Video summarization by video structure analysis and graph optimization

Shi Lu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR
slu@cse.cuhk.edu.hk

November 26, 2003

Abstract

Video over Internet is pervasive nowadays. Since downloading and browsing through the whole video file is time consuming, video summarization techniques, which aim at providing a way for the user to grasp the major content of the video without viewing the whole video, has received more and more attention. In this paper we propose a novel method for moving video skimming generation that combines video structure analysis and graph optimization. First, we segment the video into video shots, build up the video structure, determine the scene boundaries; Second, we figure out the target skimming length for each video scene according to its length and complexity; Finally, we model each video scene into a graph then select the final video skimming shots by doing optimization in that graph. Experimental results show that our approach preserves the scene level structure of the original video and ensures balanced coverage of the original video content.

1 Introduction

With the rapid growth of network bandwidth and high-capacity storage devices, videos have become pervasive nowadays. However, the abundance of video data gives rise to a new challenge: since it is time-consuming to download and browse most parts of a video before we know the video contents, it is difficult to find out a video file that we want from the vast video repositories. Moreover, in some cases in which we do not have enough bandwidth to stream all the original video, e.g, some wireless hand-hold devices using MMS (Multimedia message services), a digest version of the video is needed to meet the limited bandwidth. Therefore, in order to help the user to grasp the essence of the video quickly, *video summarization* has received more and more attention.

Video summarization is a short summary of the content of a longer video document. Specifically, it is a condensed sequence of still or moving images representing a video in such a way that the user is provided with concise presentation of the video content. There are two different kinds of video summarizations:

1. **Still-image Storyboard**– The still-image representation is composed of a collection of salient images extracted or synthesized from the underlying video source. For example, some MMS service provides video summarization services to the users by delivering still-image storyboard accompanied with audio track.

2. **Moving-image Skimming**– The moving-image representation, also called video skimming, is made up of a set of video clips. Movie trailer is a good example for moving-image skimming.

To generate a perfect video summary 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 some low-level features to generate video summaries.

From the user's point of view, a video summary with good quality should have the following attributes:

1. **Conciseness**–For dynamic video skimming, the length of the final skimming should not exceed the given target length L_{vs}; For static story board, there should not be too many images else the user may get distorted.

2. **Comprehensive coverage**–To ensure the generated video summary is informative, both the visual diversity and temporal distribution of the original video should be covered;

3. **Visual coherence**–For moving video skimming, jumpy feel is mainly caused by too frequent scene change. A coherent video skimming is for sure more preferable to the user.

In this paper we describe a novel two-stage video summary generation method that combines video structure analysis and graph optimization. Both the moving and static video summaries will be generated. We first segment the original video into video shots, analyze its structure and determine the scene boundaries, calculate the distribution of the video skimming quota according to the video scene importance, then we model the whole video shot set as a directional graph then select several video shots from the original shot set by doing optimization on that graph. Experiments show that our method is able to meet the three objectives. Fig. shows the flowchart of our approach.

The paper is organized like follows: In section 2 we review some related work done in video summarization field in recent years. In section 3 we describe our method to analyze the video structure and determine the target skim length of each video scene. In section 4 we will describe our two-stage video summary generation problem, give our solution and algorithms to it. In section 5 we show our experiment results and make some discussion. In section 6 we make conclusion and discuss our future work.

2 Literature Review

In these years many work was done on video summarization. For video skimming generation, in the VAbstract system [1], key movie segments are selected to form a movie trailer. The Informedia system at CMU first generates caption text from the audio stream of the video by speech recognition, then selects the video segments according to the occurrence of important keywords in the video sequence [2]. However, selecting video shots solely by the occurrence of keywords cannot ensure that the video summary will cover the main idea of the original video. The CueVideo system from IBM provides faster playback speed when playing long, static video scenes and slower speed for short, dynamic video scenes [3]. Although the playback time has been reduced but the temporal property of the original video is distorted, which may mislead the users.

Later works employ some perceptional important feature to generate video skimmings. In [4], a user attention curve is constructed to simulate the user's attention toward visual, audio, linguistic contents in the video, and a moving video skimming and a static image presentation is created according to the maximum points of the user attention curve. In our *previous VS paper*, each considered feature(human face, caption text, gunshot, fire color) is assigned a weight score and the video segments that maximize the summation of the weight score is regarded as the final video skimming. To increase the visual coherence of the video skimming, an refinement process is conducted to merge the short segments. However, the method does not consider the video structure, and temporal coverage. [5] proposed to generate a video summary with least information redundancy. In [6] the authors defines an utility function for each video shot, then propose a utility maximization framework to generate video skims. In [7], the video shots are modelled into a graph then the video structure was attained by graph partitioning. Then a video skimming is generated based on the detected structure. In broadcasted sports video, highlight detection and extraction have been achieved in basketball videos [8] and baseball videos [9]. Highlight detection is highly dependent on domain-specific knowledge, and it is not a general solution. The methods listed above are either domain-specific or focus on only features, but neglect the coverage of the whole content of the video.

For still-image-based static summary, many key frame extraction methods also have been proposed in these years. Most of the early work selects key frame images by random or uniform sampling, like the MiniVideo systems [10]. Later work tends to extract key frame images by adapting to the dynamic video content. According to the applied features, we can categorize the methods into color-based approach like [11, 12], motion-based

approach like pixel-based image difference [13], optical flow [14], and mosaic-based approach [15]. In [16] and [17], the authors made video segmentation and analyzed the video structure to get a tree-structured Video-Table-Of-Contents(V-TOC). In [18], the authors proposed an importance measure for each selected frame, and a frame packing algorithm to adjust the shown image size of each selected key frame according to its importance. Still image story boards can be constructed faster but their descriptive ability is limited for they cannot convey the temporal and in many cases the audio information of the original video is not preserved.

Edited video has its intrinsic structures. In [16], the authors build a V-TOC(Video Table Of Contents) to describe the structure of video, [19] build a scene transition graph to describe the story line in the video. [7] uses a graph partitioning approach to group the visually similar video shots then determine the scene boundaries.

A still-image summarization does not convey any temporal properties of the video, while the moving-image summarization, may make more senses and more attractive to the user. In this paper we mainly focus on generating moving image skimmings, although we also generates a static video summary. In this paper we propose a two stage video skimming generation method that combines the video structure analysis and graph based optimization. Given the target video skimming length, we first determine the scene boundaries in the original video, then we determine each scene's skim length according to their length and complexity. Our approach preserves the scene-level structure of the video and can ensure the balanced content coverage.

3 Video structure analysis

Edited video has its intrinsic structures. From bottom to up, the video can be decomposed into video shots, each of them is a coherent image sequence captured uninterruptedly by a single camera. Temporally adjacent similar shots form a video shot group, while the video scene is composed by intersecting video shot groups or a series of continuously visually different video shots.

A video narrates a story like a article does. A video contains several video scenes, each of them depicts an event 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.

In this section we will describe how we analyze the video structure then use it to help us to determine each scene's target skim length. 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, then based on the video shot groups we can construct video scenes, so that a three layered video structure is constructed.

3.1 Terms and definitions

Here we give definition for some terms we use in this section. We followed the terms in [16].

1. **Video shot**: A video shot sh_i is a video segment uninterruptedly captured by a camera. It is the building block of edited videos. The length l_{sh_i} of video shot sh_i is the number of image frames it contains.

2. **Key frame**: The visual content of a video shot can be represented by its key frames. We use the first frame kf_{begin_i} and last frame kf_{end_i} of the video shot sh_i as its key frames.

3. **Video shot group** A video shot group Sg_i 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.

4. **Video scene**: A video scene is the intermediate entity between video shot groups and the whole video. It is composed by several intersecting video shot groups or a 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 also the summation of the video shot length that it contains.

5. **Video**: The video contains all elements above.

3.2 Video shot boundary 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.

To measure the similarity of two images, traditional methods use the frame difference, and color histogram. Frame difference is easy and quick to compute, but visually similar images may have a movement so that directly computed frame difference is very sensitive to camera motion. To overcome this we may search a proper offset to compensate the motion, which will be quite time consuming. Another metric is the color histogram. Since the color histogram is derived from the statistic of the original image, it can only describe the composition of the image but does not contain any information about how the image looks, this may lead to some mis-detections. Regional color histogram is also sensitive to camera motion.

To find a efficient and accurate method to detect video shot boundaries, we extract a *video slice image* [**?**] from the original video then detect cuts by analyze the video slice image. A video slice image is a spacial sampling of the video over the temporal axis, it 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 is shown in Fig 1:

We can choose whatever fixed line on the video image to generate a video slice. But now we choose the center horizontal line of the image to generate the video slice. This is because when a video is recorded, the camera normally moves in the horizontal plane, and the horizontal panning happens more frequently than the vertical panning. Another 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.

With the slice image generated, we can measure the similarity of the consequent video images by measuring the similarity of the columns in the slice image. The measure we use to measure the similarity between consequent pixel rows is *minimum pixel difference*.

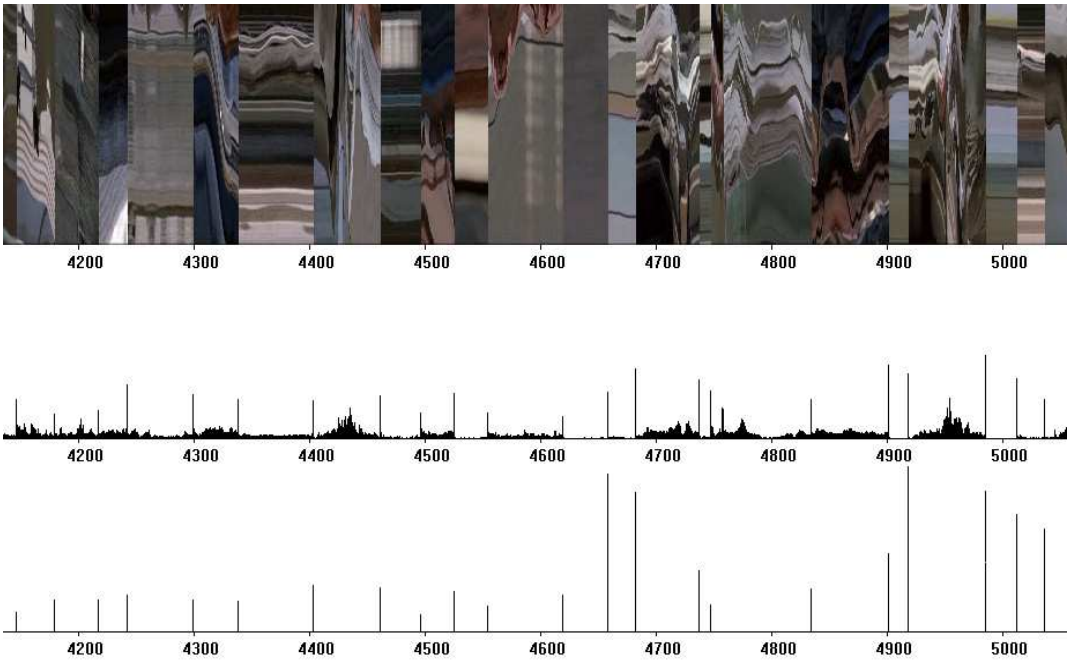

Figure 1: A video slice

Suppose pixel row r_i and r_{i+1} contain n pixels, the minimum difference between the i_{th} row and the $i + 1_{th}$ row is computed like follows:

$$D_{min}(i) = \min_{x=-m}^{m} (\sum_{j=1}^{n} (|r_i(j) - r_{i+1}((j + x) \mod n)|))$$

. 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 Fig 1.

From the calculated difference function we can see that under normal situation without intense motion, the minimum difference seems to be good for shot cut detection, for with a global thresholding we can find those cut points. There will be a sudden jump in the feature function, with the width is exact equal to 1. But when there is a intense motion or luminance change like the flash of cameras, the curve goes worse like shown in Fig. 2

In this cases, performing global thresholding on the calculated difference function will get poor results with the presence of noises caused by motion and luminance change.

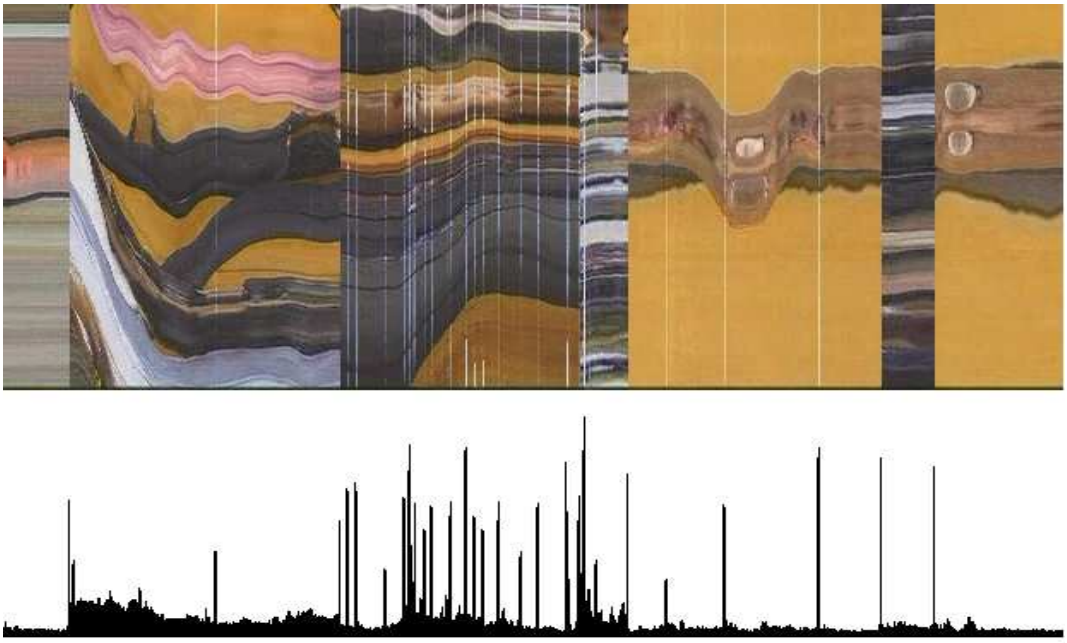

Figure 2: Flash effect

The shot cut point we want should have two properties: first, it is a local sudden jump; second, since the two video shots are coherent, the width of the jump should be only 1. Based on these properties we devise the following window-based order ratio filtering to gain robustness under noisy circumstance:

$$if D_i > threshold_1$$

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

, w is the half width of the window, and $threshold_1$ is a prior threshold for possible video cuts.

By applying this filtering, the transformed function $D'(i)$ in those parts where the difference function is lower than $threshold_1$ will be 0, in most parts, if there is not a local maximum, the value will be less than 1; for those local maximum, the transformed value will be more than 1. By dividing the "second largest value" in the window, the flash effect, which often causes a sudden jump with the width 2, will be mostly suppressed. Those noise caused by intense motion will also get filtered by this method. After that, by applying a global threshold greater than 1, say 1.5, we will find most of

the video cuts, and filter out most of the noises. Example of the effects of this filtering is shown in Fig. 3:

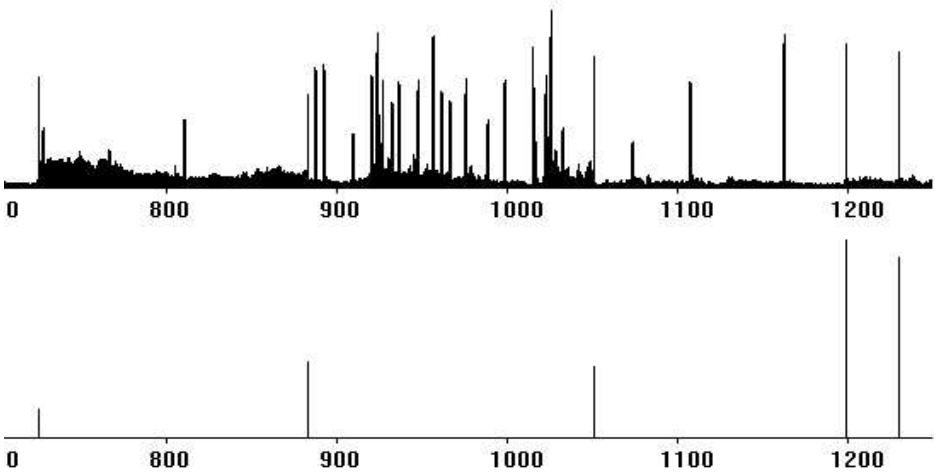

Figure 3: Refined least difference

After applying a global threshold on the transformed difference function, we can easily find the video cuts.

We tried the above window-based order filtering on several movie video segments and news video segment and get the following results shown in Table 1:

Table 1: Video cut detection for Movie clip

Video type	**Ground truth**	**Detected**	**F. D.**	**M. D.**	**Right Per.**
Movie	166	157	0	9	94.6
News	40	39	1	1	95
Movie	138	137	2	3	97.8

3.3 Constructing the video structure

The reason we use the shot-group-scene structure is that it is the sequence with which the edited video are build. The editor of the video place different video shots to form a video story, while we decompose the video then recover this structures.

With the video shots detected, we can continue building up the video shot groups and video scenes from the video shots we detected from the source video. A video shot group is composed by visually similar and temporal adjacent video shots. A video scene is composed by several related video shot groups, and then we can get the V-TOC (Video Table Of Contents) of the video, which is a hierarchy tree structure. An example of V-TOC is shown in Fig. 4.

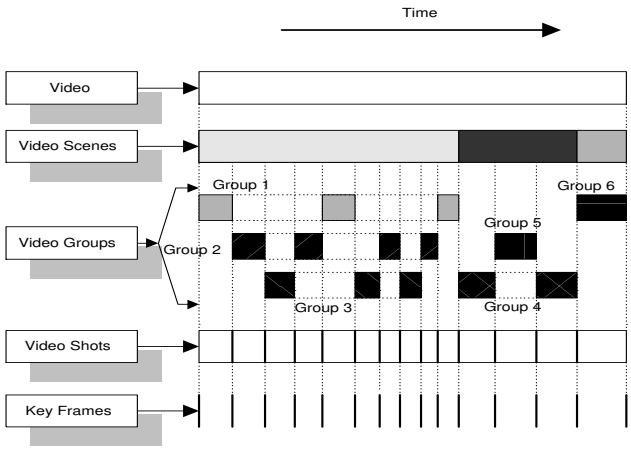

Figure 4: V-TOC tree structure

For building the video groups, we use the H-S histogram correlation between video shot key frames to measure the visual similarity between video shots, and we use the distance of center frames to measure the temporal distance between video shots. There is two requirements that one shot can be put into one shot group: First, the visual similarity between the video shot and the video shot group should be big enough; Second, the temporal distance between the video shot and the video shot group should be close enough. We use the similarity between the most similar video shot in the current video shot group and the current shot as the visual similarity between video shot and a video shot group, also, we use the distance between the nearest video shot in the video shot group and the current video shot as the distance between video shot groups and a video shot.

To build up video shot groups, we use the window sweeping algorithm described in [16], shown as Algorithm 1.

After we have determined the boundaries of the video shot groups, by finding the intersection between video shot groups, we can find the boundaries of video scenes. The algorithm for finding scenes from the group boundaries are omitted here.

Algorithm 1 Video shot group formation algorithm

Input: A set of video shots $S = \{sh_1...sh_n\}$, the visual similarity threshold t_v, the temporal distance threshold t_t.
Output: Output: A series of video groups $G = \{Sg_1...Sg_m\}$.
BEGIN
Add a video shot sh_1 to g_1, add g_1 to G;
for each sh_x in S **do**
  for each g_y in G **do**
    Calculate $similarity(g_y, sh_x)$
  end for
  Find $V_max(g_m, sh_x) = \min_y similarity(g_y, sh_x)$;
  if $V_min(g_m, sh_x) < t_v$ **then**
    if $d_T(g_m, sh_x) < t_t$ **then**
      Add sh_x to g_m
    else
      Add a new group g_{new} to G, assign sh_x to g_{new}.
    end if
  else
    Add a new group g_{new} to G, assign sh_x to g_{new}.
  end if
end for
END

3.4 Video scene styles

After the video groups are constructed, intersected video shot groups forms the video scenes. Video scenes can be classified into two types: loop scenes, which is composed by several intersected video shot groups; progressive scenes, which is composed by a series of different video shots. The loop scenes are mostly used to describe some events or situation, while the progressive scenes are often used to describe the transition between events. Example of this two kinds of scenes are shown in Fig 5. According to their different We will treat them differently in our video skimming generation.

3.5 Video scene skimming length calculation

Since we want the video skimming to balanced cover the contents of the source video, we should the distribute the video skimming according to the detected video structure. On the other hand, since the target skim length might be much shorter than the original video, we need to find the "key parts" of the video and discard some trivial parts. From a narrative point of view, a longer and more complex video scene might be more important.

Figure 5: Loop and progressive scenes

We now distribute the total skimming length into video scenes in a top-down level, then we select the video shots by graph optimization in the video shot level.

At the video scene level, intuitively, the more complex the video scene is, the longer should its skimming be. Also, the longer a scene is, the longer should its skimming be. We then distribute the total video skim length to each video scene.

Since the progressive scene is just a linear structure, we simply use its length to measure its importance.

For a loop video scene which is composed of several video groups, we use the content entropy to describe the complexity of a video scene, defined as follows:

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

The more complex the video scenes is, the higher will its entropy be. The complexity of a video scene also conveys the video editor's intention: he will place more video shots on those more important parts. Thus the longer and more complex video scenes should more skim length assigned to it.

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 video scene in the video like follows:

1. For each progressive scene Sc_x, $Sl_x = l_{Sc_x} \times r_s$. If S_x is less than preset threshold t_1, we discard this progressive scene for too short skim does not make any sense to people.

2. Suppose after the first round, the left skim length is L'_{vs}, for the loop scenes $\{Sc_1 ... Sc_n\}$, $Sl_i = L'_{vs} \times \frac{Entropy(Sc_i) \times l_{Sc_i}}{\sum_j Entropy(Sc_j) \times l_{Sc_j}}$. In a similar manner, we discard Sc_i if Sl_i is less than preset threshold t_2.

3. For the remain loop scenes $\{Sc'_1 ... Sc'_m\}$, we set $Sl_i = L'_{vs} \times \frac{Entropy(Sc_i) \times l_{Sc_i}}{\sum_j Entropy(Sc_j) \times l_{Sc_j}}$.

The above algorithm scatters the whole skim length to each scene, and, those more important scenes are better preferred. We discard those scenes whose skim length are too short to ensure the understandability of the key parts of the video. This mechanism ensures that the final video skimming is informative to the user.

After we have determined the target length of each of the video scenes, we can continue selecting video shots from each scene then form a video skimming.

4 Graph based video summarization

Video skimming with good quality should be able to comprehensively cover both the visual diversity and temporal distribution of the original video. At the same time, the length of the video skimming must not be too long. To generate a video skimming of the video that satisfies the above attributes, we need to discard some video shots while those remains compose the video skimming. Now that we have determined each scene's skim length, according to their duration and complexity. In this section, we will describe the way we model the video shot set into a spatial-temporal graph and find the video skimming shots by performing optimization on that graph.

4.1 Graph modelling

Given a set of video shots $S = \{sh_i ... sh_n\}$, we use the first frame $f_{i_{begin}}$ and the last frame $f_{i_{end}}$ of the shot sh_i as the key frames to represent the visual content of the video shot. We use the correlation of H-S histogram between key frames to measure the visual similarity just as we do the video shot grouping, defined as follows:

Definition 4.1 *The visual similarity between video shots can be other any functions that can describe the similarity between video shots. Here we define our visual similarity function as:*

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

where $x, y \in \{begin, end\}$*, and* $HistCorr(i_i, i_j)$ *is the H-S histogram correlation between image* i_i *and* i_j*.*

Definition 4.2 *The temporal distance between video shots is defined as the temporal distance between their center frames, in terms of frame number:*

$$d_T(sh_i, sh_j) = |\frac{f_{i_{end}} - f_{i_{begin}}}{2} - \frac{f_{j_{end}} - f_{j_{begin}}}{2}|$$

Then we can combine the visual(spatial) similarity and temporal distance together into a spatial temporal dissimilarity function:

Definition 4.3 *The spatial temporal dissimilarity function between two video shots is defined as:*

$$Dis(sh_i, sh_j) = 1 - VisualSim(sh_i, sh_j) \times e^{-k \times d_T(sh_i, sh_j)},$$

where k *is the parameter controls the slope of the exponential function, in terms of frame number.*

From the definition of our spatial-temporal dissimilarity function we can see that it changes linearly with the video shot visual similarity, but exponentially with the temporal distance between corresponding video shots so that it has counted both the visual similarity and temporal distribution.

Based on the spatial-temporal dissimilarity function we can construct a spatial-temporal relation graph for a set of video shots.

Definition 4.4 *The spatial-temporal relation graph $G(V, E)$ is a graph defined on a video shot set $S_{sh} = \{sh_1, ...sh_n\}$ such that:*

1. *$G(V, E)$ is a complete graph.*
2. *Each vertex v_i in the vertex set V is corresponding to a video shot sh_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 sh_i.*
3. *On each edge e_{ij}, there is a edge weight ew_{ij} which is equal to the spatial-temporal dissimilarity function between video shot sh_i and sh_j. The direction of edge e_{ij} is from the earlier shot to the later video shot. Thus G is acyclic.*

The spatial-temporal relation graph can be viewed as a special case of tournament graph. It can be viewed as a acyclic tournament, and there is a vertex weight on each vertex.

An example of a spatial-temporal relation graph on 5 video shots is shown in Fig. 6.

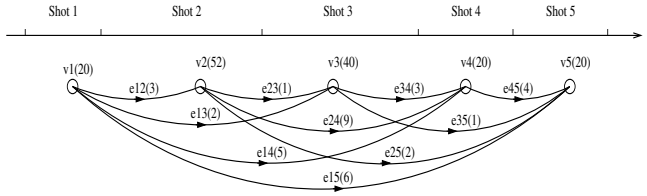

Figure 6: Spatial temporal dissimilarity graph on 5 shots

4.2 Graph optimization

Definition 4.5 *For a path $p_i = \{v_{i_1}...v_{i_n}\}$, the vertex weight summation of p_i is defined as:*

$$VWS(p_i) = \sum_k w_{i_k},$$

$v_{i_k} \in \{v_{i_1}...v_{i_n}\}$ *which is the summation of the lengths of all video shots on the path* p_i.

Since we want the video skimming to cover both the visual contents and the temporal distribution of the video scene, we can search for a longest path in the spatial-temporal relation graph and use the video shots corresponding to its vertexes as the video skimming, under the constraint that the summation of the vertex weights is equal to L_{vs_i}. Note that the length of a path p_i is the summation of the dissimilarity function between consecutive video shot pairs.

Based on the graph definition, we now give the definition of our video skimming problem as:

Problem 4.1 *Given a set of video shots* $S_{sh} = \{sh_1...sh_n\}$ *the spatial-temporal graph* $G(V, E)$ *built on* S_{sh} *and the target video skimming length* L_{vs}*, search the path* $p_s = \{v_{s_1}...v_{s_n}\}$*such that it satisfies the constraint* $VWS(p_s) = L_{vs}$ *and maximizes* $Length(p_s)$.

We can use the video shots corresponding to the vertexes in p_s as the video skimming.

Note that now the problem is a kind of resource allocation problem. The resource we have is L_{vs}, we can allocate them to several vertexes in the graph, and what we gain is the length of the path. At the same time, we want the resources wasted (not allocated) to be minimized.

However, due to that the summation of video shots are not continuous, we may not be able to get the vertex summation exactly equal to the L_{vs} we want. So here we need to consider two factors: First, we want to maximize the length of the path; second, we want the vertex weight summation on that path to be as close as possible to the target L_{vs}. To solve this problem, we change the original problem to the following two problems that deals with both the length of the path and the vertex weight summation:

In the first modification, we employ a tolerance threshold *threshold* then modify the original vertex weight summation constraint to $|VWS(p_s) - L_{vs}| < threshold$, then we search the path p_s that maximizes $Length(p_s)$ under this new constraint. In case that no path whose vertex weight summation satisfies the constraint, we loose the constraint by using a larger *threshold* value then search again.

Problem 4.2 *Given a set of video shots* $S_{sh} = \{sh_1...sh_n\}$*, the spatial-temporal graph* $G(V, E)$ *built on* S_{sh} *and the target video skimming length* L_{vs}*, search the path* $p_s = \{v_{s_1}...v_{s_n}\}$ *such that it satisfies the constraint* $|VWS(p_s) - L_{vs}| < threshold$ *and maximizes the object function* $Length(p_s)$.

This modification finds the longest path in the given interval, however, it has a defect that it cannot guarantee that it will find a solution in the interval (if there is indeed no solution in the interval), and in this case we may have to search again in a bigger interval. To overcome this potential inefficiency, we make the second modification as shown below:

In the second modification, we modify the original objective function into $f_{obj}(p_s, L_{vs}) = Length(p_s) - w \times |VWS(p_s) - L_{vs}|$, by optimizing this function with proper w value we can find a path satisfiable to both the path length and the vertex summation constraint. The defect in the first modification is overcame: this modification will definitely find a solution.

Problem 4.3 *Given a set of video shots $S_{sh} = \{sh_1...sh_n\}$, the spatial-temporal graph $G(V, E)$ built on S_{sh}, the target video skimming length L_{vs} and the parameter w, search the path $p_s = \{v_{s_1}...v_{s_n}\}$ such that it maximizes the object function $f_{obj}(p_s, L_{vs}) = Length(p_s) - w \times (VWS(p_s) - L_{vs})$, and $VWS(p_s) \leq L_{vs}$.*

4.3 Solutions and algorithms

Problem 4.2 and 4.3 are resource allocation problems on graph. The resource we have is the skimming length; we allocate the skim length to vertexes in the spatial-temporal graph and we hope to magnify the object function gained by selecting these vertexes. Brute force searching is feasible but inefficient; however, both the problems have an optimal substructure [20] and can be solved with dynamic programming, shown as follows.

Suppose the number of video shots in the spatial-temporal graph is N_{sh}. A path $p_x^i = \{v_x, ...\}$ is a path begins with vertex v_x, index by i. Let L_{remain} be the left vertex weight summation that we want to allocate to the vertexes on path p_x^i, and let p_x^o be such a optimal path begin with vertex v_x, which means $f_{obj}(p_x^o, L_{remain}) = \max_i f_{obj}(p_{i_x}, L_{remain})$. Then we have the following recursive optimal substructure for the object function problem 4.2:

1. $f_{obj}(p_x^o, L_{remain}) = \max_{t=i_x+1}^{N_{sh}}(f_{obj}(p_t^o, L_{remain} - Length(v_t)) + Dis(v_{i_x}, v_t) \times \tau(L_{remain}, j))$, for $0 < x < n$.

   Here $\tau(L_{remain}, j) = 0$ if $L_{remain} - l_{sh_j} < 0$,
   otherwise $\tau(L_{remain}, j) = 1$

2. $f_{obj}(p_n^o, L_{remain}) = 0$ for all $L_{remain} < threshold$;
   $f_{obj}(p_n^o, L_{remain}) = -penalty$ for $L_{remain} \geq threshold$; This is for ensuring that $|VWS(p_0^o) - L_{vs}| < threshold$.

Algorithm 2 Dynamic programming algorithm for problem 4.2

Input: The candidate shot set $S_{in} = \{sh_1....sh_n\}$, and the shot pairwise dissimilarity function $Dis(sh_i, sh_j)$;
Output: The maximum of the dissimilarity summation function value $LongestLength$ and all optimal sub-solutions $L_{opt}[currentshot][RemainedSize]$.
BEGIN
Set $L_{opt}[i][j] = 0$ for all i,j;
for $RemainedSize = threshold$ to L_{vs} **do**
  $L_{opt}[LastShot][RemainedSize] = -penalty$;
end for
for $ShotId = ShotNum$ to 0 **do**
  for $RemainedSize = 0$ to L_{vs} **do**
    $opt = -infinity$;
    for $NextId = ShotId + 1$ to $ShotNum$ **do**
      if $Length(NextId) < RemainedSize$ **then**
        if $opt < L_{opt}[NextId][RemainedSize - Length(NextId)] + Dis(NextId, ShotId)$ **then**
          $opt = L_{opt}[NextId][RemainedSize - Length(NextId)] + Dis(NextId, ShotId)$;
        end if
      end if
    end for
    $L_{opt}[ShotId][RemainedSize] = opt$;
  end for
end for
$LongestLength = L_{opt}[0][L_{vs}]$;
END

With this optimal substructure we can devise the following dynamic programming algorithm to solve problem 4.2.

The algorithm generates the calculate the length of the optimal path and all optimal sub-solutions. Then we can easily trace back and find the global optimal path, which is corresponding to the skimming shots of the scene. The trace back algorithm is omitted here. In case there are multiple global optimal pathes, the trace back algorithm will also find all of them. We concatenate the skimmings of each video scene then get the whole video skimming.

For problem 4.3, we can see that there is also a optimal substructure lies in this problem, although it is a bit different from the previous one.

1. $f_{obj}(p_n^o, L_{remain}) = l_{sh_n} - L_{vs}$, for all $L_{remain} \leq L_{vs}$;
2. $f_{obj}(p_i^o, L_{remain}) = \max_{j=i+1}^{n}[Dis(sh_i, sh_j) +$

$f_{obj}(p_j^o, L_{remain} - l_{sh_j}) + l_{sh_i}] \times \tau(L_{remain}, j)$.
Here $\tau(L_{remain}, j) = 0$ if $L_{remain} - l_{sh_j} < 0$,
otherwise $\tau(L_{remain}, j) = 1$

The algorithm for 4.3 is similar to the algorithm for problem 4.2. We only need to change the objective function in Algorithm 1. However, this algorithm can ensure that it will find the optimal solution in one turn. The trace back procedure for problem 4.3 is is also similar to that for problem 4.2.

In both algorithms, the skim shots generated by the dynamic programming algorithm might be a little shorter than the target skim length. f after the dynamic programming, we still select some image frames from the original video scene to fill that length.

The time complexity of both dynamic programming algorithm is $O(n^2 \times L_{vs})$, while the space complexity is $O(n \times L_{vs})$. For normal video scenes, n and L_{vs} will not be too large so this algorithms are quite efficient.

5 Experiments

To test the performance of our video summarization method, we implemented the dynamic programming algorithm and applied them to several video clips. We employed a PC platform with 2.0G hz P4 CPU on the Win2000 OS. In our experiments, we choose all the video shots with one or more features in its duration as candidate video shots. The threshold parameter t_1, t_2 we set during scene skim length calculation are set to 3 seconds and 4 seconds respectively. The exponent slope control parameter k is set to 400. We use problem 4.3 to model the problem to select video shots from a scene. The selected video clips are from movies, sitcom videos and cartoons described in Table 2. An example for a scene's key frames (shown as video shot groups) and the summarized scene key frames are shown in Fig. 7.

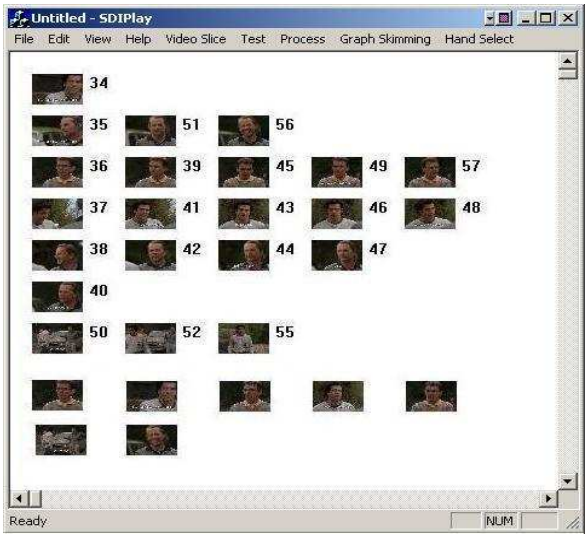

Figure 7: Summarized scene key frames

As our approach is based on video shots, we can use the key frames of the selected video shots to form a static video summary. An example is shown in Fig. 8. From Fig 8 we can see that the content coverage of our video skimming is quite good.

Since currently there is no objective way to evaluate the quality of the generated video skimming, we devise the following subjective way to evaluate our video summarization approach. We invited several people to view the skimming for several test videos then answer several questions about the contents of the videos. Ten people were invited as test users to watch the video skimming generated with two compress rates. Suppose there are N

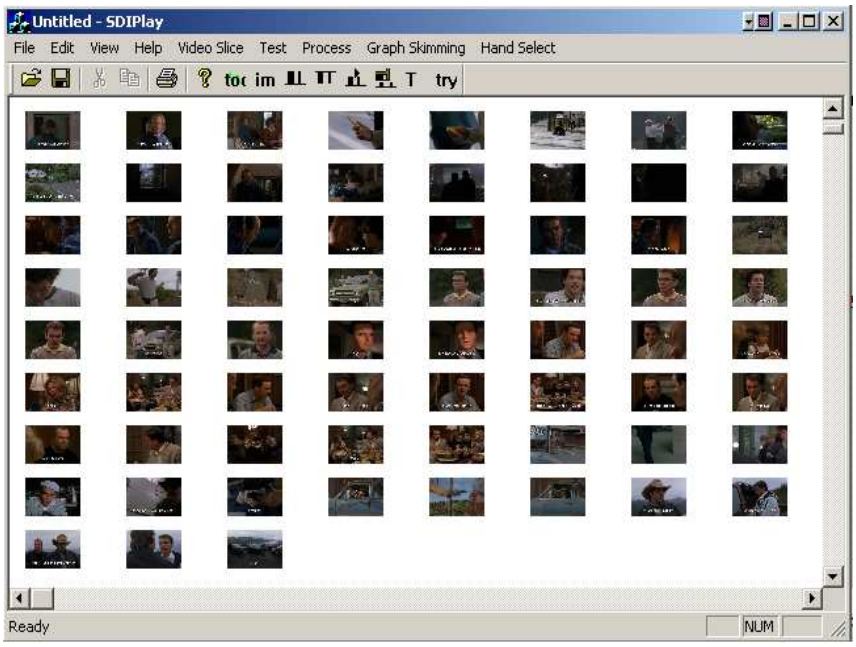

Figure 8: Static video summary

key event scenes in the video, we use the question "What?" to ask the test users to tell how many key events they can perceive by watching the video skimming. Thus we can calculate the score for "What?" question by dividing the event number that the users can find with N. Similarly, question "Who?" deals with the key actors the test users can find in the video skim. All scores are scaled to 10.

Table 2 shows the numerical results for the user test. From the table we conclude that the video skimmings still make good sense to people with

Video Clip	**Length**	**Key events**	**Key actors**	**Rate**	**What?**	**Who?**
Movie1	503 sec.	4	7	0.15	8.05	7.78
				0.30	9.50	9.07
Movie2	1230 sec.	7	8	0.15	7.50	8.33
				0.30	9.82	9.63
Sitcom1	1200 sec.	6	8	0.15	7.23	7.67
				0.30	8.68	8.62
Cartoon1	930 sec.	5	9	0.15	8.09	7.14
				0.30	9.38	8.21

Table 2: User test results

a compression rate at 0.15. The users agree that they can grasp the main line of the story by watching the video skimming. Also, we can see that by selecting coherent video shots, we can guarantee the coherence of the video. The user also agree that the generated video skim doesn't look quite jumpy.

6 Conclusion and future work

Video summarization is an important tool for efficient video browsing and management. In this paper, we describe a new automatic approach to generate moving video skimmings. We analyze the video structure, define the complexity for each video scene then determine each video scene's skim length, then we model each video scene into a spatial-temporal relation graph, and summarize each scene by doing optimization in the spatial-temporal relation graph with dynamic programming. The whole video skimming is obtained by concatenating each scene's sub-skimming. We implement the proposed algorithm and obtain encouraging experimental results.

In the future, we will further investigate the structure of the video scenes to help our skimming generation. Moreover, intra-shot compression will be also studied to shorten the video shots' length in order to further magnify the content coverage.

References

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

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

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

[4] Y. F. Ma, L. Lu, H. J. Zhang, and M. J. Li. A user attention model for video summarization. In *Proceeding ACM Multimedia 2002*, 2002.

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

[6] H. Sundaram, L. Xie, and S. F. Chang. A utility framework for the automatic generation of audio-visual skims. In *Proceeding ACM Multimedia 2002*, 2002.

[7] C. W. Ngo, Y. F. Ma, and H. J. Zhang. Automatic video summarization by graph modelling. *Proceedings of ICCV 03*, 2003.

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

[9] N. Bagaguchi. Generation of personalized abstract of sports video. In *Proceeding of IEEE ICME*, pages 800–803, 2001.

[10] 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.

[11] 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.

[12] 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.

[13] W. Wolf. Key frame selection by motion analysis. In *Proceeding IEEE ICASP96*, 1996.

[14] R. L. Lagendijk, A. Janjalic, M. Ceccarelli, M. Soletic, and E. Persoon. Visual search in a smash system. In *Proceeding of IEEE ICIP*, 1996.

[15] 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.

[16] 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.

[17] Ng Chung Wing, Michael R. Lyu, and Irwin King. Advise: Advaced digital video information segmentation engine and its applications. In *Proceeding of World Wide Web*, 2002.

[18] S. Uchihashi and J. Foote. Summarizing video using a shot importance measure and a frame-packing algorithm. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3041–3044, 1999.

[19] M. Yeung, B. L. Yeo, and B. Liu. Extracting story units from long programs for video browsing and navigation. In *IEEE Proceedings of Multimedia*, pages 296–305, 1996.

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