Multimedia Systems 10: 98–115 (2004)
Digital Object Identifier (DOI) 10.1007/s00530-004-0142-7

Multimedia Systems

Exploring video content structure for hierarchical summarization

Xingquan Zhu[1], **Xindong Wu**[1], **Jianping Fan**[2], **Ahmed K. Elmagarmid**[3], **Walid G. Aref**[3]

[1] Department of Computer Science, University of Vermont, Burlington, VT 05405, USA
[2] Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA
[3] Department of Computer Science, Purdue University, W. Lafayette, IN 47907, USA

Published online: 15 September 2004 –

Abstract. In this paper, we propose a hierarchical video summarization strategy that explores video content structure to provide the users with a scalable, multilevel video summary. First, video-shot- segmentation and keyframe-extraction algorithms are applied to parse video sequences into physical shots and discrete keyframes. Next, an affinity (self-correlation) matrix is constructed to merge visually similar shots into clusters (supergroups). Since video shots with high similarities do not necessarily imply that they belong to the same story unit, temporal information is adopted by merging temporally adjacent shots (within a specified distance) from the supergroup into each video group. A video-scene-detection algorithm is thus proposed to merge temporally or spatially correlated video groups into scenario units. This is followed by a scene-clustering algorithm that eliminates visual redundancy among the units. A hierarchical video content structure with increasing granularity is constructed from the clustered scenes, video scenes, and video groups to keyframes. Finally, we introduce a hierarchical video summarization scheme by executing various approaches at different levels of the video content hierarchy to statically or dynamically construct the video summary. Extensive experiments based on real-world videos have been performed to validate the effectiveness of the proposed approach.

Keywords: Hierarchical video summarization – Video content hierarchy – Video group detection – Video scene detection – Hierarchical clustering

1 Introduction

Owing to the decreased cost of storage devices, higher transmission rates, and improved compression techniques, digital videos are becoming available at an ever-increasing rate. However, the manner in which video content is presented for access (such as browsing and retrieval) has become a challenging task, both for video application systems and for users. Some approaches have been described in [1–3] for presenting the visual content of the video through hierarchical browsing, storyboard posters, etc. The users can quickly browse through a video sequence, navigate from one segment to another to rapidly get an overview of the video content, and zoom to different levels of details to locate segments of interest.

Research in the literature [3] has shown that, on average, there are about 200 shots for a 30-min video clip across various program types, such as news and drama. Assuming that one keyframe is selected to represent each shot, the presentation of 200 frames will impose a significant burden in terms of bandwidth and time. Using spatially reduced images, commonly known as thumbnail images, one can reduce the size, but it will still be expensive if all shots must be shown for a quick browse of the content. Hence, a video summarization strategy is needed to provide users with a compact video digest that shows only parts of the video shots.

Generally, a video summary is defined as a sequence of still or moving pictures (with or without audio) presenting the content of a video in such a way that the respective target group is rapidly provided with concise information about the content, while the essential message of the original is preserved [4]. Three kinds of video summary styles are commonly used:

- A *pictorial summary* [3,7–13,18,19,25,30,31] is a collection of still images (possibly varying in size) arranged in time. The use of compact images provides the user with an overview of the video directly and efficiently; however, the motion and audio information of the video is lost.
- A *video skimming* [14–16,21–24,30] is a collection of video clips (e.g., video shots) arranged in a time series. Compared with still pictorial summaries, video skimming is more attractive to the users, since both motion and audio signals are preserved. However, the amount of time required for viewing a skimming suggests that skimmed video is not appropriate for quick browsing.
- A *data distribution map* [20–22] is a synthetic picture that illustrates the distribution of some specific data in the database. Due to the inherent complexity of images and videos, using pictorial frames or video skimming to

Correspondence to: X. Zhu (e-mail: xqzhu@cs.uvm.edu)
This research has been supported by the *NSF* under grants 9972883-EIA, 9974255-IIS, 9983248-EIA, and 0209120-IIS, a grant from the state of Indiana 21th Century Fund, and by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant DAAD19-02-1-0178.

abstract a video database may be impractical. The data distribution map, therefore, appears to be an efficient mechanism for providing users with an overview of the whole dataset. Unfortunately, such a distribution map may not be as friendly and intuitive as pictorial frames or skimming, since synthetic points or curves may not make sense for naïve users.

A survey of video summarization is given in [5]. When users are provided with a compact video digest, they can obtain video content quickly and comprehensively. Moreover, the power of visual summaries can be helpful in many applications such as multimedia archives, video retrieval, home entertainment, and digital magazines. To achieve a suitable video summarization for the various applications above, we introduce a hierarchical video summarization strategy that uses pictorial frames for presentation of the video summaries. In comparison with most approaches that use only visual and temporal clustering, we integrate the video content structure and visual redundancy among videos. Moreover, much emphasis has been put on scalable and hierarchical video summarization. The final results consist of various levels of static or dynamic summaries that address the video content in different granularities (Sect. 2, Definition 1).

The rest of the paper is organized as follows. In the next section, we review related work on video summarization. Our overall system architecture is presented in Sect. 3. In Sect. 4, we introduce techniques on video content hierarchy detection that are used to construct video content structure from keyframes, groups, and scenes to clustered scenes. With the constructed content structure as a basis, in Sect. 5 we discuss a hierarchical video summarization scheme that uses the hierarchical clustering algorithm to construct a video summary at different levels. In addition to supporting the passive browsing of statically constructed video summaries, a dynamic summarization scheme has also been integrated to allow users to specify their preferred summary length. In Sect. 6, the effectiveness of the proposed approach is validated by using real-world videos. Concluding remarks are given in Sect. 7.

2 Related work

In practical terms, video summarization is an inseparable research area for many video applications, including video indexing, browsing, and retrieval [1,2,29]. The earliest developed research on video summarization is perhaps best exemplified in Video Magnifier [6], where video summaries were created from a series of low-resolution frames that were uniformly sampled from the original video sequence at certain time intervals. The obvious drawback of such an arrangement is the loss of content in short but important videos. Accordingly, various research efforts have addressed the generation of video overviews using more suitable approaches [7–25]. Basically, the quality of a generated video summary is influenced by the following factors:

1. How the selected summary units are visualized.
2. What kinds of units are selected to generate the summary.
3. Whether the summary is static or dynamic (Definitions 1 and 2 below).

Since most research assumes the video summary is generated for efficient browsing, indexing, and navigating, the pictorial summary is popularly used to visualize the video summary via icon frames [3,7,11–13,25,30,31], mosaic images [18, 19,34], video objects [8–10], or a data distribution map [20–22]. A curve-simplification video-summarization strategy is introduced in [12], which maps each frame into a vector of high- dimensional features and segments the feature curve into units. The video summary is extracted according to the relationship between the units. In Video Manga [7], a pictorial video summary is presented with keyframes in various sizes, where the importance of each keyframe determines its size. Given a camera panning/tilting shot, however, none of the selected keyframes generated by the above strategies captures the underlying dynamics of the shot. Hence, the mosaic-based approach is utilized to generate a synthesized panoramic image that can represent the entire content in an intuitive manner [18,19,34]. Unfortunately, this approach works well only when specific camera motions are contained in the shot, and this critical drawback prevents it from being applied effectively to real-world videos. Instead of using features from the whole image for summarization, object- based strategies have been proposed [8–10]; however, due to the challenging task of object segmentation [17], it might be a long time before impressive results in general videos can be acquired.

Instead of abstracting videos by pictorial images, some approaches provide video skimming [14–16,21–24,30], which trims the original video into a short, highlight stream. In [4, 15], the most representative movie segments are extracted to produce a movie trailer. The approach in [24] uses a method that applies different sampling rates to the video to generate a summary. The sample rate is controlled by motion information in the shot. In addition to using visual features, the strategies in [21–23] utilize closed-caption and speech signals to select segments of "interest" for skimming. Video skimming may be useful for some purposes, since a skimmed video stream is a more appealing method of browsing for users. However, the amount of time required for viewing a skim suggests that it is not appropriate for a quick overview or transmission.

Defining what kinds of units should be selected is the most challenging and subjective process in creating video summaries. Many algorithms believe that the scenario (such as a story unit) is more suitable for summarization [3,30,31,42] than others, where video scenarios are detected using time-constrained clustering [3], speaker or subject change detection [30,31,42], etc. The summary is constructed using the representative frames (or shots) of the story units in the form of a storyboard or a skimming. A summarization scheme that integrates video scenarios outperforms most other strategies since the presentation of scenarios in the summary may be the most effective way to tell the user "what the video talks about". Unfortunately, although many videos from our daily life have scenario evolution (i.e., it is possible to detect story units), there are a significant number of videos that have no scenario structure, such as surveillance videos and sports videos. In these cases, a selection of video highlights might be more suitable for summarization. Generally, video highlights are selected by predefined rules, such as selecting frames with the highest contrast [4], close-up shots of leading action, special events like explosions and gunfire [15], frames with faces, camera motions, or text [21–23], or by considering the time

and data information of the video [14]. On the other hand, since domain knowledge may assist us in acquiring semantic cues, some abstracting strategies have been executed for specific domains, such as home videos [14], stereoscopic videos [8], online presentation videos [16], and medical videos [25, 30,42], where rules and knowledge followed by the videos are used to analyze semantics [26–28]. However, another problem emerges: rather than abstracting the video, highlight strategies usually present some selected "important" samples that address the video details but not the topic and theme; as a result, a glance at the constructed summary imparts very limited information about video content to users. Moreover, since most highlight selection strategies are domain specific, they cannot be extended to other videos easily. Therefore, compared with scenario-based strategies, the generality of highlight-based strategies is limited.

For the generation of static or dynamic summaries,many approaches supply a dynamic scheme that integrates user specifications to dynamically generate a summary online. Usually, a dynamic strategy requires the user to directly or indirectly specify the summary parameters, e.g., the length of the summary [4,8,9,13–15,24] (number of summary frames or time duration) or the types of interesting features for determining the summary frames (e.g., slides, close-ups of people) [44]. In this way, the scalability of the summary is preserved, since each user is supplied with a summary he/she prefers. Unfortunately, a user's specification for the summary is not always reasonable for addressing video content, especially if the user is unfamiliar with the videos in the database. Conversely, the approaches in [3,7,25,30,42] construct static summaries beforehand and supply all users with the same summary (the one that has been optimized by the system). However, static methods have the drawback of depriving the user of scalable summaries, which may be desirable in some cases. A dynamic scheme maintains the scalability of the summary but may not result in the best summary point for naive users, and a static scheme may supply the users with an optimal point in generating the summary, but the scalability is lost. Obviously, neither is perfect, and a new approach that integrates both may provide a better solution.

Ideally, a video summary should briefly and concisely present the content of the input video source. It should be shorter than the original, focus on the content, and give the users an appropriate overview of the whole video. However, the problem is that *appropriateness* varies from user to user, depending on the user's familiarity with the source and genre, and with a given user's particular goal in watching the summary. Hence, hierarchical summarization strategies [13,20, 30,42] have been introduced to support various levels of summary and assist users in determining what is *appropriate*. In [13], a keyframe-based hierarchical strategy is presented, where keyframes are organized in a hierarchical manner from coarse to fine temporal resolution using a pairwise clustering method. Unfortunately, this strategy is unable to merge frames that are visually similar but temporally separated. By considering video content structure and domain knowledge, a multilevel static summary [30,42] has been constructed for medical video data. This strategy, however, still limits the scalability of the summary by supplying only the static summaries, and, moreover, it works only on medical video data.

All the reviews above support the following observations:

1. For efficient presentation and browsing purposes, a pictorial summary may outperform video skimming.
2. Since defining video highlights is a relatively subjective and domain-dependent process, if the video content structure is available, a scenario-based summarization strategy might be a better choice.
3. Due to the obvious problems in static and dynamic summarization, an efficient scheme should integrate the two strategies for maximum benefit.
4. To capture an *appropriate* summary for a video overview, a hierarchical strategy should be adopted.

Based on these observations, we propose a hierarchical video summarization strategy. We use pictorial keyframes to visualize the video content and introduce a hierarchical strategy to construct multilevel summaries, increasing in granularity from top to bottom. Moreover, instead of selecting video highlights, we use the semantically richer video units (scenes and groups) for summarization. In this paper, we concentrate on summarizing videos with content structures. For those having no "story lines", the highlight selection scheme might be the best choice for summarization. Before we explore video summarization techniques, it would be worthwhile to define the terminology that will be used in our analysis.

Definition 1. A *static summarization scheme* is an approach that requires no user participation for summary generation. For all users of the same video, this strategy will generate the same summary, which is called a *static summary* in this paper.

Definition 2. A *dynamic summarization scheme* is an approach where user actions may result in differences in the generated summary. The summary generated by this scheme is called a *dynamic summary* in this paper.

Definition 3. The *video content structure* is defined as a hierarchy of clustered scenes, video scenes, video groups, and video shots (keyframes), increasing in granularity from top to bottom in addressing video content, as shown in Fig. 1. A content structure can be found in most videos in our daily life.

Definition 4. A *video shot* (denoted by S_i) is the basic element in videos; it records the frames resulting from a single continuous running of the camera, from the moment it is turned on to the moment it is turned off.

Definition 5. A *keyframe* (denoted by K_i) is the frame that is representative to describe the content of a given shot. Depending on the content variance of the shot, one or multiple keyframes can be extracted.

Definition 6. A *video supergroup* (denoted by SG_i) consists of visually similar shots within an entire video.

Definition 7. A *video group* (denoted by G_i) is an intermediate entity between physical shots and semantic scenes and consists of visually similar shots that belong to the same story unit.

Definition 8. A *video scene* (denoted by SE_i) is a collection of semantically related and temporally adjacent groups depicting and conveying a high-level concept or story.

Definition 9. A *clustered scene* (CSE_i) is a collection of visually similar video scenes.

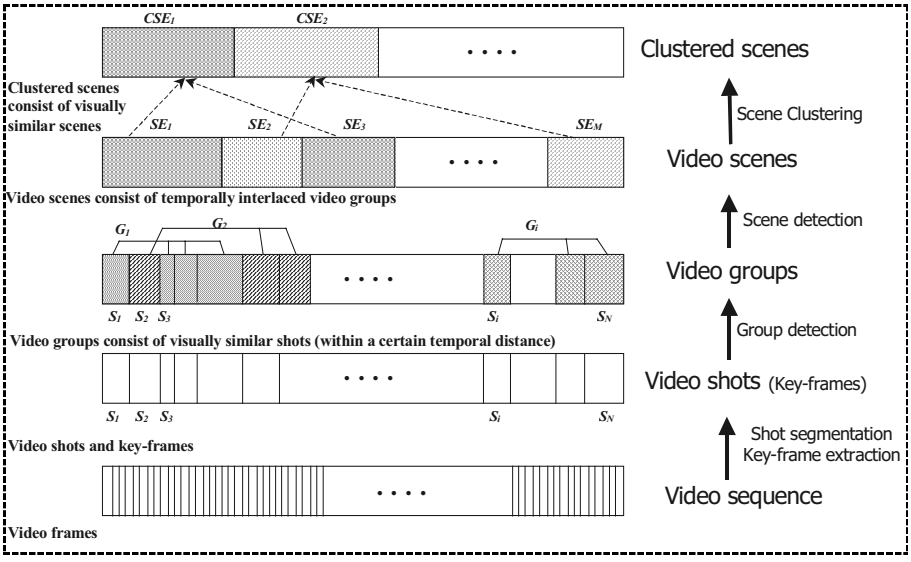


Fig. 1. Pictorial video content structure

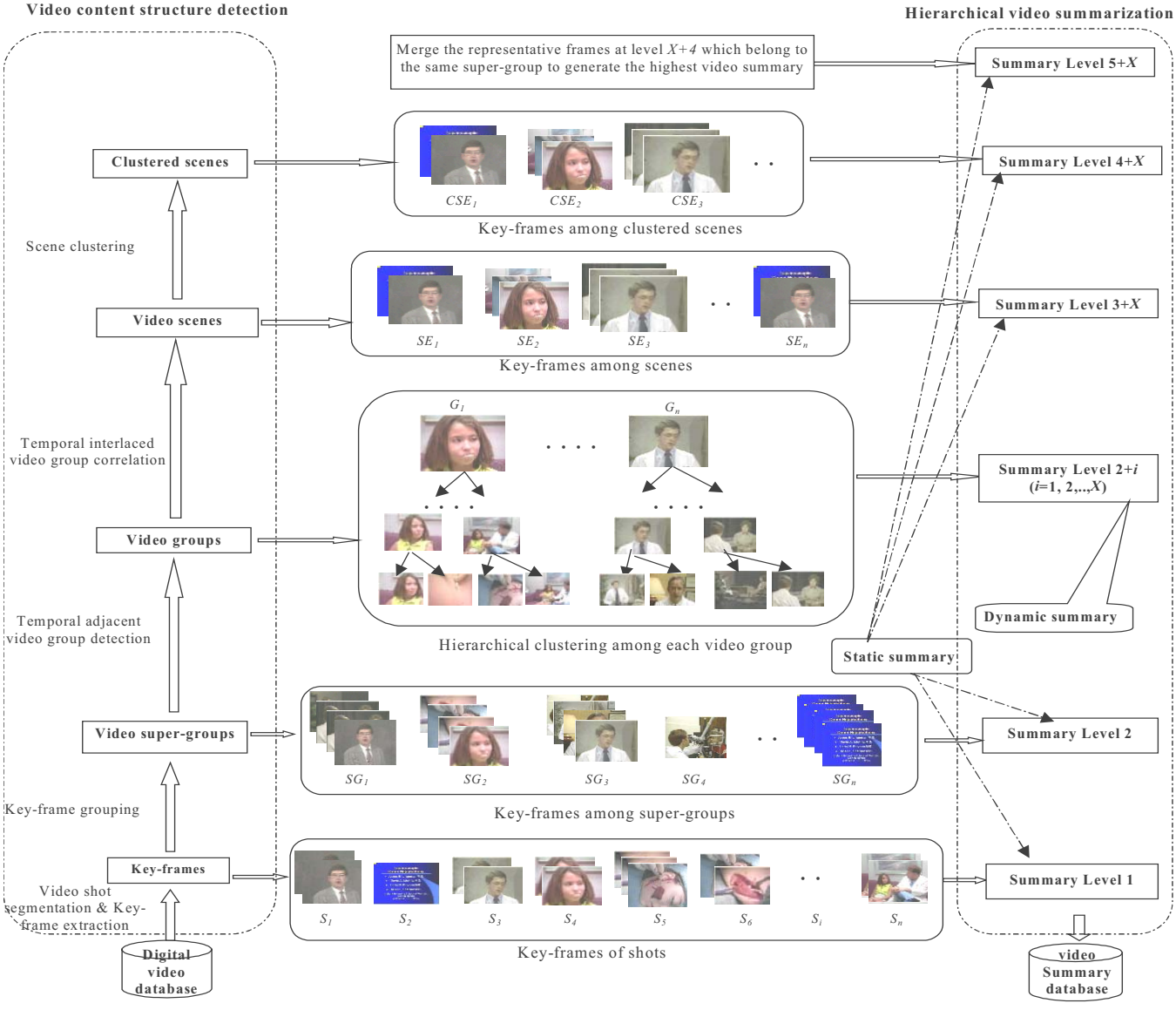


Fig. 2. System architecture

3 System architecture

Figure 2 shows that our system consists of two relatively independent components: *video content structure detection* and *hierarchical video summarization*.

To explore video content structure, the shot segmentation and keyframe extraction are first applied to parse continuous video sequences into independent shots and discrete keyframes. For all extracted keyframes, an affinity matrix is constructed to explore the correlations among them. With the transferred affinity matrix, the visually similar shots are merged into *supergroups*. For each supergroup, the video shots with temporal distance smaller than a predefined threshold are then taken as one *video group*. Generally, a *video scene* consists of temporally interlaced (or visually similar and neighboring) video groups to convey a relatively independent story unit. Accordingly, the temporal and visual information among them are integrated to cluster video groups into scenes. To eliminate visual redundancy caused by repeated scenes in different parts of the video, the scene-clustering strategy is employed to merge visually similar scenes into unique clusters. As a result, a video content hierarchy from clustered scenes, scenes, groups to keyframes is constructed.

Given the detected video content structure, an $X + 5$ level video summary (X indicating the dynamic summary level, which is determined by the user's specification of the summary length) is constructed, where the summary becomes more concise and compact from lower to higher levels. The lowest summary level consists of keyframes of all shots. By utilizing the affinity matrix, the visually similar shots are clustered into supergroups. It is likely that any supergroup containing only one shot will convey very limited content information. Hence, by eliminating supergroups with only one shot, the summary at the second level consists of all remaining keyframes. For each supergroup, a predefined threshold is applied to segment the temporally adjacent member shots into groups. Given any video group, the *Minimal Spanning Tree* [50] algorithm is utilized to cluster keyframes into a hierarchical structure. A dynamic summarization scheme is employed to select a certain number of representative frames from each group to form the user-specified dynamic summary. Beyond the dynamic summary levels, the summary at level $X + 3$ is constructed by eliminating groups that contain less than three shots and do not interlace with any other groups, since they probably contain less scenario information. With the scene-clustering scheme, visual redundancy among similar scenes is partially removed. Consequently, the summary at level $X + 4$ consists of representative frames of all clustered scenes. It is not unusual for different video scenes to contain visually similar video groups, and eliminating similar groups provides the user with a visually compact summary. Hence, the summary at the highest level is constructed by merging all representative frames at *level X+4* that belong to the same supergroup.

4 Video content structure detection

Most videos in our daily life have their own content structure, as shown in Fig. 1. While browsing videos, the hidden structure behind frames (scenes, groups, etc.) conveys themes and scenarios to us. Hence, the video summary should also fit with this structure by presenting overviews at various levels and with various granularities. To achieve this goal, we first explore the video content structure.

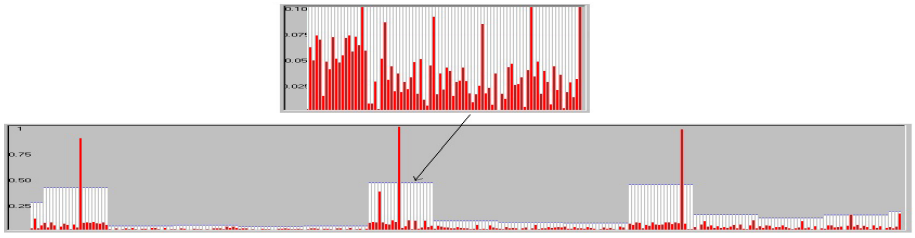
The simplest method of parsing video data for efficient browsing, retrieval, and navigation is segmenting the continuous video sequence into physical shots [35,37,38] and then selecting a constant number of keyframes for each shot to depict its content [1,2,32,43,45,47]. Unfortunately, since the video shot is a physical unit, it is incapable of conveying independent scenario information. To this end, many solutions have been proposed to determine a cluster of video shots that conveys relatively higher- level video semantics. Zhong et. al [33] propose a strategy that clusters visually similar shots and supplies the user with a hierarchical structure for browsing. Similar strategies are found in [11,32,43,44]. However, spatial shot clustering considers only visual similarities, and the video context information is lost. To address this problem, Rui et. al [39] present a method that merges visually similar shots into groups and then constructs a video content table by considering temporal relationships among groups. A similar approach has been reported in [36]. In [41], a time- constrained shot-clustering strategy is proposed to parse temporally adjacent shots into clusters, and a scene transition graph is constructed to detect video scenes by utilizing the acquired cluster information. A temporally time- constrained shot-grouping strategy has also been discussed in [40,42] that considers the similarity between the current shot and the shots preceding and following it (within a small window size) to determine the boundary of the video scene.

In a narrow sense, both temporal and spatial clustering schemes above have their disadvantages in exploring video content. The spatial clustering ignores temporal information, so video context information will not be available in the final result. On the other hand, the temporal clustering crucially depends on a predefined distance function that integrates the temporal window size and visual similarity between shots, but this function may not fit with user perceptions very well [39,36,40–42]. In this section, a new method is proposed to detect video content structure. We first discuss techniques to determine shots and keyframes. Then, an affinity matrix is constructed to cluster visually similar shots into supergroups. Although visually similar shots have a high probability of belonging to one scene, similar shots with a large temporal distance often belong to similar scenes in different parts of the video, e.g., similar dialogs between two actors may appear in different parts of the movie. Therefore, in each supergroup, adjacent shots with their temporal distance less than a predefined threshold are segmented into video groups. Due to the fact that video scenes usually consist of interlaced video groups (or visually similar and neighboring groups), a video-scene- detection algorithm is proposed by considering visual and temporal information among video groups. Finally, we apply a scene-clustering scheme that merges visually similar scenes to remove visual redundancy and produces a compact summary.

4.1 Video shot detection and keyframe extraction

To support shot-based video content access, we have developed a shot-detection technique [38] that works on *MPEG*

a



b

Fig. 3. The video shot detection results from a medical video. **a** Detected video shots. **b** Corresponding frame difference and the determined threshold for different video shots, where the small window shows the local properties of the frame differences

compressed videos. It adaptively adjusts the threshold for shot detection by considering the activities in different video sequences. Unfortunately, such techniques are not able to adapt the thresholds for different video shots within the same sequence. To adapt the thresholds to local activities within the same sequence, we adopt a small window (e.g., 2–3 s), and the threshold for each window is adapted to local visual activity by applying our automatic threshold detection and local activity analysis techniques. Figure 3 demonstrates the results from one video source in our system. As we can see, the threshold has been adapted to small changes between adjacent shots for successful shot segmentation, such as changes between eyeballs from various shots in Fig. 3a.

After shot segmentation, the keyframe for each shot is extracted using the following strategy.

1. Given shot S_i, its tenth frame is always taken as the keyframe of the shot, and this frame is denoted by K_1. (In this way, we may avoid selecting the gradual effect frames between shots as keyframes).
2. From the position of the most recently select keyframe, sequentially calculate the similarity between the succeeding frames (F_i) and all former selected keyframes K_l ($l = 1, 2, \ldots, N$) using Eq. 1. If this value is less than a threshold $T_{keyframe}$ (which is determined by the threshold used to detect the current shot), the current frame F_i is taken as a new keyframe and added to the keyframe set.

$$KFmSelect(F_i) = Max\{FmSim(F_i, K_l); \\ l = 0, 1, \ldots, N\}, \quad (1)$$

where $FmSim(F_i, K_l)$ indicates the visual similarity between F_i and K_l, which is defined by Eq. 2.
3. Iteratively execute step 2 until all frames in shot S_i have been processed.

To calculate the visual similarity between frames, three sets of visual features (256-dimensional HSV color histogram, 10- dimensional tamura coarseness, and 8-dimensional tamura directionality texture) are extracted. Suppose $H_{i,j}, j \in [0, 255]$, $TC_{i,j}$, $j \in [0, 9]$, and $TD_{i,j}$, $j \in [0, 7]$ are the normalized color histogram, coarseness, and directionality texture of frame F_i. The similarity between F_i and F_j is defined by Eq. 2:

$$FmSim(F_i, F_j) = W_c \sum_{k=0}^{255} \min(H_{i,k}, H_{j,k}) \\ +W_{TC}\left(1 - \sqrt{\sum_{k=0}^{9}(TC_{i,k} - TC_{j,k})^2}\right) \\ +W_{TD}\left(1 - \sqrt{\sum_{k=0}^{7}(TD_{i,k} - TD_{j,k})^2}\right), \quad (2)$$

a

b

Fig. 4. Keyframe extraction results. **a** Constantly sampled frames in a given shot, where similar frames appear repeatedly (from *left* to *right*, *top* to *bottom*). **b** Extracted keyframes with our strategy

where W_C, W_{TC}, and W_{TD} indicate the weight of each feature. We set $W_C = 0.6$, $W_{TC} = 0.2$, $W_{TD} = 0.2$ in our system.

Figure 4 presents the detected keyframes from a single shot, where Fig. 4a shows constantly sampled frames in the shot (a doctor examines a patient's face) and Fig. 4b denotes the keyframes detected with our strategy. In some cases, similar frames may appear repeatedly in different locations of a shot. The strategy that considers only the similarity between the current frame F_i and the most recently selected keyframe [47] may result in redundancies among keyframes. With our strategy, the selected keyframes are those that exhibit enough dissimilarity (low redundancy) and capture the most visual variances of the shot.

4.2 Video supergroup detection

After the keyframe extraction, we apply a new strategy to merge visually similar shots into one cluster (supergroup). The following steps will segment the temporally adjacent shots within each supergroup into semantically richer units.

Assume N keyframes K_l ($l = 1, \ldots, N$) have been extracted from a given video. To explore the correlations among them, an affinity matrix M [48,49,52,53] is constructed, with entries of the matrix given in Eq. 3

$$M(i,j) = M(j,i) = Exp[-(1 - FmSim(K_i, K_j))^2/\sigma]; \quad i,j \in [1,N], \tag{3}$$

where $FmSim(K_i, K_j)$ is the visual similarity between K_i and K_j (defined by Eq. 2) and σ is a constant scaling factor for stretching ($\sigma = 0.5$ in our system). In fact, $M(i,j)$ is a self-correlation matrix, where the relationship among keyframes of the current video can be visualized and analyzed directly. As shown in Fig. 7c, where the transferred affinity matrix of 105 keyframes is utilized to explore the correlations, the gray intensity in the figure indicates the similarities between elements: the higher the intensity, the greater the similarity.

To address the correlations within the affinity matrix M (in the following sections we use M as an abbreviation of $M(i,j)$) and adopt the Scott and Longuet-Higgins (SLH) [49] relocalization algorithm since it has been successfully utilized in image segmentation [52,53] and video event detection [46]. Given an affinity matrix M and a number k, SLH outputs a newly transferred matrix Q that is calculated in the following steps [48]:

1. Calculate the eigenvector of affinity matrix M, rank the eigenvectors in descending order by their eigenvalues, and denote the ranked eigenvectors by $E_1, E_2, .., E_N$.
2. Construct the affinity matrix W whose columns are the first k eigenvectors of M, i.e., $W = [E_1, E_2, .., E_k, 0, ..0]$.
3. Normalize the *rows* of W with Eq. 4 so that they have the unit Euclidean norm:

$$W(i,\rightarrow) = W(i,\rightarrow)/\|W(i,\rightarrow)\|. \tag{4}$$

4. Construct the matrix Q with $Q = WW^T$, i.e., $Q_{i,j} = \sum_{r=1}^{k} W_{ir}W_{jr}$. Then, each element $Q_{i,j}$ can be interpreted as an enhanced measurement of the interaction between keyframes K_i and K_j.
5. Segment the points by looking at the elements of Q. Ideally, the larger $Q_{i,j}$ is, the higher the probability that keyframes K_i and K_j belong to one group. Figure 7c represents the constructed Q matrix of 105 keyframes with the top 10 eigenvectors.

4.2.1 Q matrix processing

As we have indicated, each element of matrix Q provides information about whether a pair of keyframes belongs to the same group. Now, the problem of grouping the keyframes has been reduced to that of sorting the entries of matrix Q by swapping pairs of rows and columns until it becomes block diagonal, with each diagonal indicating one cluster. Hence, the Q matrix processing for keyframe grouping consists of two steps: (1) sorting and (2) diagonal block detection, as shown in Fig. 5.

By swapping pairs and rows of matrix Q, highly correlated keyframes are merged together. To do this, we apply an energy maximal resorting process to rearrange the matrix into diagonal block styles, as shown in Fig. 6.

Given Q matrix, $Q_{i,j}$ ($i = 1, \ldots, N$; $j = 1, \ldots, N$), select one row as the seed for sorting. At each iteration, say, k, the current state is represented by a $k \times k$ submatrix Q^{*k} that contains the sorted keyframes so far. To select a candidate keyframe for resorting, a cost function (Eq. 5) is given by the energy of the first k elements,

$$E_h^k = \sum_{l=1}^{k} Q_{l,h}^2 \quad \text{for} \quad (h = k+1, \ldots, N), \tag{5}$$

which represents the total energy of interaction between each of the candidate keyframes and the set of already sorted keyframes. By maximizing the cost function E_h^k, our search strategy selects the keyframe whose global energy of interaction with the current segmentation is the largest.

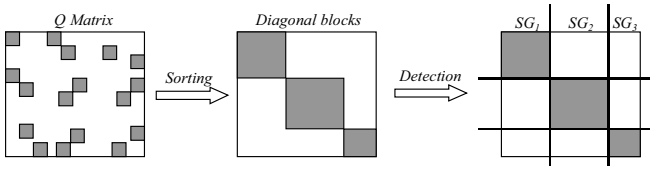


Fig. 5. Q matrix processing

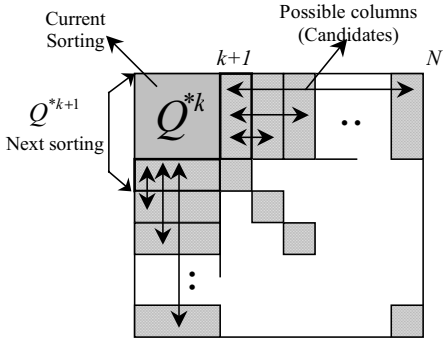


Fig. 6. Q matrix sorting algorithm: at iteration k, columns $k+1$ to N are permuted and the column with the highest norm selected to form Q^{*k+1}

The updated state Q^{*k+1} is obtained by augmenting Q^{*k} with the column and row of the best selected keyframe. The column corresponding to this keyframe is first permuted with column $k+1$, followed by a permutation of rows with the indices. Matrix Q^{*k+1} is then formed with the first $(k+1) \times (k+1)$ elements of the permuted shape interaction matrix. As a result of this maximization strategy, submatrix Q^{*k+1} has the maximal energy among all possible $(k+1) \times (k+1)$ submatrices of Q.

Recursively execute the resorting steps above until all rows have been selected. The rows and columns of matrix Q will then be sorted in descending order of similarity. Due to the fact that visually similar keyframes have similar interactions with all other keyframes, the resorted matrix Q will take a roughly block diagonal form, as shown in Fig. 5b, with each diagonal block representing one cluster of keyframes (supergroup). The successfully detected diagonal block will help us in determining the final keyframe clusters.

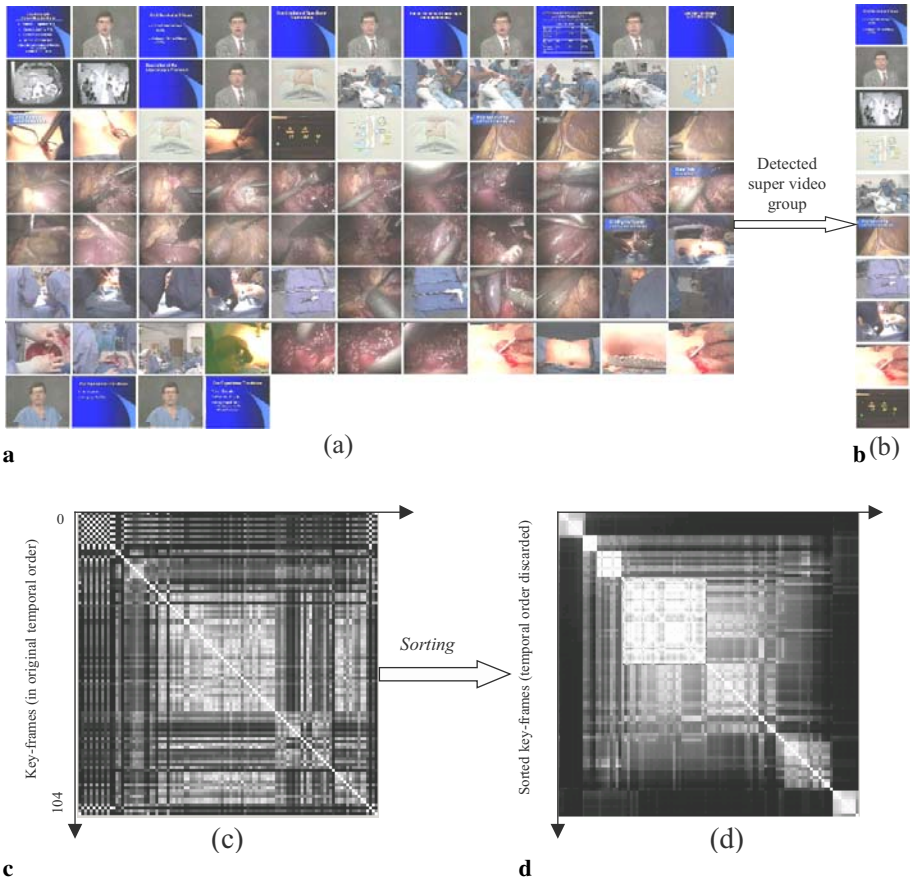
To sort the Q matrix, a critical row (the *seed row*) should be selected before sorting. Since it is not possible to optimize the selection of such a seed, we randomly select several rows as seeds and determine the corresponding sorted Q matrices. Those that have distinct diagonal block results are retained for further processing.

After the Q matrix has been sorted into diagonal blocks, various approaches might be utilized to detect the boundary (clusters) of the blocks [52]. However, since we do not actually know how many diagonal blocks the Q matrix contains, the performance of these strategies might be decreased since most of them need information about the number of clusters in advance. Therefore, a simple threshold scheme is applied to eliminate those elements belonging to a predefined threshold and detect the blocks along the diagonal. The successfully detected elements in one block are taken as one supergroup.

As we indicated above, instead of using only one keyframe as the seed, we execute the algorithm several times using different seeds. The union of all detected supergroups is taken as the detection result. For example, if K_2, K_9, K_{14}, K_{35} are detected as one supergroup in the first iteration, and K_2, K_9, K_{22} are detected as one supergroup in the next iteration, their union, $K_2, K_9, K_{14}, K_{22}, K_{35}$, will be taken as the final detected supergroup.

4.2.2 Keyframe grouping for supergroup detection

After the diagonal blocks have been detected, the keyframes belonging to the same block are taken as one supergroup, denoted by SG_i. The keyframe that does not belong to any diagonal block is taken as a separate supergroup. Given a video shot S_i, assume there are T keyframes $K_l\,(l = 1, \ldots, T)$ contained in S_i. In the ideal situation, all T keyframes should be merged into one supergroup. Unfortunately, this assumption does not always hold due to the fact that a shot may exhibit significant visual variances. Suppose all T keyframes in S_i have been clustered into NSG supergroup $SG_l (l = 1, \ldots, NSG)$ with T_1 keyframes contained in SG_1, T_2 keyframes contained in SG_2, and so on. It is obvious that $T_1 + T_2 + \ldots + T_{NSG} = T$. The final supergroup to which S_i belongs is selected using



a **b** **c** **d**

Fig. 7. Video supergroup detection. **a** Sampled frames from 105 keyframes (a kidney transplant surgery), with its original temporal order from *left* to *right*, *top* to *bottom*. **b** Ten detected supergroups. **c** Q matrix of corresponding 105 keyframes. **d** Sorted Q matrix

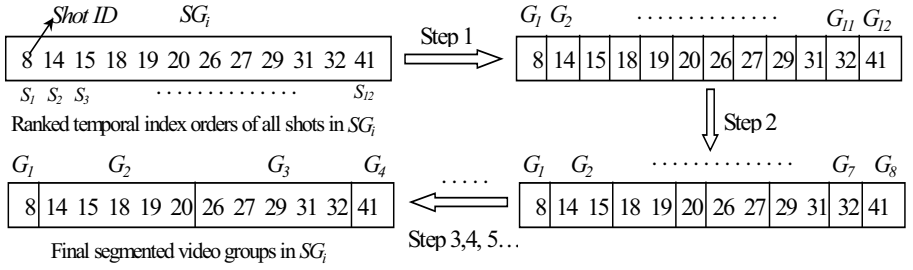


Fig. 8. Video group detection by temporal shot segmentation

Eq. 6:

$$S_i \in SG_t\ , \quad t = \arg_l\{\max(T_l)\ ; \quad l = 1, 2, \ldots, NSG\}, \tag{6}$$

i.e., the supergroup with the maximal number of keyframes of S_i is selected as the target to which S_i belongs.

Figure 7 shows the pictorial results of our supergroup detection strategy applied to a real-world video with 105 keyframes. The sample frames are shown in Fig. 7a in their temporal order, from left to right, top to bottom. Figures 7c and d show the constructed Q matrix with the top ten eigenvectors and the sorted result. Obviously, the sorted diagonal blocks can be used separately for supergroup detection. The representative frames of ten detected super video groups are shown in Fig. 7b.

4.3 Video group detection

In the strategy above, the temporal order information among shots has not been considered. Although visually similar shots have a high probability of belonging to one scene, the shots with a large temporal distance are more likely to belong to different scenarios. Accordingly, given any supergroup (SG_i), those adjacent shots in SG_i with their temporal distance larger than a predefined threshold are segmented into different video groups.

Given two shots S_l andS_h in video V_i, assume their *temporal index order* (shot ID) is denoted by ID_{S_l} and ID_{S_h}, respectively. The *temporal index order distance* (IOD) between S_l and S_h is defined by Eq. 7:

$$IOD(S_l, S_h) = \|ID_{S_l} - ID_{S_h}\| - 1\ . \tag{7}$$

That is, the IOD between S_l and S_h is the number of shots between S_l and S_h in their temporal orders.

Given a supergroup SG_i with NT 1shots and a specific threshold T_{Group}, the strategy below will segment all NT shots into corresponding groups.

1. Rank all shots S_l ($l = 1, \ldots, NT$) of SG_i in increasing order by their *temporal index order* (that is, the shot with the smallest index number (ID) will be assigned to the first place, as shown in Fig. 8).
2. Initially, assume there are NT groups $G_l(l = 1, \ldots, NT)$ in SG_i with each group containing one shot.
3. Given any two neighboring groups G_i (with NT_i shots) and G_j (with NT_j shots) in SG_i, if the IOD between G_i and G_j is smaller than the threshold T_{Group}, G_i and G_j are merged into one group. Accordingly, we define the IOD between G_i and G_j, denoted by $IOD(G_i, G_j)$, using Eq. 8:
$$IOD(G_i, G_j) = \min_{l=1,\ldots,NT_i; S_l \in G_i} \{IOD(S_l, S_h); \\ h = 1, \ldots, NT_j, S_h \in G_j\}. \tag{8}$$
If $IOD(G_i, G_j) \leq T_{Group}$, G_i and G_j are merged into a new video group; otherwise they remain separate.
4. Iteratively execute step 3 until there are no more groups to be merged. Assume that *NG* groups ($G_i, i = 1, \ldots, NG$) remain. For any two remaining groups G_l and G_h, $IOD(G_l, G_h) > T_{Group}$; $G_l, G_h \subset SG_i$. All remaining units are taken as video groups in SG_i.

Figure 8 presents an example to indicate our group segmentation algorithm. Given a supergroup SG_i containing 12 shots, with the ranked temporal index orders of all shots given by 8, 14, 15, 17,..., 41. Initially, each shot is taken as one group, and any two neighboring groups with an IOD less than T_{Group} are merged into one group (we define $T_{Group} = 4$ in our system, and a detailed analysis on selecting T_{Group} is discussed in Sect. 6.1). Iterative execution finally segments all shots in SG_i into four groups.

4.4 Video scene detection

Using the strategy above, all shots can be parsed into groups, with each group containing visually similar and temporally adjacent shots. However, as Definition 7 in Sect. 2 indicates, *video group* is an intermediate entity between *physical shots* and *semantic scenes*. This means that a single video group may be incapable of acting as an independent story unit to convey a video scenario. Actually, a video scene usually consists of temporally interlaced groups or visually similar and neighboring groups. In this section, a group-merging algorithm is proposed for video scene detection.

Before describing our scene-detection algorithm, we will discuss the characterization of video scenes. By considering the definitions in [51], we conclude that three types of scenes are widely used by filmmakers when they capture or edit videos:

- *N-type scene:* These scenes (also known as normal scenes) are characterized by a long-term consistency of chromatic composition, lighting conditions, sound, etc. All shots in the scene are assumed to exhibit some similarity with their visual or audio features. An example of this type of scene is shown in Fig. 9a, where various shots captured from different cameras are organized to report a news story on a flight show in an international aviation exhibition. Since all shots are captured in the same scene, they are somewhat similar in visual features, and all shots in the scene

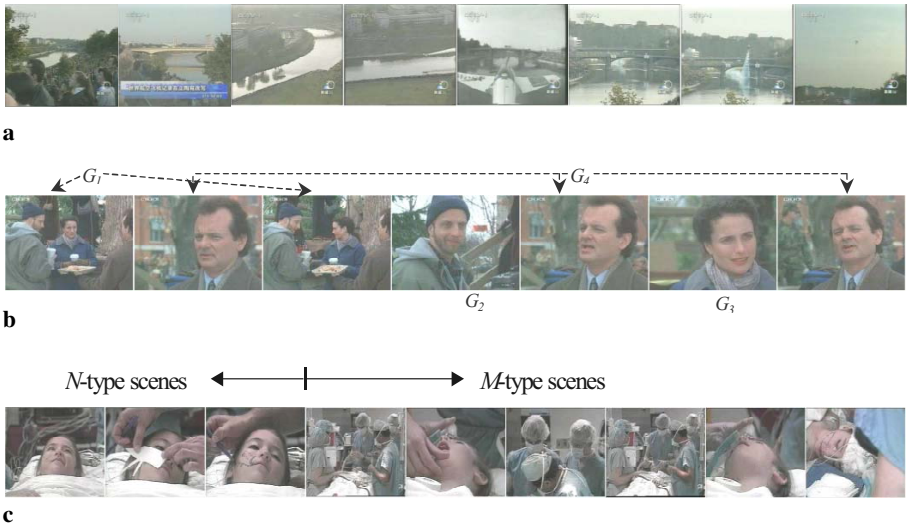


Fig. 9. Examples of various types of scenes. **a** N-type scenes. **b** M-type scenes. **c** Hybrid scenes where both N- and M-type scenes exist in the same story unit

might be merged into one or several visually similar video groups.

- *M-type scenes:* These scenes (also known as montage scenes) are characterized by widely different visuals (different in location, time of creation, and lighting conditions) that create a unity of theme by the manner in which they have been juxtaposed. An example of this type of scene is shown in Fig. 9b, where three actors are involved in a dialog scene. Obviously, the shots in the scene have a large visual variance and have been classified into four distinct groups (G_1, G_2, G_3, and G_4). However, these four groups are temporally interlaced to convey the same scenario information.
- *Hybrid scenes*: The hybrid scenes consist of both N- and M-type scenes, where both visually similar neighboring groups and temporally interlaced groups are utilized to convey a scenario unit. This is shown in Fig. 9c, where the beginning is a typical N-type scene and the end is an M-type scene.

Using the above observations and definitions, we now introduce a temporally and spatially integrated strategy for parsing video groups into semantic scenes.

Given groups G_i and G_j, Fig. 10 presents three types of temporal relationship among them: *interlaced*, *uninterlaced*, and *neighboring*. To parse video semantic scenes, the temporally interlaced video groups and visually similar neighboring groups should be merged into one scene since they have a high probability of conveying the same scenario. Assume $IOD^b_{G_i}$ and $IOD^e_{G_i}$ denote the minimal and maximal temporal index order of the shots in G_i, as shown in Fig. 10a. To classify G_i and G_j into interlaced, uninterlaced, or neighboring categories, the *temporal group distance* (*TGD*) between G_i and G_j ($TGD(G_i, G_j)$ is defined by Eq. 9:

$$TGD(G_i, G_j) = \begin{cases} ID^b_{G_j} - ID^e_{G_i}\ ; & \text{if } ID^b_{G_j} \geq ID^b_{G_i} \\ ID^b_{G_i} - ID^e_{G_j}\ ; & \text{if } ID^b_{G_j} < ID^b_{G_i}. \end{cases} \quad (9)$$

If $TGD(G_i, G_j) > 1$, there is no interlacing between shots in G_i and G_j, and hence G_i and G_j are classified as uninterlaced groups, as shown in Fig. 10a. If $TGD(G_i, G_j) = 1$, we classify G_i and G_j as neighboring groups, as shown in Fig. 10c. If $TGD(G_i, G_j) < 0$, G_i and G_j are classified as interlaced groups, as shown in Fig. 10b. Obviously, since a shot can be assigned to only one group, $TGD(G_i, G_j)$ will never equal 0.

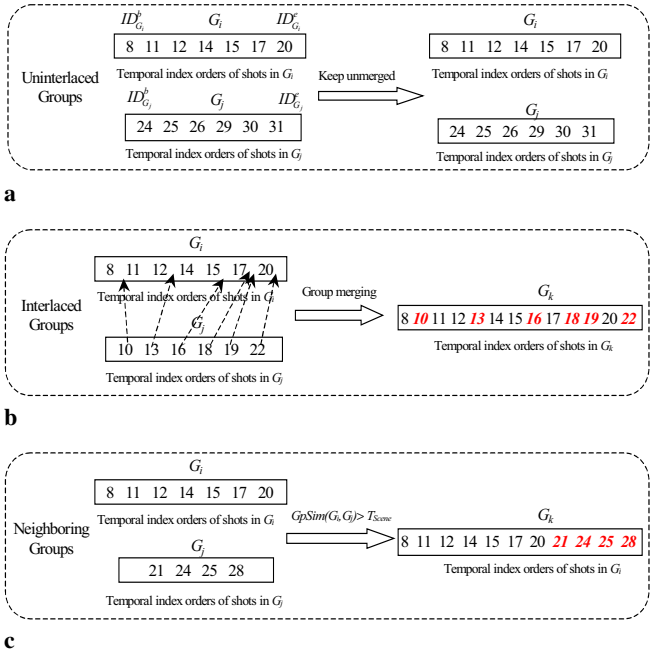


Fig. 10. Group merging for scene detection. **a** Uninterlaced groups. **b** Interlaced groups are merged into one unit. **c** Neighboring groups with similarity larger than the threshold are merged into one unit

After classifying any two video groups into the appropriate categories, our scene detection follows the steps below.

1. Given any two video groups G_i and G_j, if $TGD(G_i, G_j) < 0$, G_i and G_j are merged into a new group by combing all shots in G_i and G_j. As shown in Fig. 10b, the interlaced group G_i and G_j are merged together to form a new group G_k, with numbers in different styles (bold & italic vs. roman) indicating shots from different groups.

2. For $TGD(G_i, G_j) = 1$, if the similarity between G_i and G_j (given by Eq. 11) is larger than a given threshold T_{Scene}, G_i and G_j are merged into a new group. Otherwise, they remain unmerged, as shown in Fig. 10c.
3. Iteratively execute the steps above until no more groups can be merged. All remaining and newly generated groups are taken as video scenes.

To measure the similarity between groups for scene detection, we first consider the similarity between a shot and a group. Based on Eq. 2, given shot S_i and group G_j, the similarity between them is defined by Eq. 10:

$$StGpSim(S_i, G_j) = \max_{K_l \in S_i, K_h \in G_j} \{FmSim(K_l, K_h)\}. \tag{10}$$

This implies that the similarity between S_i and G_j is the similarity between the keyframes in S_i and G_j that have the highest similarity. In general, when we evaluate the similarity between two video groups using the naked eye, we usually take the group with fewer shots as the benchmark and then determine whether there are any shots in the other group similar to shots in the benchmark group. If most shots in the benchmark group are similar enough to the other group, these two groups are treated with a high similarity [30,42]. Given groups G_i and G_j, assume $\hat{G}_{i,j}$ represents the group containing fewer shots and $\tilde{G}_{i,j}$ denotes the other group. Suppose $NT(x)$ denotes the number of shots in group x; then the similarity between G_i and G_j is given by Eq. 11.

$$GpSim(G_i, G_j) = \frac{1}{NT(\hat{G}_{i,j})} \times \sum_{i=1; S_i \in \hat{G}_{i,j}}^{NT(\hat{G}_{i,j})} StGpSim(S_i, \tilde{G}_{i,j}) \tag{11}$$

That is, the similarity between G_i and G_j is the average similarity between shots in the benchmark group and their most similar shots in the other group.

Some typical scene detection results from real-world videos (movies, news, and medical videos) are presented in Fig. 11, where the video shots in the same story units are successfully detected. Actually, in some situations, the scene boundaries might be too vague to be distinguished by even the naked eye; we don't believe any state-of-the-art methods can attain satisfactory results from those videos. However, for the scenes with relatively clear boundaries, our method obtains attractive results (as demonstrated in Sect. 6.3), and these results will guide us in constructing hierarchical video summaries.

4.5 Video-scene clustering

Thus far, a video content structure from video scenes and groups to keyframes has been constructed. Nevertheless, since our final goal is to use the video content structure to construct video summaries, a more compact level beyond the scenes is necessary for creating an overview of the video. This is accomplished by clustering visually similar scenes that appear at different places in the video.

Given a scene SE_i in video V_k, assume there are NT shots contained in SE_i, and assume further that all shots in SE_i have been merged into NSG_i supergroups SG_l ($l = 1, \ldots, NSG_i$). The union of all these supergroups is then denoted by Eq. 12:

$$USG(SE_i) = SG_1 \cup SG_2 \cup \cdots \cup SG_{NSG_i}. \tag{12}$$

Obviously, given any two scenes SE_i and SE_j, if $USG(SE_i) \subseteq USG(SE_j)$, this indicates that all shots in SE_i come from the same supergroups as SE_j. Hence, the visual redundancy between SE_i and SE_j can be eliminated by

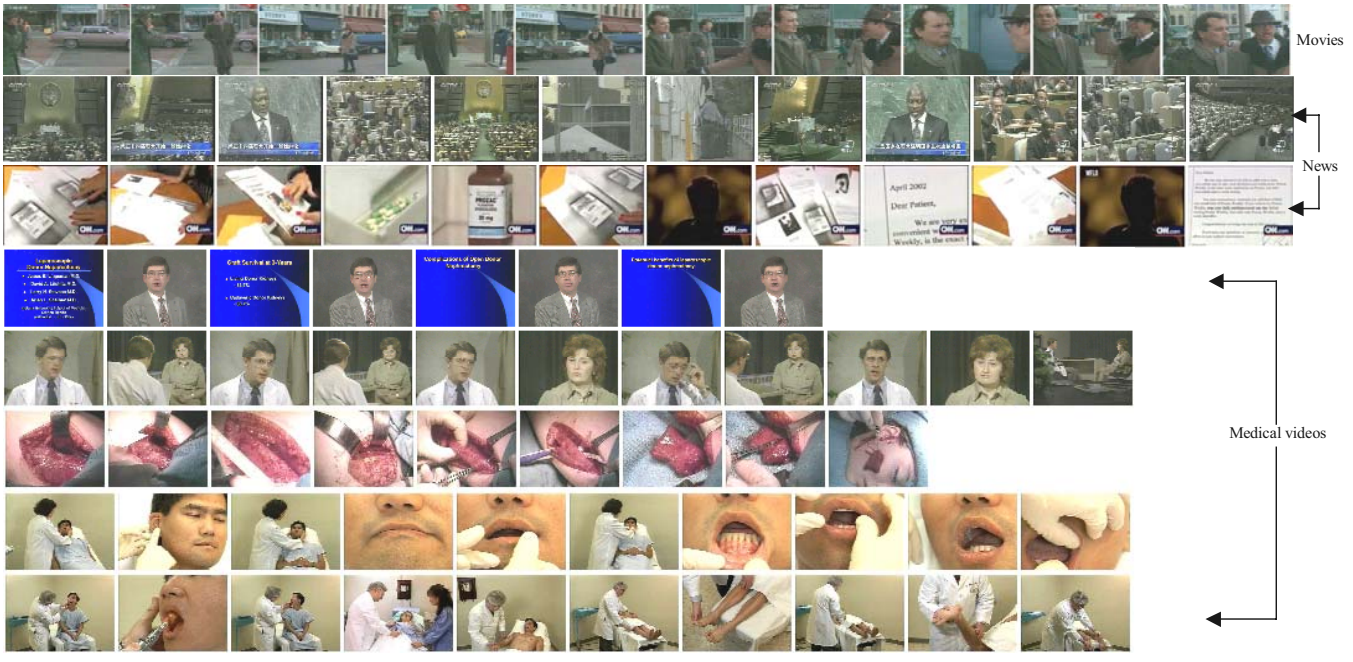


Fig. 11. Some typical examples of the successfully detected video scenes among various types of videos (movies, news, medical video) with each row denoting one scene

removing SE_i and retaining SE_j. Based on this observation, our scene clustering follows the steps below:

1. Given any two scenes SE_i and SE_j, if $USG(SE_i) \subseteq USG(SE_j)$, the scene SE_i is removed by using SE_j to represent its content; otherwise, both SE_i and SE_j are retained.
2. Iteratively execute step 1 until no more scenes can be removed; the remaining scenes are temporarily taken as the clustered video scenes.
3. Usually, the video scenes containing fewer shots are less important for conveying video content than those with more shots. Hence, a predefined threshold $T_{Cluster}$ (we set $T_{Cluster} = 3$ in our system) is applied to remove scenes that contain no more than $T_{Cluster}$ shots. All remaining scenes are taken as the final results.

5 Hierarchical video summarization

In this section, techniques for constructing summaries at different levels are introduced. Instead of using only the spatial or temporal clustering schemes, we use the acquired video content structure to guide us in selecting summary units and trimming the selected representative frames into a hierarchy, with each level of summary conveying the "appropriate" and "necessary" information.

5.1 Constructing summaries at the lowest two levels

As introduced in Sect. 3, the construction for *level 1* and *level 2* summaries is trivial. After video shots and keyframes have been successfully extracted, all keyframes are packed together in their original temporal order as the *level 1* summary. Obviously, this level summary records the most video theme and scenario details; however, the redundancy appears to be significant since no redundancy reduction has been conducted.

After merging visually similar shots into supergroups, the number of shots in each supergroup may reflect its importance in addressing the video content. It is likely that supergroups containing only one shot will convey very limited content information, and the shots in these supergroups usually consist of some special frames that are distinct from others, e.g., special editing shots. Instead of addressing theme (or scenario) information, these special editing shots generally emphasize visual details. Accordingly, after removing supergroups with only one shot, the summary at *level 2* consists of all remaining keyframes. By abandoning those one-member supergroups to construct the second level summary, our method is distinct from highlight-based schemes: the shots abandoned by our scheme might be selected as the important units by those methods.

5.2 Constructing an X-level dynamic summary

The summaries at the two lowest levels are statically constructed. Since summary scalability is required for many applications, the dynamical summarization should also be considered.

Given that the summary at *level 2* has been successfully constructed, suppose there are NSG supergroups SG_i $(i = 1, \ldots, NSG)$ containing NG video groups $G_i\,(i = 1, \ldots, NG)$ and NK keyframes $K_l\,(l = 1, \ldots, NK)$. Our dynamic summarization is conducted by using the hierarchical clustering algorithm (*Minimal Spanning Tree* [50]) to construct a keyframe hierarchy for each group G_i, followed by a procedure to select a certain number of representative frames from the hierarchy to construct the summary (according to the user's specification with respect to the length). Obviously, the dynamic summary at the lowest and the highest levels consists of NK and NG representative frames, respectively. The user can specify any number between NK and NG to generate a dynamic summary.

To construct the keyframe hierarchy for each group G_i, the *Minimal Spanning Tree* algorithm is adopted.

- Given a group G_i, assume there are NK_i keyframes K_l (l=1,...,NK_i) contained in G_i. Use the affinity matrix M in Sect. 4.2 to acquire the similarities between any two keyframes in G_i.
- Merge the two keyframes with the highest similarity to form a new cluster, and use the feature average of these two keyframes to represent its centroid. The cluster is represented by the member keyframe that is the closest to the centroid of the cluster.
- Iteratively execute the above step to merge the two closest points (and treat each acquired cluster as a point). To measure the similarity between any two clusters or between a cluster and a keyframe, we treat each cluster as a video group and apply Eq. 11.
- At the highest level of the cluster, a single keyframe is selected as the representative frame of G_i, as shown in Fig. 12.

After the hierarchy for each group G_i has been determined, we conduct the dynamic summarization by selecting a certain number of representative frames from each group and assembling them to form the user- specified summary. Assume that the user specifies a summary with T keyframes (obviously, $NG \leq T \leq NK$ which is guaranteed by the system); the selection of keyframes from each group is determined by the following steps:

1. Supergroups that contain more keyframes likely convey more content information. If the number of representative keyframes is limited, we should first consider extracting representative frames from the supergroups with more keyframes. Consequently, we rank the supergroups SG_i $(i = 1, \ldots, NSG)$ in descending order according to the number of keyframes they contain.
2. Beginning with the supergroup with the most keyframes, select one representative frame from each supergroup using the *SelectSuperGroupRFrame*() strategy described below. If this procedure has been applied to all SG_i $(i = 1, \ldots, NSG)$ and more representative frames are needed, repeat the same procedure on the ranked supergroups until enough frames have been extracted.
3. For any supergroup SG_l, if there are no more representative frames available for extraction in the current iteration, we remove SG_l from consideration in the next iteration.
4. At any point, if T representative frames have been extracted, the procedure halts.

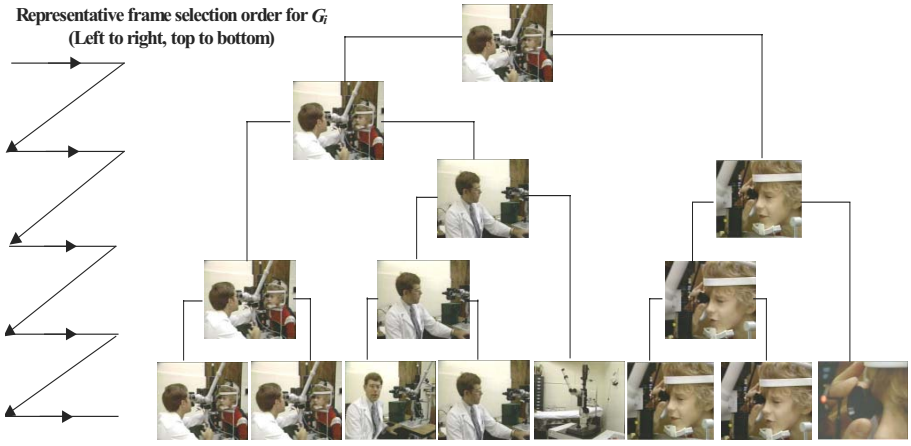


Fig. 12. Hierarchical keyframe clustering for group G_i and the representative frame selection order (in the keyframe hierarchy, the temporal order of the keyframes has been discarded)

Because keyframes in different supergroups are visually distinct, we sequentially select representative frames from each supergroup to guarantee that the selected frames cover the visual variances of the video, with the important supergroups having a relatively higher probability for selection. To select representative frames from each supergroup, the following strategy is applied.

[*SelectSuperGroupRFrame*]

- ***Description*:** Select one representative frame from a supergroup SG_i.
- ***Procedure*:**

1. Denote the current supergroup as SG_i and all groups in SG_i as G_l ($l = 1, \ldots, NG_i$). After parsing all shots in SG_i into groups, we can determine the rank of each group, with the group containing the smallest temporal index order (shot ID) placed furthest to the *left* in the video and the group with the largest shot ID placed furthest to the *right*, as shown in Fig. 8.
2. Beginning with the group occupied at the leftmost position, select one representative frame from each group G_l using the *SelectVideoGroupRFrame*() strategy described below. If the current G_l was processed in the former iteration, execute *SelectVideoGroupRFrame*() on the succeeding group G_{l+1}. If all groups G_l (l=1,...,NG_i) were processed in former iterations, repeat the same procedure on all groups.
3. For any group G_l, if no more representative frames are available for extraction in the current iteration, G_l is removed from further processing. The same operation is executed on the succeeding video group G_{l+1}. If there are no more representative frames that can be extracted from any video group in SG_i, return "False"; otherwise, return "True" as the result of the current procedure.

To select a representative frame from the supergroup (SG_i), we consider the temporal order of video groups in SG_i because this information will help us construct a summary consisting of representative frames from "*all over*" the video. Finally, by considering the keyframe hierarchy within each video group, the representative frame selection for each video group is executed using the procedure below.

[*SelectVideoGroupRFrame*]

- ***Description*:** Select one representative frame from a video groupG_l.
- ***Procedure*:**

1. Denote the current video group as G_l. Assume the keyframe hierarchy of G_l has been constructed as shown in Fig. 12.
2. From left to right and top to bottom of the cluster, select a representative frame (that has not been selected in any previous iteration) from the hierarchy of G_l, as shown in Fig. 12. If all keyframes in G_l were selected as representative frames in former iterations, return "False"; otherwise, return "True" as the result of the current procedure.
3. Generally, the selected representative frame from each cluster of the hierarchy in G_l is also the representative frame of the current cluster. However, if the representative frame of the current cluster was selected in former iterations, a second member that is relatively close to the cluster centroid is taken as the representative frame.

With the strategies above, dynamic video summaries, with length varying from *NG* to *NK* frames, can be determined. It is obvious that, in addition to considering the visual variance and redundancy among all keyframes, video scenario information is also considered. Even at the highest level of the dynamic summary, the overview still consists of representative frames from most scenes.

5.3 Constructing summaries at the three highest levels

In previous summary levels, the video scenario information is well preserved, but such a mechanism likely results in considerable visual redundancy among summaries. At higher summary levels, it is appropriate for us to abandon some scenarios and focus on visual conciseness of the summaries.

To construct a summary beyond the highest level of the dynamic summary (*level X+3*), all video groups in the dynamic summary are taken as candidates. Our observations show that video groups that contain less than three shots and do not interlace with other groups (as defined in Sect. 4.4) likely address less content information than other groups. Accordingly, the summary at *level X+3* consists of representative frames of the groups with at least three shots or interlace with other groups. Using our scene-clustering strategy, visually similar scenes

from the same supergroups have been merged into unique clusters, thus partially reducing the visual redundancy. Therefore, the summary at *level X+4* consists of representative frames of all groups at *level X+3* that belong to clustered scenes.

At the highest level of the summary, the video scenario is not a concern; a visually compact overview is our target. Accordingly, we discard temporal information among keyframes and consider only visual similarities. Because keyframes in one supergroup are visually similar, our solution in constructing the highest level summary is to merge representative frames at *level X+4* that belong to the same supergroup into one cluster with the following steps:

1. Collect all representative frames of the summary at *level X+4*. Merge those from the same supergroup into one cluster and use the feature average of all members (representative frames) as its centroid. The member frame that is closest to the centroid is selected as the representative frame of the cluster.
2. Temporally assemble all selected representative frames to construct the summary at the highest level (*level X+5*).

By integrating the dynamic summary and five levels of static summary, an $X+5$ level hierarchical summary is constructed; the higher the level, the more concise the summary. From this point of view, the proposed scheme is successful in integrating scalability, adaptability, scenarios, and visual redundancy of the video for efficient summarization.

6 Experimental results and analysis

To evaluate the performance of the above proposed strategies with respect to existing approaches, and to determine the threshold for various critical parameters used in our approach, we now present a series of experimental results and comparisons, including the determination of the threshold T_{Group}, results for video group and scene detection, and hierarchical summarization. About 10 h of medical videos, 4 h of news programs, and two movies are used as our test bed (all video data are MPEG- I encoded, with the digitization rate equal to 30 frames/s). All videos are first parsed with the shot-segmentation algorithm to detect the gradual and break shot changes. This is followed by keyframe selection, and the extracted keyframes from each video are utilized for processing.

6.1 Determining the threshold T_{Group}

As we indicated in Sect. 4.3, to determine video groups in each supergroup, a threshold T_{Group} is predefined to segment visually similar and temporally related shots into groups. To determine a relatively optimal value for T_{Group}, the statistical information among visually similar shots in each scene has been determined manually: after supergroups have been detected, we manually separate the shots in each supergroup into groups, with each group belonging to one scene. The temporal index order distance (IOD) (as defined in Eq. 7) between neighboring shots that belong to the same group are counted. The statistical results of the IODs for about 200 video groups are reported in Fig. 13, where the x-axis denotes the IOD

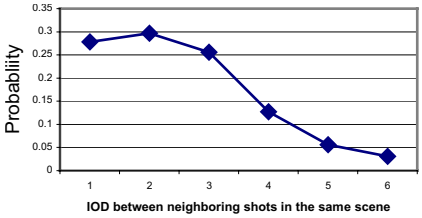


Fig. 13. Statistical distribution of the IOD between neighboring shots in the same video group

between neighboring shots and the y-axis indicates its percentage. It was found that for all visually similar neighboring shots that belong to the same semantic unit, about 29% of them have a two-shot IOD (which is also the maximum). Moreover, almost 90% of the visually similar neighboring shots in the same semantic scene have an IOD of no more than four shots. This also fits with the memory model adopted in the literature [51]. Increasing the value of T_{Group} will definitely decrease the precision of video group detection (as defined in Eq. 13); however, it may also enhance the *group compression rate* (as defined in Eq. 14). A relatively good balance might be acquired by setting $T_{Group} = 4$.

6.2 Group detection results

Group detection is evaluated by selecting various thresholds for T_{Group} (from 1 to 6), and the detection results are manually verified with the original video source. The results are shown in Figs. 14 and 15, with the x-axis denoting the value of T_{Group} and the y- axis denoting the group detection precision (from Eq. 13) and GCRF (from Eq. 14), respectively. To judge the quality of the detection results, the following rule is applied: the group is judged to be correctly detected if and only if all shots in the current group belong to the same semantic scenario. Thus the group detection precision (GP) in Eq. 13 is used for performance evaluation.

$$GP = Correctly\ detected\ groups\ /\ All\ detected\ groups \quad (13)$$

Clearly, if we treat each shot as a group, group detection precision would be 100%. Therefore, another group *compression rate factor* (GCRF) is defined by Eq. 14.

$$GCRF = Detected\ group\ number\ /\ Total\ shot\ number \quad (14)$$

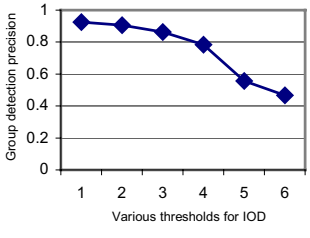


Fig. 14. Video group detection performance with different T_{Group}

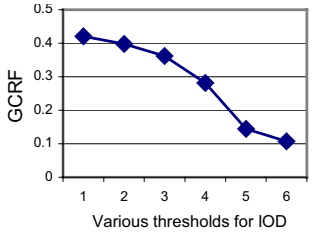


Fig. 15. Group compression rate factor with different T_{Group}

Experimental results from Fig. 14 to Fig. 15 show that as the threshold T_{Group} increases, group detection precision deteriorates, but GCRF improves. With $T_{Group} = 4$, about 77% shots are assigned to the right groups, and the compression rate of this level is about 28%. That is, each detected video group consists of 3.5 shots on average. Our experimental results show that GCRF may vary significantly among different types of videos (movies, news, or medical videos). Even in different scenes of the same video, this number may have a large variance. To segment semantically related groups for video story unit detection, we may prefer a higher precision rather than a better compression rate because the group detection results will directly affect the quality of the story unit detection. Falsely detected video groups will propagate to the scene detection and summarization and bring negative impacts to our objective. Hence, we set $T_{Group} = 4$ in our system.

6.3 Scene detection results

Scene detection comparisons are made with the methods in [39,40]. To judge the quality of the detected results, a rule that is similar to that for group detection evaluation is applied: a scene is judged to be correctly detected if and only if all shots in the current scene belong to the same semantic scenario. Accordingly, the scene detection precision (SP) is defined by Eq. 15.

$$SP = \textit{Correctly detected scenes / All detected scenes} \quad (15)$$

Clearly, by treating each shot as one scene, we have a 100% detection precision. Hence, another scene *compression rate factor* (SCRF) is defined by Eq. 16

$$SCRF = \textit{\# of detected scene / \# of shot in the video} \quad (16)$$

The experimental results and comparisons are given in Figs. 16 and 17, with our strategy denoted by "A", and other methods in [39,40] denoted by "B" and "C", respectively. The results demonstrate that, among all types of video sources, about 66% of video scenes can been correctly detected with our approach, and each scene consists of 12 shots on average. Obviously, since our strategy emphasizes that only those temporally and visually related groups are merged to form semantic scenes, more restrictions are imposed for scene detection. Thus, some visually distinct shots may be treated as single scenes, such as the anchorperson shots from news videos. Using the strategy in [40], they might be absorbed into adjacent scenes. Hence, the SCRF of our strategy is relatively low. Fortunately, our hierarchical strategy inherently addresses this problem by supplying multilevel summaries. Instead of providing users with highly compressed scenes with a relatively high error rate, we may prefer a high precision rate.

6.4 Hierarchical video summarization results

6.5 Subjective evaluation

As described previously, a multilevel summary with $X+5$ layers is produced for each video, with five layers of static summary and X levels of dynamic summary. To assess the quality

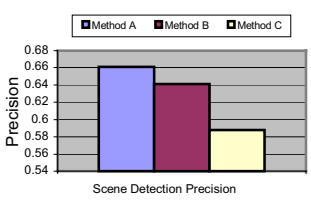


Fig. 16. Scene detection precision

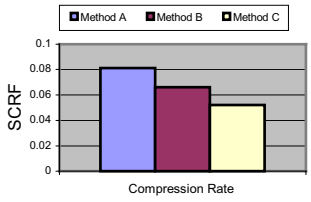


Fig. 17. Scene compression rate

of the summary at each layer in abstracting video content, three questions are posed to the members of our evaluation group: (1) How well do you think the summary addresses the main topic of the video? (2) How well do you think the summary covers the scenarios of the video? (3) Is the summary concise? For each question, a score from 0 to 5 (5 indicating the best) is specified by five student users after viewing the summary at each level. Before the evaluation, the users are asked to browse the entire video to get an overview of the video content. An average score for each level is computed from the students' scores. To evaluate the dynamic summaries, we ask the user to view only two levels, where the lengths of these two levels are *NK/2* and *NG* keyframes (using *NK* and *NG* as specified in Sect. 5.2), respectively. Hence, for each video, a summary with $5 + 2 = 7$ levels is used for evaluation.

To compare our scheme with other methods, we implemented the Hierarchical Video Summarization (*HVS*) scheme [13] and the *SVD*-based strategy [11]. In the figures, we depict our method as *NEW* and the other two methods as *HVS* [13] and *SVD* [11], respectively. The experimental comparisons are shown in Figs. 18 to 20, with each figure showing the results of one question evaluated by selected users. The x-axis denotes the summary levels and the y-axis indicates the corresponding score at that level. Because SVD is a static summary approach, it is depicted as a single line in each figure. Moreover, to make the experimental results more comparable (and thus more convincing), we use HVS to construct a summary with approximately the same number of keyframes for each summary level determined by our method.

The results shown in Figs. 18–20 demonstrate that since SVD discards the temporal information among the keyframes for summarization, it exhibits a relatively poor ability to cover video topic and scenario information. However, the summaries created from SVD are remarkably concise. As shown in Fig. 20, it obtains a score of 3.4 in conciseness, which is much higher than most other summaries. For summaries at lower levels, we emphasize preserving video scenarios; this will inevitably incur redundancy in the summary. Consequently, the conciseness of most summaries from our method is worse than SVD.

Compared with HVS, our approach shows comparable (or worse) performance with summaries at lower levels (levels 1 to 3). Since HVS performs the clustering among neighboring keyframes, it may produce relatively better results when neighboring keyframes contain significant redundancy. This is commonly found in summaries at lower levels. However, this method cannot remove global visual redundancy, and its performance deteriorates rapidly at higher summary levels. This indicates that the performance of HVS may be especially

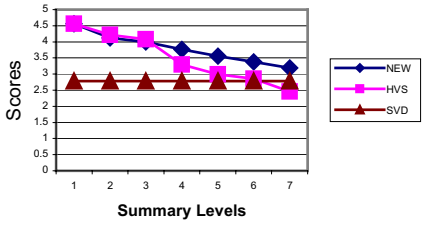


Fig. 18. Experimental comparisons with Q.1

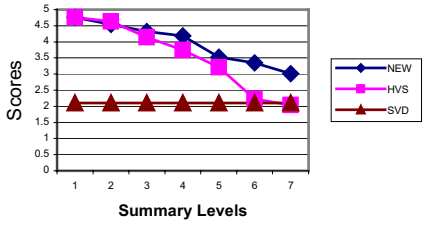


Fig. 19. Experimental comparisons with Q.2

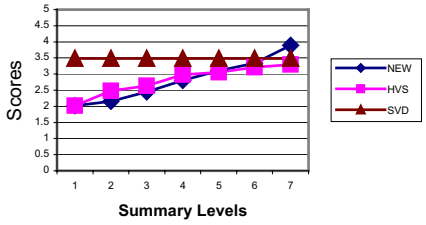


Fig. 20. Experimental comparisons with Q.3

unpleasant in situations where the length of a summary is very limited.

In conclusion, our method achieves a relatively good balance in addressing the three questions given above. By utilizing a hierarchical scheme, both visual conciseness and scenarios can be emphasized at various levels and with different granularities. Obviously, it would be very difficult, if not impossible, to produce a summary that optimizes these three questions at the same time. With very limited representative frames, the system cannot provide a summary that consists of visually distinct frames from which the video scenario and topics can also be well addressed. In a narrow sense, we may actually need a strategy to balance what the user wants and what we can supply. We believe the hierarchical summarization provides a practical solution in this regard.

6.5.1 Objective evaluation

All evaluations above are determined by selected users, which implies that user subjectivity has significant impact on the results. We subsequently execute relatively objective comparisons by evaluating the ability of a summary to address the major visual content of the video. We first construct the *major visual content table* (*MVCT*) for each video by displaying to the user the representative frames of all supergroups and removing those supergroups that are not important in addressing the video content, as determined by the majority vote of the users. The representative frames of the remaining supergroups are packed as the MVCT of each video, and we denote the remaining representative frames as major visual content frames MVC_i $(i = 1, \ldots, T)$, where T indicates the number of selected MVC frames.

Let VS_L denote the video summary (*VS*) at level L. We use the *coverage* to specify the ability of the summary to address the visual content in MVCT and denote this by $Cov(VS_L)$ as defined by Eq. 17:

$$Cov(VS_L) = \frac{Card\{i : \exists j, K_j \in VS_L, K_j \propto MVC_i;\ i = 1, \ldots, T\}}{T}, \tag{17}$$

where $K_j \propto MVC_i$ denotes that frame K_j and MVC_i belong to the same supergroup. Equation 17 indicates that the *coverage* of the summary is the ratio between the number of MVC frames that have at least one similar frame in VS_L (both frames belong to the same supergroup) and the total number of MVC frames in MVCT. The larger the value of $COV(VS_L)$, the better the summary (VS_L) addresses the visual information of the video. Note, however, that if all keyframes are selected as the video summary, we have a 100% coverage. Thus the rate between the number of representative frames at each layer and the number of all keyframes is utilized to determine the *summary compression rate* (*SMCR*) of each summary level, as defined by Eq. 18.

$$SMCR(VS_L) = \#\ of\ rep.\ frames\ in\ VS_L\ /\ \#\ of\ keyframes \tag{18}$$

The results in Fig. 21 demonstrate that the higher the summary level, the worse the coverage of the summary, because with more and more keyframes abandoned, some important visual information is inevitably lost. With HVS, the coverage at each level deteriorates rapidly, since this method does not optimize global visual redundancy. Instead, it executes clustering among neighboring keyframes, which will inevitably eliminate many visually important frames. With our approach, about 65% of the major visual content is preserved at the highest summary level, which is a little worse than SVD.

At each summary level, both HVS and our approach have the same number of keyframes; therefore, the SMCR for both strategies is the same at each level, and we denote it by *NEW/HVS* in Fig. 22. From the results in Fig. 22, we find that at the highest level, a 10% compression rate is produced. That is, we can compress 90% of the keyframes to supply a visually compact summary. This indicates that with a hierarchical approach, we can provide a scalable summary that addresses most of the visual content and scenario information and compresses a large number of redundant frames to supply a compact video overview.

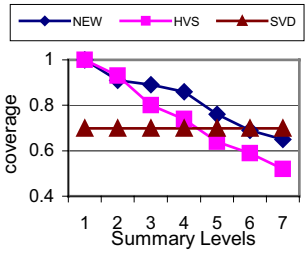


Fig. 21. Coverage of the summary at each level

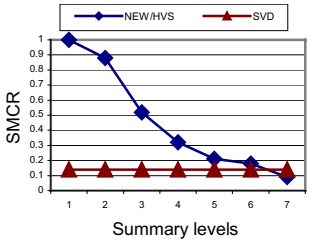


Fig. 22. Summary compression rate at each level

7 Conclusion

In this paper, we have introduced a technique for hierarchical video summarization. Unlike many other strategies that select low-level feature-related video units (highlights) to construct video summaries, we explore video content structure for summarization, where the selected summary units emphasize video scenarios but not visual details. The acquired video content hierarchy is used to construct an $X + 5$ level summary, with various granularities to address video content. The higher the summary level, the more visually compact the summary. The lower the level, the more details about video scenario information. Experimental results have demonstrated that the proposed approach provides a good solution for addressing the scalability, adaptability, and scenario/visual redundancy of the video summary.

The novel features that distinguish our approach from existing research are the following. First, most other strategies either discard the temporal information among keyframes by using only spatial clustering schemes or merely consider the visual similarity between neighboring keyframes to construct local optimal summaries (i.e., they fail to consider the global visual redundancy). Both temporal and visual information is seamlessly integrated in our approach. Second, unlike other strategies where static and dynamic video summarizations are mutually exclusive, we integrate both in a unified model, so the users have scalability and references in acquiring an *appropriate* summary. Third, instead of using highlight selection strategies to select important video units, we emphasize a balance between video scenarios and visual redundancy for summarization. Obviously, important unit selection strategies are domain dependent and subjective, but we believe the proposed approach is appropriate for summarizing most videos (with content structure) in our daily life.

References

1. Zhang H, Kantankanhalli A, Smoliar S (1993) Automatic partitioning of full-motion video. Multimedia Syst 1(1):2
2. Zhang H, Low C, Smoliar SW, Zhong D (1995) Video parsing, retrieval and browsing: an integrated and content-based solution. In: Proc. ACM Multimedia
3. Yeung M, Yeo B (1997) Video visualization for compact presentation and fast browsing of pictorial content. IEEE Trans CSVT 7:771–785
4. Pfeiffer S, Lienhart R, Fischer S, Effelsberg W (1996) Abstracting digital movies automatically. VCIP3 7(4):345–353
5. Li Y, Zhang T, Tretter D () An overview of video abstract techniques. HP Technical Report 4
6. Mills M (1992) A magnifier tool for video data. In: Proc. ACM Human Computer Interface, pp 93–98
7. Uchihashi S, Foote J, Girgensohn A, Boreczky J (1999) Video Manga: Generating semantically meaningful video summaries. In: Proc. 7th ACM Multimedia conference, Orlando, FL, pp 383–392
8. Doulamis N, Doulamis A, Avrithis Y, Ntalianis K, Kollias S (2000) Efficient Summarization of Stereoscopic Video Sequences. IEEE Trans CSVT 10(4) 5
9. Stefanidis A, Partsinevelos P, Agouris P, Doucette P (2000) Summarizing video datasets in the spatiotemporal domain. In: Proc. 11th international workshop on dataset and expert systems applications, pp 906–912
10. Kim C, Hwang J (2000) An integrated scheme for object-based video abstraction. In: Proc. 8th ACM Multimedia conference, Los Angeles, pp 303–311
11. Gong Y, Liu X (2000) Generating optimal video summaries. In: Proc. ICME, New York
12. DeMenthon D, Kobla V, Doermann D (1998) Video summarization by curve simplification. In: Proc. of 6th ACM Multimedia conference, Bristol, UK, pp 13–16
13. Ratakonda K, Sezan M, Crinon R (1999) Hierarchical video summarization. In: Proc. IS&T/SPIE conference on visual communications and image processing, San Jose, 3653:1531–1541
14. Lienhart R (1999) Abstracting home video automatically. In: Proc. 7th ACM Multimedia conference, Orlando, FL
15. Lienhart R, Pfeiffer S, Effelsberg W (1997) Video abstracting. Commun ACM 40(12):54–62
16. He L, Sanocki W, Gupta A, Grudin J (1999) Auto-summarization of audio-video presentations. In: Proc. 7th ACM Multimedia conference, Orlando, FL, 30 October–5 November 1999, pp 489–498
17. Fan J, Zhu X, Wu L (2001) Automatic model-based semantic object extraction algorithm. IEEE Trans Circuits Sys Video Technol 11(10):1073–1084
18. Iran M, Anandan P (1998) Video indexing based on mosaic representation. Proc IEEE 86(5) 6
19. Taniguchi Y, Akutsu A, Tonomura Y (1997) PanoramaExcerpts: Extracting and packing panoramas for video browsing. In: Proc. ACM Multimedia conference, Seattle, pp 427–436
20. Ponceleon D, Dieberger A (2001) Hierarchical brushing in a collection of video data. In: Proc. 34th Hawaii international conference on system sciences
21. Christel M, Hauptmann A, Warmack A, Crosby S (1999) Adjustable filmstrips and skims as abstractions for a digital video library. In: Proc. IEEE conference on advances in digital libraries, Baltimore, MD, 19–21 May 1999
22. Christel M (1999) Visual digest for news video libraries. In: Proc. 6th ACM Multimedia conference, Orlando, FL
23. Smith M, Kanade T (1995) Video skimming for quick browsing based on audio and image characterization. Technical Report, CMU-CS-95- 186, School of Computer Science, Carnegie Mellon University, Pittsburgh
24. Nam J, Tewfik A (1999) Dynamic video summarization and visualization. In: Proc. 6th ACM Multimedia conference, October 1999, Orlando, FL
25. Ebadollahi S, Chang S, Wu H, Takoma S (2001) Echocardiogram video summarization. Proc SPIE MI7, San Diego
26. Zhou W, Vellaikal A, Kuo CJ (2001) Rule-based video classification system for basketball video indexing. In: Proc. 9th ACM Multimedia conference workshop, Los Angeles
27. Haering N, Qian R, Sezan M (1999) Detecting hunts in wildlife videos. In: Proc. IEEE international conference on multimedia computing and systems, Florence, Italy, vol I
28. Zhu X, Wu L, Xue X, Lu X, Fan J (2001) Automatic scene detection in news programs by integrating visual feature and rules. In: Proc. 2nd IEEE Pacific-Rim conference on multimedia, Beijing, 24–26 October 2001. Lecture notes in computer science, vol 2195. Springer, Berlin Heidelberg New York, pp 837–842
29. Smoliar S, Zhang H (1994) Content based video indexing and retrieval. IEEE Multimedia 1(2):62–72
30. Zhu X, Fan J, Elmagarmid A, Aref W (2002) Hierarchical video summarization for medical data. In: Proc. SPIE: Storage and Retrieval for Media Databases, vol 4676, San Jose
31. Toklu C, Liou A, Das M (2000) Videoabstract: a hybrid approach to generate semantically meaningful video summaries. In: Proc. ICME, New York

32. Doulamis A, Doulamis N, Kollias S (2000) A fuzzy video content representation for video summarization and content-based retrieval. Signal Process 80(6): 8
33. Zhong D, Zhang H, Chang S (1997) Clustering methods for video browsing and annotation. Technical report, Columbia University
34. Vasconcelos N, Lippman A (1998) A spatiotemporal motion model for video summarization. In: Proc. IEEE conference on computer vision and pattern recognition (CVPR), Santa Barbara, CA, June 1998
35. Fan J, Yu J, Fujita G, Onoye T, Wu L, Shirakawa I (2001) Spatiotemporal segmentation for compact video representation. Signal Process Image Commun 16:553–566
36. Kender J, Yeo B (1998) Video scene segmentation via continuous video coherence. In: Proc. CVPR, Santa Barbara, CA
37. Yeo B, Liu B (1995) Rapid scene analysis on compressed video. IEEE Trans CSVT 5(6):533–544
38. Fan J, Aref W, Elmagarmid A, Hacid M, Marzouk M, Zhu X (2001) MultiView: multilevel video content representation and retrieval. J Electron Imag 10(4):895–908
39. Rui Y, Huang T, Mehrotra S (1999) Constructing table-of-content for video. ACM Multimedia Syst J on Video 7(5):359–368
40. Lin T, Zhang H (2000) Automatic video scene extraction by shot grouping. In: Proc. ICPR, Barcelona
41. Yeung M, Yeo B (1996) Time-constrained clustering for segmentation of video into story units. In: Proc. ICPR, Vienna, Austria
42. Zhu X, Aref WG, Fan J, Catlin A, Elmagarmid A (2003) Medical video mining for efficient database indexing, management and access. In: Proc. IEEE ICDE, pp 569–580, India
43. Hanjalic A, Zhang H (1999) An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. IEEE Trans CSVT 9(8): 9
44. Girgensohn A, Boreczky J (1999) Time-constrained keyframe selection technique. In: Proc. IEEE conference on multimedia computing and systems, Florence, Italy, pp 756–761
45. Zhuang Y, Rui Y, Huang T, Mehrotra S (1998) Adaptive key frame extraction using unsupervised clustering. In: Proc. IEEE ICIP, Chicago
46. Manor L, Irani M (2002) Event-based analysis of video. In: Proc. CVPR, Kauai, HI, pp II-123-II-130
47. Zhang H, Wu J, Zhong D, Smoliar S (1997) An integrated system for content-based video retrieval and browsing. Pattern Recog 30(4):643–658
48. Weiss Y (1999) Segmentation using eigenvectors: a unifying view. In: Proc. IEEE ICCV, Corfu, Greece, pp 975–982
49. Scott G, Longuet-Higgins H (1990) Feature grouping by relocalisation of eigenvectors of the proximity matrix. In: Proc. British Machine Vision conference, Oxford, UK
50. Rasmussen E (1992) Clustering algorithms. In: Frakes W, Bazea-Yates R (eds) Information retrieval: data structure and algorithm. Prentice- Hall, Upper Saddle River, NJ, pp 419–442
51. Sundaram H, Chang S (2000) Determining computable scenes in films and their structures using audio-visual memory models. In: Proc. ACM Multimedia conference, Los Angeles
52. Costeira J, Kanade T (1994) A multi-body factorization method for motion analysis. Technical Report, CMU-CS-TR-94-220, Department of Computer Science, Carnegie Mellon University, Pittsburgh
53. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905