Constructing Table-of-Content for Videos *

Yong Rui, Thomas S. Huang, and Sharad Mehrotra

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
E-mail:{yrui, huang}@ifp.uiuc.edu, sharad@cs.uiuc.edu

Abstract

A fundamental task in video analysis is to extract structures from the video to facilitate user's access (browsing and retrieval). Motivated by the important role that Table-of-Content (ToC) plays in a book, in this paper we introduce the concept of ToC in the video domain. Some existing approaches implicitly use the ToC, but are mainly limited to low-level entities (e.g. shots and key frames). The drawbacks are that low-level structures (1) contain too many entries to be efficiently presented to the user; and (2) do not capture the underlying semantic structure of the video based on which the user may wish to browse/retrieve.

To address these limitations, in this paper we present an effective semantic-level ToC construction technique based on intelligent unsupervised clustering. It has the characteristics of better modeling the time locality and scene structure. Experiments based on real-world movie videos validate the effectiveness of the proposed approach. Examples are given to demonstrate the usage of the scene based ToC in facilitating user's access to the video.

Key words: video accessing, scene level ToC construction

1 Introduction

Recent years have seen a rapid increase of the usage of multimedia information. Of all the media types (text, image, graphic, audio and video), video is the most challenging one, as it combines all other media information into a single data stream. Owing to the decreasing cost of storage devices, higher transmission rates, and improved compression techniques, digital video is becoming available at an ever increasing rate.

However, because of its length and unstructured format, efficient access to video is not an easy task. From the perspective of browsing and retrieval, video is analogous to a book. Access to a book is greatly

*This work was supported in part by ARL Cooperative Agreement No. DAAL01-96-2-0003, and in part by a CSE Fellowship, UIUC.

facilitated by a well designed table of content (ToC) which captures the semantic structure of the book. For current existing video, a lack of such a ToC makes the task of browsing and retrieval inefficient, where a user searching for a particular object of interest has to use the time-consuming "fast forward" and "rewind". Efficient techniques need to be developed to construct video ToC to facilitate user's access.

In the real world, there exist various types of videos, including movies, newscasts, sitcoms, commercials, sports, and documentary videos, etc. Some of them have "story lines", such as movies, while other do not (e.g. sports). In the rest of the paper, we will concentrate on videos having story lines.

Before we explore video ToC construction techniques, it is worth while to first define the terminologies used in video analysis.

- *Video shot* is an unbroken sequence of frames recorded from a single camera. It is the building block of a video. It is a physical entity and is delimited by shot boundaries.
- *Key frame* is the frame which can represent the salient content of a shot. Depending on the content complexity of a shot, one or more key frames can be extracted.
- *Video scene* is defined as a collection of semantically related and temporally adjacent shots, depicting and conveying a high-level concept or story. While *shot* is the building block of a video, it is *scene* that conveys the semantic meaning of the video to the viewers [1]. Scene boundary detection is far more difficult compared with shot boundary detection and is the major focus of this paper.
- *Video group* is an intermediate entity between the physical shots and semantic scenes and serves as the bridge between the two. Examples of groups are temporally adjacent shots [21] or visually similar shots [26].

[1]Some of the early literatures in video parsing misused the phrase *scene change detection* for *shot boundary detection*. To avoid any later confusion, we will use *shot boundary detection* for the detection of physical shot boundaries while using *scene boundary detection* for the detection of semantic scene boundaries.

In summary, videos can be represented by using a hierarchy consisting of five levels (video, scene, group, shot, and key frame), from top to bottom increasing in granularity (see Figure 1).

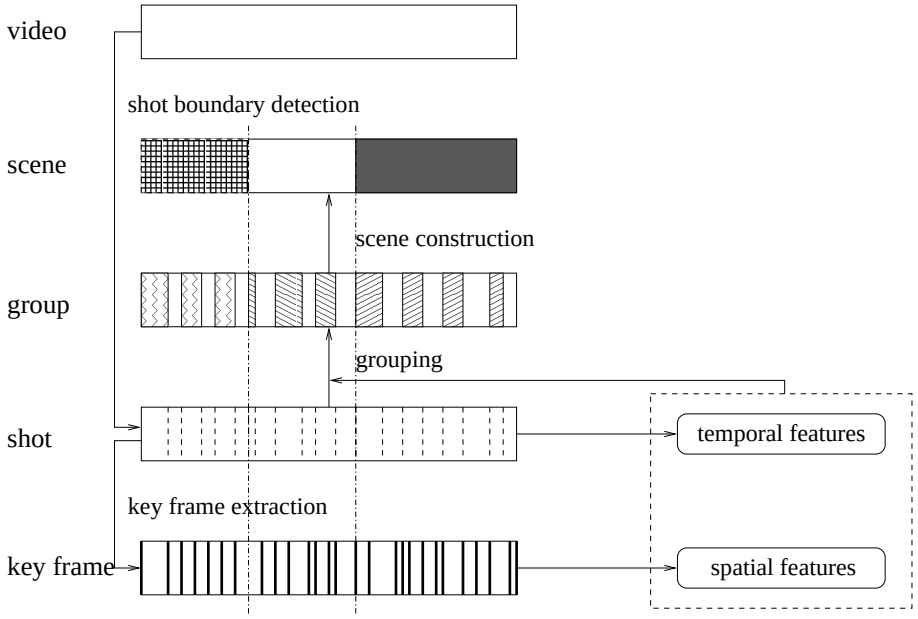

Figure 1: A hierarchical video representation

Over the past few years, researchers have implicitly used video ToC to facilitate user's access to video content, but mainly limited to the *shot*, *key frame*, and *group* levels. Since the video ToC at these levels is not closely related to the semantics of the video and normally has large number of entries, further investigations are needed.

Take the shot and key frame based video ToC for example. It is not uncommon that a modern movie contains a few thousand shots and key frames. This is evidenced in [17] – there are 300 shots in a 15-minute video segment of the movie "Terminator 2 - the Judgment Day" and the movie lasts 139 minutes. Because of the large number of key frames, a simple 1D sequential presentation of key frames for the underlying video is almost meaningless. More importantly, people watch the video by its semantic scenes not the physical shots or key frames. While *shot* is the building block of a video, it is *scene* that conveys the semantic meaning of the video to the viewers. The discontinuity of shots is overwhelmed by the continuity of a scene [4]. The video ToC construction at the scene level is thus of fundamental importance to video browsing and retrieval.

This paper presents a novel framework for scene level video ToC construction. It utilizes an intelligent unsupervised clustering technique. It better models the "temporal locality" without having the "window

effect", and it constructs more accurate scene structures by taking comprehensive information into account. The rest of the paper is organized as follows. In the next section, related work in video ToC construction at various levels is reviewed and evaluated. In section 3, the proposed video ToC construction technique is presented in detail. To put the proposed algorithm into practice, in section 4, techniques based on Gaussian normalization are utilized to determine the algorithm's parameters. In section 5, the effectiveness of the proposed approach is validated by experiments over real-world movie video clips. Concluding remarks are given in section 6.

2 Related Work

Work on extracting video ToC has been done at various levels (*key frame*, *shot*, *group*, and *scene*). Below, we briefly review and evaluate some of the common approaches proposed.

2.1 Shot and key frame based video ToC

In this approach, the raw video stream is first segmented into a sequence of shots by using automatic shot boundary detection techniques. Key frames are then extracted from the segmented shots. The video ToC is constructed as a sequence of the key frames. A user can access the video by browsing through the sequence of key frames. The supporting techniques of this approach, automatic shot boundary detection and key frame extraction, are summarized below:

- *Shot boundary detection*: In general, automatic shot boundary detection techniques can be classified into five categories, i.e. *pixel based*, *statistics based*, *transform based*, *feature based*, and *histogram based*. Pixel based approaches use the pixel-wise intensity difference as the indicator for shot boundaries [23, 9]. One of its drawbacks is its sensitivity to noise. To overcome this problem, Kasturi and Jain propose to use intensity statistics (mean and standard deviation) as the shot boundary detection measure [10]. Exploring how to achieve faster speed, Arman, Hsu and Chiu propose to use the DCT coefficients in the compressed domain as the boundary measure [3]. Other transformed based shot boundary detection approaches make use of the motion vectors, which are already embedded in the MPEG stream [11, 16]. Zabih et al. address the problem from another angle. The edge features are first extracted from each frame. Shot boundaries are then detected by comparing the edge difference [19]. So far, the histogram based approach is the most popular approach. Several researchers claim

that it achieves good trade-off between accuracy and speed [23]. Representatives of this approach are [12, 14, 23, 25, 24]. Two comprehensive comparisons of various shot boundary detection techniques are in [5, 6].

- *Key frame extraction*: After the shot boundaries are detected, corresponding key frames can then be extracted. Simple approaches may just extract the first and last frames of each shot as the key frames [24]. More sophisticated key frame extraction techniques are based on visual content complexity indicator[27], shot activity indicator [8], and shot motion indicator [15].

After the shot boundaries are detected and key frames extracted, the sequence of key frames, together with their frame id's, are used as the video ToC. This approach works well when the number of key frames is not large. However, this approach does not scale to long video clips. In those cases, a simple sequential display of key frames is almost meaningless, as discussed in Section 1.

2.2 Group based video ToC

To obtain a video ToC at a higher level in the video representation hierarchy (Figure 1), related shots are merged into groups, based on which a browsing tree can be constructed [21, 26]. In [21], Zhang et al. divide the whole video stream into multiple video segments, each of which contains equal number of consecutive shots. Each segment is further divided into sub-segments; thus constructing a hierarchy of video content which is used to assist browsing. In this approach, time is the only factor considered and no visual content is used in constructing the browsing hierarchy. In contrast, in [26], Zhong et al. proposed a cluster based video hierarchy, in which the shots are clustered based on their visual content. Although this approach takes into account the visual content, the time factor is lost. One of the common drawbacks of the above approaches is that the video structuring is not at a high enough semantic level. Although these group based video ToCs provide better solutions than the shot and key frame based video ToCs, they still convey only little underlying *semantic* concepts and stories of the video.

2.3 Scene based video ToC

To provide the user with better access to the video, construction of video ToC at a semantic level is needed. Existing approaches to scene level video ToC construction can be classified into two categories, *model-based* and *general purpose*. In model-based approach, an *a priori* model of a particular application or domain is

first constructed. Such a model specifies the scene boundary characteristics, based on which the unstructured video stream can be abstracted into a structured representation. The theoretical framework of this approach is proposed by Swangberg, Shu and Jain in [14], and it has been successfully realized in many interesting applications, including news video parsing [20] and TV soccer program parsing [7]. Since this approach is based on specific application models, it normally achieves high accuracy. One of the drawbacks of this approach, however, is that for each application a model needs to be first constructed before the parsing process can proceed. The modeling process is time consuming and requires good domain knowledge and experience.

Another approach to scene based video ToC construction does not require such an explicit domain model. Three of the pioneering works of this approach are from IRIT, France [1], Princeton University and IBM [16, 18, 17, 4] and Toshiba Corp. [2]. In [1], Aigrain, Joly and Longueville propose a multimodal rule based approach. They first identify local (in time) rules which are given by the medium contents; and then construct scenes (they call *macro-segments*) by combining the rules. In [17, 4], the video stream is first segmented into shots. Then a *time-constrained clustering* is used to construct visually similar and temporally adjacent shots into clusters. Finally, a *Scene Transition Graph* (STG) is constructed based on the clusters, and *cutting edges* are identified to construct the scene structure. In [2], instead of using STG, the authors group shots of alternating patterns into scenes (they call *acts*). A 2D presentation of the video structure is then created, with scenes displayed vertically and key frames displayed horizontally.

The advantages of scene based video ToC over the other approaches are summarized as follows:

- The other approaches produce too many entries to be efficiently presented to the viewer.
- Shots, key frames, and even groups convey only physical discontinuities while scenes convey semantic discontinuities, such as scene changes in time and/or location.

3 The Proposed Approach

Based on the discussion in the previous section, it is obvious that scene based ToC has advantages over other approaches. Within the scene based approaches, the multimodal rule based method is not at a matured stage yet. More rules need to be generated and tested before the method can be put into practice. Furthermore, the process of generating the rules may be as time consuming as generating application models. We thus focus our attention along the approaches developed in [17, 4, 2].

Our proposed approach utilizes an intelligent unsupervised clustering technique for scene-level video ToC construction. It better models the "temporal locality" without having the "window effect", and it constructs more accurate scene structures by taking comprehensive information into account. It has four major modules: shot boundary detection and key frame extraction, spatio-temporal feature extraction, time-adaptive grouping, and scene structure construction. We discuss each of the modules in turn below:

3.1 Shot Boundary Detection and Key Frame Extraction

As described in Section 2.1, a lot of work has been done in shot boundary detection and many of the approaches achieve satisfactory performance [22]. In this paper, we use an approach similar to the one used in [23]. For key frame extraction, although more sophisticated techniques exist [27, 8, 15], they require high computation effort. We select the beginning and ending frames of a shot as the two key frames to achieve fast processing speed.

3.2 Spatio-Temporal Feature Extraction

At the shot level, the shot activity measure is extracted to characterize the temporal information:

$$Act_i = \frac{1}{N_i - 1} \sum_{k=1}^{N_i - 1} Diff_{k,k-1} \quad (1)$$

$$Diff_{k,k-1} = Dist(Hist(k), Hist(k-1)) \quad (2)$$

where Act_i and N_i are the activity measure and number of frames for shot i; $Diff_{k,k-1}$ is the color histogram difference between frames k and $k-1$; $Hist(k)$ and $Hist(k-1)$ are the color histograms for frames k and $k-1$; $Dist()$ is a distance measure between histograms. We adopt the Intersection Distance [13] in this paper. The color histograms used are $2D$ histograms along the H and S axes in HSV color space. We disregard V component because of its less robustness to the lighting condition.

At the key frame level, visual features are extracted to characterize the spatial information. In the current algorithm, color histograms of the beginning and ending frames are used as the visual feature for the shot:

$$Hist(b_i) \quad (3)$$

$$Hist(e_i) \quad (4)$$

where b_i and e_i are the beginning and ending frames of shot i.

Based on the above discussion, a shot is modeled as:

$$shot_i = shot_i(b_i, e_i, Act_i, Hist(b_i), Hist(e_i)) \tag{5}$$

which captures both the *spatial* and the *temporal* information of a shot. At higher levels, this spatial-temporal information is used in grouping and scene structure construction.

3.3 Time-Adaptive Grouping

Before we construct the scene structure, it is convenient to first create an intermediate entity *group* to facilitate the later process. The purpose of grouping is to group similar shots into groups, since similar shots have high possibilities to be in the same scene. For shots to be similar, the following properties should be satisfied:

- Visual similarity

  Similar shots should be visually similar. That is, they should have similar spatial ($Hist(b_i)$ and $Hist(e_i)$) and temporal (Act_i) features.

- Time locality

  Similar shots should be close to each other temporally [17]. For example, visually similar shots, if far apart from each other in time, seldom belong to the same scene and hence not to the same group.

In [17], Yeung et al. proposed a *time-constrained clustering* approach to grouping shots, where the similarity between two shots is set to 0 if their time difference is greater than a predefined threshold. We propose a more general *time-adaptive grouping* approach based on the two properties for similar shots described above. In our proposed approach, the similarity of two shots is an increasing function of visual similarity and a decreasing function of frame difference. Let i and j be the indices for the two shots whose similarity is to be determined, where *shot* $j >$ *shot* i. The calculation of the shot similarity is described as follows:

1. Calculate the shot color similarity $ShotColorSim$:

   (a) Calculate the four raw frame color similarities: $FrameColorSim_{b_j,e_i}$, $FrameColorSim_{e_j,e_i}$, $FrameColorSim_{b_j,b_i}$, and $FrameColorSim_{e_j,b_i}$, where $FrameColorSim_{x,y}$ is defined as:

   $$FrameColorSim_{x,y} = 1 - Diff_{x,y} \tag{6}$$

   where x and y are two arbitrary frames, with $x > y$.

(b) To model the importance of time locality, we introduce the concept of *temporal attraction*, $Attr$, which is a decreasing function of the frame difference:

$$Attr_{b_j,e_i} = max(0, 1 - \frac{b_j - e_i}{baseLength}) \quad (7)$$

$$Attr_{e_j,e_i} = max(0, 1 - \frac{e_j - e_i}{baseLength}) \quad (8)$$

$$Attr_{b_j,b_i} = max(0, 1 - \frac{b_j - b_i}{baseLength}) \quad (9)$$

$$Attr_{e_j,b_i} = max(0, 1 - \frac{e_j - b_i}{baseLength}) \quad (10)$$

$$baseLength = MULTIPLE * avgShotLength \quad (11)$$

where $avgShotLength$ is the average shot length of the whole video stream; $MULTIPLE$ is a constant which controls how fast the temporal attraction will decrease to 0. For our experiment data, we find $MULTIPLE = 10$ gives good results. The above definition of *temporal attraction* says that the farther apart the frames, the less the *temporal attraction*. If the frame difference is larger than $MULTIPLE$ times the average shot length, the attraction decreases to 0.

(c) Convert the raw similarities to *time-adaptive* similarities, which capture both the visual similarity and time locality:

$$FrameColorSim'_{b_j,e_i} = Attr_{b_j,e_i} \times FrameColorSim_{b_j,e_i} \quad (12)$$

$$FrameColorSim'_{e_j,e_i} = Attr_{e_j,e_i} \times FrameColorSim_{e_j,e_i} \quad (13)$$

$$FrameColorSim'_{b_j,b_i} = Attr_{b_j,b_i} \times FrameColorSim_{b_j,b_i} \quad (14)$$

$$FrameColorSim'_{e_j,b_i} = Attr_{e_j,b_i} \times FrameColorSim_{e_j,b_i} \quad (15)$$

(d) The color similarity between shots i and j is defined as the maximum of the four frame similarities:

$$ShotColorSim_{i,j} = max(\quad FrameColorSim'_{b_j,e_i}, \quad FrameColorSim'_{e_j,e_i}, \quad (16)$$

$$FrameColorSim'_{b_j,b_i}, \quad FrameColorSim'_{e_j,b_i}) \quad (17)$$

2. Calculate the shot activity similarity $ShotActSim$:

$$ShotActSim_{i,j} = Attr_{center} \times |Act_i - Act_j| \quad (18)$$

$$Attr_{center} = max(0, 1 - \frac{(b_j + e_j)/2 - (b_i + e_i)/2}{baseLength}) \quad (19)$$

where $Attr_{center}$ is the *temporal attraction* between the two center frames of shot i and shot j.

3. Calculate the overall shot similarity $ShotSim$:

$$ShotSim_{i,j} = W_C * ShotColorSim_{i,j} + W_A * ShotActSim_{i,j} \tag{20}$$

where W_C and W_A are appropriate weights for color and activity measures.

3.4 Scene Structure Construction

Similar shots are grouped into a group, but even non-similar groups can be grouped into a single scene if they are *semantically related*. Video is a sequential medium. Therefore, even though two or more processes are developing simultaneously in a video, they have to be displayed sequentially, one after another. This is common in movie. For example, when two people are talking to each other, even though both people contribute to the conversation, the movie switches back and forth between these two people. In this example, clearly there exist two groups, one corresponding to person A, and the other corresponding to person B. Even though these two groups are non-similar groups, they are semantically related and constitute a single scene.

We propose to use an intelligent unsupervised clustering technique to perform scene structure construction. This is achieved in a two-step process:

- collect similar shots into groups using *time-adaptive grouping*, and
- merge semantically related groups into a unified scene.

The advantages of the proposed approach over existing approaches are summarized in the following:

- *Temporal continuity*: In [17], a time-window of width T is used in the *time-constrained clustering*. Similarly, in [2], a search window of 8 shots long is used when calculating the shot similarities. While this "window" approach is a big advance from the plain unsupervised clustering in video analysis, it has the problem of discontinuity ("window effects"). For example, if the frame difference between two shots is $T-1$, then the similarity between these two shots is kept unchanged. But if these two shots are a bit further apart from each other, making the frame difference to be $T+1$, the similarity between these shots is suddenly cleared to 0. This discontinuity may cause potential wrong clustering and make the clustering results sensitive to the window size. To overcome this discontinuity problem, in our proposed approach, we introduce the concept of *temporal attraction*, which is a continuous

and decreasing function of frame difference (Equations 7-11 and Equation 19). Temporal attraction effectively models the importance of time locality and does not cause any discontinuity in grouping.

- *Direct merge to a scene*: In many cases, current shot may not be similar enough to any group in a scene; thus can not be directly merged to the scene. However, it may be similar to a certain degree to most of the groups in a scene. For example, a camera shoots three sets of shots of a person from three angles: 30^o, 60^o, and 45^o. Obviously, the three sets of shots will form three groups. Suppose the first two groups have already been formed and the current shot is a shot in group 3. While the current shot may not be very similar to either of groups 1 and 2, it is similar to some extent to both groups 1 and 2; and all the three groups are semantically related. This situation occurs quite often in video shooting. The approaches in [17] and [2] will not be effective in handling this, since both of them only compare current shot to the individual groups but not to the scene as a whole. In the proposed approach, besides calculating the similarities between current shot and groups (Step 3 in the main procedure below), we also calculate the similarities between current shot and the scene which consists of multiple groups (Step 4 in the main procedure). This added functionality effectively takes into account the above example situation and merges all the 3 groups into a unified scene.

The details of the proposed approach is described below.

[**Main procedure**]

- Input: Video shot sequence, $S = \{shot\ 0, ..., shot\ i\}$.
- Output: Video structure in terms of *scene*, *group*, and *shot*.
- Procedure:
  1. Initialization: assign shot 0 to group 0 and scene 0; initialize the group counter $numGroups = 1$; initialize the scene counter $numScenes = 1$.
  2. If S is empty, quit; otherwise get the next shot. Denote this shot as shot i.
  3. Test if shot i can be merged to an existing group:
     (a) Compute the similarities between the current shot and existing groups: Call $findGroupSim()$.
     (b) Find the maximum group similarity:

$$maxGroupSim_i = \max_g \quad GroupSim_{i,g} \quad , g = 1, ..., numGroups \tag{21}$$

where $GroupSim_{i,g}$ is the similarity between shot i and group g. Let the group of the maximum similarity be group g_{max}.

(c) Test if this shot can be merged into an existing group:

If $maxGroupSim_i > groupThreshold$, where $groupThreshold$ is a predefined threshold:

i. Merge shot i to group g_{max}.

ii. Update the video structure: Call $updateGroupScene()$.

iii. Goto Step 2.

otherwise:

i. Create a new group containing a single shot i. Let this group be group j.

ii. Set $numGroups = numGroups + 1$.

4. Test if shot i can be merged to an existing scene:

(a) Calculate the similarities between the current shot i and existing scenes: Call $findSceneSim()$.

(b) Find the maximum scene similarity:

$$maxSceneSim_i = \max_s \quad SceneSim_{i,s} \quad , s = 1, ..., numScenes \tag{22}$$

where $SceneSim_{i,s}$ is the similarity between shot i and scene s. Let the scene of the maximum similarity be scene s_{max}.

(c) Test if shot i can be merged into an existing scene:

If $maxSceneSim_i > sceneThreshold$, where $sceneThreshold$ is a predefined threshold:

i. Merge shot i to scene s_{max}.

ii. Update the video structure: Call $updateScene()$.

otherwise:

i. Create a new scene containing a single shot i and a single group j.

ii. Set $numScenes = numScenes + 1$.

5. Goto Step 2.

The input to the algorithm is an unstructured video stream while the output is a structured video consisting of scenes, groups, shots, and key frames, based on which the video ToC is constructed (Figure 2).

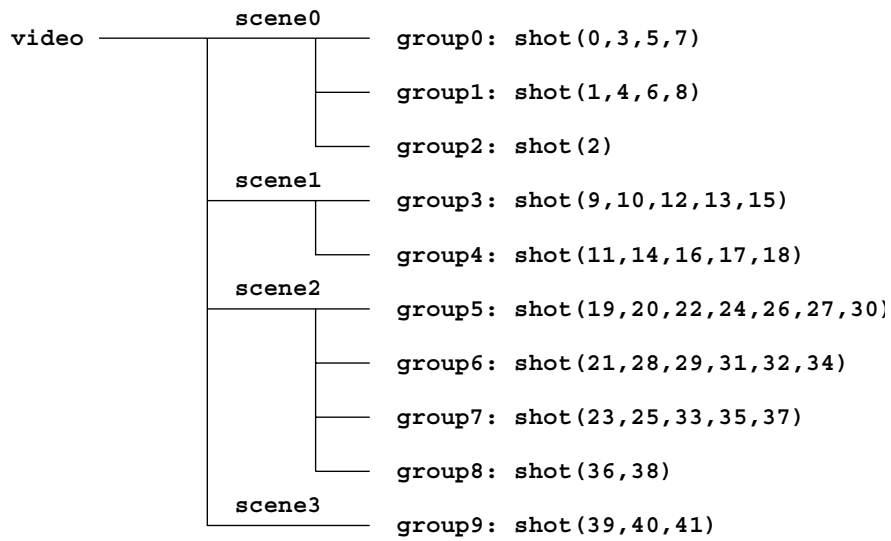

Figure 2: An example video ToC

[**findGroupSim**]

- Input: Current shot and group structure.
- Output: Similarity between current shot and existing groups.
- Procedure:
  1. Denote current shot as shot i.
  2. Calculate the similarities between shot i and existing groups:

$$GroupSim_{i,g} = ShotSim_{i,g_{last}} \quad , g = 1, ..., numGroups \tag{23}$$

     where g is the index for groups and g_{last} is the last (most recent) shot in group g. That is, the similarity between current shot and a group is the similarity between the current shot and the most recent shot in the group. The reason of choosing the most recent shot to represent the whole group is that all the shots in the same group are visually similar and the most recent shot has the largest *temporal attraction* to the current shot.

  3. Return.

[**findSceneSim**]

- Input: Current shot, group structure and scene structure.
- Output: Similarity between current shot and existing scenes.
- Procedure:
  1. Denote current shot as shot i.
  2. Calculate the similarity between shot i and existing scenes:

$$SceneSim_{i,s} = \frac{1}{numGroups_s} \sum_{g}^{numGroups_s} GroupSim_{i,g} \tag{24}$$

  where s is the index for scenes; $numGroups_s$ is the number of groups in scene s; and $GroupSim_{i,g}$ is the similarity between current shot i and g^{th} group in scene s. That is, the similarity between current shot and a scene is the average of similarities between current shot and all the groups in the scene.

  3. Return.

[**updateGroupScene**]

- Input: Current shot, group structure, and scene structure.
- Output: An updated version of group structure and scene structure.
- Procedure:
  1. Denote current shot as shot i and the group having the largest similarity to shot i as group g_{max}. That is, shot i belongs to group g_{max}.
  2. Define two shots *top* and *bottom*, where *top* is the second most recent shot in group g_{max} and *bottom* is the most recent shot in group g_{max} (i.e. current shot).
  3. For any group g, if any of its shots (*shot* g_j) satisfies the following condition

$$top < shot\ g_j < bottom \tag{25}$$

  merge the scene that group g belongs to into the scene that group g_{max} belongs to. That is, if a scene contains a shot which is interlaced with the current scene, merge the two scenes. This is illustrated in Figure 3 (shot i = shot 4, $g_{max} = 0$, $g = 1$, *top* = shot 1, and *bottom* = shot 4).

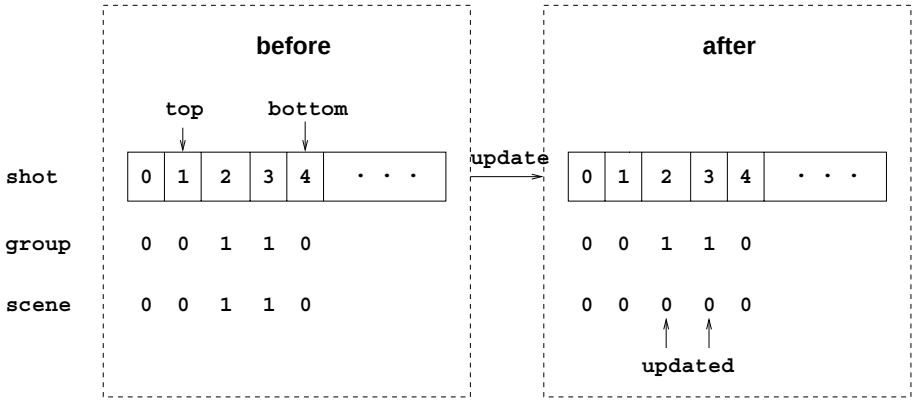

Figure 3: Merging scene 1 to scene 0

4. Return.

[**updateScene**]

- Input: Current shot, group structure, and scene structure.
- Output: An updated version of scene structure.
- Procedure:
  1. Denote current shot as shot i and the scene having the largest similarity to shot i as scene s_{max}. That is, shot i belongs to scene s_{max}.
  2. Define two shots *top* and *bottom*, where *top* is the second most recent shot in scene s_{max} and *bottom* is the current shot in scene s_{max} (i.e. current shot).
  3. For any scene s, if any of its shots ($shot\ s_j$) satisfies the following condition

$$top < shot\ s_j < bottom \tag{26}$$

     merge scene s into scene s_{max}. That is, if a scene contains a shot which is interlaced with the current scene, merge the two scenes.
  4. Return.

What distinguishes the proposed approach from the plain clustering based approach is the intelligence involved. The intelligence manifests itself in the algorithm in three aspects. First, the "temporal attraction"

is an intelligent way of modeling the temporal factor of the similarity. Second, *updateGroupScene* intelligently merges related groups to a single scene. Finally, *updateScene* intelligently updates related scenes into a unified one.

The procedure *updateGroupScene* and *updateScene* are of significant importance to the proposed scene structure construction algorithm. While $findGroupSim$ helps to group similar shots into a group and $findSceneSim$ helps to merge a shot (or a single-element group) into a scene, it is *updateGroupScene* and *updateScene* that link semantically related shots into a single scene. For example, for scene 0 in Figure 2, while $findGroupSim$ helps to group shots 0, 3, 5, 7 into group 0; and $findSceneSim$ helps to group shot 2 to scene 0, it is *updateGroupScene* and *updateGroupScene* that link all the three groups into one unified scene.

4 Determination of the Parameters

There are 4 parameters in the proposed video ToC construction algorithm: W_C, W_A, $groupThreshold$, and $sceneThreshold$. For any algorithm to be of practical use, all the parameters should be determined either automatically by the algorithm itself or easily by the user. In our proposed algorithm, Gaussian normalization is used in determining the 4 parameters. Specifically, W_C and W_A are determined automatically by the algorithm, and $groupThreshold$ and $sceneThreshold$ are determined by user's interaction.

4.1 Gaussian Normalization

In Equation (20), we combine color histogram similarity and activity similarity to form the overall shot similarity. Since the color histogram feature and activity feature are from two totally different physical domains, it would be meaningless if we combine them together without normalizing them first. The Gaussian normalization process ensures that entities from different domains are normalized to the same dynamic range. The normalization procedure is described as follows:

[**findMeanAndStddev**]

- Input: Video shot sequence, $S = \{shot\ 0, ..., shot\ i\}$ and a feature F associated with the shots. For example, the feature F can be either color histogram feature or activity feature.
- Output: The mean μ and standard deviation σ of this feature F for this video.
- Procedure:

1. If S is not empty, get the next shot; otherwise goto 3.
2. Denote current shot as shot i.
   (a) Compute the similarity in terms of F between shot i and shot i', $i' = i - MULTIPLE, ..., i - 1$. Note that only the similarities of the previous $MULTIPLE$ shots need to be calculated, since shots outside $MULTIPLE$ have zero *temporal attraction* to the current shot.
   (b) Store the calculated similarity values in an array A_s.
   (c) Goto step 1.
3. Let N_A be the number of entries in array A_s. Consider this array as a sequence of Gaussian variables and compute the mean μ_{A_s} and standard deviation σ_{A_s} of the sequence.

The means and standard deviations for color histogram and activity measure are first calculated (denoted as μ_C, σ_C, μ_A, and σ_A) by the above normalization procedure before the scene construction procedure is applied in Section 3.4. During the scene construction procedure, μ_C, σ_C, μ_A, and σ_A are used to convert the raw similarity values to normalized ones. That is, Step 3 in Section 3.4 (Equation (20)) is modified into:

3. Calculate the overall shot similarity:

$$ShotSim_{i,j} = W_C * ShotColorSim'_{i,j} + W_A * ShotActSim'_{i,j} \quad (27)$$

$$ShotColorSim'_{i,j} = \frac{ShotColorSim_{i,j} - \mu_C}{\sigma_C} \quad (28)$$

$$ShotActSim'_{i,j} = \frac{ShotActSim_{i,j} - \mu_A}{\sigma_A} \quad (29)$$

where W_C and W_A are appropriate weights for color and activity measures; $ShotColorSim_{i,j}$ and $ShotActSim_{i,j}$ are the raw similarity values. The above procedure converts the raw similarities into similarities obeying the normal distribution of $N(0,1)$. Being of the same distribution, the normalized color histogram similarity and the normalized activity similarity can be meaningfully combined into an overall similarity. How to determine the appropriate values for W_C and W_A is discussed in the following sub-section.

4.2 Determining W_C and W_A

After the Gaussian normalization procedure, the raw similarities of both color histogram and activity measure are brought into the same dynamic range. That is, the normalized similarities of the color histogram feature and the activity feature contribute equally to the overall similarity. To reflect the relative importance of each feature, different weights are then associated with the features.

The relative "importance" of a feature can be estimated from the statistics of its feature array A_s. For example, if all the elements in A_s are of similar value, then this particular feature is of little discriminating power and should receive low weight. On the other hand, if the elements in A_s demonstrate variation, then the feature has good discriminating power and should receive high weight. Based on this intuition, the standard deviation of the feature array A_s furnishes a good estimation of the feature's importance (weight). In our case, W_C and W_A can be automatically determined as follows:

$$W_C = \frac{\sigma_C}{\sigma_C + \sigma_A} \tag{30}$$

$$W_A = \frac{\sigma_A}{\sigma_C + \sigma_A} \tag{31}$$

where σ_C and σ_A are obtained from the procedure **findMeanAndStddev**.

4.3 Determining *groupThreshold* and *sceneThreshold*

The *groupThreshold* and *sceneThreshold* are two important parameters in the proposed algorithm. The determination of these two thresholds would be difficult and time-consuming, if the shot similarities were not normalized, since in that case the thresholds are

- *feature dependent*: different features (color histogram vs. activity measure) may require different thresholds;
- *case dependent*: different videos may require different thresholds.

But after the Gaussian normalization procedure, the similarity distribution of any feature for any video is normalized to the Gaussian $N(0,1)$ distribution, making the determination of thresholds much easier. The Gaussian $N(0,1)$ distribution is plotted in Figure 4.

A learning process can be designed to find appropriate values for the two thresholds. Since any feature's similarity in any video is mapped to the same distribution, the training feature and video can be arbitrary. During the training, human manually adjusts the two thresholds to obtain good video scene structure. This learning process needs to be done only once. The determined two thresholds can then be used for any other features in any other videos. Experimentally we find that $groupThreshold = 0.65 \times \sigma = 0.65$ and $sceneThreshold = 0$ give good scene structure. This set of thresholds are used through out all the experiments reported in Section 5. Note that $P(X < x | x = groupThreshold = 0.65) = 0.74$ and $P(X < x | x = sceneThreshold = 0) = 0.5$. These two probabilities indicate that

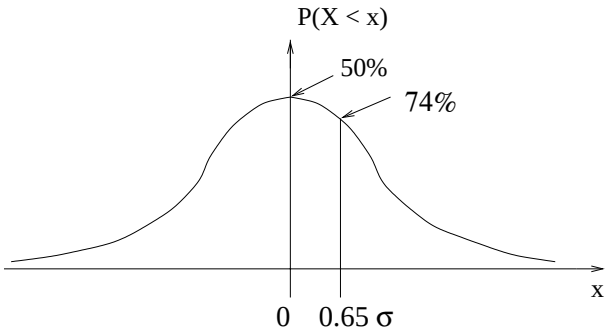

Figure 4: The Gaussian N(0,1) distribution

- only if a shot is very similar to a group (better than 74% of all the shots) can it be merged to the group, and
- when a shot is similar to some extent (better than 50% of all the shots) to all the groups in a scene, it is considered similar to the scene,

which match the physical meaning of the two thresholds, as discussed in Section 3.4.

5 Experimental Results

In all the experiments reported in this section, the video streams are MPEG compressed, with the digitization rate equal to 30 frames/sec. To validate the effectiveness of the proposed approach, representatives of various movie types are tested. Specifically, the test set includes Movie1 (romantic-slow), Movie2 (romantic-fast), Movie3 (music), Movie4 (comedy), Movie5 (science fiction-slow), Movie6 (science fiction-fast), and Movie7 (action). Each video clip is about 10-20 minutes long, and the total length is about 175,000 frames. The experimental results are shown in Table 1, where "detected scenes" denotes the number of scenes detected by the algorithm; "false negatives" indicates the number of scenes missed by the algorithm; and "false positives" indicates the number of scenes detected by the algorithm but are not considered as scenes by human.

Since *scene* is a semantic level concept, the ground truth of scene boundary is not always concrete and this might be the reason that the authors of [17, 4] and [2] do not include the two columns of "false negative" and "false positive" in their experimental result tables. But in order to judge the effectiveness of a video ToC construction approach, we believe it is useful to include those two columns. Although *scene* is a semantic concept, relative agreement can be reached among different people. The ground truth of the scene boundaries for the tested video sequences are obtained from subjective tests. Multiple human subjects

Table 1. Scene structure construction results.

movie name	frames	shots	groups	detected scenes	false negatives	false positives
Movie1	21717	133	16	5	0	0
Movie2	27951	186	25	7	0	1
Movie3	14293	86	12	6	1	1
Movie4	35817	195	28	10	1	2
Movie5	18362	77	10	6	0	0
Movie6	23260	390	79	24	1	10
Movie7	35154	329	46	14	1	2

are invited to watch the movies and then asked to give their own scene structures. The structure that most people agreed with is used as the ground truth of the experiments.

From the results in Table 1, some observations can be made:

- The proposed scene construction approach achieves reasonably good results in most of the movie types.

- The approach achieves better performance in the "slow" movies than in the "fast" movies. This is because that in the "fast" movies, the visual content is normally more complex and more difficult to capture. We are currently integrating close-caption information into the framework to enhance the accuracy of the scene structure construction.

- The proposed approach seldom misses a scene boundary, but tends to over-segment the video. That is, the number of the "false positive" is more than that of "false negative". This situation is expected for most of the automated video analysis approaches and has also been observed by other researchers [17, 4].

- The proposed approach is practical in terms of parameter determination. A single set of thresholds ($groupThreshold = 0.65 \times \sigma = 0.65$ and $sceneThreshold = 0$) is used through out the experiments, and W_C and W_A are determined automatically by the algorithm itself, as described in Section 4.

The scene structure information (Figure 2), together with the representative frames, leads to a semantic level ToC to facilitate user's access to the video. The scene level ToC is illustrated in Figure 5.

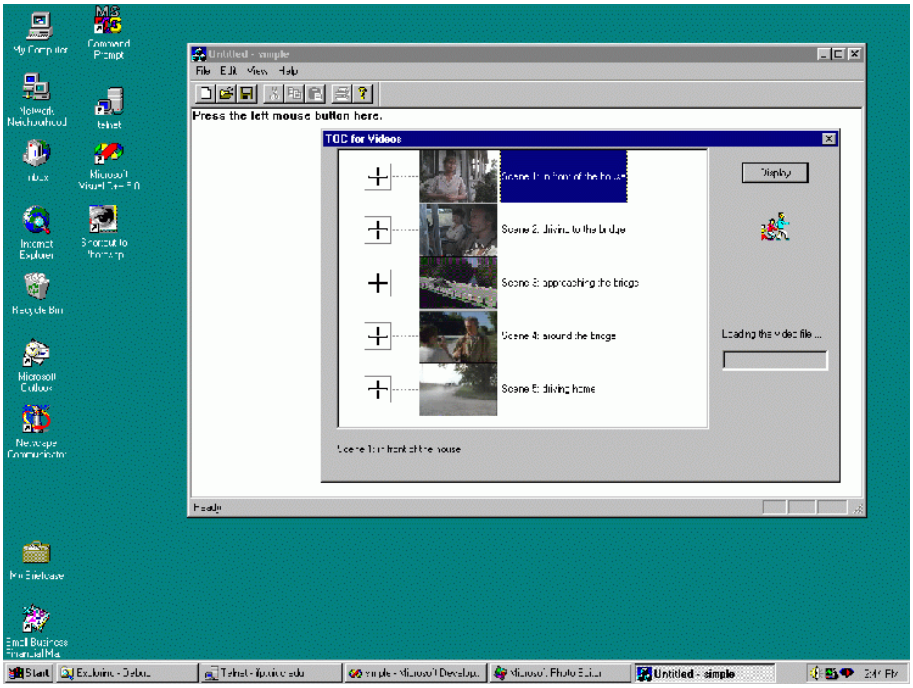

Figure 5: Video ToC for Movie1 (scene level)

In Figure 5, five scenes are created from the 21717-frame video clip (Movie1). By looking at the representative frames and the text annotation, the user can easily grasp the global content of the video clip. He or she can also expand the video ToC into more detailed levels, such as groups and shots (see Figure 6).

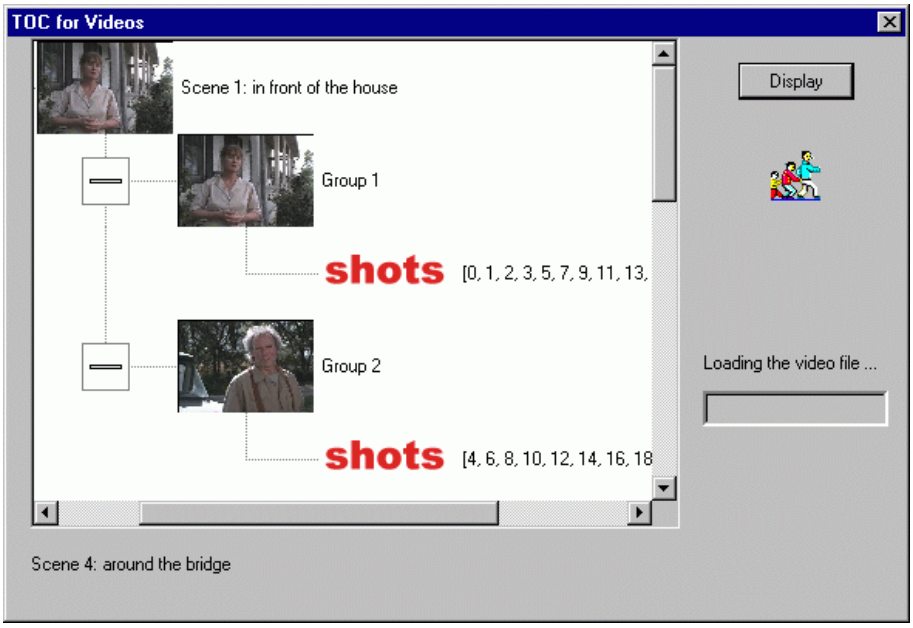

Figure 6: Video ToC for Movie1 (group level)

After browsing the hierarchical video ToC (scenes, groups, shots, etc.), the user can quickly position the portion of the video that he or she is interested in. Then, by clicking the "Display" button, the user can view that particular portion of video, without doing tedious "fast forward" and "rewind" (see Figure 7).

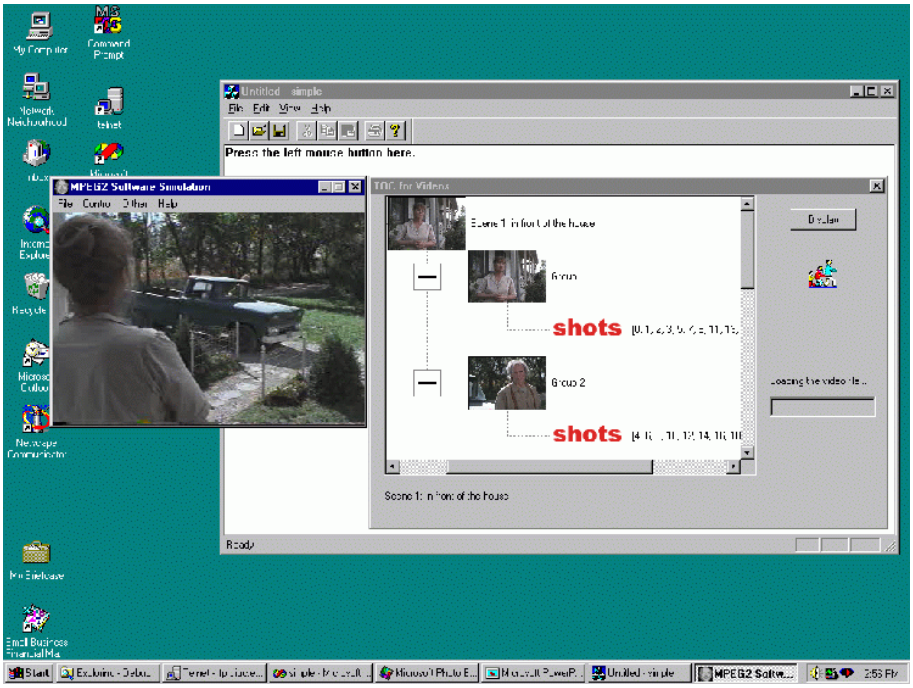

Figure 7: Video ToC for Movie1 (non-linear access)

Imagine the situation if we do not have a ToC for a long book. It may take hours to just grasp the main content. The same is true for videos. The above scene based video ToC greatly facilitates user's access to the video. It not only provides the user with a non-linear access to the video (in contrast to conventional linear *fast-forward* and *rewind*), but also gives the user a global "picture" of the whole story.

If, instead, we were using a 1D array of key frames to present the same video, $2 \times 133 = 266$ frames have to be presented sequentially. Because of the 1D linear display nature, even if a user can patiently browse through all the 266 frames, it is still difficult for him or her to perceive the underlying story structure.

6 Conclusions

Motivated by the important role of ToC in a book, we introduced the ToC into the video domain in this paper. We reviewed and evaluated the video parsing techniques at various levels (shots, groups, scenes).

We concluded that the scene level ToC has the advantages over other techniques. We then presented an effective scene level ToC construction technique based on intelligent unsupervised clustering. It has the characteristics of better modeling the time locality and scene structure. Experiments over real-world movie videos validate the effectiveness of the proposed approach. Examples are given in demonstrating the usage of the scene based ToC to facilitate user's access to the video.

The proposed approach provides an *open* framework for structure analysis of video – features other than the ones used in this paper can be readily incorporated for the purpose of video ToC construction. We are currently exploring integration of other visual features, as well as, audio features (speech and background music) and text features (close caption) into the developed framework. We envision that an appropriate fusion of these multi-modalities can result in a more semantically correct video ToC.

References

[1] P. Aigrain, P. Joly, and V. Longueville. Medium knowledge-based macro-segmentation of video into sequences. In *IJCAI Workshop on Intelligent Multimedia Information Retrieval*, pages 5–14, 1995.

[2] Hisashi Aoki, Shigeyoshi Shimotsuji, and Osamu Hori. A shot classification method of selecting effective key-frames for video browsing. In *Proc. ACM Conf. on Multimedia*, 1995.

[3] Farshid Arman, Arding Hsu, and Ming-Yee Chiu. Feature management for large video databases. In *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[4] Ruud M. Bolle, Boon-Lock Yeo, and Minerva M. Yeung. Video query: Beyond the keywords. Technical report, IBM Research Report, Oct 17 1996.

[5] John S. Boreczky and Lawrence A. Rowe. Comparison of video shot boundary detection techniques. In *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1996.

[6] Ralph M. Ford, Craig Robson, Daniel Temple, and Michael Gerlach. Metrics for scene change detection in digital video sequences. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1997.

[7] Yihong Gong, Lim Teck Sin, Chua Hock Chuan, Hongjian Zhang, and Masao Sakauchi. Automatic parsing of tv soccer programs. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1995.

[8] P. O. Gresle and T. S. Huang. Gisting of video documents: A key frames selection algorithm using relative activity measure. In *The 2nd Int. Conf. on Visual Information Systems*, 1997.

[9] A. Hampapur, R. Jain, and T. Weymouth. Digital video segmentation. In *Proc. ACM Conf. on Multimedia*, 1994.

[10] R. Kasturi and R. Jain. Dynamic vision. In *Computer Vision: Principles*, 1991.

[11] Jianhao Meng, Yujen Juan, and Shih-Fu Chang. Scene change detection in a mpeg compressed video sequence. In *SPIE Symposium on Electronic Imaging: Science & Technology- Digital Video Compression: Algorithms and Technologies*, 1995.

[12] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In *Visual Database Systems II*, 1992.

[13] Michael Swain and Dana Ballard. Color indexing. *International Journal of Computer Vision*, 7(1), 1991.

[14] Deborah Swanberg, Chiao-Fe Shu, and Ramesh Jain. Knowledge guided parsing in video databases. In *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[15] Wayne Wolf. Key frame selection by motion analysis. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, 1996.

[16] Boon-Lock Yeo. Efficient processing of compressed images and video. Technical report, PhD thesis, Princeton University, 1996.

[17] Minerva Yeung, Boon-Lock Yeo, and Bede Liu. Extracting story units from long programs for video browsing and navigation. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1996.

[18] Minerva Yeung, Boon-Lock Yeo, W. Wolf, and Bede Liu. Video browsing using clustering and scene transitions on compressed sequences. In *Proc. of Multimedia Computing and Networking*, volume SPIE 2417, 1995.

[19] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *Proc. ACM Conf. on Multimedia*, 1995.

[20] H. Zhang, Yihong Gong, S. W. Smoliar, and Shuang Yeo Tan. Automatic parsing of news video. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1994.

[21] H. Zhang, S. W. Smoliar, and J. J. Wu. Content-based video browsing tools. In *Porc. IS&T/SPIE Conf. on Multimedia Computing and Networking*, 1995.

[22] H.J. Zhang, John Y. A. Wang, and Yucel Altunbasak. Content-based video retrieval and compression: A unified solution. In *Proc. IEEE Int. Conf. on Image Proc.*, 1997.

[23] HongJiang Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *ACM Multimedia Systems*, 1(1), 1993.

[24] HongJiang Zhang, C. Y. Low, Stephen W. Smoliar, and D. Zhong. Video parsing, retrieval and browsing: An integrated and content-based solution. In *Proc. ACM Conf. on Multimedia*, 1995.

[25] HongJiang Zhang and Stephen W. Smoliar. Developing power tools for video indexing and retrieval. In *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1994.

[26] Di Zhong, HongJiang Zhang, and Shih-Fu Chang. Clustering methods for video browsing and annotation. Technical report, Columbia Univ., 1997.

[27] Yueting Zhuang, Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *Proc. IEEE Int. Conf. on Image Proc.*, 1998.

Yong Rui received the B.S. degree from Southeast University, P. R. China in 1991 and the M.S. degree from Tsinghua University, P. R. China in 1994, both in Electrical Engineering. He is currently finishing the Ph.D. degree in Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. He was a Graduate Research Assistant at the Department of Automation, Tsinghua University from 1991 to 1994. He joined KangTuo Microcomputer Corp., Beijing in 1994 as a Research Engineer. Since 1995 he has been a Graduate Research Assistant at the Image Formation and Processing Group of the Beckman Institute for Advance Science and Technology at UIUC. During the summer of 1998 he is a summer Research Intern at the Vision Technology Group of Microsoft Research, in Redmond, WA.

His research interests include multimedia information retrieval, multimedia signal processing, computer vision and artificial intelligence. He has published over 20 technical papers in the above areas.

He is a Huitong University Fellowship recipient 1989-1990, a Guanghua University Fellowship recipient 1992-1993, and a CSE Engineering College Fellowship recipient 1996-1998.

Thomas S. Huang received his B.S. Degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, China; and his M.S. and Sc.D. Degrees in Electrical Engineering from the Massachusetts Institute of Technology, Cambridge, Massachusetts. He was on the Faculty of the Department of Electrical Engineering at MIT from 1963 to 1973; and on the Faculty of the School of Electrical Engineering and Director of its Laboratory for Information and Signal Processing at Purdue University from 1973 to 1980. In 1980, he joined the University of Illinois at Urbana-Champaign, where he is now William L. Everitt Distinguished Professor of Electrical and Computer Engineering, and Research Professor at the Coordinated Science Laboratory, and Head of the Image Formation and Processing Group at the Beckman Institute for Advanced Science and Technology.

During his sabbatical leaves, Dr. Huang has worked at the MIT Lincoln Laboratory, the IBM Thomas J. Watson Research Center, and the Rheinishes Landes Museum in Bonn, West Germany, and held visiting Professor positions at the Swiss Institutes of Technology in Zurich and Lausanne, University of Hannover in West Germany, INRS-Telecommunications of the University of Quebec in Montreal, Canada and University of Tokyo, Japan. He has served as a consultant to numerous industrial firms and government agencies both in the U.S. and abroad.

Dr. Huang's professional interests lie in the broad area of information technology, especially the transmission and processing of multidimensional signals. He has published 12 books, and over 300 papers in Network Theory, Digital Filtering, Image Processing, and Computer Vision. He is a Fellow of the International Association of Pattern Recognition, IEEE, and the Optical Society of American; and has received a Guggenheim Fellowship, an A.V. Humboldt Foundation Senior U.S. Scientist Award, and a Fellowship from the Japan Association for the Promotion of Science. He received the IEEE Acoustics, Speech, and Signal Processing Society's Technical Achievement Award in 1987, and the Society Award in 1991. He is a Founding Editor of the International Journal Computer Vision, Graphics, and Image Processing; and Editor of the Springer Series in Information Sciences, published by Springer Verlag.

Sharad Mehrotra received his M.S. and PhD at the University of Texas at Austin in 1990 and 1993 respectively, both in Computer Science. Subsequently he worked at MITL, Princeton as a scientist from 1993-1994. He is an assistant professor in the Computer Science department at the University of Illinois at Urbana-Champaign since 1994. He specializes in the areas of database management, distributed systems, and information retrieval. His current research projects are on multimedia analysis, content-based retrieval of multimedia objects, multidimensional indexing, uncertainty management in databases, and concurrency and transaction management. Dr. Mehrotra is an author of over 50 research publications in these areas. Dr. Mehrotra is the recipient of the NSF Career Award and the Bill Gear Outstanding junior faculty award in 1997.