Content Analysis and Summarization for Video Documents

Lu, Shi

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Department of Computer Science & Engineering

December, 2004

Content Analysis and Summarization for Video Documents

submitted by

Lu, Shi

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Video is increasingly becoming the favorite medium for many communication entities for its extraordinary expressive power. With the fast advancement of network bandwidth and high-capacity storage devices, large scale digital video library systems are growing rapidly. However, such an augmenting video repository gives rise to new challenges to both the end users and the content providers. Since it is time consuming to download and browse through the whole contents of the video, browsing and managing the video database can be quite tedious. To solve this problem, video summarization, which engages in providing concise and informative video summaries to help people to browse and manage video files efficiently, has received more and more attention in these years.

Basically there are two kinds of video summaries: *static video story board*, which is composed of a set of salient images extracted or synthesized from the original video, and *dynamic video skimming*, which is a shorter version of the original video made up of several short video clips.

This thesis presents our work on automatic video summarization. First, we present our early work on greedy method based video skim generation approach. Several important features are extracted from the video. Given the distribution of the important features, a greedy selection algorithm is used to select video segments to form a video skimming. A refinement process is employed to increase the coherence of the video skimming.

Then, we propose a graph optimization based video summarization framework. The workflow in this framework consists four major steps: video structure analysis, video scene analysis, graph modeling and optimization. Both a dynamic video skimming and a static video summary are generated as the content preview for a video document.

In order to adapt the high level semantic information of the video, we have designed a semi-automatic video annotation system to help the user to efficiently create content descriptions for each detected video shot. Given the semantic content descriptions of the video shots, we employ the *mutual reinforcement principle* to calculate an importance measure for each video shot, then create video summaries based on the importance measure and shots arrangement patterns.

We have implemented the three proposed video summarization frameworks, provided the functionality to the users and conducted some experiments. The user tests show that the video summaries generated by our framework are able to guarantee both the balanced content coverage and the visual coherence.

Content Analysis and Summarization for Video Documents

submitted by

Lu, Shi

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

論文摘要

隨著現今寬帶網路和大容量記憶體的快速發展，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。由於要瞭解視頻的內容，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。 對於最終用戶而言，在互聯網上的大量視頻來源中，很難逐一去根據內容來找到所想要的視頻。對於視頻資料庫的管理者而言，管理海量的視頻資料也是一件不方便的工作。爲了解決以上問題，近年來出現了視頻摘要技術，其目的是從視頻中摘取片斷組成簡略摘要，供用戶快速流覽以節省流覽時間。視頻摘要有靜態和動態兩種形式，靜態摘要由一系列靜止圖像組成，而動態摘要則是由一系列動態視頻片斷組成。

我們在本論文中描述我們在視頻摘要方面所做的工作。首先我們提出一種基於視頻特徵的摘要演算法。若干種重要特徵被提取出來，並用貪心演算法根據其分佈選出視頻摘要片斷。我們還使用了一個平滑過程來提高所得到的摘要的平滑度。

我們還提出了一個基於視頻結構資訊和圖優化的視頻摘要方案。首先我們分析待摘要視頻的結構資訊，基於此資訊，我們建立一個圖來描述每個段落中的鏡頭之間的視覺相似度和時序關係。最終的摘要鏡頭由在此圖上搜索一條最長路徑而得到。動態和靜態摘要均能由此方案生成。

爲了進一步提高摘要的質量，我們提出了一種基於語義的摘要方案。我們設計並實現了一個半自動的視頻內容標注系統來幫助用戶有效地標注視頻。基於得到的語義描述，我們使用互支援機制得到視頻鏡頭的重要性度量。基於此度量，我們可以得到新的視頻摘要。

我們實現了以上提出的幾種視頻摘要方案並且進行了一系列試驗以測試其效果。從

最終用戶的反饋可以看出我們的摘要方法的確可以兼顧視頻內容和摘要的平滑度，可以起到幫助用戶快速瞭解視頻內容的功用。

Acknowledgment

I would like to take this opportunity to express my gratitude to my supervisors, Prof. Irwin King and Prof. Michael R. Lyu, for their patient guidance, ceaseless support and encouragement during my M.Phil study. The inspiring advice and insightful criticism from Prof. King and Prof. Lyu are extremely essential and valuable in my research papers and my thesis. This dissertation could not be completed without their effort.

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

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 in the department of computer science, they helped me in solving technical problems, enlightened me with new research ideas, helped me to finish the user experiments and gave me encouragement. It is they that have made my two years' research time joyful and wonderful.

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.

Last but not least, I would like to thank all the scientists and engineers for their endeavor and contributions upon which I can ground my research work. All my work done is based on the fact that I can stand on their shoulders.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and objectives

Video enriches the content delivery by combining visual, audio, and textual information in multiple data streams. Thus it is always the favorite medium for most people and communication entities for its extraordinary expressive power. With the fast development of the network bandwidth and large-capacity storage devices, video data has become pervasive on the Internet in these days. On today's network, many companies provide video sharing services, which further speeds up the growth of the volume of Internet videos. People can retrieve and enjoy multimedia information in the form of text, image and particularly video. Moreover, individual people also begin to share their own edited videos. In 2000, a survey by PC Data showed that an estimated 57.2% of Internet users watch online video clips, and 7.3% of them edited video clips on their personal computers [11]. In 2003, another survey by Broadband4Britain.com [5] show that 34% of British broadband user in often enjoy video-on-demand service.

The evident growing video database thus gives rise to new challenges to both the content providers and the end-users: for users, browsing through the large video repository for the interesting content in a tedious work; for content providers, they are in urgent need of solutions to help them managing the large video database efficiently.

To solve this problem, video summarization, which engage in providing concise and informative video summaries to help people to browse and manage video files efficiently, have received more and more attention in these years.

Basically there are two kinds of video summaries: *static video story board*, which is composed of a set of salient images extracted or synthesized from the original video, and *dynamic video skimming*, which is a shorter version of the original video made up of several short video clips.

1.2 Our contributions

In this thesis we present our research work done on generating video summaries. Our work has the following contributions:

1. **Specific target goals**–We have proposed several targets that a video summary with good quality should be able to achieve, and we have achieve them in our framework.

2. **Video structure utilization**–We analyze the intrinsic shot-group-scene structure of the video and employ the structure information to help us in video skimming generation. By incorporating the video structural information, the balanced content coverage is thus guaranteed.

3. **Graph based skim generation**–We model each of the video scenes into a graph based on the video shots contained in it, and select video shots to from video summaries by searching a constrained longest path in that graph. Both the static storyboard and dynamic skimming are generated from the framework. Our graph based method is able to cover both the visual content coverage and the temporal content distribution simultaneously.

4. **Video content annotation and semantic video summarization**–We propose a semi-automatic semantic video description framework to

help the user to annotate the video shots efficiently. Given the video semantic content description, an importance measure is derived by *mutual reinforcement principle* and video summaries are thus generated upon this importance measure. The performance of our new work is compared with our previous work.

1.3 Thesis outline

The thesis is organized as follows:

Chapter 2 reviews some technology related to our research work emerged in recent years. First, we describe the development of today's large scale digital video libraries. Then both the static video summary generation and the dynamic video skimming generation techniques will be reviewed. We also review the work done on intrinsic video structure analysis, which provides important information that help us to generate meaningful video summaries.

Chapter 3 describes our early video skimming generation method based on video feature detection and greedy method;

Chapter 4 describes our video structure analysis procedure. We illustrate how we process the raw video data and recover the shot-group-scene structure information. After the video scene boundaries have been determined, video shot arrangement pattern strings are analyzed and recorded as candidate elements for video skim. The structural information and the shot arrangement pattern strings are stored for further video summary generation.

Chapter 5 describes our graph-optimization based video summary generation method. To generate a dynamic video skimming, first, we analyze the content complexity and distribute the total skimming length to each of the scenes; second, each detected video scene is modelled into a graph, then the representative video shots are selected by solving a constrained-longest path problem on that graph. Both the generation of moving image video skim and the static video storyboard will be discussed. Experiments show that our video skimming selection scheme does generate meaningful video summaries to the users.

Chapter 6 describes our effort engaged in video content annotation and semantic video summarization. We have developed a semi-automatic system to assist the user to make semantic annotation to video shots efficiently. Given the semantic content description of the video content, we use *mutual reinforcement principle* to derive an importance measure for each annotated video shot. Similarly with previous chapter, we employ video arrangement pattern strings as the video skim candidates. Finally, the most important video shot strings are selected as the final video skimming. The performance of the new method is compared with our previous work done upon low-level features.

Finally, Chapter 7 concludes the whole thesis and discusses our future work.

Chapter 2

Related work

Video summarization engages in providing concise and informative video summaries in order to help people browsing and managing video files more efficiently. As an valuable tool, video summarization has received more and more attention in recent years. Basically here are two fundamentally different kinds of video summaries: *static-image summary* and *moving-image skimming*. The static-image summary, also known as a static storyboard, is a small collection of salient images extracted or generated from the underlying video source. In this report, we call this type of summary a video summary. The moving-image abstract, also known as moving skim, or multimedia summary, consists of a collection of video clips, as well as the corresponding audio segments extracted from the original sequence and is thus itself a shorter version of the original video. In this report, we call this type of abstract a video skimming.

In this chapter, we review some recent technologies related to static video summary and dynamic video skimming generation.

2.1 Static video summary

A static video summary is a collection of images that can represent the underlying video content. In recent years, muck work has been done on static video summary generation. According to the method used to extract representative images, we can classify static video summary methods into sampling-based method, shot-based, motion based and mosaic based methods.

Early work selects video key frames by random or uniform sampling the video image sequence, like the MiniVideo system [66] and Video Magnifier [41]. Their methods are simple, however, since they do not analyze the contents of the video at all, it is unable to guarantee that the important contents can be covered by the selected key frames.

Video shots are the building blocks of edited videos. Video shots transitions include hard cut, wipe, and fade in/outs. Video shot transition detection has been suggested in various work [16, 17, 83, 82, 48, 56, 13, 30, 6, 33, 4]. Since a shot is a image sequence captured continuously, its visual content can be represented by its first frame. In order to cover the content of the dynamic video shots, in the work reported in [83], the key frames are extracted in a sequential fashion for each shot. Particularly, the first frame within the shot is always chosen, then the color-histogram difference between the subsequent frames and the latest key frame is computed. Once the difference exceeds a certain threshold, a new key frame will be extracted. A similar work is also reported in [78]. In [54], a design pattern based video analysis method is proposed to discover high structure information. In [85], the authors propose to extract the key frames using an unsupervised clustering scheme. Basically, all video frames within a shot are first clustered into certain number of clusters based on the color-histogram similarity comparison where a threshold is predefined to control the density of each cluster. Next, all the clusters that are big enough are considered as the key clusters, and a representative frame closest to the cluster centroid is extracted from each of them. Color based algorithms are robust to background noises and motion, but their performance highly depends on particular threshold selection.

When the camera motion can be detected, a mosaic image can be constructed to represent the whole contents of a dynamic video shot with camera motion (pan, tilt, zoom, translate) [29, 75, 40, 2]. Although this approach is quite informative to represent a dynamic shot, it still has several drawbacks. First, it only provides an extended panoramic spatial view of the entire static background scene, but contains no information about the moving object in the foreground. Moreover, recent mosaic generation methods are not robust enough. In the situation that the background-foreground of the scene is changing frequently and the camera operation is quite complex. current algorithm tend to achieve poor performance.

Based on the temporal dynamics of the video by motion analysis, various techniques to control the selected number of key frames have been addressed [70, 68, 69]. Most of them are based on image pixel difference [28] or optical flow [74, 34]. In [27] motion and human gesture analysis are employed to create summaries for video taped presentation files.

Normal edited videos comprise quite a lot video shots. Directly presenting a large amount of video shot images to the user might not be a good idea, and later work concentrate on organizing the shot images by analyzing the intrinsic video structure. Since edited videos depicts a story just like am article, they also have a similar structure just like articles do. In [53], the content of a video is represented in a tree structured manner: from top to down, a video consists several scenes, each of which is like a paragraph in an article; each scene is composed by several semantically-related video shot groups; each video shot group is composed by several visually similar and temporally adjacent video shots. The tree structure is presented to the user as an abstraction of the video content.High level scene structure based on shot similarity has been also addressed in [22]. [77] construct a scene transition graph (STG) for a video by time constrained clustering on the video shots. In the scene transition graph, each video shot cluster is represented by one node in the graph, and the transitions between nodes can reflect the structure of the video. Spectral graph clustering has been proposed in [57] for image region detection, and in [49, 47], the method is used to cluster the video shots detected from a video and higher structure recovery. In [72], a comic book style video summary is generated such that the size of selected images are adjusted according to their importance. The video structure reflects how the editor choose and arrange video shots, they are very valuable information for video summarization. Hierarchical representation of video contents has been proposed in [58], in which a hierarchical video content representation are built, and a tree style video summary is presented, with the level of detail is adjustable. A fuzzy logic based video shot grouping algorithm is proposed in [18]. In ShotWeave system [67], various cues between video shots have been analyzed for better video structure analysis result. A computational scene model combining audio and visual cues and film syntax is proposed in [62].

Some other work on static video summary generation integrates different mathematical methodologies. A EM algorithm based approach is suggested in [73]. In [12], the video are modelled into a curve in high-dimensional feature space, and a set of representative images are extracted by curve simplification and approximation. An approach based on singular value decomposition is proposed in [20], and user log analysis are employed to refine the video skimming [79]. A genetic algorithm based technique is proposed in [9]. A CPR model is proposed in [15]. A variation of classic K-mean clustering algorithm is applied to generate content plots for videos in [55].

2.2 Dynamic video skimming

Compared with static video summary, dynamic video skimming preserves the dynamic properties of the original video, thus is more attractive and helpful to the user. Moreover, the audio information is also preserved thus the video skimming is able to make more senses.

The moving video skimming can be classified into two types: *Overview* and

Highlight. Facing a new movie, mostly the user is totally unaware about the content thus can only specify a target length thus hope to see enough detail about the movie. The request may be like “Give me 3 minutes of preview showing that this movie is about”, and we call this kind of video skimming “overview”. But for specific domain like sports video and news, the user already knows some domain specific knowledge and he may just request those video shots that he is interested in. In this situation, the request many be like ”give me 3 minutes of video about goals and corner kicks”. This kind of video skimming is called “highlight”.

For overview generation, early attempts includes the VAbstract [50] and MOCA project [50, 30]. In VAbstract system, the most characteristic video segments are extracted to form a movie trailer. The frames with high-contrast are detected as the parts containing important contents; the frames having largest frame differences are extracted as the high-action parts; also, to preserve the basic mood of the original movie, the scenes that have a color composition similar to the average color composition of the entire movie, are included in the skimming; moreover, the recognition of dialog scenes is performed by detecting the spectrum of a spoken “a” since “a” occurs frequently in most languages. Finally all selected scenes (except the last part of the movie), organized in their original temporal order, forms the movie trailer. There are some interesting ideas in this paper, but some parts of the algorithm are too simple to be effective and will need lots of improvement. The researchers also lack thorough user studies to support their conclusions. In the MoCA project, which is an improved version of VAbstract, special events such as closed-up shots of leading actors, explosions and gunfire, are detected to help determine the important scenes.

Another straightforward approach to save the video viewing time is by speeding up the playback speed of the original video. As studied by Omoigui, et al. at Microsoft Research, the entire video could be watched in a shorter amount of time by fast playback with almost no pitch distortion using the time compression technology. CueVideo system provides faster playback speed when playing long, static video scenes and slower speed for short, dynamic video scenes [51]. Although the playback time has been reduced but the temporal property of the original video is distorted, which may mislead the users about the nature of the original video. Similar work is also reported by Amir, et al. by using audio time scale modification technology [51]. These techniques, however, only allow a maximum time compression of 1.5-2.5 depending on the speech speed [25], else the speech will become incomprehensible.

The Informedia project aims to create a very short synopsis of the original video by extracting the significant audio and video information [59, 23, 60]. Particularly, text keywords are first extracted from manual transcript and closed captioning by using the well-known *tf-idf* (Term-Frequency- Inverse Document Frequency) technique, then the audio skimming is created by extracting the audio segments corresponding to the selected keywords as well as including some of their neighboring segments for better comprehension. Next, the image skimming is created by selecting the video frames which are: (a) frames with faces or texts; (b) static frames following camera motion; (c) frames with camera motion and human faces or text, and (d) frames at the beginning of a video scene, with a descending priority. As a result, a set of video frames, which may not align with the audio in time, but may be more appropriate for image skimming in visual aspect are extracted. Finally the video skimming is generated by analyzing the word relevance and the structure of the prioritized audio and image skimming. Experiments of this skimming approach have shown impressive results on limited types of documentary video that have very explicit speech or text contents. However, satisfying results may not be achievable using such a text-driven approach on other videos with a soundtrack containing more complex audio contents. Some improvements of their algorithms have been made and a subjective evaluation of this project is reported in [60].

Later work on video overview generation explore and employ multiple features. In [39], the authors tries to model and simulate the attention of the video viewer. To simulate human reaction, first, multiple features like contrast, motion, audio volume, caption text are extracted respectively, then for each feature, a user attention function is defined and calculated. Finally, the net user attention function is obtained as the weighted summation of the user attention functions. Given the length limit, a moving video skimming is extracted by maximizing the attention value summation on the skimming parts. Simultaneously, a static image presentation is extracted according to the local maximum points of the user attention curve. However, humans understand the video not only by perceiving the low level features, but also by the high level semantic features, and the paper does not address how to determine the weights for each of the user attention values. The video structure information is also neglected.

In [64, 63, 8, 61], the authors discussed the issue of determining movie scenes based on audio and visual cues and film syntax. Later, a utility framework for video skimming generation is proposed in [65]. In the framework, the video is first segmented into continuous video shots, and a utility function is calculated for each video shot based on the scene complexity of the shot and other cues. Finally, the video skimming is selected by maximizing the utility summation. In [19], a entropy based visual content redundancy function is defined, and the video summary is extracted based on doing optimization on that function. In [38, 37], video shot arrangement patterns are extracted and utilized to increase the coherence of the video skimming.

For video highlight generation, most work tries to find solution for domain-specific videos where some special features can be used, like sports videos, news, and presentation videos. For sports video, most work are based on event detection, like in soccer video [52, 31], baseball video [7, 26], and basketball video [45]. For news, a question answering system is proposed in [76]. Highlight video skimming for presentation is proposed in [24].

Most of the traditional video skimming generation approaches are based on low level video features, and they may not be able to guarantee that the generated video skim contains the semantically important contents thus the video skim may not make sense to the users. To attack this problem, semantic information is needed to make a meaningful video skimming. Unfortunately, although quite a lot attempts have been done to automatically annotate generic video and image contents [44, 32] and event detection in specific video categories like sports video [3], recognition of high level semantic information like key actors, action taken is still beyond the capacity of present techniques. To collect reliable video semantic information we still need to manually annotate the video contents. Video summarization based on semantic annotation can be found in [84, 71, 38].

2.3 Summary

Video summarization is practically an inseparable research area for many video applications including video indexing, browsing and retrieval. A concisely and intelligently generated video summary will not only enable a more informative interaction between human and computer during the video browsing, but also help to build more meaningful and quicker video indexing and retrieval systems. Recently, video summarization has been attracting considerable research interest, and it is gradually coming to play an important role in the multimedia database area.

For most up-to-date video summarization techniques, most of them is based on some specific features as heuristics for video skim shots selection, and the video structure information is not fully exploited. Moreover, for dynamic video skim generation, most of the suggested solutions concentrate on only "informativeness", while few work has considered the quality goals that a video summary should achieve.

Chapter 3

Greedy method based skim generation

In this chapter we describe our early work done for video skimming generation. To generate a perfect summarization of a given video requires good understanding of the video semantic content. However, understanding the semantic content of the video is still far beyond the capability of today's intelligent systems, despite the significant advances in computer vision, image understanding, and pattern recognition algorithms. So, we can only rely on extracting and analyzing some low-level features to generate video summaries.

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. To resolve this we propose an adaptive merging process to merge close segments so that the coherency of the video skimming can be improved.

We first give some term definitions for this chapter:

1. A video is defined as a sequence of images. It is represented by $V = (I_1, I_2 ... I_n)$, where I_i is the *i-th* indexed frame image of the video and n is the number of total frames in the video V. The video can also be represented in the short form as $V = [I_{begin}, I_{end}]$ where $1 \leqslant begin \leqslant end \leqslant n$. Moreover, the length of a video, $length(V)$ is $end - begin + 1$.

2. The feature score function f_i of a frame is a function over the whole frame image set I. Without loss of generality, we assume these functions to be non-negative. It can be discrete, which indicates the occurrence of a

feature of interest like human face; or it can be continuous to describe the magnitude of the feature like noise volume, percentage of the interesting color, etc.

3. An extracted video is defined as a mapping of a video with a set of feature functions to a set of video clips through a video feature extraction algorithm. This can be written as $h : V \times F \rightarrow \{V_i\}$, where F is a nonempty set of feature functions and V is a set of zero or more of valid video clips. If some items in $\{V_i\}$ are overlapping, i.e., $V_i \bigcap V_j \neq \emptyset, i \neq j$, we call this the overlapping extracted video. Otherwise, we call it the non-overlapping extracted video. One may assume that the items in the set are sorted in a non-descending order by the I_{begin} variable. We define the length of an extracted video to be the summation of all the elements' length in the extracted video.

The overview of our feature based video summarization generation framework is shown in Figure. 3.1.

3.1 Selected video features for video 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

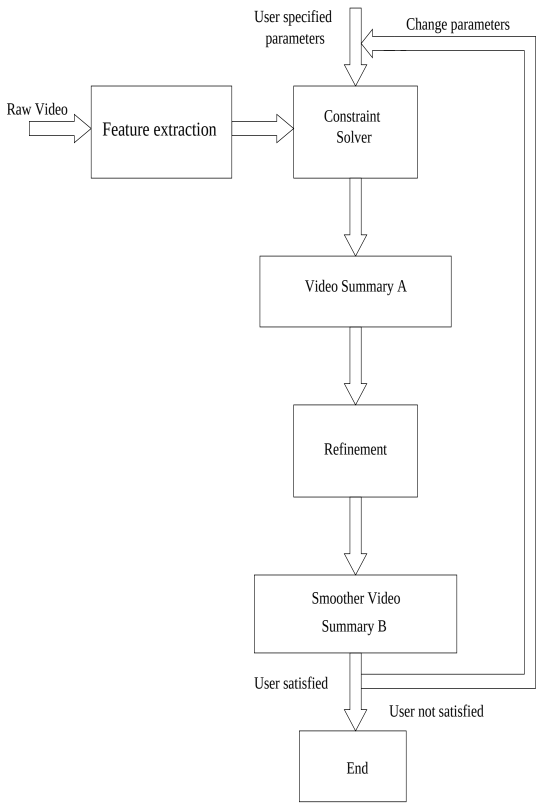


Figure 3.1: Flowchart of the framework

information about the video content and serve a good indication of important video content that should be put into the video summary.

3.2 Video summarization problem

After we choose a set of feature score distribution functions on the frame set of the original video, we aim to get the final video summary that mostly represents the content of the original video. To achieve this, we can select to maximize the feature score summation of the video summary. Then the problem can be formalized as described in Problem 1:

Problem 1 Given a set of video features, a time threshold T, and $\{V_i\}$ obtained from a set of video feature extraction functions, find the final video summary, $\{V_{final}\}$, an non-overlapping extracted video, such that it maximizes the total feature score summation and $length(\{V_{final}\}) = T$.

From the users' point of view, a smoother video summary may seem better than a jumpy one. The more broken segments the video summary contains, the more jumpy the video summary will be. The solution is to decrease the number of video segments in the video summary so that the summary may look more coherent and smoother. We thus define the transition of an extracted video to describe the smoothness of the extracted video summary as follows:

Definition 2 A transition in the non-overlapping extracted video is defined as the cardinality of the video as $\#(\{V_i\})$. It is the number of segments in $\{V_i\}$.

With the above definition we can extend Problem 1 as follows.

Problem 3 Given a set of video features, $\{V_i\}$ obtained from the a set of video feature extraction functions, find the final video summary, $\{V_{final}\}$, an non-overlapping extracted video such that $length(\{V_{final}\}) = T$, moreover, it maximizes the total feature score and minimizes the transition number.

To solve Problem 1 and Problem 3, we need to solve the constraints. We may solve the constraints with various methods like greedy method, lagrange multiplier, dynamic programming, etc. Here we use a straight-forward greedy method to solve Problems 1. Since we use the summation of the feature score as the importance measure of each frame, we first calculate the feature score summation value for each video frame, then sort them according to their feature score values, and select the T frames with highest scores. The time complexity for the greedy method is $O(n \log(n))$, which is the complexity of the sorting process. After the greedy method, we get an extracted video set $\{V_i\}$ as the video summary.

We can solve Problem 3 by refining the video summary $\{V_i\}$ that we get by the greedy method. To minimize the transition number in video summary $\{V_i\}$, we need to decrease the granularity of the selected segments to make it look smoother while preserve most of the information contained in the previous video summary. We perform a segment merging process to minimize the number of the transitions in the extracted video summary. Each time we find the two nearest selected video segments and join them to form a longer and more coherent segment until the extracted video is smooth enough. Let $\{E_i\}$ be the set of unselected segments, we introduce a penalty function to measure the granularity of the $\{V_i\}$ as,

$$p(\{V_i\}, \{E_i\}) = \sum_j \frac{\sum_t length(E_t)}{length(E_j)}. \tag{3.1}$$

Function g can be used to adjust to what extent we will reduce the transition number. Consequently, we can derive the new importance function as

$$N(\{V_i\}, \{E_i\}) = f(\{v_i\}) - w \cdot p(\{V_i\}, \{E_i\}) \tag{3.2}$$

where $f()$ is the feature score summation function for the selected segments $\{V_i\}$, $p()$ is the penalty function, and w is the weight of the penalty function. Our goal is to maximize the function N.

To find the maximum value of function N, we propose a search algorithm based on the result segments generated by the greedy method as described below:

Note that during the 4th step in the repeat loop, we merge the two adjacent selected segments in the way that we move the segment with smaller feature score values, so that the loss of feature score is minimized in this step.

When we continue merging the unselected segments, the penalty function p, which is proportional to the summation of all unselected segments' reciprocal, changes with the reciprocal of the unselected segment's length. As the merge process continues, the length of the shortest unselected segments becomes longer and its reciprocal becomes smaller. Thus function p decreases

Algorithm 1 Video summary refinement algorithm

```
Find the result segments $\{E_i\}, \{V_i\}$ with the greedy method;
**repeat**
    1. $TEMPTVALUE = N(\{E_i\}, \{V_i\})$;
    2. Find the shortest unselected segment $e_i$;
    3. Find the adjacent selected segment $s_1, s_2$ of $e_i$;
    4. Merge $s_1$ and $s_2$, so the shortest unselected segment $e_i$ is eliminated.
    Now we get updated $\{E_i\}$,$\{V_i\}$.
    5. Calculate $N(\{E_i\}, \{V_i\})$ for updated $\{E_i\}$,$\{V_i\}$.
**until** $N(\{E_i\}, \{V_i\}) < TEMPTVALUE$
Undo the last merge;
The final smoother video summary is found.
```

quickly with the merge process and the refinement process soon converges. The complexity of the merge process would not be greater than $O(n \log(n))$. Including the refinement process, we can still regard the complexity of the whole process as $O(n \log(n))$.

3.3 Experiments

To test the performance of our video summarization and the refinement process, we implemented the proposed algorithm and applied it on some video clips and observe its performance. Our experiment consists of a subjective test and a quantitative test. In the experiment, all tested video clips were in Mpeg-2 coded AVI format. We employed a PC platform with 2.0G hz P4 CPU on the Win2000 OS. We developed a system to perform the summarization procedure and played the video summary for user assessment. We will briefly describe the system in the next section.

We selected two types of video clips for our summarization experiment. The first set was news video clips, and the other set was video clips grabbed from movies. We select these two types of videos because they are quite pervasive and representative. However, their structures are rather different. Our test includes ten news video clips that were about 1 to 2 minutes long, and three movie clips that were about 7 to 10 minutes long. We show the experimental results for one sample video from each type of the videos.

For the news video clips, we selected the following features for video summarization: human face detection, male/female voices occurrence, camera zooming and caption text occurrence. Table 3.1 shows the selected features and the corresponding parameters specified by when we engaged summarization to one of the video clips. Note that the weights here are normalized so that $\sum_i Weight_i = 1$.

Table 3.1: Parameters for news clips

Features	**Weight**
f_1:Face occurrence	0.30
f_2:Text occurrence	0.20
f_3:Voice	0.20
f_4:Camera zooming	0.30

Parameters	**Value**
Original length	95 sec
Summary length	22 sec
w	20.0

The feature distribution and the finally generated segments for a sample news video are shown in Fig. 3.2. The horizontal axes denote the time.

From Fig. 3.2 we can see that the feature distribution and the selected segments are all quite long and coherent, so that no refinement process is needed. This is because the news video itself is well structured and composed with longer video shots. We played the summary and observed that it did contain the major content of the news story. Tests on most other news video clips yielded similar results.

For the movie clips, we selected the following features for video summarization generation: human face occurrence, loud human voices/cries, loud noises like gun shots and explosion, and the color of fire. Table 3.2 shows the selected features and the corresponding parameters specified in the experiment.

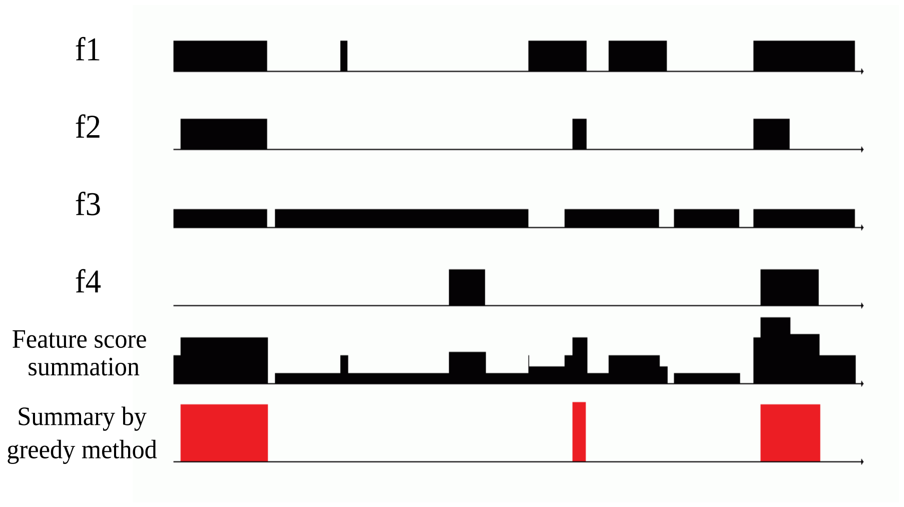


Figure 3.2: Result for news video

Table 3.2: Parameters for movie clip

Features	**Weight**
f_1:Gunshot/explode noise	0.45
f_2:loud voice	0.25
f_3:Face occurrence	0.10
f_4:Fire Color	0.20
Parameters	**Value**
Original length	477 sec
Summary length	50 sec
w	20.0

The feature distribution and the finally generated segments are shown in Fig. 3.3.

From Figure. 3.3 we can see that in the movie, shot and feature changes happen much more frequently than in the news video, especially for the action movies. The result of the greedy method contains many broken short segments, which causes jumpy scenes to the user. Consequently, in this case, the refinement process is needed for more satisfying results. The refinement process generated several longer and more coherent video segments from the

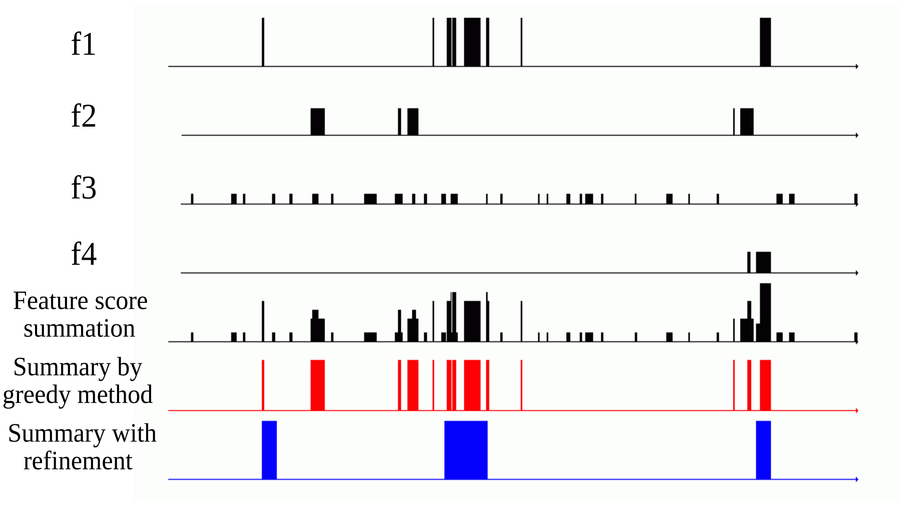


Figure 3.3: Result for movie clip

short fragments yielded by the greedy method. From Figure. 3.3 we can see that those refined longer segments still cover most of the parts that obtained the highest feature score values in the feature summation distribution graph. Moreover, they matched the segments clusters generated by the greedy method quite well , and not much information in the original summary was lost while the summary has become smoother. We played the summary and observed that it still covered the major key events in the movie and preserved the genre of the original movie.

To demonstrate the capability of our framework to generate multiple video summaries, we change the weight for some specific features, for example, if we prefer more on human faces and we increase the weight for human face occurrence, then more contents with human face will be selected into the video summary. Table 3.3 shows the effects resulted from changing the weight for the ”face occurrence” feature in the video summarization procedure. We can see that if we adjust the feature weight for the face occurrence to a higher level, the result summary will select more content with face occurrence.

Table 3.3: Effects by changing the weight for face occurrence

Weight	0.2	0.4	0.6	0.8
Rate	0.24	0.32	0.54	0.88

3.4 Summary

Video summarization is a powerful tool for video browsing and management. In this chapter we have proposed a feature-based greedy method solution to extract the perceptional important video segments to form moving video summaries. A refinement process is proposed to make a smoother summary. We also conducted some experiments to evaluate our proposed algorithms. The initial experimental results were encouraging.

The work in this chapter can be further extended in the following aspects:

1. **Video structure and style analysis**– Video structure and style analysis may also provide us with some semantical information to refine our summarization results. For different types of videos, we may build different summarization profiles for them according to their unique styles.

2. **More features and more constraints**– The suggested problem solution framework can be extended by adding informative features, especially some high-level semantic features and video structure and style, or by extending the object function with some new importance measures on the selected video segments. The refinement process can also be further improved. Besides the greedy method, we will try other constraint-solving methods to solve the constraints then get the final video summary.

3. **Minimal summarization limit**– As the video summarization length T decreases, the quality of the video summary becomes worse. Therefore, for a specified video file, there may exist a lower bound time limit for a meaningful video summary, which is determined by the content and

structure of the video. Finding the lower bound of the video summary length T_{min} within which we can still ensure the quality of the video summary is another problem left for us for further study.

We will discuss the further work on video summarization in the following chapters.

Chapter 4

Video structure analysis

A video narrates a story just like an article does, similarly, a video also have intrinsic structure just like paragraph, sentence in articles. From top to down, a video is composed by several video scenes, each of them depicts an event just like a paragraph does in the article. A video scene is composed by a series of semantically related video shots. A video shot's role is like the sentence in the articles. Between video scenes and video shots, temporally adjacent video shots can be grouped into video shot groups.

Video structural information is very important to generating a video summary with satisfying content coverage.

In this chapter, we discuss how we analyze and explore the structural information from the raw video data. We build up the video structure in a bottom-up manner. First, we decompose the video into continuous video shots, then we group visually similar and temporal adjacent video shots into video shot groups. Based on the video shot groups we can construct video scenes, so that a four-layered video structure is constructed.

Here we give definition for some terms we use in this section. In this thesis, we followed the term definitions in [53]. The structure of a video is defined as a 4-level hierarchical tree structure, from top to down, it consists of video level, video scene level, video shot group level, and video shot level, as shown in Figure 6.3.

1. **Video**: A video is defined as a image sequence $\{f_1......f_n\}$. The video contains all the elements listed below.

2. **Video shot**: A video shot sh_i is defined as a continuous image sequence uninterruptedly captured by a camera. The images frames are $\{f_{i_{begin}}...f_{i_{end}}\}$, the length l_{sh_i} of video shot sh_i is the number of image frames it contains. Video shots are the building blocks of edited videos. We can employ the begin and end frames of a video shot as the key frames to represent its visual content.

3. **Key frame**: The visual content of a video shot can be represented by its key frames. We use the first frame $f_{i_{begin}}$ and last frame $f_{i_{end}}$ of the video shot sh_i as its key frames.

4. **Video shot group**: A video shot group $Sg_i = \{sh_{i_1}...sh_{i_k}\}$ is composed by visually similar and temporally adjacent video shots. The length of a video shot group l_{sg_i} is the summation of all video shots contained in the video shot group. The length l_{sg_i} of video shot group sg_i is the summation of the video shot length that it contains. There are two properties of the member shots:

   (a) Visual similarity: The visual similarity between each video shot pairs should be larger than the similarity threshold T_{vsim}, described as follows:

   $$vsim(sh_x, sh_y) > T_{vsim} \tag{4.1}$$

   where x, $y \in \{i_1...i_k\}$.

   (b) Temporal adjacency: The temporal distance between consecutive video shots in a video shot group should not be larger than the time threshold $T_{temporal}$, described as follows:

$$d_T(sh_x, sh_{x+1}) > T_{temporal}, x \in \{i_1...i_k - 1\} \quad (4.2)$$

where $d_T(sh_x, sh_y)$ is defined as the distance between middle frame images of video shot sh_x and sh_y, in terms of frame number.

5. **Video scene**: A video scene is the intermediate entity between video shot groups and the whole video. We classify the video scene into two types, *loop scene* and *progressive scene*. Loop scenes are composed by several intersecting video shot groups, while the progressive scenes are composed by series of temporally successive but visually different video shots. A video scene describe an event, or depict the transfer between events. The length l_{sc_i} of video scene sh_i is the summation of the video shot length that it contains.

4.1 Video shot detection

A video shot is an image sequence captured continuously by a single camera. It is the basic building block of edited videos like movies, broadcast news, TV shows, etc. Detecting video shot boundaries is the first step for video content analysis.

Normally there are two kinds of video shot breaks. The first is called *cut*, and the second is called *fade*. A video cut is an instantaneous change from one video shot to another; a video fade transition effect is that when the content change from one video shot to another, the new shots gradually comes out and the old video shot gradually fades out. In this section we describe how we detect cuts and fades from the raw video.

4.1.1 Shot cut detection

Since the video shot is composed of relatively coherent images, we can use some metrics to measure the similarity between consequent image pairs then by some threshold method we can detect the interrupt changes, thus we can detect the cut occurrences. Various video detection techniques have been proposed [17, 82, 1].

To measure the similarity of two images, traditional methods use the frame difference, correlation/intersection of color histogram, etc. Frame difference is easy to compute, however, it is very sensitive to camera motion. To overcome this, a proper offset can be calculated to compensate the motion, but searching for such an offset can be quite time consuming. Another metric is the color histogram. Since the color histogram is derived from the statistics of the original image, it can only describe the composition of the image but does not contain any information about how the image looks, which may lead to some mis-detections. Regional color histogram [46] is also sensitive to camera motion. Moreover, histogram calculation requires that all the pixels in the image frame should be processed, so that the histogram based approach may not be very computationally efficient. Consequently, we use a simple, efficient yet robust video shot detection method, described as follows.

To detect video shot boundaries, we first extract a *video slice image* [48] from the original video, then we detect video transitions by analyzing the patterns in the video slice image. A video slice image is a spacial sampling of the video over the temporal axis, which can be generated by cutting through the video from one position, e.g. the center horizontal line of a frame, the diagonal line of a frame, etc. An example of the video slice image by cutting through the center horizontal line (or put all the center pixel line of the video frames together) is shown in Figure 4.1.

We can choose whatever fixed line on the video image to generate a video

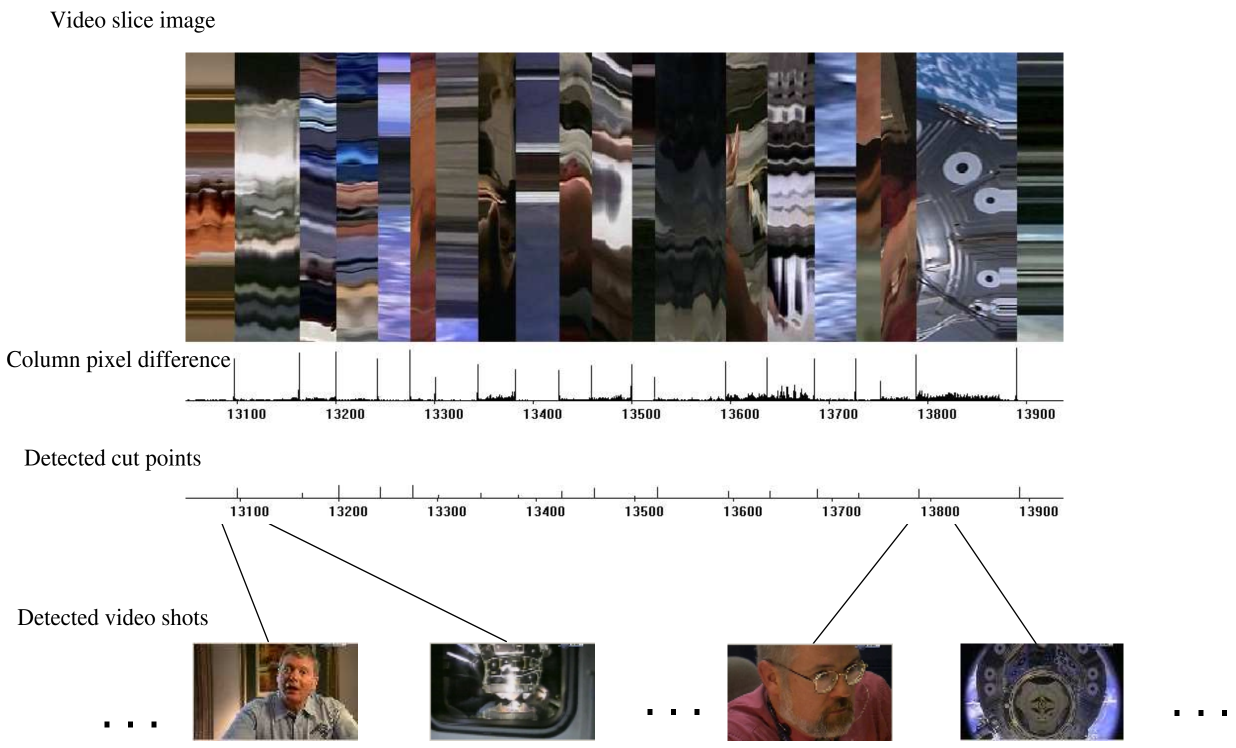


Figure 4.1: Video slice image extracted from a video, accompanied by the pixel difference and the detected shot breaks

slice. Experiments show that simply choosing the center horizontal line of the image to generate the video slice yields satisfying detection result, so in practice we select only one middle slice image from the center horizontal pixel line of the video. Two reasons contributes to the fact that the center horizontal pixel line is good. First, when a video is recorded, the camera normally moves in the horizontal plane, and the horizontal panning happens more frequently than the vertical panning. The second reason is that the camera operator normally places the interesting object in the center of the camera view. So a slice generated by the center horizontal line is good enough for video segmentation in most common conditions.

With the slice image generated, we can measure the similarity of the consequent video image pairs by measuring the similarity of the pixel rows in the slice image. Since only one pixel column is need for processing, the computation cost is highly reduced, with comparison to the traditional histogram based method. We use the *pixel difference* and *column correlation* as the measure for image similarity, described as follows:

Suppose we have pixel columns cl_i and cl_{i+1} which are two image pixel columns in the middle slice image, each of them contains n pixels. The minimum difference between the i_{th} row and the $i+1_{th}$ column is computed as follows:

$$D_{min}(i) = \min_{x=-m}^{m}(\sum_{j=1}^{n}(|cl_i(j) - cl_{i+1}((j+x) \mod n)|)) \tag{4.3}$$

We move the consequent image columns while computing the difference of the two columns, and get the least value of the difference. The reason we use an offset x up to m is for horizontal motion compensation. The computed least difference is shown in Figure 4.1.

If we view each column of pixels as a single vector, we can use the correlation and the angle between them as a new similarity measure. Given pixel column $cl_i = \{p_{i_1}...p_{i_n}\}$ and $cl_j = \{p_{j_1}...p_{j_n}\}$, we can have the correlation between the pixel columns as:

$$corr(cl_i, cl_j) = \frac{\sum_x p_{i_x} \times p_{j_x}}{\sqrt{\sum_y p_{i_y}^2 \times \sum_z p_{j_z}^2}} \tag{4.4}$$

Since the correlation between vectors is just the cosine value of the angle between them, we can also calculate the angle as:

$$\theta_{ij} = \arccos(corr(cl_i, cl_j)) \tag{4.5}$$

And now we have three functions describing the column pairwise dissimilarity in the middle slice image. Based on them, we can detect video shot cuts. We have compared their performance in the experiment section, in which shows that the best measure is the column pixel difference. Moreover, at the same time, it is also the measure which is the easiest to compute among the three measures.

From all the three dissimilarity function we can see that when there is a video shot break, the difference function will experience a sudden jump. Under normal situation without intense motion, by applying a global threshold on the pixel difference, we can find most of those cut points, and the pixel difference seems to be good enough for shot cut detection. However, in case that the motion of the scene is intense, simply applying a global threshold yields a lot of false shot detections. Several optimal threshold selection methods are proposed described in [80], but the performance of them is not quite satisfying.

We notice that there are two factors that jointly identify a video shot break. First, at shot break position the difference function should be a local maximum; Second, the width of the jump peak should be exactly equal to 1. Note that this two criterion are all local features, so that we need to determine the proper threshold locally instead of determining it globally. Based on these two criterion, we apply the following non-linear filter to the pixel difference function to find out such video shot changes:

$$D'_i = \frac{D_i}{\max_{j=-w,j\neq 0}^{w}(D_{i+j})} \tag{4.6}$$

where w is the half width of the window.

The original pixel difference function and the filtered difference function are shown in Figure. 4.2.

After applying this filtering, the transformed value will be more than 1 only at those points that are local maxima in its neighborhood. Thus we can successfully detect most of the local video shot breaks by directly applying a threshold on the filtered difference function. Our method is quite robust with camera and object motion, and the computation cost is quite low.

Another merit of our method is that it is very robust to sudden lightness change, which is mostly caused by the camera flash. In some document videos

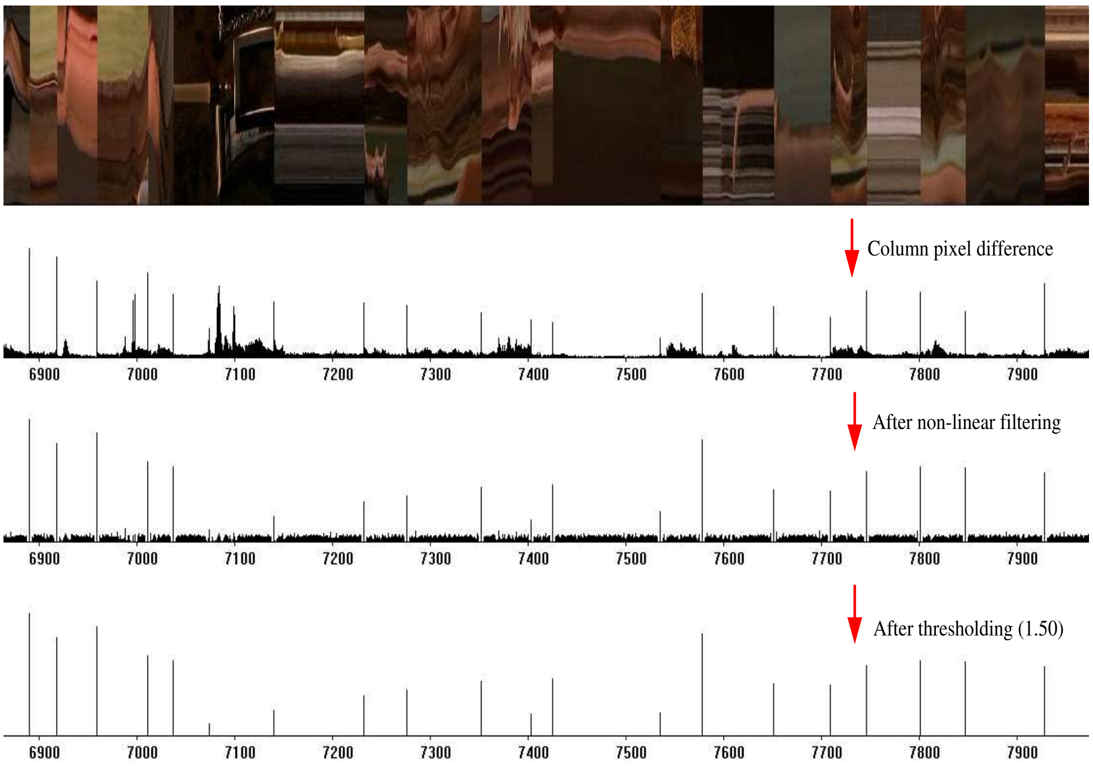


Figure 4.2: Filtering of pixel difference function then detect video shot cuts by thresholding

camera flashes often generate a lot of false positives, the dissimilarity function on those points will have a sudden jump with width equal to 2. However, our non-linear filtering method is very robust toward such sudden lighting change, as the sudden jump which is greater than 1 are all filtered out, shown in figure 4.3.

For video shot sh_i, we use its first frame $kf_{i_{begin}}$ and the last frame $kf_{i_{end}}$ as the key frames to represent the visual content of the video shot.

4.1.2 Fade detection

In document videos, it is common to use fade in/out(dissolve) as the transition between different video shots, for it feels less abrupt than direct shot cut.

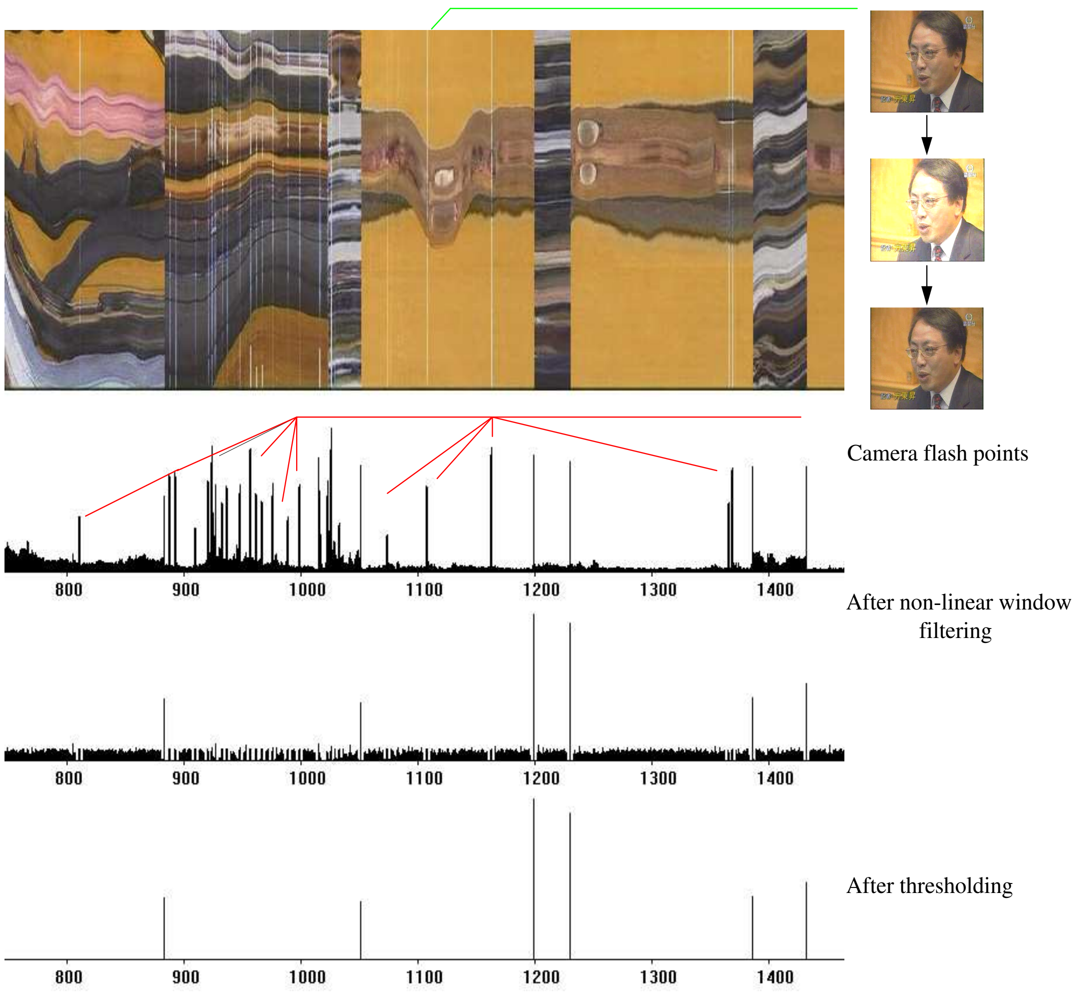


Figure 4.3: Flash effect elimination

However, the cut detection fails to detect fade in/outs for the pixels changes gradually and does not yield apparent jumps in the pixel difference function.

On the middle slice image, we can easily see the fade patterns Figure 4.4. We may regard that the fade in/out area between video shot sh_i and sh_j as the interpolation result of the two intensity function of two video shots. On the middle slice image, suppose that the intensity function of shot sh_i and sh_j are $I_i(t, x)$ and $I_j(t, x)$ respectively, and the fade area starts at t_1 and end at

t_2, we can write the intensity function in the fade area as:

$$fade(t,x) = \frac{t-t_1}{t_2-t_1} \times I_i(t_1,x) + \frac{t_2-t}{t_2-t_1} \times I_j(t_2,x) \tag{4.7}$$

which is the linear interpolation of two intensity functions according to the time. In most cases, the fade in/outs are indeed made in this way. In order to detect such fade transitions, we employ three cues, described as follows:

First, we employ the n-step column difference function on the middle slice image. A n-step column difference function is define as the pixel difference function between two image columns in the middle slice image with column-distance n, described as follows:

$$nd(i) = \sum_j (|cl_i(j) - cl_{i+n}(j)|) \tag{4.8}$$

we can expect that for video fades that the width is less or qual to n (including the cuts), we can see an step function with width n for video shot cuts. However, for video fades, the difference function shape will like an triangle. In most cases, the duration of the fade is less than 1 second, which corresponds to around 30 frames in normal digital videos. Given the cuts detected already, if there is a triangle function but no cuts detected, then very likely there will be a fade/in/out, as shown in figure 4.4.

Given the interpolation function, we can have the variance function that varies from t_1 to t_2, given as follows:

$$var(t,x) = var(\frac{t-t_1}{t_2-t_1} \times I_i(t_1,x) + \frac{t_2-t}{t_2-t_1} \times I_j(t_2,x)) \tag{4.9}$$

$$var(t,x) = var(I_i(t_1,x)) + var(I_j(t_2,x)) + var(I_j(t_2,x)) \times (\frac{t-t_1}{t_2-t_1}^2) - 2\frac{t-t_1}{t_2-t_1} \times var(I_i(t_1,x)) \tag{4.10}$$

So that the variance function on the fade area will become a parabola shape, as shown in figure 4.4. In most cases, the lowest point of the parabola will be at the center of the fade area.

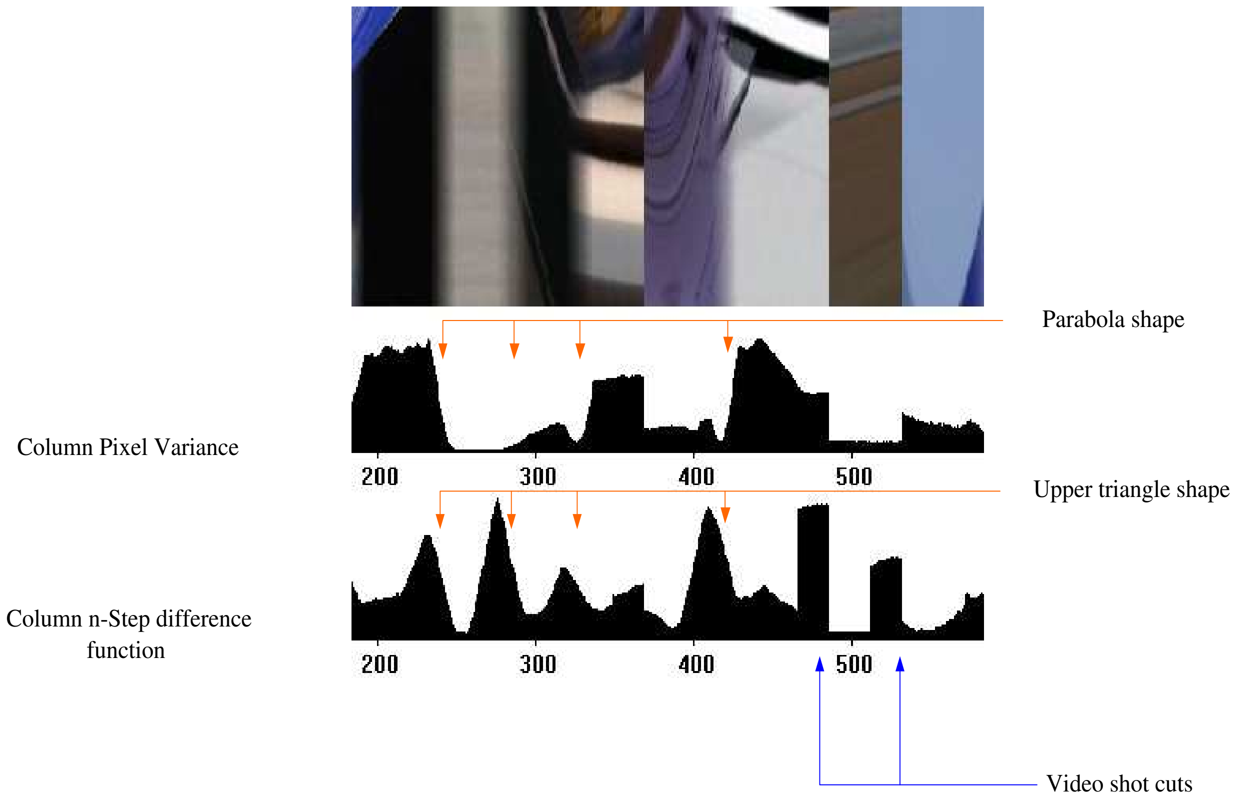


Figure 4.4: Cue features for video fade detection

We combine the two cues as the sign flag for fade occurrence. To detect parabola shapes, we employ a parabola template, and we calculate the unified auto-correlation between the variance function and the template. Peaks are considered as parabolas. The experiments shows that combining the two cues does yield an effective method for video fade detection.

4.2 Video shot group construction

A video shot's role is just like a sentence in articles. The visual content of a video shot can be represented by its key frames. A video shot group Sg_i is the intermediate entity between video scenes and video shots, which is composed of several visually similar and temporally adjacent video shots. Thus from top to down, a video has a 4-level hierarchical structure: Video, Video scenes, Video shot groups, and Video shots [53]. Figure 6.3 shows the hierarchical structure of a video.

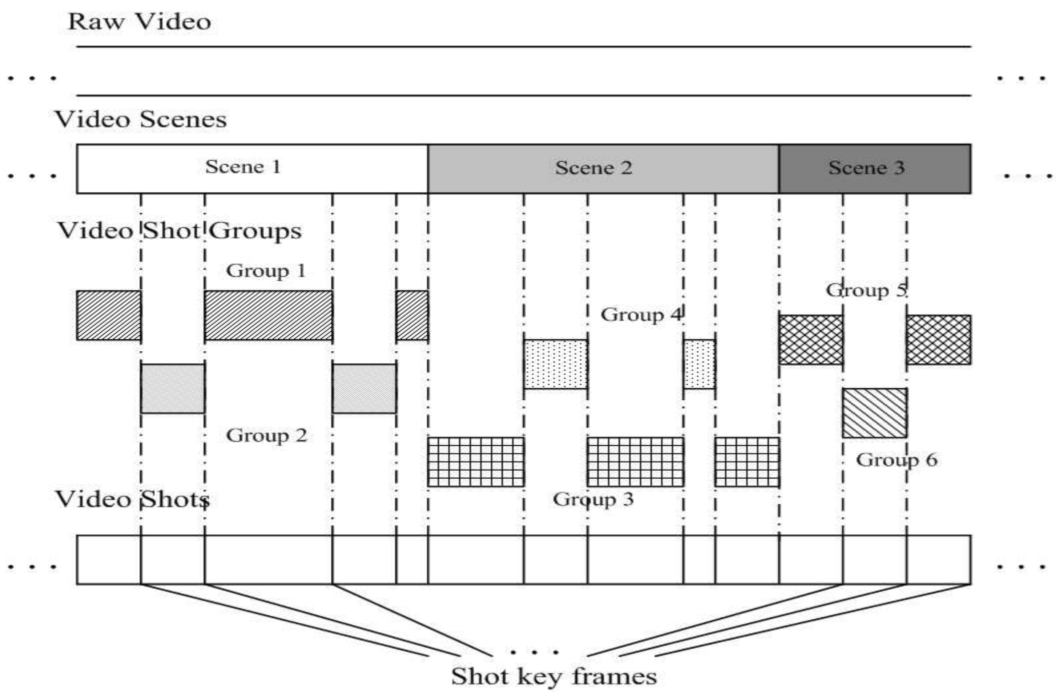


Figure 4.5: Hierarchical video structure

In the remaining part of this paper, we use l_{sh_i}, l_{Sg_j} and l_{Sc_i} to represent the length of video shot sh_i, video shot group Sg_j, and video scene Sc_i, representing the total number of images containing in them respectively.

The structure of a video is built in a bottom-up manner. After we have determined the video shot boundaries, we can continue to build up the hierarchical structure. Visually similar video shots are clustered into video shot groups, and temporal intersected video shot groups form video scenes.

To automatically group the visually similar video shots into video shot groups, many methods have been proposed in the literature, like the time-adaptive shot grouping algorithm [53], hierarchical clustering [77], and spectral graph partitioning [57]. Spectral graph partitioning techniques are known for effective perceptual grouping. It has been used for image segmentation based on pixel proximity and color similarity with good performance. We have implemented both the ToC and the spectral graph clustering algorithms to group the similar video shots. Their performance are compared in the experiment section.

4.2.1 Shot pairwise similarity measure

The similarity between video shot pairs is very crucial for video shot clustering. Normally, for a video shot group, there are two factors that should be quite consistent: first, the video shots should be visually similar; second, the camera motion of the video shots should also be similar. To measure the visual similarity between video shots, In this paper, we first transform the color image into HSV color space, then we employ a *H*-*S* histogram in HSV color space, with 8 bins for *H* channel and 4 bins for *S* channel. The maximal *H*-*S* histogram correlation between shot key frames is used as the visual similarity measure w_{ij}, shown as follows:

$$vsim(sh_i, sh_j) = \max_{x,y} HistCorr(kf_{i_x}, kf_{j_y}) \tag{4.11}$$

where $x, y \in \{begin, end\}$.

To compare the motion intensity, we don't consider the orientation of the motion. Given the image frames $\{f_{i_1}...f_{i_n}\}$ contained in video shot sh_i, the following average motion intensity is calculated for each video shot:

$$mi_{sh_i} = \frac{\sum_{j=i_1}^{i_n-1} |1 - HistCorr(f_{i_j}, f_{i_j+1})|}{n-1} \tag{4.12}$$

and the motion intensity similarity between video shots sh_i and sh_j is defined as:

$$misim_{i,j} = \frac{min(mi_{sh_i}, mi_{sh_j})}{max(mi_{sh_i}, mi_{sh_j})} \tag{4.13}$$

The middle slice image of a video shot can be analyzed to find some motion cues. Panning camera brings about slope line patterns in the middle slice image, while the static camera yields horizontal lines in the middle slice image. In this paper, we only detect and process the camera panning situation. By first doing a pass of edge detection and an edge orientation (gradient) histogram, we can successfully detect camera panning. We then classify the camera panning into three styles: panning left, panning right and no panning. If two video shots have the similar panning style, the panning similarity between them will be 1 and otherwise 0.

Finally we linearly combine the visual similarity and the motion similarity into a similarity measure between video shots, given as follows:

$$sim_{ij} = (1-t_1-t_2)vsim(sh_i, sh_j) + t_1 \times misim(sh_i, sh_j) + t_2 \times msim(sh_i, sh_j) \tag{4.14}$$

where t_1 is the weight for the motion intensity similarity and t_2 is the weight for camera panning similarity.

4.2.2 Video shot grouping by ToC

In [53], the authors proposed the following window-sweeping algorithm to group the visually similar video shots:

The ToC algorithm set up the video shot-group-scene structure with the following time-constrained grouping method. First, two measures are calculated for each video shot:

1. Color histogram of the entire shot key frame image.

2. Shot activity, as the average color difference of the frames contained in the shot.

The similarity between video shot pairs is thus the weighted sum of the similarity between histograms and the shot activity similarity. Their shot similarity measure is quite similar to our similarity in equation 4.14, in the experiments, we employ the similarity measure in equation 4.14 in both cases.

To determine if a video shot sh_i should be put in group $g_j = \{sh_{j_1}...sh_{j_m}\}$, a shot-to-group similarity should be calculated. The shot-to-group similarity is calculated as:

$$sim_{sh_i,g_j} = \max_t sim_{i,j_t} \tag{4.15}$$

which is the maximum similarity or

$$sim_{sh_i,g_j} = \frac{\sum_t sim_{i,j_t}}{m} \tag{4.16}$$

which is the average similarity.

The temporal distance $d_T(sh_i, sh_j)$ between video shots sh_i and sh_j is defined as the temporal distance between their center image frames.

And the temporal distance between video shot sh_i and video shots group g_k is defined as:

$$d_T(sh_i, g_j) = \min_t d_T sh_i, sh_{k_t} \tag{4.17}$$

where $sh_{k_t} \in g_k$

When the shot pairwise similarity is obtained, the ToC algorithm make the video shot groups in the following way, as shown in algorithm:

The performance of ToC algorithm is very significantly determined by the similarity threshold s_{th}. However, since different video shot groups may have different proper similarity thresholds, so that the time-constrained grouping with a hard threshold may not be able to correctly detect all video shot groups and is likely to generate a lot of over-groupings and miss-groups.

Algorithm 2 ToC video shot grouping algorithm

```
Input: The set of video shots $SH = \{sh_1....sh_n\}$, the shot pair similarity $\{sim_{ij}\}$ for all i,j $\in \{1...n\}$;
Output: The set of video shot groups $G = \{g_1....g_m\}$
1. Add the first group $g_1$ to $G$, assign $sh_1$ to $g_1$;
for $sh_i \in SH$ do
  for $g_j \in G$ do
    calculate the shot-to-group similarity $sim_{sh_i,g_j}$;
    if $sim_{sh_i,g_j} \geq s_{th}$ and $d_T(sh_i, g_j) \leq T_t h$ then
      assign $sh_i$ to $g_j$
    else
      add new shot group $g_k$ to $G$, assign $sh_i$ to $g_k$
    end if
  end for
end for
```

4.2.3 Spectral graph partitioning

Spectral graph clustering was introduced in [57], it is a recursive clustering method proposed to segment images into regions based on the color similarity and spacial distance between image pixels. In [57], a completed graph is constructed with the pixels in the image as vertexes and on each edge there is a weight describing the color similarity and spacial distance between image pixels. The graph is then recursively partitioned into smaller clusters, on each partitioning, a global optimal normalized cut is calculated to ensure the performance. Finally the original image is segmented into coherent regions.

The graph partitioning can be migrated from image segmentation to other clustering scenarios, in which if the graph can be built. Our shot clustering problem is one of such cases that can utilize the spectral graph clustering. Given a series of video shots, we can also construct a graph $G(V, E)$, where V is the vertex set, in which each element corresponds to a video shot. E is the edge set, in which the edges connects each shot pair in V. On each edge e_{ij} there is a edge weight w_{ij}, which is a measure of the visual similarity between the two video shots.

Given the graph $G(V, E)$, we may cut the vertex set V into disjointed sets A and B, and compute the Normalized Cut Value to evaluate a cut:

$$NCut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{4.18}$$

where

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij} \tag{4.19}$$

is the cut value and

$$assoc(A, V) = \sum_{i \in A, j \in V} w_{ij} \tag{4.20}$$

is the association of A with the vertex set V, an example figure is shown in Figure 4.6. In the figure, the vertex set V is divided into vertex set A and B. The edges contribute to $cut(A, B)$ is red, the edges contribute to $assoc(A, V)$ is green, while the edges contribute to $assoc(B, V)$ is blue.

The meaning of the cut value is the total similarity summation between partition A and B. And the meaning of the association is in fact the total similarity summation within the vertexes in partition A and B. A good partition would create two self-similar and mutual-dissimilar partitions.

We hope that by cutting the graph into A and B, we can separate the dissimilar node into different parts as many as possible, which means $Cut(A, B)$ should be small; also, we hope that the $assoc(A, V)$ and $assoc(B, V)$ be large for they should contain similar video shots. Thus the optimal goal for the graph cut is like follows:

Given G, the optimal partition for G is the partition that minimize the Normalized Cut Value $NCut(A, B)$.

According to [57], the $NCut$ minimization problem can be transformed into solving a standard eigen system:

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}x = \Lambda x \tag{4.21}$$

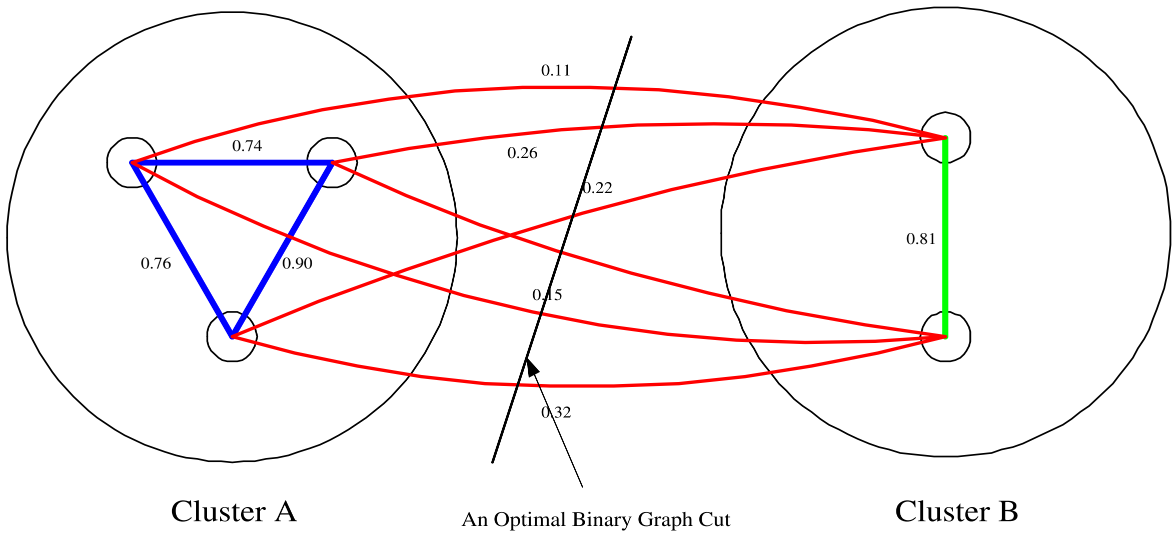


Figure 4.6: Spectral graph cut sample

Here D is a diagonal matrix, $d_{ii} = \sum_j w_{ij}$. W is the shot similarity matrix. The eigenvector corresponding to the *second smallest eigenvalue* can be used to partition V into A and B.

We can recursively construct sub-graphs A and B and solve the eigen system to get finer clusters until some ending condition is met. In this way we can partition the vertex set into smaller sets. When the elements in a vertex set is "similar" enough we cease partitioning. Then we get several video shot groups and a series of "single" video shot groups, in which only one video shot is in the group. We put all the single un-grouped shot together to form a background video shot group. With this grouping information we can easily build up the video scene structure. The merit of this recursive algorithm is that every time it split the vertex set, a global optimal cut is found thus the clusters can better adjusted to the nature of the original data. The recursive algorithm is described in algorithm:

The graph partitioning algorithm outperform the highly threshold-dependent

Algorithm 3 Recursive clustering algorithm based on spectral graph partitioning

```
Input: The set of video shots $SH = \{sh_1....sh_n\}$, the shot pair similarity $\{sim_{ij}\}$ for all i,j $\in \{1...n\}$;
Output: The set of video shot groups $SG = \{sg_1....sg_m\}$
1. Add the first group $sg_1$ to $SG$, assign $sh_1$ to $sg_1$;
for $sh_i \in SH$ do
  for $sg_j \in SG$ do
    calculate the shot-to-group similarity $sim_{sh_i,sg_j}$;
    if $sim_{sh_i,sg_j} \geq s_{th}$ and $d_T(sh_i, sg_j) \leq T_th$ then
      assign $sh_i$ to $sg_j$
    else
      add new shot group $sg_k$ to $SG$, assign $sh_i$ to $sg_k$
    end if
  end for
end for
```

window sweeping algorithm for every time the algorithm generates a new cluster, it considers the global similarity information and the splitted clusters are global optimal. It is also easy to create self-adaptive thresholds for clusters with different sizes, while the hard threshold grouping algorithm tend to over-divide groups or fail to classify different groups, according to the fixed threshold. Some shot groups generated by each method are shown in Figure **??**.

According to the definition of the video shot groups, we can pose a time threshold T_{th} to the intervals between consecutive video shots in the detected video shot clusters, then we split the cluster into video shot groups if the interval is greater than T_{th}.

4.3 Video scene detection

After we have detected the video shot groups, we can continue find the video scenes. According to our definition there are two kinds of video scenes, *loop scenes*, which is composed by more than one video shot groups, and *progressive scenes*, which is composed by a series of visually dissimilar video shots.

According to the definition, the contents of interlaced video shot groups should be relevant thus should be put in the same scene, which is a story unit in the video like a paragraph in an article. Given the detected video shot groups $\{sg_1...sg_n\}$, 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.

All video scenes are formed after every video group is examined in turns. The video scene formation algorithm is given as algorithm 2, where we use $start_{sc_i}$, $start_{sg_i}$ to denote the start frames of video scene sc_i and video shot group sg_i, and end_{sc_i} and end_{sg_i} to denote the end frame of video scene sc_i and video shot group sg_i.

Algorithm 2 detects all the loop scenes, as the examples shown in Figure 4.7. The left un-grouped single video shots then compose progressive scenes.

A loop scene is composed of more than one video shot groups, while a progressive scene is composed of a series of dissimilar video shots. Loop scenes are

Algorithm 4 Video loop scene formation algorithm from a set of video shot groups

```
Input: The set of video shot groups $SG = \{sg_1 ... sg_n\}$, where $sg_i$ is sorted by the temporal sequence of their first member shots;
Output: The set of $k$ video scenes $SC = \{Sc_1 ... Sc_k\}$,
1. Add the first scene $Sc_1$ to $Sc$, assign the first video shot group $Sg_1$ to $Sc_1$;
for $sg_i \in SG$ do
  for $sc_j \in SC$ do
    if $start_{sc_j} < start_{sg_i}$ and $end_{sc_j} > start_{sg_i}$ then
      assign $sg_i$ to $sc_j$
    else
      if $start_{sc_j} < start_{sg_i} AND end_{sc_j} < start_{sg_i}$ then
        add new video scene $sc_k$ to $SC$, assign $sg_i$ to $sc_k$
      end if
    end if
  end for
end for
```

often used to depict an event happening in a place that needs detailed description, e.g., a conversation, while the progressive scenes are often used to depict changes between two events or some dynamic scene. We think that normally the loop scenes contain more important contents that need repeated illustration, thus they are relatively more important than the progressive scenes. The loop scenes and progressive scenes are treated differently during video skimming generation.

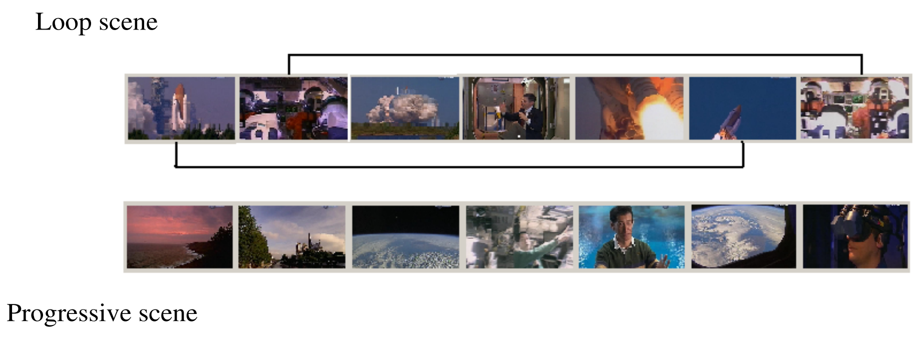


Figure 4.7: Example of loop and progressive scenes

4.4 Shot arrangement patterns

In each loop scene, we can assign a unique label for each detected video shot group it contains. A video shot group composed with more than one video shot is called a key video shot group, and all the remaining single video shots are regarded as a background video shot group. Thus a loop video scene can be regarded as composed of several key video shot groups and one background video shot group. After label assignment, each video shot group g_k has a group label lb_k, shared among the video shots contained in it. Let the set of group labels be LB. Given a video scene $Sc_x = \{sh_1....sh_n\}$ we can have a group label string $lb_1....lb_n$ corresponds to the video shot list in the scene, where $lb_i \in LB$.

In the label string we can find specific video shot arrangement patterns. Such patterns reflect how the editor arrange the video shots and weave them to tell a story. Thus the patterns in the video shot string are very important information about video editing and should be analyzed and utilized for video summarization.

Here we give some definition for video shot string analysis.

1. A *video shot string* str is defined as a series of consecutive video shots $\{sh_1....sh_x\}$, with the group label string $\{lb_1...lb_x\}$; The importance value of a video shot string I_{str} is defined as $I_{str} = \sum_{j=1}^{x} v_j$, v_j is the importance value of video shot sh_j.

2. A *non repetitive shot string* (nrs string) is defined as a video shot string $\{sh_1....sh_x\}$, $\forall i, j \in \{1...x\}$, $lb_i \neq lb_j$.

3. A *k-non repetitive shot string* (k-nrs string) is defined as a non repetitive shot string with length k. We use $\{k\text{-}nrs_j\}$ to denote a set of nrs string with length k.

4. If str_i is the sub-string of str_j, we say that str_j *covers* str_i. For example,

the 4-nrs string 3124 covers two 2-nrs strings $\{312, 124\}$, three 2-nrs strings $\{31, 12, 24\}$ and four 1-nrs strings $\{3, 1, 2, 4\}$.

nrs string carries important information about how the video editor arrange the video shots. They are the reflection of the intention of the video editor. We can easily find all k-nrs strings by scanning through the video label string. Then we use them as skimming candidates. Some sample nrs strings are shown in Figure 6.7.

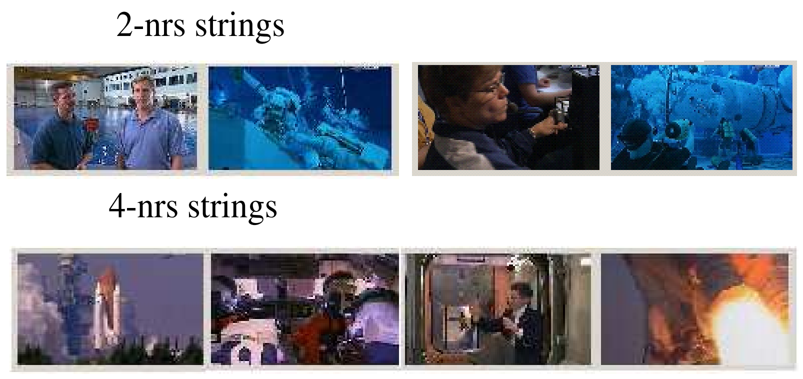


Figure 4.8: Several detected nrs shot strings

To ensure a balanced content coverage, the skimming shots should be able to cover as many semantically important shots as possible. To guarantee the coherence of the video skimming, on the other hand, we hope to pick more longer substrings from the video shot list. Thus the k-nrs strings become very good candidates for video skimming. First, they are composed of video shots depicting non repetitive contents with the least redundancy; second, they are a coherent part of the original video. By scanning the video shot string we can easily obtain all k-nrs strings for all k.

Given a video shot sh_i and a k-nrs string $str_j = \{sh_{j_1} ... sh_{j_k}\}$, we can define the visual similarity between them as:

$$sim(sh_i, str_j) = \sum_{x=1}^{k} sim(sh_i, sh_{j_x}) \times \frac{Length(sh_{j_x})}{\sum_{y=1}^{k} Length(sh_{j_y})} \tag{4.22}$$

After that, we can define the visual similarity function between two *k-nrs* strings str_i and str_j:

$$sim(str_i, str_j) = \sum_{x} sim(sh_{i_x}, str_j) \times \frac{Length(sh_{i_x})}{\sum_{y} Length(sh_{i_y})} \tag{4.23}$$

where $sh_{i_x} \in str_i$.

4.5 Experiments

In this section we evaluate the algorithms proposed for video structure analysis. We carry out the experiments on a PC platform with win2000 OS, P4 2.0G CPU and 512Mb Ram. The video processing module is developed with Visual C++, based on Microsoft DirectsShow SDK. We employ several documentary videos and several movie clips as the test data set.

For video shot cut detection, we have implemented our proposed approach and apply it to several test videos from movie, news and documentary video. The experiment results are shown in Table 4.1. We compare the result with the approach in [46] and found that our method's indeed outperform the previous method in [46], in which the shot cut detection accuracy reported varies from 63.4% to 85.9%. The difference should lies in our non-linear neighborhood filtering. From the result data we can see that our method is quite robust and accurate, in most cases, our method achieves a correct detection rate around 95 percent, while the computation cost is quite low.

For fade detection, our result is shown in Table 4.2. We can see that the result is quite satisfying too. However, since the feature is not as good as video cuts and is more sensitive to noises, the correct detection is a bit less than the video cut detection method.

Table 4.1: Video cut detection results

Video type	**Length**	**Ground truth**	**Detected**	**F. D.**	**M. D.**	**Right Per.**
Movie	477 sec.	166	157	0	9	94.6
Movie	1440 sec.	237	235	6	8	96.6
Document	1380 sec.	147	141	3	9	94.3
News		40	39	1	1	95.0

Table 4.2: Video fade detection results

Video type	**Length**	**Ground truth**	**Detected**	**F. D.**	**M. D.**	**Right Per.**
Document	620	12	10	1	3	75.0
Document	1380	21	23	5	3	85.7
Movie	477	4	3	0	0	75.0

For videos shot group formation, we have implemented the graph cut algorithm and the window-sweeping algorithm in [53]. Some sample video shot groups generated by the two algorithms are shown in Figure 4.9 and Figure 4.10.

From the detected video shot groups we can construct the video scenes. However, as video scene is a high level concept, the ground truth of "video scene boundary" might not be definite. So in the following experiment we employ the "Human opinion" as the ground truth. Table 4.3 and Table 4.4 summarize some results for scene detection results generated by our graph based method and the original ToC method (F. D. for False Detection and M. D. for Missed Detection).

Table 4.3: Video scene detection results by ToC

Video type	**Length**	**Human Opinion**	**Detected**	**F. D.**	**M. D.**
Movie	477 sec.	4	5	2	1
Movie	1440 sec.	9	10	2	1
Document	620 sec.	6	8	4	2
Document	1380 sec.	7	10	4	1

From the experimental result data in Table 4.3 and Table 4.4, we can see that the spectral clustering method outperform the ToC method in correctly detecting video scenes. And our work is based on the video shot group-scene

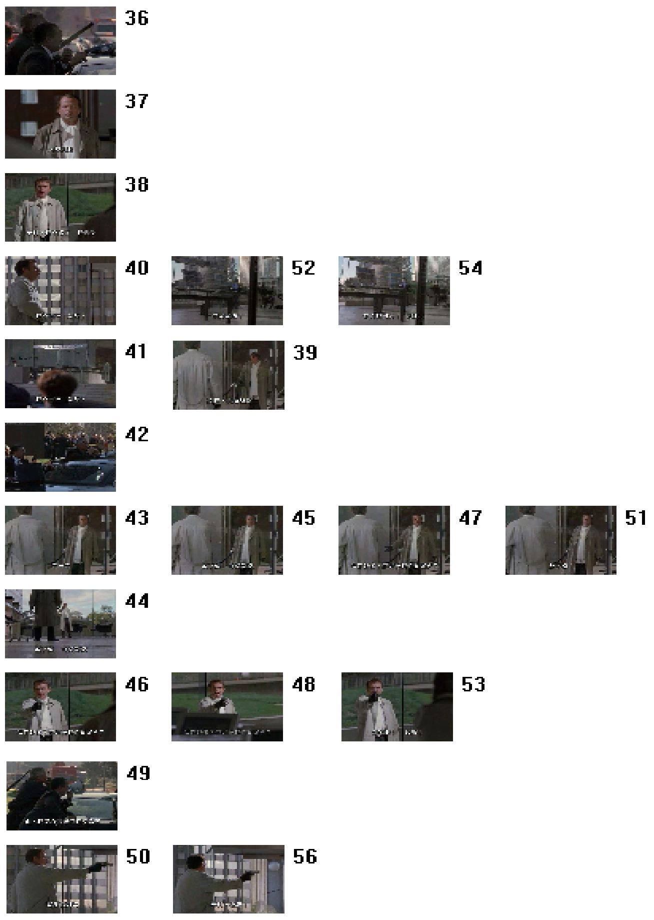


Figure 4.9: Several detected video shot groups by ToC method

detection result of this method. Although the performance of the automatic video scene segmentation seems to be not quite satisfying, the quality of the video summary based on it is not too bad, as we will show in the next chapter.

4.6 Summary

A video has its intrinsic structure just like an article does. In this chapter, we describe our work done on analyzing the structure information of the video

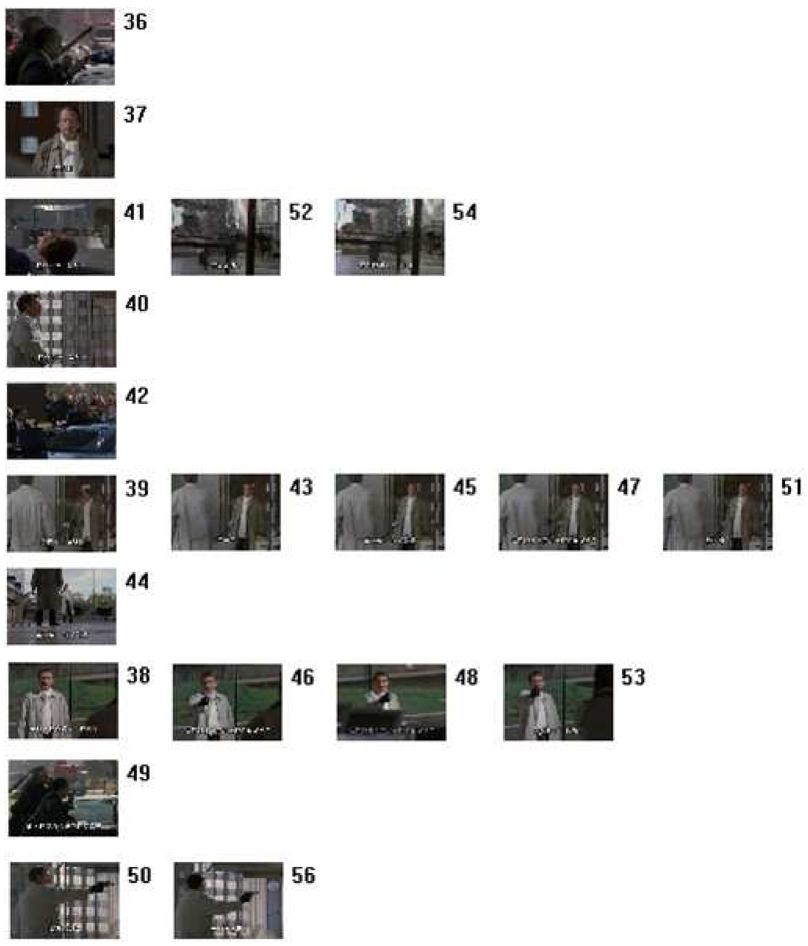


Figure 4.10: Several detected video shot groups by spectral graph partition

documents. A video file is first broken into coherent video shots by cut detection and fade detection, and its key frames are extracted; then the similarity function between video shots are defined and calculated based on color and motion information; similar video shots are grouped and the video story is divided into video scenes just like paragraphs in articles. In each detected scene, we discover and analyze the non-repetitive arrangement patterns for further video summary purpose for they carry important information about video editing.

Table 4.4: Video scene detection results by Spectral graph method

Video type	**Length**	**Human Opinion**	**Detected**	**F. D.**	**M. D.**
Movie	477 sec.	4	5	1	0
Movie	1440 sec.	9	10	2	1
Document	620 sec.	6	6	1	1
Document	1380	7	9	2	0

Chapter 5

Graph optimization based video summary generation

In the previous chapter we analyze the structural information of a video, and in this chapter we describe our video summarization scheme based on the recovered structural information and graph optimization. Based on different user requests, we can classify video summaries into two types:

1. **Overview**–In cases that the user is totally unaware about the content, like facing several movie files that he has never seen, he can only specify a target length thus hope to see enough detail about the movie. Such requests may be like "Give me 3 minutes of preview showing that this movie is about", and we call this kind of video skimming "overview". In this thesis we concentrate on the movie overview generation.

2. **Highlight**–For specific domain like sports video and news, the user already knows some domain specific knowledge and he may just request those video shots that she is interested in like "give me 3 minutes of video about goals and corner kicks". This kind of video summary is called "highlight".

Based on the video structure analysis result, we describe our automatic video summary generation method in this chapter. First, we specify several targets that a good video summary should be able to achieve, shown as follows:

1. **Conciseness**–For conciseness, the length of the generated video skimming should be within the user-specified length L_{vs}; the static summary should not contain too many images.

2. **Balanced content coverage**–As the video is a structured document, the video skimming should be able to represent the original contents with balance. At the same time, both the visual diversity and the temporal coverage of the original contents should be reflected by the video skimming.

3. **Visual coherence**–One problem for traditional video skimming generation is that the user often feel that the video skimming is quite choppy. A good video skimming should increase the coherence of the video skimming while preserving the content coverage.

In this chapter we describe how we select video shots from the video to form video summaries that is able to meet the above goals. The work flow of our video skim generation framework is shown in Figure 5.1:

5.1 Video scene analysis

To ensure the informativeness of the generated video skimming, we need to preserve the important contents of the original video. In the previous chapter, we have already found the scene boundaries in the video, since each video scene can be viewed as a paragraph, we can create summary for each video scene then concatenate them to form a video summary. Given a series of detected video scenes $\{Sc_i\}$ and the target video skimming length L_{vs}, we need to determine the target skimming length for each of the video scenes first.

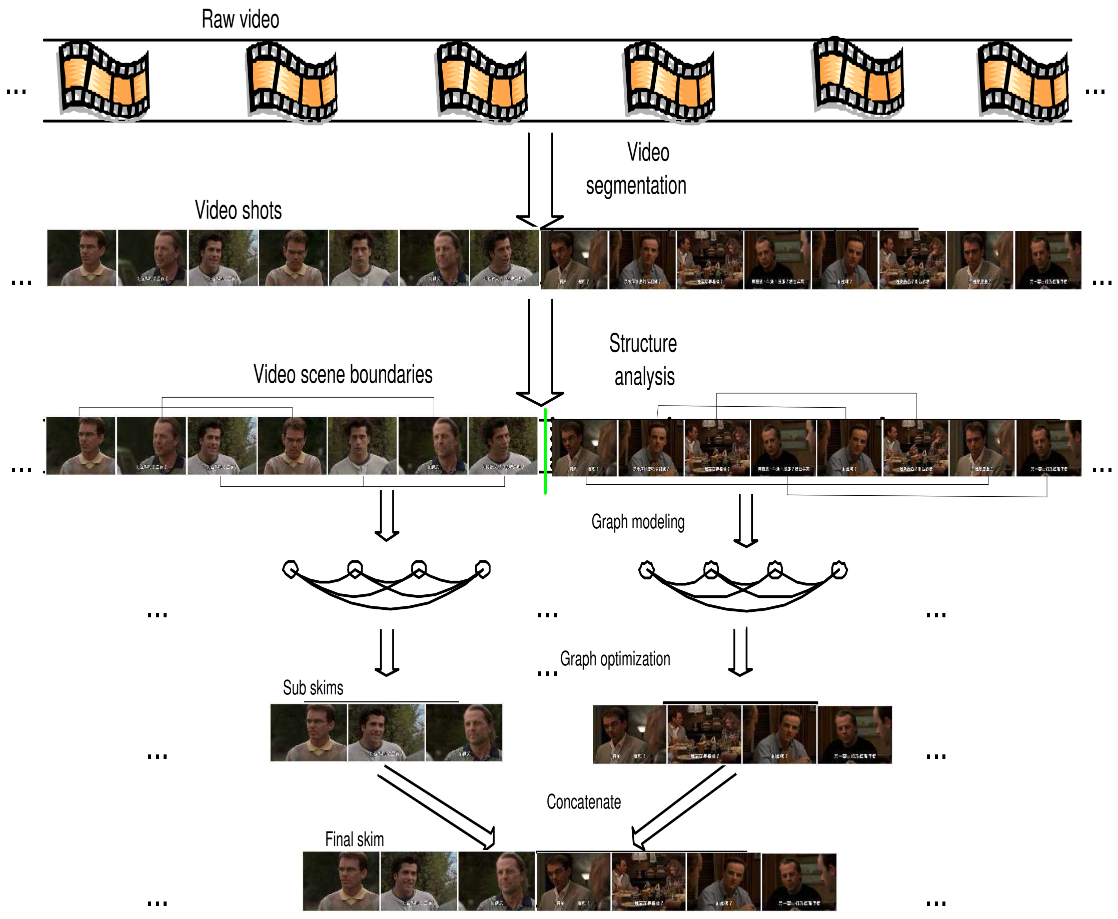


Figure 5.1: Work flow of our video summarization framework

5.1.1 Scene content entropy

In term of importance, we think that the longer and the more complex a video scene is, the more important it should be. For progressive scenes, we simply use their length to measure their importance. For loop scenes, however, since they are composed of several video shot groups, their content complexity should relate with both the number of video shot groups it contains and the lengthes of the member video shot groups. To quantitatively evaluate the complexity of a loop video scene based on these two factors, we define the content entropy of a video scene and employ it as the measure of content complexity, defined in equation 5.1:

$$Entropy(Sc_i) = \sum_{Sg_j \in Sc_i} -\frac{l_{Sg_j}}{l_{Sc_i}} \log_2(\frac{l_{Sg_j}}{l_{Sc_i}}). \tag{5.1}$$

5.1.2 Target skim length assignment

With the above definition, given the target video skimming length L_{vs} and the length of the video L_v, the skim ratio r_s is thus $\frac{L_{vs}}{L_v}$. We determine the skim length Sl of each of the video scenes in the video as follows:

1. For each progressive scene Sc_x,

$$Sl_x = l_{Sc_x} \times r_s \tag{5.2}$$

   If Sl_x is less than the preset threshold t_1, we discard scene Sc_x as too short skim does not make sense to people.

2. Suppose that after the first step, the left skim length is L'_{vs}, for the loop scenes $\{Sc_1 ... Sc_n\}$,

$$Sl_i = L'_{vs} \times \frac{Entropy(Sc_i) \times l_{Sc_i}}{\sum_{j=1}^{n} Entropy(Sc_j) \times l_{Sc_j}} \tag{5.3}$$

   In a similar manner, we discard Sc_i if Sl_i is less than a preset threshold t_2.

3. Suppose that after step 2, the remaining loop scenes are $\{Sc'_1 ... Sc'_m\}$, then for the remaining video scenes we set

$$Sl_i = L'_{vs} \times \frac{Entropy(Sc'_i) \times l_{Sc'_i}}{\sum_{j=1}^{m} Entropy(Sc'_j) \times l_{Sc'_j}} \tag{5.4}$$

The above skim length assignment algorithm ensures that more important scenes are assigned with more skim length, thus the balanced content coverage can be achieved. Moreover, loop scenes are assigned with more skim length, since they are regarded as more important than the progressive scenes.

5.2 Graph modelling of video scenes

Coherency is another important goal that the video skimming should achieve. If a lot of breaks are included in the video skimming, the user may feel that it is jumpy and less enjoyable. Since video shots are continuous image sequences and they are the building block of videos, in our work, we select complete video shots as the elements in the video skimming to guarantee the coherence of the video skimming. Moreover, we employ nrs strings as skim candidate to achieve better coherency.

With each scene's target skimming length determined, we need to select several video shots according to the skim length of each video scene and generate the final skimming. The selected video shots should be able to cover both the visual diversity and the temporal distribution of the original video scene; meanwhile, the coherency of the video skim should be ensured. To achieve all these objectives simultaneously, we model each video scene with a graph based on the video nrs shot strings it contains, then we select the skimming video shots by performing optimization on that graph.

5.2.1 Decompose the video scene into candidate video strings

In the previous chapter we define and detect non-repetitive video shot strings (nrs strings). Since the non-repetitive string is a longer coherent part of a video and it contains no visual redundancy, it is very good candidate that can be selected into the video skim.

To model the scene as a graph, we first need to decompose the video scene into a set of non-overlapped nrs strings $Nrs_{l_{str}}$. However there are still some problems to meet: First, the longer the nrs strings are, the more coherent the final selected video skim will be. However, if the candidate nrs strings are too long, the content coverage might not be able to get guaranteed. Another problem is that there may be more than one method to decompose the video scene into a set of video shot strings.

So we propose the following bounded decomposition process for a video scene:

1. We specify an integer l_{nrs}, which is the upper bound of shot number in the nrs strings to be decomposed from the video scene.

2. We decompose the video scene shots into a set of nrs strings whose length is upper bounded by l_{str}. Given a video scene Sc_i, we note the nrs string set decomposed from the scene as Nrs_k if we set the parameter l_{nrs} with value k. Since there are more than 1 possible methods to decompose the video scene, every time we decompose the first longest possible nrs string from left to right. For example, the Nrs_3 set for a video scene $\{1245141316\}$ should be $\{124, 514, 13, 16\}$.

We can use the integer l_{nrs} to control the coherence of the extracted video skim. As a special case, the Nrs_1 set of a scene is just all the video shots it contains. The longer l_{str} is, the more coherency will the selected skim be, but the content coverage might decrease accordingly.

5.2.2 The spatial-temporal relation graph

Based on the shot nrs strings we detect from the video shot list, we define the spatial-temporal relation graph as follows:

The spatial-temporal relation graph $G(V, E)$ is a graph defined on a video shot string set $S_{sh} = \{str_1, ...str_n\}$ such that:

1. $G(V, E)$ is a complete graph.

2. Each vertex $v_i \in V$ is corresponding to a video shot string str_i in S_{sh} and vise versa. On each v_i there is a weight w_i which is equal to the length of video shot string str_i.

3. On each edge $e_{ij} \in E$, there is an edge weight $w_{e_{ij}}$ which is equal to the spatial-temporal dissimilarity function $Dis(str_i, str_j)$ between video shot strings str_i and str_j. The direction of edge e_{ij} is from the temporally earlier shot string to the temporally later video shot string. Thus G is acyclic.

A simple example of the spatial-temporal relation graph on a scene is shown in Figure. 5.2.

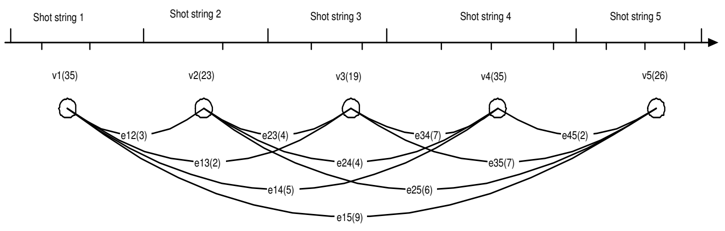


Figure 5.2: Spatial temporal dissimilarity graph on five shot strings

To determine the value on each edge, we define the spatial-temporal dissimilarity function between two video shot strings str_i, str_j as:

$$Dis(str_i, str_j) = 1 - sim(str_i, str_j) \times e^{-k \times d_T(str_i, str_j)} \tag{5.5}$$

and

$$w_{e_{ij}} = Dis(str_i, str_j) \tag{5.6}$$

Here $sim(str_i, str_j)$ can be any visual similarity measure between video shot strings, and here we use the definition given in the previous chapter. $d_T(str_i, str_j)$ is the temporal distance between the temporal middle point of video string str_i and str_j, in terms of frame number. k is the parameter to control the slope of the exponential function, also in terms of frame number. An example of similarity function between video shots in a sample video is shown in Figure 5.3:

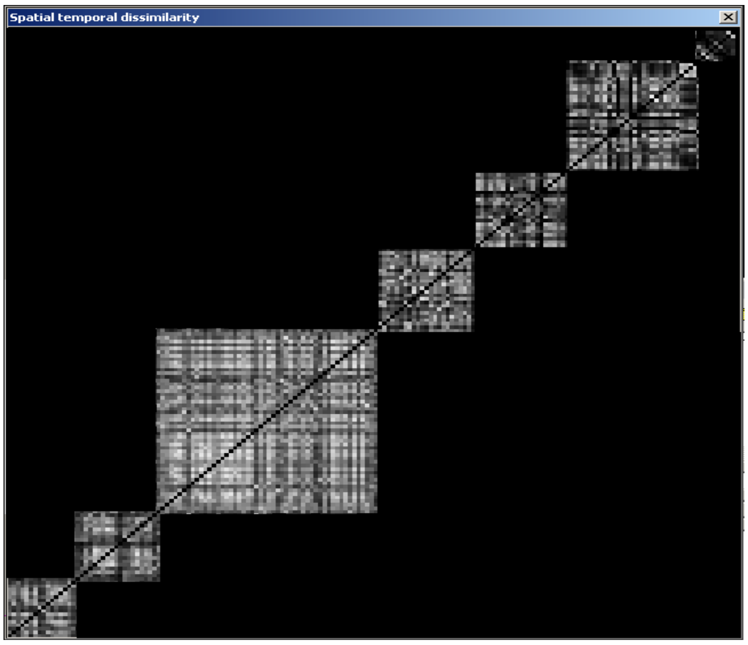


Figure 5.3: The spatial-temporal dissimilarity function for a sample video with 7 scenes

To allow for a good coverage of both the visual and temporal contents of the video scene, we define the dissimilarity function such that it changes linearly with the visual similarity, but exponentially with the temporal distance. Such a dissimilarity definition will guarantee those visually dissimilar video shot strings are chosen, and it can guarantee that the chosen video shot strings will be well distributed on the time axis, so that a good temporal coverage of the original contents can be achieved.

5.2.3 The optimal skim problem

Given the target skimming length L_{vs}, we can search a path in the spatial-temporal graph then use the video shots corresponding to the vertexes in that path as the video skimming for the video shot set. A path $p = \{v_{x_1}, ...v_{x_n}\}$ in the spatial-temporal graph consists of a set of video shot strings $\{str_{x_1}, ...str_{x_n}\}$, which is a video skimming whose total length is the summation of the weights on the vertexes $v_{x_1}, ...v_{x_n}$ in the path. We let $VWS(p_i)$ represent the vertex weight summation of the path p_i. The length of the path is the summation of the spatial-temporal dissimilarity function between consecutive video shot pairs.

For this optimal path p_s, we have two goals to meet: First, we want to maximize the length of the path L_{p_s}, which is the summation of dissimilarity function between consecutive video shot strings; Second, we want $VWS(p_s)$ to be as close to L_{vs} as possible, but not to exceed it. So the problem is a constrained longest path problem. The result path varies with different target skim lengths, as shown in Figure 5.4:

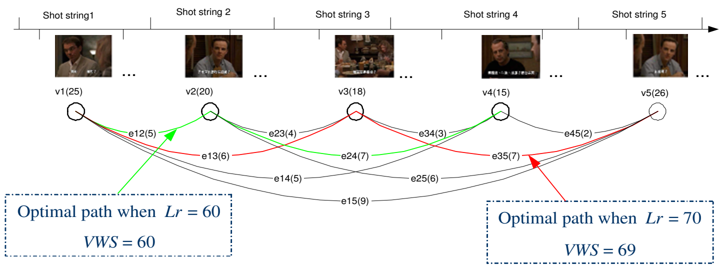


Figure 5.4: The optimal pathes with different VWS values

We combine the above two goals in the objective function f_{obj}, which is described in the following definition for our video skimming generation problem.

Problem 4 Given a set of video strings $S_{str} = \{str_1...str_n\}$, the spatial-temporal graph $G(V, E)$ built on S_{str}, the target video skimming length L_{vs}, and a weight parameter w, search the path $p_s = \{v_{s_1}...v_{s_n}\}$ such that it maximizes the object function

$$f_{obj}(p_s, L_{vs}) = L_{p_s} + w \times (VWS(p_s) - L_{vs}) \tag{5.7}$$

under the constraint that $VWS(p_s) \leq L_{vs}$.

Here w is the weight parameter for the length difference between the selected video skim length and the target length.

5.3 Graph optimization

Problem 4 is a constrained optimization problem. Brute force searching is feasible but inefficient; however, the problem has an *optimal substructure* [10] and can be solved with dynamic programming, shown as follows.

Suppose there are n video shot strings in the video shot set. We add a virtual vertex v_0 such that $w_0 = 0$ and $w_{e_{0j}} = 0$ for all $0 < j \leq n$. We use $p^i_{v_x,l_r} = \{v_x, ...\}$ to denote a path in the spatial-temporal relation graph such that it begins with vertex v_x, and its vertex weight summation is upper bounded by l_r. We then use $p^o_{v_x,l_r}$ to denote the optimal path among all such paths, which means $f_{obj}(p^o_{v_x,l_r}) = \max_i f_{obj}(p^i_{v_x,l_r})$. Thus $p^o_{v0,L_{vs}}$ is the path we want to find.

Then we have the following optimal substructure:

1. $f_{obj}(p^o_{v_n,l_r}) = w \times (l_{sh_n} - L_{vs})$, for all $l_r \leq L_{vs}$;

2. $f_{obj}(p^o_{v_x,l_r}) = \max^n_{y=x+1}[Dis(str_x, str_y) +$
   $f_{obj}(p^o_{v_y,l_r-l_{str_y}}) + w \times l_{str_x}] \times \tau(l_r, y)$

   Here $\tau(l_r, y) = 1$ if $l_r - l_{sh_y} \geq 0$, otherwise $\tau(l_r, y) = 0$.

With the above optimal-substructure we can calculate the object function value of the optimal path $f_{opt}(p^o_{v0,L_{vs}})$ and all optimal sub-solutions with the following dynamic programming algorithm:

After the objective function of the optimal path is found, we can easily trace back and find the global optimal path as well as the skimming shots of the scene. In case there are multiple global optimal paths, the trace back algorithm will also find all of them. We concatenate the skimmings of each video scene and get the whole video skimming. Note that the algorithm may

Algorithm 5 Video skim selection algorithm based on dynamic programming

```
Input: The spatial-temporal relation graph $G(V, E)$ based on the candidate video string set $Str_{in} = \{str_1....str_{S_n}\}$.
Output: The objective function value for the optimal path $p^o_{v_0,L_{vs}}$, denoted by $F_{opt}$.
BEGIN
Set $L_{opt}[i][j] = 0$ for all i,j;
for $L_r = TH$ to $L_{vs}$ do
  $L_{opt}[LastShot][L_r] = -penalty$;
end for
for $i_x = S_n$ to 0 do
  for $L_r = 0$ to $L_{vs}$ do
    $opt = -infinity$;
    for $t = i_x + 1$ to $S_n$ do
      if $l_t < L_r$ then
        if $opt < L_{opt}[t][L_r - l_t] + Dis(str_t, str_{i_x})$ then
          $opt = L_{opt}[t][L_r - l_t] + Dis(str_t, str_{i_x})$;
        end if
      end if
    end for
    $L_{opt}[i_x][L_r] = opt$;
  end for
end for
$F_{opt} = L_{opt}[0][L_{vs}]$;
END
```

generate a video skimming that is a little shorter than the target length L_{vs}. As this will not affect much about the content coverage of our video skim, we randomly select some video shots to fill that length.

The time complexity of this dynamic programming algorithm is $O(n^2 \times L_{vs})$, while the spatial complexity is $O(n \times L_{vs})$. For most video scenes, n and L_{vs} would not be very large and the algorithm can complete quite quickly.

5.4 Static video summary generation

In some cases the user may prefer to a static video summary, for the static summary requires less bandwidth and can be viewed with a glance. In this section we discuss how to generate a meaningful static video summary for the user.

The graph optimization based approach can easily generate a static video summary for the user. Most straightforwardly, once we have extracted a moving video skimming, we can use the key frame of the selected video shots to form a video static summary. One example of the detected video scene (shown as grouped key frames) and the selected video skim shots images are shown in Figure 5.5 and Figure 5.6.

In most cases the key frames for selected video skim shots are good enough for a static video summary. However, sometimes if the user only want to see a static video summary, they might specify a desired image number and hope to see a video summary that composed by exactly that number of images. In that case, we have to redefine and solve the static video summary generation problem, in which the user may specify the number of images to be selected, noted as N_{vs}. In such a situation, we need to reformulate the graph optimization problem and propose a new solution.

We can still construct the spatial-temporal graph based on the video shots contained in a video scene. However, in this situation some change need to be made. First, since nrs strings are employed for better video skimming coherency and is not helpful here, we do not use the video shot strings here, but directly build up the graph on video shots. Second, in the new situation there is no skimming-length constraint, only one constraint on the number of images N_{vs} is left.

The spatial-temporal graph $G(V,E)$ in the new situation is defined as follows:

1. $G(V,E)$ is a completed graph;
2. Each vertex $v_i \in V$ is corresponding to a video shot.
3. On each edge $e_{ij} \in E$, there is an edge weight $w_{e_{ij}}$ which is equal to the

spatial-temporal dissimilarity function $Dis(sh_i, sh_j)$ between video shot sh_i and sh_j. The direction of edge e_{ij} is from the temporally earlier shot to the temporally later video shot. Thus G is acyclic.

We continue using the shot-pairwise dissimilarity function in the previous sub-section, which is defined as:

Our target is to generate N_{vs} of static shot images that is able to cover the visual content diversity and content temporal distribution. The spacial-temporal dissimilarity function has already combined the visual similarity and the temporal distance between video shots. Since the only constraint is the number of images to be comprised in the video summary N_{vs}, we can search a path p_s with exactly N_{vs} vertexes that maximize the path length, and use those shot key frames corresponding to the vertexes on the path as the final video static summary. We formulate the problem in problem 4.2:

Problem 5 Given a set of video shot $S_{sh} = \{sh_1...sh_n\}$, the spatial-temporal graph $G(V, E)$ built on S_{sh}, the target video shot images N_{vs} ,search the path $p_s = \{v_{s_1}...v_{s_{N_{vs}}}\}$ such that the length of p_s is maximized.

We use $p^o_{i,n}$ to denote the optimal path such that it begin with vertex v_i, and it contain exactly n video shots. We use $l^o_{i,n}$ to denote the length of the optimal path.

We observe that the length of the path $p^o_{i,n}$ also has a optimal structure property, shown as follows:

1. $l^o_{i,n} = \max_{j=i+1}^{N_{sh}} w_{e_{ij}} + l^o_{j,n-1}$, $i \in \{0...N_{sh} - 1\}$;

2. $l^o_{i,n} = 0$, when $n = 1$ or $i = N_{sh}$;

With the above optimal substructure we can use dynamic programming to find the optimal path comprising N_{sh} shots as the solution to the problem. The algorithm is shown in Algorithm 2:

Algorithm 6 Video static summary selection algorithm based on dynamic programming

```
Input: The spatial-temporal relation graph $G(V, E)$ based on the candidate video shot set $Sh_{in} = \{sh_1....sh_{S_n}\}$.
Output: The objective function value for the optimal path $p^o_{v0,L_{vs}}$, denoted by $F_{opt}$.
BEGIN
Set $L_{opt}[i][j] = 0$ for all i,j;
for $i_x = S_n$ to 0 do
  for $N_r = 0$ to $N_{vs}$ do
    $opt = -infinity$;
    for $t = i_x + 1$ to $S_n$ do
      if $l_t < L_r$ then
        if $opt < L_{opt}[t][L_r - l_t] + Dis(sh_t, sh_{i_x})$ then
          $opt = L_{opt}[t][L_r - l_t] + Dis(sh_t, sh_{i_x})$;
        end if
      end if
    end for
    $L_{opt}[i_x][L_r] = opt$;
  end for
end for
$F_{opt} = L_{opt}[0][L_{vs}]$;
END
```

Algorithm 2 find the longest path with exactly N_{vs} shots in the spacial-temporal graph G as the representative video shots. Since in this scenario we don't have the target skim length constraint, the time complexity is now $O(n^2)$ and spacial complexity $O(n)$, for normal scenes, the algorithm finishes quickly.

5.5 Experiments

We implement the video summarization algorithms then apply them to some video clips. We carry out the experiments on a PC platform with win2000 OS, P4 2.0G CPU and 512Mb Ram. The video processing module is developed with Visual C++, based on Microsoft DirectShow SDK. The exponent control parameter k in the spatial-temporal dissimilarity function is set to 250 (in term of frame numbers), and the weight factor w in the objective function is set to 0.01. The threshold parameters t_1, t_2 are set to 3 seconds and 4 seconds, respectively. The test video materials include two movie clips, two documentary videos and one sitcom videos, and video skimmings at skim rate 0.15 and 0.30 are extracted for each test video clip. At each video skim rate, we generate two video skimmings with l_{str} equal to 1 and 3. Detailed information about the test video clips are described in Table 1. An example for a scene's key frames (shown as video shot groups) and the selected skimming video shots' key frames are shown in Figure 5.5 and Figure 5.6.

To evaluate the quality of the generated video skimming, we employ two criterion: *meaningfulness* and *favorite*. Since it is hard to objectively evaluate a video skimming, we use the following subjective test to evaluate the performance of our video skimming generation scheme. To test the meaningfulness of the video skimmings we have attained, 10 people were invited as test users to watch the video skimming generated with two skim rates 0.15 and 0.30 then answer several questions about the video contents. To evaluate meaningfulness, the test users are asked to watch the video skimmings then answer several questions about the major events that the video depicts (Who has done what?). From the number of the questions that the users are able to answer after they have seen the video skimming, we can get a score to measure the meaningfulness of the video skimming. The scores are scaled to $[0, 100]$. To compare the favorite, we ask the users to select a "better" video skimming between the video skims generated with different l_{str} values, and the number of users who choose the skimming as the "better" one is recorded as the favorite score.

Table 1 shows the numerical results for the user test. From the table we conclude that the video skimmings' content coverage is still quit good at a skim rate of 0.15. Moreover, when the skim rate is 0.30, the skimming content coverage is even better.

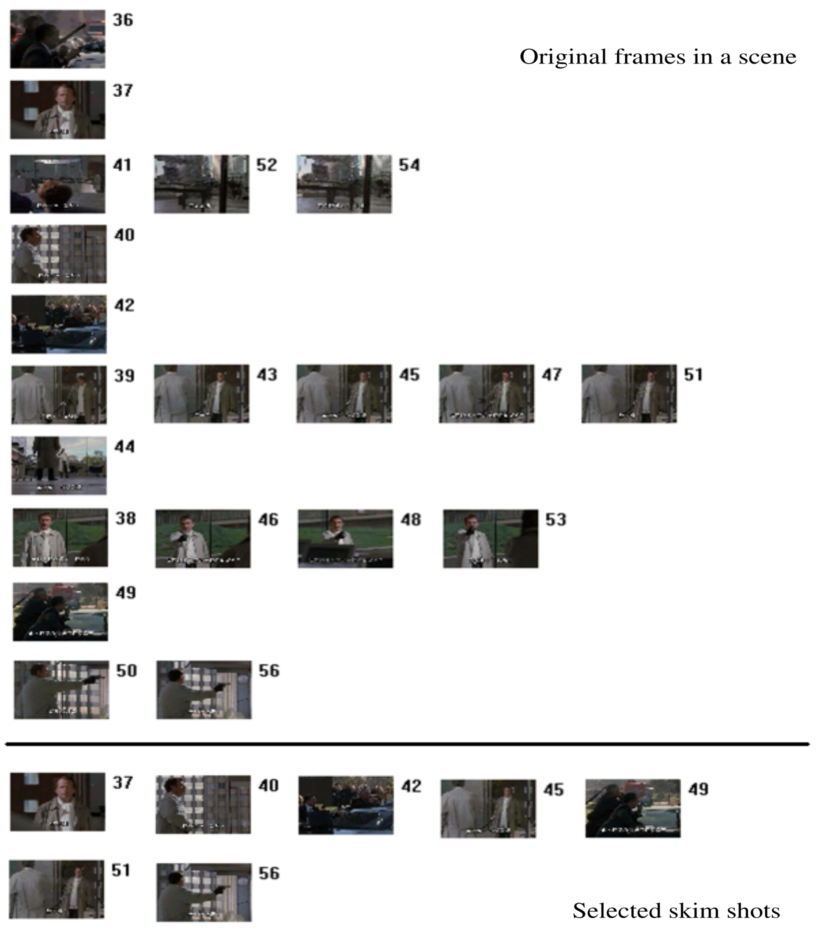


Figure 5.5: Summarized scene key frames

We can also see the effect of the parameter l_{str}. The meaningfulness scores for both video skimmings with different l_{str} are quite similar, but in terms of favorite, most video skimmings generated with bigger l_{str} value gain better favorite scores, which means that more people prefer to view more coherent video skimmings.

For the compression limit of the video skimming, in our experiments, most users say that when the skim rate is less than 0.1, they feel difficulty to understand the selected video skimming, which indicates that, to make sense to

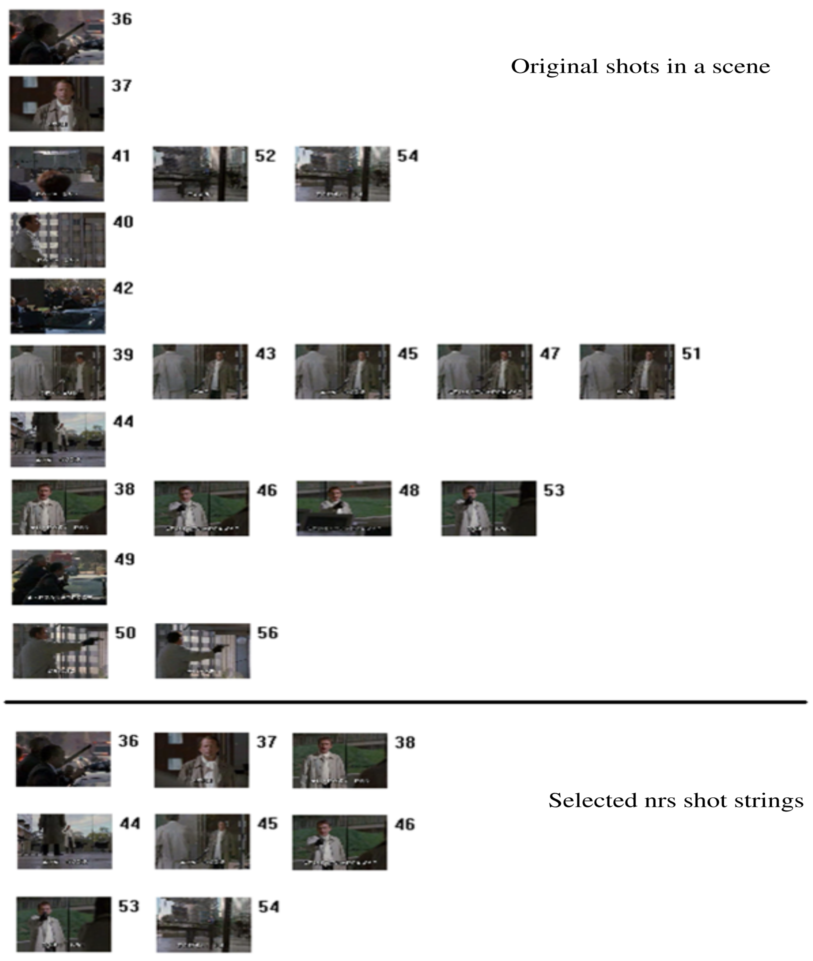


Figure 5.6: Summarized scene key frames based on nrs shot strings

the users, for normal videos a reasonable skim rate should be at least 0.1.

For static video summary, we have generated several set of static video summaries for a test movie clip. One static summary with the shot number equals to 30 is shown in Figure 5.7.

From the static video summary we observe that the visually dissimilarity video shots have been selected; also the video shots distribute quite evenly on the temporal axis, shown in Figure 5.8. Thus we make the conclusion that the graph optimization based algorithm is able to simultaneously guarantee the

Video Clip	**Duration**	**Major events**	**Skim Rate**	**Mfn.**	**Fav.**
Document 1	2403 sec.	7	0.15	82.8/**85.7**	4/**6**
Document 2	3230 sec.	8	0.15	78.8/**76.4**	3/**7**
Document 3	1477 sec.	5	0.15	88.0/**86.0**	3/**7**
Movie 1	1183 sec.	9	0.15	82.2/**85.6**	4/**6**
Movie 2	602 sec.	4	0.15	77.5/**75.0**	4/**6**
Sitcom1	1183 sec.	8	0.15	71.1/**76.4**	3/**7**

Table 5.1: User test results with skim rate 0.15. The scores with l_{str} is equal to 3 are in **bold**

Video Clip	**Duration**	**Major events**	**Skim Rate**	**Mfn.**	**Fav.**
Document 1	2403 sec.	7	0.30	94.3/**92.9**	5/**5**
Document 2	3230 sec.	8	0.30	88.9/**92.9**	6/**4**
Document 3	1477 sec.	5	0.30	96.0/**96.0**	2/**8**
Movie 1	1183 sec.	9	0.30	94.4/**97.8**	5/**5**
Movie 2	602 sec.	4	0.30	92.5/**97.5**	3/**7**
Sitcom1	1183 sec.	8	0.30	84.3/**88.2**	2/**8**

Table 5.2: User test results with skim rate 0.30. The scores with l_{str} is equal to 3 are in **bold**

visual content diversity and temporal coverage.

5.6 Summary

Video summarization is an important technique for efficient video browsing and management. In this paper, we formulate the video skimming generation problem as a two-stage graph based optimization problem. We obtain the video scene boundaries, determine each video scene's skim length, then we model each scene into a spatial-temporal relation graph, and employ dynamic programming to find each scene's optimal skimming. The whole video skimming is concatenated by each scene's skimming. We implemented the proposed algorithm and obtained encouraging experimental results.

In the future, we will further incorporate audio channel analysis to help our skimming generation. Moreover, intra-shot compression will be studied

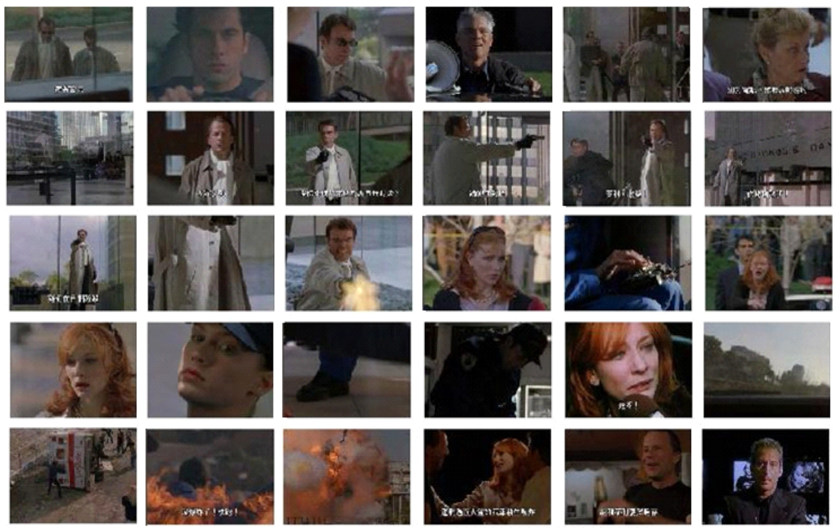


Figure 5.7: An example of static summary

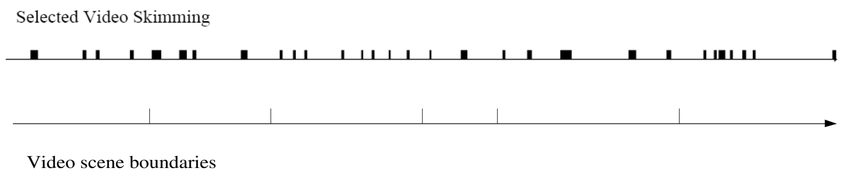


Figure 5.8: Temporal distribution of selected video shots

to shorten the video shots' length in order to further magnify the content coverage.

Chapter 6

Video content annotation and semantic video summarization

6.1 Introduction

Video is increasingly becoming the favorite medium for many communication entities for its extraordinary expressive power. With the rapid growing computing power and storage device capacity, the large scale digital video library system is growing rapidly. This massive growing video data thus gives rise to a challenge for efficient video browsing and management since it is time consuming to download and browse through the whole contents of the video. To solve this problem, video summarization, which engage in providing concise and informative video summaries to help people to browse and manage video files more efficiently, has received more and more attention in recent years. Basically there are two kinds of video summaries: *static video story board*, which is composed of a set of salient images extracted or synthesized from the original video, and *dynamic video skimming*, which is a shorter version of the original video made up of several short video clips.

In recent years much work has been conducted on video summarization. For static summary generation, [70] tends to adapt to the dynamic video content.

A mosaic-based approach is suggested in [29]. Later work present video contents according to the detected video structure. In [53], the authors analyzed the video structure after video segmentation, and then get a tree-structured Video-Table-Of-Contents(V-TOC). In [77], a scene transition graph was constructed as the video content presentation. A curve simplification approach is proposed in [12].

Compared with static video summary, dynamic video skimming is more attractive for it reserves the dynamic property of the video thus it is able to make more sense to the user. Much effort is also devoted to dynamic video skimming generation. In the VAbstract system [30], key movie segments are selected to form a movie trailer. The Informedia system [59] selects the video segments according to the occurrence of important keywords in the corresponding caption text. Later work employs perceptional important features to summarize video. In [39] the authors construct a user attention curve to simulate the user's attention toward different video contents. [65] proposes a utility function for each video shot, and video skimmings are generated by utility maximization. [35] assigns different weight scores on several important features of the video then selects the video skimming that maximizes the feature score summation. [47] analyzes video structure by graph modelling then the video skimming is generated according to this structure and the motion attention values for video shots. In [36], a graph optimization approach is proposed to guarantee the content coverage of the generated video skim.

Most of the traditional video skimming generation approaches are based on low level video features, and they may not be able to guarantee that the generated video skim contains the semantically important contents thus the video skim may not make sense to the users. Video summarization based on semantic annotation can be found in [42, 84, 71, 43]. To attack this problem, semantic information is needed to make a meaningful video skimming. Unfortunately, although quite a lot attempts have been done to automatically annotate generic video and image contents [44, 32, 14] and event detection in specific video categories like sports video [3], their overall performance is still not satisfactory. Thus the automatic recognition of high level semantic information like key actors, action taken is still beyond the capacity of up-to-date techniques. Consequently, to collect reliable video semantic information we still need to manually annotate the video contents, and we build a semi-automatic system to help the user to annotate the video.

In this chapter, we propose a framework for dynamic video skimming generation that emphasizes both the *balanced content coverage* and the *visual coherence*. Figure 6.1 shows the overview workflow of our approach. We first segment the video into video shots, then we create a semantic content description for each of them with a semi-automatic annotation tool. To guarantee the balanced content coverage, by video structure analysis we determine the scene boundaries, and the target skimming length for each scene is determined. For each video shot, an importance value is calculated according to the *Mutual Reinforcement Principle* [81], and the video shots are clustered according to their semantic content descriptions. Finally, we analyze the arrangement pattern of the video shots and the important shot strings are selected as the video skimming. In comparison with the traditional approaches, our approach has the following contributions: First, we employ the *Mutual reinforcement principle* to calculate a global importance rank value for each shot, based on which we can ensure that the semantic important contents can be covered by the skimming; Second, we analyzed the shot arrangement patterns, which is neglected by most existing approaches, and we utilize this information to make a tradeoff between content coverage and visual coherence.

The chapter is organized like follows. In Section 6.2 we review the Mpeg 7 standard. In Section 6.2 we describe the video annotation process. In Section 6.3 we analyze the structure of the video and describe how we calculate the semantic importance value for each video shot by mutual reinforcement

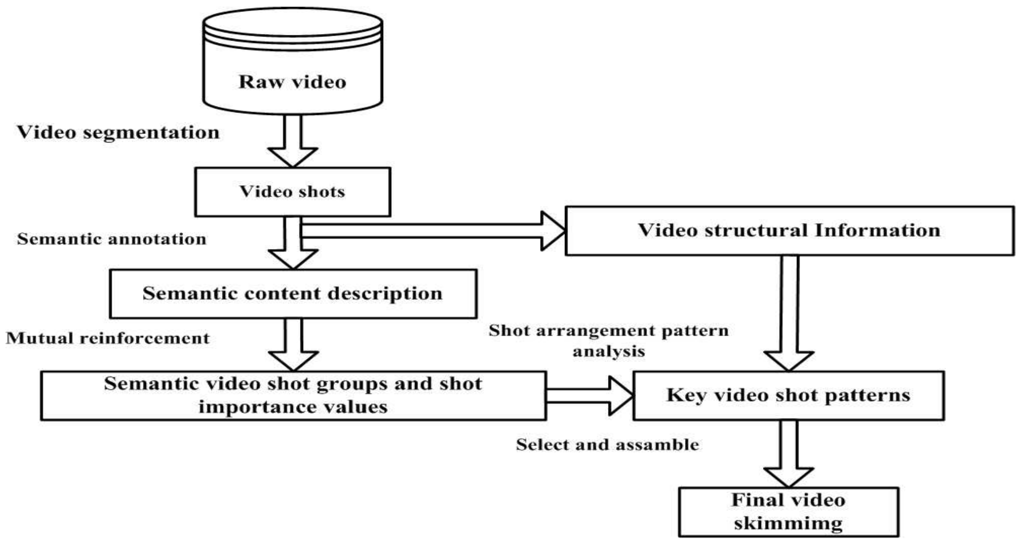

Raw video
Video segmentation
Video shots
Semantic annotation
Video structural Information
Semantic content description
Mutual reinforcement
Shot arrangement pattern analysis
Semantic video shot groups and shot importance values
Key video shot patterns
Select and assamble
Final video skimmimg

Figure 6.1: Overview of the semantic video summarization framework

principle. In Section 6.4 we present our video skim generation scheme. Section 6.5 we show some experimental results. Finally, we make conclusion in Section 6.6.

6.2 Semantic video content annotation

6.2.1 Video shot segmentation

A video shot is a image sequence captured continuously by a single camera. It is the basic building block of edited videos like movies, broadcast news, TV shows, etc. With the methods described in chapter 3, we can efficiently detect the video shot boundaries. With the video shots detected, we can further make annotation for them, and explore the higher level structure of the video.

For video shot sh_i, we use its first frame $kf_{i_{begin}}$ and the last frame $kf_{i_{end}}$ as the key frames to represent the visual content of the video shot.

6.2.2 Semi-automatic video shot annotation

Given the detected video shots, we continue to define the content description for a video shot, then we make annotation for the video shots.

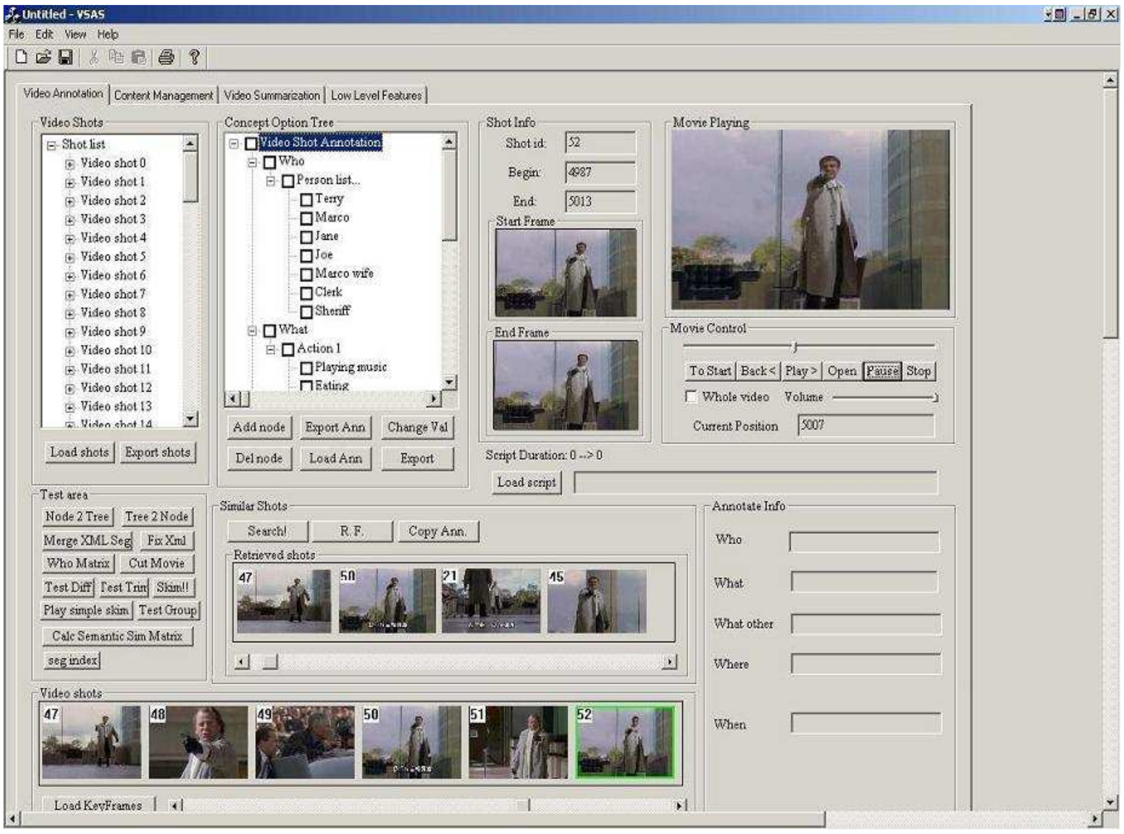
Normally when we see a video, the two questions we mostly want to ask is "Who?" (Who is the person this video is depicting?) and "What?" (What is the person doing?/What's happening?). Thus in this paper, for each video shot's content description, we currently use the following two *semantic concept contexts* to describe the semantic concept of the video shots:

1. **Who**–This context describes the main person in this video shot.
2. **What**–This context describes the action taken by the actors, or events happening.

Under each context there are several concept terms describing the contents. The video concept description, the corresponding contexts and the possible optional concept terms are organized in a tree structured manner named *shot concept tree.* and the user can freely choose the right concept terms for the semantic context. Moreover, the user can easily extend, edit and reuse the shot concept tree.

To accelerate the annotation process we employ a image retrieval module to assist the annotator. When doing annotation, the annotator is provided with the video shot key frames for a preview. He can use the relevance feedback module to retrieve similar video shots and copy the annotation to the similar shots thus the whole semantic annotation process can be highly accelerated. The similar video shots confirmed by the user are stored for further usage. The relevance feedback assisted video annotation interface is shown in Figure 6.2.

After annotation, the video shot content description can be written in a two-unit tuple: $\{F_l, F_s\}$, which is the low level features and the high level semantic concept. Here $F_s = \{c_i\}$, each context c_i contains several semantic



Shot id: 52
Begin: 4987
End: 5013
Current Position: 5007

Figure 6.2: Video annotation interface

concept terms $\{t_{ij}\}$. For a video shots sh_i, we can put the concept terms together into a keyword set $T_i = \{t_{ij}\}$.

6.3 Video structures and semantics

6.3.1 Video structure analysis

A video narrates a story just like an article does. From a narrative point of view, a video is composed of several video scenes $\{Sc_1...Sc_n\}$, each of which depicts an event like a paragraph does in the articles; a video scene is composed by a series of video shots $\{sh_1...sh_n\}$, each of which is an unbroken image sequence captured continuously by a camera. A video shot's role is just like a sentence in articles. The visual content of a video shot can be represented by its key frames. A video shot group Sg_i is the intermediate entity between video scenes and video shots, which is composed of several visually similar and temporally adjacent video shots. Thus from top to down, a video has a 4-level hierarchical structure: Video, Video scenes, Video shot groups, and Video shots [53].

Figure 6.3 shows the hierarchical structure of a video.

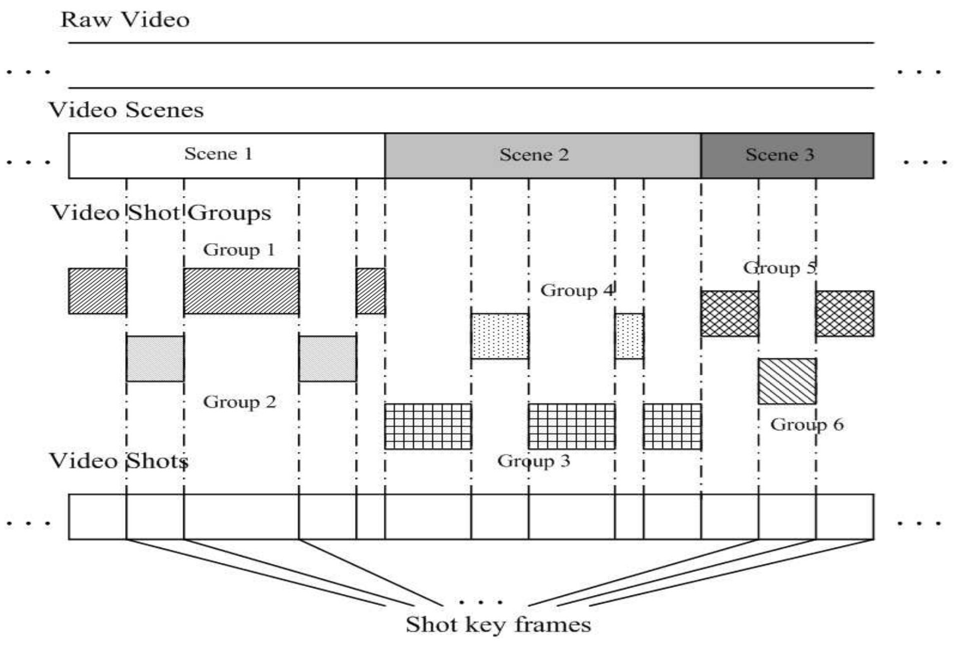


Figure 6.3: Hierarchical video structure

In the remaining part of this paper, we use l_{sh_i}, l_{Sg_j} and l_{Sc_i} to represent the length of video shot sh_i, video shot group Sg_j, and video scene Sc_i, which is the total number of images containing in them respectively.

Given the low level features and the high level semantic description for each video shot, we can define two similarity measures sim^l_{ij} and sim^s_{ij} as the similarity between two video shots based on low level features and high level semantic features.

The low level similarity between two video shots is defined as the maximal H-S histogram correlation between their key frames, that is

$$VisualSim(sh_i, sh_j) = \max_{x,y} HistCorr(kf_{i_x}, kf_{j_y}) \tag{6.1}$$

where $x, y \in \{begin, end\}$.

and we use the keyword similarity to measure their semantic similarity, defined as follows:

$$sim^s_{ij} = \frac{|T_i \bigcap T_j|}{|T_i \bigcup T_j|} \tag{6.2}$$

and we can linearly combine the two similarity measure into a net similarity measure sim_{ij}:

$$sim_{ij} = k \times sim^l_{ij} + (1-k) \times sim^s_{ij} \tag{6.3}$$

and based on measure sim_{ij} we can use the window sweeping algorithm in chapter 3 to find the video shot group and video scene boundaries.

6.3.2 Video structure and video edit process

We have just determined the scene boundaries based on the visual and semantic similarity. And we continue to explore the semantic structure of a video scene.

The video editing process, described in follows [21], is like follows: To describe an event the director will first shoot the environment from several different angles, then mix the video shots from various angles to assemble the final edited video. For example, to depicting a conversation, there should be some overview shots showing all the people involved at the beginning and the end of the scene, and there may be several sets of video shots depicting each involved actor from different angles. The video shot sets are depicting the same content(person) but since they might be shot from different angles so they might not be able to get grouped together by analyzing the low level features. However, the shot semantic description can help us to find such structure.

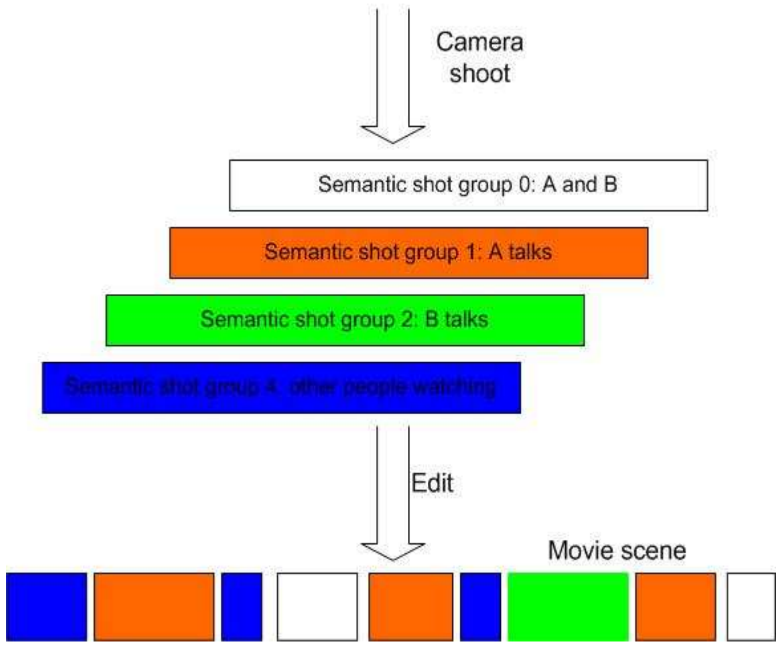

Camera shoot
Semantic shot group 0: A and B
Semantic shot group 1: A talks
Semantic shot group 2: B talks
Semantic shot group 4: other people watching
Edit
Movie scene

Figure 6.4: Movie edit process

To better model the intention of the director, we propose a new concept called *semantic video shot group*. It is made of a set of video shots that depicting the same semantic content. However, a semantic video shot group might not be composed by visually similar video shots. The semantic video shot group can be viewed as an intermediate entity between video group and the video scene, and we can expect that video skimming generated upon this new structure can achieve better performance since it carries the semantic structure of the video. Another important sign of the director's intention is the way he arrange the semantic shot groups. The pattern of the shot arrangement will be analyzed in Section 6.5.

6.3.3 Mutual reinforcement and semantic video shot group detection

Given a video scene composed by a set of annotated video shots and a set of video annotation concept terms and the corresponding contexts, we need to measure the relative importance of each video shots and each different concept term. We employ the following mutual reinforcement principle [81] to detect the semantic video shot groups and give a importance evaluation for each detected video shots. Suppose that we have obtained a set of video shot descriptions $D = \{d_1...d_n\}$ based on a set of concept terms $T = \{t_1...t_m\}$ under the description context c, we hope to get a rank to measure the priority of the description items video shot description set. A weighted bipartite graph can be built from T to D in the following way: if description d_i contains term t_j, then we set up a edge between d_i and t_j, and we can compute a weight w_{ij} associated with the edge. w_{ij} can be any non-negative measure of the relationship between concept terms and descriptions. In this paper, we define the weight such that if description d_i contains concept term t_j then $w_{ij} = 1$, else $w_{ij} = 0$.

The idea of mutual reinforcement principle [81] is as follows: an important term should occur in many important descriptions; and an important description should contain many important terms. The principle dictates that, the importance score of a concept term is determined by the importance scores of the descriptions it appears in; And the importance score of a semantic description is determined by the importance scores of the concept terms contained in it. Given the shot description set D and term set T, the weight matrix $W = [w_{ij}]$, we use the vector U and V to denote the importance scores for the term set T and the description set D. Mathematically, we have the following relationship:

$$U = \frac{1}{k_1} W V \tag{6.4}$$

and

$$V = \frac{1}{k_2} W^T U \tag{6.5}$$

, where k_1 and k_2 are some constants.

We can easily get

$$U = \frac{1}{k_1 k_2} W W^T U \tag{6.6}$$

and

$$V = \frac{1}{k_1 k_2} W^T W V \tag{6.7}$$

Thus we can see that U and V should be the eigenvectors of the matrix WW^T and matrix $W^T W$. Since the elements in W are all non-negative, the largest eigenvalue of $W^T W$ and WW^T must be also non-negative. In that case, we may choose the eigenvectors corresponding to the largest singular value of WW^T and $W^T W$ as the importance scores for the concept terms and shot descriptions.

By this mutual reinforcement process we can find the importance score vector for all the video shot descriptions. We can see that video shot with similar content will have similar importance values. Thus by this importance value vector we can group the semantic similar video shots into semantic video shot groups. For a set of video shots, we compute importance value vectors based on context "who" and "what", thus result in two vector V_{who} and V_{what}. And the final importance vector is obtained by

$$V = V_{what} + V_{who} \tag{6.8}$$

We combine this two value vectors then use it to classify video shots. Since the importance value is a measure for this shot's relative semantic rank, according to the importance value vectors, we can select several semantic video shot groups with high importance values called *key video shot groups*, and the rest content are treated as the *background shot group*. Figure 6.5 shows the importance vector we get from a video scene. Figure 6.6 show some classified shots based on the importance vector. On the top row the shots contains two key actors thus has the highest importance value; The second and third row are those video shots depicting one main actors; The above 3 video shot groups forms the key video shot groups; Those video shots do not contain key actors form the background video shots group, shown in the bottom row.

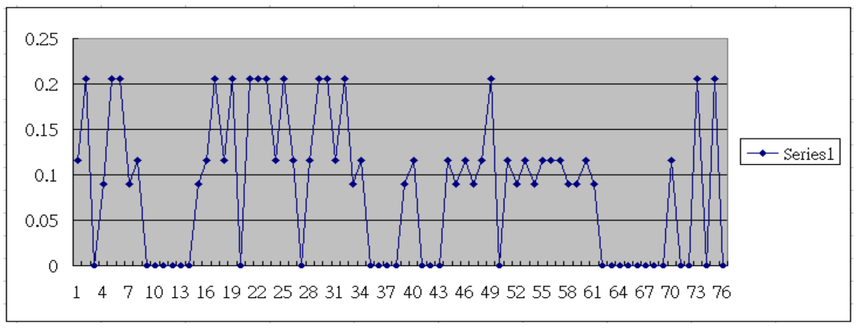


Figure 6.5: Importance vector of video shots

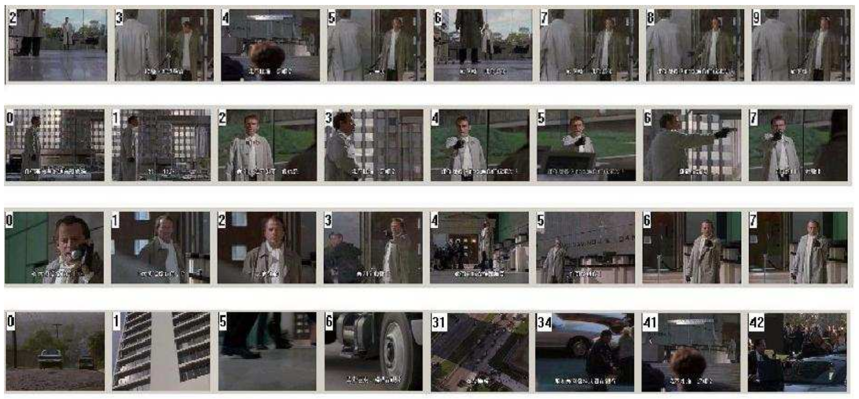


Figure 6.6: Some classified video shots

When we have found the semantic video shot groups, our video summarization process will be just like a inversion of the video edit process followed by another edit process. We first group the video shots depicting the same content, which is just the inversion of movie editing, then we select some shots from the same group according to some rules then reassemble them into the final video skimming.

6.4 Semantic video summarization

6.4.1 Summarization requests and goals

Basically there are two kinds of video skimming: *overview* and *highlight*. For specific domain like sports video and news, the user already knows some domain specific knowledge and he may just request those video shots that he is interested in like "give me 3 minutes of video about goals and corner kicks". This kind of video skimming is called "highlight". But for movies, mostly the user is totally unaware about the content thus can only specify a target length thus hope to see enough detail about the movie. The request may be like "Give me 3 minutes of preview showing that this movie is about", and we call this kind of video skimming "overview". In this paper we concentrate on the movie overview generation.

To obtain a meaningful video skimming, we specify several goals that we would achieve as follows:

1. **Conciseness**–To be useful, the length of the generated video skimming should be within the user specified length L_{vs}.

2. **Balanced content coverage**–As the video is a structured document, the video skimming should be able to represent the original content with balance. At the same time, the visual and semantic diversity of the original contents should be reflected by the video skimming.

3. **Visual coherence**–One problem for traditional video skimming generation is that the user often feel that the video skimming is quite choppy.

Thus we hope to increase the coherence of the video skimming while preserving the content coverage.

Now that we have attained the formation of the semantic video shot groups and video scene boundaries, the final video skimming can be then made by first make sub-skimmings for each scene then concatenate them. Thus our video skimming generation scheme has three steps: First, determine the target length for each scene; Second, extract the sub-skimming according to each length; Finally, assemble the sub-skims to form the final skimming.

6.4.2 Determine the sub-skimming length for each scene

Suppose that the video is composed by a set of detected scenes $\{Sc_i\}$, given total video skimming length L_{vs}, we need to distribute the skimming length to each of scenes. It's natural that longer and more complex video scenes should share a longer part in the final video skimming. To describe the complexity of a video scene, we define the content entropy for a video scene Sc_i as follows:

$$Entropy(Sc_i) = \sum_{Sg_j \in Sc_i} -\frac{l_{Sg_j}}{l_{Sc_i}} \log_2\left(\frac{l_{Sg_j}}{l_{Sc_i}}\right) \qquad (6.9)$$

Here l_{Sg_j} and l_{Sc_i} are the length of the video shot group and video scene, in terms of image frame number.

After we have calculated the content entropy for each video scene Sc_i, given the total video skimming length L_{vs}, we determine the target skimming length Sl_i for each video scene in the following way:

1. For the video scenes $\{Sc_1...Sc_n\}$, we first calculate $Sl_i = L_{vs} \times \frac{Entropy(Sc_i) \times l_{Sc_i}}{\sum_{j=1}^{n} Entropy(Sc_j) \times l_{Sc_j}}$. If Sl_i is less than the preset threshold t_1, then the corresponding scene is considered as non-important thus will be discarded.

2. For the remaining scenes $\{Sc'_1...Sc'_m\}$, we set $Sl_i = L_{vs} \times \frac{Entropy(Sc'_i) \times l_{Sc'_i}}{\sum_{j=1}^{m} Entropy(Sc'_j) \times l_{Sc'_j}}$.

6.4.3 Extracting video shots by string analysis

Now that we have determined all the semantic video scenes' target length, and we can continue to extract some video shots from each video scene to form the sub-video skimming. In [36] we proposed a graph optimization algorithm to select video skimming shots. Each detected video scene is modeled into a graph, and the video skimming is generated by searching a constrained longest path in that graph, such that balanced content coverage can be achieved. However, this method select separate video shots thus the video skimming seems choppy. In this paper, we use a new method based on string sequence analysis to select the shots, which is able to generate a more coherent video skimming while still guarantee the content coverage. We will compare the performance of the two methods in the experiment.

After the mutual reinforcement process, we have the importance vector V for the video shots, each component v_i is the importance value of video shot sh_i. Based on the importance value we can classify the video shots into a set of semantic shot groups $G = \{g_k\}$, including several key video shot groups and one background video shot group. Each semantic group g_k has a group label lb_k, shared by the video shots contained in it. Let the set of group labels be LB. Now that given a video scene $Sc_x = \{sh_1....sh_n\}$ and we can have a group label string $lb_1....lb_n$, where $lb_i \in LB$.

Here we give some definition for video shot string analysis.

1. A *video shot string str* is defined as a series of consecutive video shots $\{sh_1....sh_x\}$, with the group label string $\{lb_1...lb_x\}$; The importance value of a video shot string I_{str} is defined as $I_{str} = \sum_{j=1}^{x} v_j$, v_j is importance values of videos shot sh_j.

2. A *non repetitive shot string* (nrs string) is defined as a video shot string $\{sh_1....sh_x\}$, $\forall i, j \in \{1...x\}$, $lb_i \neq lb_j$;

3. A *k-non repetitive shot string* (k-nrs string) is defined as a non repetitive shots string with length k. We use $\{k\text{-}nrs_j\}$ to denote a set of nrs string with length k.

4. If str_i is the sub-string of str_j, we say that str_j *covers* str_i. For example, the 4-nrs string 3124 covers 2 2-nrs strings $\{312, 124\}$, 3 2-nrs strings $\{31, 12, 24\}$ and 4 1-nrs strings $\{3, 1, 2, 4\}$.

nrs strings carries important information about how the video editor arrange the video shots. We can easily find all $k - nrs$ string by scanning the video label string. Then we use them as skimming candidates. Some sample nrs strings are shown in Figure 6.7:

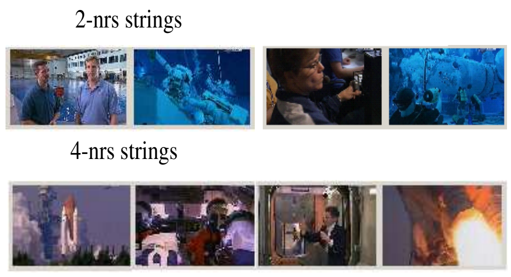


Figure 6.7: Several detected nrs string in a movie scene

To ensure the balanced content coverage, we hope that the skimming shots is able to cover as many semantically important shots as possible. To guarantee the coherence of the video skimming, we hope that in the skimming, we can pick more longer substrings from the video shot list. Thus the k-nrs strings become good candidates for video skimming since they are composed by video shots depicting non repetitive contents, and they are a coherent part of the original video. By scanning the video shot string we can easily get all k-nrs strings for all k.

We then formulate the video skimming generation problem as follows:

Problem 6 For a video scene, given the target skimming length L_{vs}, a set of video shots $\{sh_1...sh_n\}$ contained in the scene, the corresponding video shot length set $\{l_i\}$, and the corresponding video shot group label set $\{lb_1...lb_n\}$, find a continuous nrs string set $SKIM = \{nrs_j\}$, such that:

1. $\sum_j I_{nrs_j}$ is maximized (semantic importance summation is maximized);
2. $|SKIM|$ is minimized;
3. Minimize the duplicated items in $SKIM$;
4. $\sum_j (l_{nrs_j}) = l_t$;

To solve the above problem, we propose a greedy method algorithm, which is described in Algorithm 1:

Algorithm 7 Video skimming selecting algorithm

```
Input: The set of all $nrs$ strings $NRS$; The target skimming length $L_{vs}$;
Output: The selected $nrs$ set $SKIM$ that form the video skimming
BEGIN $SKIM = \emptyset$
STEP 1: Sort the $nrs$ strings in $NRS$ according to their importance value;
while $L_{vs} > 0$ do
  Select the best $nrs$ string $nrs_{opt}$, such that:
   1.   $L_{nrs_{opt}} < L_{vs}$
   2.   $\forall nrs_i \in N$ and $L_{nrs_i} < L_{vs}$, $I_{nrs_{opt}} \geq I_{nrs_i}$
  if Found then
   1.      $SKIM = S \cup \{nrs_{opt}\}$
   2.      $L_{vs} = L_{vs} - L_{nrs_{opt}}$
   3.      $NRS = NRS - \{nrs_t | nrs_{opt}$ covers $nrs_t\}$
  else if Not found then
    GOTO END
  end if
end while
END
```

Algorithm 1 continue selecting the most uncovered important nrs strings into the video skimming, and discard the already covered short nrs strings so that the semantic important contents are selected while the redundancy of the video skimming is minimized. By this algorithm we obtain a set of coherent video segments as the video skimming, such that the content coverage and coherency can be simultaneously achieved.

6.5 Experiments

To test the performance of our proposed approach, we have implemented the proposed video annotation and summarization framework then apply it to some movie clips. We employed a PC with 2.0G hz P4 CPU and 512Mb RAM on the Win2000 OS as the test bed. The weight parameter k is set to 0.6, and the time threshold t_1 is set to 4 seconds. Three movie clips and 1 sitcom clip are processed, and two skimmings generated at skim rate 0.15 and 0.30 are extracted for each test clip. Details about the video clips are shown in Table 6.5.

To evaluate the quality of the generated video skimming, we employ two criterion: *meaningfulness* and *favorite*. Since it is hard to objectively evaluate a video skimming, we use the following subjective test to compare the performance of our new video skimming generation scheme and the method we proposed in [36]. We have invited 10 test users to watch the video skimming generated from the video by the two methods at skim rate 0.15 and 0.30. To evaluate meaningfulness, the test users are asked to answer several questions about the key events that the video depicts(who has done what?). The scores are scaled to $[0, 100]$. To compare the favorite, we ask the user to select a "better" video skimming between the video skims generated by the two approaches, and the number of users who choose the skim as "better" is recorded as the favorite score. Figure 6.8 and Figure 6.9 show the average meaningfulness and favorite scores for the video skims generated by our proposed method and method in [36] respectively. The numbers of continuous video segments according to the original video that the video skims contain are also employed as a measure of coherence. The experimental results are also shown in Table 6.5 (Mfn. means Meaningfulness, Fav. means favorite, N.S. means Number of Segments, SEM means the new semantic approach, GRA means our old graph based approach).

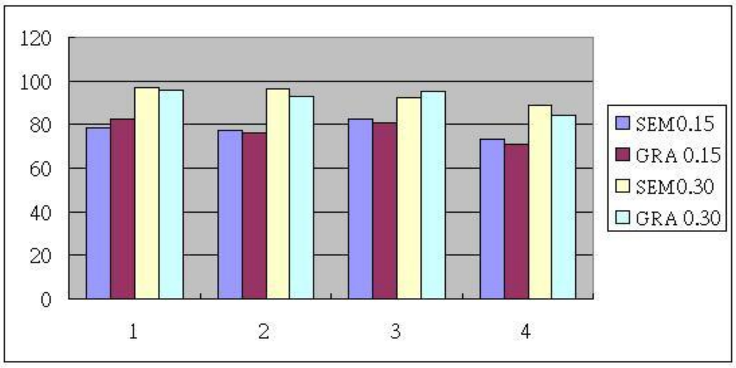


Figure 6.8: Meaningfulness Scores

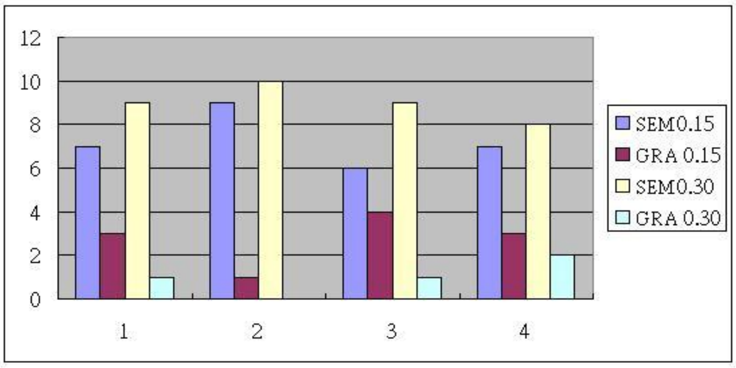


Figure 6.9: Favorite Scores

Our experimental results are quite encouraging. In terms of the meaningfulness, at the skim rate 0.15, the proposed semantic video summarization method obtain a quite high mean score 77.95. At skim rate at 0.30, the score achieved is even higher. Moreover, in most cases, our new semantic approach

Video Clip	**Duration**	**Actors**	**Events**	**Skim Rate**	**Mfn.**	**Fav.**	**N.S.**
Movie1	1403 sec.	9	7	0.15	**78.5**/82.3	**7**/3	**15**/59
				0.30	**97.1**/95.6	**9**/1	**22**/89
Movie2	1230 sec.	7	8	0.15	**77.5**/ 76.4	**9**/1	**16**/44
				0.30	**96.2**/ 92.9	**10**/0	**22**/65
Movie3	477 sec.	6	4	0.15	**82.5**/ 80.5	**6**/4	**12**/30
				0.30	**92.5**/95.0	**9**/1	**19**/46
Sitcom1	1183 sec.	8	9	0.15	**73.3**/71.1	**7**/3	**24**/54
				0.30	**88.8**/84.3	**8**/2	**46**/87
Average	—	—	—	0.15	**77.95**/77.57	**7.25**/2.75	—
				0.30	**93.65**/91.65	**9**/1	

Table 6.1: User test results. Scores for the new approach are **bold**

has gained a higher score than our previous graph-optimization approach. In terms of favorite, we can see that although both video skimming is meaningful, at both skim rates, most users would prefer the video skimming generated by the new method. The major reason should be that the new skim is more coherent. We also find that our new approach generates much less video segments than the previous approach, which greatly increase the coherence. From the experimental results we can make the conclusion that our proposed method is able to generate a better video skimming in comparison with our previous work.

6.6 Summary

In this chapter, we illustrate a novel framework for semantic video summarization. We provide the users a semi-automatic system to help them annotating the video semantic contents efficiently. Then we combine the semantic information and structure information of the video, compute the semantic importance for each video shot, analyze the arrangement patterns of the video shots. Finally, we obtain a dynamic skimming by selecting the key video shot strings. The experimental results show that our approach ensures both the balanced content coverage and visual coherence. Experimental results show that the framework is effective in generating good quality video skims.

Chapter 7

Conclusion remarks

7.1 Summary

Video is getting more and 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. Many organizations produce huge volume of video data everyday. Facing the massive data volume, end users find that it is inefficient to browse a favorite video from the Internet, and the content providers have to face the tedious work of managing the ever growing video database. The urgent problem brings a lot attention to video summarization, which is a new technology intends to solve the problem by providing the people with concise and informative content presentations so that the users can quickly grasp the major contents of a video.

Most video summaries goes into the following two types: *static video story board*, which is composed of a set of salient images extracted or synthesized from the original video, and *dynamic video skimming*, which is a shorter version of the original video made up of several short video clips.

This thesis presents our work done on automatic video summarization. We first specify three goals that a video summary with good quality should achieve:

1. **Conciseness**–For conciseness, the length of the generated video skimming should be within the user-specified length L_{vs}; the static summary

should not contain too many images.

2. **Balanced content coverage**–As the video is a structured document, the video skimming should be able to represent the original contents with balance. At the same time, both the visual diversity and the temporal coverage of the original contents should be reflected by the video skimming.

3. **Visual coherence**–One problem for traditional video skimming generation is that the user often feel that the video skimming is quite choppy. A good video skimming should increase the coherence of the video skimming while preserving the content coverage.

In conclusion, we have proposed a automatic video content analysis and summarization framework. First, analyze the structure of the video, based on the structure, we create video summaries that is able to achieve the above three goals. Our research work also comprises some initial work on video semantic content annotation and semantic video summarization. To obtain and utilize the semantic information of the video, a semi-automatic video annotation system is built and video summary are generated based on the semantic content descriptions.

Our research work has the following contributions:

1. We have proposed several targets that a video summary with good quality should have, and we have achieve them in our framework.

2. We analyze the intrinsic video shot-shot group-scene structure of the video and employ the structure information to help us in video skimming generation. The balanced content coverage is thus guaranteed.

3. We model the video scenes into a graph based on the video shots contained in it, and select video shots by searching a constrained longest

path in that graph. Our method is able to cover both the visual content coverage and the temporal content distribution simultaneously.

4. We propose a semantic video annotation and summarization framework to help the user to annotate the video shots semi-automatically. An importance measure is derived from the semantic content descriptions based on mutual reinforcement principle. Video summaries are then generated with the semantic descriptions. The performance of our new work is compared with our previous work.

7.2 Future work

In this paper we describe the work we have done on video content analysis and summarization. A graph-optimization based video summarization framework and a semantic video summarization framework have been proposed. The frameworks themselves are quite flexible; many other features and constraints can be added into this framework as its extension. In the future, we may enhance the system by incorporating better feature analysis technique into our framework.

Currently our system provides only limited interactivity for the user. Our framework can be further extended by incorporating the user defined preference. The user may specify what he is specially interested in, or what he is not interested in to make a personalized video summary that is specially useful to himself. Different ways for user to describe and express his preference during interaction with the video summarization system will be investigated.

For video semantic recovery, one possible way to obtain video semantic description is using the video production documents. During video document production, a lot of planning work and paper materials is produced, like movie screenplays, scripts, plans, etc. The information contained in them contains time, location, order, actor, dialog script, background music, etc., which is detailed enough for further processing. Moreover, recovering such detailed information from produced video automatically is far beyond up-to-date intelligent systems' abilities. So one good practical way to resolve the semantic gap and obtain accurate semantic video descriptions is to find a method to better incorporate Mpeg-7 during video production and we can directly obtain video semantic description with very low cost, which is a alternative to obtain reliable video semantic descriptions.

Bibliography

[1] A. Aner and J. R. Kender. A unified memory-based approach to cut, dissolve, key frame and scens analysis. In *Proceedings of the IEEE International Conference on Image Processing*, 2001.

[2] A. Aner and J. R. Kender. Video summaries through mosaic-based shot and scene clustering. In *Proceedings of European conference on Computer Vision*, number 2353 in LNCS, pages 388–402, 2002.

[3] N. Bagaguchi. Generation of personalized abstract of sports video. In *Proceeding of IEEE International Conference on Multimedia and Expo*, pages 800–803, 2001.

[4] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. In *Proceedings of SPIE Storage and Retrieval for Still Image and Video Databases IV*, pages 170–179, 1996.

[5] Broadband4Britain. Uk broadband usage survey. *http://www.net.com/pdf/BB4Britian-ir.pdf*, 2003.

[6] Z. Cernekova, C. Nikou, and I. Pitas. Entropy metrics used for video summarization. In *Proceedings of the 18th spring conference on Computer Graphics*, pages 73–82, 2002.

[7] P. Chang, M. Han, and Y. H. Gong. Extract highlights from baseball game video with hidden markov models. In *Proceedings of the 2002 IEEE conference on Image Proceesing*, pages 609–612, 2002.

[8] S. F. Chang and H. Sundaram. Structure and semantic analysis of video. In *Proceedings of the 2000 IEEE Conference on Multimedia and Expo*, 2000.

[9] P. Chiu, A. Girgensohn, W. Polak, E. Rieffel, and L. Wilcox. A genetic algorithm for video segmentation and summarization. In *Proceedings of the 2000 IEEE International Conference on Multimedia and Expo*, pages 1329–1332, 2000.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C.Stein. Introduction to algorithms. *The MIT Press*, 2001.

[11] Creativepro.com. Popcast selected by mgi to offer personal broadcasting services to mgi videowave 4 users. *http://www.creativepro.com/story/news/10207.html*, 2000.

[12] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *Proceedings of ACM Multimedia 98*, pages 13–16, 1998.

[13] P. England, R. B. Allen, M. Sullivan, and A. Heybey. I/browse: The bellcore video library toolkit. In *Proceedings of 1996 SPIE conference*, volume 2620, pages 254–264, 1996.

[14] J. P. Fan, X. Q. Zhu, and L. D. Wu. Automatic model based semantic object extraction algorithm. *IEEE Transaction on Circuits and Systems for Video Technology*, 11(10):1073–1084, 2001.

[15] M. Fayzullin, V. S. Subrahmanian, A. Picariello, and M. L. Sapino. The cpr model for summarizing video. In *Proceedings of the Proceedings of the*

first ACM international workshop on Multimedia databases, pages 2–9, 2003.

[16] A. M. Ferman and A. M. Tekalp. Multiscale content extraction and representation for video indexing. *Proceedings of SPIE*, 3229:23–31, 1997.

[17] A. M. Ferman and A. M. Tekalp. Efficient filtering and clustering methods for temporal video segmentation and visual summarization. *Jounal of Visual Communication and Image Representation*, 9(4):336–51L, 1998.

[18] A. M. Ferman and A. M. Tekalp. A fuzzy framework for unsupervised video content characterization and shot classification. *Multimedia Tools and Applications*, 1:89–111, 1999.

[19] Y. H. Gong and X. Liu. Video summarization with minimal visual content redundancies. 2001.

[20] Y. H. Gong and X. Liu. Video summarization and retrieval based on singular value decomposition. *Multimedia Systems*, 9(2):157–168, 2003.

[21] M. Greg. Film production : the complete uncensored guide to independent filmmaking. *Los Angeles, CA : Lone Eagle Pub*, 1998.

[22] A. Hanjalic, L. Langendijk, and J. Biemond. Automatic high-level movie segmentation for advanced video retrieval systems. *Multimedia Tools and Applications*, 9(4):580–588, 1999.

[23] A. G. Hauptmann and M. A. Smith. Text, speech, and vision for video segementation: The informedia project. In *Proceedings of the AAAI Fall Symposium on Computer Models for Intergrating Language and Vision*, pages 213–220, 1995.

[24] L. He, E. Sanocki, A. Gupta, and J. Grudin. Auto summarization of audio-video presentations. In *Proceedings of 1999 ACM conference on Multimedia*, pages 489–498, 1999.

[25] G. W. Heiman, R. J. Leo, G. Leighbody, and K.Bowler. Word intelligibility decrements and the comprehension of time-compressed speech. *Journal of Perception and Psychophysics*, 40(6):407–411, 1986.

[26] C. L. Huang and C. Y. Chang. Video summarization using hidden markov model. In *Proceedings of the 2001 IEEE International Conference on Information Technology: Coding and Computing*, pages 348–351, 2001.

[27] S. X. Ju, M. J. Black, S. Minneman, and D. Kimber. Summarization of video-taped presentations: Automatic analysis of motion and gestures. *IEEE Transaction on Circuits and Systems for Video Technology*, 8(5):686–696, 1998.

[28] R. L. Lagendijk, A. Hanjalic, M. Ceccarelli, M. Soletic, and E. Persoon. Visual search in a smash system. In *Proceedings of the 1996 IEEE Conference on Image Processing*, pages 671–674, 1996.

[29] M. Lee, W. Chen, C. Lin, C. Gu, and T. Markoc. A layered video object coding system using sprite and affine motion model. *IEEE Transactions on Circuits and Systems for Video Technology*, 1:130–145, 1997.

[30] R. Leinhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communication of the ACM*, pages 55–62, December 1997.

[31] B. X. Li, H. Pan, and I. Sezan. Comparison of video shot boundary detection techniques. In *Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 169–172, 2003.

[32] R. Lienhart and A. Hartmann. Classifying images on the web automatically. *Journal of Electronic Imaging*, 11(4):40–52, October 2002.

[33] Rainer Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, 1(3):469–486, 2001.

[34] H. Y. Liu, R. Chellappa, and A. Rosenfeld. Accurate dense optical flow estimation using adaptive structure tensors and a parametric model. *IEEE Transaction on Image Processing*, 12(10):1170–1180, 2003.

[35] S. Lu, I. King, and M. R. Lyu. Video summarization using greedy method in a constraint satisfaction framework. In *Proceedings of 9th International Conference on Distributed Multimedia Systems*, pages 456–461, 2003.

[36] S. Lu, I. King, and M. R. Lyu. Video summarization by video structure analysis and graph optimization. In *Proceedings of The 2004 IEEE International conference on multimedia and expo*, 2004.

[37] S. Lu, I. King, and M. R. Lyu. A novel video summarization framework for document preparation and archival applications. In *Proceedings of the 2005 IEEE Aerospace Conference, to appear*, 2005.

[38] S. Lu, M. R. Lyu, and I. King. Semantic video summarization using mutual reinforcement and shot arrangement patterns. In *Proceedings of the 11th International conference on Multimedia Modeling, to appear*, 2005.

[39] Y. F. Ma, L. Lu, H. J. Zhang, and M. J. Li. A user attention model for video summarization. In *Proceedings of ACM Multimedia*, pages 533–542, 2002.

[40] M. Massey and W. Bender. Salient stills: Process and practice. *IBM System Journals*, 35:25–37, 1996.

[41] M. Mills. A magnifier tool for video data. In *Proceedings of ACM Human and Computer Interface (CHI)*, pages 93–98, 1992.

[42] M. R. Naphade, S. Basu, J. R. Smith, C. Y. Lin, and B. Tseng. Modeling semantic concepts to support query by keywords in video. In *Proceedings of 2002 IEEE International Conference on Image Processing*, pages 145–148, 2002.

[43] M. R. Naphade and T. S. Huang. Extracting semantics from audio-visual content: the final frontier in multimedia retrieval. *IEEE Transactions on Neural Networks*, 13(4):793–810, 2002.

[44] M. R. Naphade, I. V. Kozintsev, and T. S. Huang. A factor graph framework for semantic video indexing. *IEEE Transaction on Circuits and Systems for Video Technology*, 12(1):40–52, January 2002.

[45] S. Nepal, U. Srinivasan, and G. Reynolds. Automatic detection of goal segments in basketball videos. In *Proceedings of the 2001 ACM Conference on Multimedia*, pages 261–270, 2001.

[46] Chung Wing Ng. Advise: Advanced digital video video information segmentation engine. *Master thesis in The Chinese University of Hong Kong*, 2002.

[47] C. W. Ngo, Y. F. Ma, and H. J. Zhang. Automatic video summarization by graph modeling. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, pages 104–109, 2003.

[48] C. W. Ngo, T. C. Pong, and R. T. Chin. Video partitioning through temporal slices analysis. *IEEE Transaction on Circuits and Systems for Video Technology*, 1(3):445–469, 2001.

[49] J. M. Odobez, D. G. Perez, and M. Guillemot. Spectral structure of home videos. In *Proceedings of the 2003 Conference on Image and Video Retrieval*, pages 310–320, 2003.

[50] S. Pfeiffer, R. Leinhart, S. Fisher, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, December 1996.

[51] D. Ponceleon and A. Amir. Cuevideo: Automated multimedia indexing and retrieval. In *Proceeding of ACM Multimedia*, 1999.

[52] Y. Rui, A. Gupta, and A. Acero. Automatically extracting highlights for tv basketball programs. In *Proceeding of ACM Multimedia*, pages 105–115, 2000.

[53] Y. Rui, T. S. Huang, and S. Mehrotra. Constructing table-of-content for videos. *ACM Multimedia Systems Journal, Special Issue Multimedia Systems on Video Libraries*, 7(5):359–368, Sept 1999.

[54] D. M. Russell. A design pattern based video summarization technique: Moving from low-level signals to high-level structure. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pages 248–251, 2000.

[55] H. Schweitzer. Computing content-plots for video. In *Proceedings of European conference on Computer Vision*, number 2353 in LNCS, pages 491–501, 2002.

[56] B. Shahraray and D. C. Gibbon. Automatic genneration of pictorial transcripts of video programs. In *Proceedings of 1995 SPIE and IS&T Conference*, pages 512–519, 1995.

[57] J. B. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern analysis and Machine Intelligence*, 22(8):40–52, August 2000.

[58] F. Shipman, A. Girgensohn, and L. Wilcox. Generation of interactive multi-level video summaries. In *Proceedings of 2003 ACM conference on Multimedia*, pages 392–401, 2003.

[59] M. A. Smith and T. Kanade. Video skimming and charaterization through the combination of image and language understanding techniques. In *Proceedings of the IEEE Intenational Conference on Computer Vision and Pattern Recognition*, pages 775–781, 1997.

[60] M. G. Christel M. A. Smith and C. R. Taylor. Evolving video skims into useful multimedia abstractions. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 171–178, 1998.

[61] H. Sundaram and S. F. Chang. Constrained utility maximization for generating video skims. In *Proceedings of the 5th IEEE Workshop on Content based Acess of Image and Digital Libraries*, 2001.

[62] H. Sundaram and S. F. Chang. Computable scenes and structures in films. *IEEE Transaction on Multimedia*, 4(4):482–491, 2002.

[63] H. Sundaram and S. F. Chang. Video skims: Taxonomies and an optimal generation framework. In *Proceedings of 9th IEEE Conference on Image Processing*, 2002.

[64] H. Sundaram, L. Xie, and S. F. Chang. Condensing computable scenes using visual complexity and film syntax analysis. In *Proceedings of the 2001 IEEE Conference on Multimedia and Expo*, 2001.

[65] H. Sundaram, L. Xie, and S. F. Chang. A utility framework for the automatic generation of audio-visual skims. In *Proceedings of the ACM Multimedia*, pages 189–198, 2002.

[66] Y. Taniguchi, A. Akutsu, Y. Tonomura, and H. Hamada. An intuitive and efficient access interface to real-time incoming video based on automatic

indexing. In *Proceedings of the third ACM international conference on Multimedia*, pages 25–33, 1995.

[67] W. Tavanapong and J. Y. Zhou. Shot clustering techniques for story browsing. *IEEE Transaction on Multimedia*, 6(4):517–527, 2004.

[68] C. Toklu and S. P. Liou. Automatic key frame selection for content based video indexing and acess. In *Proceedings of SPIE for Storage and Retrieval for Media Databases 2000*, 2000.

[69] C. Toklu and S. P. Liou. Image and audio sequence visualization and interaction mechanisims for structured video browsing and logging. In *Proceedings of the 2000 IEEE Conference on Image Processing*, pages 396–399, 2000.

[70] C. Toklu, S. P. Liou, and M. Das. Video abstract: A hybrid approach to generate semantically meaningful video summaries. In *Proceedings of the 2000 IEEE Conference on Multimedia and Expo*, pages 268–271, 2000.

[71] B. L. Tseng, C. Y. Lin, and J. R. Smith. Video summarization and personalization for pervasive mobile devices. In *SPIE Electronic Imaging 2002 - Storage and Retrieval for Media Databases*, pages 383–392, 2002.

[72] S. Uchihashi, J. Foote, A. Girgensohn, and J. Boreczky. Video manga: Generating semantically meaningful video summaries. In *Proceedings of ACM Multimedia 99*, pages 383–392, 1999.

[73] N. Vasconcelos and A. Lippman. A spatial-temporal motion model for video summarization. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 361–368, 1998.

[74] A. Verri and T. Poggio. Motion field and optical flow: qualitative properties. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

[75] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transaction on Image Processing*, 3:99–109, 1994.

[76] H. Yang, L. Chaisorn, Y. L. Zhao, S. Y. Neo, and T. S. Chua. Video qa: Question answering on news video. In *Proceedings of the 2003 ACM Conference on Multimedia*, pages 632–641, 2003.

[77] M. Yeung, B. L. Yeo, and B. Liu. Extracting story units from long programs for video browsing and navigation. In *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 296–305, 1996.

[78] M. M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *Proceedings of the 1995 IEEE International Conference on Image Processing*, pages 348–351, 1995.

[79] B. Yu, W. Y. Ma, K. Nahrstedt, and H. J. Zhang. Video summarization based on user log enhanced link analysis. In *Proceedings of 2003 ACM conference on Multimedia*, pages 382–391, 2003.

[80] J. Yu and M. D. Srinath. An efficient method for scene cut detection. *Elsevier Pattern Recognition Letters*, 22:1379–1391, 2001.

[81] H. Y. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of 2002 Annual International ACM SIGIR Conference*, January 2002.

[82] H. J. Zhang, C. Y. Low, and S. W. Smoliar. Video parsing and browsing using compressed data. *Multimedia Tools and Applications*, 1:89–111, 1995.

[83] H. J. Zhang, D. Zhong, and S. W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, 1997.

[84] X. Q. Zhu, J. P. Fan, A. K. Elmagarmid, and X. D. Wu. Hierarchical video content description and summarization using unified semantic and visual similarity. *ACM/Springer Multimedia Systems Journal*, 9(1):31–53, 2003.

[85] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *Proceedings of the 1998 IEEE International Conference on Image Processing*, pages 73–82, 1998.