A Unified Framework for Video Browsing and Retrieval

Yong Rui and Thomas S. Huang

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

1 Introduction

Research on how to efficiently access video content has become increasingly active in the past few years [1, 2, 3, 4]. Considerable progress has been made in video analysis, representation, browsing, and retrieval, which are the four fundamental bases for accessing video content. *Video analysis* deals with the *signal processing* part of the video system, including shot boundary detection, key frame extraction, etc. *Video representation* is concerned with the *structure* of the video. An example of a video representation is the tree structured key frame hierarchy [5, 3]. Built on top of the video representation, *video browsing* deals with how to use the representation structure to help viewers browse the video content. Finally, *video retrieval* is concerned with retrieving interesting video objects. The relationship between these four research areas is illustrated in Figure 1.

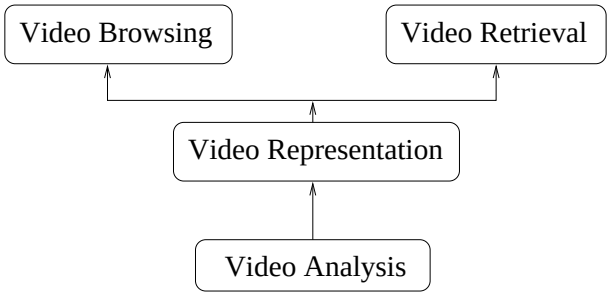

Figure 1: Relations between the four research areas

So far, most of the research effort has gone into video analysis. Although it is the basis for all the other research activities, it is not the ultimate goal. Relatively less research exists on video representation, browsing and retrieval. As seen in Figure 1, video browsing and retrieval are on the very top of the diagram. They *directly* support users' access to the video content. For accessing a temporal medium, such as a video clip, browsing and retrieval are equally important. Browsing helps a user to quickly grasp the global picture of the data, while retrieval helps a user to find a

specific query's results.

An analogy explains this argument. How does a reader efficiently access a 1000-page book's content? Without reading the whole book, he can first go to the book's Table-of-Contents (ToC), finding which chapters or sections suit his needs. If he has specific questions (queries) in mind, such as finding a term or a key word, he can go to the Index page and find the corresponding book sections containing that question. In short, the book's ToC helps a reader *browse*, and the book's index helps a reader *retrieve*. Both aspects are equally important in helping users access the book's content. For today's video data, unfortunately, we lack both the ToC and the Index. Techniques are urgently needed for automatically (or semi-automatically) constructing video ToCs and video Indexes to facilitate browsing and retrieval.

A great degree of power and flexibility can be achieved by simultaneously designing the video access components (ToC and Index) using a unified framework. For a long and continuous stream of data, such as video, a "back and forth" mechanism between browsing and retrieval is crucial.

The goals of this chapter are to develop novel techniques for constructing both the video ToC and video Index as well as how to integrate them into a unified framework. The rest of the chapter is organized as follows. In section 2, important video terminologies are first introduced. We review video analysis, representation, browsing, and retrieval in sections 2 to 5, respectively. In Section 6 we describe in detail a unified framework for video browsing and retrieval. Algorithms as well as experimental results on real-world video clips are presented. Conclusions and future research directions are summarized in section 7.

2 Terminologies

Before we go into the details of the discussion, it will be beneficial to first introduce some important terminologies used in the digital video research field.

- *Video shot*: is a consecutive sequence of frames recorded from a single camera. It is the building block of video streams.
- *Key frame*: is the frame which represents the salient visual content of a shot. Depending on the complexity of the content of the 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 shots are marked by physical boundaries, scenes are marked by semantic boundaries [1].
- *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 [5] or visually similar shots [3].

In summary, the video data can be structured into a hierarchy consisting of five levels: video, scene, group, shot, and key frame, which increase in granularity from top to bottom [4] (see Figure 2).

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

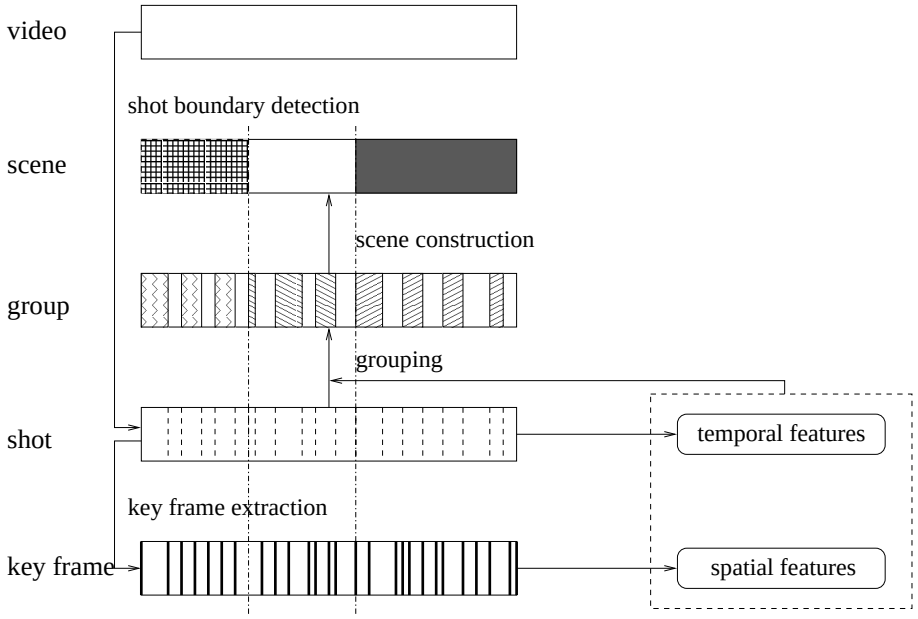

Figure 2: A hierarchical video representation

3 Video Analysis

As can be seen from Figure 1, video analysis is the basis for later video processing. It includes *shot boundary detection* and *key frame extraction*.

3.1 Shot boundary detection

It is not efficient (sometimes not even possible) to process a video clip as a whole. It is beneficial to first decompose the video clip into shots and do signal processing at the shot level.

In general, automatic shot boundary detection techniques can be classified into five categories: *pixel-based*, *statistics-based*, *transform-based*, *feature-based*, and *histogram-based*. Pixel-based approaches use pixel-wise intensity difference to mark shot boundaries [1, 6]. However, they are highly sensitivity to noise. To overcome this problem, Kasturi and Jain propose to use intensity statistics (mean and standard deviation) as shot boundary detection measures [7]. Seeking to achieve faster processing, Arman, Hsu and Chiu propose to use the compressed DCT coefficients (e.g., MPEG data) as the boundary measure [8]. Other transform-based shot boundary detection approaches make use of motion vectors, which are already embedded in the MPEG stream [9, 10]. Zabih et al. address the problem from another angle. Edge features are first extracted from each frame. Shot boundaries are then detected by finding sudden edge changes [11]. So far, the histogram-based approach is the most popular. Instead of using pixel intensities directly, the histogram-based approach uses histograms of the pixel intensities as the measure. Several researchers claim that it achieves a good tradeoff between accuracy and speed [1]. Representatives of this approach are [12, 13, 1, 14, 15]. More recent work has been based on clustering and post-

filtering [16], which achieves fairly high accuracy without producing many false positives. Two comprehensive comparisons of shot boundary detection techniques are [17, 18].

3.2 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 [15]. More sophisticated key frame extraction techniques are based on visual content complexity indicators [19], shot activity indicators [20], and shot motion indicators [21].

4 Video Representation

Considering that each video frame is a 2D object and the temporal axis makes up the third dimension, a video stream spans a 3D space. Video representation is the *mapping* from the 3D space to the 2D view screen. Different mapping functions characterize different video representation techniques.

4.1 Sequential Key Frame Representation

After obtaining shots and key frames, an obvious and simple video representation is to sequentially lay out the key frames of the video, from top to bottom and from left to right. This simple technique works well when there are few key frames. When the video clip is long, this technique does not scale, since it does not capture the embedded information within the video clip, except for time.

4.2 Group Based Representation

To obtain a more meaningful video representation when the video is long, related shots are merged into groups [5, 3]. In [5], Zhang et al. divide the entire video stream into multiple video segments, each of which contains an equal number of consecutive shots. Each segment is further divided into sub-segments; thus constructing a tree structured video representation. In [3], Zhong et al. proposed a cluster-based video hierarchy, in which the shots are clustered based on their visual content. This method again constructs a tree structured video representation.

4.3 Scene Based Representation

To provide the user with better access to the video, the construction of a video representation at the semantic level is needed [4, 2]. It is not uncommon for a modern movie to contain a few thousand shots and key frames. This is evidenced in [22] – there are 300 shots in a 15-minute video segment of the movie "Terminator 2 - 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 (or even a tree structured layout at the group level) is almost meaningless. More importantly, people watch the video by its semantic scenes rather than 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 [2]. Video ToC construction at the scene level is thus of fundamental importance to video browsing and retrieval. In [2], a scene transition graph (STG) of video representation is proposed and constructed. The video sequence is first segmented into shots. Shots are then clustered by using *time-constrained clustering*. The STG is then constructed based on the time flow of the clusters.

4.4 Video Mosaic Representation

Instead of representing the video structure based on the video-scene-group-shot-frame hierarchy as discussed above, this approach takes a different perspective [23]. The mixed information within a shot is decomposed into three components:

- *Extended spatial information*: this captures the appearance of the entire background imaged in the shot, and is represented in the form of a few mosaic images.
- *Extended temporal information*: this captures the motion of independently moving objects in the form of their trajectories.
- *Geometric information*: this captures the geometric transformations that are induced by the motion of the camera.

5 Video Browsing and Retrieval

These two functionalities are the ultimate goals of a video access system, and they are closely related to (and built on top of) video representations. The first three representation techniques discussed above are suitable for video browsing while the last can be used in video retrieval.

5.1 Video Browsing

For "Sequential Key Frame Representation", browsing is obviously sequential browsing, scanning from the top-left key frame to the bottom-right key frame.

For "Group Based Representation", a hierarchical browsing is supported [5, 3]. At the coarse level, only the main themes are displayed. Once the user determines which theme he is interested in, he can then go to the finer level of the theme. This refinement process can go on until the leaf level.

For the STG representation, a major characteristic is its indication of time flow embedded within the representation. By following the time flow, the viewer can browse through the video clip.

5.2 Video Retrieval

As discussed in Section 1, both the ToC and Index are equally important for accessing the video content. Unlike the other video representations, the mosaic representation is especially suitable for

video retrieval. Three components: moving objects, backgrounds, and camera motions, are perfect candidates for a video Index. After constructing such a video index, queries such as "find me a car moving like this", "find me a conference room having that environment", etc. can be effectively supported.

6 Proposed Framework

As we have reviewed in the previous sections, considerable progress has been made in each of the areas of video analysis, representation, browsing, and retrieval. However, so far, the interaction among these components is still limited and we still lack a unified framework to glue them together. This is especially crucial for video, given that the video medium is characteristically long and unstructured. In this section, we will explore the synergy between video browsing and retrieval.

6.1 Video Browsing

Among the many possible video representations, the "Scene Based Representation" is probably the most effective for meaningful video browsing [4, 2]. We have proposed a scene-based video ToC representation in [4]. In this representation, a video clip is structured into the scene-group-shot-frame hierarchy (see Figure 2), which then serves as the basis for the ToC construction. This ToC frees the viewer from doing tedious "fast forward" and "rewind", and provides the viewer with non-linear access to the video content. Figures 3 and 4 illustrate the browsing process, enabled by the video ToC. Figure 3 shows a condensed ToC for a video clip, as we normally have in a long book. By looking at the representative frames and text annotation, the viewer can determine which particular portion of the video clip he is interested in. Then, the viewer can further expand the ToC into more detailed levels, such as groups and shots. The expanded ToC is illustrated in Figure 4. Clicking on the "Display" button will display the specific portion that is of interest to the viewer, without viewing the entire video.

The algorithm is described below. To learn details, interested readers are referred to [24].

[**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 $findGroupSim()$.

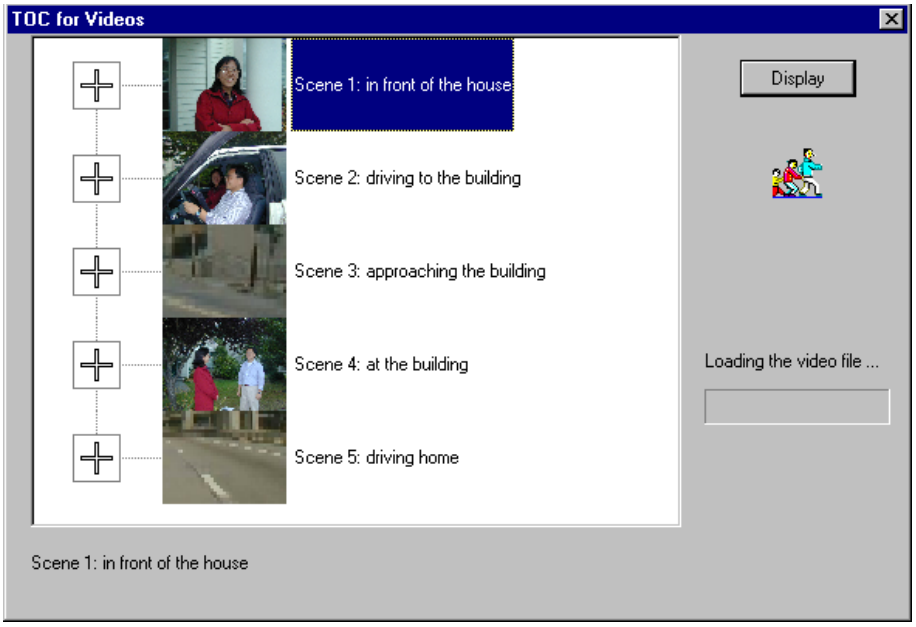

Figure 3: The condensed ToC

(b) Find the maximum group similarity:

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

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:

$$maxSceneSim_i = \max_s \quad SceneSim_{i,s} \quad , s = 1, ..., numScenes \tag{2}$$

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

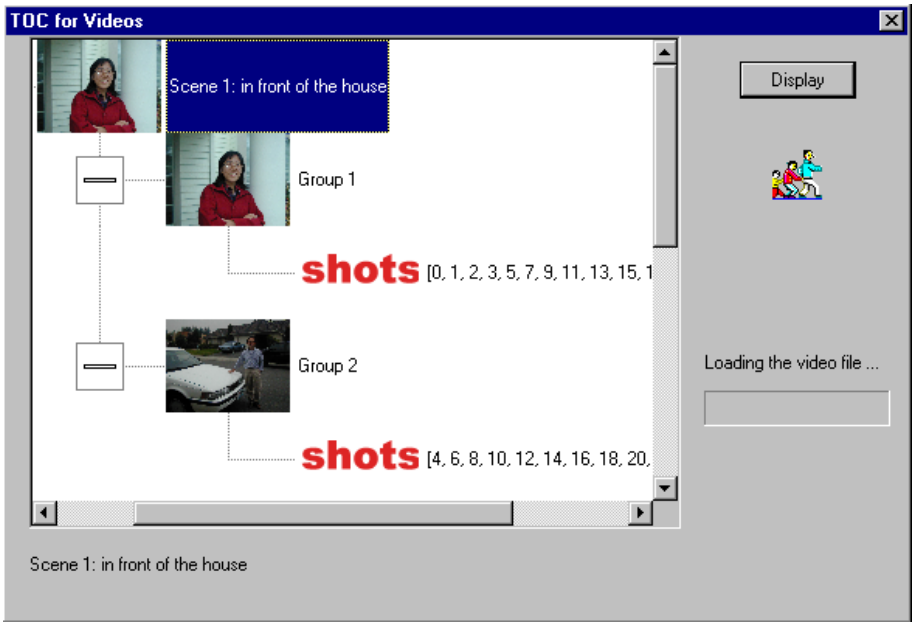

Figure 4: The expanded ToC

(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.

[**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:

$$GroupSim_{i,g} = ShotSim_{i,g_{last}} \quad , g = 1, ..., numGroups \tag{3}$$

where $ShotSim_{i,j}$ is the similarity between shots i and j; and 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 most recent shot is chosen to represent the whole group because 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: The current shot, group structure and scene structure.
- Output: Similarity between the current shot and existing scenes.
- Procedure:
  1. Denote the current shot as shot i.
  2. Calculate the similarity between shot i and existing scenes:

$$SceneSim_{i,s} = \frac{1}{numGroups_s} \sum_{g}^{numGroups_s} GroupSim_{i,g} \tag{4}$$

  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 the current shot and a scene is the average of similarities between the 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

$$top < shot\ g_j < bottom \tag{5}$$

  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).

  4. Return.

[**updateScene**]

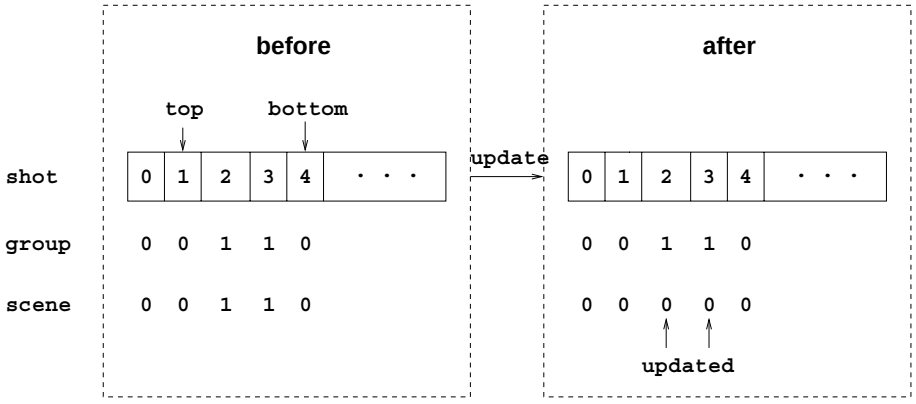

Figure 5: Merging scene 1 to scene 0

- 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

$$top < shot\ s_j < bottom \tag{6}$$

     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.

Extensive experiments using real-world video clips have been carried out. The results are summarized in Table 1 [4], where "ds" (detected scenes) denotes the number of scenes detected by the algorithm; "fn" (false negatives) indicates the number of scenes missed by the algorithm; and "fp" (false positives) indicates the number of scenes detected by the algorithm, although they are considered scenes by humans.

Some observations can be summarized as follows:

- The proposed scene construction approach achieves reasonably good results in most of the movie types.
- The approach achieves better performance in "slow" movies than in "fast" movies. This follows since in the "fast" movies, the visual content is normally more complex and more difficult to capture. We are currently integrating closed-captioning information into the framework to enhance the accuracy of the scene structure construction.

Table 1. Scene structure construction results.

movie name	frames	shots	groups	ds	fn	fp
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

- The proposed approach seldom misses a scene boundary, but tends to over-segment the video. That is, "false positives" outnumber "false negatives". This situation is expected for most of the automated video analysis approaches and has also been observed by other researchers [22, 2].

6.2 Video Retrieval

Video retrieval is concerned with how to return similar video clips (or scenes, shots, and frames) to a user given a video query. This is a little explored research area. There are two major categories of existing work. One is to first extract key frames from the video data, then use image retrieval techniques to obtain the video data *indirectly*. Although easy to implement, it has the obvious problem of losing the temporal dimension. The other technique incorporates motion information (sometimes object tracking) into the retrieval process. Although this is a better technique, it requires the computationally expensive task of motion analysis. If object trajectories are to be supported, then this becomes more difficult.

Here we view video retrieval from a different angle. We seek to construct a video Index to suit various users' needs. However, constructing a video Index is far more complex than constructing an index for books. For books, the form of an index is fixed (e.g., key words). For videos, the viewer's interests may cover a wide range. Depending on his or her knowledge and profession, the viewer may be interested in semantic level labels (building, car, people), low level visual features (color, texture, shape), or the camera motion effects (pan, zoom, rotation). In the system described here, we support all three Index categories:

- Visual Index
- Semantic Index
- Camera motion Index

To support semantic level and visual feature-based queries, frame clusters are first constructed to provide indexing. Our clustering algorithm is described as follows:

1. Feature extraction: color and texture features are extracted from each frame. The color feature is an 8×4 2D color histogram in Hue-Saturation-Value (HSV) color space. The V component is not used because of its sensitivity to lighting conditions. The H component is quantized finer than the S component due to the psychological observation that the human

visual system is more sensitive to Hue than to Saturation. For texture features, the input image is fed into a wavelet filter bank and is then decomposed into de-correlated sub-bands. Each sub-band captures the feature of a given scale and orientation from the original image. Specifically, we decompose an image into three wavelet levels; thus 10 sub-bands. For each sub-band, the standard deviation of the wavelet coefficients is extracted. The 10 standard deviations are used as the texture representation for the image [25].

2. Global clustering: based on the features extracted from each frame, the entire video clip is grouped into clusters. A detailed description of the clustering process can be found in [19]. Note that each cluster can contain frames from multiple shots and each shot can contain multiple clusters. The cluster centroids are used as the visual Index and can be later labeled as a semantic Index (see section 6.3.). This procedure is illustrated in Figure 6.

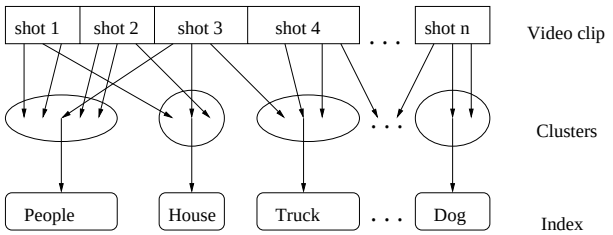

Figure 6: From video clip to cluster to index

After the above clustering process, the entire video clips are grouped into multiple clusters. Since color and texture features are used in the clustering process, all the entries in a given cluster are visually similar. Therefore these clusters naturally provide support for the visual queries.

In order to support semantic level queries, semantic labels need to be provided for each cluster. There are two techniques that have been developed in our research lab. One is based on the Hidden Markov Model (HMM) and the other is an annotation based approach. Since the former approach also needs training samples, both approaches are semi-automatic. To learn details of the first approach, readers are referred to [16]. We will introduce the second approach here. Instead of attempting to attack the unsolved automatic image understanding problem, semi-automatic human assistance is used. We have built interactive tools to display each cluster centroid frame to a human user, who will label that frame. The label will then be *propagated* through the whole cluster. Since only the cluster centroid frame needs labeling, the interactive process is fast. For a 21,717 frame video clip (Movie1), about 20 minutes is needed. After this labeling process, the clusters can support both visual and semantic queries. The specific semantic labels for Movie1 are people, car, dog, tree, grass, road, building, house, etc.

To support camera motion queries, we have developed techniques to detect camera motion in the MPEG compressed domain [26]. The incoming MPEG stream does not need to be fully decompressed. The motion vectors in the bit stream form good estimates of camera motion effects. Hence, panning, zooming, and rotation effects can be efficiently detected [26].

6.3 A Unified Framework for Browsing and Retrieval

The above two subsections described video browsing and retrieval techniques separately. In this section, we integrate them into a unified framework to enable a user to go "back and forth" between browsing and retrieval. Going from the Index to the ToC, a user can get the *context* where the indexed entity is located. Going from the ToC to the Index, a user can *pin point* specific queries. Figure 7 illustrates the unified framework.

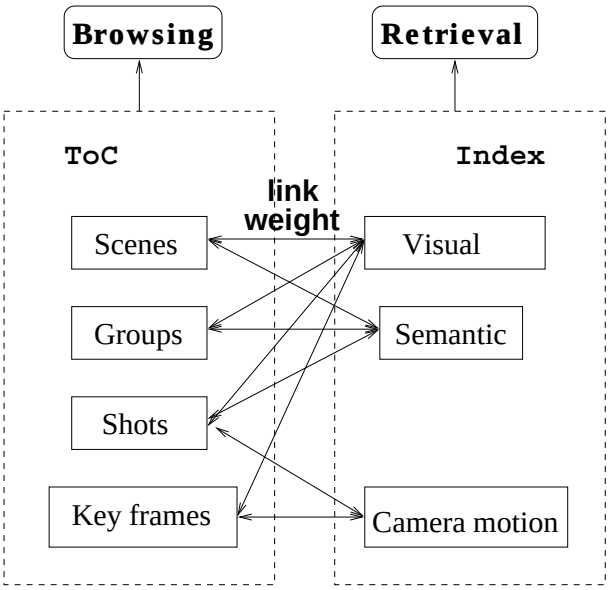

Figure 7: A unified framework

An essential part of the unified framework is the weighted links. The links can be established between Index entities and scenes, groups, shots, and key frames in the ToC structure. As a first step, in this paper we focus our attention on the links between Index entities and shots. Shots are the building blocks of the ToC. Other links are generalizable from the shot link.

To link shots and the *visual Index*, we propose the following techniques. As we mentioned before, a cluster may contain frames from multiple shots. The frames from a particular shot form a sub-cluster. This sub-cluster's centroid is denoted as c_{sub} and the centroid of the whole cluster is denoted as c. This is illustrated in Figure 8.

Here c is a representative of the whole cluster (and thus the visual Index) and c_{sub} is a representative of the frames from a given shot in this cluster. We define the similarity between the cluster centroid and sub-cluster centroid as the link weight between Index entity c and that shot.

$$w_v(i,j) = similarity(c_{sub}, c_j) \tag{7}$$

where i and j are the indices for shots and clusters, respectively, and $w_v(i,j)$ denotes the link weight between shot i and visual Index cluster c_j.

After defining the link weights between shots and the visual Index, and labeling each cluster, we can next establish the link weights between shots and the *semantic Index*. Note that multiple

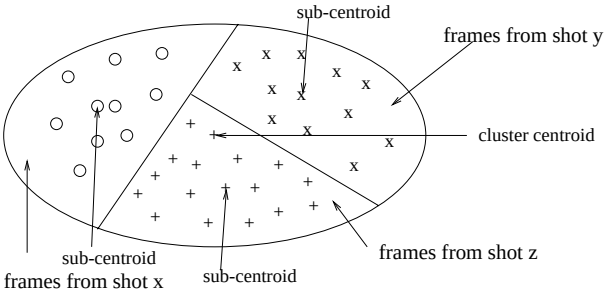

Figure 8: Sub-clusters

Table 2. From the semantic, visual, camera Index to the ToC.

shot id	0	2	10	12	14	31	33
w_s	0.958	0.963	0.919	0.960	0.957	0.954	0.920
shot id	16	18	20	22	24	26	28
w_v	0.922	0.877	0.920	0.909	0.894	0.901	0.907
shot id	0	1	2	3	4	5	6
w_c	0.74	0.03	0.28	0.17	0.06	0.23	0.09

clusters may share the same semantic label. The link weight between a shot and a semantic Index is defined as:

$$w_s(i,k) = max_j(w_v(i,j)) \tag{8}$$

where k is the index for the semantic Index entities; and j represents those clusters sharing the same semantic label k.

The link weight between shots and a *camera motion Index* (e.g., panning) is defined as:

$$w_c(i,l) = \frac{n_i}{N_i} \tag{9}$$

where l is the index for the camera operation Index entities; n_i is the number of frames having that camera motion operation; and N_i is the number of frames in shot i.

Extensive tests have been carried out using real-world video clips. The video streams are MPEG compressed, with the digitization rate equal to 30 frames/s. Table 2 summarizes example results over the video clip Movie1. The first two rows are an example of going from the semantic Index (e.g., car) to the ToC (shots) (also in Figure 9). The middle two rows are an example of going from the visual Index (e.g., Figure 10) to the ToC (shots) (also in Figure 11). The last two rows are going from the camera operation Index (panning) to the ToC (shots).

By just looking at each isolated Index alone, a user usually cannot understand the context. By going from the Index to the ToC (as in Table 2), a user quickly learns when and under which

Table 3. From the ToC (shots) to the Index.

Index	fence	mail box	human hand	mirror	steer wheel
Weight	0.927	0.959	0.918	0.959	0.916

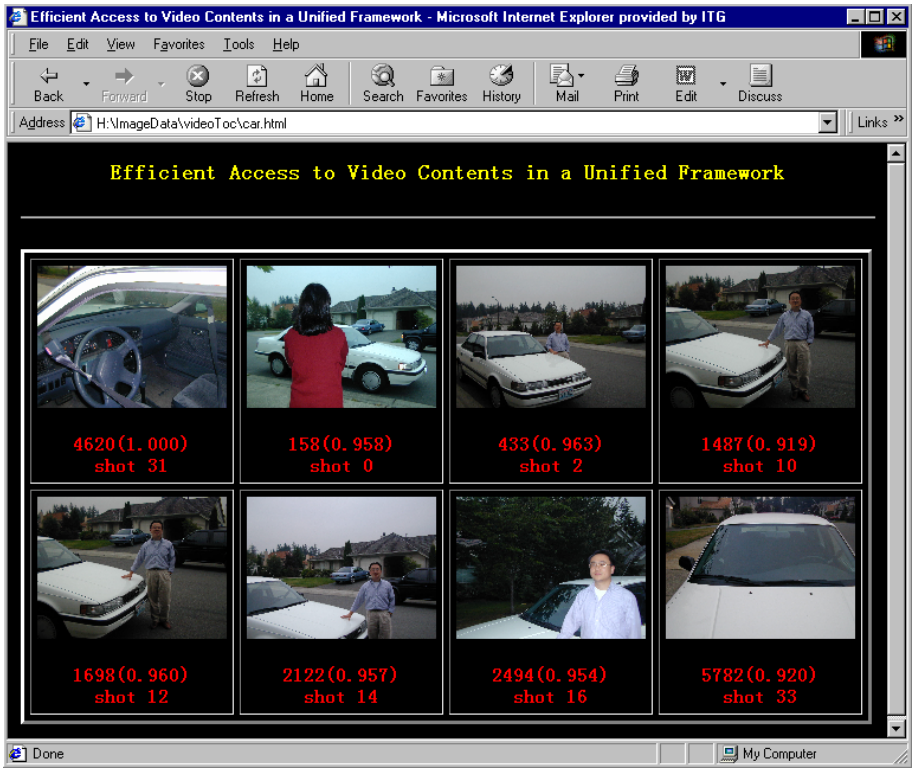

Figure 9: Interface for going from the semantic Index to the ToC

circumstances (e.g., within a particular scene) that Index entity is happening. Table 2 summarizes how to go from the Index to the ToC to find the *context*. We can also go from the ToC to the Index to *pin point* a specific Index. Table 3 summarizes which Index entities appeared in shot 33 of the video clip Movie1.

For a continuous and long medium such as video, a "back and forth" mechanism between browsing and retrieval is crucial. Video library users may have to browse the video first before they know what to retrieve. On the other hand, after retrieving some video objects, the users will be better able to browse the video in the correct direction. We have carried out extensive subjective tests employing users from various disciplines. Their feedback indicates that this unified framework greatly facilitated their access to video content, in both home entertainment and educational applications.

7 Conclusions and Promising Research Directions

This chapter covered the following topics:

- Reviewed and discussed recent research progress in video analysis, representation, browsing, and retrieval;

Figure 10: Frame 2494 as a visual Index

- Introduced the video ToC and the Index and presented techniques for constructing them;
- Proposed a unified framework for video browsing and retrieval; and proposed techniques for establishing the link weights between the ToC and the Index.

We should be aware that video is not just a visual medium. It contains text and audio information in addition to visual information and is thus "true" multimedia. Multi-model and multimedia processing is usually more reliable and robust than processing a single medium. We need to further extend our investigation to the integration of closed-captioning and audio track information into our algorithm to enhance the construction of ToCs, Indexes and link weights.

8 Acknowledgment

This work was supported in part by ARL Cooperative Agreement No. DAAL01-96-2-0003, and in part by a CSE Fellowship, College of Engineering, UIUC. The authors also would like to thank Sean X. Zhou and Roy R. Wang for their valuable discussions.

References

[1] H. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *ACM Multimedia Sys.*, vol. 1, no. 1, pp. 1–12, 1993.

[2] R. M. Bolle, B.-L. Yeo, and M. M. Yeung, "Video query: Beyond the keywords," Technical Report, IBM Research, Oct. 17, 1996.

[3] D. Zhong, H. Zhang, and S.-F. Chang, "Clustering methods for video browsing and annotation," Tech. Rep., Columbia University, 1997.

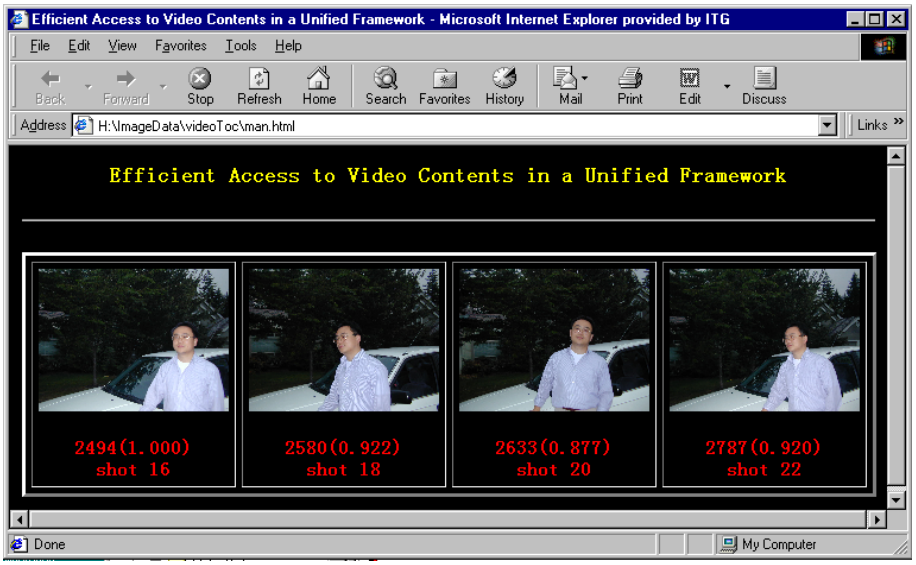

Figure 11: Interface for going from the visual Index to the ToC

[4] Y. Rui, T. S. Huang, and S. Mehrotra, "Exploring video structures beyond the shots," in *Proc. of IEEE Conf. Multimedia Computing and Systems*, 1998.

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

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

[7] R. Kasturi and R. Jain, "Dynamic vision," in *Proc. of Computer Vision: Principles*, 1991.

[8] F. Arman, A. Hsu, and M.-Y. Chiu, "Feature management for large video databases," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[9] J. Meng, Y. Juan, and S.-F. Chang, "Scene change detection in a mpeg compressed video sequence," in *Proc. SPIE Symposium on Electronic Imaging: Science & Technology- Digital Video Compression: Algorithms and Technologies*, 1995.

[10] B.-L. Yeo, "Efficient processing of compressed images and video," Ph.D. dissertation, Princeton University, 1996.

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

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

[13] D. Swanberg, C.-F. Shu, and R. Jain, "Knowledge guided parsing in video databases," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[14] H. Zhang and S. W. Smoliar, "Developing power tools for video indexing and retrieval," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1994.

[15] H. Zhang, C. Y. Low, S. W. Smoliar, and D. Zhong, "Video parsing, retrieval and browsing: An integrated and content-based solution," in *Proc. ACM Conf. on Multimedia*, 1995.

[16] M. R. Naphade, R. Mehrotra, A. M. Ferman, T. S. Huang, and A. M. Tekalp, "A high performance algorithm for shot boundary detection using multiple cues," in *Proc. IEEE Int. Conf. on Image Proc.*, (Chicago), Oct. 1998.

[17] J. S. Boreczky and L. A. Rowe, "Comparison of video shot boundary detection techniques," in *Proc. SPIE Storage and Retrieval for Image and Video Databases*, 1996.

[18] R. M. Ford, C. Robson, D. Temple, and M. Gerlach, "Metrics for scene change detection in digital video sequences," in *Proc. IEEE Conf. on Multimedia Comput. and Syss*, 1997.

[19] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *Proc. IEEE Int. Conf. on Image Proc.*, 1998.

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

[21] W. Wolf, "Key frame selection by motion analysis," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, 1996.

[22] M. Yeung, B.-L. Yeo, and B. Liu, "Extracting story units from long programs for video browsing and navigation," in *Proc. IEEE Conf. on Multimedia Comput. and Syss*, 1996.

[23] M. Irani and P. Anandan, "Video indexing based on mosaic representations," *Proceedings of IEEE*, vol. 86, pp. 905–921, May 1998.

[24] Y. Rui, T. S. Huang, and S. Mehrotra, "Constructing table-of-content for videos," *to appear in Journal of ACM Multimedia Sys.*, 1999.

[25] Y. Rui, T. S. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in MARS," in *Proc. IEEE Int. Conf. on Image Proc.*, 1997.

[26] J. A. Schmidt, "Object and camera parameter estimation using mpeg motion vectors," M.S. thesis, University of Illinois at Urbana-Champaign, 1998.