Content-based Browsing of Video Sequences

F. Arman, R. Depommier, A. Hsu, and M-Y. Chiu
Siemens Corporate Research, Inc.
755 College Road East, Princeton, New Jersey, 08540 USA
farshid@scr.siemens.com

ABSTRACT

A novel methodology to represent the contents of a video sequence is presented. The representation is used to allow the user to rapidly view a video sequence in order to find a particular point within the sequence and/or to decide whether the contents of the sequence are relevant to his or her needs. This system, referred to as content-based browsing, forms an abstraction to represent each shot of the sequence by using a representative frame, or an Rframe, and it includes management techniques to allow the user to easily navigate the Rframes. This methodology is superior to the current techniques of fast forward and rewind because rather than using every frame to view and judge the contents, only a few abstractions are used. Therefore, the need to retrieve the video from a storage system and to transmit every frame over the network in its entirety no longer exists, saving time, expenses, and bandwidth.

1. INTRODUCTION

This paper addresses the issue of rapidly viewing the contents of a given video sequence, a process referred to in this paper as browsing. Browsing video sequences is critical in many domains and applications in which the user is either required to choose a few video sequences from among many, and/or the user has to find a particular point within a single video sequence.

Such cases exist in numerous situations, such as remote access of video, video database navigation, video editing, video-based education and training, and, in the near future, video e-mail and recorded desk-top video conferencing sessions. In all such cases, the user must view the contents of the video sequences in order to choose the most relevant or to locate the desired point. Assigned labels, keyword descriptions, and database indexing are useful in reducing the possibilities somewhat; however, in most cases the user is still left to decide among at least a few possibilities. Consider, for instance, the case in which the user has submitted a query to a remote database and the database search has resulted in several possibilities. At this point the user must decide if the context and contents of the returned videos match his or her needs. This may only be achieved by viewing each of the returned videos. Viewing video would require that each video be retrieved from, presumably, a hierarchical storage system, transmitted over the network in its entirety as the user plays the video or at most fast forwards and rewinds. This process is time consuming, inefficient, not cost effective, and a waste of bandwidth.

We propose a solution in which abstractions of each of the video sequences is pre-computed, and only the abstractions are retrieved from the system, transmitted, if necessary, and viewed by the user. The abstractions are many orders of magnitude smaller in size, and, therefore, reduce the system's response time, bandwidth needs, and, most importantly, the user's viewing time. In addition, the proposed system allows the user to rapidly pinpoint the desired location within a video sequence.

Content-based video browsing is achieved by a few pre-processing steps which are performed off-line before the user gains access and a few steps during the browsing driven by the users' particular needs. Processing during the browsing is necessary because each user is different and may have varying needs at different times even for the same sequence. The pre-processing steps include detecting scene changes, or video cuts, to form video shots followed by a very simple motion analysis on each video shot, used to form an abstraction of the video shot. In addition, the content-based video browser performs shape and color content analyses. The abstraction for each video shot, referred to as a representative frame or an Rframe, are displayed to the user maintaining the temporal order of appearance. In cases where many Rframes exist, which is entirely possible, then Rframe management techniques are employed to aid the user. These techniques rely on shape and color-based pre-processing steps and are a function of users' commands and choices.

'The proposed content-based browsing is advantageous over fast forward and rewind (FF/REW) while remaining as convenient to use. Using FF/REW the user must view every frame at rapid speeds, missing shots that last a short period, while being forced to watch long lasting and irrelevant shots. In addition, users searching for a specific point within a sequence are inevitably forced to refine their search after many fast forwards and rewinds until the video is at the precise point of interest, a time consuming and tedious task. In the content-based browser, the exact points of scene changes are defined internally, and no fine tuning by the user is necessary. Note that above disadvantages of FF/REW persist even on digital video and on other random access media, such as laser disks. Lastly, FF/REW as the means for browsing of digital video is extremely inefficient considering the expense of accessing disks and/or tapes, decoding, and transmission.

The reminder of this paper is organized as follows: Section 2 presents related research in browsing digital video. Section 3 presents the typical usage scenarios, and Section 4 presents technical details of the browser. Conclusions are presented in Section 5.

2. RELATED WORK

Analogous to headlines in a newspaper, which are used for fast browsing, video sequences require a similar "basic browsing unit" which can also be used. Unlike newspapers, however, where an editor manually chooses the headline for each article, the process of choosing the video browsing unit must be automatic and efficient to deal with the enormous amount of data that is normally associated with video collections. In our browser, as is the case with some of the others, the basic browsing unit is the video shot, defined as a subset of the sequence in-between two consecutive scene changes.

Earlier, we proposed a novel approach to process encoded video sequences for scene change detection [arman93e, arman94]. It is reasonable to assume the input to the system is encoded since digital video requirements for storage and communication are extremely high; hence, many standards for video encoding have been developed and are currently in use. Our method, referred to as selective decoding, takes advantage of the information already encoded in a DCT-based compressed video data, such as MPEG, JPEG, and H.261, and performs many processing steps needed on every frame of a video sequence prior to full decompression. Certain blocks of the encoded frames, which are defined a priori but are arbitrary, are monitored over time. Using only a few of the DCT coefficients of each selected block, a vector is formed as a compact representation of the frame, which is then used in detecting scene changes. Scene change detection has been addressed by others, for example [tonomura90] and [udea91] discuss using histograms, and [miller93] prefer a manual approach in their system. In the remainder of this paper we assume that the browser has been provided with a list of scene changes, and the exact processing steps is not relevant for the content-based browser presented here.

Tonomura et. al [tonomura90] introduced several approaches to view the contents of video shots: variable speed, sampling flash, rush, and time-space browser. The variable speed browser, is very similar to VCR's jog and shuttle functions; the sampling flash browser is a series of icons formed from the first frame of each video shot without any clues to the contents; in the rush browser, instead of using video shots the sequence is divided along equally spaced time intervals; and the time-space browser displays a temporal sequence on several icons. In [tonomura93] much emphasis is placed on characterizing the contents of video shots with respect to camera and object motions.

Similar to Tonomura, Elliot [elliot93] introduced a browser which stacks every frame of the sequence. This approach suffers from several shortcomings: First, the stack is built as the user is watching the sequence. This is not useful for video browsing because the user is "forced" to watch the video sequence because the stack can make sense only once you have seen the video. The second shortcoming is that the stack holds only about 20 seconds of video; this amount of video is not practical for use in actual cases. Third, once the stack is built, the use may "stroke" the stack to watch the contents. This is a minor improvement, from the user's point of view, to FF/REW. Lastly, we believe that this approach fails to provide the user with a basic browsing unit, and it's meant more for video editing than for browsing.

Zhang et. al [zhang94] used the video shot as their basic browsing unit. Similar to Tonomura, the frames of the shot are stacked to relay motion information and duration of the shot, and a frame from a shot may be "picked up" by placing the mouse along the side of the icon. In another mode, rather than stacking the frames, the icon thickness is used to convey shot duration; a wasteful use of screen space since the importance of the information does not justify the amount of screen space that is used.

Mills et. al [mills92] introduced a browser for quicktime video sequences. Similar to Tonomura's rush browser, this browser does not take into consideration the contents of the video and rather systematically divides the sequence into several equal segments. Once the user has chosen a segment it in turn is divided into equal lengths and so on until the user can view each frame. In each case, the segment is represented using its first frame. This approach is a minor improvement to FF/REW and fails to provide the user with a sense of the contents of the video. The user could easily miss the information he or she is interested in because the representation of each segment has no relation to the reminder of the frames in that segment.

The common disadvantages of the above work are that either no basic browsing unit is used and/or that each frame of the video is needed by the user during the browsing operations making it unsuitable for use over the network. Additionally, none of the above systems address the problem of icon management. This is very important since as many as several'thousand icons could be needed to represent the shots for each two hour video sequence. Ueda et. al [ueda93] do address this issue by using color information. Color, however, can not be the sole means of representation because color histograms are a many to one mapping functions. In our video browser, shape, as well as color information is used to help the user manage icons and navigate throughout a given video sequence.

3 USAGE

As mentioned earlier, video sequences require a "basic browsing unit" which can be used in browsing, and unlike newspapers or books where an editor manually chooses the headline for each article or chapter, the process of choosing the video browsing unit must be automatic. This is because of the vast amount of data that will exist in the video sequences. Furthermore, manual intervention would inherently incorporate extrinsic influences into the material. This influence could in turn impede users' search by providing false leads or not enough leads, forcing the user to use FF/REW. While the process of choosing the video browsing unit must be automatic, its result must also be meaningful to the user because this is the tool used to decide whether the returned video sequences are relevant to the task at hand. The last issue in designing a video browser is its speed; the video browser must be significantly faster, as compared with FF/REW, while remaining convenient to use.

We propose a video browser which will satisfy all the above set requirements. The proposed video browser uses shots as the basic building blocks of a video sequence characterized using "representative frames", or Rframes. The sequences in the video collection are pre-processed once to detect the scene changes and to build the Rframes. Then, to browse a particular video sequence, the user may scroll through all the Rframes to view the visual contents of the sequence. Once the user has chosen an Rframe, the corresponding video shot may be played back. Further information, such as the length of each shot and the approximate motions, are easily represented as well. In cases in which several hundred scenes, and therefore several hundred Rframes, may exist in a given video sequence, advanced techniques are used to allow the user to easily manage the information.

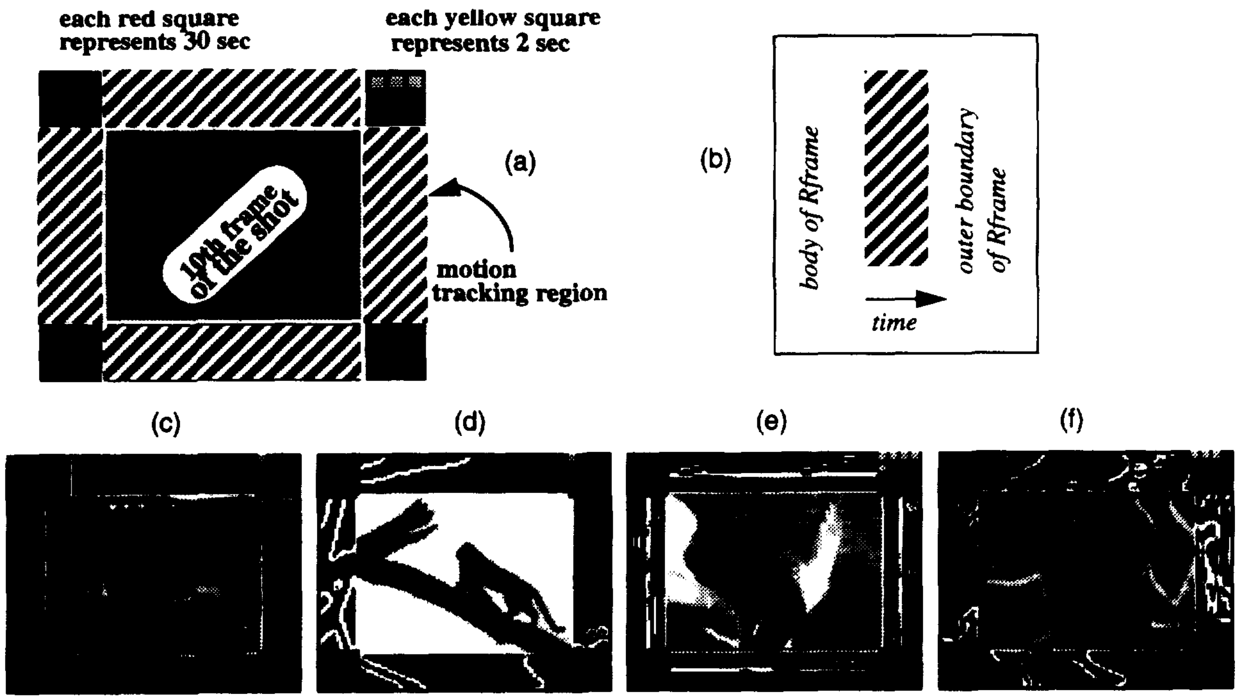

FIGURE 1: Representative frame, Rframe, for each video shot. (a) the structure of the Rframe, (b) motion tracking region; t=0 starts from the center of Rframe, (c)-(f) several examples: (c) the anchorman has moved his hands but the camera is stationary as evident by the straight lines, (d) the camera has panned to the left following the motion of the animal, the curves start (t=0) and move to the right, (e) example of a missed scene change, the straight lines not in contact with the center indicate the possibility that the shot may contain a scene change, (f) camera is stationary but the objects have moved in various directions.

3.1 Rframe

Each detected shot is represented using an Rframe, which is designed to allow the user to perform four tasks: First, to be able to judge the contents of the shot. Second, to decide if the scene change detection may have missed a shot. While many of the proposed scene change detectors have high accuracy rates of 90% and above, none claim 100% accuracy; in addition, many complicated transitions can cause false negatives during scene change detection. Therefore, from the user's point of view, there must be a mechanism to insure the user that no scene changes have been missed during this shot. The third task of the Rframe is to provide the user with the sense of motion within the shot. And, the last feature allows the user to easily determine the length of the shot in seconds.

Each Rframe consists of a body, four motion tracking regions, and shot length indicators (see Figure 1). The body of the Rframe is a frame chosen from the video shot; currently, the tenth frame is chosen, but other possibilities exist, such as the last frame for zoom-in shots. The four motion tracking regions trace the motion of boundary pixels through time; hence they can be used as guides to camera, or global, motion. The motion tracking regions mainly serve as an indicator of missed scene changes. In case the shot contains a scene change, there will be motion discontinuities along every pixel of each edge causing a straight line to appear in the motion tracking region (see Figure 1-e). The time indicators are designed so that a very quick glance at each Rframe allows the user to determine if the corresponding shot is long or short, more precise length of the shot is possible as well by counting the 2 and 30 second squares. This representation of shot length does not occupy any valuable screen space; printing the exact number of seconds, on the other hand, would not allow the user to quickly compare shot lengths.

To construct the motion tracking regions, the shot is subsampled to select a few of the frames. Four slices, one from each side, of each selected frame are then stacked and an edge detection algorithm is applied to each of the four stacks. This simple operation in effect tracks the border pixels from one from frame to the next enabling the user to visualize the motion.

3.2 User Interface

At start up the browser displays all the precomputed Rframes in the chronological order, (see Figure 2). The user may scroll through the Rframes and once an Rframe is chosen, then the video is played from precisely that point. User's second option is to choose one Rframe and view all other similar Rframes. The degree in which each Rframe in the sequence is similar to the chosen Rframe is conveyed to the user by varying the size of each Rframe. The most similar Rframes are displayed at their original scale, somewhat similar Rframes are displayed at smaller scale (default 33%), and the dissimilar Rframes are displayed at even a smaller scale (default 5%), see Figure 3. The defaults are easily adjustable by the user and may be used, for example, to see what is most similar only (see Figure 4), or to see all Rframes except the chosen one (i.e., show me the sequence without the anchorman).

As mentioned earlier, the browser must be as convenient to use as the current method of FF/REW. The proposed browser satisfies this criterion; the only user required actions are scroll and single or double clicks.

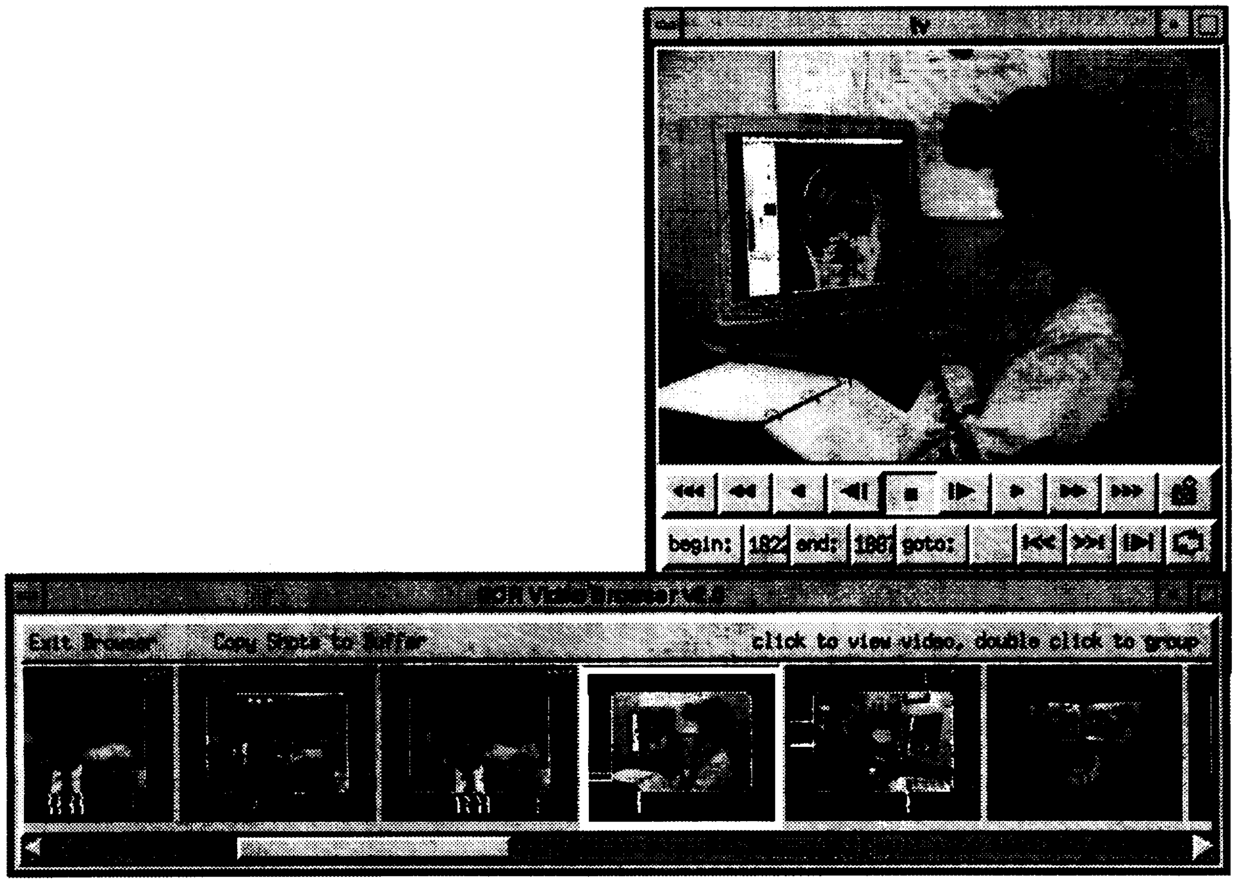
lv
begin: 182 end: 188 goto:
SCR Video Browser v2.0
Exit Browser
Copy Shots to Buffer
click to view video, double click to group

FIGURE 2: The browser in the basic mode of operation. The row of Rframes is on the bottom, and the sequence at the point chosen by the user is displayed on top. The user may play the video from that point and automatically stop at the end of the shot, or continue past the scene change.

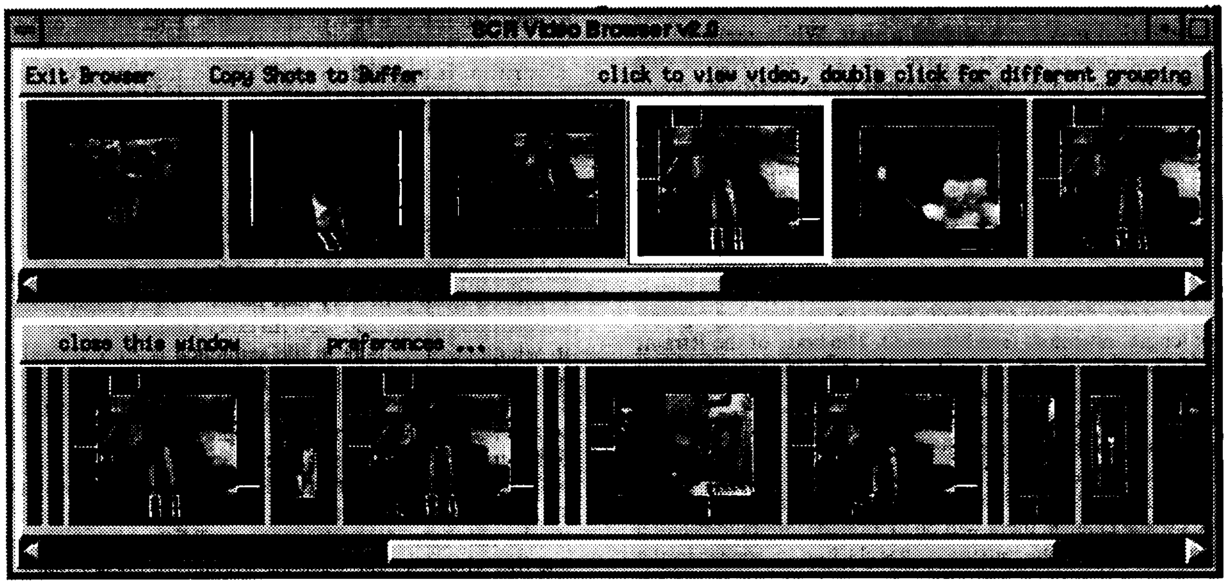

FIGURE 3: The browser in the advanced mode of operation. The top row is the original set of Rframes, the user has the chosen one Rframe (outlined by the red square) and the bottom row show all other similar Rframes, somewhat similar Rframes are shown at 33% of the original width, and non-similar Rframes are shown at 5% of the original width - scene as black bars.

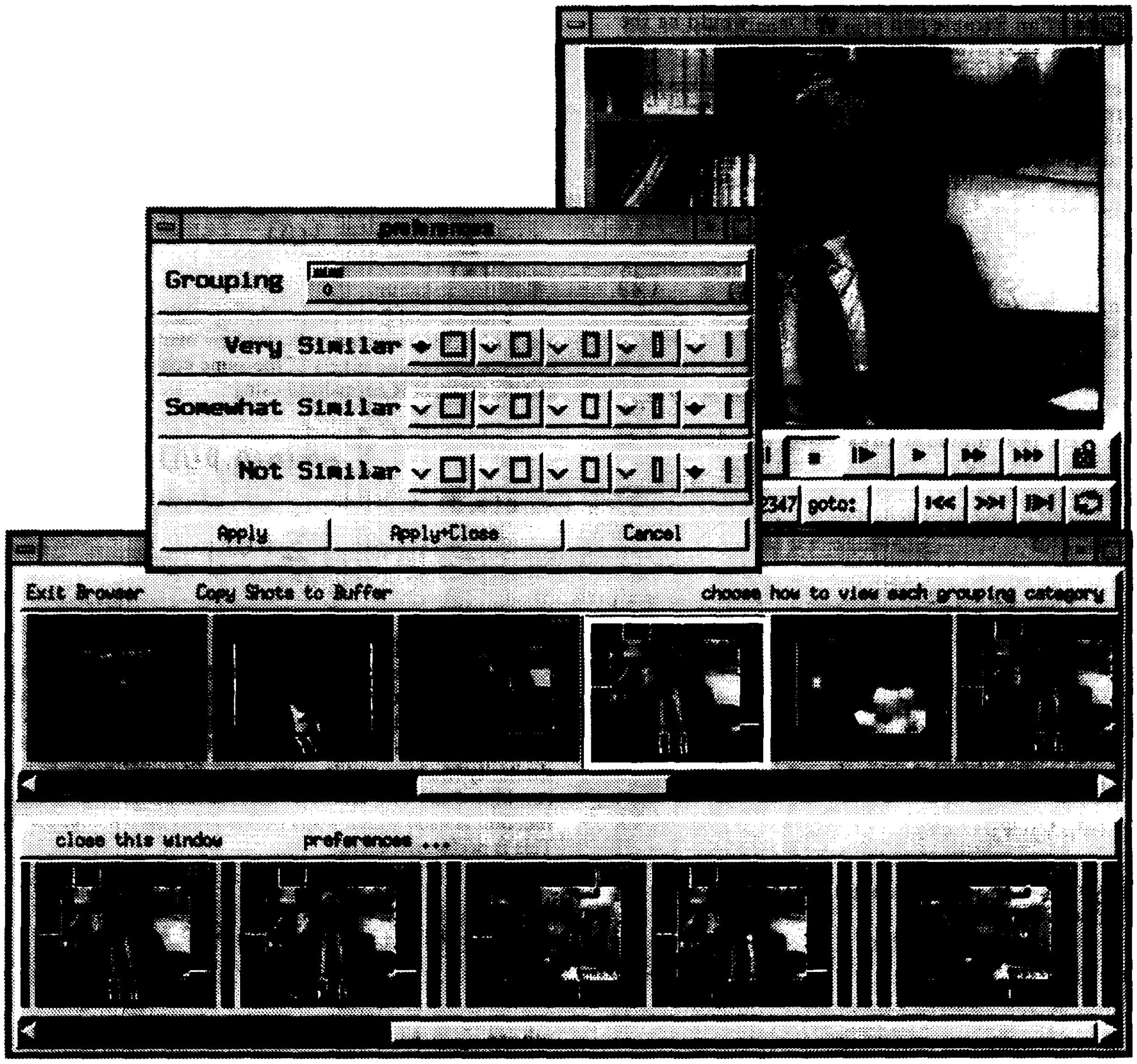

FIGURE 4: The browser in the advanced mode of operation as the user is choosing how to view each grouping category in the `preferences` *window. The shown setting indicates that the somewhat and not similar Rframes be shown as black bars, and only the similar Rframes be shown at full scale.*

4 RFRAME MANAGEMENT

Several issues arise when there are numerous Rframes - more than the user can easily search and navigate through. As mentioned earlier, the user may choose one Rframe and ask the system to return all similar Rframes in the same video sequence. The key to measure this similarity effectively and correctly is the means by which each Rframe is represented internally. Representations are used to describe Rframes, a key issue in the field of computer vision. The representations dictate the matching strategy, its robustness, and the system's efficiency. Also, the descriptions are used in the calculations of various properties of objects in the scene needed during the grouping stage. In almost all cases, the two-dimensional array of numbers used to display the Rframes is not of much use in its "raw" form.

The browser uses two representation schemes which compliment one another: Shape properties represented using moments and color properties represented using color histograms. Both representation schemes are insensitive to minor changes in the scene, such as object motion, viewing distance, etc., and both are compact representations allowing for efficient similarity measurements. The following two sections describe these representation schemes and their usage in more detail.

4.1 Shape

Shape of the objects within an Rframe is the main property used in Rframe management, and it is represented using moment invariants. The moment of an image *f(x,y)* is defined as:

$$m_{pq} = \sum\sum x^p y^q f(x, y) \qquad (1)$$

Physical interpretation of moments is possible if the grey level of each Rframe is regarded as its mass; then, m_{00} would be the total mass of an Rframe and m_{20} and m_{02} would be the moments of inertia around the x and y axes.

Moments invariants exhibit characteristics which makes them an ideal representation mechanism in the video browser. Invariance to scale change, rotation and translation are some of these characteristics which are used in the browser to describe Rframes. Moment invariants are derived from normalized central moments defined as:

$$\eta_{pq} = \frac{1}{m_{00}^{\gamma}} \sum\sum (x-\bar{x})^p (y-\bar{y})^q f(x,y) \qquad (2)$$

where $\gamma = \left(\frac{p+q}{2}+1\right)$, $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$. Then, the first few moment invariants are defined as [hu61][hu62][gonzalez77][1]:

$$\varphi_1 = \eta_{20} + \eta_{02}$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \qquad (3)$$

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

The shape of each Rframe is then represented using the vector $\vec{\sigma}$ defined as:

$$\vec{\sigma} = \{\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_7\} \qquad (4)$$

Finally, the euclidean distance is used to measure the similarity of two Rframes:

$$\psi(\alpha, \beta) = \left|\vec{\sigma_\alpha} - \vec{\sigma_\beta}\right|^2 \qquad (5)$$

4.2 Color

Color is the second feature used extensively in Rframe management. Color has many of the characteristics of moments, such as the ability to simply represent, or describe each Rframe. Contrary to moments, however, it is less sensitive to differences, such as due to motion within a frame. Color can not be the sole representation of Rframe contents because most means of representing color rely on color histograms which by definition are a many to one mapping functions. Hence, many completely different Rframes, or video frames, may have very similar color representations. Color histograms alone are not sufficient to detect any differences in a red and white checkered board versus a white board with red parallel lines, for example, since the color contents of the two are identical.

The browser represents the color contents of each Rframe using the color histogram, which is essentially the frequency distribution function of the color of each pixel. Given a color model (RGB, HSI, etc.), the histogram is obtained by counting how many times each color appears in each Rframe (see [novak92] for more details). As in our earlier work [arman94], we use the hue and saturation components of the HSI color space to calculate the color histogram for each Rframe.

In order to measure the similarity of two given Rframes, we borrow the technique of *histogram intersection* from Swain and Ballard [swain91]. The intersection of two histograms is defined as:

$$\sum_{j=1}^{n} min(\alpha(j), \beta(j)) \qquad (6)$$

where α and β are the two histograms each with n bins. The result of this intersection indicates how many pixels in one image have corresponding pixels of the same color in the other image, and the measure is normalized using:

$$\varepsilon(\alpha, \beta) = \frac{\left(\sum_{j=1}^{n} min(\alpha(j), \beta(j))\right)}{\left(\sum_{j=1}^{n} \beta(j)\right)} \qquad (7)$$

4.3 Combining Properties

Once the user has chosen an Rframe, the moments and the color histogram of that Rframe are compared to the remaining Rframes. The output of the moment-based and color histogram-based analyses are two floating point numbers describing the similarity in shape and in color of the Rframes' body. In order to combine and compare these two different entities a mapping function is used which maps both entities onto a common space. This is performed using:

$$\Omega(\zeta) = \begin{cases} 3 & \text{if} \quad \zeta < \tau_1 \\ 2 & \text{if} \quad \tau_1 \le \zeta \le \tau_2 \\ 1 & \text{if} \quad \zeta > \tau_2 \end{cases} \qquad (8)$$

where $\zeta = \varepsilon(\alpha, \beta_i)$ for mapping of color histogram intersection output of Equation (7):

$$\Omega_{\text{histogram}}[\varepsilon(\alpha, \beta_i)] \in \{1, 2, 3\} \qquad (9)$$

and $\zeta = \psi(\alpha, \beta_i)$ for mapping moment distance measure of Equation (5):

$$\Omega_{\text{moment}}[\psi(\alpha, \beta_i)] \in \{1, 2, 3\}. \qquad (10)$$

Ω=3 signifies very similar, Ω=2 somewhat similar, and Ω=1 not similar.

1. See also [reiss91] for a revised version of Hu's theory on fundamental theorem on moment invariants.

The rules of Table 1 are then used to combine the mapped properties. Generally, the output of Equation (10) is considered more important: If $\Omega_{moment}=1$ (not similar), then the output of color-histogram-based analysis is ignored; i.e., the final output will always be that the two Rframes under examination are not similar. If $\Omega_{moment}=3$ then the final output is also very similar, the only exception is when color-based output $\Omega_{histogram}=1$ in which case the final output will also be 2, or somewhat similar. The mapping from color histogram is used when Ω_{moment} is not conclusive; i.e., $\Omega_{moment}=2$; in this case the final output is set to the value of the color histogram mapping.

moment	color	final
3	3	3
3	2	3
3	1	2
2	3	3
2	2	2
2	1	1
1	3	1
1	2	1
1	1	1

TABLE 1: The rules for combining the results of the moment-based and histogram-based matching: 3=very similar, 2=somewhat similar, and 1=not similar.

The processing time for the grouping takes advantage of two points. First, the moments and the histograms are calculated a priori and the only step needed at run time is measuring similarity; i.e., Equation (5) and Equation (7). Second, using the rules specified in Table 1, the histogram intersection operation, the more expensive of the two, has to be performed on a subset of the Rframes providing additional time saving steps.[2] The future plans include an indexing scheme to store the histogram and the moment calculations; this will greatly speedup the grouping time.

5 CONCLUSIONS

This paper presents an efficient browser to represent the contents of a given sequence. Abstractions of each video shot are derived *a priori* which are then presented to the user instead of every frame as is the case with fast forward and rewind. This allows more efficient and accurate searches of the sequence, a faster comprehension of the contents, and it does not require that every frame be retrieved from storage and transmitted over the network, therefore saving time, expenses, and bandwidth. Each shot of the video is represented using Rframes which allow the user to comprehend the contents of each shot, the rough motion during the shot, the shot's length, and whether the scene change detection may have missed a video cut. The browser further allows users to navigate the information by providing techniques which automatically match Rframes and decide on their similarity level.

2. In practice, the time has been less than one second for every minute of video but it depends greatly on the video contents and the chosen Rframe.

6 REFERENCES

[arman93e] F. Arman, A. Hsu, and M-Y. Chiu, Image Processing on Compressed Data for Large Video Databases, in *Proceedings of First ACM International Conference on Multimedia*, Anaheim, CA, 1-6 August, 1993, pp 267-272.

[arman94] F. Arman, A. Hsu and M-Y. Chiu, Image Processing on Encoded Video Sequences, in *Multimedia Systems Journal*, vol 1, no. 5, 1994, pp 211-219.

[elliot93] E. Elliott, Watch, Grab, Arrange, See: Thinking with Motion Images via Streams and Collages, Ph.D. Thesis, MIT, Feb 1993.

[gonzalez77] R. Gonzalez and P. Witz, *Digital Image Processing*, Addison-Wesley, Readings, MA, 1977.

[hu61] M.-K. Hu, Pattern Recognition by moment invariants, in *Proc. IRE*, vol. 49, 1961, p 1428.

[hu62] M.-K. Hu, Visual pattern recognition by moment invariants, in *IRE Trans. Inform. Theory*, vol. 8, Feb 1962, pp 179-187.

[miller93] G. Miller, G. Baber and M. Gilliland, News On-demand for Multimedia Networks, in proceedings *of ACM International Conference on Multimedia*, Anaheim, CA, 1-6 August, 1993, pp 383-392.

[mills92] M. Mills, J. Cohen and Y-Y. Wong, A Magnifier Tool for Video Data, in *Proceedings of ACM Computer Human Interface (CHI)*, May 3-7, 1992.

[novak92] C. L. Novak and S. A. Shafer, Anatomy of a Color Histogram, in *Proceeding of Computer Vision and Pattern Recognition*, Champaign, IL, June, 1992, pp 599-605.

[reiss91] T. H. Reiss, The revised fundamental theorem of moment invariants, in *IEEE transactions on Pattern analysis and machine intelligence*, vol. 13, no. 8, Aug. 1991, pp 830-834.

[swain91] Swain, M. J. and Ballard, D. H., Color Indexing, in *Int. J. of Computer Vision*, vol. 7, no. 1, 1991, pp 11-32.

[tonomura90] Tonomura, Y. and Abe, S., Content Oriented Visual Interface Using Video Icons for Visual Database Systems, in *Journal of Visual Languages and Computing*, vol. 1, 1990, pp 183-198.

[tonomura93] Y. Tonomura, A. Akutsu, K. Otsuji and T. Sadakata, VideoMAP and VideoSpaceIcon: Tools for Anatomizing Video Content, in *InterCHI'93 Conference Proceedings*, Amsterdam, The Netherlands, 24-29 April, 1993, pp 131-136.

[udea91] Udea, H., Miyatake, T. and Yoshizawa, S., IMPACT: An Interactive Natural-motion-picture Dedicated Multimedia Authoring System, in *Proceedings of Human Factors in Computing Systems (CHI 91)*, New Orleans, Louisiana, April 27-May 2, 1991, pp 343-350.

[ueda93] H. Ueda, T. Miyatake, S. Sumino and A. Nagasaka, Automatic Structure Visualization for Video Editing, in *InterCHI'93 Conference Proceedings*, Amsterdam, The Netherlands, 24-29 April, 1993, pp 137-141.

[zhang94] H-J. Zhang and W. Smoliar, Developing Power Tools for Video Indexing and Retrieval, in *proceedings of SPIE conference on Storage and Retrieval for Image and Video Databases*, San Jose, CA, 1994.