JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 8, No. 2, June, pp. 146–166, 1997
ARTICLE NO. VC970355

Review of Image and Video Indexing Techniques

F. Idris and S. Panchanathan*

Visual Computing and Communications Laboratory, Department of Electrical Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5

Received May 17, 1996; accepted March 25, 1997

Visual database systems require efficient indexing to facilitate fast access to the images and video sequences in the database. Recently, several content-based indexing methods for image and video based on spatial relationships, color, texture, shape, sketch, object motion, and camera parameters have been reported in the literature. The goal of this paper is to provide a critical survey of existing literature on content-based indexing techniques and to point out the relative advantages and disadvantages of each approach. © 1997 Academic Press

1. INTRODUCTION

Multimedia information systems are becoming increasingly important with the advent of broadband networks, high-powered workstations, and compression standards. There are several applications including distance learning, telemedicine, interactive television, digital libraries, multimedia news, and geographical information systems which are already dominated and are expected to be increasingly populated by visual (images and video) information. Since visual media requires large amounts of memory and computing power for storage and processing, there is a need to efficiently index, store, and retrieve the visual information from multimedia databases.

Several indexing techniques for factual data have been presented in the literature [1, 2]. Content-based searching has also been explored and used for text data [2]. A straightforward approach to indexing image and video is to represent the visual contents in textual form (e.g., keywords and attributes). These keywords serve as indices to access the associated visual data. This approach has the advantage that visual databases can be accessed using standard query languages such as SQL. However, this entails extra storage and needs a large amount of manual processing. A more serious consideration from the point of view of reliability is that the descriptive data (i) do not conform to a standard language, (ii) are inconsistent, and (iii) might not capture the image/video content. Thus the retrieval results may not be satisfactory since the query is based on features that have been inadequately represented. Another approach to indexing visual data is to apply image analysis/understanding techniques. Image pattern recognition techniques are first used to classify an image/video into one of several categories. Interpretation to each class is then provided using knowledge bases. For example, the shape features in a scene may be mapped into symbols which represent elementary shapes such as circles, rectangles, etc. Semantics of the scene is developed by interpreting the collection of symbols. This is accomplished by using visual models and rules that imitate the human understanding. However, this is a computationally intensive and complex task. As a result, there has been a new focus on developing image/video indexing techniques which (i) have the capability to retrieve visual data based on their contents, (ii) are domain independent, and (iii) can be automated.

Recently, several image indexing techniques have been proposed in the literature. Two approaches have been commonly used: (i) a content indexing approach, where the index terms serve to encode the content of images; and (ii) a structural approach where images are represented as a hierarchy of regions, objects, and portions of objects. The content indexing approach is based on features such as color [8], texture [25], shape [42], and sketch [48] extracted from an image which essentially serve as the index. On the other hand, the structural approach is based on spatial relationships between objects or regions in a scene [49].

Video has both spatial and temporal dimensions and hence a good video index should capture the spatiotemporal contents of the scene. In order to achieve this, a video is first segmented into elemental scenes called shots. A shot is a sequence of frames generated during a continuous operation and therefore represents a continuous action in time or space [104]. The purpose of the segmentation process is to partition the video stream into a set of meaningful and manageable segments, which then serve as basic units for indexing. A number of algorithms for video segmenta-

* E-mail: panch@elg.uottawa.ca.

tion in both compressed and uncompressed domains based on variables such as intensity/color template matching, histograms, and motion vectors have been reported in the literature [69–85].

Aigrain and Joly [89] have reviewed the different types of visual content analysis, representation, and their application in indexing, retrieval, abstracting, relevance assessment, and interactive perception, and they survey some of the research problems. Ahanger and Little [110] have presented research trends in multimedia applications, the requirements of future data delivery systems, and some video segmentation techniques. In this paper, we present a comprehensive survey of existing literature on content-based indexing techniques and point out the relative advantages and disadvantages of each approach. The paper is organized as follows. An overview of visual storage and retrieval system is presented in Section 2. Image indexing is reviewed in Section 3. Section 4.1 presents video segmentation techniques which is followed by video indexing techniques using spatial and temporal features in Sections 4.2 and 4.3, respectively. Finally, the summary and conclusions are presented in Section 5.

2. VISUAL STORAGE AND RETRIEVAL SYSTEM

The access to visual database has two main components: storage and retrieval. In the storage process, images and video are processed to extract features which describe their semantics. The extracted features are then represented, organized, and stored in the database. In the retrieval process, the system analyzes a query and extracts the appropriate feature vector and then a search process is performed. The search process is carried out by computing the "similarity" (using a similarity metric) between the feature vector of the query and those of the candidate images and video stored in the database. The retrieved images and video are presented to the user in the descending order of the similarity to the query.

Several image and video database systems have been proposed in the literature [11, 12]. An architecture for a generic image/video database system is shown in Fig. 1. It consists of the user interface, content-based retrieval, organization, and database management modules. A functional description of each module is presented below.

1. *User interface:* In visual information systems, user interaction plays an important role in almost all of its functions (e.g., semiautomatic and manual feature extraction, navigation, selection, and refinement). The user interface consists of a query processor and a browser to provide the interactive graphical tools and mechanisms for querying and browsing the database, respectively. The query processor provides the means to query images and video using a variety of query methods and interfaces. A query can range from a simple keyword-based to a complex query

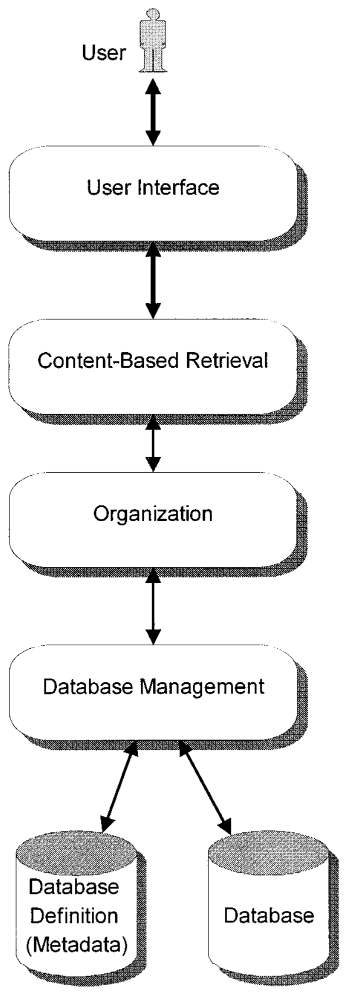


FIG. 1. Storage and retrieval of images and video.

where the user specifies a sketch or an object track. In contrast to textual database systems, image and video databases are required to evaluate properties of the data specified in a query. For example, to retrieve all images similar to a query image based on color, the color attributes (e.g., color histogram) of the query image must be calculated. After obtaining the responses to a query, the browser is used to display the results. The browser allows users to further refine and navigate through the database visually.

2. *Content-based retrieval module:* As shown in Fig. 2, the content-based retrieval module consists of the following:

• *Scene Change Detection:* The first step in video indexing is to decompose a video sequence into shots. Once a video sequence is segmented into shots, a set of representative frames is then selected to represent the shot [87, 88]. Each shot is represented using spatial and temporal features. The spatial features refers to the spatial content of the representative frame of a shot, while the temporal

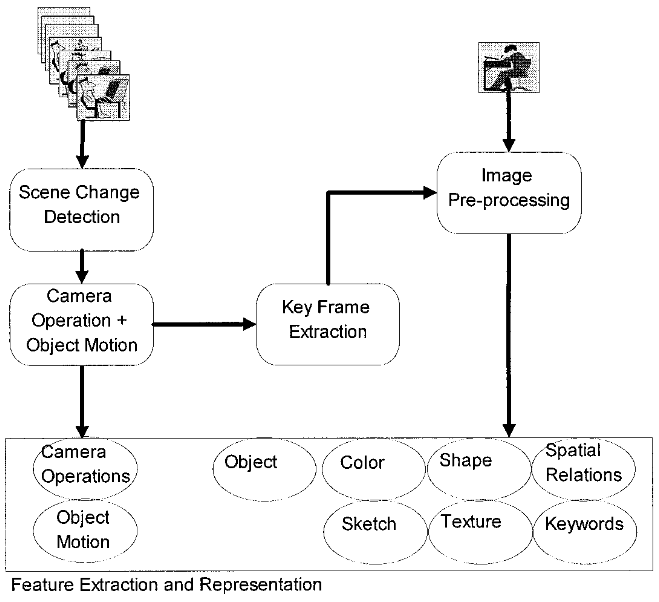


FIG. 2. Content-based retrieval module.

features represents the temporal content of a shot. The representative frames of a shot are fed to the image preprocessing stage in order to generate the spatial features of the shot, while the shots are input to the feature extraction and representation stage in order to extract the temporal features.

• *Image preprocessing:* The image is first processed in order to extract the features which describe its contents. The processing might involve decompression, enhancement, filtering, normalization, segmentation, and object identification. If the input image is in the compressed form, decompression is required to facilitate execution of the pixel domain algorithms. The output of the image preprocessing stage is typically a collection of objects and regions of interest.

• *Feature extraction and representation:* In this stage, the semantics of image/video content are extracted and represented. Features (of the objects, regions, and/or the whole image) such as texture, color, etc., are used to describe the content of a still image. For video, the spatial features are generated using still image techniques [87, 88], while the temporal features are extracted based on motion and/or camera operations within the shot [98]. Image and video features can be classified into primitive and logical features [7]. Primitive features such as color, shape centroids, etc., are quantitative in nature and can be extracted automatically or semiautomatically. Logical features are qualitative in nature and provide abstract representations of visual data at various levels of detail. Typically, logical features are extracted manually. One or more features can be used in a specific application. For example, in a satellite information system, the texture features are important, while shape and color features are more important in trademark registration systems. Once the features have been extracted, the textual, numerical, alphanumerical, etc., index keys are derived.

3. *Organization:* Efficient query processing necessitates the organization of image/video indices such that efficient search strategies can be used. We note that image/video indices are approximately represented, may have interrelated multiple attributes and may not have an embedded order [102]. Therefore, conventional indexing structures like B-tree and hashing, etc., cannot be used for the organization of image/video indices. Flexible data structures should be used in order to facilitate storage/retrieval in visual information systems. Structures such as R-tree family [105], R^*-tree [106], quad-tree [107], and grid file [108] are commonly used. Each structure has its advantages and disadvantages; some have limited domains and some can be used concurrently with others. Niu *et al.* [109] have discussed some issues concerning novel indexing structures for image retrieval. Ahanger and Little [110] have also presented a review of indexing structures for video.

4. *Database management module:* The database management module provides internal level physical storage structure and access path to the database. The database management module has the following characteristics: (i) provides insulation between programs and data, (ii) provides users with a conceptual representation of the data, (iii) supports multiple views of the data, and (iv) ensures data consistency.

3. IMAGE INDEXING

In image databases, we are generally interested in searching through a large number of images to find an image similar to the input image, known as the query image. We recall from Section 1 that keyword-based systems and classical pattern matching techniques cannot be used to solve this problem. In this section, we present a review of image indexing techniques.

3.1. Color

Color is an important attribute for image representation. The color distribution of an image is typically represented using the image histogram. The histogram of an image f_n is an N-dimensional vector $\{H(f_n, i);\ i = 1, 2, \ldots, N\}$, where N is the number of colors (bins) and $H(f_n, i)$ is the number of pixels having color i. Histograms are invariant to image rotation, translation and viewing axis [9]. In image indexing using histogram [8–11], the histograms are the feature vectors used as image indices. We note that a similarity measure is used in the histogram space to measure the similarity of two images.

Given a pair of images, f_n and f_m, the similarity between the two images may be measured using the normalized intersection of their histograms

$$\frac{\sum_{i=1}^{N} \min(H(f_n, i), H(f_m, i))}{\sum_{i=1}^{N} H(f_m, i)}. \tag{1}$$

It has been shown that this metric (Eq. (1)) is fairly insensitive to changes in image resolution, histogram size, occlusion, depth, and viewpoint [8]. However, histogram intersection does not consider the perceptual similarity between the different bins. A metric which takes into account the similarity between the bins is defined as

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} [H(f_n, i) - H(f_m, i)][H(f_n, j) - H(f_m, j)], \tag{2}$$

where the weight a_{ij} denotes the cross correlation between the colors corresponding to bins i and j. We note that the metric in Eq. (2) has a higher computational complexity than the histogram intersection. However, it is closer to human judgment of color similarity.

The color histogram requires additional storage space and a large amount of processing. For an image of size $X \times Y$, the histogram calculation requires $O(XY)$ additions and $O(XY)$ increments. In addition, $O(N)$ operations are required to compare a pair of histograms.

To decrease the computational complexity, the number of bins should be reduced. The first approach to reduce the number of bins is to represent the red (R), green (G), and blue (B) components using the RG, BY, and WB color axes as follows [8]:

$$RG = R - G \tag{3}$$

$$BY = 2 \times B - R - G \tag{4}$$

$$WB = R + G + B. \tag{5}$$

This representation allows the intensity (WB) to be more coarsely quantized than RG and BY. A histogram of 2048 bins is obtained if RG and BY are divided into 16 sections, while WB is divided into 8 sections.

The second approach is to quantize the color space into K-levels using a minimum sum of squares clustering algorithm [11]. Thus, the color space is partitioned into K cells, each corresponding to a histogram bin. The image histogram is the normalized count of the number of pixels in each cell. Wan and Kuo [19] have studied the color quantization in different color spaces including RGB, HSV, YUV, and Munsell spaces. To optimize the mean square quantization error along each axis independently, a Lloyd–Max quantizer is applied to each axis of the color space separately. It has been shown that the quantization in the HSV space gives a much better retrieval rate than other color spaces.

The third approach to reducing the number of bins is based on the observation that a small number of bins capture the majority of pixel counts in a histogram [9, 10]. Therefore, only the bins with the largest counts are compared. Experiments have shown that this approach results in a marginal degradation in performance [9, 10].

An alternative approach to reducing the computational complexity in color indexing is to use the dominant features of a histogram. A color distribution of an image is interpreted as a probability distribution which can be characterized by its moments [13]. If the first three moments of each color component are used, only nine floating point numbers are required to represent each index. The similarity metric is a weighted sum of the absolute differences between the corresponding moments. This approach outperforms the approaches based on reducing the number of bins in terms of storage requirements, retrieval speed, and robustness [13]. However, the use of low-order moments is sensitive to changes in illumination.

Another approach to reduce the computational complexity is to use a lower dimensional histogram or a lower complexity metric to filter a large fraction of the database. A higher-dimensional histogram or a higher complexity metric can then be applied to the small set of retrieved images. Vellaikal and Kuo [16] have proposed representing the color histogram at different resolutions. Here, the histogram is decomposed using the three-dimensional Haar wavelet basis functions. In the search process, the top W wavelet coefficients of the histogram of the query image are compared. Hafner *et al.* [21] have proposed the use of a lower dimensional and computationally simple distance measure based on Eq. (2). This technique not only reduces the complexity of the search process, but can also be implemented in a hierarchical manner, where a finer match can be obtained by increasing the number of coefficients employed in the search process.

We note that in the previous techniques a histogram describes the entire image content, without taking into account the location of the colors. This may result in unsatisfactory retrievals (e.g., false positives). For example, an image with a white car on the left might be matched to an image with white birds on the right. This problem can be eliminated by incorporating spatial information in the content representation. Gong *et al.* [10] have proposed the use of local histograms. Here, an image is partitioned into 9 (3×3) subimages. For each subimage, the color histogram is generated. The content of the image is represented by the histogram of entire image and the histograms of the subimages. This technique captures the locality of colors in an image. Stricker and Dimai [17] have proposesd a technique where the image is partitioned into 5 overlapping fuzzy regions. The histogram for each region is calculated and is represented by the first 3 moments. The 15 moments are used to represent the image. The fuzziness of the regions make the feature vector insensitive to small rotations of an image. In addition, a similarity function which exploits the spatial arrangement of the 5 regions is employed resulting in invariance of retrieval results with respect to rotations of 90° around the center of an image.

Including spatial information in the color representation of an image will not only improve the retrieval rate but also allow user queries based on the color of a subimage. For example, queries such as "retrieve all images with white birds on the right" could be answered. However, this technique requires the use of efficient segmentation and representation of the subimages.

3.2. Texture

Texture is an important feature of a visible surface where repetition or quasirepetition of fundamental pattern occurs. Texture features such as contrast, uniformity, coarseness, roughness, regularity, frequency, density, and directionality provide significant information for scene interpretation and image classification [22]. Texture modeling and classification techniques can be grouped into structural, statistical, and spectral methods [5].

Structural techniques characterize textures as weak or strong textures according to the spatial interaction between primitives [22]. A primitive is a connected set of cells characterized by gray level, shape, or homogeneity. Statistical methods categorize textures as smooth, fine, coarse, granulated, rippled, regular, irregular, or linear. Statistical techniques include autocorrelation function, cooccurrence matrices, and random field models. Spectral methods are based on the properties of Fourier transform, where a texture image may be analyzed by identifying high-energy and narrow peaks in the spectrum. In image indexing, a texture pattern is analyzed in order to extract the texture features, and the features are represented and used as an index. In this section, a survey of texture modeling and classification techniques that have been employed in image databases is presented.

Karhunen–Loève (KL) transform has been studied extensively for image compression applications [5]. Although KL is optimal, it is not widely used due to its high computational complexity. A technique based on the principal components analysis (Karhunen–Loève transform) combined with DFT (discrete Fourier transform) has been applied in the Photobook [28]. Let $\mathbf{x}_i$, $i = 1, 2, \ldots, N$ be the vector representing the DFT magnitude of the ith image in a training set of M images. The covariance matrix, $\mathbf{C}$, of the training set is estimated. Each principal component is an eigenvector of the covariance. The computational complexity is reduced by noting that $\mathbf{C}$ can have at most M eigenvectors. The KL transform coefficients of the images to be stored in the database are obtained using the M eigenvectors. The transform coefficients are the features used for comparing textures. KL transform has two advantages. First, vector components are decorrelated, and second, components are compressed into a small number of coefficients. In addition, the use of DFT magnitudes makes the transform coefficients invariant to spatial translation. However, the performance may be degraded for images outside the training set.

Picard and Liu [27] have presented a technique based on Wold decomposition. If an image is assumed to be a homogeneous and regular 2-D random field, then the 2-D Wold-like decomposition is a superposition of three orthogonal components: a purely indeterministic field $u(n, m)$, a generalized evanescent field $v(n, m)$, and a harmonic field $w(n, m)$. This model provides a description of textures in terms of periodicity, directionality, and randomness. Each one of these attributes is associated with the prominence of a different component. The conspicuous components in periodic, directional, and less structured textures are $w(n, m)$, $v(n, m)$, and $u(n, m)$, respectively [27]. Simulations on about a 1000 test images (cropped from 112 Bordatz texture images [30]) have shown that the Wold-based parameters are closer to human judgment of texture similarity than the principal components approach.

Tamura *et al.* [29] have proposed the use of coarseness, contrast and directionality as texture features. *Coarseness* is a measure of the granularity of the texture. *Contrast* can be measured based on the gray-level distribution. *Directionality* determines whether or not the image has a certain direction. A modified set of the Tamura features has been used in the QBIC project [11].

Several techniques based on features derived from a multiresolution representation of the texture image have been reported [31–35]. Here, the image pattern is decomposed into a set of different resolution subimages using Gabor decomposition [31–33], multiresolution simultaneous autoregressive model (MR-SAR) [34], or wavelet transform [35]. Features such as the mean and standard deviation [31–33], MR-SAR parameters [34], or statistics based on first-order distribution of gray levels modeled as a hidden Markov model [35], associated with each subimage are used to construct the index. The combination of MR-SAR model with Tamura's coarseness features and gray level histograms gives better performance than MR-SAR [34]. However, the improvement in retrieval rates is at the expense of increasing the size of the feature vector which increases the complexity of indexing and searching.

Rao *et al.* [36] have studied the relationships between categories of texture images and texture words (in the English language). Here, a set of 56 texture images from the Brodatz album and a set of 98 texture words (e.g., asymmetrical, bumpy, irregular, porous, and ribbed) are used. Ten human subjects are asked to describe the texture image from the set of texture words. Another 10 human subjects are given the opposite, i.e., identification of texture images corresponding to texture words. Pearson's coefficient of contingency, which measures the degree of association, was determined to be 0.63 and 0.56 for the association mapping of images to words and words to images, respectively. This study shows the major groups in the textual texture space and identifies the important dimensions to be covered in the design of a textual interface.

Retrieval by texture is useful when the user is interested in retrieving texture images which are similar to the query image. However, the use of texture features in a general visual database system requires texture segmentation (which remains a challenging problem) and the combination of texture features with spatial information.

3.3. Sketch

Another approach to describing the content of an image is by using a sketch. A sketch is an abstract image which contains the outline of objects. In this technique, for each image to be stored in the database, a sketch image is generated and stored. Typically, a sketch is created by using edge detection, thinning, and shrinking algorithms. The sketch is used as a key to retrieve the desired images from the database. The similarity of two images is measured by using the similarity of their sketches.

A technique for sketch-based image retrieval has been proposed by Kato *et al.* [48] and is implemented in the QBIC [11]. Each color image is converted into luminance and chrominance components. An edge operator is applied to the luminance component to compute the binary edge image. The edge image is reduced to 64×64 pixels and the reduced image is thinned. Query processing is performed by matching the user-drawn sketch to the sketches stored in the database. The matching process between the query image and a candidate image is executed as follows. The query sketch is first reduced to 64×64 pixels and divided into blocks of 8×8 pixels. Each block in the query image is correlated with the blocks within a search area in the candidate image. The size of the search area is 16×16 pixels and is centered around the block.

The main disadvantage of this approach is that it is orientation and scale dependent. Similar images with different orientation or scale will not be retrieved when compared with the query image. This problem can be eliminated by using sophisticated edge representation and matching algorithms.

3.4. Shape

The shape of an object refers to its profile and physical structure [5]. Shape features are fundamental to systems such as medical image databases, where the color and textures of objects are similar. In general, shape features can be represented using traditional shape analysis such as invariant moments [37], Fourier descriptors [38], autoregressive models [39], and geometry attributes [5]. However, in image storage and retrieval applications, shape features can be classified into global and local features.

Global features are the properties derived from the entire shape. Examples of global shape features are roundness or circularity, central moments, eccentricity, and major axis orientation. Roundness or circularity (γ) is defined in terms the perimeter (P) and the area (A) of a shape

$$\gamma = \frac{P^2}{4\pi A}, \tag{6}$$

where P and A may be calculated as the number of pixels on the boundary and the number of pixels in the binary image that are set to 1, respectively [11]. Central moments are given by

$$\mu_{p,q} = \sum_{(m,n)\in R} \sum (m - \overline{m})^p (n - \overline{n})^q, \tag{7}$$

where m and n are the center of mass of a shape represented by region R. Eccentricity is defined as the ratio of $R_{\max}$ to $R_{\min}$, where $R_{\max}$ and $R_{\min}$ are the maximum and minimum distances, respectively, from the center of mass to the boundary. In general, global features are robust to distortion, however, they cannot handle occluded shapes. Eakins *et al.* [46] have developed the ARTISAN shape retrieval system (automatic retrieval of trademark images by shape analysis). Edge detection techniques are applied to the trademark images to derive a set of region boundaries. The boundaries are approximated as a sequence of straight line and circular arc segments. The boundaries are then grouped into families using proximity and shape similarity criteria. Shape features for retrieval are extracted from the entire image, each boundary family, and each individual boundary. Eight global features were used (circularity, aspect ratio, discontinuity angle irregularity, length irregularity, complexity, right angledness, sharpness, and directedness).

Local features are those derived by partial processing of a shape and do not depend on the entire shape. Examples of local features are size and orientation of consecutive boundary segments [44], points of curvature, corners, and turning angle. The turning function, $\theta(s)$, measures the tangent angles as a function of arc length s measured from a reference point on the boundary. Mathematically,

$$\theta(s) = \int k(s)ds, \tag{8}$$

where $k(s)$ is the curvature function. Gary and Mehrotra

[43] have presented a shape similarity technique based on local boundary features encoded as multidimensional points. Although local features can be used for occluded shapes, they are noise sensitive. Ang *et al.* [47] have presented multidimensional feature measures of object shapes and feature blobs for retrieval of ceramic artifacts. Object shape is represented by boundary eccentricity and region compactness, moment, and convexity. High detailed regions are represented by number of blobs, dispersion of blobs, central moment of blobs, and total blob size.

Retrieval by shape similarity is a difficult problem because of the lack of mathematically exact definition of shape similarity which accounts for the various semantic qualities that humans assign to shapes. The majority of such techniques have been aimed at retrieving simple 2-D image objects capable of being represented by a single shape boundary. Recently, Scassellati *et al.* [42] have studied shape similarity using algebraic moments, spline curve distances, cumulative turning angle, sign of curvature, and Hausdroff distance and compared it to human perception. It has been shown [42] that turning angle, sign of curvature, and algebraic moments most closely match human judgment.

3.5. *Spatial Relationships*

In this technique, objects and the spatial relationships among objects in an image are used to represent the content of an image. First, each image is converted into a symbolic picture. The symbolic pictures are then encoded typically using 2-D strings which are stored in the database [49, 55]. Queries are expressed in the same 2-D string notation. The problem of image retrieval thus becomes a problem of 2-D sequence matching.

The basic algorithm for image indexing using spatial relationships was presented by Chang *et al.* [49]. To start with, the objects in an image are segmented and recognized. The image is converted into a symbolic picture, where the objects are represented using a set of symbols S. The position of an object in the symbolic picture is determined by the object's centroid. For example, the symbolic picture corresponding to the image in Fig. 3a is shown in Fig. 3b. The relationships among the objects in an image are expressed using the set of operators $A = \{<, =, :\}$. The symbol $<$ denotes the left-right or below-above spatial relationship. The $=$ stands for "at the same spatial location as" and the symbol : denotes the relation "in the same set as." A 2-D string over S is defined as

$$(x_1 y_1 x_2 y_2 \ldots x_n y_n, x_{p(1)} z_1 x_{p(2)} z_2 \ldots x_{p(n)} z_n), \qquad (9)$$

where $x/x_2 x_3 \ldots x_n$ is a 1-D string over $S (n \geq 0$ and $x_i \in S)$, $p(.)$ is a permutation over $\{1, \ldots, n\}$, $y_1 y_2 y_3 \ldots y_n$ and $z_1 z_2 z_3 \ldots z_n$ are 1-D strings over A. For example, in Fig. 3a, a and b are to the left of c, c is above b, and b is above a; the image can be represented using the 2-D string $\{a = b < c, a < b < c\}$.

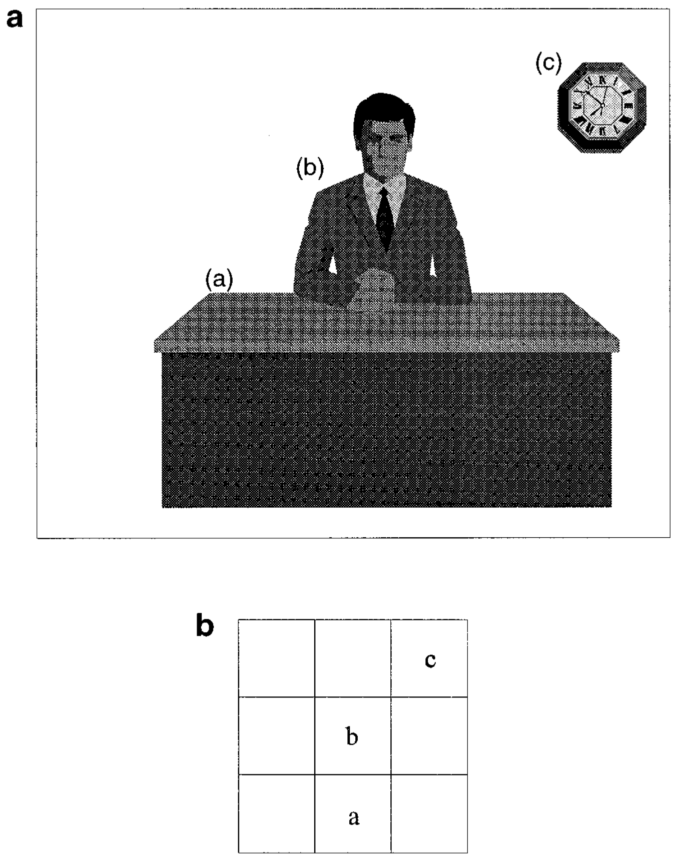


FIG. 3. Example of image representation using 2-D string. (a) An image containing three main objects; (b) symbolic picture which represents the image (a).

Matching 2-D strings is based on a ranking scheme for the object symbols in the strings. The rank values of objects represent the relative ordering in the strong representing the spatial positioning in an image. The rank of a symbol in a 2-D string is defined to be one plus the number of $<$ preceding it.

The matching algorithm is simple; however, for images with large number of objects, (i) the 2-D string representation is complex, and (ii) the spatial operators ($A = \{<, =, :\}$) are not sufficient to give a complete description of the spatial relationships among the objects. To increase the range of relationships that can be expressed, especially among overlapping objects, Jungert [50] has extended the basic 2-D string (2-D E-string) by adding a set of new local symbols. These local operators are \, %, |, |=, =|, /, =, and <, which represent the relations "overlap inverse," "contain," "meet," "begin," "end," "overlap," "equal," and "less than," respectively. Although, the local symbols facilitate a more precise description of binary relations

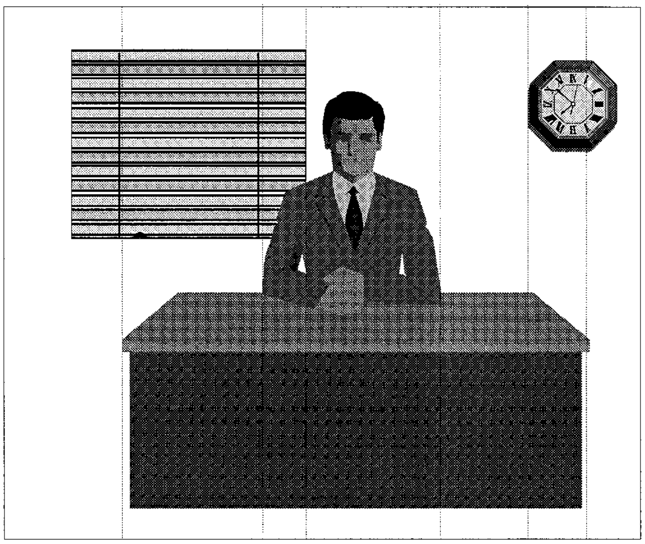


FIG. 4. The cutting lines of a 2-D G-string along the *x*-axis.

among objects, they cannot be unified with the global 2-D string representation of symbolic pictures [53].

To overcome the problems of 2-D string and E-string, Chang *et al.* [51, 52] have presented a generalization of the 2-D string called 2-D G-String. G-string employs a cutting mechanism to solve the problems of representing overlapping objects. Each object is first cut against the extreme points of all other objects in the image along the horizontal and vertical axis. Thus, every object is divided into smaller subobjects at the boundary lines of other overlapping objects. The spatial relationships among the subobjects rather than the objects is expressed using 2-D strings. Fig. 4 shows an example of the cutting lines for an image containing four objects. For images with many objects, the number of subobjects becomes very large. This increases the storage requirements as well as the complexity of comparison. To reduce the number of cuts in an image, Lee and Hsu [53] have presented a spatial knowledge representation known as 2-D C-string. This representation includes a set of spatial operators as the G-string and in addition has the "begin" and "end" operators. The cut is performed at the point of partial overlap instead of at the extremes of each object. This reduces the number of cuts required. Fig. 5 shows the cutting lines for a 2-D C-string. It can be seen from Figs. 4 and 5 that a 2-D C-string requires less cutting lines than a 2-D G-string. Hence, a 2-D C-string is more efficient than a 2-D G-string in terms of complexity and storage requirement. Although a 2-D C-string is more efficient than a 2-D G-string, it still partitions the objects in an image in order to obtain the spatial relationships. For images with large number of objects, this increases both the storage and processing requirements. To overcome this problem, Lee *et al.* [54] have proposed the 2-D B-string for image indexing without the need for object partitioning. Let S be the set of object symbols where each symbol could represent the begin boundary or end boundary of an object. The special symbol = which is not in S is used to indicate that the projections are at the same spatial location. A 2-D B string over S is defined as (u, v), where u and v are 1-D strings over $\{=\}$ representing the relative ordering of the boundaries of the objects projected along the horizontal and vertical axis, respectively. For example, the 2-D B-string representing the image in Fig. 3a is $(abbcac, aa = bcbc)$. The object symbols in a B-string are assigned a rank:

$$rank(s_i) = \sum_{j=1}^{i} order(s_j), \tag{10}$$

where

$$order(s_j) = \begin{cases} 1 & s_j \neq \text{“="} \\ -1 & otherwise. \end{cases}$$

Hence each object has two ranks: a begin rank and an end rank, along the horizontal and vertical axis.

Image and video data are voluminous; compression is essential for storage and transmission. Hence, the visual data in future multimedia databases is expected to be stored in the compressed form. In order to avoid the unnecessary decompression operation in the searching process, it is efficient to index the image/video in the compressed form. Compressed domain image/video indexing techniques based on compression parameters have been reported in the literature [60, 64]. These techniques combine image compression and image indexing and have a lower cost for computing and storing the indices. In the following

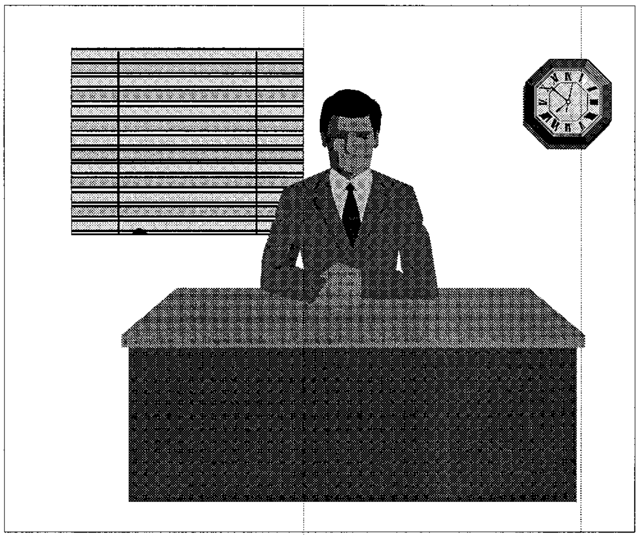


FIG. 5. The cutting lines of a 2-D C-string along the *x*-axis.

section, we present a review of compressed domain image/video indexing techniques.

3.6. *Indexing in the Compressed Domain*

Recently, the International Standards Organization has proposed the JPEG [56] and MPEG [57] standards for image and video compression, respectively. Chang [60] has proposed a texture-based indexing technique in the transform domain (DCT, subband, and wavelet). The energy of the subbands are used to define the texture feature sets. For an $N \times N$ DCT transform, N^2 bands are obtained using the DCT/Mandala transform. For a three-level wavelet transform, feature vectors with 10 terms are produced. The texture feature vector is reduced by using Fisher discriminant technique which reduces the search complexity. The transform domain feature elements of the input image are mapped to a set of eigenvectors with the maximum reparability significance. The Mahalanobis distance in the transformed feature space is used to measure the similarity between two images. It has been shown [60] that the classification rates of the wavelet transform and the uniform subband outperform the rates of the DCT/Mandala transform. Shneier and Abdel-Mottaleb [61] have proposed a technique for image retrieval using JPEG. Here, a set of $2m$ windows is selected. The windows are randomly paired, with the constraint that each window has only one partner. For each pair of windows, a bit is allocated in the m bit index. For each window the average of each DCT coefficient is computed giving a 64-dimensional feature vector. For each feature value and each window pair, the index is computed by comparing the values of the first window with those of the second window. If the difference is greater than a threshold the corresponding bit is set to 1 otherwise it is reset to 0. The similarity between two images is measured by computing the similarity between their keys. In contrast to the technique proposed by Chang [60] which is based on texture similarity, the similarity in this technique has no semantic meaning. Recently image indexing techniques using vector quantization (VQ) have been reported [62–64]. A VQ encoder maps each input vector onto one of a finite set of codewords (codebook) using a nearest neighbor rule, and the labels (indices) of the codewords are used to represent the input image. Hence, VQ is naturally an indexing technique. Two VQ-based techniques have been reported [62–64]. In the first technique [62], the images are compressed using vector quantization and the labels are stored in the database. The histograms of the labels are the feature vectors used as image indices. For an image of size $X \times Y$ pixels, this approach requires $O(XY/L)$ additions, $O(XY/L)$ increments, and $O(K)$ operations for comparing a pair of histograms, where K and L are the size of the codebook and the vector dimension, respectively. The use of the histogram of the labels requires about $1/L$ the number of operations required by the use of the histogram of the pixels in the uncompressed domain. In the second technique [63, 64], the images are compressed using adaptive VQ. Here, a large codebook is first generated. For each image to be stored in the database, the labels are determined using the codebook. A usage map is generated to indicate the used codewords. The labels and the usage map are stored in the database. We note that the used codewords reflect the contents of the image. The usage map corresponding to an image constitutes a feature vector which is used as an index to store and retrieve the image. We note that histogram of the label-based approach outperforms this approach in terms of retrieval rate. However, the computational complexity of the usage map based approach is $O(K)$ bitwise operations, where K is the size of the codebook [63, 64].

4. VIDEO INDEXING

A video sequence is a series of sequentially ordered (in time) images. Prior to storage in a database, a video stream is segmented into elementary units (shots) to be identified and indexed. Since video data consist of still images (frames), all the techniques presented in Section 3.1 are applicable to the individual frames of a video sequence (spatial features). In addition, video has temporal properties such as motion, camera operations, sequential composition, and interframe relationships. In this section, we present a review of video segmentation methods followed by video indexing using spatial and temporal features.

4.1. Video Segmentation

The structure within a video stems from the fact that video streams are formed by editing different video segments known as shots. A shot is a sequence of frames generated during a continuous operation and it represents continuous action in time and space [104]. Shots can be joined together in either an abrupt transition mode, in which two shots are simply concatenated, or through gradual transitions, in which additional frames may be introduced using editing operations such as dissolve, fade-in, fade-out, and wipe.

The purpose of the segmentation process is to partition a video stream into a set of meaningful and manageable segments as shown in Fig. 6, which then serve as the basic units for indexing. A number of algorithms for video segmentation in both the uncompressed and the compressed domains have been reported in the literature [69–91]. We now present an overview of these techniques.

4.1.1. *Segmentation in the Uncompressed Domain*

The different algorithms in the uncompressed domain can be broadly classified into four categories: template

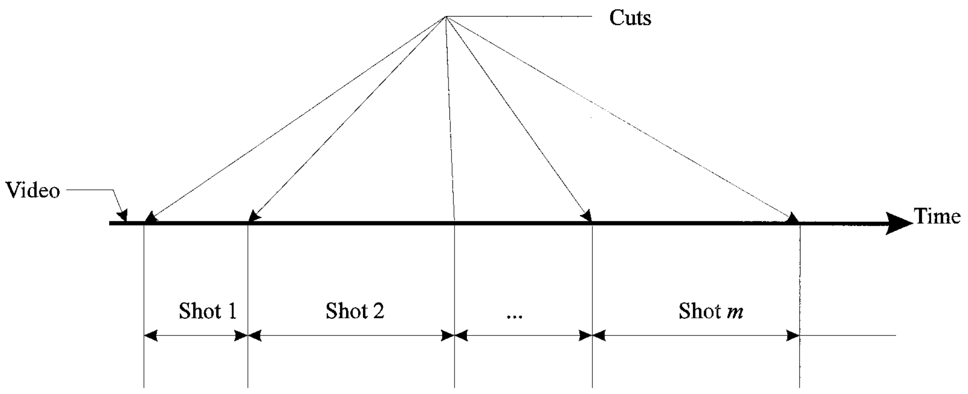


FIG. 6. Video segmentation.

matching, histogram-based techniques, twin-comparison, and block based techniques.

4.1.1.1. Intensity/color template matching. Scene change can be detected by emphasizing the spatial similarity between two frames [14, 69]. The simplest way to measure the spatial similarity between two frames f_m and f_n is using template matching, where each pixel at the spatial location (i, j) in f_m is compared with the pixel at the same location in f_n. Typically, the difference magnitude, $D(f_m, f_n, i, j)$, of f_m and f_n is used for comparison where

$$D(f_m, f_n, i, j) = \begin{cases} |P(f_m, I, i, j) - P(f_n, I, i, j)| & \text{for gray level images} \\ \sum_{l=1}^{3} |P(f_m, C_l, i, j) - P(f_m, C_l, i, j)| & \text{for color images,} \end{cases} \tag{11}$$

where $P(f_m, I, i, j)$ is the intensity of the pixel at (i, j) and $P(f_m, C_1, i, j)$, $P(f_m, C_2, i, j)$, $P(f_m, C_3, i, j)$ are the color components C_1, C_2, and C_3 of the pixel (i, j), respectively. For example, in case of using the RGB color coordinate system, C_1, C_2, and C_3 are equal to R, G, and B, respectively.

Nagasaka and Tanaka [14] have proposed the use of the sum of the difference magnitude

$$S_1(f_m, f_n) = \sum_{i=1}^{X} \sum_{j=1}^{Y} D(f_m, f_n, i, j). \tag{12}$$

A scene change is declared whenever $S_1(f_m, f_n)$ exceeds a prespecified threshold.

Zhang *et al.* [70] have presented an algorithm based on the number of changed pixels. A pixel is changed if $D(f_m, f_n, i, j)$ is greater than a certain threshold. A cut is detected if the percentage of the changed pixels is greater than a threshold.

We note that, using the previous metrics, it is difficult to distinguish between a small change in a large area and a large change in a small area. Therefore, template matching methods are sensitive to noise, object motion, and camera operations (e.g., panning and zooming) since they result in false detections.

4.1.1.2. Histogram-based techniques. We recall from Section 2.1, that the intensity/color histogram of a gray/color image f is an N-dimensional vector $\{H(f, i);\ i = 1, 2, \ldots, N\}$ where N is the number of levels/colors and $H(f, i)$ is the number of pixels of level/color i in the image f. The rationale behind histogram-based approaches is that two frames that exhibit minor changes in the background and object content will also show insignificant variations in their intensity/color distributions. In addition, histograms are invariant to image rotation and change slowly under the variations of viewing angle, scale, and occlusion [9]. Hence, this technique is less sensitive to camera operations and object motion compared to template matching-based techniques.

Tonomura [72] has proposed a technique based on the gray-level histogram difference. Let the histograms of frames f_m and f_n be denoted by $H(f_m, i)$ and $H(f_n, i)$, respectively. The sum of the histograms difference magnitude is defined as

$$S_2(f_m, f_n) = \sum_{i=1}^{N} |H(f_m, i) - H(f_n, i)|. \tag{13}$$

A cut is declared if $S_2(f_m, f_n)$ is greater than a threshold. Gargi *et al.* [75] have investigated the performance of the histogram-based method using different color spaces. The RGB histogram is computed as three sets of 256 bins. The other histograms are represented as a 2-D distribution over the two nonintensity dimensions as shown in Table 1. It has been found that using the sum of histograms difference magnitude, the YIQ, $L^*a^*b^*$, and Munsell coordinate sys-

TABLE 1
The Two Nonintensity Dimensions and Number of Bins Used in the Study by Gargi *et al.* [75]

Color space	Dimension used	Number of bins
YIQ	IQ	1600 (40 × 40)
L*a*b	a*b*	1600 (40 × 40)
L*u*v	u*v*	1600 (40 × 40)
HSV	SV	1800 (60 hues × 30 saturations)
Munsell	Hue and chroma	1800 (60 hues × 30 chromas)

tems perform well followed by the HSV, $L^*u^*v^*$, and RGB, respectively.

To strongly reflect the difference between two frames across a cut, Nagasaka and Tanaka [14] have proposed the use of the χ^2 test to compare two histograms $H(f_m, i)$ and $H(f_n, i)$. The χ^2 test is defined as

$$S_3(f_m, f_n) = \sum_{i=1}^{N} \frac{(H(f_m, i) - H(f_n, i))^2}{H(f_m, i)}. \quad (14)$$

If the difference $S_3(f_m, f_n)$ is larger than a given threshold, a scene boundary is declared. We note that $S_3(f_m, f_n)$ enhances the difference between cuts as well as changes due to object or camera motion [70]; however, the computational complexity of $S_3(f_m, f_n)$ is greater than that of $S_2(f_m, f_n)$.

Histogram-based techniques tend to lose scene changes which have small variations in their intensity distribution [73]. In addition, histogram comparisons may not reflect the content difference [76]. Hence, histogram-based techniques are not necessarily superior to template matching approaches.

4.1.1.3. Block-based techniques. We note that the techniques presented in Sections 4.1.1.1 and 4.1.1.2 use global attributes of images such as point by point differences and intensity or color histograms. Block-based techniques [76, 77] use local attributes to reduce the effect of noise and camera flashes. Here, each frame f_m is partitioned into a set of r blocks. Rather than comparing a pair of frames, every subframe in f_m is compared with the corresponding subframe in f_n. The similarity between f_m and f_n is measured using

$$S_4(f_m, f_n) = \sum_{i=1}^{r} C_i \times S_p(f_m, f_n, i), \quad (15)$$

where C_i is a predetermined weighting factor and $S_p(f_m, f_n, i)$ is a partial match obtained by comparing the ith region in f_m and f_n. Kasturi and Jain [69] have presented a metric based on statistical characteristics of the intensities of subimages. Corresponding blocks in two frames are compared using a likelihood ratio,

$$\frac{\left(\frac{\mu_{m,i} + \mu_{n,i}}{2} + \frac{\sigma^2_{m,i} - \sigma^2_{n,i}}{2}\right)^2}{\sigma^2_{m,i} \times \sigma^2_{n,i}}, \quad (16)$$

where $\mu_{m,i}$ and $\sigma^2_{m,i}$ are the mean and variance, respectively, of block i in frame f_m. If the likelihood ratio is greater than a threshold, $S_p(f_m, f_n, i)$ is set to 1. Otherwise, $S_p(f_m, f_n, i)$ is set to 0. A scene change is declared whenever the number of changed blocks is large enough (i.e., whenever $S_4(f_m, f_n)$ is greater than a given threshold and C_i is 1 for all i). Compared to the intensity/color template matching, this approach reduces the number of overdetected (incorrectly) camera breaks. This reduction is a result of the increased tolerance to slow camera and object movements. However, cuts may be misdetected between two frames that have similar pixel values, but different density functions.

To incorporate the advantages of spatial locality to histogram comparisons, Swanberg *et al.* [76] have proposed a technique where $S_p(f_m, f_n, i)$ is measured by comparing the color histograms of the subframes using

$$S_p(f_m, f_n, i) = \sum_{c \in \{R,B,G\}} \sum_{j=0}^{256} \frac{(H_c(f_m, j) - H_c(f_m, j))^2}{H_c(f_m, j) + H_c(f_n, j)} \quad (17)$$

and C_i is $1/r$ for all i.

Video segmentation is the identification of two types of segment boundaries (abrupt changes and gradual transitions) which take place over a sequence of frames. The previous techniques are based on a single threshold and lack the power of detecting gradual scene changes, since the frame to frame difference in a gradual transition is smaller than the threshold. Lowering the threshold results in both false detections and misdetections. We now review a technique for the detection of gradual scene changes.

4.1.1.4. Twin-comparison. Twin-comparison [70] has been proposed for the detection of gradual scene changes using a dual threshold. In the first pass, a high threshold T_c is employed to detect abrupt scene changes. In the second pass, a reduced threshold T_g is used to identify the potential starting frame f_s of a transition as shown in Fig. 7. Once f_s is identified, it is compared with subsequent frames, measuring the accumulated difference instead of the frame to frame difference. The end frame f_e of the transition is detected when the difference between successive frames decreases to less than T_g, while the accumulated difference becomes larger than T_c. If the consecutive frame difference falls below T_g before the accumulated difference exceeds T_c, then the potential starting point f_g

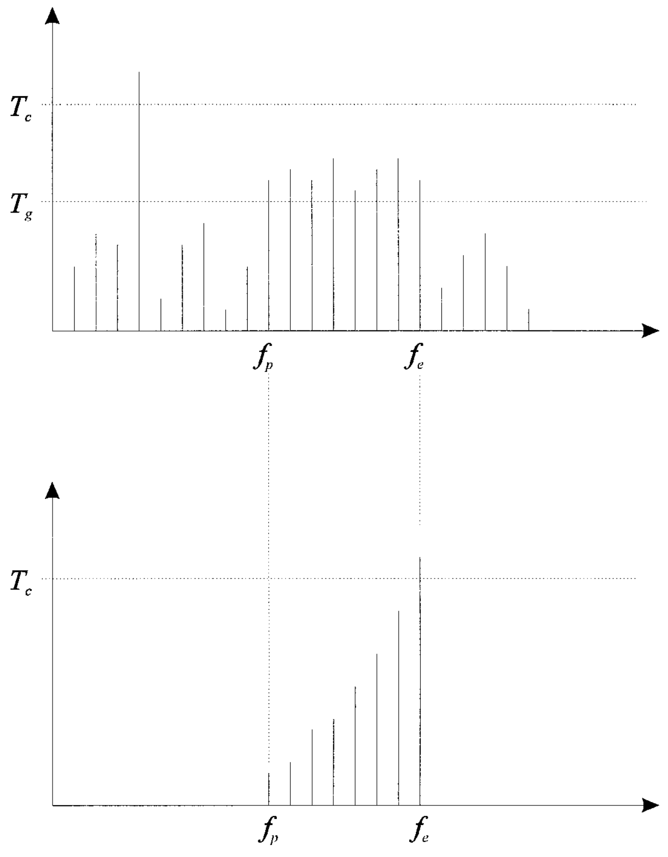


FIG. 7. Twin-comparison.

is dropped, and the search continues for other gradual transitions. Although the twin-comparison approach is effective in detecting gradual scene changes, however, the type of scene change (fade, wipe, . . . etc.) cannot be identified.

4.1.1.5. Model-based segmentation. In a video sequence, gradual transition from a scene to another is the result of the editing process. Editing has direct influence on how the audience responds to the video material, their interpretation, and their emotional reaction. Film editing is to be considered as not only the glue between shots, but also an essential contributor to the meaning conveyed by the video. Hence, it is not only important to identify the transition position, but also the type of the transition.

In model-based techniques, the problem of video segmentation is viewed as the process of locating the edit boundaries within the video sequence. Here, different edit types, such as cuts, translate, wipes, fades, and dissolves are modeled by mathematical functions. Hampapur *et al.* [90] have presented a model for the video edit. Let $S_1(x, y, t)$ and $S_2(x, y, t)$ be two shots that are being edited, and let $S(x, y, t)$ be the edited shot. All the chromatic processes can be described as a linear pixel intensity manipulation:

$$S(x, y, t) = S_1(x, y, t)\left(1 - \frac{t}{l_1}\right) + S_2(x, y, t)\left(\frac{t}{l_2}\right), \quad (18)$$

where l_1, l_2 are the length (in number of frames) for which the scaling of each of the two shots lasts. This technique is efficient for detecting chromatic edits which results from scaling the color space. However, it cannot be used in detecting other types of chromatic translations, rotation, etc. Aigrain and Joly [89] have presented a technique for the detection of scene boundaries based on a differential model of motion picture. The algorithm is based on an estimation of the density function for the difference between two frames.

4.1.2. Video Segmentation in the Compressed Domain

We note that the previous techniques are based on uncompressed data. The large channel bandwidth and memory requirements for the transmission and storage of visual data necessitates the use of compression techniques. Recently, several algorithms for video segmentation in the compressed domain have been reported [78–85]. According to the type of information used, the algorithms for video segmentation in the compressed domain are divided into four classes, namely, segmentation using DCT coefficients, motion vectors, motion/DCT, and subband decomposition.

4.1.2.1. DCT coefficients. The standards for image and video compression (JPEG, MPEG, and H.261) are DCT-based techniques [56–58]. The transform coefficients in the frequency domain are related to the spatial domain. Therefore, the DCT coefficients can be used for scene change detection in compressed video sequences.

Arman *et al.* [78, 79] have proposed a technique for scene change detection in motion JPEG using DCT coefficients. For each compressed frame f_m^I, B blocks are first chosen *a priori* from R connected regions in f_m^I. A set of randomly distributed coefficients $\{c_x, c_y, z, \ldots\}$ is selected from each block where c_x is the xth coefficient. A vector $Vf_m^I = \{c_1, c_2, c_3, \ldots\}$ is formed by concatenating the sets of coefficients selected from the individual blocks in R. The vector Vf_m^I represents f_m^I in the transform domain. The normalized inner product is used as a metric to judge the similarity of frame f_m^I to frame f_n^I

$$\psi = 1 - \frac{Vf_m^I \cdot Vf_u^I}{|Vf_m^I||Vf_n^I|} \quad (19)$$

A scene transition is detected if Ψ is greater than a threshold. In case of false positives, which result from camera and object motion, f_m^I and f_n^I are decompressed and their color histograms are compared to detect camera breaks [79]. Zhang *et al.* [80, 81] have presented a pairwise comparison technique in the transform domain similar to template matching techniques in the uncompressed domain. Here, the pairwise normalized absolute difference $D(f_m^I, f_n^I, i, j)$ of the (i, j) block in two frames f_m^I and f_n^I is determined using

$$D(f_m^I, f_n^I, i, j) = \frac{1}{64}\sum_{k=1}^{64} \frac{|c(f_m^I, k, i, j) - c(f_n^I, k, i, j)|}{\max(c(f_m^I, k, i, j), c\,(f_n^I, k, i, j))}, \tag{20}$$

where $c(f_m^I, f_n^I, i, j, k)$ is the kth coefficient of block (i, j) in f_m^I. If the difference $D(f_m^I, f_n^I, i, j)$ is larger than a threshold, the block (i, j) is considered to be changed. If the number of changed blocks exceeds certain threshold, a scene change in the video sequence from frame f_m^I to frame f_n^I is declared. Compared to the technique by Arman *et al.* [78], the processing time of this technique is less, however, it is more sensitive to gradual changes [81].

We note that the previous two algorithms are applied on video sequences compressed using motion JPEG. In case of MPEG video, only I-frames are compressed with DCT coefficients and hence the previous two techniques cannot be directly applied to the B- and P-frames. In addition, the techniques based on I-frames may result in false positives. To overcome these problems, Yeo and Liu [82] have proposed a unified approach for scene change detection in motion JPEG and MPEG. This algorithm is based on the use of only the DC coefficients. To start with, a DC frame f_m^{DC} is constructed for every frame in the sequence. The DC coefficients in JPEG and I-frames in MPEG are obtained directly from each block. For B- and P-frames in MPEG video the DC coefficients are estimated. The sum of the difference magnitude of the DC frames f_m^{DC} and f_n^{DC} is used as a measure of similarity between two frames, i.e.,

$$S_5(f_m^{DC}, f_n^{DC}) = \sum_{i=1}^{X/8}\sum_{j=1}^{Y/8} |P(f_m^{DC}, I, i, j) - P(f_n^{DC}, I, i, j), \tag{21}$$

where $P(f_m^{DC}, I, i, j)$ is the DC coefficient of block (i, j). A scene change from f_m to f_n is declared if: (i) $S_5(f_m^{DC}, f_n^{DC})$ is the maximum within a symmetric sliding window and (ii) $S_5(f_m^{DC}, f_n^{DC})$ is 2–3 times the second largest maximum in the window. Although this technique is fast, cuts may be misdetected between two frames which have similar pixel values but different density functions. A metric for gradual transition has also been proposed [82] based on temporal subsampling where 1 in every 20 frames is tested rather than successive frames. This technique is sensitive to camera flashes and variations in scene that typically occur before scene changes.

4.1.2.2. Motion vectors. The apparent motion in a video sequence can be attributed to camera or object motion. Motion estimation/compensation plays an important role in video compression. The objective is to reduce the bit rate by taking advantage of the temporal redundancies between adjacent frames in a video sequence. Typically, this is accomplished by estimating the displacement (motion vectors) of uniformly sized blocks between two consecutive frames. In general, motion vectors exhibit relatively continuous changes within a single camera shot, while this continuity will be disrupted between frames across different shots.

In MPEG, B- and P-frames contain the DCT coefficients of the error signal and motion vectors. Liu and Zick [84] have presented a technique based on the error signal and the number of motion vectors. A scene cut between a P-frame f_m^P and a past reference P-frame f_n^P increases the error energy. Hence, the error energy provides a measure of similarity between f_m^P and the motion compensated frame f_n^P.

$$S_6(f_m^P, f_n^P) = \frac{\sum_{i=1}^{F_p} E_i}{F_p^2}, \tag{22}$$

where E_i is the error energy of macroblock i and F_p is the number of forward predicted macroblocks. For the detection of scene changes based on B-frames, the difference between the number of forward predicted macroblocks F_p and backward predicted B_p is used. A scene change between a B-frame and its past reference B-frame will decrease F_p and increase B_p. A scene change is declared if the difference between F_p and B_p changes from positive to negative.

Zhang *et al.* [80] have proposed a technique for cut detection using motion vectors in MPEG. This approach is based on the number of motion vectors M. In P-frames, M is the number of motion vectors. In B-frame, M is the smaller of the counts of the forward and backward nonzero motion. Then $M < T$ will be an effective indicator of a camera boundary before or after the B- and P-frame, where T is a threshold value close to zero. However, this method yields false detection when there is no motion. This is improved by applying the normalized inner product metric (Eq. (19)) to the two I-frames on the sides of the B-frame where a break has been detected.

4.1.2.3. Hybrid motion/DCT. Meng *et al.* [85] have presented a segmentation algorithm based on motion information and the DC coefficients of the luminance component. To start with, the DC coefficients in the P-frames are reconstructed. The variance of the DC coefficients $|\Delta\sigma^2|$ for the I- and P-frames is then computed. Three ratios are computed, namely,

$$R_p = \frac{\text{Number of intra compressed macroblocks}}{\text{Number of blocks with motion compensation}}$$

$$R_b = \frac{\text{Number of backward motion vectors}}{\text{Number of forward motion vectors}}$$

$$R_f = \frac{1}{R_b}.$$

A two-pass algorithm is applied. In the first pass, suspected scene change frames are marked. A P-frame and B-frame are suspected frames if R_p and R_b peak, respectively. An I-frame is a suspected frame if $|\Delta\sigma^2|$ peaks and R_f of the B-frames in front of them peaks. In the second pass, all suspected frames which fall in a dissolve region are unmarked. All the marked frames are then examined. If the difference between the current marked frame and the last scene change exceeds a threshold, then the current marked frame is a true scene change.

4.1.2.4. Segmentation using subband decomposition. Lee and Dickinson [86] have presented a scene detection algorithm where the temporal segmentation is applied on the lowest subband in subband compressed video. Four metrics have been investigated, namely:

• *Difference of histograms* measures the absolute sum of the histograms of f_m^L and f_n^L:

$$S_7(f_m, f_n) = \sum_{i=1}^{N} |H(f_m^L, i) - H(f_n^L, i). \qquad (23)$$

This metric is insensitive to object motion; however, it is sensitive to camera operations such as panning and zooming.

• *Histogram of difference frame* is the histogram of the pixel to pixel difference frame and measures the change between two frames f_m and f_n. The degree of change between f_m and f_n is large if there are more pixels distributed away from the origin. Hence

$$S_8(f_m, f_n) = \frac{\sum_{i \notin [-\alpha,\alpha]} H(f_m^L - f_n^L, i)}{\sum_{i=-N}^{N} H(f_m^L - f_n^L, i)}. \qquad (24)$$

where α is a threshold for determining the closeness to zero. The histogram of the difference frame (Eq. (24)) is more sensitive to object motion than the difference of hitograms (Eq. (23)) [86].

• *Block histogram difference* is the lowest subband is divided into R blocks and the sum of the absolute differences of the blocks defined as

$$S_9(f_m, f_n) = \sum_{i=1}^{R} \sum_{j=1}^{N} |H(f_m^L, i, j) - H(f_n^L, i, j)| \qquad (25)$$

is used as a metric for scene change detection. This metric is sensitive to local object motion.

• *Block variance difference:* instead of using the histogram, the variance of the block is used, i.e.,

$$S_{10}(f_m, f_n) = \sum_{i=1}^{R} \sum_{j=1}^{N} |\sigma^2(f_m^L, i, j) - \sigma^2(f_n^L, i, j)|. \qquad (26)$$

This metric is block based and hence it is sensitive to local object motion.

After the segmentation of a video stream, features within each shot such as content, length, and camera operations are used for indexing purposes. Two approaches for video representation are distinguished. The first approach is based on image indexing techniques, while the second is based on temporal features. We now present video indexing techniques based on spatial features. Temporal-based indexing techniques are presented in Section 3.3.

4.2. Spatial Features

A set of representative (reference) frames are selected to represent each shot to be stored in the database. Image indexing techniques (Section 3.1) are then applied on the reference frame(s). Arman *et al.* [87] have proposed a technique where each video shot is represented using the shape and color features of a reference frame. The reference frame is the 10th frame in the shot. The shape and color properties are represented using moments (the mass and the moments of inertia around the horizontal and vertical axes) and color histogram, respectively. Zhang and Smoliar [88] have presented an algorithm where the reference frame(s) is first segmented based on prominent color. In addition, the reference frame is partitioned into 9 subframes (3×3). Each frame is indexed using the size, color, shape, and location of the segmented regions and the color histograms of the frame and the 9 subframes.

The major drawback of spatial-based video indexing techniques is that video sequences are treated as still images, thus the semantics contained in a sequence are lost. This results in restricting the user queries. Temporal features allow the user to specify queries that involve the exact positions and trajectories of the objects in a shot.

4.3. Temporal Features

The apparent motion in a video sequence can be attributed to camera or object motion. In this section, we present a review of video indexing techniques using motion information and camera operations.

4.3.1. Motion

Here, image sequences are indexed based on the motion properties of objects within the sequence. The goal of the

TABLE 2
Trajectory Representation and Matching

Representation method	Matching function
Absolute frame coordinates	Exact matching
Relative frame coordinates	Exact matching
B-spline	Curve comparison
Chain code	Approximate matching
Differential chain code	Qualitative matching

system is to be able to retrieve a ranked set of sequences which have object motions similar to that specified by the query.

Ioka and Kuroka [96] have presented a method for retrieving sequences using motion information as a key. To start with, each frame is partitioned into rectangular blocks. Motion vectors are derived from the image sequences using block matching. These vectors are mapped into spatiotemporal space and the motion of each block is then represented as a single vector in the feature space. The vectors are clustered and a representative trajectory is generated for each group of vectors. A representative trajectory of a cluster is the closest to the mean vector of the cluster and has the longest lifetime. The representative trajectories are stored in the database. Queries can be specified using an interactive query specification mechanism which allows the user to enter a motion trajectory. The specified trajectory is matched with the trajectories of the sequences in the database using a distance measure and the sequences with the smallest distance are retrieved.

Although this technique does not address the problem of correspondence of trajectories, it can be incorporated as a low level tool into a complete video data management system for raw feature-based retrieval.

Dimitrova and Golshani [98] have proposed a technique based on the motion compensation component of the MPEG video encoder. The trajectory of a macroblock is computed from the forward and backward motion vectors that belong to the macroblock. The position of a macroblock in a P-frame is computed using block coordinates and forward motion vectors. The position of a macroblock in a B-frame is computed by averaging the positions obtained from (i) the next predicted block coordinates and the backward motion vector and (ii) the previous block coordinates and forward motion vector. Each trajectory can be thought of as an n-tuple of motion vectors. The macroblock trajectories are the feature vectors used for indexing. Trajectory representations and matching functions are shown in Table 2.

The trajectories of the macroblocks are used for extracting object motion. A hierarchical clustering algorithm is used. The algorithm starts with clusters that contain only one trajectory. At subsequent steps, neighboring clusters that have similar trajectories are merged. The trajectory for the extended cluster is computed by averaging the trajectories within a cluster.

Lee and Kao [97] have presented a video indexing technique based on the motion representation for the track of a moving object (using optical flow for motion extraction). Object motion is represented using a combination of the following 16 primitive motion types:

1. *Translation:* North, northeast, east, southeast, south, southwest, west, northwest.
2. *Translation in depth:* close to the camera, away from the camera.
3. *Rotation:* clockwise, counterclockwise.
4. *Rotation in depth:* rotate to left, rotate to right, rotate upward, rotate downward.

4.3.2. Camera Operations

The seven basic camera operations are fixed, panning (horizontal rotation), tracking (horizontal transverse movement), tilting (vertical rotation), booming (vertical transverse movement), zooming (varying the focusing distance), and dollying (horizontal lateral movement) as shown in Fig. 8. Camera operations include the basic operations and all the different possible combinations [91].

Akutsu *et al.* [91] have used motion vectors and their Hough transforms to identify the seven basic camera operations. The motion vectors pattern is characterized physically and spatially by (i) the magnitude of the motion vectors and (ii) the divergence/convergence point. For example, in case of a simple zoom in (Fig. 9a), pan right (Fig. 10a), and tilt up (Fig. 11a) at a constant speed, the motion vectors are shown in Figs. 9b, 10b, and 11b, respectively. The algorithm has two stages. The first stage employs block matching to determine the motion vectors between successive frames. In the second stage the motion vectors are transformed to the Hough space. The Hough transform of a line in the spatial domain is just a point in the Hough space. A group of lines in the spatial domain are represented by

$$\rho = x_0 \cos(\varphi) + y_0 \sin(\varphi) \tag{27}$$

in the Hough space, where (x_0, y_0) is the point of divergence/convergence. The least-squares method is used to fit the transformed motion vectors to the curve represented by (27). Seven categories of camera operations have been estimated: pan, zoom, tilt, pan and tilt, pan and zoom, tilt, zoom, and pan. We note this technique based on motion vectors is noise sensitive and has a high computational complexity.

An alternate approach in detecting camera operations is to examine what are known as the X-ray images [92]. An edge detection is first performed on all the frames

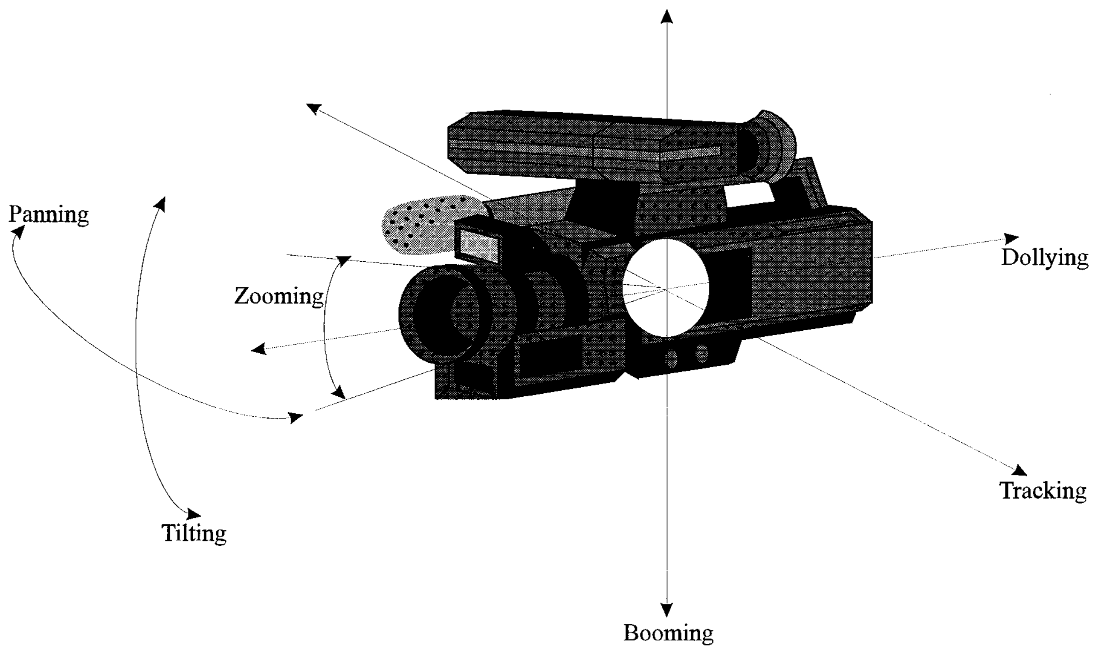


FIG. 8. Basic camera operations.

within a shot. A horizontal X-ray image is then obtained by taking a weighted integral of the edge frames in the horizontal direction. Similarly, a vertical X-ray image is obtained by taking a weighted integral of the edge frames in the vertical direction. Camera operations are obtained by approximating the spatial distribution of the edge angles of the horizontal and vertical X-ray images. We note that performing edge detection on all frames in the sequence is time consuming.

We note that in all the previous techniques, only a subset of the camera operations are extracted. In addition, it is not possible to distinguish tracking from panning, and booming from tilting. Recently, Srinivasan *et al.* [93] have proposed a technique based on optical flow in order to distinguish tracking from panning, and booming from tilting. This technique is based on the idea that if the components of the optical flow due to camera rotation and zoom are subtracted from the optical flow, the residual flow will be parallel.

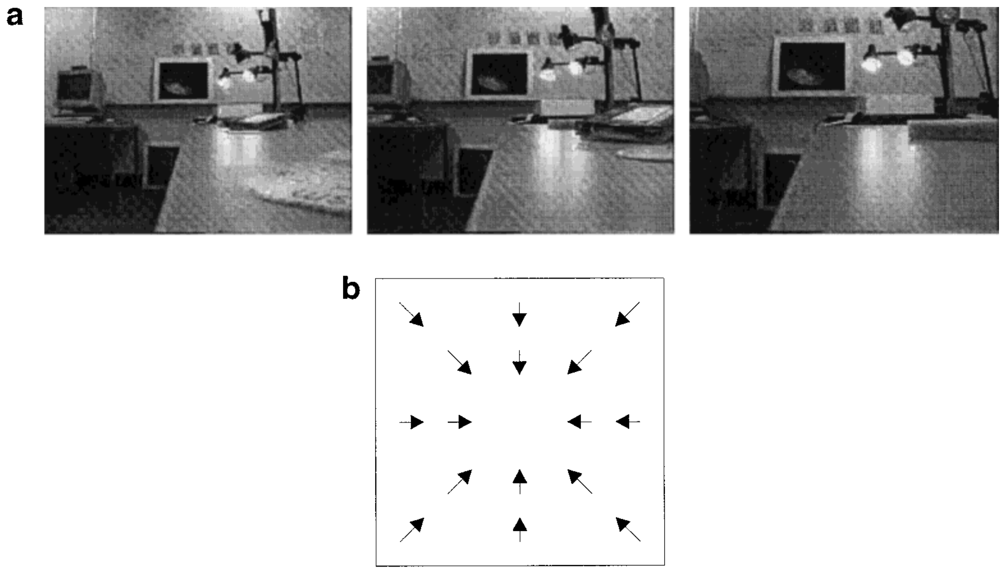
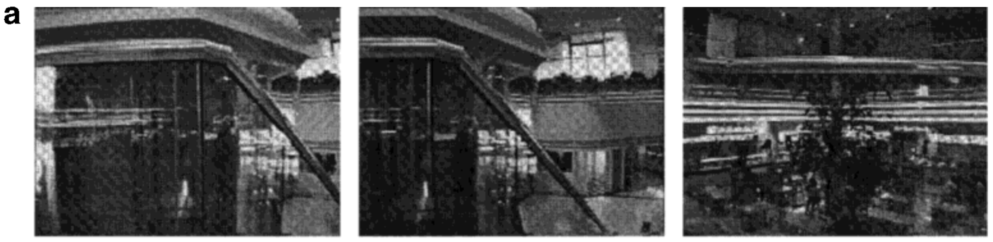


FIG. 9. (a) Frames 5, 30, and 60 of a zoom-in sequence; (b) motion vectors.



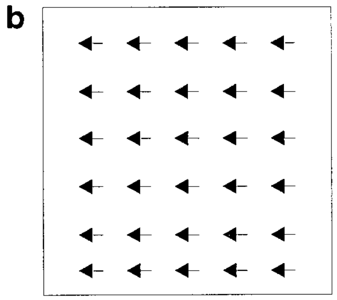


FIG. 10. (a) Frames 1, 7, and 41 of the pan sequence; (b) motion vectors.

We note that in all these techniques for the detection of camera operations, it is assumed that there is no large moving object dominating the visual field in the video sequences. In case of the presence of a large moving object dominating the visual field, false detection of a camera operation may occur. The effect of a large moving object on the detection process can be reduced by employing techniques to detect the moving objects and compensate for their effects.

5. CONCLUSIONS

With the progress of multimedia technology, large amounts of visual data will be widely accessible and thus will become one of the primary sources of information, much as text is today. Whether the application is distance learning, digital libraries, interactive television, multimedia news, or banking, large volumes of visual data will be required to be accessed precisely and efficiently [111].

Image and video indexing is crucial in many applications for efficient retrieval of visual information from multimedia databases. A straightforward extension of existing text indexing techniques for image and video indexing is inefficient and complex. Moreover, text-based approaches are not generic and hence are not useful in a wide variety of applications. As a result, there has been a new focus on developing content-based indexing techniques which are domain independent and an be automated. This raises several issues which need to be addressed:

• Different features have shown to be effective in image and video retrieval. However, there are several issues that must be resolved: What other features can be used to represent an image or a video sequence? How can the features be modified to tolerate noise and distortion and to provide invariance properties? How do we represent the feature in compact form? (such to enable hierarchical retrieval).

• The understanding of how the human abstracts certain attributes or features will help in integrating the different features.

• Exploitation of psychological findings regarding human perception of similarity in order to develop novel similarity measures.

• In a generic visual database system, it is impossible to foresee all possible queries in advance. In addition, it is impractical to have an attribute for each possible query (e.g., color, shape, etc.). The index should be generic and enable us to derive a dominant feature to perform the search operation based on the specific query.

• A natural level for representing visual content would be the object level (e.g., a horse or a racing car). An important research issue is how to develop techniques that can automatically extract such information from image and video data both in the compressed and uncompressed domains.

• Visual data in future multimedia databases is expected to be stored in the compressed form. Compressed domain image/video indexing techniques based on compression parameters such as DCT coefficients, motion vectors, etc. are being currently pursued. However, compression schemes may have some limitations imposed on the functions that can be implemented. Consequently, one important research issue involves the development of joint compression-indexing techniques that are optimized not

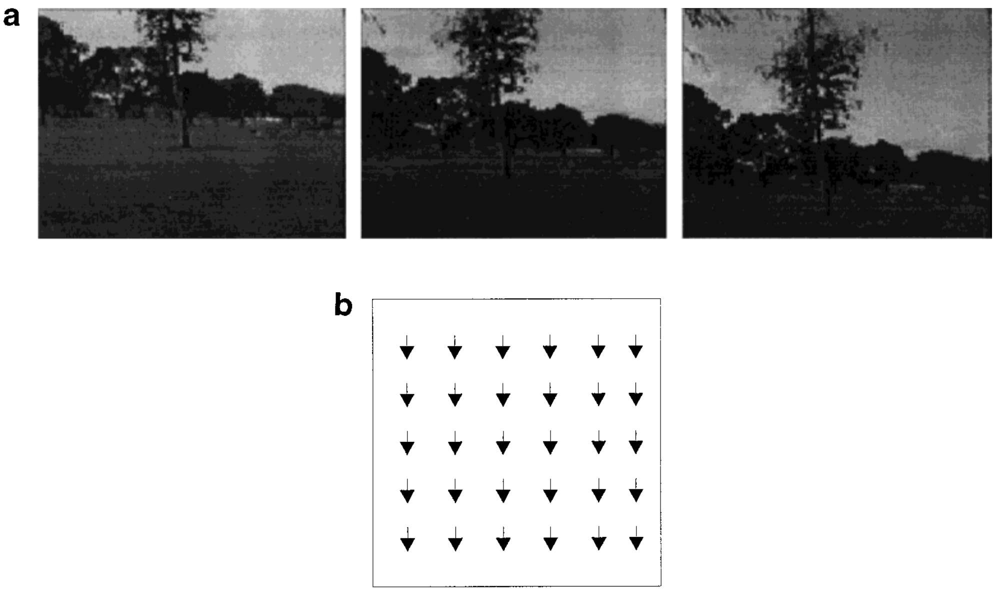


FIG. 11. (a) Frames 5, 10, and 15 of a sequence which involves tilt-up camera operations; (b) motion vectors.

only in terms of signal distortion, bit rate, and complexity but also provide the feature of indexing.

• Compared to alphanumeric data, image/video data have different characteristics. First, image/video contents cannot be precisely described. An image modeling technique must hide the underlying data types from the user and provide mechanisms to map and select the proper data type [102]. Second, objects and image/video features do not carry semantic meaning by themselves. Therefore, image/video should be modeled with spatial meanings (features) and then associate semantic meanings to image/video features for different usage [102].

• The large computational costs associated with similarity measures necessitates the organization of image/video indices such that efficient search strategies can be used. Flexible data structures should be used in order to facilitate storage/retrieval in visual information systems. Structures such as R-tree family [105], R*-tree [106], quad-tree [107], and grid file [108] are commonly used. Each structure has its advantages and disadvantages; some have limited domains and some can be used concurrently with others. In large part, the choice of data structure is determined by the characteristics of the data model.

In summary, visual data indexing is an active area of research with continuing contributions from several domains of research including image processing, computer vision, database systems, and artificial intelligence.

REFERENCES

1. G. Salton and M. J. McGill, *An Introduction to Modern Information Retrieval*, McGraw–Hill, New York, 1983.
2. J. J. Fan and K. Y. Su, An efficient algorithm for matching multiple patterns, *IEEE Trans. Knowl. Data Eng.* **5**(2), April 1993, 339–351.
3. P. Aigrain, H.-J. Zhang, and D. Petkovic, Content-based represented and retrieval of visual media: A state-of-the-art-review, *Multimedia Tools Applications* **3**(3), November 1996, 179–202.
4. H. Tamura and N. Nokoya, Image database systems: A survey, *Patt. Recognition* **17**(1), 1984, 29–43.
5. A. K. Jain, *Fundamentals of Digital Image Processing,* Prentice Hall International, Englewood Cliffs, NJ, 1989.
6. E. L. Hall, *Computer Image Processing and Recognition*, Academic Press, New York, 1979.
7. V. N. Gudivada and V. V. Raghvan, Content-based image retrieval systems, *IEEE Comput.* **28**(9), September 1995, 18–22.
8. M. J. Swain and D. H. Ballard, color indexing, *Int. J. Comput. Vision* **7**(1), 1991, 11–32.
9. M. J. Swain, Interactive indexing into image databases, *Proc. SPIE: Storage Retrieval Image Video Databases* **1908,** February 1993, 95–103.
10. Y. Gong, H. Zhang, H. Chuant, and M. Sakauuchi, An image database system with content capturing and fast image indexing abilities, in *Proceedings of the International Conference on Multimedia Computing and Systems, May 1994,* pp. 121–130.
11. W. Niblack, R. Barber, W. Equitz, M. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, The QBIC project: Querying images by content using color, texture and shape, *Storage Retrieval Image Video Databases* **1908,** February 1993, 173–187.
12. S. W. Smoliar and H.-J. Zhang, Content-based video indexing and retrieval, *IEEE Multimedia* **1**(2), Summer 1994, 62–72.
13. M. Stricker and M. Orengo, Similarity of color images, *Storage Retrieval Image Video Databases III,* **2420,** February 1995, 381–392.
14. A. Nagasaka and Y. Tanaka, Automatic video indexing and full-video search for object appearance, *IFIP: Visual Database Systems II,* 1992, 113–127.
15. T.-S. Chua, S.-K. Lim, and H.-K. Pung, Content-based retrieval of segmented images, *ACM Multimedia 94,* 1994, 211–218.

16. A. Vellaikal and C.-C. J. Kuo, Content-based image retrieval using multiresolution histogram representation, *Digital Image Storage Archiving Systems* **2602,** October 1995, 312–323.

17. M. Stricker and A. Dimai, Color indexing with weak spatial constraints, *Storage Retrieval Still Image Video Databases IV* **2670,** February 1996, 29–39.

18. A. Vellaikal and C.-C. J. Kuo, Content-based retrieval of color and multispectral images using joint spatial-spectral indexing, *Digital Image Storage Archiving Systems* **2602,** October 1995, 232–243.

19. X. Wan and C.-C. J. Kuo, Color analysis and quantization for image retrieval, *Storage Retrieval Still Image Video Databases IV* **2670,** February 1996, 8–16.

20. R. Rickman and J. Stonham, Content-based image retrieval using color tuple histograms, *Storage Retrieval Still Image Video Databases IV* **2670,** February 1996, 2–7.

21. J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, Efficient color histogram indexing for quadratic form distance function, *IEEE Trans. Patt. Anal. Mach. Intell.* **17**(7), July 1995, 729–736.

22. M. Tuceryan and A. K. Jain, Texture analysis, in *Handbook of Pattern Recognition and Computer Vision* (C. H. Chen, L. F. Pau, and P. S. P. Wang, Eds.), pp. 235–276, World Scientific, Singapore, 1993.

23. I. M. Elfadel and R. W. Picard, Gibbs random fields, cooccurences, and texture modeling, *IEEE Trans. Patt. Anal. Mach. Intell.* **16**(1), January 1994, 118–125.

24. M. Unser, Sum and difference histogram for texture classification, *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-8**(1), 1986, 118–125.

25. L. J. Guibas, B. Rogoff, and C. Tomasi, Fixed-window image descriptors for image retrieval, *Proc. SPIE: Storage Retrieval Image Video Databases III* **2420,** February 1995, 352–362.

26. R. W. Picard and T. Kabir, Finding similar patterns in large image databases, in *International Conference on Acoustics, Speech and Signal Processing, April 1993, Vol. V, pp. 161–164.*

27. R. W. Picard and F. Liu, A new wold ordering for image similarity, *International Conference on Acoustics, Speech and Signal Processing, April 1994,* Vol. V, pp. 129–132.

28. A. Petland, R. W. Picard, and S. Sclaroff, Photobook: Tools for content-based manipulation of image databases, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 34–47.

29. H. Tamura, S. Mori, and T. Yamawaki, Texture features corresponding to visual perception, *IEEE Trans. Systems Man Cybernetics* ***SMC-8***(6), June 1978, 460–473.

30. P. Brodatz, *Textures: A Photographic Album for Artists and Designers,* Dover, New York, 1966.

31. W. Y. Ma and B. S. Manjunath, Image indexing using a texture dictionary, *Digital Image Storage Archiving Systems* **2606,** October 1995, 288–298.

32. A. D. Alexandrov, W. Y. Ma, A. El Abbadi, and B. S. Manjunath, Adaptive filtering and indexing for image databases, *Storage Retrieval Image Video Databases III* **2420,** February 1995, 12–23.

33. B. S. Manjunath and W. Y. Ma, Texture features for browsing and retrieval of image data, *IEEE Trans. Patt. Anal. Mach. Intell.* **18**(8), August 1996, 837–842.

34. H. Zhang and D. Zhong, A scheme for visual feature based image indexing, *Storage Retrieval Image Video Databases III* **2420,** February 1995, 36–46.

35. J.-L. Chen and A. Kundu, Rotation and gray scale invariant texture identification using wavelet decomposition and hidden Markov model, *IEEE Trans. Patt. Anal. Mach. Intell.* **16**(2), February 1994, 208–214.

36. A. R. Rao, N. Bhushan, and G. L. Lohse, The relationship between texture terms and texture images: A study in human texture perception, *Storage Retrieval for Still Image Video Databases IV* **2670,** February 1996, 206–214.

37. G. Taubin and D. B. Coper, Recognition and positioning of rigid objects using algebraic moment invariants, *Proc. SPIE: Geometric Methods Comput. Vision* **1570,** 1991, 175–186.

38. E. Persoon and K. S. Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. Systems Man Cybernetics* **SMC-8,** 1977, 170–179.

39. S. R. Dubois and F. H. Glanz, An autoregressive model approach to dimensional shape classification, *IEEE Trans. Patt. Anal. Mach. Intell.* **8,** 1986, 55–66.

40. H. H. Chen and J. S. Su, ‘‘A syntactic approach to shape recognition, *Proc. Int. Comput. Symp.* 1986, 103–122.

41. W. I. Grosky and Z. Jiang, A hierarchical approach to feature indexing, *Proc. SPIE: Image Storage Retrieval Systems* **1662,** February 1992, 9–20.

42. B. Scassellati, S. Alexopoulos, and M. Flickner, Retrieving images by 2D shape: A comparison of computation methods with human perceptual judgments, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 2–14.

43. J. E. Gary and R. Mehrotra, Shape retrieval in image database systems, *Proc. SPIE: Image Storage Retrieval Systems* **1661,** February 1992, 2–8.

44. R. Mehrotra and J. E. Gary, Similar shape retrieval in shape data management, *IEEE Comput.* **28**(9), September 1995, 57–62.

45. D. Tegolo, Shape analysis for image retrieval, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 59–69.

46. J. P. Eakins, K. Shields, and J. Boardman, ARTISAN—A shape retrieval system based on boundary family indexing, *Proc. SPIE: Storage Retrieval Image Video Databases IV* **2670,** February 1996, 17–28.

47. Y. H. Ang, Z. Li, and S. H. Ong, Image retrieval based on multidimensional feature properties, *Proc. SPIE: Storage Retrieval Image Video Databases III* **2420,** February 1995, 47–57.

48. T. Kato, T. Kurita, N. Otsu, and K. Hirata, A sketch retrieval method for full color image database, in *International Conference on Pattern Recognition, September 1992, pp. 530–533.*

49. S.-K. Chang, Q.-Y. Shi, and C.-W. Yan, Iconic indexing by 2-D strings, *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-9**(3), May 1987, 413–428.

50. E. Jungert, Extended symbolic projection as a knowledge structure for image database systems, *Fourth BPRA Conference on Pattern Recognition, March 1988,* pp. 343–351, Springer-Verlag, Berlin/New York.

51. S. Chang, E. Jungert, and Y. Li, Representation and retrieval of symbolic pictures using generalized 2-D strings, *Proc. SPIE: Visual Commun. Image Process. IV* **1199,** 1989, 1360–1372.

52. S. K. Chang, C. M. Lee, and C. R. Dow, A 2-D string matching algorithm for conceptual pictorial queries, *Proc. SPIE: Image Storage Retrieval Systems* **1662,** February 1992, 47–58.

53. S.-Y. Lee and F.-J. Hsu, Spatial reasoning and similarity retrieval of images using 2-D C-string knowledge representation, *Patt. Recognition* **25**(3), 1992, 305–318.

54. S.-Y. Lee, M.-C. Yang, and J.-W. Chen, Signature files as a spatial filter for iconic image database, *J. Visual Lang. Comput.* **3**(4), December 1992, 373–397.

55. T.-Y. Hou, A. Hsu, and M.-Y. Chiu, A content based indexing technique using relative geometry features, *Image Storage Retrieval Systems* **1662,** February 1992, 59–68.

56. G. K. Wallace, The JPEG still picture compression standard, *Commun. ACM* **34**(4), April 1991, 31–45.

57. D. L. Gall, MPEG: A video compression standard for multimedia applications, *Commun. ACM* **34**(4), April 1991, 59–63.

58. M. Liou, Overview of the p × 64 kbits/s video coding standard, *Commun. ACM* **34**(4), April 1991, 46–58.

59. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer, Boston, 1991.

60. S.-F. Chang, Compressed domain techniques for image/video indexing and manipulation, in *IEEE International Conference on Image Processing (1), October 1995*, 314–317.

61. M. Shneier and M. Abdel-Mottaleb, Exploiting the JPEG compression scheme for image retrieval, *IEEE Trans. Patt. Anal. Mach. Intell.* **18**(8), August 1996, 849–853.

62. F. Idris and S. Panchanathan, Image indexing using vector quantization, *Proc. SPIE: Storage Retrieval Image Video Databases III* **2420,** February 1995, 373–380.

63. F. Idris and S. Panchanathan, Storage and retrieval of compressed images, *IEEE Trans. Consumer Electronics* **41,** August 1995, 937–941.

64. F. Idris and S. Panchanathan, Image indexing using wavelet vector quantization, *Proc. SPIE: Digital Image Storage Archiving Systems* **2606,** October 1995, 269–275.

65. H. Sakamoto, H. Suzuki, and A. Uemori, Flexible montage retrieval for image data, *Storage Retrieval Image Video Databases II* **2185,** February 1994, 25–33.

66. R. C. Jain, S. N. J. Murthy, and P. L.-J. Chen, Similarity measures for image databases, *Proc. SPIE: Storage Retrieval Image Video Databases III* **2420,** February 1995, 58–65.

67. L. A. Rowe, J. S. Boreczky and C. A. Eads, Indexes for user access to large video databases, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 150–161.

68. B. Holt and L. Hartwick, Visual image retrieval for applications in art and history, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 70–81.

69. R. Kasturi and R. Jain, Dynamic vision, in *Computer Vision: Principles,* pp. 469–480, IEEE Comput. Soc., Los Alamitos, CA, 1990.

70. H. J. Zhang, A. Kankanhalli, S. W. Smoliar, and S. Y. Tan, Automatic partitioning of full motion video, *ACM Multimedia Systems* **1**(1), 1993, 10–28.

71. H. J. Zhang, Y. Gong, S. W. Smoliar, and S. Y. Tan, Automatic parsing of news video, in *Proceedings of the International Conference on Multimedia Computing and Systems, May 1994,* pp. 45–54.

72. Y. Tonomura, Video handling based on structured information for hypermedia systems, in *ACM Proceedings: International Conference on Multimedia Information Systems '91, 1991,* pp. 333–344.

73. K. Otsuji, Y. Tonomura, and Y. Ohba, Video browsing using brightness data, *Visual Commun. Image Process. '91* **1606,** November 1991, 980–989.

74. K. Otsuji and Y. Tonomura, Projection detecting filter for video cut detection, *ACM Multimedia 93,* 1993, 251–257.

75. U. Gargi, S. Oswald, D. Kosiba, S. Devadiga, and R. Kasturi, Evaluation of video sequence indexing and hierarchical video indexing, *Proc. SPIE: Storage Retrieval Image Video Databases III* February 1995, 144–151.

76. D. Swanberg, C.-F. Shu, and R. Jain, Knowledge guided parsing in video databases, *Proc. SPIE* **1908,** 1993, pp. 13–24.

77. S. Shahraray, Scene change detection and content-based sampling of video sequences, *Digital Video Compression: Algorithms Tech.* **2419,** February 1995, 2–13.

78. F. Arman, A. Hsu, and M.-Y. Chiu, Image processing on compressed data for large video databases, *Proc. SPIE: Storage Retrieval Image Video Databases* 1993, pp. 267–272.

79. F. Arman, A. Hsu, and M.-Y. Chiu, Feature management for large video databases, *ACM Multimedia 93* **1908,** February 1993, 2–12.

80. H. J. Zhang, C. Y. Low, Y. Gong, S. W. Smoliar, and S. Y. Tan, Video parsing compressed data, *Proc. SPIE: Image Video Processing II* **2182,** 1994, 142–149.

81. H. J. Zhang, C. Y. Low, S. W. Smoliar, and S. Y. Tan, Video parsing and browsing using compressed data, *Multimedia Tools Applications* **1,** 1995, 89–111.

82. B.-L. Yeo and B. Liu, A unified approach to temporal segmentation of motion JPEG and MPEG compressed videos, in *Proceedings of the International Conference on Multimedia Computing and Systems, May 1995,* pp. 81–88.

83. S.-F. Chang and D. G. Messerschmitt, Manipulation and composition of MC-DCT compressed video, *IEEE J. Selected Areas Commun.* **13**(1), January 1995, 1–11.

84. H.-C. H. Liu and G. L. Zick, Scene decomposition of MPEG compressed video, *Digital Video Compression: Algorithms Tech.* **2419,** February 1995, 26–37.

85. J. Meng, Y. Juan, and S.-F. Chang, Scene change detection in a MPEG compressed video sequence, *Digital Video Compression: Algorithms Tech.* **2419,** February 1995, 14–25.

86. J. Lee and B. W. Dickinson, Multiresolution video indexing for subband coded video databases, *Proc. SPIE: Image Video Process. II* **2185,** 1994, 162–173.

87. F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu, Content-based browsing of video sequences, *ACM Multimedia 94,* 1994, 97–103.

88. H. J. Zhang and S. W. Smoliar, Developing power tools for video and retrieval, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 140–149.

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

90. A. Hampapur, R. Jain, and T. E. Weymouth, Production model based digital video segmentation, *Multimedia Tools Applications* **1,** 1995, 9–46.

91. A. Akutsu, Y. Tonomura, H. Hashimoto, and Y. Ohba, Video indexing using motion vectors, *Proc. SPIE: Visual Commun. Image Process. '92* **1818,** 1992, 1522–1530.

92. A. Akutsu and Y. Tonomura, Video tomography: An efficient method for camerawork extraction and motion analysis, *ACM Multimedia 94,* 1994, 349–356.

93. M. V. Srinivasan, S. Venkatesh, and R. Hosie, "*Qualitative Estimation of Camera Motion Parameters from Video Sequences*, submitted to publication.

94. Y. Wu and D. Suter, A comparison of methods for scene change detection in noisy image sequence, in *Proceedings of Visual '96: The First International Conference on Visual Information Systems, February 1996,* pp. 459–468.

95. A. Dailianas, R. B. Allen, and P. England, Comparison of automatic video segmentation algorithms, *Proc. SPIE: Integration Issues Large Commercial Media Delivery Systems* **2615,** 1995, 2–16.

96. M. Ioka and M. Kuroka, A method for retrieving sequences of images on the basis of motion analysis, *Proc. SPIE: Storage Retrieval Systems* **1662,** February 1992, 35–46.

97. S. Y. Lee and H. M. Kao, Video indexing—An approach based on moving object and track, *Proc. SPIE: Storage Retrieval Image Video Databases,* February 1993, 25–36.

98. N. Dimitrova and F. Golshani, $\mathscr{R}_\chi$ for semantic video database retrieval, *ACM Multimedia 94,* 1994, 219–226.

99. H. S. Sawhney, Motion video analysis using planar parallax, *Proc. SPIE: Storage Retrieval Image Video Databases II* **2185,** February 1994, 231–242.

100. G. K. Wallace, JPEG still picture compression standard, *Commun. ACM* **34**(4), April 1991, 31–45.

101. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems,* Benjamin–Cummings, Redwood City, CA, 1989.

102. S.-K. Chang and A. Hsu, Image information systems: Where do we go from here? *IEEE Trans. Knowl. Data Eng.* **4**(5), October 1992, 431–441.

103. A. Gupta, T. Weymouth, and R. Jain, Semantic queries with pictures: The VIMSYS model, *Proc. VLDB'91,* 1991, 69–79.

104. G. Davenport, T. A. Smith, and N. Pincever, Cinematic primitives for multimedia, *IEEE Comput. Graphics Applications,* July 1991, 67–74.

105. T. Sellis, N. Roussopoulos, and C. Faloutsos, The R^+ tree: A dynamic index for multidimensional objects, in *Proceedings of the 13th International Conference on Very Large Databases, 1987,* pp. 507–518.

106. 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 ACM SIGMOD the International Conference on the Management of Data, May 1990,* pp. 322–331.

107. I. Gargantini, An effective way to represent quadtrees, *Commun. ACM* **25**(12), 1982, 905–910.

108. J. Nievergelt, H. Hinterberger, and K. C. Sevcik, The grid file: An adaptable symmetric multikey file structure, *ACM Trans. Database Systems* **9**(1), March 1984, 38–71.

109. Y. Niu, M. T. Ozsu, and X. Li, *A Study of Image Indexing Techniques for Multimedia Database Systems,* Department of Computing Science, University of Alberta, Technical Report TR 95-19, July 1995.

110. G. Ahanger and T. D. C. Little, A survey of technologies for parsing and indexing digital video, *J. Visual Commun. Image Represent.* **7**(1), March 1996, 28–43.

111. *IEEE Spectrum,* Special Issue on Interactive Multimedia, March 1993.

FAYEZ M. IDRIS is a Ph.D. candidate in the Visual Computing and Communications Laboratory in the Department of Electrical and Computer Engineering at the University of Ottawa, Canada. He received his M.A.Sc. in electrical engineering from the University of Ottawa, Canada, in 1993 and the B.Sc. in electrical engineering from Yarmouk University, Jordan, in 1985. From 1986 to 1990, he was a research and development engineer at the International Systems and Electronics Development Company in Jordan (SEDCO). He is a member of SPIE. His research interests are in the areas of image and video indexing, image and video databases, image and video compression, VLSI architectures for image and video processing, parallel processing, and computer architectures.

SETHURAMAN PANCHANATHAN received his B.Sc. degree in physics from the University of Madras, India, in 1981; B.E. degree in electronics and communication engineering from the Indian Institute of Science, India, in 1984; M. Tech degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1986; and Ph.D. degree in electrical engineering from the University of Ottawa in 1989. Dr. Panchanathan is presently an associate professor in the Department of Electrical and Computer Engineering at the University of Ottawa, Canada. He is the director of the Visual Computing and Communications Laboratory at the University of Ottawa and leads a team of post-doctoral fellows, research engineers, and graduate students working in the areas of compression, indexing, storage, retrieval, and browsing of images and video; VLSI architectures for video processing; multimedia hardware architectures; parallel processing; and multimedia communications. He is a principal investigator of projects funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Provincial Centre of Excellence on Telecommunications Research (TRIO), and industry. He is also a co-investigator of projects funded by the Federal Centers of Excellence on Telecommunications Research (CITR) and Microelectronics Network (MICRONET). He has published over 100 papers in refereed journals and conferences. He was the co-chair of the IS&T/SPIE Digital Video Compression-Algorithms and Technologies '96 and Multimedia Hardware Architectures '97 Conferences held in San Jose. He is also the tutorials chair of the IEEE International Conference on Multimedia Systems '97. He is a senior member of the IEEE and a member of the Professional Engineers of Ontario, SPIE, and EURASIP. He is and associate editor of the *IEEE Transactions on Circuits and Systems for Video Technology* and an area editor of the *Journal of Visual Communications and Image Representation.* He is also a guest editor of special issues on "Indexing, Storage, Browsing and Retrieval of Images and Video" in the *Journal of Visual Communication and Image Representation* and on "Visual Computing and Communications" in the *Canadian Journal of Electrical and Computer Engineering.* His laboratory's home page can be found at http://vccl.uottawa.ca.