Computer Vision and Image Understanding
Vol. 75, Nos. 1/2, July/August, pp. 3–24, 1999
Article ID cviu.1999.0761, available online at http://www.idealibrary.com on IDEAL®

A Stochastic Framework for Optimal Key Frame Extraction from MPEG Video Databases

Yannis S. Avrithis,* Anastasios D. Doulamis, Nikolaos D. Doulamis, and Stefanos D. Kollias

Electrical and Computer Engineering Department, National Technical University of Athens, Zografou 15773, Athens, Greece

A video content representation framework is proposed in this paper for extracting limited, but meaningful, information of video data, directly from the MPEG compressed domain. A hierarchical color and motion segmentation scheme is applied to each video shot, transforming the frame-based representation to a feature-based one. The scheme is based on a multiresolution implementation of the recursive shortest spanning tree (RSST) algorithm. Then, all segment features are gathered together using a fuzzy multidimensional histogram to reduce the possibility of classifying similar segments to different classes. Extraction of several key frames is performed for each shot in a content-based rate-sampling framework. Two approaches are examined for key frame extraction. The first is based on examination of the temporal variation of the feature vector trajectory; the second is based on minimization of a cross-correlation criterion of the video frames. For efficient implementation of the latter approach, a logarithmic search (along with a stochastic version) and a genetic algorithm are proposed. Experimental results are presented which illustrate the performance of the proposed techniques, using synthetic and real life MPEG video sequences.

1. INTRODUCTION

Efficient access to video data located on distributed platforms is a very hard task, mainly due to large bandwidth requirements imposed by the large amount of video information. Traditionally, video is represented by numerous consecutive frames, each of which corresponds to a constant time interval. However, such a representation is not adequate for new emerging multimedia applications, such as content-based indexing, retrieval, and video browsing. Moreover, tools and algorithms for effective organization and management of video archives are still limited. For this reason, the MPEG-4 standardization phase aims at effective audiovisual coding, giving a new dimension to creating, accessing and manipulating video content [25, 34]. Furthermore, the MPEG-7 standard aims at developing an integrated framework for a multimedia content description interface [26], with the objective to specify a set of descriptors that can be used to represent various types of multimedia information [20, 31].

* Corresponding author. E-mail: iavr@image.ntua.gr.

The active research effort in this area has been reflected in many conferences and special issues of leading journals dedicated to this topic [16, 37, 38]. In addition, several prototype systems, such as QBIC [12], Virage [14], VisualSEEK [35], Photobook [30], MARS [32], Netra [21], and VideoQ [4] have already been developed and are now in the first stage of commercial exploitation for content-based image manipulation. However, these systems cannot be easily extended to handle video databases, since it is very inefficient and time consuming to perform queries on every video frame. Storage requirements of digitized video information, even in compressed domain, are very large and present challenges to most multimedia servers. To make thinks worse, most video databases are often located on distributed platforms, imposing great transmission bandwidth requirements. For this reason, apart from developing appropriate congestion schemes or proposing algorithms for effective network design through modeling of video sources [10], new methods for efficient video content representation should also be implemented.

In particular, a "preindexing" stage should be introduced, extracting limited and meaningful information of the video content. The objective is to divide a video sequence into separate representative shots and then to extract the most characteristic frames (*key frames*) within the selected shots by means of a content-based sampling algorithm [33]. In this framework, instead of performing a query on all available video frames, one can only consider the selected ones. Video queries can thus be performed faster and more efficiently, since the redundant information is rejected. Furthermore, key frame extraction also permits fast video browsing and provides a powerful tool for video content summarization and visualization. For example, it has been observed in [42] that a 30-min video stream consists of approximately 200 shots. Consequently, using five key frames per shot, only 1000 out of 45,000 frames are required to represent the video content.

Some approaches [29, 41] are oriented to detecting shot changes; they can, therefore, be used as the first stage of video visualization algorithms. Video representation based on the extraction of frames at regular time instances has been proposed in [23]. This algorithm exploits neither shot information nor frame similarity. Consequently important shots of small duration may have no representatives while shots of longer duration may be represented by multiple frames with similar content. Exploiting

1077-3142/99 $30.00

shot information and selecting one key frame for each shot has been presented in [1, 36]. However, a single key frame cannot provide sufficient information about the video content of a given shot, especially for shots of long duration. Recently some other approaches dealing with construction of a compact image map or image mosaics have been described in [17, 40]. In [17] all frames of a shot are aligned with respect to the dominant object, while in [40], a panoramic view of the shot frames is displayed. Although such a representation can be very good for specific applications, it cannot be effectively implemented in real world complex shots, where background/foreground changes or complicated camera effects may appear. A method for analyzing video and building a pictorial summary for visual representation has been proposed in [42]. This work is concentrated on dividing a video sequence into consecutive meaningful segments (story units) and then constructing a video poster for each story unit based on shot dominance, instead of extracting key frames. Other approaches for content-based video indexing include the works reported in [5, 15, 28].

In this paper, extraction of several key frames within a shot is proposed for efficiently describing the shot content. First, video frame-based representation is transformed into a feature-based one, which is closer to the semantic characterization of the shot. This is accomplished by applying several image processing and analysis techniques, exploiting color and motion information, to each video frame. To reduce the required computations and simultaneously exploit information existing in MPEG video databases, such as block color average and motion vectors, our analysis is performed directly on the MPEG compressed domain. In this way, decoding can be omitted or performed with minimal effort. Then, all the aforementioned features are gathered together, using a fuzzy feature vector formulation. Two approaches for key frame extraction are proposed. The first one is based on temporal variation of feature vectors, while the second relies on minimization of a cross-correlation criterion.

In order to present the two key frame extraction approaches which constitute the main originality of the paper, we first introduce, in Section 2, the feature-based video representation incorporated in the proposed framework. The description of this representation includes shot detection, video sequence analysis (color and motion segmentation), and fuzzy feature vector formulation. Then, in Section 3, the temporal variation approach for key frame extraction is presented, along with an example of a synthetic video sequence that also demonstrates the properties of the proposed feature vector representation. The theoretical analysis of the cross-correlation approach is included next in Section 3, while its actual implementation is discussed in Section 4. In particular, a deterministic and a stochastic version of a logarithmic search, as well as a genetic algorithm are proposed for efficient implementation of this approach. Finally, experimental results on video sequences are presented in Section 5, demonstrating the performance of the proposed techniques, and Section 6 concludes the paper.

2. FEATURE-BASED VIDEO REPRESENTATION

A block diagram of the proposed architecture is illustrated in Fig. 1, consisting of four modules: shot cut detection, video sequence analysis, fuzzy classification, and key frame extraction. The first three modules which are described in this section produce a feature vector representation of the video sequence by first segmenting it into distinct video shots and then applying image analysis techniques to the frames of each shot. Such a representation provides a more meaningful description of the video content and, therefore, key frame extraction can be implemented more efficiently.

2.1. Shot Detection

Since a video sequence is a collection of different shots, each of which corresponds to a continuous action of a single camera operation [41], a shot detection algorithm is first applied in order to temporally segment the sequence into shots. Several algorithms have been reported in the literature for shot change detection which deal with the detection of cut, fading, or dissolve changes, either in the compressed or the uncompressed domain [29, 41]. In our approach the algorithm proposed in [41] has been adopted for shot detection since it is based on the dc coefficients of the DCT transform of each frame. These coefficients are directly available in the case of intracoded frames (I frames) of MPEG compressed video sequences, while for the intercoded ones (P and B frames), they can be estimated by the motion-compensated error with minimal decoding effort. Adoption of the above-described algorithm results in significant reduction of the required computations, compared to other algorithms which require a full resolution search of the video data.

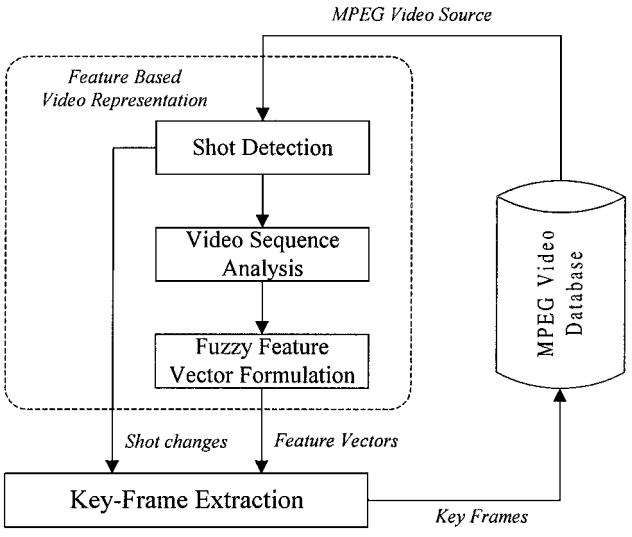

FIG. 1. Block diagram of the proposed architecture.

2.2. *Video Sequence Analysis*

Once a video sequence is temporally partitioned into video shots, the next step of the analysis is segmentation of each shot into semantically meaningful objects and extraction of essential information describing those objects. The goal of semantic segmentation is to determine the presence of a set of regions representing known objects with semantic meaning. This, however, is a difficult problem, since it involves a priori knowledge about the objects to be detected (e.g., detection of human faces) and thus can only be solved for a limited range of application contexts (e.g., videophone systems, news bulletins, etc.) [9, 18]. In this paper, color and motion segmentation is applied to the image sequence representing each video shot. Although semantic segmentation would be essential in a content-based retrieval environment, color and motion segmentation provide a powerful representation of video shots for the problem of key frame extraction. In the following, color and motion information is kept distinct in order to provide a flexible video content representation, where each piece of information can be handled separately. In particular, the number, size, location, and average color components of all color segments are used for the construction of a color feature vector. In a similar way, the number, size, location, and average motion vectors of all motion segments are used for the construction of a motion feature vector. These two vectors are then combined as explained in the next section.

A. Color segmentation. The recursive shortest spanning tree (RSST) [24] algorithm is our basis for color segmentation of each frame in a given video shot. Despite its relative computational complexity, it is considered as one of the most powerful tools for image segmentation, compared to other techniques (including color clustering, pyramidal region growing, and morphological watershed) [27].

The flowchart of the algorithm is depicted in Fig. 2a. Initially an image I of size $M_0 \times N_0$ pixels, is partitioned into $M_0 \times N_0$ regions (segments) of size 1 pixel and links are generated for all 4-connected region pairs. Each link is assigned a weight equal to the distance between the two respective regions, which is in general defined as the Euclidean distance between the average color components of the two regions, using a bias for merging small regions. Using, for example, the RGB color space, a distance

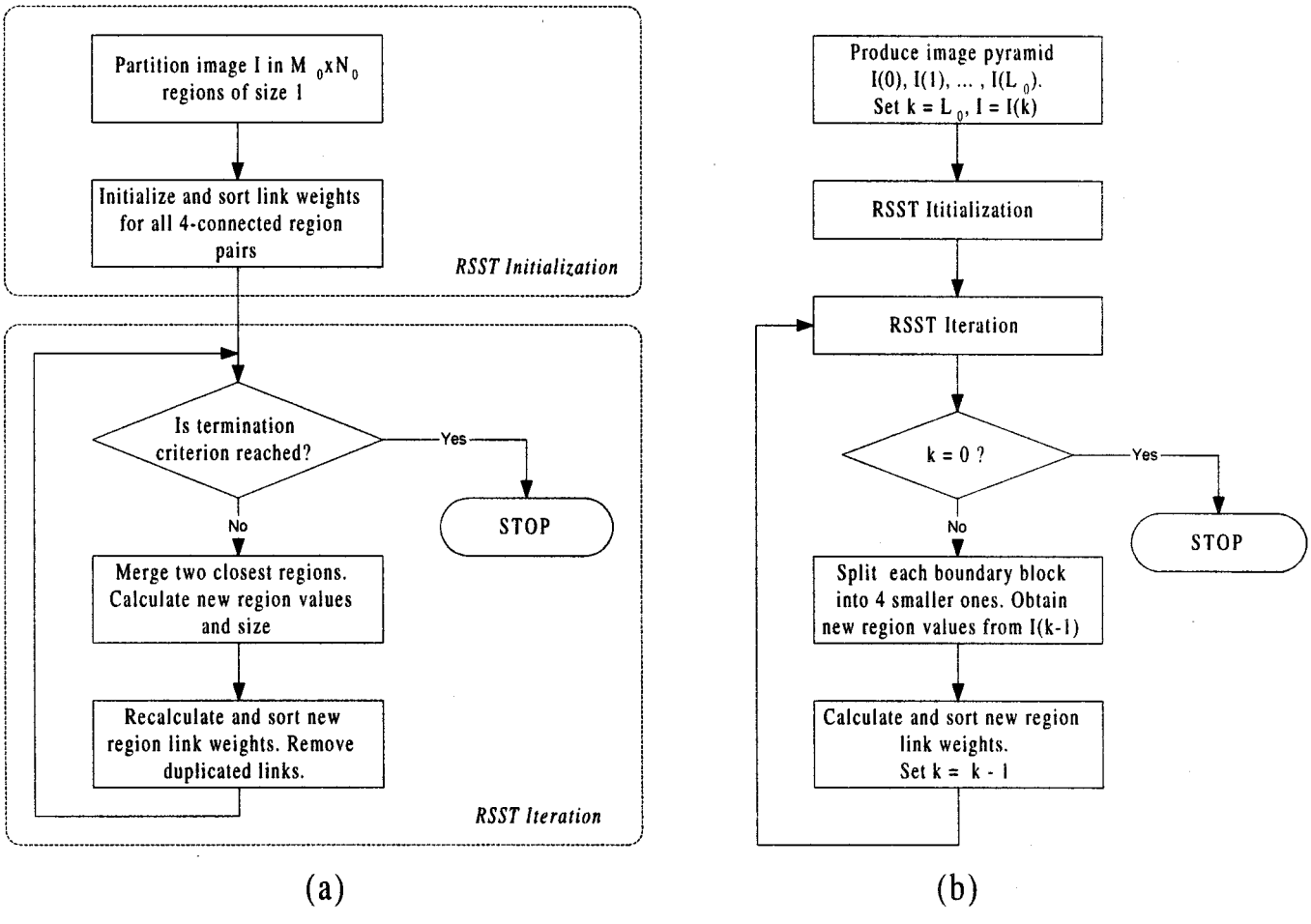

FIG. 2. (a) Flowchart of the RSST; (b) flowchart of the proposed segmentation algorithm.

measure between two adjacent regions X and Y is defined as

$$d(X,Y) = [(R_X - R_Y)^2 + (G_X - G_Y)^2 + (B_X - B_Y)^2]^{1/2} \frac{A_X A_Y}{A_X + A_Y}, \quad (1)$$

where R_X, G_X, and B_X respectively represent the average R, G, and B values of all pixels inside region X and A_X is the number of pixels within the region. All link weights are then sorted in ascending order, so that the least weighed link corresponds to the two closest regions. The iteration phase of the RSST is then initiated, where neighboring regions are recursively merged by applying the following actions in each iteration: (i) the two closest regions are merged and the new region color components and size are calculated; (ii) the new region link weights from all neighboring regions are recalculated and sorted; and (iii) any duplicated links are removed. The iteration terminates when either the total number of regions or the minimum link weight (distance) reaches a target value (threshold). A distance threshold is in general preferable since it provides a result that is independent of the image content.

The execution time of the RSST is heavily dependent upon the choice of the sorting algorithm, which is certainly a bottleneck of the algorithm. For this reason, a new multiresolution RSST (M-RSST) approach is proposed, which recursively applies the RSST algorithm on images of increasing resolution, as depicted in the flowchart of Fig. 2b. Initially a multiresolution decomposition of image I is performed with a lowest resolution level of L_0 so that a hierarchy of frames $I(0) = I, I(1), \ldots, I(L_0)$ is constructed, forming a truncated image pyramid, with each layer having a quarter of the pixels of the layer below. The RSST initialization takes place for the lowest resolution image $I(L_0)$ and then an iteration begins, involving the steps: (i) regions are recursively merged using the RSST iteration phase; (ii) each boundary pixel of all resulting regions is split into four new regions, whose color components are obtained from the image of the next higher resolution level; (iii) the new link weights are calculated and sorted. This "split-merge" procedure is repeated until the highest resolution image $I(0)$ is reached.

The results of the proposed color segmentation algorithm are depicted in Fig. 3 for a target number of segments equal to 5 and for an initial resolution level $L_0 = 3$ (equivalent to 8×8 blocks). After application of the RSST iteration on $I(3)$ (Fig. 3c), the boundary pixels are split into four new segments each. As shown in Fig. 3d, the total number of segments for the next RSST iteration at resolution level 2 is considerably reduced, compared to the initial number of segments of the conventional RSST algorithm at the same resolution level. The same applies for the next level (Figs. 3e and f). Since the speed of the RSST depends heavily on the initial number of segments, it is clear that the proposed M-RSST approach yields much faster execution, compared to RSST. The computational complexity of the M-RSST, however, is not straightforward to calculate, since it depends on the number, shape, and size of segments. For this reason, the execution times of both algorithms for the image of Fig. 3 at different sizes are compared in Table 1. The execution times have been obtained using a C implementation on a Sun SparcStation-20 system. It can be seen that the improvement ratio is heavily affected by the adopted image size and that the M-RSST is approximately 400 times faster than the RSST for a typical image size of 720×576 pixels.

TABLE 1
Execution Times of the RSST and the Proposed Multiresolution RSST (M-RSST) for Various Image Sizes

Image size	Execution time (s)		Improvement ratio
	RSST	M-RSST	
176×144 (QCIF)	5.65	0.13	43.46
352×288 (CIF)	44.21	0.38	116.35
720×576 (PAL)	534.22	1.36	392.81

Note. The initial resolution level for the M-RSST is $L_0 = 3$ (equivalent to 8×8 blocks) in all cases.

Also, it is observed from Fig. 3c that very small segments cannot be found by the algorithm at the initial (lowest) resolution, and, since no segments are created or destroyed at each iteration, these segments are also eliminated from all resolution levels (Figs. 3b, e). For example, even if the target number of segments was higher than 5, some facial details would not produce separate segments. The final segmentation result of the M-RSST is thus different from that of the conventional RSST, as far as small segments are concerned. However, such filtering according to object size is desirable in the context of key frame selection, since it achieves a high level of video content representation. This is illustrated in Fig. 4, where a landscape image is correctly segmented into building, sea, or forest areas. Oversegmentation is avoided and, thus, texture images can be handled.

Finally, it can be seen in Figs. 3c, e that effectively only the segment contour shapes are affected at each iteration, since no segments are created or destroyed. It is therefore possible to acquire the exact contour shapes of the segments that are retained at the highest resolution level (as in Fig. 3b). Moreover, this is achieved without using the entire images of the truncated image pyramid, but only parts of them at the object boundaries. This can be exploited for segmenting frames in MPEG video streams by adopting block resolution for initialization of the algorithm. In this case, the truncated image pyramid is actually not constructed entirely. Instead, decoding of a very small percentage of blocks near object boundaries is performed in order to obtain the required parts of the higher resolution images, resulting in a very fast implementation.

B. Motion segmentation. In order to solve the motion segmentation problem, numerous different techniques can be employed, such as direct intensity-based methods, optical flow-based methods, or simultaneous motion estimation and segmentation methods [39]. Each method has its own advantages and disadvantages, restricting its use to specific applications. For

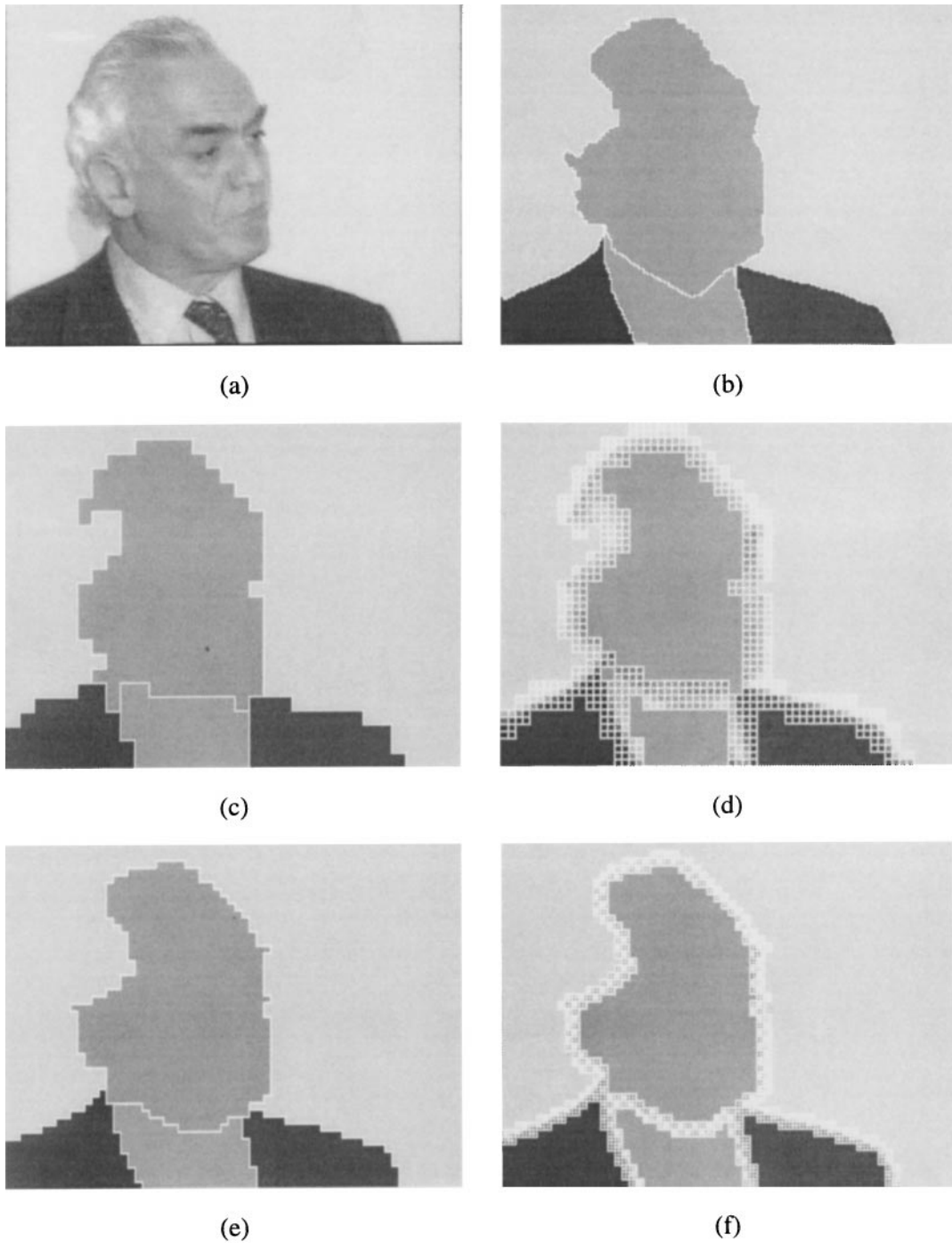

FIG. 3. Color segmentation: (a) initial image; (b) final segmentation; (c) segmentation at resolution level 3; (d) boundary pixels split at level 3; (e) segmentation at level 2; and (f) boundary pixels split at level 2.

example, simultaneous motion estimation and segmentation methods are unattractive due to their high computational complexity, while direct intensity-based methods cannot handle camera noise and illumination changes. Optical flow methods are quite popular and widely used both for video coding and image analysis and understanding.

Having in mind the huge amount of computations involved in the analysis of large video databases on the one hand, and the limited requirements for accuracy in the problem of key frame extraction on the other hand, we have chosen to exploit the motion vector information that is directly available in MPEG streams, thus eliminating the need for motion analysis

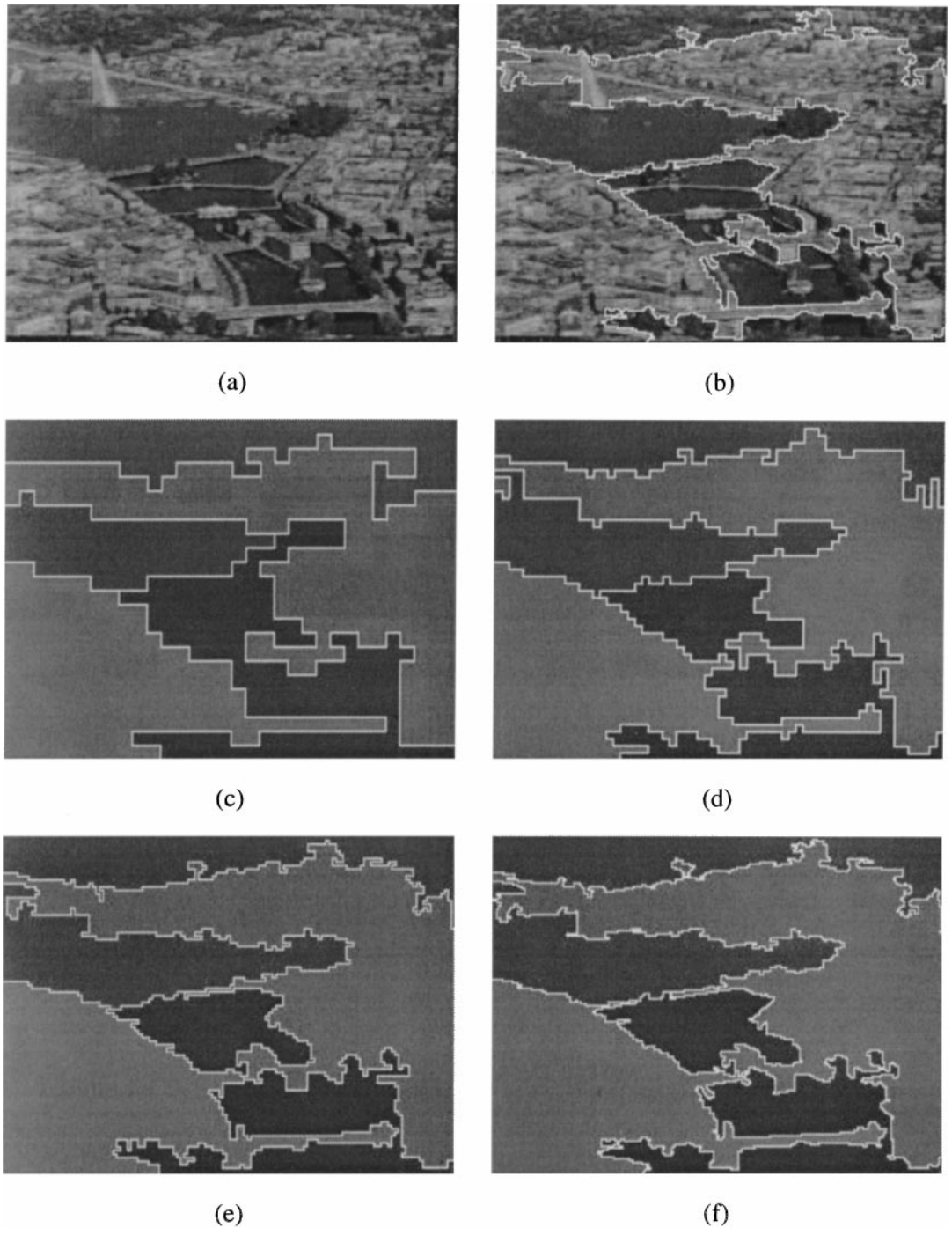

FIG. 4. Color segmentation: (a) initial image; (b) final segmentation (initial image shown inside segments); (c) segmentation at resolution level 3; (d) segmentation at level 2; (e) segmentation at level 1; and (f) segmentation at level 0 (final).

altogether. Since no decoding is necessary, an extremely fast implementation is achieved. However, poor motion estimates are obtained, since motion vectors of MPEG streams are usually very noisy. For this reason, a postprocessing step for spatial filtering (smoothing) of motion vectors is necessary. A median filter is selected for this purpose due to its speed and its ability to preserve object contours. Motion segmentation is performed by dividing each frame into regions of homogeneous motion. The proposed M-RSST algorithm described above is applied at the MPEG block resolution, and motion vector differences are used instead of color differences for the evaluation of region distances.

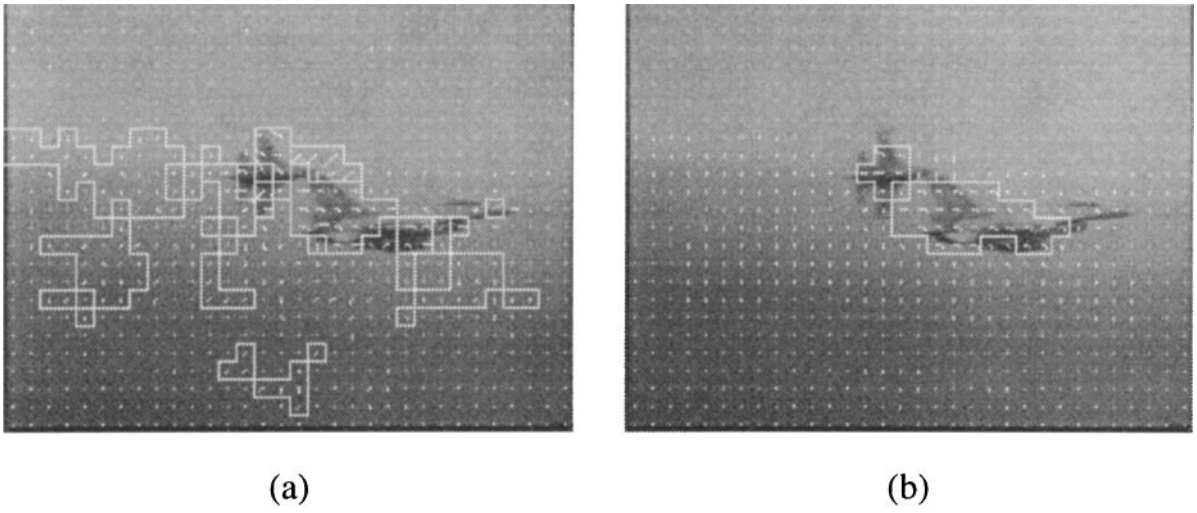

FIG. 5. Motion segmentation results (a) without and (b) with smoothing.

Figure 5 illustrates the motion segmentation results of a frame extracted from a TV news program. It is clear from Fig. 5a that without motion vector smoothing, wrong segmentation results are produced, even in a uniform and almost stationary background. On the contrary, only the actually moving objects are extracted in the case of smoothed motion vectors, as shown in Fig. 5b.

2.3. Fuzzy Feature Vector Formulation

All features extracted by the video sequence analysis module (i.e., size, location, color, or motion of each segment) can be used to describe the visual content of each video frame. However, they are not directly included in a vector to be used for this purpose, since their size differs between frames. For example, a frame consisting of 20 segments requires twice the number of feature elements than does a frame consisting of 10 segments. Moreover, there can be absolutely no correspondence between the elements of the feature vectors of two frames, making any comparison between the vectors unfeasible. To overcome this problem, we classify color as well as motion segments into predetermined classes, forming a multidimensional histogram. In this framework, each feature vector element corresponds to a specific feature class (equivalent to a histogram bin) and contains the number of segments that belong to this class. Segment size is accounted for by assigning separate feature classes for small and large segments: i.e., size is considered a segment feature just like color or motion. For example, a large moving segment is classified to a different feature class than a small moving segment. Although large objects might be considered more important than small ones, the above approach ensures that all information is kept separate and that, in a content-based retrieval environment, the degree of importance of each feature can be specified by the end user, possibly by assigning weights to feature vector elements [8].

In order to reduce the possibility of classifying two similar segments to different classes, causing erroneous comparisons, a degree of membership is allocated to each class, resulting in a fuzzy classification formulation [19]. In conventional histograms, each sample—i.e., segment, in our case—may belong only to one histogram bin, so that two similar samples, located, say, in opposite sides of the boundary of two bins, are considered to belong to different bins. Using fuzzy classification, each sample is allowed to belong to several (or all) classes, but with different degrees of membership. Therefore, in the previous example, the two similar samples would slightly differ in their degrees of membership with respect to the two adjacent bins. Fuzzy representation permits the user to perform more complex queries, such as seeking a blue and *somehow* large object, which is located *near* the bottom of an image.

Let us first consider the simple case of a one-dimensional feature s, e.g., the area of an image segment, taking values in a domain, which, without loss of generality, is assumed to be [0, 1]; i.e., feature s is normalized between 0 and 1. This domain is partitioned, or quantized, into Q classes by means of Q membership functions $\mu_n(s)$, $n = 1, 2, \ldots, Q$. For a given real value s, $\mu_n(s)$ denotes the degree of membership of s in the nth class. The membership functions $\mu_n(s)$, $n = 1, 2, \ldots, Q$, take values in the range [0, 1], so that values of $\mu_n(s)$ near unity (zero) indicate that the degree of membership of feature s in the nth class is high (low). The most common membership functions are the triangular ones, defined as

$$\mu_n(s) = \begin{cases} 1 - 2|s - m_n|/w, & |s - m_n| < w/2, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

for $n = 1, 2, \ldots, Q$, where w is the width of each triangle base and $m_n = (n-1)/(Q-1)$ is the center of each triangle, so that $m_1 = 0$ and $m_Q = 1$. An example of fuzzy classification using $Q = 5$ triangular membership functions of width $w = 2/(Q-1)$

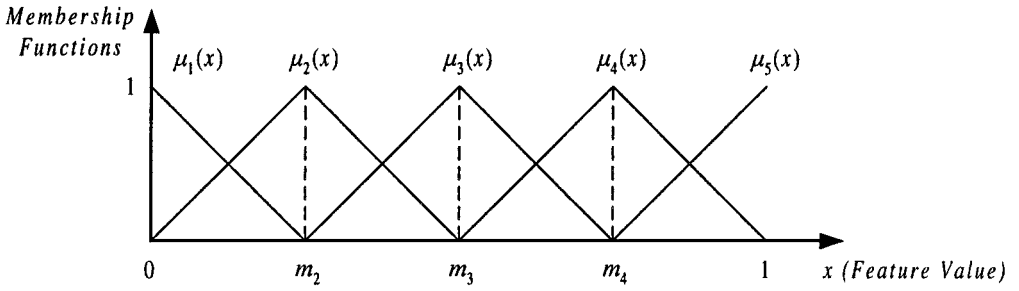

FIG. 6. Example of one-dimensional fuzzy classification using five triangular membership functions with 50% overlap between successive partitions.

is depicted in Fig. 6. It can be seen that width w controls the overlap between successive partitions, indicating how vague the classification is, and in this case 50% overlap is used. The exact shape and the overlap percentage of functions $\mu_n(s)$ can be greatly varied [19]. However, according to experimental results, the effect of the membership function shape on key frame extraction is minimal (except for cases of synthetic video sequences as explained in Subsection 3.1); therefore triangular functions have been selected mainly due to the very simple calculations involved.

Using this partition or quantization scheme, a fuzzy histogram can be constructed from a large number of feature samples s_i, $i = 1, \ldots, K$, each of which corresponds to an image segment, where K denotes the total number of segments. Then, the value of fuzzy histogram, say, $H(n)$ corresponding to the nth class is defined

$$H(n) = \frac{1}{K} \sum_{i=1}^{K} \mu_n(s_i), \quad n = 1, 2, \ldots, Q. \tag{3}$$

We should note that the above definition reduces to the definition of conventional histograms if membership functions take binary values (0 or 1). Since, however, each sample value has nonzero degree of membership to more than one class, the histogram can be meaningful even when the number of samples is small. Fuzzy representation thus permits the construction of histograms from a very limited set of data. This is very important since the number of segments in a frame, K, is typically much smaller than the total number of classes.

In the more general case of more than one segment features, such as color, motion, and location, a multidimensional feature vector is constructed for each segment. In particular, for each color segment S_i^c, $i = 1, \ldots, K$, an $L^c \times 1$ vector $\mathbf{s}_i^c$ is formed, while for each motion segment S_i^m, an $L^m \times 1$ vector $\mathbf{s}_i^m$ is formed:

$$\mathbf{s}_i^c = \left[\mathbf{c}^T\left(S_i^c\right)\mathbf{l}^T\left(S_i^c\right)a\left(S_i^c\right)\right]^T \tag{4a}$$

$$\mathbf{s}_i^m = \left[\mathbf{v}^T\left(S_i^m\right)\mathbf{l}^T\left(S_i^m\right)a\left(S_i^m\right)\right]^T, \tag{4b}$$

where a denotes the size of the color or motion segment, and $\mathbf{l}$ is a 2×1 vector, indicating the horizontal and vertical location of the segment center; the 3×1 vector $\mathbf{c}$ includes the average values of the color components of the color segment, while the 2×1 vector $\mathbf{v}$ includes the average motion vector of the motion segment. Thus, $L^c = 6$ for color segments and $L^m = 5$ for motion segments. For the sake of notational simplicity, the superscripts c and m will be omitted in the sequel; each color or motion segment will be denoted as S_i and will be described by the $L \times 1$ vector $\mathbf{s}_i$, where $L = 5$ or $L = 6$, depending on the segment type.

According to the above, let us denote by $\mathbf{s}_i = [s_{i,1}\, s_{i,2} \cdots s_{i,L}]^T$, $i = 1, 2, \ldots, K$, the vector describing the (color or motion) segment S_i, where K is the total number of segments. The domain of each element $s_{i,j}$, $j = 1, 2, \ldots, L$, of vector $\mathbf{s}_i$ is then partitioned into Q regions by means of Q membership functions $\mu_{n_j}(s_{i,j})$, $n_j = 1, 2, \ldots, Q$. As in the one-dimensional case, for a given real value of $s_{i,j}$, $\mu_{n_j}(s_{i,j})$ denotes the degree of membership of element $s_{i,j}$ to the class with index n_j. Gathering class indices n_j for all elements $j = 1, 2, \ldots, L$, and L-dimensional class $\mathbf{n} = [n_1\ n_2\ \cdots\ n_L]^T$ is defined. Then, the degree of membership of each vector $\mathbf{s}_i$ to class $\mathbf{n}$ can be performed through a product of the membership functions $\mu_{n_j}(s_{i,j})$ of all individual elements $s_{i,j}$ of $\mathbf{s}_i$ to the respective elements n_j of $\mathbf{n}$:

$$\mu_{\mathbf{n}}(\mathbf{s}_i) = \prod_{j=1}^{L} \mu_{n_j}(s_{i,j}). \tag{5}$$

In order for vector $\mathbf{s}_i$ to belong to class $\mathbf{n}$, all its elements $s_{i,j}$ should belong to the respective classes n_j. The membership functions $\mu_{n_j}(s_{i,j})$ should thus be combined with the "AND" operator, which is most commonly represented by multiplication in fuzzy logic.

A simple example of two-dimensional vectors is illustrated in Fig. 7. Assume that a segment S is described here by vector $\mathbf{s} = [s_1\ s_2]^T$, and $Q = 2$ membership functions $\mu_1(s_j)$ and $\mu_2(s_j)$ are used to quantize both elements s_j, $j = 1, 2$, of $\mathbf{s}$. Since $\mu_1(s_j)$ is used to express "low" values of s_j and $\mu_2(s_j)$ to express "high" values of s_j, we can denote classes n_j as "L" and "H" and the two membership functions as $\mu_{\mathrm{L}}(s_j)$ and $\mu_{\mathrm{H}}(s_j)$. The two-dimensional classes $\mathbf{n} = [n_1\ n_2]^T$ can then be denoted as "LL," "LH," "HL," and "HH," and the degree of membership of vector $\mathbf{s}$ to class $\mathbf{n}$ is $\mu_{\mathbf{n}}(\mathbf{s}) = \mu_{n_1}(s_1)\mu_{n_2}(s_2)$, or, taking all combinations, $\mu_{\mathrm{LL}}(\mathbf{s}) = \mu_{\mathrm{L}}(s_1)\ \mu_{\mathrm{L}}(s_2)$, $\mu_{\mathrm{LH}}(\mathbf{s}) = \mu_{\mathrm{L}}(s_1)\ \mu_{\mathrm{H}}(s_2)$, $\mu_{\mathrm{HL}}(\mathbf{s}) = \mu_{\mathrm{H}}(s_1)\,\mu_{\mathrm{L}}(s_2)$, and $\mu_{\mathrm{HH}}(\mathbf{s}) = \mu_{\mathrm{H}}(s_1)\,\mu_{\mathrm{H}}(s_2)$.

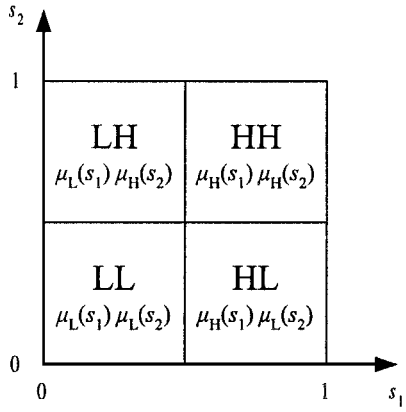

FIG. 7. Example of two-dimensional fuzzy classification using two partitions for each dimension.

It is now possible to construct a multidimensional fuzzy histogram from the segment feature samples $\mathbf{s}_i$, $i = 1, \ldots, K$, exactly as in the one-dimensional case. The value of the fuzzy histogram, $H(\mathbf{n})$, is defined similarly as the sum, over all segments, of the corresponding degrees of membership $\mu_{\mathbf{n}}(\mathbf{s}_i)$:

$$H(\mathbf{n}) = \frac{1}{K}\sum_{i=1}^{K}\mu_{\mathbf{n}}(\mathbf{s}_i) = \frac{1}{K}\sum_{i=1}^{K}\prod_{j=1}^{L}\mu_{n_j}(s_{i,j}). \qquad (6)$$

$H(\mathbf{n})$ thus can be viewed as a degree of membership of a whole frame to class $\mathbf{n}$. A frame feature vector $\mathbf{f}$ is then formed by gathering values of $H(\mathbf{n})$ for all classes $\mathbf{n}$, i.e., for all combinations of indices, resulting in a total of Q^L feature elements: $\mathbf{f} = [f_1 \;\; f_2 \;\; \cdots \;\; f_{Q^L}]^T$. In particular, an index function is defined which maps the Q^L feature vector elements into an integer between 1 and Q^L,

$$z(\mathbf{n}) = 1 + \sum_{j=1}^{L} n_j Q^{L-j}. \qquad (7)$$

Then, the elements f_i, $i = 1, \ldots, Q^L$, of feature vector $\mathbf{f}$ are calculated as $f_{z(\mathbf{n})} = H(\mathbf{n})$ for all classes $\mathbf{n}$. In fact, since the above analysis was based on features $\mathbf{s}_i^c$ and $\mathbf{s}_i^m$ of color segments S_i^c and motion segments S_i^c, respectively, two feature vectors will be calculated: a color feature vector $\mathbf{f}^c$ for color segments and a motion feature vector $\mathbf{f}^m$ for motion segments. Finally, based on color and motion feature vectors, the feature vector, of length $Q^{L^c} + Q^{L^m}$, corresponding to the whole frame, is formed as

$$\mathbf{f} = [(\mathbf{f}^c)^T\,(\mathbf{f}^m)^T]^T. \qquad (8)$$

It should be noted that the dimension of the feature vector $\mathbf{f}$, and consequently, the computational complexity, increases exponentially with respect to the number of partitions, Q. Moreover, a large number of partitions does not necessarily improve the effectiveness of the key frame extraction algorithm. On the contrary, it results in a very large number of classes, leading to "noisy" classification. Based on several experiments, we have concluded that a reasonable choice with respect to complexity and effectiveness is $Q = 3$.

Global frame characteristics, obtained through global frame analysis, could also be included as additional features in the feature vector, such as the color histogram of each frame or the average texture complexity, estimated using the ac DCT coefficients of each block derived from the MPEG stream. More segment properties could also be incorporated, such as contour shapes or high order moments. The feature vector would be much more representative of the frame content in this case, enabling selection of those features that are considered as more important for key frame extraction or content-based retrieval. It should be mentioned that the feature vector representation is independent of the key frame selection algorithms described in the sequel, so that any modification can be made without affecting the key frame selection module.

3. EXTRACTION OF KEY FRAMES

Once a feature-based representation of each frame is available, a shot feature vector can be constructed, characterizing a whole shot. One way of achieving this is by calculating the average value of the frame feature vectors over the whole shot duration. This information can be exploited for extracting a set of representative shots (*key shots*) using a shot-clustering algorithm, similar to that described in [11]. Key frames can then be selected from the key shots in order to provide a representation of a whole video sequence. The rest of the paper concentrates on key frame extraction from a given shot. Two approaches are proposed for this purpose. The first exploits the temporal variation of the frame feature vectors, while the second is an optimal solution, based on the minimization of a cross-correlation criterion which ensures that the selected frames are not similar to each other. In the following both methodologies are described, while results are given in Section 5.

3.1. Temporal Variation Approach

Since every frame in a shot corresponds to a specific time instance and is characterized by a specific feature vector, the feature vectors of all frames in the shot form a trajectory, or manifold, in a multidimensional space which expresses the temporal variation of the frame feature vectors of the shot. Therefore, selection of the most representative frames within a shot is equivalent to selection of the appropriate curve points which are able to characterize the corresponding trajectory. The selected curve points should provide sufficient information about the trajectory curve shape, so that the shape can be reproduced using some kind of interpolation. This can be achieved by extracting the time instances, i.e., the frame numbers which reside in the extreme locations of this trajectory. The magnitude of the second derivative of the feature vector with respect to time is used as a curvature measure in this case. The second derivative expresses

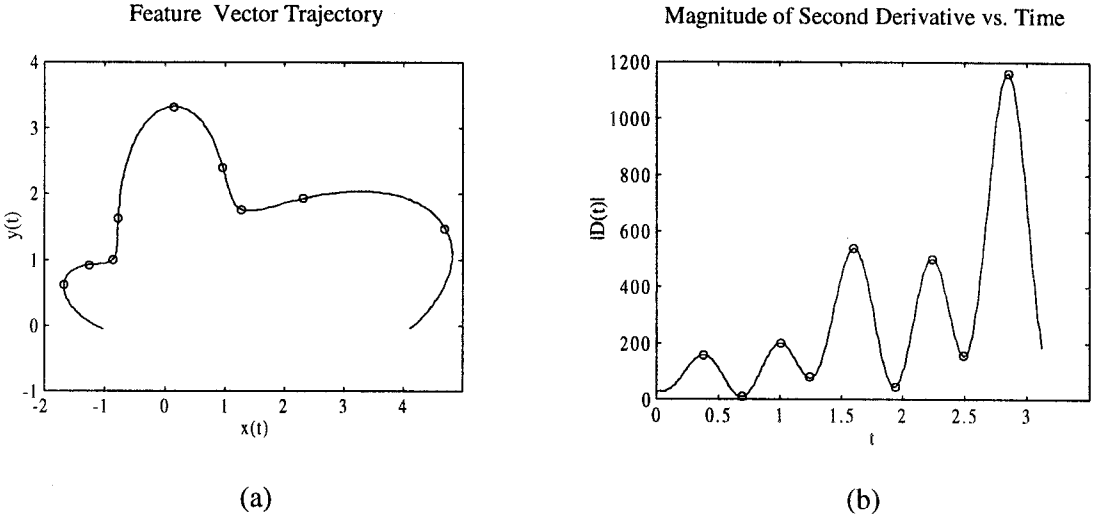

FIG. 8. (a) A continuous curve $\mathbf{r}(t) = (x(t), y(t))$, and (b) the magnitude of the second derivative $D(t)$ versus t.

the degree of acceleration or deceleration of an object that traces out the feature trajectory. Since local maxima correspond to time instances of peak variation of object velocity, while local minima correspond to almost constant velocity, representative frames are detected at those time instances. For example, suppose that we have a two-dimensional feature vector whose trajectory is illustrated in Fig. 8 as a continuous curve $\mathbf{r}(t) = (x(t), y(t))$. Then the local maxima and minima of the magnitude of the second derivative $D(t)$, shown as small circles in Figs. 8a, b, provide sufficient information about the curve shape, since it can be reproduced using some kind of interpolation.

Let us first assume that a video shot consisting of N_s images frames has been selected. Let us also denote as $\mathbf{f}(k)$, $k = 0, \ldots, N_s - 1$, the feature vector of the kth frame, as defined in Eq. (8). The first derivative of $\mathbf{f}(k)$ with respect to k is estimated, in discrete time, as the difference between two successive frames, i.e., $\mathbf{d}_1(k) = \mathbf{f}(k+1) - \mathbf{f}(k)$, $k \in \{0, \ldots, N_s - 2\}$. However, this operation is rather sensitive to noise, since differentiation of a signal amplifies its high-pass components. Thus, a weighted average of the first derivative is used over a window of predefined length to eliminate the noise influence, generating the first windowed derivative $\mathbf{d}_1^w(k)$,

$$\mathbf{d}_1^w(k) = \sum_{l=\alpha_1(k)}^{l=\beta_1(k)} w_{l-k}\mathbf{d}_1(l) = \sum_{l=\alpha_1(k)}^{l=\beta_1(k)} w_{l-k}(\mathbf{f}(l+1) - \mathbf{f}(l)), \quad k = 0, \ldots, N_s - 2, \tag{9}$$

where $\alpha_1(k) = \max(0, k - N_w)$, $\beta_1(k) = \min(N_s - 2, k + N_w)$, and $2 * N_w + 1$ is the length of the window, centered at frame k. It can been seen from Eq. (9) that the window length linearly reduces at shot limits. The weights w_l are defined for $l \in \{-N_w, N_w\}$; in the simple case of a rectangular window, they are all equal to $1/(2N_w + 1)$. The second windowed derivative, $\mathbf{d}_2^w(k)$, for the kth frame is defined in a similar way,

$$\mathbf{d}_2^w(k) = \sum_{l=\alpha_2(k)}^{l=\beta_2(k)} w_{l-k}\big(\mathbf{d}_1^w(l+1) - \mathbf{d}_1^w(l)\big), \quad k = 0, \ldots, N_s - 3, \tag{10}$$

where $\alpha_2(k) = \min(0, k - N_w)$, $\beta_2(k) = \min(N_s - 3, k + N_w)$, and w_l, $l \in \{-N_w, N_w\}$, equal the previous weights, assuming that the same window type is used for the first and second windowed derivative.

The elements of the second windowed derivative, $\mathbf{d}_2^w(k)$, express the variation of the elements of $\mathbf{f}(k)$ with respect to time. Thus, in order to take into consideration the variation of all elements of $\mathbf{f}(k)$, the magnitude of the derivative $D(k) = |\mathbf{d}_2^w(k)|$ is computed. The time instances corresponding to local maxima and minima of $D(k)$ are then detected as the key frame time instances. Note that $D(k)$ is a discrete time sequence here, in contrast with $D(t)$ of Fig. 8, which is a continuous curve.

To demonstrate the temporal variation method, an example of a synthetic video shot is examined below. The shot consists of $N_s = 100$ video frames of size 256×256 pixels and depicts a solid black circle of radius 25 pixels, following a vertical elliptic trajectory in a static background. Figure 9a illustrates the video shot, by presentating 20 of its frames, whose time instances are uniformly distributed between 1 and 100. As is observed, the ball traces the elliptic trajectory twice. Let us take into account only color frame features; i.e., let the feature vector equal $\mathbf{f}(k) = \mathbf{f}^c(k)$, Two partitions (classes) are defined for each feature ($Q = 2$), resulting in a feature vector length of $Q^{L^c} = 2^6 = 64$. The partition indices in this case are $n_i \in \{1, 2\}$, $i = 1, \ldots, 6$, with $n_i = 1$ representing a "low" value and $n_i = 2$ representing a "high" value for the ith feature. This interpretation of feature values as "low" and "high" is explained in the two-dimensional example of Subsection 2.3. Two triangular membership functions

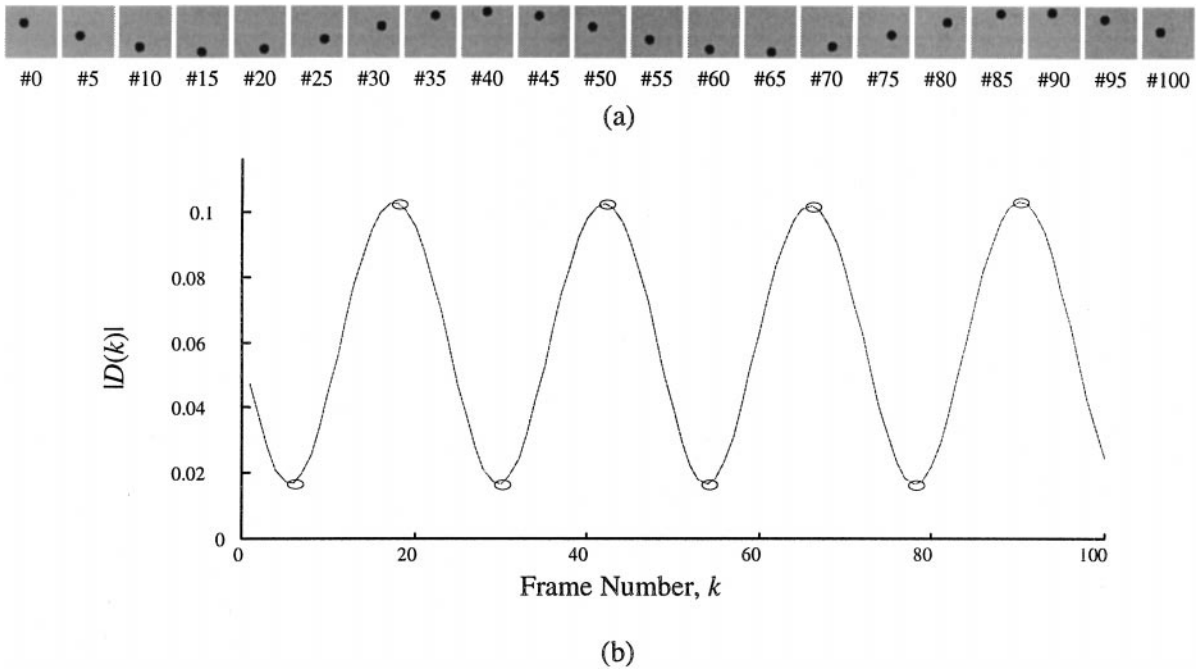

FIG. 9. Temporal variation approach example: (a) synthetic video sequence; (b) magnitude of second windowed derivative, $|D(k)|$, versus the frame number, k.

are used for each feature with 50% overlap. As observed the color feature vector varies, although the color components of this sequence remain constant. This is due to the fact that the color feature vector, $\mathbf{f}^c(k)$, also contains geometric properties of the color segments and, in particular, the horizontal and vertical location of the segment centers.

Figure 9b depicts $D(k)$ for all shot frames $k = 0, \ldots, N_s - 1 = 99$. Four local maxima and four local minima are present in this figure, indicated as small circles. Local maxima correspond to frames where the circle reaches the outmost vertical positions (top and bottom of the image) while local minima correspond to frames where the circle reaches the outmost horizontal positions. Since the trajectory is traced twice, the first two local maxima and minima correspond to the first period while the other two correspond to the second one. The time instances of local maxima and minima are selected as key frame instances. Only eight frames out of 100 are thus required to represent the shot content. This leads to a 92% reduction of the storage requirements. Although the video information has been reduced at the same ratio, the visual content of the sequence is retained since the most representative frames are extracted.

Figure 10 depicts the eight selected frames, together with the 64 feature elements of $\mathbf{f}(k)$ corresponding to each key frame. It is observed that two main groups of feature elements have nonzero values for each feature vector of this figure. The first group corresponds to the small black circle, while the second group corresponds to the background area. In this example, class indices n_1, n_2, and n_3 correspond to R, G, and B color components, n_4 and n_5 correspond to horizontal and vertical segment location (x and y, respectively), and n_6 corresponds to segment size. Thus, classes $\mathbf{n}$ of the first group, whose index function $z(\mathbf{n})$ has integer values in $\{1, \ldots, 8\}$, correspond to dark red, dark green, and dark blue color components (circle group). Similarly, classes $\mathbf{n}$ of the second group with $z(\mathbf{n})$ in $\{25, \ldots, 32\}$ correspond to dark red, light green, and light blue color components (background group). The background group remains static during the shot while the circle group changes, since both the horizontal and vertical location of the circle segment fluctuate. Even integer values of $z(\mathbf{n})$ correspond to large segment size while old ones correspond to small segment size. As a result, the background group is characterized by zero values at odd indices, while the circle group is characterized by zero values at the even indices.

Figure 11 presents plots of feature elements f_1, f_3, and f_5 versus one another for the circle segment. These elements refer to the same classes n_1, n_2, n_3, and n_6 (i.e., R, G, B, and segment size), and differ only in classes n_4 and n_5 (segment location x and y). In particular, f_1 corresponds to "low" x and "low" y (LL), f_3 to "low" x and "high" y (LH), and f_5 to "high" x and "low" y (HL), similarly to the two-dimensional example of Subsection 2.3. In addition, Fig. 12 illustrates plots of specific sums of two feature elements ($f_1 + f_3$, $f_1 + f_5$, and $f_3 + f_5$) versus one another. Since f_1 refers to "low" y and f_3 to "high" y, adding them actually removes the effect of vertical location y, and thus the sum $f_1 + f_3$ refers to "low" x, independently of y. Likewise, the sum $f_1 + f_5$ refers to "low" y independently of x. The locations of the selected key frames are also shown as small circles in these figures. In fact, these plots represent projections of the feature vector trajectory onto the subspace defined by the respective feature elements. It is observed that, in all cases, the selected key frames reside near the extreme

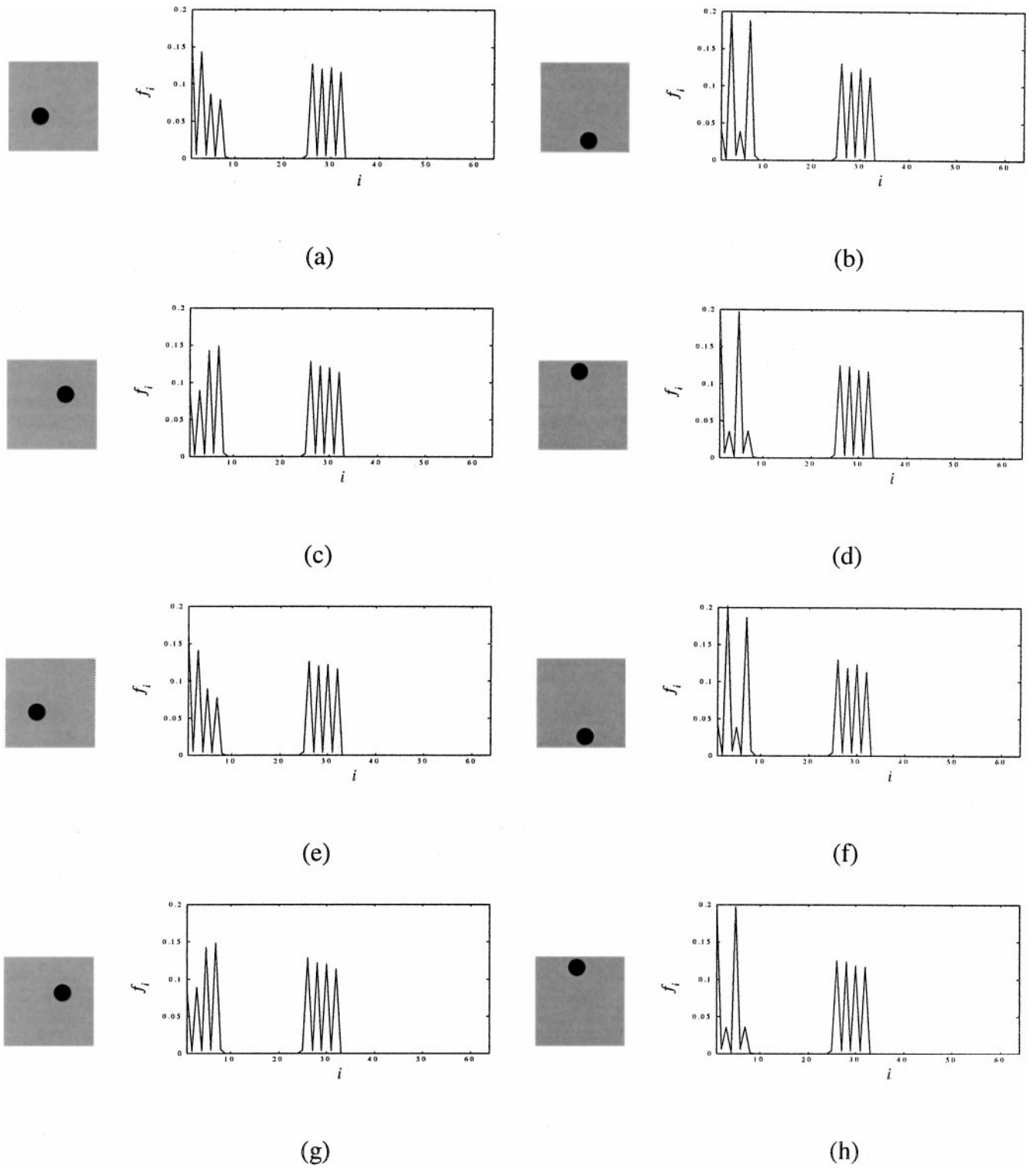

FIG. 10. Set of eight selected key frames from synthetic video sequence example, along with plots of the respective feature vectors.

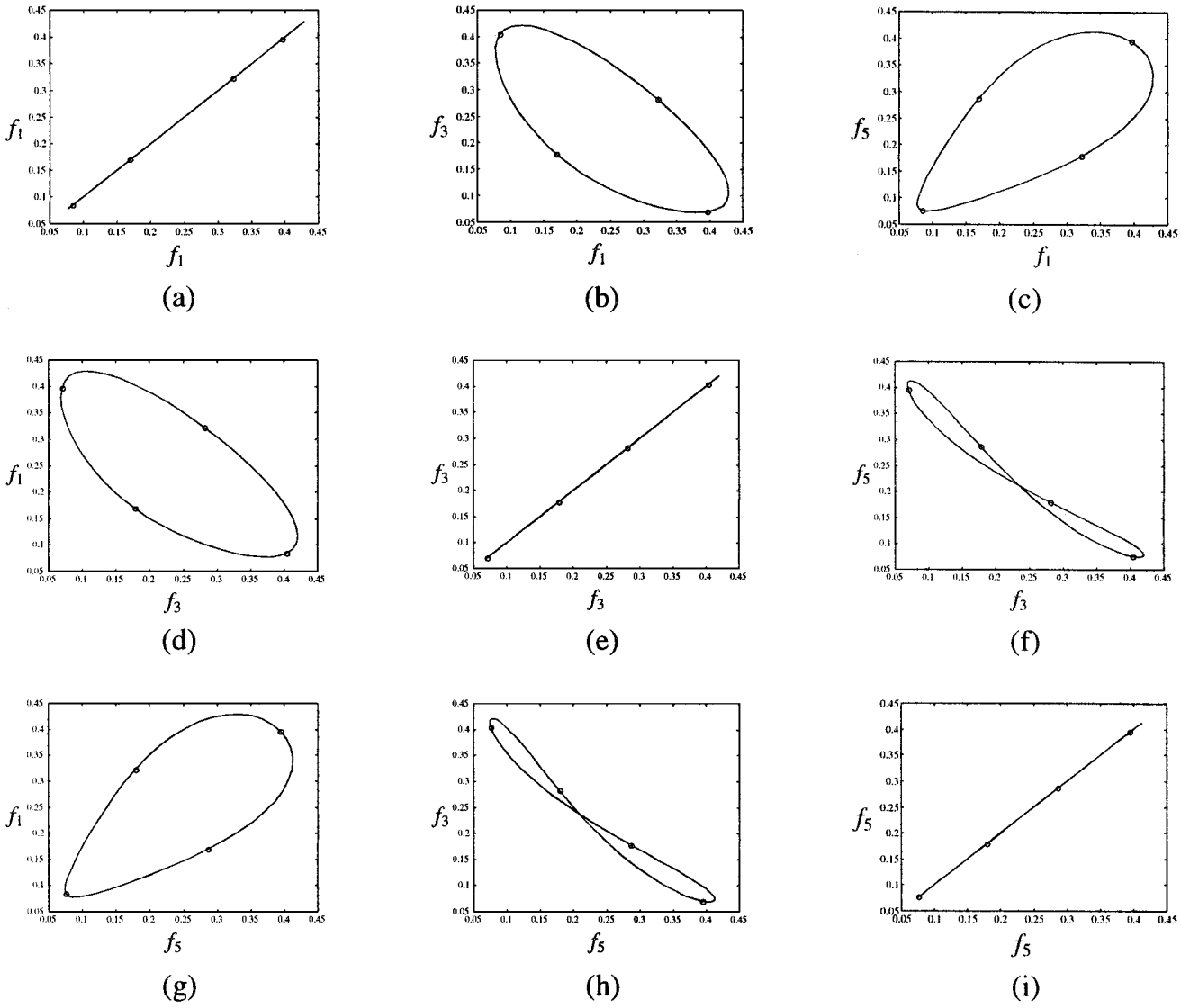

FIG. 11. Two-dimensional plots of all pair combinations of feature elements f_1, f_3, and f_5, corresponding to specific combinations of horizontal and vertical locations.

locations of the projected trajectory. A plot of particular interest is that depicted in Fig. 12b, where in effect the horizontal location of the circle is plotted versus the vertical location. In this case, it is ascertained that the elliptic movement of the circle is extracted.

Frame extraction based on temporal feature vector variation is an extremely fast and very straightforward algorithm since, in discrete time, the second derivative is implemented as a difference equation. In addition, the number of key frames for a shot is not required to be known a priori. Instead, it is estimated by the feature vector trajectory. In cases where constant variation of the feature vector is presented versus time, the second derivative may not work well for detecting the representative frames of a shot. However, these are rare situations that might be present only in synthetic video sequences; instead, in real world ones, feature elements do not obey a specific mathematical or physical law. This behavior can be eliminated, even in the case of synthetic sequences, by the use of a slightly higher number (e.g., 3 or 4) of membership functions. The use of nonlinear membership functions, such as sigmoid or Gaussian, also helps in this direction, since triangular functions may be nonlinear, but consist of linear segments.

3.2. Cross-Correlation Approach

As demonstrated in the previous example, the temporal variation approach has the ability of detecting several repetitions of the content of a frame. This is useful for understanding the flow of action in a video shot. In cases, however, where temporal evolution of the shot is not of great interest, this approach

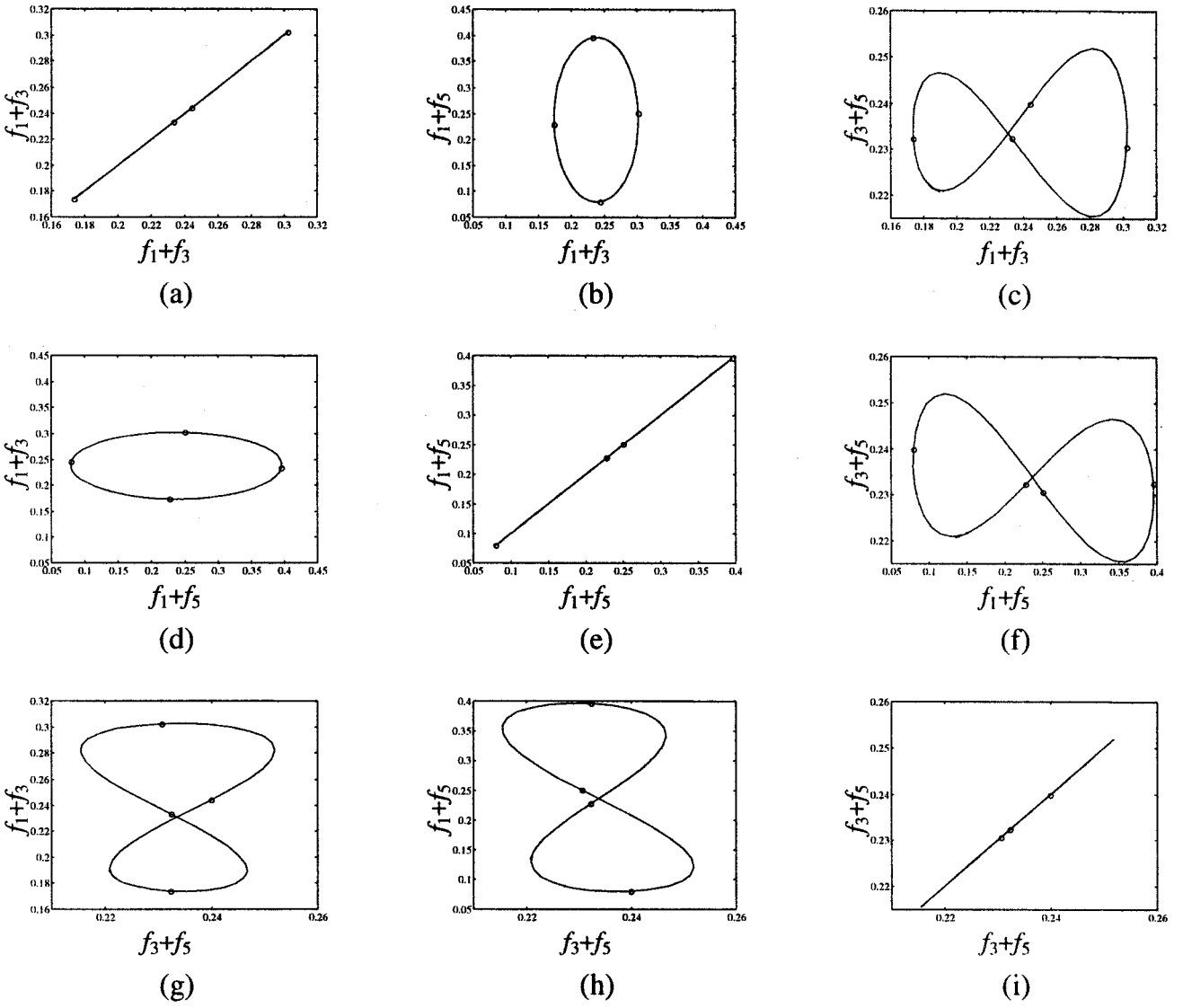

FIG. 12. Two-dimensional plots of all pair combinations of sums $f_1 + f_3$, $f_1 + f_5$, and $f_3 + f_5$.

does not provide a compact video content representation, since it contains redundant information. In such cases, which generally include complex shots of large duration, it is necessary to select a small number of frames that are representative of the shot and are not similar to each other. For this reason a key frame selection algorithm is introduced, based on an optimization method for locating a set of minimally correlated feature vectors. This is achieved by minimizing a cross-correlation criterion among the frames of a given shot.

Let us recall that $\mathbf{f}(k)$ is the feature vector of the kth frame of the shot under examination, with $k \in V = \{0, 1, \ldots, N_s - 1\}$, where N_s is the total number of frames in the shot. Let us also denote by K_s the number of key frames that should be selected. This number is either known a priori or it can be estimated by the temporal variation algorithm, as was described in the previous subsection. In particular, using several experiments it can be shown that in most cases the number K_s should be approximately half of the key frames extracted by the temporal variation method. Then, the correlation coefficient of two feature vectors $\mathbf{f}(k)$, $\mathbf{f}(l)$ is defined as $\rho_{k,l} = C_{k,l}/(\sigma_k \sigma_l)$, with $k, l \in \{0, \ldots, N_s - 1\}$, where $C_{k,l} = (\mathbf{f}(k) - \mathbf{m})^T (\mathbf{f}(l) - \mathbf{m})$ is the covariance of the two vectors $\mathbf{f}(k)$, $\mathbf{f}(l)$, while $\mathbf{m} = \sum_{i=0}^{N_s-1} \mathbf{f}(i)/N_s$ is the average feature vector of the shot and $\sigma_i^2 = C_{i,i}$ is the respective variance. Without loss of generality, it is next assumed that $N_s = 2^M$, where M is an integer number; i.e., the number of shot frames is a power of 2. In case the actual number does not meet this constraint, extension of the shot is performed by adding dummy frames. In this case, correlation coefficients of dummy frames are set to infinity so that they cannot be selected as key frames.

Based on the correlation coefficients between pairs of feature vectors, a measure of correlation among K_s feature vectors can be defined. For this purpose, an index vector is defined

first,

$$\mathbf{x} = (x_1, \ldots, x_{K_s}) \in W \subset V^{K_s}, \tag{11}$$

where

$$W = \{(x_1, \ldots, x_{K_s}) \in V^{K_s} : X_1 < \cdots < x_{K_s}\} \tag{12}$$

is the subset of V^{K_s} containing all sorted index vectors $\mathbf{x}$ corresponding to sets of frame numbers or time indices. The correlation measure of the feature vectors $\mathbf{f}(k),\ k = x_1, \ldots, x_{K_s}$, can then be defined as

$$R(\mathbf{x}) = R(x_1, \ldots, x_{K_s}) = \frac{2}{K_s(K_s-1)} \sum_{i=1}^{K_s-1} \sum_{j=i+1}^{K_s} (\rho_{x_i,x_j})^2, \tag{13}$$

taking values in the real interval [0, 1]. Based on the above definition, it is clear that searching for a set of K_s minimally correlated feature vectors is equivalent to searching for an index vector $\mathbf{x}$ that minimizes $R(\mathbf{x})$. Searching is limited in the subset W, since index vectors are used to construct sets of feature vectors. Thus, any permutations of the elements of $\mathbf{x}$ will result in the same sets. It is clear that the correlation measure of the K_s features is independent of the feature arrangement. Finally, the set of the K_s least correlated feature vectors, corresponding to the K_s most characteristic frames, is represented by

$$\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_{K_s}) = \arg\min_{\mathbf{x} \in W} R(\mathbf{x}). \tag{14}$$

Unfortunately, the complexity of an exhaustive search for obtaining the minimum value of $R(\mathbf{x})$ is such that a direct implementation of the method is practically unfeasible. For example, about 264 million combinations of frames should be considered (each of which requiring several computations for estimation of $R(\mathbf{x})$) if we wish to select five representative frames out of a shot consisting of 128 frames. For this purpose, two methods are proposed next for efficient implementation of the optimization procedure: the logarithmic search and the genetic algorithm.

4. IMPLEMENTATION OF THE CROSS-CORRELATION APPROACH

4.1. Logarithmic Search Algorithm

The first approach is based on a technique similar to the one used in MPEG standards for block motion estimation [39]. The main difference is that it is implemented in the multidimensional space W. In particular, instead of performing an exhaustive search over all indices of W, a single path of points is followed, beginning from a certain initial point. At each point of the path, only the set of its neighbors is examined, so that the next point in the path is selected toward the direction of the neighbor corresponding to the minimum cross-correlation measure. In each iteration of the algorithm, the neighboring region is decreased until it reduces to a single point, which is selected as a solution of the optimization problem. Although a very small subset of the search space W is considered, the algorithm presents good performance, since frames that are close to each other (in time), usually have similar properties, and therefore, indices which are close to each other (in W) should have similar correlation measures.

An initial index vector, say, $\mathbf{x}(0)$ should be selected at the initialization of the algorithm. A common choice would be to locate the initial index vector at the middle point $\tilde{\mathbf{x}} = (\mu, \ldots, \mu)$, where $\mu = 2^{M-1} - 1$ is the central time instance of the shot for a shot length $N_s = 2^M$. However, this point does not belong to space W, since its elements do not satisfy the corresponding inequality properties, as defined in Eq. (12). Hence $\mathbf{x}(0)$ is selected as the element of W which is closest to the middle point $\tilde{\mathbf{x}}$,

$$\mathbf{x}(0) = (\mu - \lfloor K_s/2 \rfloor, \ldots, \mu - 1, \mu + 1, \ldots, \mu + \lfloor K_s/2 \rfloor) \tag{15a}$$

if K_s is even and

$$\mathbf{x}(0) = (\mu - \lfloor K_s/2 \rfloor, \ldots, \mu - 1, \mu, \mu + 1, \ldots, \mu + \lfloor K_s/2 \rfloor) \tag{15b}$$

if K_s is odd. Let us now assume that, at the nth iteration of the algorithm, an index vector $\mathbf{x}(n)$ has been selected. The next index vector $\mathbf{x}(n+1)$ is then obtained by evaluating the correlation measure of all neighbors of $\mathbf{x}(n)$ in a region defined as

$$N(\mathbf{x}(n), \delta(n)) = \{\mathbf{y} \in W : \mathbf{y} = \mathbf{x}(n) + \delta(n)\mathbf{p},\ \mathbf{p} \in G^{K_s}\}, \tag{16}$$

where $G = \{-1, 0, 1\}$ and $\delta(n)$ is an integer indicating the step size of the neighborhood region. The above equation indicates that the neighbors of $\mathbf{x}(n)$ are located on the lattice G^{K_s} expanded by the step size $\delta(n)$. The step size is initialized as $\delta(0) = 2^{M-2}$ so that the algorithm covers all possible points of the space W. Based on the above, the actions that are repeated in each iteration of the algorithm are (i) to select the neighbor of $\mathbf{x}(n)$ with the minimum correlation measure as the next index vector $\mathbf{x}(n+1)$ and (ii) to divide the step size by two:

$$\mathbf{x}(n+1) = \arg\min_{\mathbf{x} \in N(\mathbf{x}(n), \delta(n))} R(\mathbf{x}) \tag{17a}$$

$$\delta(n+1) = \delta(n)/2. \tag{17b}$$

The above steps are repeated for $n = 0, 1, \ldots, M-2$, until $\delta(n) = 1$. After $M-1$ iterations the algorithm stops and the final result is $\hat{\mathbf{x}} = \mathbf{x}(M-1)$. This means that the time indices of the K_s key frames of the shot are selected as the elements of the vector $\mathbf{x}(M-1)$. Figure 13 depicts a graphical representation of the algorithm for $K_s = 2$ and $N_s = 16$ ($M = 4$), where the horizontal and vertical axes correspond to the two elements, x_1 and x_2, of the index vector $\mathbf{x}$. The dark region includes index vectors that

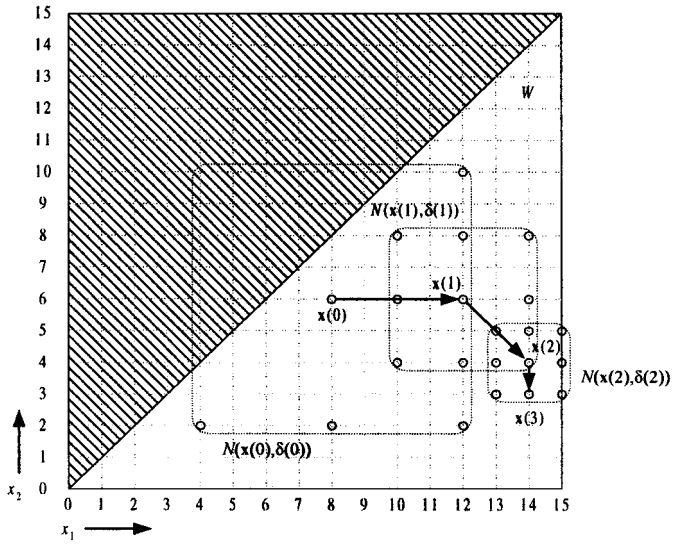

FIG. 13. Graphical representation of the logarithmic search algorithm for $K_s = 2$ and $N_s = 16\,(M = 4)$. The dark region indicates index vectors that do not belong to space W.

do not belong to space W. The neighborhood area of each index vector is represented by a dotted rectangle, while the neighbors inside W are shown as small circles.

Although the proposed scheme significantly reduces the required time for estimating key frames, it cannot conclude a different solution than that provided by the examined path. Consequently, it is most frequently trapped in a local minimum of $R(\mathbf{x})$. To make the algorithm more flexible, a stochastic approach is introduced in the following, providing the possibility of considering more than one different paths. Experimental results, which indicate the performance of both algorithms, are given in Section 5.

4.2. Stochastic Approach

The main difference of this algorithm is the introduction of a stochastic term in the selection of the next index vector in each iteration. Therefore Eq. (17a) is modified, based on a probabilistic criterion, while the rest of the algorithm remains the same. The concept of this stochastic approach is to assign a probability to every neighbor point of the current examined point $\mathbf{x}(n)$, i.e., every point belonging to the set $N(\mathbf{x}(n), \delta(n))$, and then to select the next index vector $\mathbf{x}(n+1)$, using the assigned probabilities. These probabilities are inversely proportional to the respective correlation measure. The search procedure is repeated several times, so that, in effect, multiple logarithmic search experiments take place in a random way. Due to the stochastic behavior of the algorithm, different neighbors are selected in every new experiment, resulting in the generation of several random paths.

Let us denote by $\mathbf{x}_m(n)$ the index vector at the nth iteration step for the mth experiment, and by $\mathbf{y}_i$, $i = 1, \ldots, |N|$, its neighbors, i.e., the elements of set $N(\mathbf{x}_m(n), \delta(n))$, where $|N|$ is the set cardinality. Then, a probability value is assigned to each $\mathbf{y}_i$, according to the respective correlation measure as

$$p_i = 1 - \frac{R(\mathbf{y}_i)}{\sum_{j=1}^{|N|} R(\mathbf{y}_j)}, \quad i = 1, \ldots, |N|. \tag{18}$$

A cumulative probability function is then constructed for all $\mathbf{y}_i$, i.e., $q_i = \sum_{j=1}^{i} p_j$, $i = 1, \ldots, |N|$, with $q_0 = 0$. Using a given random number r, uniformly distributed in the range $[0, 1]$, the next index vector $\mathbf{x}_m(n+1)$ is chosen among the neighbors $\mathbf{y}_i$ as

$$\mathbf{x}_m(n+1) = \left\{\mathbf{y}_i \in N(\mathbf{x}_m(n), \delta(n)) : q_{i-1} < r \le q_i\right\}. \tag{19}$$

The iteration is repeated $n = 0, 1, \ldots, M-2$, as in the case of the logarithmic search algorithm, and the result of the mth experiment is the index vector $\hat{\mathbf{x}}_m = \arg\min_{i=0,\ldots,M-1} R(\mathbf{x}_m(i))$ corresponding to the minimum correlation measure along the path of the experiment. The final result is the index vector corresponding to the minimum correlation measure of all vectors in all experiments. After J experiments, the optimal solution $\hat{\mathbf{x}} = \arg\min_{m=1\ldots,J} R(\hat{\mathbf{x}}_m)$ is selected, containing the indices of the K_s key frames.

The stochastic version of the logarithmic search provides better results (see Section 5) than the logarithmic one, since different random paths are generated in each step of the algorithm. The search space W is thus explored in a more efficient way, since local minima of $R(\mathbf{x})$ cannot trap the algorithm. As the number of experiments increases, the number of examined points increases too; thus the solution obtained by the algorithm reaches the optimal one. However, in this case the complexity increases as well, as shown in the experimental results below. For this reason, a genetic algorithm is proposed in the following for efficient key frame selection.

4.3. Genetic Algorithm

As we have seen, the logarithmic search algorithm provides very fast convergence to a suboptimal solution of the correlation minimization problem, with a significant possibility of converging to a local minimum of $R(\mathbf{x})$. This drawback is alleviated by the use of its stochastic version at a higher computational cost. The idea of using a guided random search procedure can be further extended by employing an evolution program (EP) [22]. In contrast to enumerative search techniques, such as dynamic programming, which may break down on complex problems of moderate size, evolutionary programs provide unique flexibility and robustness on such problems. For this reason, a genetic algorithm (GA) [13] approach is adopted next. GAs are a special case of EPs, mainly used for discrete optimization problems. The approach seems to be very efficient for the particular optimization problem, given the size and dimensionality of the search space and the multimodal nature of the objective function. This is evident in Section 5, where experimental results are presented.

Possible solutions of the optimization problem, i.e., sets of frames, are represented by chromosomes whose genetic material consists of frame numbers (indices). Chromosomes are thus represented by index vectors $\mathbf{x} = (x_1, \ldots, x_{K_s}) \in W$ following an integer number *encoding scheme*, that is, using integer numbers for the representation of chromosome elements (genes) x_i, $i = 1, \ldots, K_s$. The reason for selecting integer numbers (instead of binary) representation is that all genetic operators, such as crossover and mutation, should only be applied to genes x_i, and not to arbitrary bits of their binary representation. An *initial population* of P chromosomes, $\mathbf{X}(0) = \{\mathbf{x}_1, \ldots, \mathbf{x}_P\}$ is then generated by selecting P sets of frames whose feature vectors reside in extreme locations of the feature vector trajectory, as described in the temporal variation approach. Since we do have some knowledge about the distribution of local optima, the above approach exploits the temporal relation of feature vectors and increases the possibility of locating sets of feature vectors with small correlation within the first few GA cycles.

The correlation measure $R(\mathbf{x})$ is used as an objective function to estimate the performance of all chromosomes $\mathbf{x}_i$, $i = 1, \ldots, P$, in a given population. However, a *fitness function* is used to map objective values to fitness values, following a *rank-based normalization scheme*. In particular, chromosomes $\mathbf{x}_i$ are ranked in ascending order of $R(\mathbf{x}_i)$, since the objective function is to be minimized. Let $\text{rank}(\mathbf{x}_i) \in \{1, \ldots, P\}$ be the rank of chromosome $\mathbf{x}_i$, $i = 1, \ldots, P$ (rank $= 1$ corresponds to the best chromosome and rank $= P$ to the worst). Defining an arbitrary fitness value F_B for the best chromosome, the fitness $F(\mathbf{x}_i)$ of the ith chromosome is given by the linear function

$$F(\mathbf{x}_i) = F_B - [\text{rank}(\mathbf{x}_i) - 1]D, \quad i = 1, \ldots, P, \tag{20}$$

where D is a decrement rate. The major advantage of the rank-based normalization is that, since fitness values are uniformly distributed, it prevents the generation of *super chromosomes*, avoiding premature convergence to local minima. Furthermore, by simply adjusting the two parameters F_B and D, it is very easy to control the *selective pressure* of the algorithm, effectively influencing its convergence speed to a global minimum.

After fitness values, $F(\mathbf{x}_i)$, $i = 1, \ldots, P$, have been calculated for all members of the current population, *parent selection* is then applied so that a more fit chromosome gives a higher number of offspring and, thus, has a higher chance of survival in the next generation. The *roulette wheel selection* procedure [22] is used for parent selection, by assigning each chromosome a probability of selection proportional to its fitness values, exactly as in Eq. (19) for neighbor selection in the stochastic logarithmic approach. The roulette wheel selection is one of the most popular methods, because it ensures that each chromosome has a growth rate proportional to its fitness value. Note also that, due to rank-based normalization, selection probabilities remain constant between generations.

A set of new chromosomes (offspring) is then produced by mating the selected parent chromosomes and applying a *crossover operator*. The genetic material of the parents is combined in a random way in order to produce the genetic material of the offspring. For example, for a single random crossover point at position c, two parents

$$\mathbf{a} = (a_1, a_2, \ldots, a_c, a_{c+1}, \ldots, a_{K_s}),$$
$$\mathbf{b} = (b_1, b_2, \ldots, b_c, b_{c+1}, \ldots, b_{K_s})$$

would generate the offspring

$$\mathbf{a}' = (a_1, a_2, \ldots, a_c, b_{c+1}, \ldots, b_{K_s}),$$
$$\mathbf{b}' = (b_1, b_2, \ldots, b_c, a_{c+1}, \ldots, a_{K_s}).$$

A more general technique, employed in the context of this paper, is the *uniform crossover*, where each parent gene is considered to be a potential crossover point. This means that two parents

$$\mathbf{a}_0 = (a_1^0, a_2^0, \ldots, a_K^0), \quad \mathbf{a}_1 = (a_1^1, a_2^1, \ldots, a_K^1)$$

generate two offspring:

$$\mathbf{a}_0' = (a_1^{s_1}, a_2^{s_2}, \ldots, a_K^{s_K}), \quad \mathbf{a}_1' = (a_1^{1-s_1}, a_2^{1-s_2}, \ldots, a_K^{1-s_K}),$$

where K_s has been replaced by K for notational convenience and s_i, $i = 1, \ldots, K_s$, are random numbers taking values of 0 or 1 with equal probabilities, so that each component comes from the first or the second parent. Although single-point crossover is considered to be inferior to other techniques, no evidence has been reported in favor of uniform, multipoint, or other types of crossover operators (such as *arithmetical*, *segmented*, or *shuffle*) [22]. Instead, this selection is heavily problem-dependent, and in our case, uniform crossover has exhibited slightly better performance in the experiments.

The next step is to apply *mutation* to the newly created chromosomes, introducing random gene variations that are useful for restoring lost genetic material, or for producing new material that corresponds to new search areas. *Uniform mutation* is the most common mutation operator and is selected for our optimization problem. In particular, each offspring gene x_i is replaced by a randomly generated one $x_i' \in W$, with probability p_m. That is, a random number $r \in [0, 1]$ is generated for each gene and replacement takes place if $r < p_m$; otherwise the gene remains intact. Other alternatives, such as *nonuniform*, *boundary*, or *swap* operators, are also possible. Nonuniform mutation is in general preferable in numerical optimization problems with respect to accuracy and convergence speed, but does not achieve better performance in the problem under consideration.

Once new chromosomes have been generated for a given population $\mathbf{X}(n)$, $n \geq 0$, the next generation population, $\mathbf{X}(n+1)$, is formed by inserting these new chromosomes into $\mathbf{X}(n)$ and deleting an appropriate number of older chromosomes, so that each population consists of P members. The exact number of old

chromosomes to be replaced by new ones defines the *replacement strategy* of the GA and greatly affects its convergence rate. An *elitist* strategy has been selected for replacement, where a small percentage of the best chromosomes is copied into the succeeding generation, together with their offspring, improving the convergence speed of the algorithm [13]. Several GA cycles take place by repeating the procedures of fitness evaluation, parent selection, crossover, and mutation, until the population converges to an optimal solution. The GA terminates when the best chromosome fitness remains constant for a large number of generations, indicating that further optimization is unlikely.

5. EXPERIMENTAL RESULTS

An MPEG video database consisting of real life video sequences is used in the following to test the performance of the proposed algorithm. The database consists of video sequences of total duration about 3.5 h and includes several shots of news programs, films, commercials, sports, and cartoons. The sequences have been encoded using the Optibase Fusion MPEG encoder at a bitrate of 2 Mbits/s. The shot detection and feature extraction algorithms have been applied offline to all sequences, so that all information regarding the shot change instances, as well as the feature vector representation of all frames, is stored in the database and is readily available. Hence, the key frame extraction algorithms are separately performed on each shot, using directly the feature vectors of all frames within the respective shot. The feature domains are partitioned in $Q = 3$ classes using three triangular membership functions with 50% overlap, so that the total feature vector length is $Q^{L_c} + Q^{L_m} = 972$ for $L_c = 6$ and $L_m = 5$, as mentioned in Section 2.3. The number of key frames for the cross-correlation methods is determined as the integer part of half the number obtained from the temporal variation approach. This selection gives satisfactory results in most cases, as mentioned in Subsection 3.2.

One shot of the database is used for demonstration of the performance of the proposed techniques. The shot, coming from a test drive sequence and consisting of $N_s = 223$ frames, is illustrated in Fig. 14. One every 10 frames is depicted, resulting in 23 frame thumbnails. The results of the temporal variation approach on this shot are presented in Fig. 15. In particular, Fig. 15a shows the magnitude of the second windowed derivative, $|D(k)|$, versus the frame number, k. The variation of $|D(k)|$ denotes that, due to the complexity of the shot, the feature vector manifold is much more complex than that of the synthetic example of Fig. 9b. Still, the smoothness of the curve is maintained, due to the windowing procedure described in Section 3; this ensures that local extrema of $|D(k)|$, shown with small circles in Fig. 15a, actually correspond to variations of the shot content and not to segmentation noise. The seven selected frames are depicted in Fig. 15b. It can be seen that these frames provide sufficient visualization of the total 223 frames of the shot. Some of them, however, are similar to each other (e.g., #30 and #55). For this reason, the remaining implementations of the cross-correlation approach are applied, considering a smaller number of key frames; in particular, K_s is determined using the rule mentioned above. However, in this specific experiment, $K_s = 4$ (slightly greater) was selected in order to sufficiently describe the visual content of the shot.

The results of the cross-correlation approach are shown in Fig. 16. In order to estimate the performance of the algorithms in terms of the obtained correlation measure $R(\mathbf{x})$, a test of 100,000 random index vectors is first performed, and a histogram of $R(\mathbf{x})$ is constructed, as depicted in Fig. 16a. The optimal values of $R(\mathbf{x})$ obtained through the three algorithms are then compared to the minimum value of the histogram. In Fig. 16a, these values are shown with a vertical dashed line for the logarithmic search, a vertical solid line for the stochastic version, and a vertical dotted line for the genetic algorithm. It is first observed that all three algorithms return values that are lower than the minimum value of the histogram. Second, it is clearly shown that the genetic algorithm provides much more accurate results. Actually, the minimum value obtained through the genetic algorithm is much lower than that of the random test, although the random test requires about 100 times more computational time. Finally, it is illustrated that key frames are extracted, based on an objective numerical criterion, i.e., minimization of the cross correlation function of frame feature vectors. Other criteria, which take account of human perception, can also be used to evaluate the

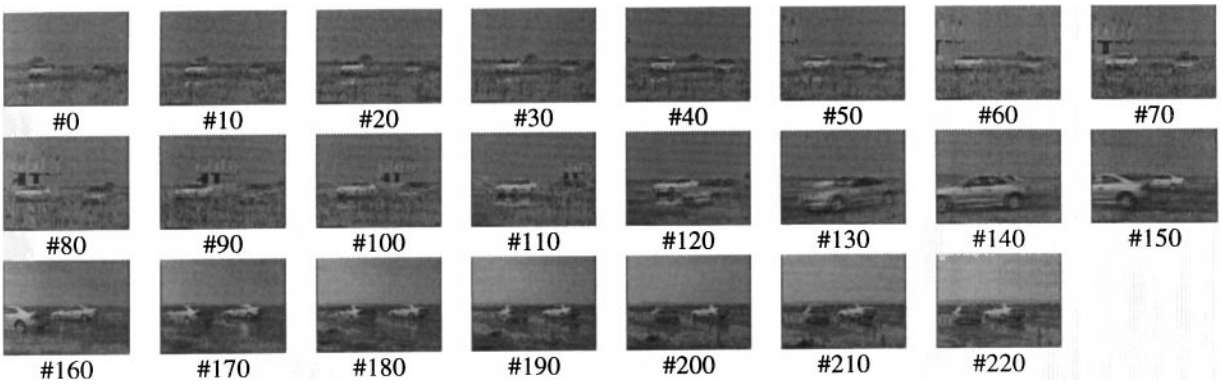

FIG. 14. Test drive sequence, frames #0 to #220.

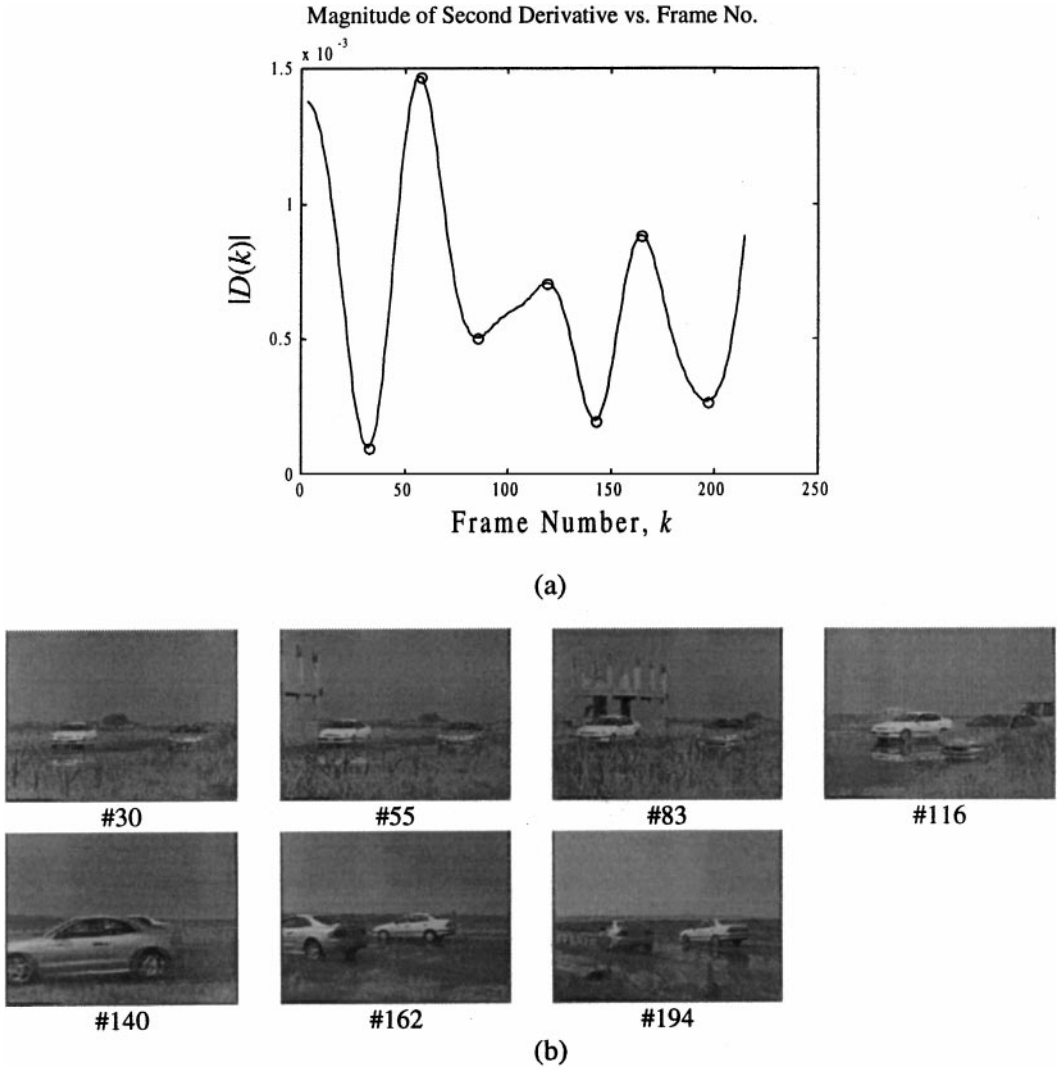

FIG. 15. Temporal variation approach on test drive sequence: (a) magnitude of second windowed derivative, $|D(k)|$, versus the frame number, k, and (b) selected frames.

performance of the proposed scheme. In this case, the extracted frames could be compared to those provided by several appropriately selected humans to indicate which results are closer to human subjectivity.

Figure 16b shows the minimum, over the whole population, value of the correlation measure versus the cycle of the genetic algorithm. As expected, $R(\mathbf{x})$ decreases as the GA cycle increases, until it reaches a minimum at generation 40. Since in the specific experiment half chromosomes are replaced by new ones at each generation, there are cases where all generated offspring have lower fitness than their parents. In these cases the value of the correlation measure remains at the same level, hence the "stepwise" appearance of the curve in the above figure. Note that the step "width" increases with the GA cycle, since it is directly related to the probability of further optimization.

The four selected key frames of the given shot are shown in Figs. 16c, d, e for the logarithmic, stochastic, and genetic algorithms, respectively. Although a very small percentage of frames is retained, it is clear that, in all cases, one can visualize the content of the shot by just examining the four selected frames. Consequently, the selected frames give a meaningful representation of the content of the video shot. Although a comparison of the three algorithms is rather subjective, it can still be argued that key frames selected by the genetic algorithm are more representative of the shot than those of the other two algorithms.

Finally, the same experiments are repeated for all shots in the database, so as to obtain a reliable comparison between different approaches. The temporal variation approach is first applied in order to estimate the number of key frames required, as well as the initial index vectors for the genetic algorithm. The cross-correlation approach follows, with K_s equal to half the number detected by the temporal variation approach. The average, over all shots, correlation measure, $\overline{R}$, obtained by each method, is displayed in Table 2, together with the average computational

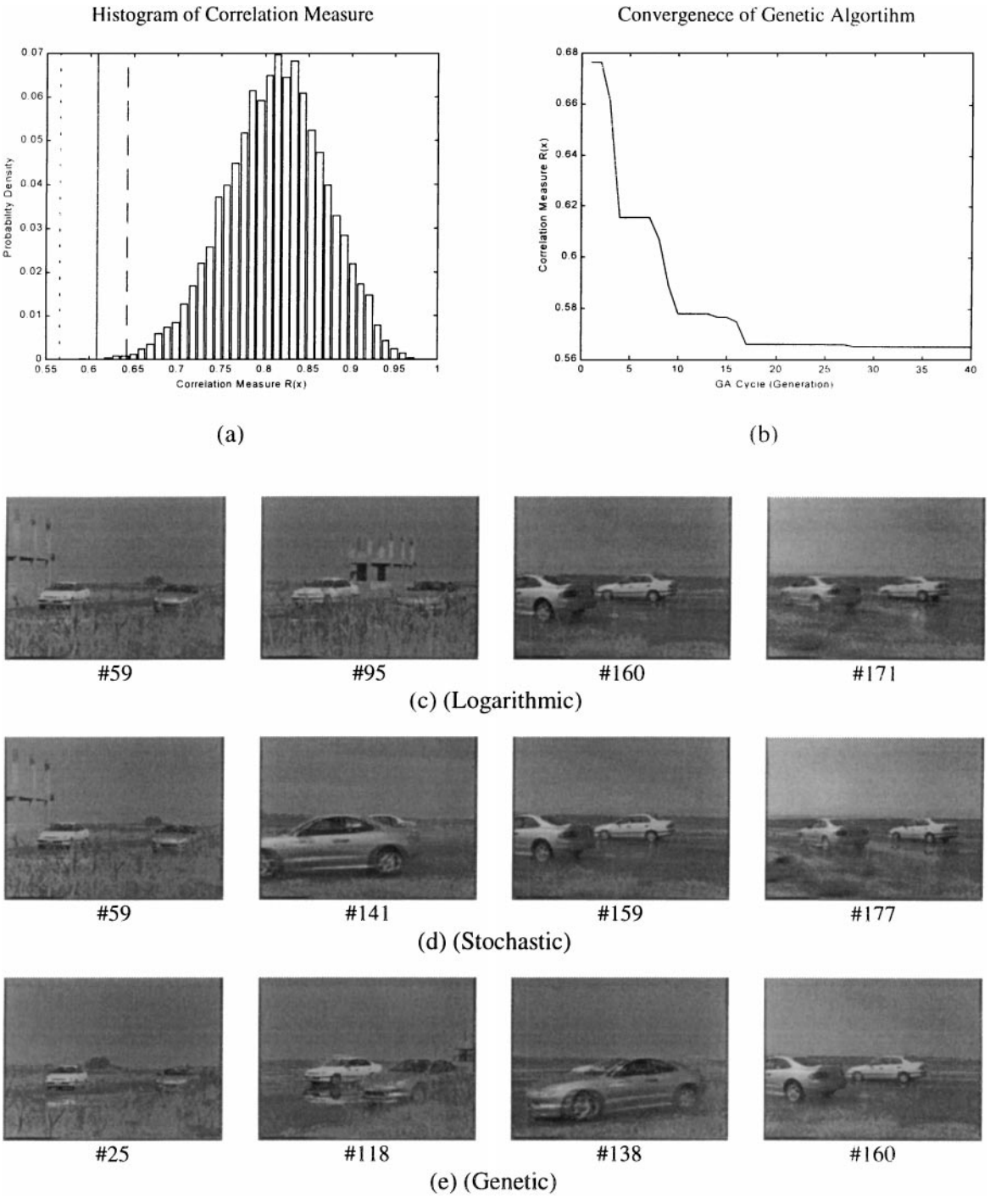

FIG. 16. Cross-correlation approach on test drive sequence: (a) histogram of correlation measure $R(\mathbf{x})$, together with optimal values (dashed line, logarithmic; solid, stochastic; dotted, genetic); (b) minimum value of $R(\mathbf{x})$ versus cycle of genetic algorithm; (c) key frames (logarithmic); (d) key frames (stochastic); and (e) key frames (genetic).

time, $\bar{T}$. As a conclusion, the genetic algorithm outperforms the other two methods in terms of both speed and accuracy of results. Still, however, the temporal variation approach is very useful, mainly as a preprocessing tool for estimation of the shot complexity.

6. CONCLUSIONS

In this paper, an efficient video content representation has been presented for extracting a small but meaningful information of video data. Our study has been concentrated on extracting

TABLE 2

Results of Cross-Correlationl Approach over All Shots of the Database: Average Optimal Correlation Measure, $\bar{R}$, and Average Computational Time, $\bar{T}$, for Logarithmic, Stochastic, and Genetic Algorithms

Method	Average correlation measure, $\bar{R}$	Average computational time, $\bar{T}$(s)
Logarithmic search	0.63	1.92
Stochastic logarithmic search	0.59	12.43
Genetic algorithm	0.44	0.54

several characteristic frames for a given shot by applying a content-based rate-sampling algorithm. To provide a more meaningful description of the shot content, several features are extracted first through a hierarchical color and motion segmentation algorithm. Additional properties such as segment size and location are also included. All the above features are gathered using a fuzzy feature vector formulation, providing more flexibility and simultaneously reducing the influence of segmentation discontinuities. The whole procedure has been oriented to exploit information that exists in MPEG video databases, so as to achieve fast implementation. In this case, many parameters used in the process such as the average block color and motion vectors are already precomputed.

For key frame extraction, temporal variation of the feature trajectory and minimization of a cross-correlation criterion have been proposed. The former technique is very fast and easy to implement; moreover, it provides satisfactory results in case a collection of shot frames is periodically repeated. Instead, the latter one optimally selects the key frames for a given shot according to frame similarity. Since an exhaustive search for the optimal solution is practically unfeasible, a deterministic and a stochastic approach for logarithmic search, as well as a genetic algorithm have been proposed for implementation of the cross-correlation approach. Experimental results have been presented indicating the good performance of the proposed architecture in real life video recordings. The genetic algorithm has been shown to outperform the other two cross-correlation approach implementations in terms of both speed and accuracy of results, while the temporal variation approach has been proved a powerful preprocessing tool for estimation of shot complexity.

The proposed video representation provides a sufficient framework for many multimedia applications. Examples include video content visualization and summarization, efficient management of large video databases, content-based indexing and retrieval, fast video browsing and access to video archives, and finally, the automatic creation of video clip previews (trailers). Further improvement of the proposed techniques can be achieved by applying more robust object segmentation algorithms. In particular, integration of color, motion, as well as depth information in a common segmentation scheme in the proposed representation is currently under investigation. Another objective is the implementation of semantic object segmentation and tracking so that meaningful entities of a video frame can be extracted. Finally, an object graph can be incorporated into the fuzzy classification so that the location and the relationship among different video objects are exploited.

REFERENCES

1. F. Arman, R. Depommier, A. Hsu, and M. Y. Chiu, Content-based browsing of video sequences, *ACM Multimedia*, Aug. 1994, 77–103.
2. Y. Avrithis, A. Doulamis, N. Doulamis, and S. Kollias, An adaptive approach to video indexing and retrieval using fuzzy classification, in *Proc. Int. Workshop on Very Low Bitrate Video Coding (VLBV), Urbana, IL, Oct. 1998*.
3. Y. Avrithis, N. Doulamis, A. Doulamis, and S. Kollias, Efficient content representation in MPEG video databases, in *Proc. of IEEE workshop on Content-Based Access of Image and Video Libraries (CBAIVL)*, pp. 91–94, *Santa Barbara, USA, June 1998*.
4. S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, A fully automated content-based video search engine supporting spatiotemporal queries, *IEEE Trans. Circ. Syst. Video Technol*. **8**, No. 5, 1998.
5. S.-F. Chang, A. Eleftheriadis, and R. McClintock, Next-generation content representation creation, and searching for new-media applications in education, in *Proc. of the IEEE*, Vol. 86, No. 5, pp. 884–904, *May 1998.*
6. J.-Y. Chen, C. Taskiran, E. J. Delp, and C. A. Bouman, ViBE: A new paradigm for video database browsing and search, in *Proc. of IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL)*, pp. 96–100, *Santa Barbara, CA, June 1998*.
7. A. Doulamis, Y. Avrithis, N. Doulamis, and S. Kollias, Indexing and retrieval of the most characteristic frames/scenes, in *Proc. of Workshop on Image Analysis for Multimedia Interactive Systems (WIAMIS)*, pp. 105–110, *Louvain-la-Neuve, Belgium, 1997*.
8. A. Doulamis, Y. Avrithis, N. Doulamis, and S. Kollias, Interactive content-based retrieval in video databases using fuzzy classification and relevance feedback, in *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS), Florence, Italy, June 1999* (to appear).
9. A. Doulamis, N. Doulamis, and S. Kollias, A neural network based scheme for unsupervised video object segmentation, in *Proc. of IEEE Int. Conference on Image Processing (ICIP), Chicago, USA, Oct. 1998*.
10. A. Doulamis, N. Doulamis, G. Konstantoulakis, and G. Stassinopoulos, Traffic characterization and modeling of VBR coded MPEG source, *IFIP: ATM Networks* **3**, 1997, 62–82.
11. N. Doulamis, A. Doulamis, Y. Avrithis, and S. Kollias, Video content representation using optimal extraction of frames and scenes, in *Proc. of IEEE Int. Conference on Image Processing (ICIP), Chicago USA, Oct. 1998.*
12. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: the QBIC system, *IEEE Comput. Mag*. 1995, 23–32.
13. D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison–Wesley, Reading, MA, 1989.
14. A. Hamrapur, A. Gupta, B. Horowitz, C. F. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain, Virage video engine, in *SPIE Proc. Storage and Retrieval for Video and Image Databases V*, pp. 188–197, *San Josc, CA, Feb. 1997.*
15. F. M. Idris and S. Panchanathan, Spatio-Temporal Indexing of Vector Quantized Video Sequences, *IEEE Circuits and Systems for Video Technology*, Oct. 1997, 728–740.
16. *IEEE Workshop on Content Based Access of Image and Video Libraries (CBAIVL), Santa Barbara, June 21, 1998.*
17. M. Irani and P. Anandan, Video indexing based on mosaic representation, *Proceedings of the IEEE*, Vol. 86, No. 5, pp. 805–921, *May 1998.*

18. D. Kalogeras, N. Doulamis, A. Doulamis, and S. Kollias, Very low bit-rate coding of image sequences using adaptive regions of interest, in *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 8, No. 8, pp. 928–934, *Dec. 1998.*
19. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice–Hall, Englewood Cliffs, NJ, 1992.
20. C.-S. Li, R. Mahan, and J. Smith, Multimedia content description in the infopyramid, in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3789–3792, *Seattle, WA, May 1998.*
21. W. Y. Ma and B. S. Manjunath, Netra: A Toolbox for navigating large image databases, in *Proc. of ICIP, Santa Barbara, CA, Oct. 1997.*
22. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York/Berlin, 1994.
23. M. Mills, J. Cohen, and Y. Y. Wong, A magnifier tool for video data, in *Proc. ACM Computer Human Interface (CHI)*, pp. 93–98, *May 1992*.
24. O. J. Morris, M. J. Lee, and A. G. Constantinides, Graph theory for image analysis: An approach based on the shortest spanning tree, in *IEEE Proceedings*, Vol. 133, pp. 146–152, *April 1986.*
25. MPEG Video Group, MPEG-4 Video Verification Model-Version 2.1, in *ISO/IEC/JTCI/SC29/WG11, May 1996.*
26. MPEG-7: Context and Objectives (v.5), in *ISO/IEC/JTC1/SC29/WG11 N1920 MPEG-7, Oct. 1997.*
27. P. J. Mulroy, Video content extraction: Review of current automatic segmentation algorithms, in *Proc. of Workshop on Image Analysis and Multimedia Interactive Systems (WIAMIS), Louvain-la-Neuve, Belgium, June 1997.*
28. J. Nam and A. Tewfik, Progressive resolution motion indexing of video object, in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seattle, USA, May 1998.*
29. N. V. Patel and I. K. Sethi, Video shot detection and characterization for video databases, *Pattern Recognition* **30**, No. 4, 1997, 583–592.
30. A. Pentland, R. W. Picard, and S. Sclaroff, Photobook: Content-based manipulation of image databases, *Int. J. Comput. Vision* **18**, No. 3, 1996, 233–254.
31. Y. Rui, T. S. Huang, and S.-F. Chang, Digital image/video library and MPEG-7: Standardization and research issues, in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3785–3788, *Seattle, WA, May 1998.*
32. Y. Rui, T. S. Huang, and S. Mehrotra, Content-based image retrieval with relevance feedback in MARS, in *Proc. of ICIP, Santa Barbara, CA, Oct. 1997.*
33. B. Shahraray, Scene change detection and content-based sampling of video sequences, in *Proc. of SPIE 2419: Digital Video Compression: Algorithms and Technologies*, pp. 2–13, *Feb. 1995.*
34. T. Sikora, The MPEG-4 video standard verification model, in *IEEE Trans. Circuits Systems Video Technol.* **7**, No. 1, 1997, 19–31.
35. J. R. Smith and S. F. Chang, VisualSEEk: A fully automated content-based image query system, *ACM Multimedia Conf.*, pp. 87–98, *Boston, MA, Nov. 1996.*
36. S. W. Smoliar and H. J. Zhang, Content-Based Video Indexing and Retrieval, *IEEE Multimedia*, Summer 1994, 62–72.
37. V. N. Gudivada and J. V. Raghavan (Guest Eds.), *IEEE Comput. Mag.* **28**, No. 9, 1995. [Special Issue on Content-Based Image Retrieval Systems]
38. R. Jain (Guest Ed.), *Comm. ACM*, Dec. 1997. [Special Issue on Visual Information Management]
39. A. M. Tekalp, *Digital Video Processing*, Prentice-Hall, 1995.
40. N. Vasconcelos and A. Lippman, A spatiotemporal motion model for video summarization, in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 361–366, *Santa Barbara, CA, June 1998.*
41. B. L. Yeo and B. Liu, Rapid scene analysis on compressed videos, *IEEE Trans. Circuits Systems Video Technol.* **5**, 1995, 533–544.
42. M. M. Yeung and B.-L. Yeo, Video visualization for compact presentation and fast browsing of pictorial content, *IEEE Trans. Circuits Systems Video Technol.* **7**, No. 5, 1997, 771–785.