A New Hybrid Approach to Video Organization for Content-Based Indexing [*]

Madirakshi Das
Department of Computer Science
University of Massachusetts
Amherst, MA
e-mail : mdas@cs.umass.edu

Shih-Ping Liou
Multimedia and Video Technology
Siemens Corporate Research
Princeton, NJ
liou@scr.siemens.com

Abstract

Video organization is a key step in the content-based indexing of video archives. The objective of video organization is to capture the semantic structure of a video in a form which is meaningful to the user. We present a hybrid approach to video organization which automatically processes video, creating a video table of contents (VTOC), while providing easy-to-use interfaces for verification, correction and augmentation of the automatically extracted video structure. Algorithms are developed to solve the subproblems of shot detection, shot grouping and VTOC generation without making very restrictive assumptions about the structure or content of the video. We use a nonstationary time series model of difference metrics for shot boundary detection, color and edge similarities for shot grouping and observations about the structure of a wide class of videos for the generation of the VTOC. The use of automatic processing in conjunction with input from the user allows us to produce meaningful video organization efficiently.

1. Introduction

For a multimedia information system to better meet the users' needs, it must capture the semantics and terminology of specific user domains and allow users to retrieve information according to such semantics. This requires the development of a content-based indexing mechanism that is rich in its semantic capabilities for abstraction of multimedia information and also able to provide canonical representation of complex scenes in terms of objects and their spatio-temporal behavior. A key initial stage in this content-based indexing process is *video organization*. The objective of the video organization is to capture the structure present in the video, providing a *video table of contents* analogous to the table of contents in a book.

[*]This material is based on work done at Siemens Corporate Research, Princeton, NJ.

Obtaining a video table of contents to a reasonable degree of accuracy automatically, is a complex process. First, video frames must be segmented into a set of units called *shots*. A shot in a video refers to contiguous frames depicting continuous action in time and space. Repeating shots are then identified to detect parallel events in a scene and this information is finally used to construct the video table of contents. Since the generation of the table of contents consists a sequence of dependent steps, even one mistake could create completely useless results.

There have been two different approaches to video organization. The research in database systems has mostly focussed on attribute-based indexing of multimedia information which entails a level of abstraction that reduces the scope for posing ad hoc queries to the database [4]. The research in computer vision, offers an alternative approach that relies on an integrated feature extraction/object recognition subsystem [16, 18, 13] to segment the video into meaningful semantic units.

Both approaches to video organization have their own limitations. The attribute-based approach needs a human operator to manually index the multimedia information. On the other hand, the automatic approach is computationally expensive, difficult, and tends to be very domain specific. It is nearly impossible, in practice, to obtain useful video organization based solely on automatic processing.

What we need is a hybrid approach, i.e., automatically segmenting video and creating the video table of contents in a preprocessing step, while providing an easy-to-use interface for verification, correction and augmentation of the automatically extracted video structure. This paper describes such a hybrid approach that includes three automatic indexing algorithms closely integrated with three graphic user interface objects. The automatic indexing algorithms include cut detection, shot grouping, and video table of contents creation, whereas the graphic user interface objects consist of the video player, browser, and table of contents viewer.

This paper is organized as follows. Section 2 reviews the past literature and provides the motivation for this work. A brief summary of the novel aspects of the proposed method is given in section 3. Section 4 describes the algorithms used in automatic generation of the table of contents from the raw video. Section 5 describes the interactive aspects of video organization, including the user interfaces. Finally, it concludes with comments on future directions in section 6.

2. Literature review

There has been work in extracting the semantic structure of the video using strong domain knowledge. Zhang et al [18] use known templates of anchor person shots to separate news stories. Swanberg et al [13] use the known structure of news programs in addition to models of anchor person shots to parse news videos. The presence of the channel logo, skin tones of the anchor person and the structure of a news episode have been used in [6]. These approaches only create a hierarchy with a few fixed number of levels. Content-based indexing at the level of shots using motion is described in [3], without developing a high-level description of the video. Although domain knowledge, in general, constrains the problem and makes it possible to provide a reliable solution, it can never account for all possible scenarios even for a simple domain such as news video. For example, it is not possible to define an anchor person image model that is independent of broadcast stations.

Recently, Yeung and Yeo [15] presented a domain-independent approach that extracts story units (the top level in a hierarchy) for video browsing applications and creates a scene transition graph (Fig 3). Such a graph leads to a compact representation that serves as a summary of the story and may also provide useful information for automatic classification of video types. However, this representation reveals little information about the semantic structure *within* a story unit e.g. an entire news broadcast is classified as a single story, making it difficult to browse through individual news stories. The clustering strategy proposed in this work uses temporal constraints, making it difficult to cluster similar shots which are temporally far apart e.g. the anchor-person shots in a news broadcast are usually scattered throughout the video.

The problem of capturing the semantic structure in a video also requires solution to both the cut detection and the shot grouping problems. Most existing cut detection algorithms are based on preset thresholds or assumptions that reduce their applicability to a limited range of video types [2, 7, 10]. For example, it is often assumed that both the incoming and outgoing shots are static scenes and the transition only lasts for a period less than half a second. This type of model is too simple for modeling gradual shot transitions that are often present in films/videos. Another frequent assumption is that the frame difference signal computed at each individual pixel can be modeled by a stationary independent identically distributed random variable which obeys a known probability distribution [17]. This assumption is generally not true as shown in Figure 1. Neither a Gaussian nor a Laplace distribution fits both curves well. A Gamma function fits the curve on the left, but not the one on the right. In addition, existing methods assume that time-series difference metrics are stationary, when actually, such metrics are highly correlated time signals.

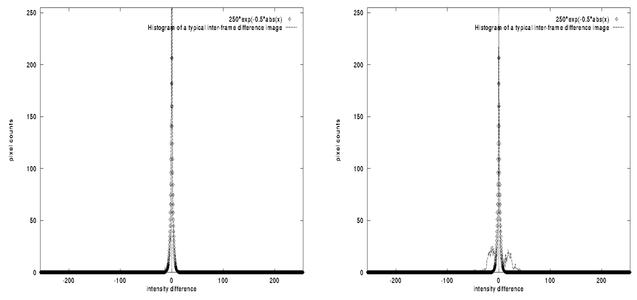


Figure 1. The histogram of a typical inter-frame difference that does not correspond to a shot change. The shape of the curve changes as the camera moves slowly (left) versus fast (right).

Since many videos are converted from films and the two media are played at different frame rates, creating videos from films requires making every other film frame a little bit longer. This process results in video frames that are made up of two fields with totally different (although consecutive) pictures in them. As a result, the digitization produces duplicate video frames and almost zero inter-frame differences at five frame intervals. A similar problem occurs in animated videos where almost zero inter-frame difference is produced in as often as every other frame.

In grouping visually similar shots, most existing approaches use color histograms [12]. However, the commonly used RGB and HSV color spaces are sensitive to the illumination to varying degrees and uniform quantization of the color space is against the principles of human perception [14].

3. Our approach

Our approach to the video organization problem is illustrated in Figure 2. This hybrid approach consists of a set of automatic video organization algorithms and a collection of interactive video organization interfaces. Our approach differs from existing approaches in the following aspects.

First, none of the existing algorithms provide manual feedback mechanisms during the automatic creation of video structure. Mistakes made by either automatic cut detection and/or shot grouping algorithms will ruin the final

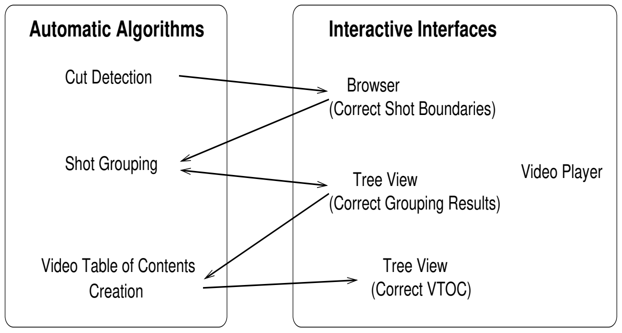


Figure 2. Our hybrid approach to video organization

video structure produced by the algorithm. Therefore, it is important to provide interfaces so that a human operator can verify and correct the results produced automatically at every step of the processing.

Second, we use the observation that *repeating shots which are similar in some way, alternating or interleaving with other shots, are often used to convey parallel events in a scene or to signal the beginning of a semantically meaningful unit.* Since we do not use pre-defined models for similar shots, this observation can be used for a wide variety of videos for which organization is particularly relevant e.g. news, sports events, interviews, documentaries etc. News and documentaries have the anchor-person appearing before each new segment of the story to introduce it. Interviews have the interviewer appearing to ask each new question. Sports events have sports action between shots of the stadium or the commentator. So, it is possible to create a forest structure directly from the list of identified recurring shots. This forest structure (video table of contents) preserves the time order among shots. It also captures the syntactic structure of the video which is a hierarchy composed of stories, and, under stories, sub-plots, which may have further subplots embedded in them. For most structured videos, it provides interesting insights into the semantic context of the video and is therefore a more useful representation than the scene transition graph. An example is shown in Fig 3.

Third, we use a name-based color system, **ISCC-NBS** [9], to describe the color of images during the shot grouping process. The NBS color system divides the Munsell color space into irregularly shaped regions and assigns a color name to each region based on common usage. This enables the use of color histograms with perceptually based color space quantization and allows us to construct a color description independent of the illumination of the image.

Finally, unlike existing cut detection methods which have no notion of *time series* and *non-stationarity*, we treat a sequence of difference metrics as *nonstationary* time series signals and model the time trend deterministically. The sequence of difference metrics are just like any economic

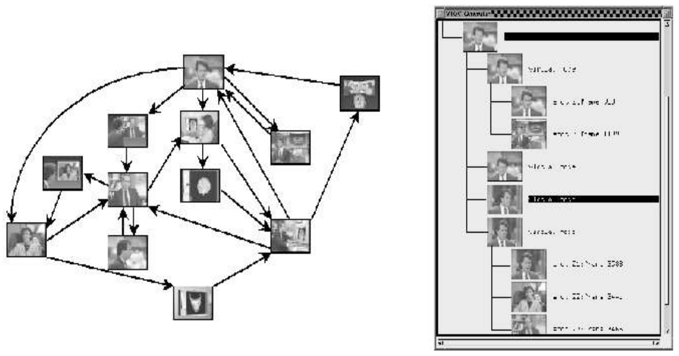


Figure 3. Scene transition graph (left) and video table of contents (right) of the same video clip

or statistical data collected over time. In this view, shot changes as well as the film-to-video conversion process will both create *observation outliers* in time series, while the gradual shot transition and gradual camera moves will produce *innovation* outliers. Fox [5] defines the observation outlier to be the one that is caused by a gross error of observation or recording error and it only affects a *single* observation. Similarly, the innovation outlier is one that corresponds to the situation in which a single "innovation" is extreme. This type of outlier affects not only the particular observation but also subsequent observations.

4. Automatic organization of video

We have developed a set of automatic algorithms that can produce an organized structure from a raw video. The first task in the automatic organization of video is to recover the shots present in the video. We have developed a scene change detection algorithm which provides good shot detection in the presence of outliers and difficult shot boundaries like fades and zooms. Each shot is completely defined by a start and end frame and a list of all shots in a video is stored in a *shotlist*. Each shot is represented by a single frame, the *representative frame*, which is stored as an image (icon).

The more complex task is to organize the shots into a higher level structure reflecting the semantics of the video. We have automated the process of inferring the semantics of the video using the grouping of shots by similarity and the observation made in the earlier section about the relevance of repetition of similar shots. In the following sub-sections, we will describe the cut detection, shot grouping and video table of contents creation algorithms respectively.

4.1. Cut detection

Pixel-based metrics (e.g. inter-frame difference) and distribution-based difference metrics (e.g. statistics) respond differently to different types of shots and shot transitions. For example, the former are very sensitive to camera moves but are a good indicator for shot changes; the latter are relatively insensitive to camera and object motion, but can produce little response when two shots look quite different but have similar distributions. We feel that it is necessary to combine both measures in cut detection.

We model the sequence of difference metrics as nonstationary time series signals. There are standard methods [1, 5, 8] that detect both innovation and observation outliers based on the estimate of time trend and autoregressive coefficients. These standard methods, however, cannot be applied to the cut detection problem directly because of the following three reasons. First, most methods require intensive computation (e.g. least squares) to estimate time trend and autoregressive coefficients. Second, the observation outliers created by slow motion and the film-to-video conversion process could occur as often as one in every other sample, making the time trend and autoregressive coefficient estimation an extremely difficult process. Finally, since gradual shot transitions and gradual camera moves are indistinguishable in most cases, location of gradual shot transitions requires not only the detection of innovation outliers but also an extra camera motion estimation step.

In our solution, we use a zero*th*-order autoregressive model and a piecewise-linear function to model the time trend. With this simplification, samples from both the past and the future must be used in order to improve the robustness of time trend estimation. We also need to discard more than half the samples because the observation outliers created by slow motion and the film-to-video conversion process could occur as often as one in every other sample. Fortunately, these types of observation outliers are least in value, and hence easily identifiable. After removing the time trend, the remaining value is tested against a normal distribution $N(0, \sigma)$ in which σ can be estimated recursively or in advance.

To make the cut detection method more robust, we apply the Kolmogorov-Smirnov test to eliminate false positives. This test is chosen because it does not assume *a priori* knowledge of the underlying distribution function. The traditional Kolmogorov-Smirnov test procedure compares the computed test metric with a preset significance level (normally at 95%), and has been used [11] to detect cuts from videos. This use of single pre-selected significance level ignores the non-stationary nature of the cut detection problem. We feel that the use of Kolmogorov-Smirnov test should take into account the non-stationary nature of the problem i.e. the significance level should be *automatically adjusted* to different types of video contents.

One way to represent video content is to use measurement in both the spatial and the temporal domain together. For example, image contrast is a good spatial domain measurement and the amount of intensity changes across two neighboring frames measures video content in the temporal domain. The adjustment should be made such that,

• the higher the image contrast is, the more sensitive the cut detection mechanism should be, and

• the more changes occur in two consecutive images, the less sensitive the detection mechanism should be.

The traditional Kolmogorov-Smirnov test also cannot differentiate the long shot from the close up of the same scene. To guard against such transitions, we propose a *hierarchical* Kolmogorov-Smirnov test. In this test, each frame is divided into four rectangular regions of equal size and the traditional Kolmogorov-Smirnov test is applied to every pair of regions as well as to the entire image. This test therefore produces five binary numbers that indicate whether there is a change in the entire image as well as in each of the four sub-images.

Finally, instead of directly using these five binary numbers to eliminate false positives, the significance of the test result of a shot change frame is compared against that of its neighboring frames.

4.2. Shot grouping

Similarity between shots is determined by comparing the *representative frame* images for the shots. We have used color to cluster the shots into initial groups, and then used edge information within each group to refine the clustering results.

4.2.1. Using color:

The use of color in computing image similarity has found wide use in image retrieval systems, color histograms [12] being especially popular. The ability of the color histogram to detect similarity in the presence of illumination variations is greatly affected by the color space used as well as how the color space is quantized.

In our solution, we have selected a name-based color system, **ISCC-NBS** [9], which is constructed from the human perception of color. Since the color names are based on common usage, the results are more likely to agree with the user's perception of color similarity and natural language user interfaces can be developed in the future.

Each color name in the NBS system has two components : the hue name and the hue modifier. Fig 4 shows a list of hue names and Fig 5 shows the hue modifiers used in the NBS system. e.g. 'very deep purplish blue' is a possible color name. However, all combinations of hue name and modifiers are not valid - there are a total of 267 valid color names, obtained by dividing the Munsell color space into irregularly shaped regions.

Since we would like to keep the description of an image independent of the illumination, we use only the hue names

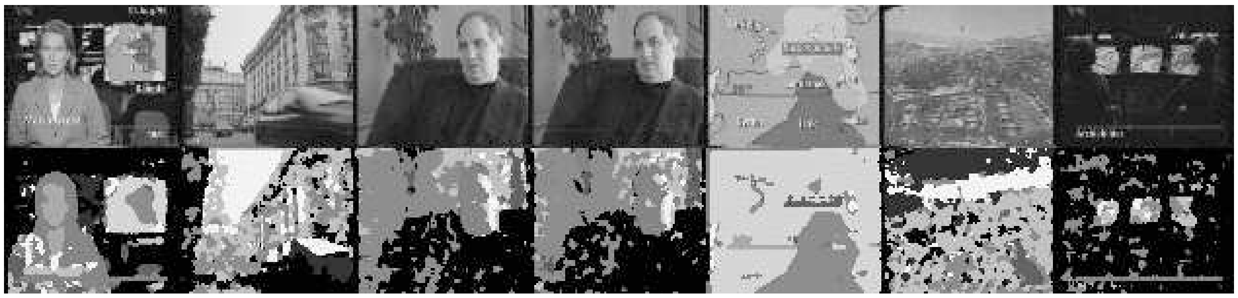


Figure 6. (top row) The original images (bottom row) Images labeled using 14 colors

red	reddish orange	reddish purple
reddish brown	green	bluish green
purplish red	brown	greenish blue
purplish pink	yellow green	orange
orange yellow	blue	yellowish brown
yellow	purplish blue	yellowish pink
olive brown	pink	greenish yellow
yellowish green	violet	brownish pink
olive	purple	brownish orange

Figure 4. ISCC-NBS hue names

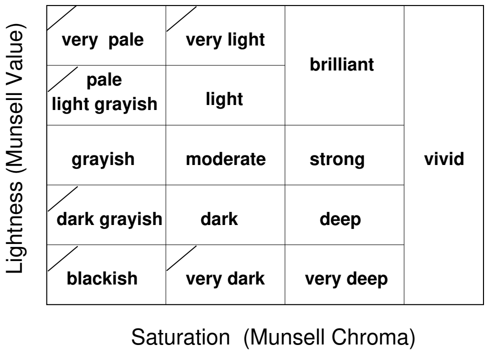


Figure 5. ISCC-NBS hue modifiers

instead of the full color names. In our experiments with this color system, we have observed that replacing the color names containing the modifiers specially marked in Fig 5 by 'black' or 'white' results in better classification of the color. e.g. 'very pale green' is in fact closer to white and 'very dark green' is closer to black than green. We have further reduced the number of colors to 14, by merging some of the colors into their more dominant component e.g. 'reddish orange' is considered to be 'orange'. Fig 6 shows images labelled using 14 colors. Reducing the number of colors improves the chances of two similar images being clustered in the same group. Fig 7 shows the histograms obtained from two images of a soccer match. There is variation in the green color of the grass. Using 14 colors, all types of green are labelled 'green' (color # 2) and therefore, the histograms are very similar. However, when all hue names are used, the green label is divided into 'olive green', 'yellowish green', 'bluish green' etc. The histograms appear different now since the proportions of the different shades of green are not the same in both images.

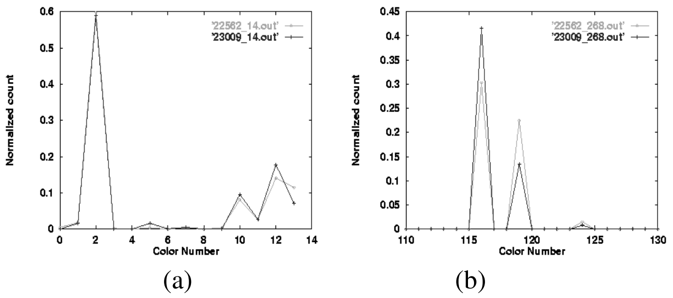


Figure 7. Effect of number of colors on similarity: (a) 14 colors (b) All hue names

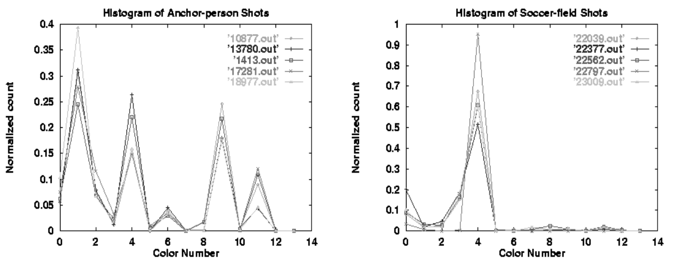


Figure 8. Color histograms of similar images

The normalized histogram bin counts are used as the feature vector to describe the color content of an image. Fig 8 shows the color histograms of the shots grouped together on the basis of similar color distributions.

4.2.2. Using edges:

When images are grouped only on the basis of their color histograms, visually different images with similar color distribution may be grouped together. Edge information is used as a filter to remove shots which have incorrectly been grouped using color alone.

We use a straightforward global measure of the edge information in an image. Each edge pixel is classified as belonging to one of four directions based on the sign and relative magnitude of its response to edge operators along x and y directions. The histogram of pixel counts along each of the four directions is used as a feature vector to describe the edge information in the image. The gross edge information is computed after simplifying the image by quantizing it to a few levels (4 or 8) and converting the quantized image to an intensity image. This edge information is sufficient to filter out most of the false shots in a group.

4.2.3. Clustering strategy:

We are constrained in the choice of clustering strategy by having no a priori knowledge of the number or the nature of the clusters. We cannot make the assumption that similar images will be temporally close to each other in the video, since the repeating shots are likely to be scattered throughout the video. Therefore, clustering strategies which involve comparisons among all points in a limited window [15] are not suitable in our case. We also do not know the number of potential clusters a priori, so K-means clustering and other strategies using this a priori information are also not useful. Moreover, it would be advantageous if the clustering strategy was not offline i.e. did not require all the shots to be present before starting. This would allow us to process shots as they are generated.

The clustering algorithm we have used is based on nearest neighbor classification, combined with a threshold criterion. The initial clusters are generated based on the *color* feature vector of the shots. Each cluster is specified by a feature vector which is the *mean* of the vectors of its members. When a new shot is available, the city block distance between its color feature vector and the means of the existing clusters is computed. The new shot is grouped into the cluster with the minimum distance from its feature vector, provided the minimum distance is less than a threshold. If an existing cluster is found for the new shot, the mean of the cluster is updated to include the feature vector of the new shot. Otherwise, a new cluster is created with the feature vector of the new shot as its mean. The threshold is selected based on the percentage of the image pixels that need to match in color, in order to call two images similar.

Shots are deleted from the clusters produced above if the distance of their *edge* feature vector from the mean edge vector of the group is greater than a threshold, starting with the shot furthest from the mean. The mean is recomputed each time a member is deleted from the cluster. This is continued till all the edge feature vectors of the members in the cluster are within the threshold from the mean of the cluster, or there is a single member left in the cluster. The threshold in this case is a multiple of the variance of the edge vectors of the cluster members. So, the final clusters are based on color as well as edge similarity, allowing the color feature to be the main criterion in determining the clusters.

A *mergelist* is produced by automatic clustering, identifying the group number for each shot in the shotlist. This information is used for the automatic construction of the video structure.

4.3. VTOC creation

We have found that a hierarchical tree structure captures the organization of the video adequately and is easy for the user to understand and work with. The whole video is the root node, which can have a number of child nodes, each corresponding to a separate ‘story’ in the video. A story is a self-contained unit which deals with a single or related subject(s). Sub-plots are different elements in a story unit or sub-plot unit. The tree structure has nodes of different types to provide semantic information about its contents. Each node also has a representative icon visible to allow browsing without having to unravel the full structure. Each new story starts with a *story node*, which consists of *sub-plot nodes* for each sub-plot. *Similar nodes* are used to bind together all *consecutive* frames found to be in the same cluster. Frequently, these nodes may be replaced by any one of its members by merging the other shots. The leaf nodes contain the shots from the *shotlist*.

The algorithm contains two major functions. A modified version of the algorithm presented in [15], finds all story units, creates a story node for each story and calls the second function *Find-structure* to find structure within each story. Each story unit extends to the last re-occurance of a shot within the body of the story. *Find-structure* is a function that takes a segment of shot indices, traverses through the segment to create a node for each shot until it finds one shot that reoccurs later. At this point, it divides the rest of the segment into sub-segments each of which is lead by the recurring shot as a sub-plot node and recursively calls itself to process each sub-segment. If consecutive shots are found to be similar, they are grouped under a single similar node. The structure produced by *Find-structure* is attached as a child of the story node for which it was called.

5. Interactive organization

We now describe the tools provided to the user to modify the results of automatic organization at both the shot and VTOC levels. The steps in the interactive generation of the final organized video can be summarized as follows :

1. Automatic construction of *shotlist* from raw video.

2. * Viewing and editing the shot structure to add new shots or merge shots.

3. Automatic clustering of shots into visually similar groups.

4. * Viewing and editing the clusters generated automatically to create new clusters and modify existing clusters.

5. Automatic generation of tree structure to describe the high level units in the video using the cluster information from the earlier step.

6. * Viewing and modifying the tree structure to reorganize the video.

The steps marked with an asterix in the above list need user interaction and therefore, an interface needs to be provided with the required functionality. Since these interfaces work with higher level representations of the video, a separate component is also provided to view the raw video accompanied by its audio. The interactive video organization environment provides three main interfaces to the user which communicate with each other so that changes made using one component produce the appropriate updates in the other interfaces.

- *Browser :* This interface is used to view and modify the shot list. The video stream is represented as a composite image making it easy to detect shot boundaries visually. The results of automatic shot detection are displayed alongside using colored bars, and can be altered easily.

- *TreeView :* This interface serves a dual purpose. Before the creation of the tree structure of the video, it is used to view and alter the automatic clustering of shots based on visual similarity. Once the similarity-based grouping results have been finalized, it is used as an interface to organize the video into a tree structure.

- *VideoPlayer :* This interface is used for playing the video, with audio, from any point in the video. It has the functionality of a VCR including fast forward, rewind, pause and step.

These components allow easy manipulation of the results obtained by automatic processing to fit the user's interpretation of the video. Fig 9 shows the components of this system along with the interactions between them. Fig 11 shows the graphical interface presented to the user by each component.

If the steps in the construction of the tree structure were followed sequentially, there would be very little interaction between the interfaces in the environment. However, it

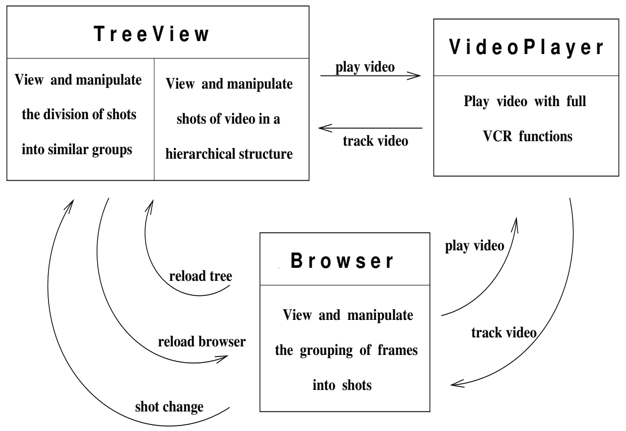

TreeView
View and manipulate the division of shots into similar groups
View and manipulate shots of video in a hierarchical structure
VideoPlayer
Play video with full VCR functions
play video
track video
Browser
View and manipulate the grouping of frames into shots
reload tree
reload browser
shot change
play video
track video

Figure 9. System Overview

would be very useful for the user to see the effect of changes made at one level propagate to the other levels and be able to move between levels. Our environment aims to provide a fully integrated system where change is reflected throughout the system, wherever possible.

The environment is designed so that each of the three interfaces can run individually or in combination with others. Each interface has the capability of starting up the other interfaces. Any one of them can be started first, provided the required files are present. The *shotlist* is needed to start the *Browser*. *TreeView* starts with the group structure if only the *mergelist* is present, otherwise it starts with the tree structure which is stored in the *treelist*. When more than one interfaces are running simultaneously, the issue of interactions has to be considered. Fig 9 shows the communications between the interfaces in this case. Each interface and its communications with the other interfaces is described in greater detail in the following sections.

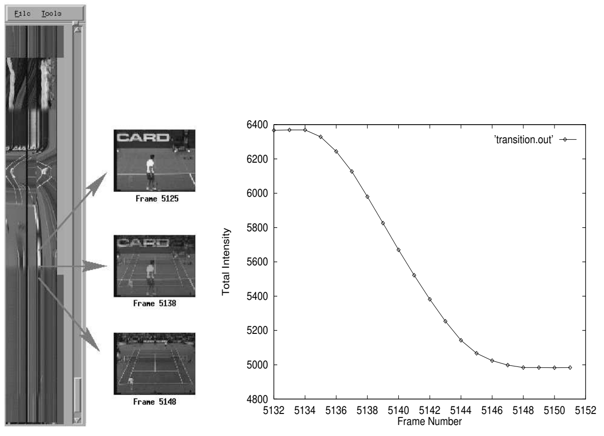
5.1. Browser

The browser is used to view and alter the grouping of frames into shots. The video is presented to the user in the form of a composite image which is constructed by including a horizontal and a vertical single pixel slice from the center line of each frame in the video along the time axis. This form makes it easier to check the shots produced automatically which are depicted visually by colored bars alongside the composite image as shown in Fig 11(d).

Automatic shot boundary detection may produce unsatisfactory results in the case of slow wipes from one shot to the next, momentary changes in illumination (flashes), zoom etc. Changes may be neccessary in the *shotlist* generated automatically to include additional shots or merge two shots. Any changes made in the shot boundaries changes the *shotlist* and the set of representative icon images. The following operations are implemented to make it simple to modify the shot boundaries.

• **Split** : A shot can be split into two shots by marking the point of split. A representative frame for the new shot is generated and the internal shot structure of the *Browser* is updated. This operation is used to incorporate shots which were not detected automatically.

• **Split Ahead** : When there is a gradual shot change, it may not be easy for the user to locate the point where the shot should be split to get a good representative icon for the new shot created. Using this operation, any point selected in the transition region produces a correct split. The point where the transition is completed is detected by processing the region following the point selected by the user.



(a) (b)

Figure 10. Split Ahead Operation: (a) Example icons (b) Corresponding intensity plot

As an example, the middle icon image in Fig 10(a) shows the frame selected by the user based on visual inspection of the shot boundary displayed in the browser. This image does not represent the next shot as it still contains an overlap from the earlier shot. The last icon image in Fig 10(a) shows the frame correctly picked by the *split ahead* operation in which the dissolve process is complete. This is achieved by following the gradient along the smoothed intensity plot [Fig 10(b)] till the gradient direction changes or the gradient becomes negligible.

• **Merge** : This operation is used to merge a shot into either of its adjoining shots. A shot to be merged is specified by selecting any frame within it. The icon representing the merged shot is deleted by this operation.

• **Play video** : The actual video can be played on the *VideoPlayer* from any selected frame. This playback may be needed to determine the content of the shots and detect subtle boundaries. While the video is playing, the *Browser* may track the video to keep the frame currently playing at the center of the viewing area.

• **Interactions** : The *Browser* can produce a change in the *shotlist*. This information is provided to the *TreeView* via a message and the change becomes visible immediately i.e. a new shot appears at the specified location or a shot gets deleted automatically. This helps the user to actually see the icons representing the shots that are being created or deleted. In fact, the visual information from the tree can be used to determine actions taken in the browser e.g. when two consecutive representative icons shown in the *TreeView* cover very similar subject matter, the user may choose to merge them using the *Browser*.

The user may opt to reload the *TreeView* interface using the new *shotlist* to redo the clustering, when there have been enough changes in the shotlist to make the earlier tree organization obsolete. The *Browser* can store the modified *shotlist* containing the changes made by the user and trigger the automatic clustering of shots in the *shotlist*, to produce a *mergelist* which is used by *TreeView*.

5.2. TreeView : correction of VTOC

The *TreeView* interface is used to interact with the tree description of the video generated from the *mergelist*. The user is allowed full freedom in restructuring the tree since semantic information can often be missed or misinterpreted by automatic processing. The following operations are provided to facilitate changes in the structure of the tree.

• **Move nodes** : Nodes can be moved either one at a time or in groups by selecting the node(s) to be moved and the destination node. The moved node(s) are added as siblings of the destination node selected.

• **Add nodes** : New leaf nodes can only be added through changes in the *shotlist* using *Browser*. However, all types of non-leaf nodes can be added as parents of existing nodes.

• **Update** : This operation is an utility to make it easier to move large number of nodes. It repeats the previous operation on all siblings of the node which was selected for the previous operation. For example, if a subplot node is to be deleted by moving all its children to another subplot node, just one child needs to be explicitly moved - the others can be forced to move by using the update operation.

• **Interactions** : When changes are made in the order of the shots using *TreeView* and the user wants to see these changes reflected in the *Browser*, (s)he can opt to send a signal to the *Browser* to reload the rearranged *shotlist* after saving it from *TreeView*. The *VideoPlayer* can be played from this interface exactly as in the *Browser* interface.

No explicit delete function is provided in this interface since leaf nodes can only be deleted through changes in the *shotlist* using the *Browser*. All other (non-leaf) nodes are deleted automatically when they have no children.

The user can invoke these operations to regroup the shots into more meaningful stories and sub-plots. The order of shots can also be changed from their usual temporal order to a more logical sequence. When used along with the *Browser*, all possible changes to the content and organization of the tree are supported.

The tree structure is stored as a *treelist* file so that organized videos can display the tree structure without going through the processing steps again. Modifications made by the user in the tree structure are also saved in the *treelist*.

5.3. TreeView : modifying shot groups

Though the primary function of the *TreeView* interface is to interact with the high-level structure, the same interface can also be used to view and modify the groups generated by the automatic clustering process. In this case, there are only two types of nodes attached to the root node. If the group contains a single member, the member shot is attached as a leaf node to the root. For groups containing more than one member, an intermediate *Group* node is attached, which contains the member shots as its children.

The tree operations described in the earlier section can be used to move shots out of existing groups or create new groups. A modified *mergelist* can be generated which reflects the changes made by the user. This step needs to be performed before the tree structure is loaded since the tree structure is constructed from the *mergelist*.

6. Conclusion and future work

We have proposed a hybrid video organization methodology which combines the benefits of automatic processing with human input. Algorithms for producing a table of contents automatically from the raw video have been developed by solving the sub-problems of cut detection, shot grouping and generation of table of contents. Automatic processing reduces the user's work from going through the whole process manually to just providing error checking and higher level semantics. We have constructed an environment which provides effective tools to the user to guide the video organization process. The system has been tested on news stories and sports videos and produces reasonable organization of the videos requiring only small alterations from the user.

An important direction of future work is to have the system utilize the feedback from the user in an intelligent way. When the user merges two groups which were found to be different at the color or edge level, this may be due to the fact that some parts of the images from both groups match. Partial match templates could be generated from this information. Instead, the similarity may be based on audio, which could be found by comparing the audio streams associated with the merged shots. Using the cues from the user will further reduce the work which is needed from the user to modify the index structure produced automatically.

References

[1] B. Abraham and A. Chuang. Outlier detection and time series modeling. *Technometrics*, 31(2):241–248, May 1989.

[2] P. Aigrain and P. Joly. The automatic real-time analysis of film editing and transition effects and its applications. *Computer and Graphics*, 18(1):93–103, 1994.

[3] F. Arman, R. Depommier, A. Hsu, and M. Y. Chiu. Content-based browsing of video sequences. *ACM Multimedia*, pages 97–103, Aug 1994.

[4] P. England, R. B. Allen, S. M, and H. A. I/browse: The bellcore video library toolkit. *Storage and Retrieval for Still Image and Video Databases, SPIE*, IV:254–264, Feb 1996.

[5] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, Series B(34):350–363, 1972.

[6] B. Gunsel, A. M. Ferman, and A. M. Tekalp. Video indexing through integration of syntactic and semantic features. *IEEE Multimedia Systems*, pages 90–95, 1996.

[7] H. Hampapur, R. Jain, and T. Weymouth. Digital video segmentation. *Proc. of ACM Multimedia Conference*, pages 357–363, 1994.

[8] L. K. Hotta and M. M. C. Neves. A brief review of tests for detection of time series outliers. *ESTADISTICA*, 44(142):103–148, 1992.

[9] K. L. Kelly and D. B. Judd. The iscc-nbs method of designating colors and a dictionary of color names. *National Bureau of Standards Circular*, (553), Nov 1 1955.

[10] J. Meng, Y. Juan, and S. F. Chang. Scene change detection in a mpeg compressed video sequence. *Digital Video Compression Algorithms and Technologies, SPIE*, pages 14–25, Feb 1995.

[11] I. K. Sethi and N. Patel. A statistical approach to scene change detection. *Storage and Retrieval for Image and Video Databases, SPIE*, III:329–338, Feb 1995.

[12] M. J. Swain and D. H. Ballard. Indexing via color histograms. *Third International Conference on Computer Vision*, pages 390–393, 1990.

[13] D. Swanberg, C. F. Shu, and R. Jain. Knowledge guided parsing in video databases. *Storage and Retrieval for Image and Video Databases, SPIE*, 1908:13–25, Feb 1993.

[14] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, Inc., 1982.

[15] M. M. Yeung and B. L. Yeo. Time-constrained clustering for segmentation of video into story units. *International Conference on Pattern Recognition*, pages 375–380, 1996.

[16] M. M. Yeung, B. L. Yeo, W. Wolf, and B. Liu. Video browsing using clustering and scene transitions on compressed sequences. *Multimedia Computing and Networking, SPIE*, 2417:399–413, 1995.

[17] H. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic parsing of full-motion video. *ACM Multimedia Systems*, 1:10–28, 1993.

[18] H. J. Zhang, Y. H. Gong, S. W. Smoliar, and S. Y. Liu. Automatic parsing of news video. *International Conference on Multimedia Computing and Systems*, pages 45–54, 1994.

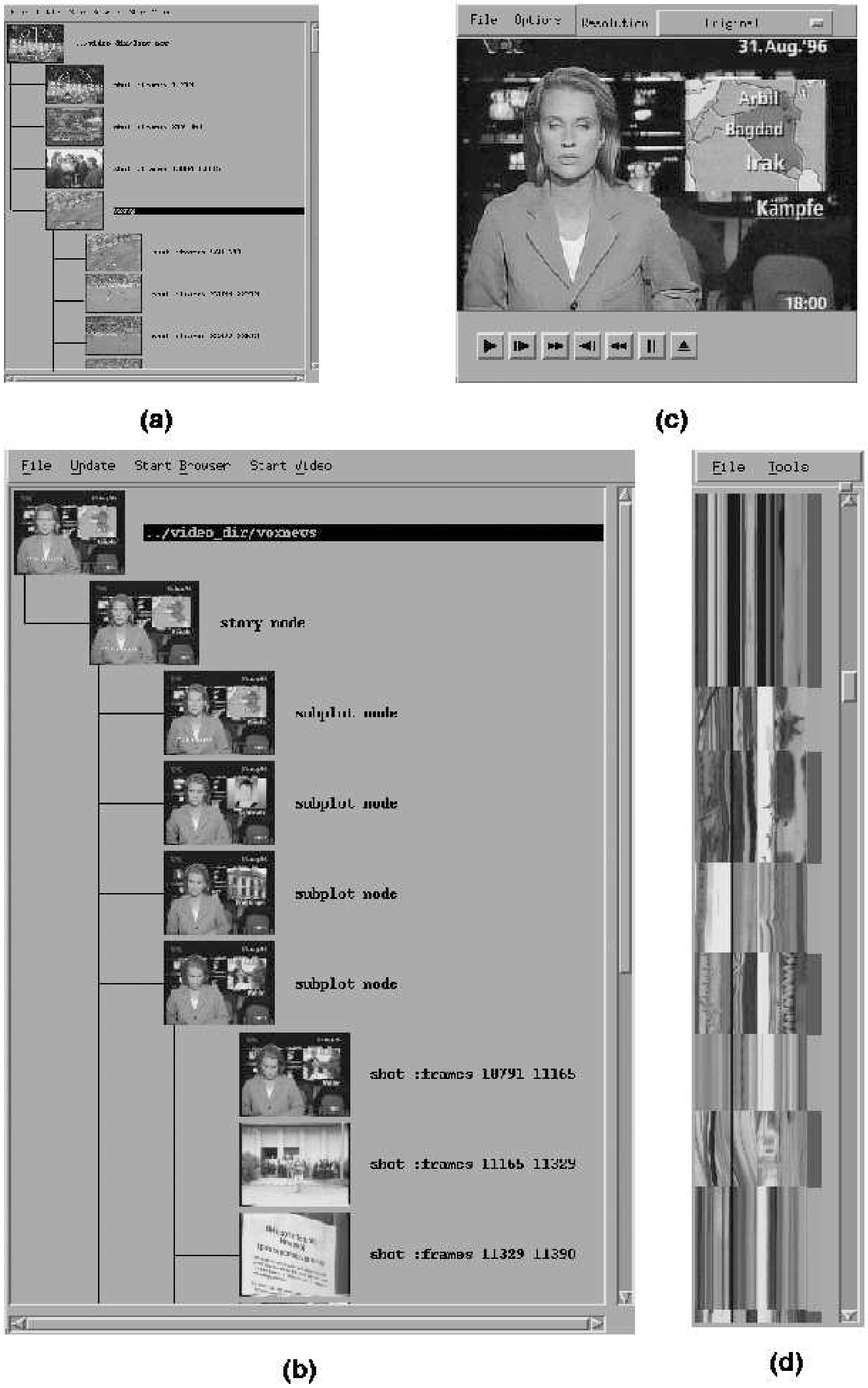


Figure 11. Interfaces: (a) TreeView showing group structure (b) TreeView showing tree structure (c) Video player (d) Browser