Real-Time Imaging **6**, 449–459 (2000)
doi:10.1006/rtim.1999.0197, available online at http://www.idealibrary.com on IDEAL®

Video Summarization Using R-Sequences

In this paper, we propose a new method of temporal summarization of digital video. First, we address the problem of extracting a fixed number of representative frames to summarize a given digital video. To solve it, we have devised an algorithm called content-based adaptive clustering (*CBAC*). In our algorithm, shot boundary detection is not needed. Video frames are treated as points in the multi-dimensional feature space corresponding to a low-level feature such as color, motion, shape and texture. The changes of their distances are compared globally for extraction of representative frames. Second, we address how to use the representative frames to comprise representative sequences (*R-Sequence*) which can be used for temporal summarization of video. A video player based on our devised algorithm is developed which has functions of content-based browsing and content-based video summary. Experiments are also shown in the paper.

[1]Xinding Sun and [2]Mohan S. Kankanhalli

[1]*Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106*
E-mail: xdsun@iplab.ece.ucsb.edu
[2]*School of Computing, National University of Singapore, Kent Ridge, Singapore 119260*
E-mail: mohan@comp.nus.edu.sg

Introduction

Among the information media delivered through the Internet, digital video is playing an increasingly important role. In recent years, the development of compression and network technology has enabled the creation of a large amount of digital video content. Owing to the rapid increase in the size of digital video databases, users are being provided with a very broad selection of video content, and thus they require more flexible as well as more powerful video handling tools. Therefore, development of advanced video data management tools is a very important area of research.

A digital video is usually very long temporally, requiring large storage capacity. Therefore, given a long video, a user would like to obtain a pre-determined fixed number of important frames to describe the video. This would help the user in getting an approximate idea of the video content. This number can be 5 or 10 percent of the total number of frames in the original video. Such a request is reasonable since a video normally contains much redundant information. The objective of this work is actually to temporally compress a digital video for a given length criterion.

While much work has been done on video indexing and browsing, this work has not been addressed by researchers. Here, we term the important frames as *representative frames* [1]. Some researchers call them key frames as well [2,3]. The representative frames therefore should reflect the content changes of a video. Even now, automatic understanding of image and video content in terms of semantics is not possible. So, video processing is still based on low-level features like color, motion,

Videos and software in this paper can be found at http://maya.ece.ucsb.edu/~xdsun/html/rd.html shape, texture etc. [4]. As a result, in this paper we also base our analysis on these low-level features. However, unlike shot detection, our work is not based on the assumption that all kinds of low level features like color, motion, shape, texture, etc. change somewhat abruptly at shot boundaries. For extracting representative frames, our result depends on the selected low-level feature of interest. This is because each different low-level feature changes in a different manner, even in the same video. For example, in a soccer game, the most dynamically changing low-level feature is motion and not color. The above requirements can be formally described as:

1. an ordered set of input digital video sequence V with cardinality $N.V = \{F_1, F_2, \ldots, F_N\}$, where F_1, $F_2, \ldots, F_N$ are the frames of V.
2. ratio α such that $0 < \alpha < 1$.
3. low-level content P of {color, motion,...}.

To extract: a set of output frames V' with cardinality of N;

$$V' = \{R_{p1}, R_{p2}, \ldots, R_{pN'}\} \tag{1}$$

where

- $N' = N * \alpha$.
- $R_{p1}, R_{p2}, \ldots, R_{pN'} \in V$, are the representative frames of V with respect to feature P.
- $V' \subseteq V$.

So basically, given a digital video V having N frames, we would like to extract an $N * \alpha$ cardinality subset of frames which best represent the content P of the video V.

Related work

An example of early work on video content analysis was carried out by Connor [5]. Specific object changes are detected for key frames. Mills *et al.* [6] applied the temporal subsampling method in their "magnifier tool". Finkelstein *et al.* [7] utilized a multiresolution method for video browsing, which can both spatially and temporally subsample a video sequence.

After the development of shot detection techniques, researchers would select one frame (usually the first frame) from a shot to represent the entire shot. One example of such work is the content-browsing system by Arman *et al.* [8]. Boreczky *et al.* [9] compared shot boundary detection techniques such as pixel differences, statistical differences, compression differences, edge track tracking etc. Using shots as the basic unit is not enough for detailed video analysis, so other researchers have focused their work on finding representative frames.

In Zhang *et al.* [10], color and motion features are utilized for key frame extraction based on shot detection. For the color feature based criterion, the first frame is used both as a reference frame and as a key frame in the beginning. The distances to this frame for the subsequent frames are computed. When the distance exceeds a given threshold, a new representative frame is claimed and the claimed frame serves as the new reference frame for the following frames. For the motion feature based criterion, mainly two types of motions—pans and zooms are detected. The first and the last frame of a zooming process are taken as key frames. For the panning process, key frames are extracted depending on the scale of panning. The aim of this work is not to solve the problem we put forward, but it is a large step beyond the usual one key frame per shot methods of video description.

Smith *et al.* [2] have proposed a method of selecting key frames based on audio and image. The audio track is created based on keywords. The image frames are selected using a ranking system which regards faces or text as most important, static frames following camera motion the second important, etc. Though the integration of speech, language and image information is the best way to understand a video, the generation of key frames based on such a technique still requires manual intervention and there is a lot of scope for further improvement.

Similar to our earlier work, Dementhon *et al.* [11] proposed a method of temporal summarization of video by using temporal curves to fit the video content.

Until now, most of the previous work on video parsing was based on the shot detection technique. Detection of shot comes naturally from the video production process but it is difficult to accurately detect all shot boundaries in a video. While Hampapur *et al.* [12] proposed using post production editing frames to detect shot boundaries, advanced video editing techniques can blur the shot boundaries as well. The morphing technique, for example, can process a video in a way that even humans cannot find the explicit shot boundaries in it. One example of such a video is employed in our experiment in the next section. Therefore, the question arises — can we effectively extract representative frames without detecting the shot boundaries?

To meet the above considerations, in this paper we propose the CBAC technique that can extract a fixed number of representative frames from a video without shot detection. Different fractions of a video possessing different number of frames can thus be computed based on any desired low-level feature. When required, they can be retrieved and utilized for diverse applications.

Extraction of representative frames

Suppose a user is asked to select some important frames of a video based to a typical low-level content like color, motion, shape or texture. He will probably make the choice based on the amount of change of the low-level content. In the places where there are more changes he will choose more frames and will choose a lesser number of frames where there is less content change. In fact, the work of extracting representative frames is based on such an assumption.

The spatial feature of video frames

If we use an M-dimensional feature to analyse a video, each frame of the video maps to a point in the Euclidean space E_M. The visual similarity between any two frames is well reflected by the distance between the two corresponding points in the space. Thus, the entire sequence of video frames maps to a set of points in the M-dimensional space. The temporal succession of the frames in the video traces a corresponding trajectory of points in the feature space. Thus, the video begins from a point which corresponds to the first frame and the temporal sequence of frames subsequently traces a path through the points in the space. When the content of frames changes quickly, the path also moves quickly with a larger step size between the points. When the feature changes slowly, the points are also more closely spaced on the path. The feature could be color, motion, shape or texture. In this paper, we base our work on color. As histogram is a good tool [9] for video processing, we describe our algorithm using histogram. However, it must be noted that our technique is more general and can be used with the other low-level content with appropriate feature measures.

In QBIC [13], given two M bins color histograms of two images X and O, the distance metric for image retrieval is as follow:

$$D^2(X,O) = (X - O)^T A(X - O)$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{M} a_{ij}(X_i - O_i)(X_j - O_j) \tag{2}$$

where $A(a_{ij})$ is an $M{*}M$ matrix, and a_{ij} represents the proximity of the bin i and j; X_i and O_i denotes the histogram value of X and O, respectively. When A is an identity matrix, (2) yields the Euclidean distance.

In our implementation, to save computation time without degrading performance [4], in our program we use the gray-level histogram city-block distance as the metric:

$$D_c(X,O) = \sum_{i=1}^{M-1} |X_i - O_i| \tag{3}$$

If we use the first frame of a video as the reference frame and compute the distances of this frame to all the subsequent frames, we can map all the frames to a one-dimensional discrete signal. This method is similar to that used by Zhang *et al.* [10], except that only the first frame of the whole video, rather than the first frame of each shot is selected as the reference frame. Figure 8(a) shows the result of such a mapping for our test video "news" using histogram as the feature. Distances of all the frames to the first frame are computed, which reflects the content changes to a certain degree. However, this measurement does not always work.

Suppose O is the histogram of the first frame of a video and we use it as the reference frame, X is the histogram of a randomly selected frame from the same video. Given a constant ω, if the video moves on a hyper-cube:

$$\sum_{i=1}^{M} |X_i - O_i| - \omega = 0 \tag{4}$$

then, such distance measurement can not detect any change.

For the sake of simplicity, in Figure 1 we assume $M{=}2$ and O, A, B, C are frames of a video. So all the points are located on the same plane. If we use O as a reference point, because $OB \perp AC$, on the line ABC we cannot detect any distance change. Actually, we can see

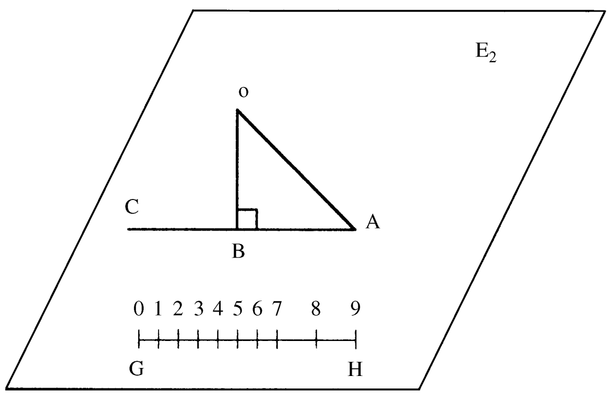

Figure 1. Examples of video trajectories in the feature-space. M=2, E_M E_2.

that the distance between A and C is even larger than that of OA.

Figure 2 shows an example video with such a problem. A test video "skate" starts with a black picture whose histogram is $O = (S,0,\ldots,0)$, where S is the number of pixels of each frame. If we use the M-bin histogram for analysis, then according to (3) we can obtain:

$$D_c(X,O) = \sum_{i=0}^{M-1} |X_i - O_i| = 2S - 2X_0 \qquad (5)$$

This will reflect changes only of the first bin of each frame. In the beginning, for more than 100 frames, X_0 is the very similar, so the above distance will not detect change although the real content (and the value of other bins) changes a lot.

Similarly, we can also apply this analysis to the Euclidean distance measure (the problem occurs when the points corresponding to frames move on a hypersphere in the feature space) and equation (2). Although the severity of the above problem may vary with the metric selected, since a video is generally very long, the overall possibility of the problem occurring is still significant.

In reference [10], the first frame of each shot was used as the reference frame in the beginning and its distances to the subsequent frames are computed. When the distance exceeds a given threshold, a representative frame is claimed and the claimed frame serves as a new reference frame. By changing the reference frame intermittently, the errors due to the above problem may be limited. However, the above problem still persists. In the case we require a very small fraction of frames to be extracted, it will require a big threshold. So it is possible that this may introduce large errors.

Also, if we want to extract a fixed pre-determined number of frames from a video, then it will be very difficult to control the output number by controlling the threshold. Moreover, our aim is to extract representative frames without shot detection. Therefore, we would like to pursue other solutions which can avoid these difficulties.

Using clustering method for extraction of representative frames

As described earlier, the temporal sequence of frames of a video maps to a trajectory of points in the feature space. The nature of the spatial distribution of the points corresponding to a video can be described as clusters connected by abrupt or gradual changes. During most of the time, the trajectory will move around in a small cluster. It is impossible for all the frames in a video to be spatially far apart and thus have unrelated content, because the frames of a video work together to convey meaningful information. This nature of the distribution of points provides a sound basis for our clustering technique.

Clustering Criterion

If we regard a video V as the set of N points in the multi-dimensional feature space, then the representative frames in fact divide V into N' clusters. We call such clusters *units* in the paper. The units are temporally contiguous and they are delineated by the representative frames R_{p1} to $R_{pN'}$. According to (1), we can also use the units to describe V as:

$$V = U_1 \circ U_2 \circ \cdots \circ U_{N'} \qquad (6)$$

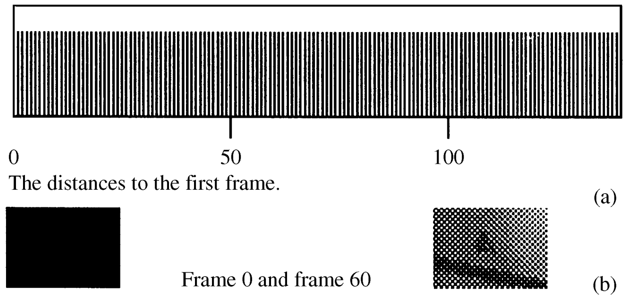

Figure 2. Part of experiment result on "skate".

where

- $U_1 = \{R_{pi}, R_{pi1}, \ldots, R_{p(i+1)}\}$, $i \in [1, N']$
- $\bigcirc$ is the temporal concatenation operation.

Since we are analyzing the feature-points trajectory of a video in temporally localized regions, it is possible to use the change between consecutive representative frames to represent the change within a unit. Given a unit U_i and a selected feature P, we define the *unit change* as following:

$$Change(U_i) = D_c(R_{pi}, R_{p(i+1)}) \tag{7}$$

From the optimization point of view, given a selected feature, the objective of representative extraction is to divide a video into units that have very similar unit changes. Our algorithm should thus try to make the unit changes as similar as possible. This can be described as to minimize:

$$\sum_{i=1}^{N'-1} \sum_{j=i+1}^{N'-1} |Change(U_i) - Change(U_j)| \tag{8}$$

Therefore, we base our algorithm on this criterion. We begin the algorithm with finely divided units which are classified into two clusters after each iteration. Then, the units are modified by deleting redundant frames from the units of one of the clusters so as to make the unit changes similar. This clustering and deletion process is carried out several times. The algorithm thus iteratively converges to the desired number of units in an adaptive way based on the content change within units.

The Adaptive Extraction Process
The whole process of our clustering algorithm is shown in Figure 3. Here, we provide the description.

For a given video V with length N, suppose we want to extract N' representative frames. The feature (in this paper we use histogram) of each frame in V is computed first. This algorithm works in an iterative fashion. We start initially with all the frames of the video and iteratively drop frames till the desired result is obtained.

The sequence of the video frames is partitioned into small units whose length are all L. All the units are temporally consecutive. Figure 4 shows the partitioning with L=2 and L=3 respectively. The units for L=3 are {(0,1,2), (2,3,4), (4,5,6), (6,7,8)}. In each unit, the unit change is computed, which is the distance between the first frame and the last frame of the unit.

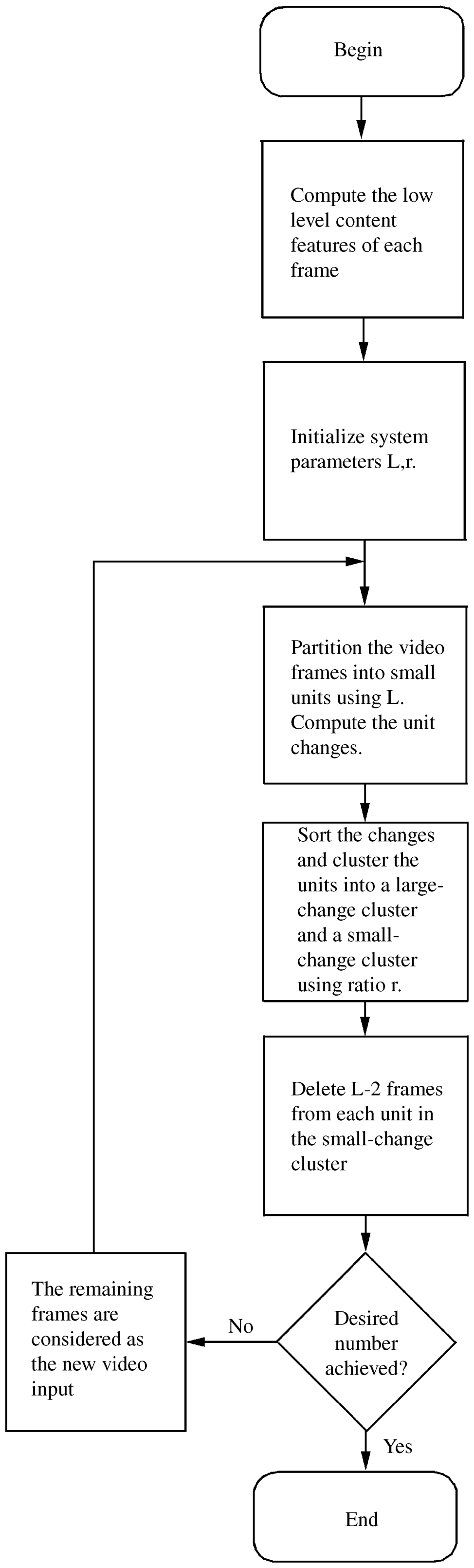
Begin
Compute the low level content features of each frame
Initialize system parameters L,r.
Partition the video frames into small units using L. Compute the unit changes.
Sort the changes and cluster the units into a large-change cluster and a small-change cluster using ratio r.
Delete L-2 frames from each unit in the small-change cluster
Desired number achieved?
No
The remaining frames are considered as the new video input
Yes
End

Figure 3. Adaptive clustering algorithm.

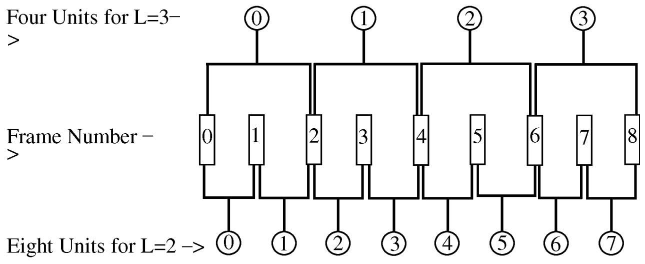

Figure 4. Sequence partitions.

The computed changes stand for each unit and they construct an array of length $K = \lceil N/(L-1) \rceil$. Because our objective is to extract representative frames according to frame content changes, the changes do reflect the actual degree of content change in all the units. This is because the distance metric is computed in a temporally localized region. By sorting the unit changes in an ascending manner, we get an array which represents the general content change of the video. The elements which are located in the beginning part of the array represent the frames where there are small changes, while the units in the later part consists of frames having large changes.

By selecting a ratio $0<r<1$ we cluster the array into two clusters according to the value of unit change. The first cluster comprises of the smallest elements of the array and its length is $K*r$, here we call it the *small-change cluster*. The rest of the elements comprise the *large-change cluster*.

If the change of a unit belongs to the currently large-change cluster, then we take all of its frames as part of the current extracted representative frames. If the change of a unit belongs to the small-change cluster, then we will delete all the frames except the first and the last frames from the unit. The first and the last frames are retained as part of the current extracted representative frames. After the deletion process, $K*r*(L-2)$ frames will be deleted.

Suppose the number of frames left is N''. If $N' \geq N''$, then our desired result is obtained and we can stop the algorithm. If it is not true, we can dynamically regroup all the retained frames as a new video and repeat the last procedure.

With the decrease in the number of frames for comparison, small units are consequently clustered together. A unit will physically span across more frames in the original video. So it will adaptively represent a larger range of frame changes in the original video. The smaller the number we desire, the more times the algorithm would adaptively repeat the procedure. After each iterative process, there will be frames deleted from the sequence, so the overall number of frames left will decrease each time. Therefore, no matter how small a number may be required, the algorithm will adaptively and iteratively converge to the desired requirement.

Throughout the algorithm, shot boundaries do not need to be detected. The algorithm will automatically converge. So, an adaptive way of extracting is achieved.

Selection of Parameters

As the whole extraction process is basically unsupervised, the result will depend on the proper selection of the parameters L and r.

1) Selection of L

If L=2, the distance is in the fact consecutive frame difference. Consecutive frame difference has been successfully applied for shot detection, but it is not suitable for finding representative frames. As illustrated in Figure 1, assume on line GH are ten frames of a video. They are labeled 0 to 9. Their positions are shown as ticks on the line in the figure. Each step in 7–9 is larger than those in 0–7. Suppose we want to select two frames from G–H and use consecutive frame difference as measurement, we would delete all the frames in 0–7. However, the overall distance from 0–7 is actually even larger than that of 7–9, so we should extract at least one frame in 0–7. The failure of consecutive frame difference arises from the fact that it loses the cumulative change information in a video.

In general, if we use a large L, the algorithm will converge very fast and it will save a lot of computation time. In the beginning of the algorithm, a large L will not degrade results. However, if the required number is very small, the algorithm will iterate many times. With the iterations of the algorithm, the unit will in the end may physically span across many frames. Consequently, the smaller the L the better the result quality will be. Table 1 shows the frame numbers extracted from our test video "news". Its content is listed in Table 2. Required representative number is $N' = 330 \times 0.05 \approx 17$. When L=3, the main information is indeed extracted by the algorithm but when L=5, frames of section 4 are all missed. In practice, if a video is very short, then we use

Table 1. The Extracted Frames of "news", $N'=17$. N_r is the actually extracted number. The positions of the frames for L=3 and L=5 are displayed in Figure 8(e) and 8(f) respectively

L	r	N_r	Representative Frames
3	0.3	16	0 67 68 78 109 110 114 118 169 170 258 259 265 300 306 307
5	0.3	17	0 67 68 69 70 113 114 115 116 168 265 299 300 301 305 306 307

$L=3$ in the algorithm. If the video is very long, then we use a variable L. In the beginning of algorithm, we let $L=5$ and when the extracted number drops to no more than 20% of original video, we then change L to 3.

2) Selection of r

If $L=3$ or 5, then 1 or 3 frames in each unit of the small-distance cluster will be deleted after the execution of one loop of the iterative algorithm. Accordingly, if before the iteration the retained number is N'', then after the iteration, around:

$$N''/2 * r * 1 = N'' * r * (1/2) \quad \text{for } L = 3,$$

$$N''/4 * r * 3 = N'' * r * (3/4) \quad \text{for } L = 5 \qquad (9)$$

number of the frames will be deleted.

In many cases, it is really not critical that the number of extracted representative frames is strictly equal to the required number. Assume that the maximum allowed error is 20%. Then we can calculate that the maximum allowed r is

$$r = 0.2/(1/2) = 0.4 \quad \text{for } L = 3$$

$$r = 0.2/(3/4) \approx 0.3 \quad \text{for } L = 5 \qquad (10)$$

Since the bigger the ratio r, the faster the algorithm converges, we try to use the largest r that we can possibly use in our algorithm. In practice, we select $r=0.3$ in our work.

Experimental Results

As our aim is to extract representative frames without shot detection, here we classify the videos into two types, one type is the video with explicit shot boundaries and the other is without explicit boundary. We provide experimental results on the two types of videos to see how the algorithm works effectively.

1) Video without explicit shot boundaries

The content of our test video "mjackson" is shown in Table 3. The morphing technique has been used in the video sequence. The changes across the sections are gradual and it is very difficult even for human beings to define precisely where the shot boundaries exist. So, we only give a rough boundary in the table. There have been a lot of techniques proposed by researchers on video segmentation, yet none of them has claimed work on this kind of video. Figure 7(a) shows the consecutive frame histogram differences of "mjackson". If we apply the threshold method proposed in Zhang *et al.* [4], apparently we can not detect any shot boundary in it. However, using CBAC, we successfully extract representative frames from the video according to content changes. When the required number reaches 5% of the length of the original video, the main information of the video is still captured very well in the representative frames, as shown in Figure 5.

2) Video with explicit shot boundaries

Table 3 shows the content of our test video "news". In sections 1 and 4 there are only people speaking and the video content changes very little, This can also be seen in Figure 8(a) which shows the distances to the first frame for all frames in the video. The result shows that though the two sections are much longer than other sections, the number of extracted representative frames from them are smaller than all other sections for all the fractions selected. Representative frames are extracted with

Table 2. The Content of "news"

Section	Range	Content
1	0–67	Two anchor persons are speaking
2	68–109	The first girl is receiving award
3	110–169	A man is receiving award
4	170–258	An official is commenting
5	259–306	The second girl is receiving award
6	307–329	The third girl is receiving award

Table 3. The Content of "mjackson"

Section	Range	Content
1	0–41	The first man' head is moving
2	38–89	The first girl' head is moving
3	89–150	The second girl' head is moving
4	148–205	The second man' head is moving
5	203–243	The third girl' head is moving

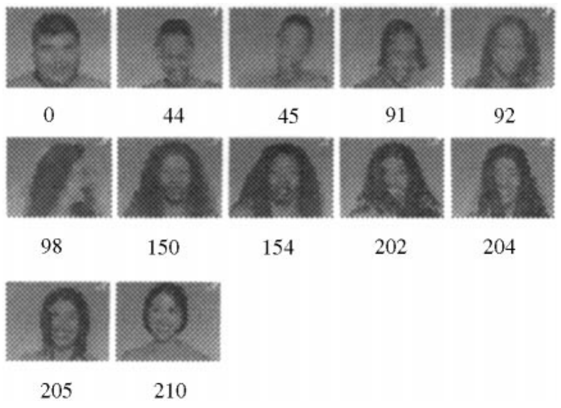

Figure 5. The representative frames in Figure 6(e) of "mjackson".

respect to the changes of content selected, shown in Figure 8. We can also see from the figure, that although we do not explicitly segment a video into shots, the algorithm automatically produces representative frames following the change of shots. Therefore, it can reflect the content of the video very well. Figure 6 shows the representative frames in Figure 8(e).

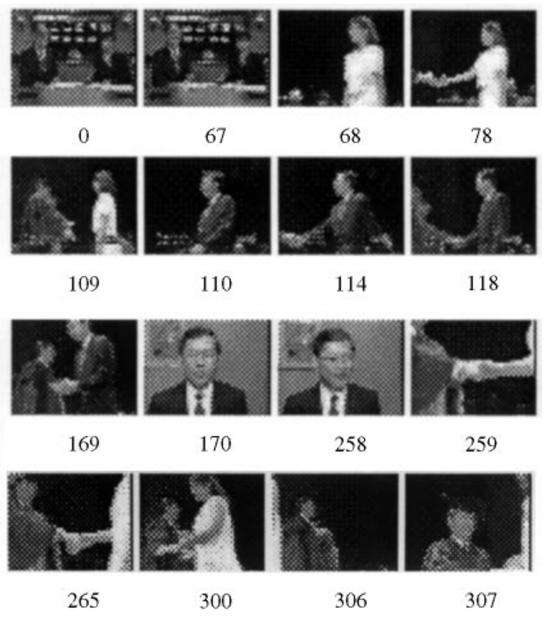

Figure 6. The representative frames of Figure 1(e) in "news".

Video summarization using R-sequences

To gather information from videos, we go back and forth between three stages: grazing, browsing and watching [10]. In the grazing stage, the user is passive, just waiting for interesting information to appear. In the browsing stage, a user interactively searches for information with no specific target in mind. In the watching stage, a user concentrates on understanding information. Since video data is usually very large, before starting a video and beginning the understanding stage, it is worthwhile to spend sometime getting a general idea about its content. After the extraction of representative frames, we obtain a subset which represents the important content of a video very well. It therefore provides a good basis for the development of video summarization tools. Based on our content-based representative frame extraction technique, we have developed a CBAC video player system which has the functions of content-based video summarization and content based browsing.

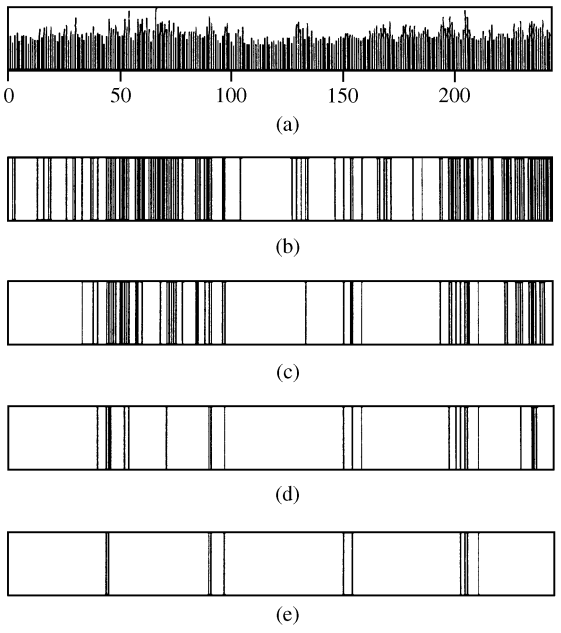

Figure 7. Experiment result on "mjackson". The vertical lines in (b), (c), (d) and (e) represent the positions of extracted representative frames. $L=3$, $r=0{\cdot}3$. (a) frame to frame histogram differences, (b) 50 percent frames extracted, (c) 25 percent frames extracted, (d) 10 percent frames extracted, (e) 5 percent frames extracted.

R-Sequence generation and video summarization

Browsing tools provide interactive functions which help users to get hierarchical views of a video, from

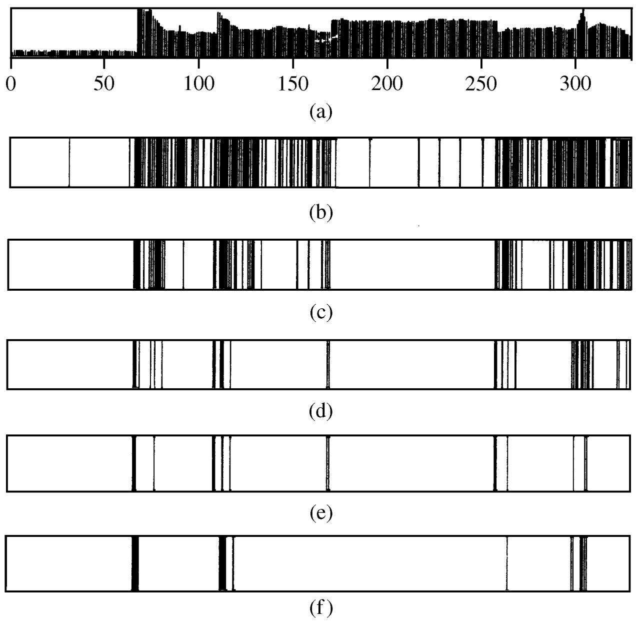

Figure 8. Experiment result on "news". The vertical lines in (b), (c), (d) and (e) represent the positions of extracted representative frames. $L=3$, $r=0.3$, in (f), $L=5$, $r=0.3$. (a) histogram distance to the first frame, (b) 50 percent frames extracted, (c) 25 percent frames extracted, (d) 10 percent frames extracted, (e) 5 percent frames extracted, (f) 5 percent frames extracted.

a global view to a detailed one. The interactive functionality of a browsing tool is very useful. However, the interaction process itself involves a lot of feedback between the user and the computer. There can be anywhere from 500 to 1000 shots per hour in a typical video program [15]. If one frame of each shot is selected from the video, it will compose a very large structure graph. So, traversing the graph is still very time consuming for many end users. Therefore, many users would prefer a grazing view over the traversal of a browsing structure.

Zhang *et al.* [4] have proposed a way of finding representative frames within a shot. The extracted frames are output sequentially for video review. However, from the process of frames extraction we can see that these frames are of different content.

From our experiments we have found that if a sequence of unrelated pictures is output sequentially at the normal frame rate, the user will find it very difficult to grasp some information from it. In our content-based video summary, we use the *representative sequences* (*R-Sequences*) which are composed of a representative frame plus its several successive frames to describe a video. The length of the following frames is called the *smoothing factor* and can be tuned by a user to obtain a smooth output. if we use (1) to describe representative frames, then representative sequences can be described as:

$$\begin{aligned} V'' &= V'_1 \circ V'_2 \circ \cdots \circ V'_{N'} \\ &= \{R_{p11}, R_{p22}, \ldots, R_{p1s}\} \\ &\quad \circ \{R_{p21}, R_{p22}, \ldots, R_{p2s}\} \circ \cdots \\ &\quad \circ \{R_{pN'1}, R_{pN'2}, \ldots, R_{pN's}\} \end{aligned} \tag{11}$$

where

- $S \geq 0$, is the smoothing factor
- $V'_1 = \{R_{pi}, R_{pi1}, \ldots, R_{piS}\}$, is the representative sequence of R_{pi} of feature P, $i \in [1, N']$
- $V'_1, V'_2, \ldots, V'_{N'}$ and $V'' \subseteq V$
- $\circ$ is the temporal concatenation operation. The representative sequences are connected sequentially and compose a new video.

So, given a video V, we can obtain the representative frames by using CBAC with an appropriately specified ratio. These representative frames can be augmented by "S" successive frames and this entire concatenated sequence constitutes the summary video of V. If $S=0$ then $V'' = V'$ and the result is the representative frame sequence. From our experiments, we find that to obtain a visually pleasing result, $S=5$ is the smallest smoothing factor required.

Video browsing based on R-sequences

The functionality of content-based browsing is often compared to the function of a video cassette recorded (VCR). It is the next logical step beyond simple browsing using the fast-forward/rewind (FF/REW) buttons. After the development of shot detection techniques, many efforts have been made by researchers on how to provide an effective browsing tool. These efforts range from using a representative frame in a shot in Arman *et al.* [8], to the clustering of shots by Yeung *et al.* [16] and Zhong *et al.* [17], to the clustering of story units of Yeung *et al.* [18]. All of the development of such browsing techniques concentrates on providing a general structural description of a video.

In practice, a user mostly wants to use the VCR's fast forward and rewind functions. The user may just want to skip over some uninteresting sequence of frames when watching a video. Because within a shot there maybe many changes, skipping a shot will probably not meet the requirement of the user. The user in fact would like to skip to the next interesting part of a video.

Zhang *et al.* [10] have proposed a way for extraction of frames from a video for content-based video browsing. But their work does not provide a way to obtain a fixed number of frames for video description. Based on our CBAC technique, we can select each representative frame to represent its according R-SEQUENCE. We can then let the user select on-line his feature of interest and the percent of representative frames he would like to use for skipping. The user is thus provided with a very flexible tool. This is actually video summarization in single frame level, the smoothing factor S is therefore shrinked to one.

CBAC MPEG Video player

The video browsing and summary technique has been incorporated in our CBAC system. The system works on the Sun Solaris system and has been written in C using Motif (The software is written based on Brown's MPEG decoders, which is written around the Berkeley's decoders). The user interface of the system is shown in Figure 9.

The representative frames pre-computed at different percents and on different features are indexed and stored in the user data are of a given MPEG video data or in a separate file. This pre-computation can be done off-line (For example, it may take around two hours to extract the r-frames from a one hour video for a typical feature using a SGI ORIGIN 200 machine). When a user opens a video file in the system, he has three choices: play, summary and browsing. If he only wants to play the video, he can set the "Ratio" to 100% and click the play button.

If the user wants to perform content-based reviewing, then he has to select his content of interest by changing the "Feature" first, followed by selecting a ratio ($<100\%$) of the representative frames he wants to use. In the case that he wants to do content-based browsing, he may then just click on the FF/REW buttons to skip to the positions of his interest. Each time the user clicks the FF/REW button, the system will automatically jump to the next/last consecutive representative frame. If he wants a content-based video summary to be played, he has to select the smoothing factor as well. After the selection of a proper smoothing factor, he can press the play button and a representative sequence is displayed. The rightmost button on the top row displays the current frame number.

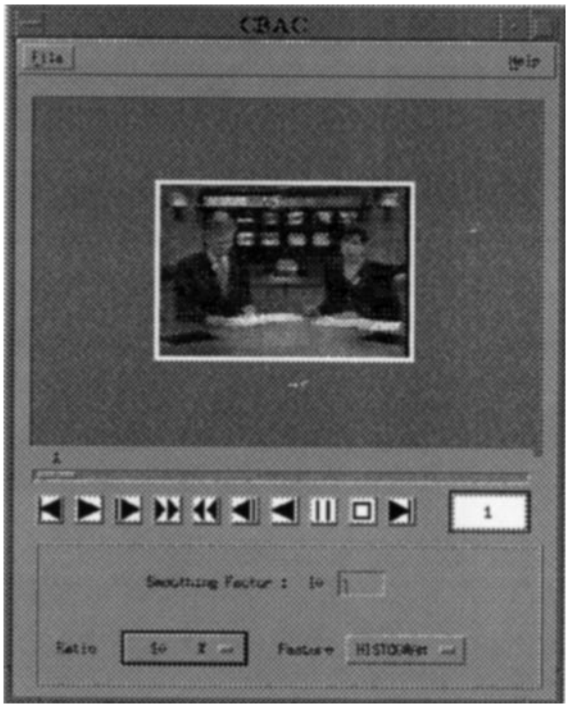

Figure 9. The CBAC MPEG video player.

Future work and discussions

Given a certain video for description, different people may give different subjective interpretations. This is because there is no standard hierarchical information structure in a video. Automatic processing of video story plots is still beyond current computer vision techniques. Yet, it is useful to develop new techniques for content-based video processing with the objective of helping users manage video information as much as possible.

The extraction of representative frames is the key step in many areas of video processing. In this paper, we propose a general approach (CBAC) which uses an adaptive clustering technique for the content-based extraction of representative frames of a digital video without shot boundary detection. We analyse all the frames of a video together for content changes without segmenting video into shots. The changes of the content in iteratively increasing units are compared globally to determine which part of a video is important for description. The proposed approach has been applied to histogram analysis and shows promising results.

Our ongoing work includes applying this technique into the analysis of motion, shape and texture which will work together to provide more advanced functions.

Since the method proposed here actually provides a way of temporally segmenting video into micro-sequences in multisale, it also provides a good basis for MPEG-7 [19] description scheme which is also a topic we are working on. Also, since the algorithm explores the general temporal information, we can apply it to compression domain as well. One example is our work on a motion activity descriptor [20].

We have proposed our algorithm concentrating on analyzing video without shot detection. The algorithm can extract as small as 5 percent length of representative frames from a video. The small percentage is very suitable for our application on video content-browsing and content-summary. This is because even if a video has explicit shot boundaries, a shot is on average over 86 frames long (a typical video would have from 500 to 1000 shots per hour [15], here we assume 24 FPS) and thus 5 percent of representative frames will still retain the video content structure. When a number less than the number of shots is required, or a user simply wants to search for very few highlights, however, which frame to drop and which frame to extract becomes a problem. Another issue regarding the length of video is how to improve the speed when a video is very long. In this case, we can segment a video into large segments before applying the CBAC algorithm. Since video has a stationary feature, this initialization will improve the processing speed while maintain a good segmentation result.

Future work in this area also includes porting the player system to the Internet. The dependencies of frames of some video formats such as MPEG-1 and MPEG-2 [19] make it difficult for saving bandwidth when we want to skip some frames during data transfer. The smallest transferred bit stream layer should then be Group of Pictures (GOP) for a completed prediction process. We can use GOP as the transferred unit in our content-based summary, but the strategy apparently is not suitable for content-based browsing. Therefore, a solution to this problem is also very important.

References

1. Sun, X., Kankanhalli, M., Zhu, Y. & Wu, J. (1998) Content-Based Representative Frame Extraction for Digital Video. *International Conference on Multimedia Computing and Systems*, 190–193.
2. Smith, M.A. & Kanade, T. (1995) Video Skimming for Quick Browsing based on Audio and Image Characterization. *Technical Report No. CMU-CS-95-186*, School of Computer Science, Carnegie Mellon University.
3. Zhang, H.J., Low, C.Y. & Smoliar, S.W. (1995) Video Parsing and Browsing Using Compressed Data. *Multimedia Tools and Applications*, **1**: 89–111.
4. Zhang, H.J., Kankanhalli, A. & Smoliar, S.W. (1993) Automatic partitioning of full-motion video. *Multimedia Systems*, **1**: 10–28.
5. O'Connor, C. (1991) Selecting Key Frames of Moving Image Documents: A Digital Environment for Analysis and Navigation. *Microcomputers for Information Management*, **8**(2): 119–133.
6. Mills, M., Cohen, J. & Wong, Y.Y. (1992) A Magnifier Tool for Video Data, *Proc. CHI'92*, 93–98.
7. Adam Finkelstein, Charles, E. Jacobs & David H. Salesin, (1996) Multiresolution Video. *Proc. of SIGGRAPH'96*, 281–290.
8. Arman, F., Depommier, R., Hsu, A. & Chiu, M.-Y. (1994) Content-based browsing of video sequences. *Proc. ACM Multimedia 94*, 97–103.
9. Boreczky, J.S. & Rowe, L.A. (1996) Comparison of Video Shot Boundary Detection Techniques. *Storage and Retrieval for Image and Video Databases IV, Proc. of IS&T/SPIE 1996 Int'l Symp. on Elec. Imaging: Science and Technology*, 170–179.
10. Zhang, H. J., Wu, J. H., Zhong, D. & Smoliar, S. W. (1997) An Integrated System For Content-Based video Retrieval And Browsing. *Pattern Recognition*, **30**(4): 643–658.
11. Dementhon, D., Kobla, V. & Doermann, D. (1998) Video Summarization by Curve Simplification. *ACM Multimedia 98, Electronic Proceedings*.
12. Hampapur, Jain, R. & Weymouth, T. (1994) Digital Video Indexing in Multimedia Systems. *Proc. Workshop on Indexing and Reuse in Multimedia Systems, American Association of Artificial Intelligence*.
13. Faloutsos, G., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D. & Equitz, W. (1994) Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems*, **3**: 231–262.
14. Taniguchi, Y., Akutsu, A., Tonomura, Y. & Hamada, H. (1995) An Intuitive and Efficient Access Interface to Real-Time Incoming Video Based on Automatic Indexing. *Proc. ACM Multimedia*, **95**: 25–33.
15. Aigrain, P., Zhang, H.J. & Petkovic, D. (1996) Content-based Representation and Retrieval of Visual Media: A State-Of-the-Art Review. *Multimedia Tools and Applications*, **3**(3): 179–202.
16. Yeung, M.M. & Liu, B. (1995) Efficient Matching and Clustering of Video Shots. *IEEE International Conference on Image Processing*, 338–341.
17. Zhong, H.J. & Zhang, S.-F. (1996) Clustering Methods for Video Browsing and Annotation. *SPIE Conference on Storage and Retrieval for Image and Video*.
18. Yeung, M.M., Yeo, B.L. & Liu, B. (1996) Extracting Story Units from Long Programs for Video Browsing and Navigation. *International Conference on Multimedia Computing and Systems*.
19. http://drogo.cselt.stet.it/mpeg/
20. Sun, X., Manjunath, B.S. *et al.* (1999) Motion Quantized α-Histogram as Video Unit Descriptor, *ISO/IEC/SC29/WG11 MPEG99/P075*.