and Indexing a Distributed Multimedia Presentation Archive using SMIL. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries, ECDL '01,* pages 415-428, September 2001.

[19] A.K. Jain, M.N. Murty, P.J. Flynn. Data Clustering: A Review. In *ACM Computing Surveys*, 31(3), pages 264-323, September 1999.

[20] V. Kobla, D.S. Doermann, C. Faloutsos. VideoTrails: Representing and Visualizing Structure in Video Sequences. In *Proceedings of ACM Multimedia Conference*, pages 335-346, November 1997.

[21] J.L. Koh, C.S. Lee, A.L.P. Chen. Semantic video model for content-based retrieval. In *IEEE International Conference on Multimedia Computer and Systems*, volume 2, pages 472-478, 1999.

[22] W. Li, S. Gauch, J. Gauch, K.M. Pua. VISION: A Digital Video Library. In *ACM Digital Libraries*, pages 19-27, 1996.

[23] R. Lienhart. Automatic Text Segmentation and Text Recognition for Video Indexing. In *Multimedia Systems. Hrsg.: ACM. Berlin-Heidelberg: Springer-Verlag*, S. 69-81, volume 8, 2000.

[24] R. Lienhart, W. Effelsberg, R. Jain. VisualGrep: A Systematic method to compare and retrieve video sequences. In *Storage and Retrieval for Image and Video Databases VI*, SPIE, volume 3312, page 271, Jan 1998.

[25] MGI, *share.videowave.com*. http://share.videowave.com/

[26] R. Mohan. Video Sequence Matching. In Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, volume 6, pages 3697-3700, 1998.

[27] C.W. Ng, I. King, M.R. Lyu. Video Comparison Using Tree Matching Algorithm. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, volume 1, pages 184-190, Las Vegas, Nevada, USA, June 2001.

[28] C.W. Ng, M.R. Lyu. ADVISE: Advanced Digital Video Information Segmentation Engine. In *Poster Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.

[29] J.Y. Pan, C. Faloutsos. VideoGraph: A New Tool for Video Mining and Classification. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*, page 116-117, June 2001.

[30] P. Pan, G. Davenport. I-Views: A Community-oriented System for Sharing Streaming Video on the Internet. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, volume 33, issue 1-6, page 567-581, June 2001.

[31] Oratrix Development, *GRiNS Player for SMIL 2*. http://www.oratrix.com/GRiNS/

[32] Real Networks Inc, *RealPlayer 8*. http://www.real.com/playerplus/index.html

[33] Real Networks Inc, *RealOne Player*. http://www.realone.com

[34] Real Networks Inc, *Real System Server*. http://www.realnetworks.com/products/servers/professional/index.html

[35] Y. Rui, T.S. Hunag, and S. Mehrotra. Constructing Table-of-Content for Videos. In *ACM Multimedia Systems Journal, Special Issue Multimedia Systems on Video Libraries*, volume 7, no. 5, pages 359-368, Sept 1999.

[36] T. Sato, T. Kanade, E.K. Hughes, M.A. Smith. Video OCR for digital news archive. In *Proceedings of IEEE International Workshop on Content-based Access of Image and Video Database*, pages 52-60, 1998.

[37] M.K. Shan and S.Y. Lee. Content-based video retrieval based on similarity of frame sequence. In *International Workshop on Multi-Media Database Management Systems*, pages 90-97, 1998.

[38] K. Shearer, H. Bunke, S. Venkatesh. Video indexing and similarity retrieval by largest common sub-graph detection using decision trees. In *Elsevier Pattern Recognition Letters*, volume 34, pages 1075-1091, 2001.

[39] J.R. Smith, S.F. Chang. Querying by Color Regions using the VisualSEEk Content-Based Visual Query System. In *Intelligent Multimedia Information Retrieval*, IJCAI, 1997.

[40] J. R. Smith, S. Srinivasan, A. Amir, S. Basu, G. Iyengar, C.Y. Lin, M. Naphade, D. Ponceleon, B.L. Tseng. Integrating Features, Models, and Semantics for TREC Video Retrieval. In *Proceedings of NIST TREC-10 Text Retrieval Conference*, pages 240-249, Gaithersburg, Maryland, Nov 2001

[41] Sony Electronics e-Solutions Company LLC, *ImageStation*. http://www.imagestation.com/

[42] M.J. Swain and D.H. Ballard. Color Indexing. *International Journal of Computer Vision*, volume 7, issue 1, pages 11-32, 1991.

[43] S. Uchihashi, J. Foote, A. Girgensohn, J. Boreczky. Video Manga: Generating Semantically Meaningful Video Summaries. In *Proceedings of the ACM Multimedia 1999*, pages 383-392, 1999.

[44] H.D. Wactlar. New Directions in Video Information Extraction and Summarization. In *Proceedings of the 10th DELOS Workshop, Santorini, Greece*, June 1999.

[45] H.D. Wactlar, T. Kanade, M.A. Smith, S.M. Stevens. Intelligent Access to Digital Video: Informedia Project. *Computer*, volume 29, issue 5, pages 46-52, May 1996.

[46] W3C Recommendation, *Extensible Makeup Language (XML) 1.0 Specification (Second Edition)*. http://www.w3.org/TR/2000/REC-xml-20001006, 6 October 2000.

[47] W3C Recommendation, *Extensible Stylesheet Language (XSL) 1.0 Specification*. http://www.w3.org/TR/2001/REC-xsl-20011015, 15 October, 2001.

[48] W3C Recommendation, *Synchronized Multimedia Integration Language (SMIL 2.0) Specification*. http://www.w3.org/TR/smil20/, 7 August 2001.

[49] W3Schools Online Web Tutorials, *DTD Tutorial*. http://www.w3schools.com/dtd/dtd_elements.asp

[50] H.H. Yu, W. Wolf. A Visual Search System for Video and Image Databases. In *Proceedings of the 1997 International Conference on Multimedia Computing and Systems*, pages 517-524, June 1997.

[51] J. Yu, M.D. Srinath. An efficient method for scene cut detection. In *Elsevier Pattern Recognition Letters*, volume 22, pages 1379-1391, 2001.

[52] D. Zhong, H.J. Zhang, S.F. Chang. Clustering methods for video browsing and annotation. Technical report, Columbia University, 1997.

[53] J.Y. Zhou, E.P. Ong, C.C. Ko. Video object segmentation and tracking for content-based video coding. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1555-1558, 2000.