A Novel Video Summarization Framework for Document Preparation and Archival Applications

Shi Lu, Irwin King, Michael R.Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR
{slu, king, lyu}@cse.cuhk.edu.hk

Abstract— With the rapid growth of network bandwidth and high-capacity storage devices, videos have become an important way of communication in the aerospace industry and many other entities. However, browsing and managing huge video databases are quite tedious. To solve the problem, in this paper, we propose a novel video summarization framework, and discuss its potential usage in the document preparation and archival applications. The proposed framework generates video skimmings that guarantee both the balanced content coverage and the visual coherence. First, we segment the raw video into video shots, analyze the structure of the video, find the boundaries of semantic scenes, then calculate each scene's skimming length by its structure and content entropy. Second, we define a spatial-temporal dissimilarity function between video shots, model each video scene as a graph, and find each scene's optimal skimming shots in the graph with dynamic programming. Shot arrangement patterns are analyzed to improve the coherence of the video skimming. Finally, the whole video's skimming is obtained by concatenating the skimmings of the scenes. Our proposed framework generates video skimmings that guarantee both the balanced content coverage and the visual coherence. Experiments are conducted to evaluate the effectiveness of our proposed approach.

Keywords— Video summarization, graph optimization, multimedia modeling, video archival applications

TABLE OF CONTENTS

1. INTRODUCTION

Video is increasingly becoming the favorite medium in the aerospace industry for training and education [1] for its extraordinary expressive power. In surveillance missions [2] and other research projects [3], a large amount of video data will be recorded and archived. The massive growing video data thus give rise to a challenge for efficient video browsing and management since it is time consuming to download and browse through the whole video contents. To solve this problem, video summarization, which engages in providing concise and informative video summaries to help people in browsing and managing 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, [4] tends to adapt to the dynamic video content. A mosaic-based approach is suggested in [5]. Since edited videos have their intrinsic structures, later work presents video contents according to the detected video structure. In [6], the authors analyzed the video structure after video segmentation, and then get a tree-structured Video-Table-Of-Contents(V-TOC). In [7], a scene transition graph is constructed as the video content presentation.

Compared with static video summary, dynamic video skimming is more attractive for it reserves the dynamic property of the video, which makes more sense to the user. Much effort is further devoted to dynamic video skimming generation. In the VAbstract system [8], key movie segments are selected to form a movie trailer. The Informedia system [9] 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 [10] the authors construct a user attention curve to simulate the user's attention toward different video contents. [11] proposes a utility function for each video shot, and video skimmings are generated by utility maximization. [12] assigns different weight scores on several important features of the video then selects the video skimming that maximizes the feature score summation. [13] 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 [14],

0-7803-8870-4/05/$20.00/
IEEEAC paper # 1415

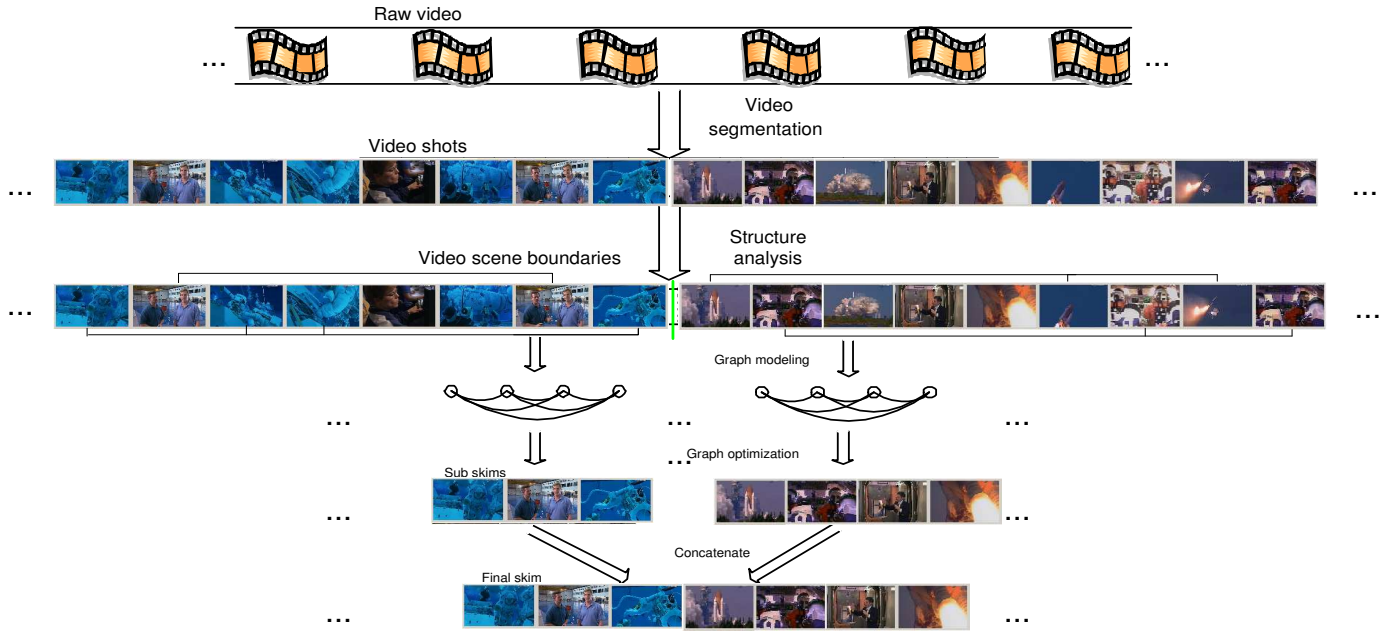

Figure 1. Workflow of the framework

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. To attack this problem, semantic information is needed to make a meaningful video skim. Unfortunately, although quite a lot of attempts have been made to automatically annotate generic video and image contents [15], [16] and event detection in specific video categories like sports video [17], 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 [18], [19].

A video summary should be able to cover the major video contents with balance. Although many video skimming generation techniques have been proposed, few of them have stressed on preserving the structure of the video. In this paper, we describe a novel video summarization approach that combines video structure analysis and graph optimization. We analyze the structure of the original video, find the scene boundaries, and determine each video scene's target skim length. We then model each scene into a graph, create video skimming for each video scene with graph optimization, and concatenate each local video skimming to get the final video skimming. The workflow of our approach is shown in Figure 1. In comparison with the previous approaches, our approach preserves balanced structural coverage of the major video contents, and ensures the coherence of the video skimming.

The paper is organized as follows: In Section 2 we describe the video preprocessing step. In Section 3 we describe our video summarization method. In Section 4 we show experimental results and make some discussions. Finally, in Section 5 we make conclusion and discuss our future work.

2. VIDEO PREPROCESSING

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.

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 [20], [21], [22].

To measure the similarity of two images, traditional methods use the frame difference, correlation/intersection of color histogram, etc. Frame difference is easy and quick to compute, however, it is very sensitive to camera motion. To overcome this, a proper offset can be calculated to compensate the motion, which is 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 is also sensitive to camera motion. Consequently, we use a simple yet robust video shot detection method, described as follows.

To detect video shot boundaries, we extract a *video slice image* [23] from the original video then detect cuts by analyzing 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 is shown in Figure 2.

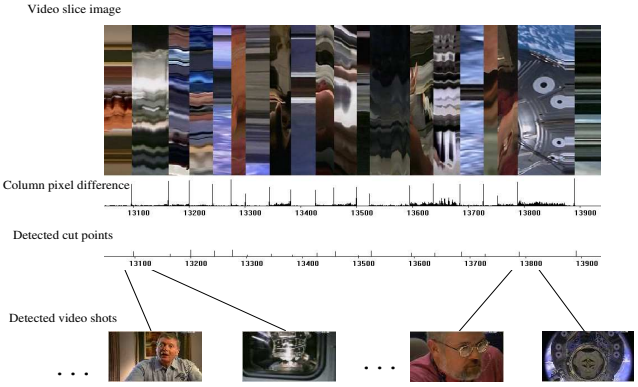

Figure 2. Video slice image extracted from a video, accompanied by the pixel difference and the detected shot breaks

We can choose any 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 pixel rows in the slice image. We use the pixel difference as the measure for image similarity.

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 as 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 Figure 2.

From the difference 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.

We notice that there are two factors that 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. Based on these two criteria, we apply the following non-linear filter to the pixel difference function to find out such video shot changes:

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

where w is the half width of the window.

The original pixel difference function and the filtered difference function are shown in Figure. 2.

After applying this filtering, the transformed value will be more than 1 only at those points that are local maxima. 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.

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.

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 several semantic-related 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 [6]. Figure 3 shows the hierarchical structure of a video.

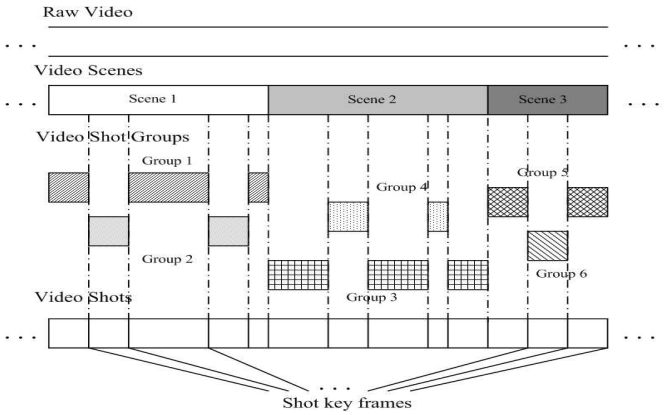

Figure 3. Hierarchical video structure

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, 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 [6], hierarchical clustering [7], and spectral graph partitioning [24]. 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 employ the algorithm to group similar video shots.

Given a series of video shots, we can 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. In this paper, we employ an 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 similarity measure w_{ij}, shown as follows:

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

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

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:

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

where

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

is the cut value and

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

is the association of A with the vertex set V.

Given G, the optimal partition for G is the partition that maximize the Normalized Cut Value $NCut(A, B)$. The $NCut$ minimization problem can be transformed into solving a standard eigen system:

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

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. 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 detected video scenes can be classified into two types: *loop scenes* and *progressive scenes*, as the examples shown in Figure 4. 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 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.

Shot arrangement pattern analysis

For each loop scene, we can assign a unique label for each detected video shot group. A video shot group composed with more than one video shot is called a key video shot group, and all the remaining 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. Each video shot group

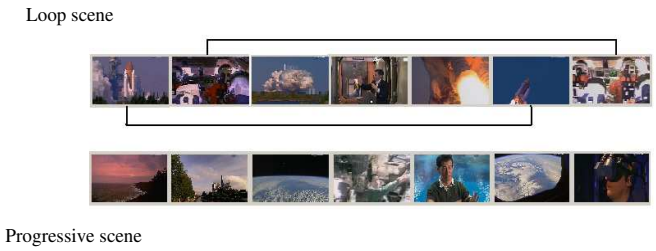

Figure 4. Example of loop and progressive scenes

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 $lb_1....lb_n$, 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 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. We can easily find all k-nrs strings by scanning the video label string. Then we use them as skimming candidates. Some sample nrs strings are shown in Figure 5.

Figure 5. 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 good candidates for video skimming since they are composed of video shots depicting non repetitive contents with the least redundancy, and 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:

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

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

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

where $sh_{i_x} \in str_i$.

3. VIDEO SUMMARIZATION PROCEDURE

Basically there are two kinds of video skimming: *overview* and *highlight*. For specific domain like sports and news, the user already knows some domain-specific knowledge and she may just request those video shots that she is interested in like "Give me three minutes of video about goals and corner kicks." This kind of video skimming is called "highlight". But for movies, the user is unaware of the contents thus can only specify a target length for seeing enough details about the movie. The request may be like "Give me three 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 video overview generation.

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

1. **Conciseness**–For conciseness, 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 contents 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 should increase the coherence of the video skimming while preserving the content coverage.

Video skim length distribution

To ensure the informativeness of the generated video skimming, we need to preserve the important contents of the original video. Given a series of detected video scenes, obviously, longer and more complex video scenes should be more important. 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, we define the content entropy of a scene Sc_i as:

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

The content entropy of a loop scene can be used as a measure for the complexity of a loop video scene.

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 scene and each group in the video as follows:

1. For each progressive scene Sc_x,

$$Sl_x = l_{Sc_x} \times r_s.$$

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 round, the left skim length is L'_{vs}, for the loop scenes $\{Sc_1...Sc_n\}$,

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

In a similar manner, we discard Sc_i if Sl_i is less than a preset threshold t_2.
3. For the remaining loop scenes $\{Sc'_1...Sc'_m\}$, we set

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

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.

Video skim generation by graph modelling and optimization

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 shot strings it contains, then we select the skimming video shots by performing optimization on that graph.

To model the scene as a graph, we first specify an integer l_{str}, then we decompose the video scene into a set of non-overlapped nrs strings $Nrs_{l_{str}}$, whose length is at most l_{str}. We can use l_{str} to control the coherence of the extracted video skim. For example, the Nrs_3 set for a video scene $\{1245141316\}$ can be $\{124, 514, 13, 16\}$. As a special case, the Nrs_1 set of a scene is just all the video shots it contains.

Based on the shot 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. 6.

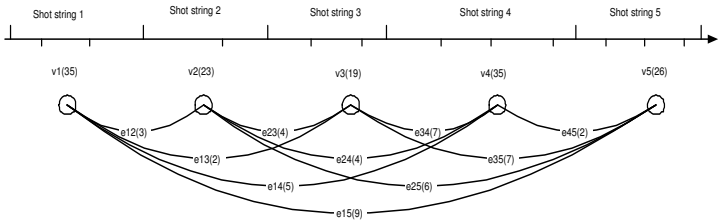

Figure 6. Spatial temporal dissimilarity graph on five shot strings

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

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

and

$$w_{e_{ij}} = Dis(str_i, str_j).$$

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 section. $d_T(str_i, str_j)$ 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. 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.

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

Problem 3.1: 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

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

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

Solution and algorithm

Problem 3.1 is a constrained optimization problem. Brute force searching is feasible but inefficient; however, the problem has an *optimal substructure* [25] 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 $p^o_{v_0,L_{vs}}$ 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 $f_{opt}(p^o_{v_0,L_{vs}})$ and all optimal sub-solutions with the following dynamic programming algorithm:

Algorithm 1 Video skim selection algorithm

```
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
```

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 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.

4. EXPERIMENTS AND DISCUSSIONS

We implement the video summarization algorithms then apply them to some video clips. We employe a PC platform with 2.0G hz P4 CPU on the Win2000 OS. The exponent control parameter k in the spatial-temporal dissimilarity function is set to 250, 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 three documentary videos and two movie clips, 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. 7.

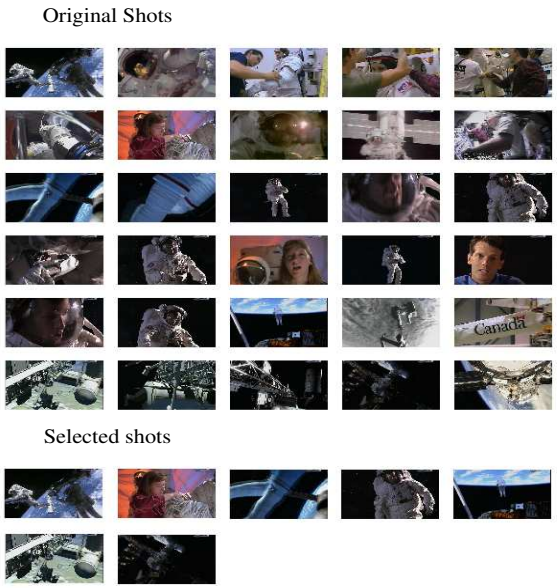

Figure 7. Summarized scene key frames

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.

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.

5. Conclusion and Future Work

Video summarization is an important tool for document preparation and archival applications in large video databases. In this paper, we first analyze the video structure, define the content entropy to measure the video scenes' complexity, and determine each video scene's skim length. In order to ensure a balanced content coverage of the selected video skimming, we model each video scene as a spatial-temporal relation graph, and propose to summarize each scene by performing optimization in the spatial-temporal relation graph with dynamic programming. We also analyze the shot arrangement patterns in each video scene to enhance the coherence of the video skimming. 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 employ high level semantic information of the video to make better video summaries. Moreover, intra-shot compression will be studied to shorten the video shots' length so that the content coverage of the video skimming can be further magnified.

6. Acknowledgements

The work described in this paper was fully supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 4351/02E and Project No. CUHK4182/03E).

References

[1] P. Trudeau and W. Specht. Meet ever increasing training delivery needs. In *Proceedings of 2003 IEEE Aerospace Conference*, pages 3885–3896, March 2003.

[2] D. A. Whelan, A. Filip, J. J. Koss, T. Kurien, and G. Pappas. Global space-based ground surveillance: mission utility and performance of discoverer ii. In *Proceedings of 2000 IEEE Aerospace Conference*, volume 5, pages 1–11, March 2000.

[3] C. Y. Chong, D. Garren, and T. P. Grayson. Ground target tracking-a historical perspective. In *Proceedings of 2000 IEEE Aerospace Conference*, volume 3, pages 433–448, March 2000.

[4] 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.

[5] 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.

[6] Y. Rui, T. S. Huang, and S. Mehrotra. Constructing

Video Clip	Duration	Major events	Skim Rate	Mfn.	Fav.
Document 1	2403 sec.	7	0.15	82.8/**85.7**	4/**6**
			0.30	94.3/**92.9**	5/**5**
Document 2	3230 sec.	8	0.15	78.8/**76.4**	3/**7**
			0.30	88.9/**92.9**	6/**4**
Document 3	1477 sec.	5	0.15	88.0/**86.0**	3/**7**
			0.30	96.0/**96.0**	2/**8**
Movie 1	1183 sec.	9	0.15	82.2/**85.6**	4/**6**
			0.30	94.4/**97.8**	5/**5**
Movie 2	602 sec.	4	0.15	77.5/**75.0**	4/**6**
			0.30	92.5/**97.5**	3/**7**

Table 1. User test results. The scores with l_{str} is equal to 3 are in **bold**

table-of-content for videos. *ACM Multimedia Systems Journal, Special Issue Multimedia Systems on Video Libraries*, 7(5):359–368, Sept 1999.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

Shi Lu *received his B.S. degree in computer science and technology from Tsinghua University, Beijing, China in 2002. He is currently a M.Phil. student in the department of computer science and engineering, the Chinese University of Hong Kong. His research interest is in multimedia processing.*

Dr. Irwin King *Irwin King received the B.S. degree in Engineering and Applied Science from California Institute of Technology, Pasadena, in 1984. He received his MSc and PhD degree in Computer Science from the University of Southern California, Los Angeles, in 1988 and 1993 respectively. While working on his degree, he worked as a Scientific Software Engineer at the Xerox Special Information System (XSIS) group in Pasadena. At Xerox, he helped to design and implement the Analyst System, which is an all-in-one software package with seamless integration of word processing, spreadsheet computation and charting, painting, drawing, and database management. Aside from working at Xerox, he also taught various undergraduate courses in Computer Science at the Occidental College, Los Angeles.*

He joined the Chinese University of Hong Kong in 1993. His research interests include content-based retrieval methods for multimedia databases, distributed multimedia information retrieval in peer-to-peer systems, and statistical learning theory.

He is a member of ACM, IEEE Computer Society, International Neural Network Society (INNS), and Asian Pacific Neural Network Assembly (APNNA). Currently, he is serving the Neural Network Technical Committee (NNTC) under the IEEE Computational Intelligence Society (formerly the IEEE Neural Network Society). He is also a governing board member of the Asian Pacific Neural Network Assembly (APNNA). He is a founding member of the Neural Computing and Engineering Laboratory (NCEL) and the Multimedia Information Processing Laboratory (MIP Lab). He is a member of the Editorial Board of the Neural Information Processing–Letters and Reviews Journal (NIP-LR). He has served as program and/or organizing member in international conferences and workshops, e.g., WWW, ICASSP, IJCANN, ICONIP, etc. He has also served as reviewer for international conferences as well as journals, e.g., Information Fusion, IEEE TCAS, SIGMOD, IEEE Transactions on Neural Networks, IEEE Pattern Analysis and Machine Intelligence, IEEE Transactions on Multimedia, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on System, Man, and Cybernetics, Machine Vision and Applications, International Journal of Computer Vision, Real-Time Imaging, SPIE Journal of Electronic Imaging, International Journal of Pattern Recognition and Artificial Intelligence, etc.

Dr. Michael R. Lyu *received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, China, in 1981, the M.S. degree in computer engineering from University of California, Santa Barbara, in 1985, and the Ph.D. degree in computer science from University of California, Los Angeles, in 1988.*

He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. He was with the Jet Propulsion Laboratory as a Technical Staff Member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, as an Assistant Professor. From 1992 to 1995, he was a Member of the Technical Staff in the applied research area of Bell Communications Research (Bellcore), Morristown, New Jersey. From 1995 to 1997, he was a Research Member of the Technical Staff at Bell Laboratories, Murray Hill, New Jersey. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, wireless communication networks, Web technologies, digital libraries, and E-commerce systems. He has published over 150 refereed journal and conference papers in these areas. He received Best Paper Awards in ISSRE98 and ISSRE2001. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. He was the editor of two book volumes: Software Fault Tolerance (New York: Wiley, 1995) and The Handbook of Software Reliability Engineering (Piscataway, NJ: IEEE and New York: McGraw-Hill, 1996).

Dr. Lyu initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the program chair for ISSRE96, and has served in program committees for many conferences, including ISSRE, SRDS, HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WWW. He was the General Chair for ISSRE2001, and the WWW10 Program Co-Chair. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia. He has been an Associate Editor of IEEE Transactions on Reliability, IEEE Transactions on Knowledge and Data Engineering, and Journal of Information Science and Engineering. Dr. Lyu is a fellow of IEEE.