JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 7, No. 1, March, pp. 28–43, 1996
ARTICLE NO. 0004

A Survey of Technologies for Parsing and Indexing Digital Video[1]

GULRUKH AHANGER AND THOMAS D. C. LITTLE

Multimedia Communication Laboratory, Department of Electrical, Computer and Systems Engineering, 44 Cummington Street, Boston University, Boston, Massachusetts 02215

Received April 10, 1995; accepted December 4, 1995

In the future we envision systems that will provide video information delivery services to customers on a very large scale. These systems must provide customers with mechanisms to select programs of their choice from live broadcasts. Customers should also be provided with easy means of browsing and accessing prerecorded digital data (e.g., distributed digital multimedia libraries), and downloading data from other information sources. To be viable for such large information sets, these systems must understand customer preferences and tailor the available information to the customer's needs. To support this vision, a number of issues must be addressed and obstacles overcome. Intuitive interfaces, powerful query formulation and evaluation techniques, comprehensive data models, and flexible presentation functionalities must be developed. To realize these components, an effective query evaluation engine with the capabilities of query resolution in different content-specific formats (e.g., by graphics, by image, by sound) and in different domain-specific models (e.g., database of movies, database of newsclips) should be present. Additionally, the digital video database will require an efficient indexing system for easy access to the stored information. In this paper we discuss existing research trends in this area and requirements for future data delivery systems. An overview of video indexing is presented followed by a discussion on current indexing techniques. © 1996 Academic Press, Inc.

1. INTRODUCTION

Today's technology cannot effectively deal with the demands made by burgeoning large scale digital data delivery systems. In the past, information has been collected as discrete units (e.g., numeric and alphanumeric) and retrieved using mechanisms such as structured query language (SQL) strings or hyperlinks as used in the World Wide Web (WWW). In contrast, digital multimedia data are voluminous and contain such a vast amount of information that new technologies are required to effectively retrieve data. Since the potential for applications like live broadcast, video-on-demand, and digital libraries is enormous, the challenges presented by these applications, though formidable, must be met. To do so we must consider and understand the following issues:

- Why traditional database retrieval techniques are insufficient for multimedia retrieval (e.g., why can we not just index and retrieve images using keywords?).
- How spatial visual data types differ from alphanumeric types in terms of data modeling.
- What the roles of the *user* and the *system* are in performing queries.

Visual data are perceived differently by different people. Because of the visual nature of video data, we end up with numerous interpretations of the same data. To represent all the different interpretations by keywords (text) is an impossible task as one cannot foresee all possible interpretations of the data during indexing. Also, representation of a small segment of video data by a large number of keywords will lead to space explosion during indexing. Keywords cannot successfully represent the temporal nature of video data nor do they support semantic relationships (inference rules, hierarchy, and similarity descriptors ("like this"), e.g., find me the image containing vases similar to the given vase shape, find the images containing colors similar to the given image). The large number of keywords necessary to formulate a query makes the process of data retrieval long, tedious, and inefficient.

As the size of the database increases, so do the problems faced by the user in retrieving data. Often a user can use a priori knowledge of the database contents to retrieve data efficiently. However, a large data space strains the user's ability to understand the data content [11]. Users need to explore large amounts of data to find the desired content. A knowledge of the database schema and the data model is necessary to do so. Possessing such detailed information about a large database is difficult. Therefore, we need to provide the user with intuitive ways of understanding the contents of a database.

In addition to the problem of possessing knowledge

[1] Portions of this work were presented at the IS&T/SPIE Symposium on Electronic Imaging Science and Technology (Digital Video Compression and Processing on Personal Computers: Algorithms and Technologies), San Jose, Feb. 1994. This work is supported in part by the National Science Foundation under Grants IRI-9211165 and IRI-9502702.

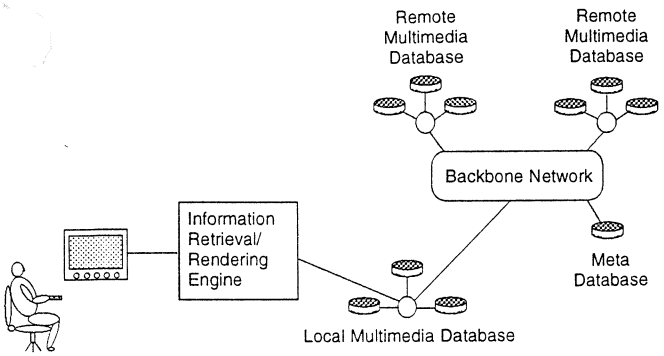

FIG. 1. Distributed databases.

about a large database, there is a problem of storing large amounts of video data. Due to the limited storage capacities of existing storage devices, multiple storage devices must be used. Moreover, the data need not be stored at a single site as illustrated in Fig. 1: therefore, it is important to be aware of data existing at different sites [41].

Only after the information is modeled and extracted from the data can a user possess knowledge about the information in the stored data. Therefore, before the raw video data can be used to issue queries they must be indexed by content and their indices stored as metadata. Figure 2 depicts the various stages of preprocessing that raw video data undergo. First, the video data should be broken into manageable segments and then their features

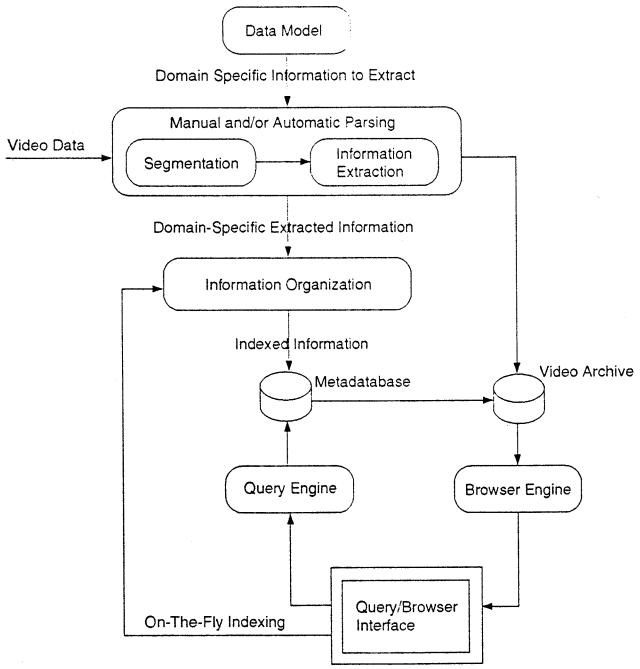

FIG. 2. Processes for construction and use of a digital video library.

extracted (e.g., color, shape, size, texture, object) based on the information required by the application model (e.g., database of movies, database of newsclips). The metadatabase should be used to store the extracted information to speed up the data retrieval process from the video archive. If some information about the video data content is not stored in the metadatabase, the user should be able to extract the information *on-the-fly*, and store them in the metadatabase.

In this paper we discuss different components that make up a successful video delivery application (Fig. 2). In Section 2 we describe the issues and requirements for video indexing. Existing video data models are discussed in Section 3. Image dynamics for compressed video and their applications in modeling the data are discussed in Section 4. Section 5 discusses methods for segmentation of video data into smaller units. In Section 6 we discuss the techniques for data representation and organization. Finally in Section 7 we present an example of how the integration of different aspects of the video computation mentioned can be achieved. Section 8 concludes the paper.

2. VIDEO DATA INDEXING

Using video as a primary multimedia data source requires effective ways of retrieving the desired video data from a database. To do so, a model that classifies video data on the basis of its semantic properties must be developed. Then the video data based on the model should be organized for easy access. We require a good indexing mechanism for this purpose. The large information content present in a video data makes manual indexing (information extraction) labor intensive, time consuming, and prone to errors. The errors introduced are generally of two types. The first are perceptual, made by the person indexing the data, and the second are simple errors introduced due to level of alertness, ambient conditions, fatigue, motivation, etc. Two primary approaches, preprocessing the video data and dynamic interaction with the data, form the basis for indexing in these systems. The first approach uses preextracted indices; i.e., they are created a priori by manual or batch-mode (manual/automatic) processes. The second approach is much more flexible but as automatic parsing techniques are used, it is difficult to achieve and requires significant computational power.

A number of visual systems have been proposed for the retrieval of multimedia data. These systems fall broadly under four categories: query by content [12, 23, 29, 30, 34], iconic query [8, 17, 69], SQL query [53, 57], and mixed queries [1, 17, 41, 58]. The query by content is based on images, tabular form, similarity retrieval (rough sketches), or by component features (shape, color, texture). The iconic query represents data with "look alike" icons and specifies a query by the selection of icons. SQL queries

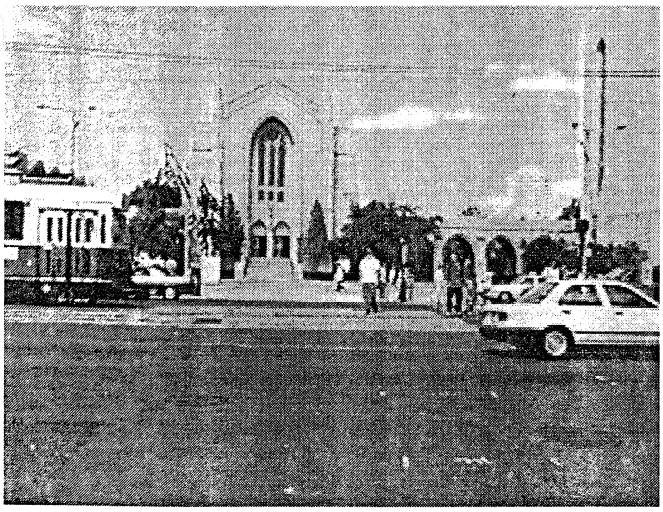

FIG. 3. Photograph of a busy street.

are based on keywords, with the keywords being conjoined with the relationship (AND, OR) between them, thus forming compound strings. The mixed queries can be specified by text and as well as icons. All of these systems are based on different indexing structures as discussed in Section 6.

The interpretation of video data depends on the perception of the viewer. A single image can convey different meanings to different people depending on what a person is looking for. For example, consider the image in Fig. 3. It can be interpreted as a picture of a busy street, a picture of people walking, a picture of a chapel, a picture of cars moving, or a train [1]. Therefore, a flexible process for video indexing is required.

To browse and retrieve data from a large multimedia data source the system must support a strong access technique. As the most complex form of multimedia data is video data, to develop an application for accessing information (e.g., digital libraries) we must understand the properties of digital video data for indexing purposes. The adaptation/indexing process for video can be subdivided into five steps, i.e., *modeling, segmentation, extraction, representation, and organization*. As mentioned before, video data are broken into logical/temporal interconnected segments. In the extraction process the content information is retrieved and then represented as text, strings, or data tokens. In the last stage the extracted information is modeled based on domain-specific attributes.

3. VIDEO DATA MODELING

In a conventional database management system (DBMS), access to data is based on distinct attributes of well-defined data developed for a specific application. For unstructured data such as audio, video, or graphics, similar attributes can be defined. A means for extracting information contained in the unstructured data is required. Next, this information must be appropriately modeled in order to support both user queries for content and data models for storage (see Fig. 4).

From a structural perspective, a motion picture can be modeled as data consisting of a finite-length of synchronized audio and still images. This model is a simple instance of the more general models for heterogeneous multimedia data objects. Davenport *et al.* [16] describe the fundamental film component as the *shot*: a contiguously recorded audio/image sequence. To this basic component, attributes such as content, perspective, and context can be assigned, and later used to formulate specific queries on a collection of shots. Such a model is appropriate for providing multiple views on the final data schema and has been suggested by Lippman and Bender [40]. Bender *et al.* [6], and Loeb [42, 43].

Smith and Davenport [64, 65] use a technique called *stratification* for aggregating collections of shots by contextual descriptions called *strata*. These strata provide access to frames over a temporal span rather than to individual frames or shot endpoints. This technique can then be used primarily for editing and creating movies from source shots. It also provides a quick access and a view of desired blocks of video. Figure 5 shows an example of this stratification technique.

Because of the linearity of the medium we cannot get a coherent description of an item but as a result of the stratification method the related information is lumped together. The linear integrity of the raw footage is erased, resulting in contextual information which relates the shot with the environment. A method which retains this contextual information is required. In Fig. 5, the first column is the tape identification number, the second column contains the frame number, the third column is the content marker, and the fourth column contains the keywords. With keywords, we can consistently find related strings of words. The keywords "AP1" and "AP2" remain constant while the content markers change; therefore, the keywords provide the context to the content marker. The patterns formed in the figure after the frame numbers are sorted to illustrate the contextual relationship among the continuously recorded video frames. Tracing this pattern provides

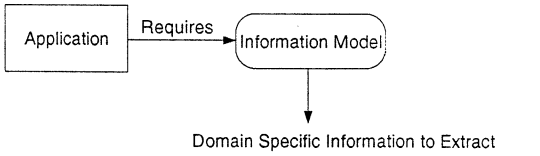

FIG. 4. First stage in video data adaptation: data modeling.

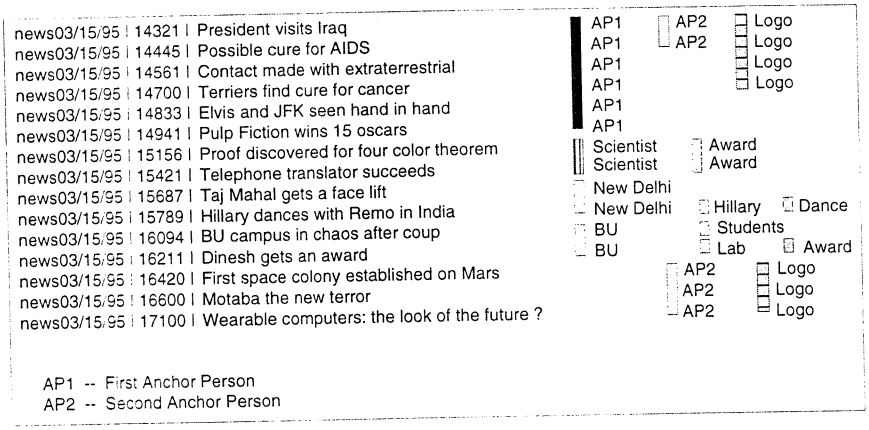

FIG. 5. Example of stratification on newscast video.

us with information about shots, e.g., where and what was shot. As shown in Fig. 5, each layer of stratum represents discrete distinctive contents of the medium. When these threads are layered one on top of another they produce a descriptive strata from which an inference about the content of each frame is drawn.

Rowe *et al.* [58] have developed a video-on-demand system for video data browsing. In this system the data are modeled based on a survey of what users query for. Three types of indices were identified to satisfy the user queries. The first is a textual bibliographic index which includes information about the video and the individuals involved in the making of the video. The second is a textual structural index of the hierarchy of movie, i.e., segment, scene, and shots. The third is a content index which includes keyword indices for the audio track, object indices for significant objects, and key images in the video which represent important events.

The above model does not utilize the semantics associated with video data. Different video data types have different semantics associated with them. We must take advantage of this fact and model video data based on the semantics associated with each data type.

4. INFORMATION EXTRACTION

For video data to become "usable" (i.e., accessed through an application), information contained in the data needs to be extracted (see Fig. 6). The information at the physical (pixel) level can be extracted by parsing the data automatically, manually, or a combination of the two (hybrid). Automatic extraction depends heavily on techniques (e.g., image processing tools) used in computer vision. Content of unstructured data such as imagery or sound is easily identified by human observation; however, few attributes lend to machine identification. Therefore, we are more dependent on hybrid (i.e., combination of automatic and manual) extraction techniques. The shortcomings of automatic extraction techniques can be handled manually.

We need to define distinct attributes of video data for content-based retrieval of the data. The information contained in the data is extensive and diverse; therefore, extracting the data can be a difficult task. It is easier to extract information from smaller data segments. Hence, depending on their dynamics, video data must be segmented into smaller logical units. In the next section we discuss the dynamics of video data in detail.

4.1. *Video Scene Dynamics*

The data comprising a video stream can be modeled as a sequence of still digital frames. The still frames are composed of pixels. A 640 × 480 image "still" contains 307,200 such pixels. If pixels are used to compare successive frames for the purpose of detecting scene transitions,[2] the resulting process would be extremely slow and require significant time. However, when a compression scheme such as the JPEG standard [70] is applied to the frames, a significant amount of information about the shots in the

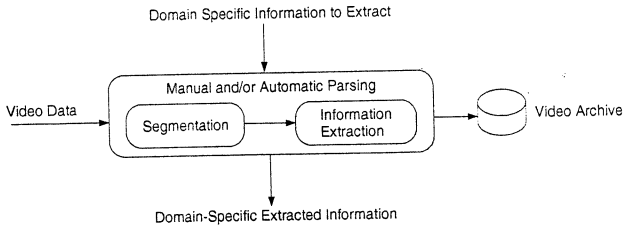

FIG. 6. Second stage in video data adaptation: information extraction.

[2] We use "scene transition" to refer to any change (scene or shot) in the video data stream.

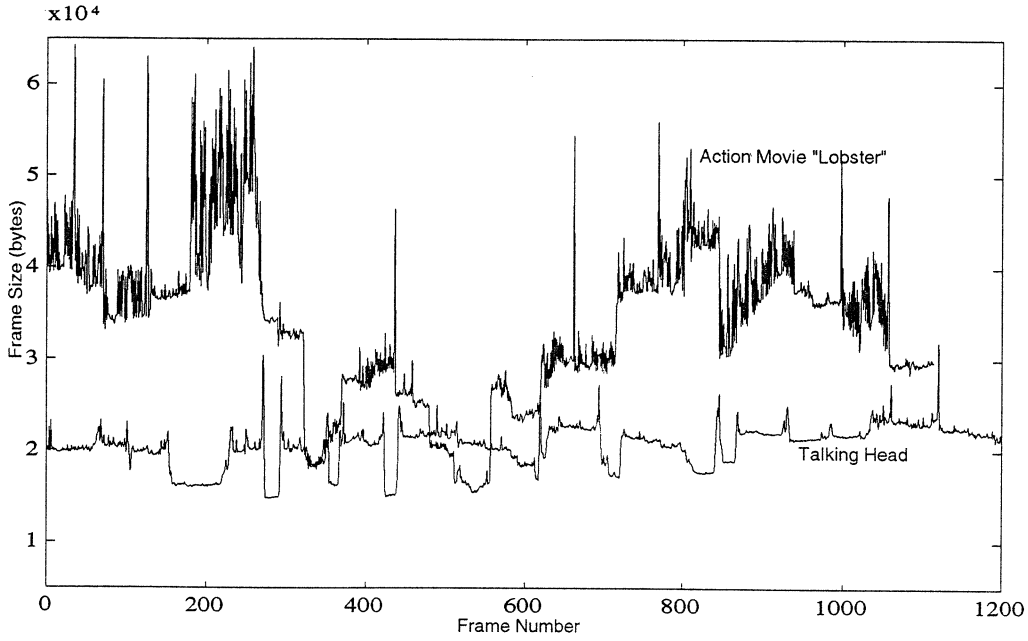

FIG. 7. Image size dynamics for JPEG-compressed movie.

frames can be extracted. This is due to short-term temporal consistency within the frames of a shot. Because there is little change in the overall image content of a shot, the compressed frames of the shot are similar in size (in bytes); i.e., frames within a single scene exhibit consistent intra-frame image complexity. Therefore, by studying the file size dynamics of the compressed frames, we can find transitions from one shot to the next. However, a filtering scheme based purely on size will fail when two shots of similar complexity follow each other. It may also erroneously detect changes when none exist. When such situations occur, more complex algorithms must be employed. The image size dynamics for two sample JPEG-compressed video sequences are shown in Fig. 7 [18].

Figure 7 illustrates the differences in the characteristics of compressed video for two different videos sequences. The more dynamic (bursty) sequence is from a movie clip with significant action (motion and complex image sequences shot over water). The second, less bursty sequence is from a video-taped classroom and is representative of "talking head" video dynamics. This sequence has little motion compared to the action movie. Consequently, the scene and shot transitions are more easily identifiable in the first case where changes are abrupt and quite prominent. Two different kinds of scene changes are illustrated in Fig. 8. The first is a "merge" and "dissolve" where the images are superimposed over each other at a shot boundary. This yields a composition of two images at the scene transition which is difficult to compress and results in a spike in the compressed frame size plot. The second type of change is a "cut." In this case, an abrupt change from one frame to the next due to the cut causes a positive or negative step in the compressed frame size plot.

An number of people are working on techniques for automating the identification of shot transitions and the generation of indices for their location [36–38, 48, 72, 73]. Two stages have been proposed for this task:

- Shot detection: Video data have been isolated into meaningful segments to serve as units to be indexed. This is achieved by detecting the camera transitions. There are two type of transitions. *abrupt transitions* or camera break and *gradual transitions,* e.g., fade-in, fade-out, dissolve, and wipe.
- Feature extraction: The semantics of the video content are extracted during this stage.

Video data are hierarchical in nature, (i.e., video, scenes, shots, and frames) and this property is exploited when indexing the video. Fundamentally, most approaches use the concept of partitioning the continuous video data into sequences for indexing. Video are segmented primarily on the basis of camera breaks or shots; therefore, each sequence is a segment of data having a frame or multiple consecutive frames. Automatic isolation of camera transitions requires support of tools that provide accurate and fast detection. Abrupt camera transitions can be detected quite easily as the difference between two consecutive frame is so large that they cannot belong to the same shot. A problem arises when the transition is gradual; the shot does not change abruptly but over a period of few frames. The difference

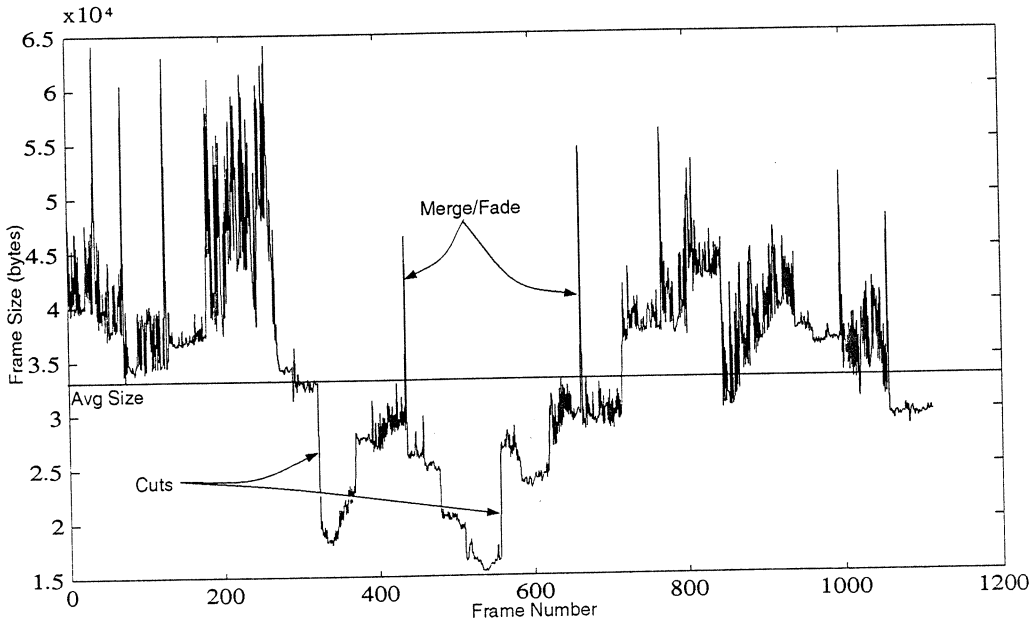

FIG. 8. Illustration of various scene transitions.

between consecutive frames is not so large as to declare it a camera break. The gradual transitions become more difficult to detect because the frame content shares the same semantic properties as frames with object motion or special camera effects such as wipe, dissolve, fade-in, fade-out.

5. VIDEO SCENE SEGMENTATION

Seyler [61] developed a frame difference coding technique for television signals. The technique is based on the fact that only a fraction of all picture elements change in amplitude in consecutive frames. Since then a number of digital video data segmentation techniques based on this concept have been developed. A number of metrics have been suggested for video scene segmentation for both the raw data and compressed data. The metrics used to detect the difference between two frames can be divided broadly into four classes: pixel or block comparison, histogram comparison (of gray levels or color codes) [36, 48, 72], using the DCT coefficients in MPEG-encoded video sequences [2, 63, 73], and the subband feature comparison method [37]. Some of the methods are discussed below.

5.1. *Pixel-Level Change Detection*

The change between the two frames can be detected by comparing the differences in intensity values of corresponding pixels in the two frames. The algorithm counts the number of the pixels changed, and the camera break is declared if the percentage of the total number of pixels changed exceeds a certain threshold [36, 48, 72]. Mathematically, the difference in pixels and threshold calculation can be represented by Eqs. (1) and (2). In Eq. (1), $F_i(x, y)$ is the intensity value of the pixel in frame i at the coordinates (x, y). If the difference between the corresponding pixels in the two consecutive frames is above a certain minimum intensity value, then $\mathrm{DP}_i(x, y)$, the difference picture, is set to one. In Eq. (2), the percentage difference between the pixels in the two frames is calculated by summing the difference picture and dividing by the total number of pixels in a frame. If this percentage is above a certain threshold T, a camera break is declared.

$$\mathrm{DP}_i(x, y) = \begin{cases} 1 & \text{if } |F_i(x, y) - F_{i+1}(x, y)| > t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\frac{\sum_{x,y=1}^{X,Y} \mathrm{DP}_i(x, y)}{X * Y} * 100 > T. \tag{2}$$

Camera movement, e.g., pan or zoom, can have the effect of a large number of pixel changes, and hence a segment will be detected. Fast moving objects also have the same effect. If the mean intensity values of the pixel and its connected pixels are compared [72] then the effects of camera and object motion are reduced.

5.2. *Likelihood Ratio*

Detecting changes at the pixel level is not a very robust approach. A *likelihood ratio* approach is suggested based on the assumption of uniform second-order statistics over

a region [36, 47, 72]. The frames can be subdivided into blocks and then the blocks can be compared on the basis of the statistical characteristics of their intensity levels. Equation (3) represents the formula that calculates the likelihood function. Let μ_i and μ_{i+1} be the mean intensity values for a given region in two consecutive frames and S_i and S_{i+1} be the corresponding variances. The number of the blocks that exceed a certain threshold t are counted. If the number of blocks exceeds a certain value (dependent on the number of blocks) a segment is declared. A subset of the blocks can be used to detect the difference between the images so as to expedite the process of block matching:

$$\lambda = \frac{[((\sigma_i + \sigma_{i+1})/2) + ((\mu_i - \mu_{i+1})/2))^2]^2}{\sigma_i * \sigma_{i+1}}$$
$$\mathrm{DP}_i(k, l) = \begin{cases} 1 & \text{if } \lambda > t \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

This approach is better than the previous approach as it increases the tolerance against noise associated with camera and object movement. It is possible that even though the two corresponding blocks are different they can have the same density function. In such cases no change is detected.

In another block matching technique that has been proposed by Shahraray [63], a typical frame is divided into 12 nonoverlapping blocks. Block matching is performed on the image intensity values and the matched parameters are normalized between the values of zero and one where zero indicates a perfect match. The match coefficient between the two images is defined as in Eq. (4). Let i be the block number, K the total number of blocks, l_i the element of the ordered set of the match values, and c_i a predetermined coefficient for each block:

$$\mathrm{IM} = \sum_{i=1}^{K} c_i l_i. \tag{4}$$

Methods for detecting gradual scene transition and intershot scene changes induced by the camera are also explained in the paper by Shahraray but no performance measures have been given to evaluate the efficacy of these techniques.

Histogram Comparison. The sensitivity to camera and object motion can be further reduced by comparing the gray level histograms of the two frames [48, 72]. This is due to the fact that two frames with not much difference in their background and some amount of object motion have almost the same histograms. The histogram is obtained from the number of pixels belonging to each gray level in the frame. In Eq. (5) G is the number of gray levels, j is the gray value, i is the frame number, and $H_i(j)$ is the histogram for the gray level j:

$$\sum_{j=1}^{G} |H_i(j) - H_{i+1}(j)| > t. \tag{5}$$

If the sum is greater than the given threshold t then a transition is declared.

Histogram comparison using color code is also suggested [48, 72]. A color code value is derived from the three color intensities. To reduce the bin size, instead of representing the code by 24 bits, the upper two bits of each color are used to compose the color code. The j in the above equation is replaced by the code value. The color histogram metric is more robust as it eliminates the necessity of converting the color level to gray level.

An χ^2-test comparison of color histograms is proposed by Nagasaka and Tanaka [48]. The function uses the square of the difference between the two histograms so as to strongly reflect the difference:

$$\sum_{j=1}^{G} \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} > t. \tag{6}$$

The χ^2 test enhances the difference between the camera breaks and also small changes due to camera or object motion [72]. Therefore, this method may not be more efficient than the gray and color histogram comparison techniques.

5.3. *Twin Comparison*

Special camera effects make it difficult to detect camera breaks by means of any of the above methods and transitions can go undetected. This is because the threshold set in the above methods is higher than the difference found between the frames in which transition takes place due to special effects. Lowering the threshold does not solve the problem because the difference value due to the special effects can be smaller than those which take place within the shot. For example, object motion and/or camera motion might contribute more changes than the gradual transition. Making the value of the threshold even smaller will lead to false detections due to camera and object motions. The beginning and end frames for the gradual transitions need to be detected; the frames in the transition can be declared as a separate segment.

The twin-comparison method [72] takes into account the cumulative differences between the frames for gradual transitions. This method (Fig. 9) requires two cutoff thresholds, one higher threshold (T_h) for detecting abrupt transitions and a lower one (T_l) for gradual transitions. In the first stage a higher threshold is used for detection of abrupt transitions. In the next stage a lower threshold is used on

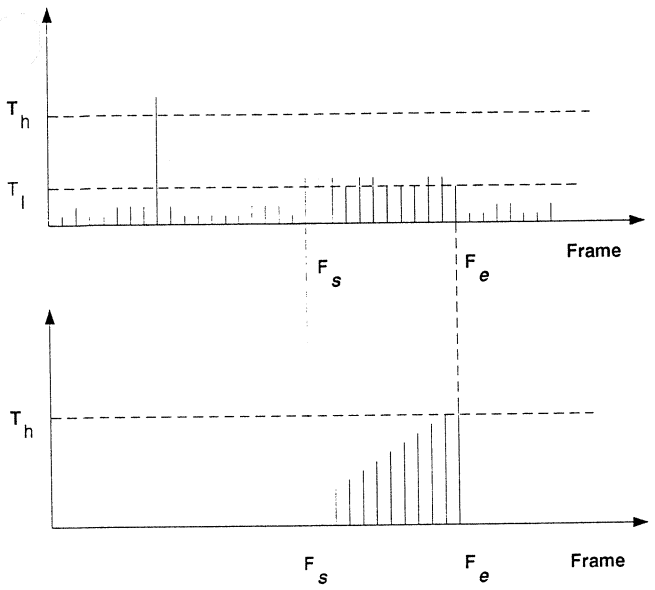

FIG. 9. Illustration of twin-comparison.

the rest of the frames: any frame that has the difference more than this threshold is declared as a potential start (F_s) of the transition. This frame is then compared with subsequent frames and the difference added. Usually this difference value increases and when this value increases to the level of the higher threshold, camera break is declared at that frame (F_e). If the value falls between the consecutive frames then the potential frame is dropped and the search starts all over. There are some gradual transitions in which the difference falls below the lower threshold. The user can set a tolerance value which will allow a certain number of consecutive frames to fall below the threshold.

The gradual transitions so detected might include special effects due to camera panning and zooming. A commonly used technique in computer vision, *optical flow*, is used to detect the camera movements. Motion vectors are computed to detect the changes due to pan and zoom.

5.4. *Detection of Camera Motion*

To detect the camera motion, optical flow techniques are utilized. Optical flow gives the distribution of velocity with respect to an observer over the points in an image. Optical flow is determined by computing the motion vector of each pixel in an image. The fields generated by zoom, pan, and tilt are shown in Fig. 10. Detecting these fields helps in separating the changes introduced due to the camera movements from the special effects such as wipe, dissolve, fade-in, fade-out.

As seen in Fig. 10 most of the motion vectors between consecutive frames due to pan and tilt point in a single direction, thus exhibiting a strong modal value corresponding to the camera movement. Disparity in direction of some of the motion vectors will result from object motion and other kinds of noise. Thus, a single modal vector is exhibited with respect to the camera motion. As given by Eq. (7), a simple comparison technique can be used to detect pan and tilt. Pan and tilt are detected by calculating the difference between the modal vector and the individual motion vectors. Let θ_l be the direction of the motion vectors and θ_m the direction of the modal vector. If the sum of the differences of all vectors is less than or equal to a threshold variation Θ_p then a camera movement is detected. This variation should be zero if no other noise is present:

$$\sum_{l}^{N} |\theta_l - \theta_m| \leq \Theta_p. \tag{7}$$

Motion vectors for zoom have a center of focus, i.e., focus of expansion (FOE) for zoom-in and focus of contraction (FOC) for zoom-out. Due to the absence of noise, it is easy to detect the zoom because the sum of the motion vectors around the FOC/FOE will be zero. But it is difficult to find the center of focus of zoom since it could be present across two consecutive frames. Zhang *et al.* [72] assume that the FOE/FOC lies within a frame; therefore, it is not necessary to locate it for vector comparison. A simple comparison technique can be used, the vertical vectors from the top (V_k^{top}) and the bottom rows (V_k^{bottom}) can be compared for magnitude, and horizontal vectors from the left-most (U_k^{top}) and right-most (U_k^{bottom}) vectors at the same row can be compared. In the case of zoom, the vectors will have opposite signs, and the magnitude of the difference of these components should exceed the magnitude of the highest individual component. This is due to the fact that in every column the magnitude of the difference between these vertical components will exceed the magnitude of both components:

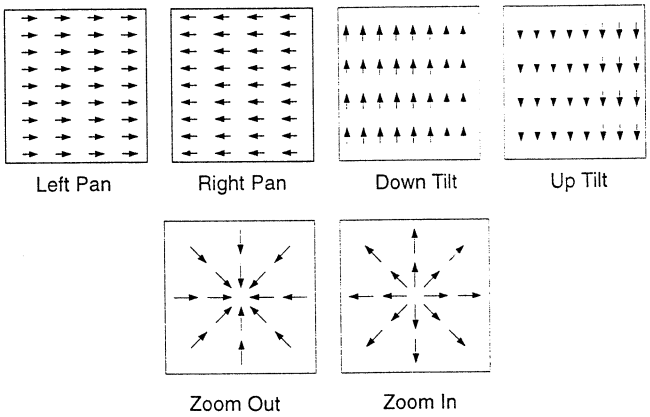

FIG. 10. Optical flow field produced by pan, tilt and zoom.

$$|V_k^{\text{top}} - V_k^{\text{bottom}}| \geq (|V_k^{\text{top}}|, |V_k^{\text{bottom}}|) \quad (8)$$

$$|U_k^{\text{top}} - U_k^{\text{bottom}}| \geq (|U_k^{\text{top}}|, |U_k^{\text{bottom}}|). \quad (9)$$

When both Eqs. (8) and (9) are satisfied, a zoom is declared. Thus, the camera movements can be separated from the gradual transitions detected by other techniques (Section 5.3).

5.5. *Using DCT Coefficients in MPEG-Encoded Video*

In this method the compressed video data are used for detecting camera breaks [2], and the amount of the data to be processed is reduced considerably. Compression of the video is carried out by dividing the image into a set of 8×8 pixel blocks. The pixels in the blocks are transformed into 64 coefficients using the discrete cosine transform (DCT), which are quantized and Huffman entropy encoded. The DCT coefficients are analyzed to find frames where camera breaks take place. The coefficients in the frequency domain are mathematically related to the spatial domain; therefore, they can be used in detecting the changes in the video sequences.

Given 8×8 blocks of a single DCT-based encoded video frame f, a subset of blocks is chosen a priori. The blocks are chosen from n connected regions in each frame. Again a subset of the 64 coefficients for each block is chosen. The coefficients chosen are randomly distributed among the AC coefficients of the block. Taking coefficients from each frame a vector is formed.

$$V_f = \{c_1, c_2, c_3, \ldots\}$$

The vector V_f represents the frame f of the video sequence in DCT space. The inner product is used to find the difference between the two frames. In Eq. (10), V_f is the vector of the frame being compared, V_{f+1} is the vector of the successor frame:

$$\Psi = \frac{\vec{V}_f \vec{V}_{f+1}}{|\vec{V}_f||\vec{V}_{f+1}|}. \quad (10)$$

A transition is detected when $1 - |\Psi| > t$, where t is the threshold.

Zhang *et al.* [73] have also experimented with motion-based segmentation using the motion vectors in the MPEG compressed data as well as DCT coefficients. Meng *et al.* [45] extend this concept further by performing more detailed operations on the MPEG/MPEG-2 compressed data. The compressed data consists of I-, P-, and B-frames. An I-frame is completely intraframe-coded. No motion compensation is performed. A P-frame is predictively coded with motion compensation from past I- or P-frames.

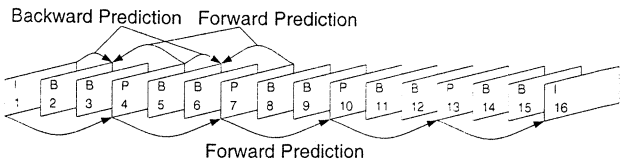

FIG. 11. A typical MPEG compressed video sequence.

Both these frames are used as a basis for bidirectionally motion compensated B-frames.

If a break occurs on a B-frame, most of the motion vectors will come from the following anchor frame and very few will come from the previous anchor frame (refer to Fig. 11). Based on the ratio of backward and forward motion vectors a scene cut is detected. P-frames have only forward motion compensation; therefore, when a scene change occurs here then the encoder cannot use macroblocks[3] [31] from the previous anchor frame for motion compensation. Based on the ratio of macroblocks without motion compensation to marcroblocks with motion compensation the scene break is detected. Since I-frames are completely intracoded without motion vectors, any of the previous methods can be used for break detection.

5.6. *Segmentation Based on Object Motion*

Video data can also be segmented based on the analysis of encapsulated object motion [38]. The dynamic-scene analysis is based on recognizing the objects and then calculating their motion characteristics. Objects in video data are recognized either by their velocity or by their shape, size, and texture characteristics.

The motion which we perceive is the effect of both an object and camera motion. Camera motion information can be exploited in the scene segmentation process based on object motion. For example, if the camera is in motion then the image has nonzero relative velocity with respect to the camera. The relative velocity can be defined in terms of the velocity of the point in the image and the distance from the camera. For segmenting moving-camera scenes, the component of the motion due to camera motion should be removed from the total motion of an object. Accumulative-difference pictures [33] can be used to find areas that are changing (usually due to object motion) and hence imply scene segmentation. Velocity components of the points in an image can be used for segmenting a scene into different moving objects. This is due to the fact that points of an object optimistically have the same velocity components under certain camera motions and image surfaces [49]. Equation (11) can be used for calculating the image

[3] The four 8 by 8 blocks of luminance data and the two, four, or eight corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture.

nsity velocity, where $I(x, y, t)$ is the image intensity, ∂x and $\partial I/\partial y$ denote the image-intensity gradient, and dx/dt and dy/dt denote the image velocity:

$$dI/dt = \partial I/\partial x * dx/dt + \partial I/\partial y * dy/dt + \partial I/\partial t. \quad (11)$$

If the image intensity velocity changes abruptly, a scene change can be declared. This method is much more computationally intensive as an object must be tracked through a sequence of frames. This is much more difficult in the presence of noise. Hough transforms have also been used to segment the scene into different moving objects [22]; however, this technique is not robust in the presence of noise.

5.7. *Segmentation Based on Subband-Coded Video Data*

Lee and Dickinson [37] have developed a method for feature indexing of subband coded video. Subband coding refers to a technique in which the input signal is decomposed into several narrow bands by splitting with the help of low-pass and high-pass filters. Temporal segmentation is performed on the lowest subband. Four different metrics for segmentation of video data have been developed as follows:

1. Difference of histogram: measures absolute sum of the histograms of two frames.
2. Histogram of difference image: measures the histogram of difference between two frames.
3. Block histogram difference: block histogram difference is obtained by computing histograms of each block and summing the absolute difference of these block histograms between the two frames.
4. Block variance difference: same as 3, except this method uses the block variance.

5.8. *Segmentation Based on Features*

Work on feature-based segmentation is being done by Zabih *et al.* [71]. The segmentation process involves analyzing intensity edges between two consecutive frames. During cut and dissolve operations, new intensity edges appear far from the old ones due to change in content. Thus, by counting the new and old edge pixels, cuts, fades, and dissolves are detected and classified. Canny's edge detector is used in their algorithm. The image is smoothed by a Gaussian matrix and the gradient thresholded by a certain value. Motion compensation is also provided in the algorithms to give better results in presence of camera and object motion.

Hampapur *et al.* [28] use a model-driven approach to digital video segmentation. Chromatic scaling models are used to classify editing effects in the video production. This model is used by feature detectors for finding editing effects. Otsuji and Tonomura [54] have proposed a cut detection method that uses an isolated sharp peak detecting filter.

The techniques mentioned above segments video with varying degrees of accuracy and robustness. The effectiveness of an algorithm also depends on the kind of video being segmented. If an animated video is used then the scene changes are usually abrupt; i.e., there are few camera effects or light flashes present. Any simple algorithm can be successfully used to segment this type of video data, whereas if a video contains complicated camera motions and effects (e.g., dissolves, wipes, fades, zooms) sophisticated algorithms must be used. The performance of these algorithms depends greatly on the noise level in the digitized video during processing, e.g., sudden changes in lightning. A proper threshold must be established such that no camera breaks lower than the threshold are missed or no false breaks are detected. Basically, a compromise between speed and robustness must be made. The processing time can be reduced either by reducing spatial resolution or by reducing temporal resolution. For spatial resolution only a subset of pixels is used which leads to information loss, which can result in camera break detection failure. For temporal resolution frames are skipped and a subset of frames is used for detection. This speeds up the process and at the same time it helps in detecting gradual camera breaks, as the difference between the frames across the skipped frames is greater than the lower threshold. But if a large number of frames are skipped then it can lead to many false detections.

Zhang *et al.* [72] use a *multipass* approach for reducing overall processing time. In this approach a fast pass is made with a lower threshold along with temporal resolution reduction. The breaks detected are declared as potential break candidates. In the second pass the processing is restricted to the vicinity of these breaks. The authors compared the twin-comparison techniques, pair-wise pixel, likelihood ratio, gray and color histogram comparison, and χ^2 test, using the twin-comparison method and the multipass method. They found that the histogram comparison, based on gray level and color, gave high accuracy, and the computation time was considerably reduced using the multipass system. Approximately 90% of the breaks and transitions were detected. Contrary to Nagasaka and Tanaka [48], the χ^2 test did not do so well in terms of either speed or accuracy. Performing segmentation on the compressed data is faster than uncompressed data. None of these methods work when there is substantial change in the intensities (camera motion, object motion, light flashes) between pairs of frames. Histogram methods are not effective in the presence of motion. Hampapur *et al.* [28] do not use histogram methods; rather, they compute the chromatic difference between consecutive frames. Their

method detects dissolves if the change due to the editing is much more than the change due to motion. The feature-based method [71] is much more robust than the previous methods in detecting dissolves in the presence of motion, but does not handle sudden changes in brightness. Also, motion compensation techniques do not handle multiple object motion.

6. DATA REPRESENTATION AND ORGANIZATION

Much work has been done in automating indexing and searching of data in databases (e.g., text [59], automated offices [44, 68], newspapers and magazines [35], archives and libraries [46, 51], image databases [9, 12, 23–25, 52]). Faloutsos [19] has discussed retrieval methods for text as well as formatted data. Some of the methods are summarized below as they relate to digital libraries.

- *Full Text Scanning.* A complete document is scanned for the specified strings. If a query is a composition of a number of strings in conjunction, then more than one step is required to satisfy the strings in the Boolean function.
- *Inversion of Terms.* This is a keyword approach to accessing data. Contents of a document can be described by an array of words. The data organization is divided into two components, the *index* and the data records or *data leaves.* The information is indexed, and the data are stored in leaves. The structure can be a hash table, tree, or a combination of the two.
- *Multiattribute Retrieval Methods.* These approaches involve superimposed coding techniques. If, for some data, n attributes are chosen, then each attribute is hashed to a fixed-length bit pattern. A prespecified number of bits are set to 1's and then these n patterns are superimposed to get a resulting big pattern. This pattern is called a *signature.* This signature then can be used to locate a data record. *Multiattribute hashing* [27] and *signature files* [20, 21] are some of the multiattribute attribute retrieval methods.
- *Clustering.* In clustering similar data are grouped together into clusters. The reasoning behind this technique is that similar data are pertinent to the same request. Each document is represented by an n-dimensional vector, where n is the number of attributes of the data. Using a similarity matrix, these vectors are clustered. The data are retrieved by comparing a cluster descriptor and a query descriptor.

Full text scanning time is linearly dependent on the size of the text document; the greater the size of the document and the database, the greater is the search time as it is a sequential access method. There are no extra storage overheads for indexing structures in this method as there are in inverted files. On the other hand, inverted files provide speedy data access mechanisms, though it is expensive to update the index structure. Inverted files incorporate a random access mechanism; i.e., the indices direct the search to the small part of the database containing the item. The index structure is a file, so if the file grows larger it can slow down the search. Therefore, another index is built on top of the existing index and the resulting hierarchy speeds up the access.

B-trees [15] are an example of an inverted file. This general-purpose index mechanism was developed in the late 1960s when people were competing for access mechanisms. Each node in the B-tree of order n consists of $2n$ keys and $2n + 1$ pointers. The advantage of this tree lies in the fact that deletion and addition of records always leave it balanced unlike binary trees which become unbalanced (long paths and some short paths) on random insertion and deletion. There are other variations of the B-tree. The B^*-tree [4, 15] increases the storage utilization as the splitting of the node is delayed until the two sibling nodes are full. This not only utilizes the space efficiently but also speeds up the access process. In the B^--tree [15], all the keys reside in the leaves. Therefore, during sequential access, no node will be visited twice and, at most, space for one node is required in the memory. The number of accesses are also reduced, requiring only access for the next consecutive operation. The index part is only the guide to the leaf nodes; therefore, there is no need for using actual keys in the index which requires considerable space. As the space utilization is reduced, the number of keys in the nodes can be increased. The branches per node increase but at the same time the depth of the tree decreases. Thus, the data access time is reduced considerably. Other variations of the B-tree have been suggested, such as compressing the keys and the pointers, binary B-trees, and multiway trees [4, 15]. The B-tree is a one-dimensional key ordering structure designed for single-key access. Thus, it is of not much use in indexing multidimensional video data.

Multidimensional data structures have been developed to overcome the shortcomings of the B-tree. Pixel-based data structures have been suggested by [14]. This is a gridded data structure where pixel information from each grid cell is stored as a record. Relational databases have also been used in pictorial queries [9] as well as video queries [41, 58]. Chang and Kunii [12] have used *2-D strings* to represent pictures in a pictorial database. Images are projected into a 2-D coordinate system, and a 2-D string is derived which preserves the spatial knowledge of the objects in an image. Lee and Shan [39] have further extended the 2-D string method. Along with the 2-D string, a signature is associated with each image. Instead of processing the 2-D string file, the image signature preselector prunes a large set of signatures whose 2-D strings will not satisfy the query. After this step the remanent strings are processed, making the processing more efficient. Bimbo *et al.* [8] have extended the 2-D strings to represent 3-D scenes which seems more relevant to video data as scene descrip-

can also be made by including reference to the relative depth of an object.

Quadtrees [60] have been used as multi-indexing structures for large databases. The quadtree is a multidimensional generalization of a binary tree which utilizes the grid file mechanism, but the quadtree has a number of limitations. First, at each node all the keys must be tested; second, there are a number of null links that use space resources; and third, the node sizes tend to be large. *K-d trees* [7] have been developed to overcome some of these deficiencies. However, both structures suffer from page faults [26]; i.e., when a node is referenced which is present in a page but not in the memory cache a fault occurs. In these tree structures, the pointers must be followed with great potential for page faults. This problem can be overcome by using methods that collect the pointers into sets which correspond to the storage units (pages) of the disks. The access to these sets (buckets) can be organized by use of a directory to help in address calculation. This technique is called the *bucket method* [60]. The goal of this method is to ensure efficient access to the disk data. The nodes in the B-tree can be used as buckets, but B-tree nodes do not correspond to an n-dimensional space. Video data require a multidimensional indexing structure for the various attributes they can be indexed under, and due to the size of the video data the contents of the tree nodes will be considerably large. Therefore, a multidimensional indexing structure using bucket methods will be appropriate for the video data such that page faults do not occur while reading the data. Robinson [56] developed a *k-d-b-tree* structure to overcome the limitation of the B-tree. It uses *k-d*-tree partitions to arrange the contents of each B-tree node. But the performance of the B-tree cannot be ensured. Chang and Li [10] adapted the 2-D strings to a quadtree data structure and came up with a new structure called *2-D-H string*. This structure is used to represent hierarchical symbolic pictures by employing new spatial relation operators. Again, this structure also provides limited capabilities for indexing the video data as not all of the features can be organized by this structure.

Nievergelt *et al.* [50] proposed a variation of the grid method called *grid files*. Grid files handle range queries. This structure adapts very well to insertion and deletion and, therefore, can retrieve records with a maximum of two disk accesses. Jagadish [32] uses rectangles for feature extraction of the spatial data. The $4k$ vectors thus derived are then indexed by using grid files, k-d-b-trees, etc. *Voronoi diagrams* [3] have also been widely used to represent the spatial data using the nearest-neighbor rule. Spatial data objects are not represented very well with point data. R-tree [26] and its family (R^+-tree [67] and R^*-tree [5]) overcome this limitation. The R-tree is a derivation of the B^+-tree that stores complete multidimensional objects without transformation or clipping. But the R^*-tree is designed with optimization of area covered by a directory rectangle; i.e., the dead space between the bounded rectangles is minimized such that the maximum information can be stored. It maximizes the space by minimizing the margins of the rectangle. The margin is the sum of the lengths of a rectangle; objects with the smallest margins are squares. Thus, by minimizing the margin the directory rectangles will have a more quadratic shape and more rectangles can be incorporated. The overlaps between the directory rectangles are reduced which decreases the number of the paths to traverse.

In the QBIC project [23] the retrieval of images is based on color, texture, and shape as basic image feature. The indexing system is derived by computing the numerical index keys based on color distribution, color histograms, and segmentation based on main color regions. These index keys are organized by using the R^*-tree structure, which is more robust for the higher dimensions and is much faster. Zhang *et al.* [73] have also used the same techniques in their tools for video indexing and retrieval. They essentially utilize the spatial information for indexing the representative frames and ignore the temporal information in the video. Higher level indexing is based on event indexing; e.g., a zooming sequence is retrieved by processing the motion vectors.

The tree types of indices identified for the video-on-demand system by Rowe *et al.* [58] are organized using a relational database. The Chabot project [52] also uses a relational database to store and retrieve images.

In the next section we discuss a few examples of how various video data types can be indexed for storage and retrieval.

7. EXAMPLES

Consider a system designed to retrieve digital video data from a large repository (newscast, instruction, video conferencing, and surveillance). For a video to be in a user accessible format (queriable), raw video data must be processed several times to extract information for content-based retrieval. Before the process of segmentation and feature extraction is applied we are required to model the video data. However, the semantics of interpretation are different for different video data types. In movie data each scene is logically connected to the previous and the following scenes but in surveillance video data does not possess such semantics. Likewise, not all video data have camera breaks in them; segmentation based solely on camera breaks is not possible. But video data must be segmented for better management of data and disk utilization. Refer to Fig. 2 for the model of video data indexing. A model for the data is developed, and based on this model the

video data are broken down into manageable segments. This is achieved by applying any of the algorithms mentioned in Section 5. For example, the color histogram method can be used for detecting abrupt scene changes and the twin-comparison method can be used for detecting gradual transitions. If required, the false detections due to camera effects can be eliminated by using the optical flow techniques. After segmentation, feature (content) information contained in these segments must be extracted. Some of the features like shape, color, and texture information can be retrieved by known image processing methods. Some examples of how video data belonging to different data types can be modeled and segmented are given below.

Example 1. Consider newscast video data. A news session consists of a number of disjointed news items presented sequentially. There is usually little logical connection between them, and the news items can be presented in any order without effecting the overall presentation. Therefore, the access mechanism is much more effective if a segment consists of a complete news item. These segments can be indexed based on their content. Various queries can be performed on the newscast video data, e.g., retrieve clips of some important event as it evolves, retrieve clips of a favorite newscast, retrieve clips of sports news in which a favorite baseball team is mentioned, and retrieve field clips of events happening in a certain country. Before performing such queries, the information needed for retrieval of these clips from the database must be extracted from the video data. The query for a certain important event from a particular broadcast station requires the segments to be indexed with the information pertaining to the event.

Example 2. In a distance learning application the information can have a more rigid structure. They are composite in nature with temporal associations. For example, consider a database consisting of course material on various subjects. Each course is composed of various topics and each topic is further composed of subtopics. Each topic or subtopic is taught in a temporal sequence, e.g., "before" or "after". One or more topics can be taught per class. Querying for a course does not involve complicated indexing but if the user tries to retrieve a topic taught in a course on "networking" which was taught after the protocols topic then a more complicated query must be formulated. For such a query an indexing structure which encompasses both the temporal as well as content is required.

Example 3. Many high security areas, public places, and buildings are equipped with surveillance cameras. A large amount of video data from these sources can be stored in a database on a daily basis. There are no camera breaks present in the recorded data and segmentation based on camera breaks does not apply. If the segments are made based on content change, then we will get a break every few frames as the content can be very dynamic. Therefore, the video can be segmented based on activity. Sometimes the recorded material does not contain information of much importance; hence, data which do not contain very useful information can be edited out. Hierarchy, generalization, and specialization concepts cannot be employed because there is not semantic flow from one event to another. Prominent information in each segment (events, people) and the time associated with the segments can be indexed. Queries based on timing information, e.g., retrieve clips taken at 5 P.M. on Saturday, March 1993, can be executed. Queries based on events, e.g., retrieve clips in which people are running across the camera view, can be executed. Queries based on different camera views, e.g., get the clips from the camera situated near the back door, can also be executed.

8. CONCLUSION

Existing systems treat video data as having rigid hierarchical structures, i.e., scenes, shots, and frames. Video data can be retrieved based on the information content present in the scenes or shots and the bibliographic data associated with the video. This is true for certain kinds of data (movies) but we cannot apply such a generic model to all video data types. Each video data type has different associated semantics. Treat all the video data types as a rigid hierarchical structure is not effective in the retrieval of video data. For example, segmenting the newscast video data based on the camera breaks is not useful. The user might try to retrieve data via a news item which might involve several scenes (scene of the anchor person talking about the item and a field view of the item). The user will likely retrieve data on an event basis, i.e., an entire story. The event could happen over a number of days by a number of broadcast stations. Therefore, we would require more information than is present in a single shot or a scene. Future applications should exploit the different semantics associated with individual data types. While developing a video data retrieval system we need to take a bottom-up approach; we must first understand the semantics present in the data type and then model the data. The applications must be built around the video data rather than tailoring the data to an application. This results in efficient and effective data retrieval mechanisms. It also resolves many issues such as the accuracy of the segmentation process which will not only segment the video data based on camera breaks but also try and identify the types of camera breaks. If a suitable data model is utilized for an application, we can identify the places where the data

ld be segmented and this would reduce considerably amount of time for segmentation of the video data. We will also know what features need to be extracted from the data, thereby reducing the information to be stored. Simple indexing techniques can be used for modeling the metadatabase so we will not require very large multidimensional indexing structures.

Emerging information retrieval technologies that have significant content in digital video form will require access to large "virtual" libraries [66]. The demand for such systems is continually increasing and these systems must be able to acquire large amounts of data, store them, and then make them available to customers. Hence, the indexing process should be fast and efficient and allow effective data access.

REFERENCES

1. G. Ahanger, D. Benson, and T. D. C. Little, Video query formulation, in *Proceedings IS&T/SPIE, Conference on Storage and Retrieval for Image and Video Databases, Feb. 1995,* Vol. 2420, pp. 280–291.
2. F. Arman, A. Hsu, and M.-Y. Chiu, Image processing on compressed data for large video databases, in *Proceedings 1st ACM International Conference on Multimedia, Anaheim, CA, Aug. 1993,* pp. 267–272.
3. F. Aurenhammer, Voronoi diagrams—A survey of a fundamental geometric data structure, *Comput. Surveys* **23**(3), 1991, 345–405.
4. R. Bayer and K. Unterauer, Prefix B-trees, *Proc. ACM Trans. Database Syst.* **2**(1), 1977, 11–26.
5. N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, The *R**-tree: An Efficient and Robust Access Method for Points and Rectangles, in *Proceedings of ACM SIGMOD, May 1990,* pp. 322–331.
6. W. Bender, H. Lie, J. Orwant, L. Teodosio, and N. Bramson, Newspace: Mass media and personal computing, in *Proceedings Summer 1991 Usenix Conference, Nashville, TN, June 1991,* pp. 329–348.
7. J. L. Bentley, Multidimensional binary search trees used for associative searching, *Commun. ACM* **18**(9), 1975, 509–517.
8. A. D. Bimbo, M. Campanai, and P. Nesi, A three-dimensional iconic environment for image database querying, *IEEE Trans. Software Eng.* **19**(10), 1993, 997–1011.
9. N. S. Chang and K. S. Fu, Picture query language for pictorial database systems, *IEEE Computer* Nov. 1981, 23–33.
10. S. Chang and Y. Li, Representation of multi-resolution symbolic and binary pictures using 2DH strings, in *Proceedings IEEE Workshop Languages for Automation, 1988,* pp. 190–195.
11. S. K. Chang, Visual reasoning for information retrieval from very large databases, in *IEEE Workshop on Visual Languages, Rome, Italy, Oct.* 1989.
12. S. K. Chang and T. Kunii, Pictorial database systems, *IEEE Computer* Nov. 1981, 13–21.
13. S. K. Chang, Q. Y. Shi, and C. W. Yan, Iconic indexing by 2-D strings, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(3), 1987, 413–427.
14. M. Chock, A. F. Cardenas, and A. Klinger, Database structure and manipulation capabilities of the picture database management system (PICDMS), *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(3), 1984, 484–492.
15. D. Comer, The ubiquitous B-tree, *Comput. Surveys* **11**(2), 1979, 121–137.
16. G. Davenport, T. G. Aguierre Smith, and N. Pincever, Cinematic primitives for multimedia, *IEEE Comput. Graphics Appl.* July 1991, 67–74.
17. M. Davis, "Media streams: An iconic visual language for video annotation, in *Proceedings IEEE Symposium on Visual Languages, Bergen, Norway, 1993,* pp. 196–202.
18. E. Deardorff, T. D. C. Little, J. D. Marshall, D. Venkatesh, and R. Walzer, Video scene decomposition with the motion picture parser, *IS&T/SPIE* **2187,** Feb. 1994, 44–55.
19. C. Faloutsos, Access methods for text, *Comput. Surveys* **17**(1), 1985, 49–74.
20. C. Faloutsos, Signature files: Design and performance comparison of some signature extraction methods, in *Proceedings ACM SIGMOD, May 1985,* pp. 63–82.
21. C. Faloutsos and S. Christodoulakis, Description and performance analysis of signature file methods for office filing, *Proc. ACM TOIFS* **5**(3), 1987, 237–257.
22. C. L. Fennema, and W. B. Thompson, Velocity determination in scene containing several moving objects, *Comput. Graphics Image process.* **9**(4), 1979, 310–315.
23. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: The QBIC system, *IEEE Computer* **28**(9), 1995, 23–32.
24. W. I. Grosky, P. Neo, and R. Mehrotra, A pictorial index mechanism for model-based matching, *Data Knowledge Eng.* **8,** 1992, 309–327.
25. V. Gudivada, V. Raghavan, and K. Vanapipat, A unified approach to data modeling and retrieval for a class of image database applications, in *Multimedia Database Systems: Issues and Research Directions* (S. Jajodia and V. Subrahmanian, Eds.), Springer-Verlag, New York, 1995.
26. A Guttman, "R-Trees: A dynamic index structure for spatial searching, in *Proceedings ACM SIGMOD, 1984,* pp. 47–57.
27. R. A. Gustafson, Elements of the randomized combinatorial file structure, in *Proceedings of the ACM SIGIR Symposium on Information Storage and Retrieval, Apr. 1971,* pp. 163–174.
28. A. Hampapur, R. Jain, and T. Weymouth, Digital video segmentation, in *Proceedings 2nd ACM International Conference on Multimedia,* pp. 357–364.
29. T. Hamano, A similarity retrieval method for image databases using simple graphics, in *IEEE Workshop on Languages for Automation, Symbiotic and Intelligent Robotics, University of Maryland, Aug, 29–31, 1988,* pp. 149–154.
30. K. Hirata and T. Kato, Query by visual example, in *Proceedings 3rd International Conference on Extending Database Technology, Vienna, Austria, Mar. 1992,* pp. 56–71.
31. ISO/IEC, Information technology—Generic coding of moving pictures and associated audio information, in *ISO/IEC DIS 13818-2, 1994.*
32. H. V. Jagdish, A retrieval technique for similar shapes, in *Proceedings ACM SIGMOD, May 1991,* pp. 208–217.
33. R. C. Jain, Segmentation of frame sequence obtained by a moving observer, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**(5), 1984, 624–629.
34. T. Joseph and A. F. Cardenas, PICQUERY: A high level query