ViBE: A Compressed Video Database Structured for Active Browsing and Search

Cuneyt Taskiran, *Student Member, IEEE*, Jau-Yuen Chen, Alberto Albiol, *Member, IEEE*, Luis Torres, *Senior Member, IEEE*, Charles A. Bouman, *Fellow, IEEE*, and Edward J. Delp, *Fellow, IEEE*

***Abstract*—In this paper, we describe a unique new paradigm for video database management known as *ViBE* (video indexing and browsing environment). *ViBE* is a browseable/searchable paradigm for organizing video data containing a large number of sequences. The system first segments video sequences into shots by using a new feature vector known as the Generalized Trace obtained from the DC-sequence of the compressed data. Each video shot is then represented by a hierarchical structure known as the shot tree. The shots are then classified into pseudo-semantic classes that describe the shot content. Finally, the results are presented to the user in an active browsing environment using a similarity pyramid data structure. The similarity pyramid allows the user to view the video database at various levels of detail. The user can also define semantic classes and reorganize the browsing environment based on relevance feedback. We describe how *ViBE* performs on a database of MPEG sequences.**

***Index Terms*—Face detection, semantic gap, shot boundary detection, shot representation, shot similarity, video browsing, video databases, video search.**

I. Introduction

THE PROLIFERATION of multimedia material, while offering unprecedented opportunities for users to search and profit from available content, also has made it much harder for users to find the material they are looking for in large collections. Depending on the specific information they are seeking, users desire flexible and intuitive methods to search and browse multimedia libraries. However, the cost of manually extracting the metadata to support such functionalities may be unrealistically high. Therefore, over the last decade there has been a great interest in designing and building systems that automatically analyze and index multimedia data and retrieve its relevant parts. Contrasted with traditional text-based approaches to indexing and search, image and video libraries present unique design challenges since visual information generally contains a large amount of information at many conceptual levels which is hard to capture using keywords [1]. Although some frameworks to define systematic methods for multimedia indexing exist [2], currently there is no accepted universal set of features to describe multimedia information. The MPEG-7 standard [3] is aiming to solve this problem by standardizing multimedia descriptors and description schemes. Proposals for systems that use MPEG-7 terminology have started to appear that describe video program description schemes and the devices that will make use of these schemes [4].

The indexing and browsing challenges are most pronounced for digital video libraries because video is a visually information-rich spatio–temporal medium, where the temporal component is essential in interpreting the video. Video data also has high data volumes, even when compressed, which causes a computational burden. A number of systems have been proposed to solve the digital video library management problem. These include the VideoQ system [5], which allows region-based queries with motion information, the Virage video Engine [6], CueVideo from IBM [7] which uses keywords obtained from speech recognition to search through video data, a home movie library management system from Intel [8], Delft University's DANCERS news retrieval system [9] where image and audio data, and speech transcripts are used to divide news programs into report segments and assign topics to each of these and the Fischlar system [10] developed by the Dublin City University for indexing and browsing broadcast TV content. One of the largest efforts in video indexing and access is the Informedia system [11] developed by Carnegie Mellon University which reportedly has more than 2,000 hours of news and documentary programs.

The browsing components of most of these systems generally consist of presenting the user with a series of keyframes extracted from consecutive video shots. The main drawback to this approach to browser interface design is that it presents the user with a localized view of the whole database rather than presenting a an overview of the entire database. For users who do not have a clear idea of the content they are searching but would rather browse through the database this approach may be disorienting and confusing. The hierarchical video database browsing system we propose in this paper takes a different approach to browsing and aims to rapidly give the user a general idea about the content stored in the database.

In this paper, we present an integrated system for managing large databases of video, which we call *ViBE* (video indexing and browsing environment) [12], [13]. The *ViBE* system introduces a variety of novel algorithms and techniques for processing, representing, and managing video while keeping the

Manuscript received September 17, 2001; revised June 27, 2002. The associate editor coordinating the review of this paper and approving it for publication was Prof. Yao Wang.

C. Taskiran, C. A. Bouman, and E. J. Delp are with the School of Electrical and Computer Engineering. Purdue University, West Lafayette, IN 47907-1285 USA (e-mail: taskiran@ecn.purdue.edu; bouman@ecn.purdue.edu; ace@ecn.purdue.edu).

J. Y. Chen is with the Epson Palo Alto Laboratory, Palo Alto, CA 94304 USA (e-mail: jauyuen@erd.epson.com).

A. Albiol is with the Departmento de Communicaciones, Universidad Politècnica de Valencia, Valencia 46022, Spain (e-mail: alabiol@dcom.upv.es).

L. Torres is with the Universidad Politècnica de Catalunya, Barcelona 08034, Spain (e-mail: luis@gps.tsc.upc.es).

Digital Object Identifier 10.1109/TMM.2003.819783

user in the loop. One of the most important objectives of this paper is to describe not only how these techniques function, but also how they integrate together into a single system. We maintain that the *ViBE* system, as presented in this paper, is useful and scalable within a domain-specific video database application that includes indexing, browsing, and search functionalities. Fig. 1 illustrates the major components of the *ViBE* system.

Shot boundary detection is the first step in processing video data. Rather than thresholding a video frame similarity feature, as was usually done in early work in this area, we pose the shot boundary detection problem as a standard classification problem, in which a high dimensional feature vector is first extracted from each frame in the video sequence. These vectors are then analyzed using a suitable classifier. In our technique, the feature vectors, which we call the *generalized trace* (GT) [14], are extracted from compressed video. We then use a classifier based on a regression tree [15] which is used to estimate the conditional probability of a shot boundary for each frame. The regression tree is essential because it automatically selects the relevant information from the GT and generates probabilities which can be directly used for shot boundary detection.

In order to capture the salient aspects of complex shots, we introduce the idea of a hierarchical representation we call the *shot tree* which is a binary tree containing a set of representative frames from the shot. The shot tree is formed by clustering the frames in a shot, hence it hierarchically organizes frames into similar groups. The tree structure also allows important operations, such as shot similarity comparisons, to be obtained efficiently using tree matching algorithms.

Ideally, one would like to query a video database using high-level semantic descriptions, such as "young girl running" or "park scene." Unfortunately, automatic labeling of such categories is currently difficult. In order to overcome this difficulty, we adopt the use of pseudo-semantic classes which are broadly defined semantic categories to bridge the gap between low-level features and semantic labels. Examples of such labels include "face," indicating that the shot contains a face, and "indoor/outdoor," among others. In *ViBE*, pseudo-semantic labels are expressed as a vector with elements that take on values in $[0, 1]$ depending on the confidence of the label. The vector labels can then be incorporated into the search, browsing, and relevance feedback operations.

The active browsing environment provides a user interface that integrates together the results of shot boundary detection, shot representation, and pseudo-semantic labeling. Our browsing environment is based on a similarity pyramid data structure [16]. The similarity pyramid uses a tree structure to organize the objects in the database for efficient access. We utilize relevance feedback in the browsing environment, so that users can dynamically modify the database's organization to suit the desired task. Finally, we demonstrate the performance of *ViBE* using a database of MPEG sequences.

The paper is organized as follows. In Section II, we review some previous approaches to shot boundary detection and describe our shot segmentation algorithm using the generalized trace feature vector and a regression tree classifier. Section III describes our approach to representation of detected shots and illustrates how our proposed shot tree structure may be used to compute shot similarity. In Section IV, pseudo-semantic shot features are introduced and we describe in detail how we extract the "face" shot feature. In Section V, we describe the active hierarchical browsing environment and examine efficient methods to perform browsing and search operations in this environment. Finally, experimental results of our video database browsing system are presented in Section VI.

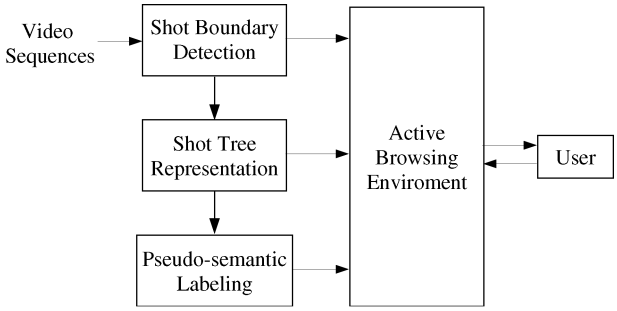

Fig. 1. Major components of the *ViBE* system.

II. Shot Boundary Detection

A *shot*, which is the basic component of a video sequence, may be defined as a group of frames that has continuity in some general conceptual or visual sense. Often, a shot is composed of frames which depict the same physical scene, signify a single camera operation, or contain a distinct event or action. Most systems for indexing, characterization, and understanding of video rely on the identification of individual shots; hence, usually the first task in processing video data is segmenting shots by detecting shot boundaries, thereby breaking the video sequence into distinct "semantic chunks." In order to exploit the hierarchical nature of video the detected shots may then be clustered to obtain scenes, which form the next semantic level for video.

The number of possible types of shot boundaries is large. However, in practice almost all shot boundaries fall into one of the following three categories: cuts, fades, and dissolves. Among these, cuts, which are abrupt changes from one shot to another between two frames, form the overwhelming majority; therefore, in this paper we shall concentrate on the detection of cuts. The generalization of our technique to other types of shot boundaries may be found in [13].

A. Previous Work

Since shot boundary detection is a fundamental component of video processing, considerable work has been done in the last decade in this topic. The general technique is to derive one or more low-level features from video frames, derive a dissimilarity value between frames using these features, and flag a shot transition whenever this value shows some nontypical behavior. The following is a brief list of some of the previous techniques that have been used in the literature. More references may be found in the survey and comparison papers [17]–[21].

Pixel-wise frame differences [22], [23], where the differences in intensity or color values of corresponding pixels in two successive frames are used, is one of the oldest methods.

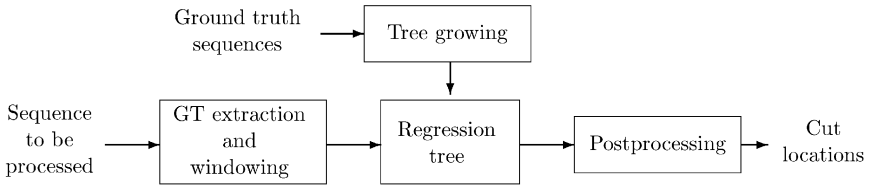

Fig. 2. Schematic diagram for cut detection.

Block-based versions of this technique [24] have also been proposed to make the comparison more robust to camera and object motion. Another robust method is to compare intensity or color histograms of successive frames to detect cuts [25]. Various distance measures between histograms have been investigated [26] but the L_1 norm seems to be the most useful. Similarity features derived from histograms for frames within a window may be clustered to detect outlier values which correspond to cuts [27].

Processing in compressed domain offers many advantages and many methods have been proposed to detect cuts for MPEG compressed sequences. These methods generally try to exploit the extra information available in the MPEG stream such as the DCT coefficients [28], [29] and macroblock types [30], [31].

A different approach to shot boundary detection is to pursue a model-driven approach where algorithms are based on mathematical models of video and shot boundaries. Methods based on modeling interframe pixel differences [32], shot lengths [33], and interesting events in soundtracks [34] have been proposed. Hidden Markov models, built using information from both images and audio, have also been used successfully to detect shots [35], [36].

Detecting and identifying shot transitions by processing a single frame dissimilarity feature has a number of problems: First, it is difficult to determine a single feature that could be used to accurately detect shot boundaries in a wide variety of situations. Second, for techniques where a global threshold is used, there is the problem of determining the optimal value of the threshold to be used, since this value may vary considerably from sequence to sequence.

B. Generalized Trace

In *ViBE*, we perform shot boundary detection by first extracting from the given compressed video sequence the "DC sequence," which is formed from the DC coefficients of 8×8 DCT blocks of the MPEG coded video frames. While the DCT DC coefficients are directly available for I frames, they must be estimated for P and B frames. We have used the method described in [37] for estimating these DC coefficients.

The next step is the extraction of a set of features from each DC frame. These features form a feature vector that we call the *generalized trace* (GT) [14]. The GT is then used in a binary regression tree to determine the probability estimate that a given frame is the start of a new shot These probabilities can then be used to detect cut locations.

The GT for frame i of a video sequence is denoted by $\mathbf{g}_i$ and its jth component by $g_{i,j}$. For the experiments described in this paper, the GT consists of the following features:

$g_{i,1-3}$: Histogram intersection of frames i and $i+1$ for the Y, U, and V color components

$$g_{i,1} = \frac{1}{2M_1}\sum_{k=1}^{64}|H_i^Y(k) - H_{i+1}^Y(k)|$$
$$g_{i,(2,3)} = \frac{1}{2M_2}\sum_{k=1}^{32}|H_i^{(U,V)}(k) - H_{i+1}^{(U,V)}(k)| \quad (1)$$

where M_1 and M_2 are the number of luminance and chrominance blocks, respectively. In our case these have the values $M_1 = 44 \times 30 = 1320$ and $M_2 = 330$.

$g_{i,4-6}$: Standard deviation of the Y, U, and V color components for frame i

$$g_{i,4} = \frac{1}{M_1 - 1}\sum_k\sum_l (f_i^Y(k,l) - \mu_i^Y)^2$$
$$g_{i,(5,6)} = \frac{1}{M_2 - 1}\sum_k\sum_l (f_i^{(U,V)}(k,l) - \mu_i^{(U,V)})^2 \quad (2)$$

where μ_i^Y, μ_i^U, and μ_i^V are the mean values for the luminance and chrominance blocks over a frame, respectively.

$g_{i,7-9}$: Number of intracoded, forward predicted, and backward predicted macroblocks in frame i, respectively.

$g_{i,10-12}$: Boolean features which identify the frame type $\{I, P \text{ or } B\}$ for frame i.

The last three features are required because the information contained in the GT must be interpreted differently for different types of frames in the MPEG sequence. For example, P frames contain no backward predicted macroblocks so $g_{i,9}$ will identically be zero for these types of frames. Similarly, the number of intracoded macroblocks per frame, $g_{i,7}$, should be interpreted differently for I, P, and B frames.

C. Binary Regression Tree

We use the GT of a video sequence and a binary regression tree [15] to estimate the conditional probability that a given frame from the sequence belongs to a shot transition. We chose to use the specific regression tree algorithm described in this section, rather than another decision tree classification algorithm, because we have found this algorithm to be robust and efficient in other applications [38], [39]. Furthermore, the posterior probabilities generated by the tree may be used for other purposes. Regression trees are used when learning a relation between a set of features and a continuous-valued output variable, i.e., to implement a function of the form $y_i = f(\mathbf{g}_i)$. An improvement over the CART [15] method of building regression trees was proposed in [40] where the data set is divided into two roughly equal sets. One set is used to grow a large tree that overfits the data. Then the other data set is used to prune the tree back

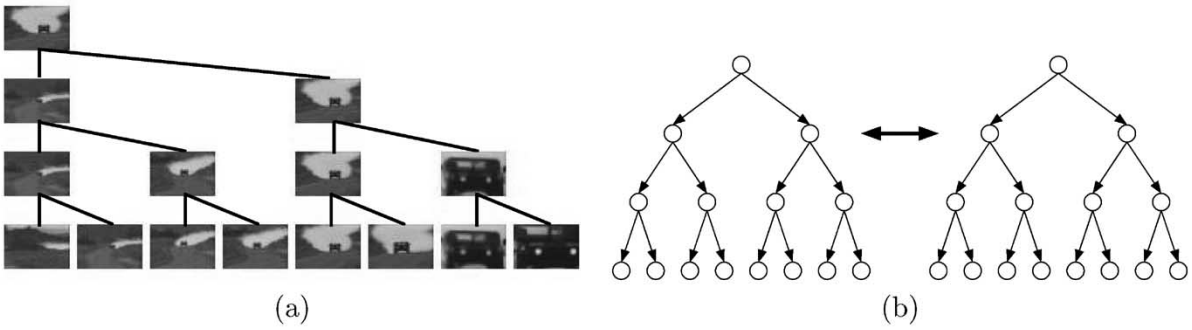

Fig. 3. (a) Each shot is represented by a hierarchy of frames which are organized with agglomerative clustering; (b) Shot dissimilarity is then determined by computing the distance between two trees.

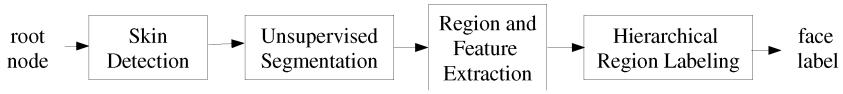

Fig. 4. Block diagram illustrating the processing steps used to obtain the pseudo-semantic "face" label for the root node of each shot tree.

to maximize some desirability criteria. This procedure is then iterated, successively interchanging the roles of the growing and pruning data sets until convergence.

The regression tree used in *ViBE* is a variation of the above technique. The difference is that the training and pruning step is used only once since we have found that the tree obtained after one iteration is adequate in classification accuracy. The training process uses two sequences of nearly equal size with known shot boundary locations, which we refer to as *ground truth sequences*. In the ground truth sequences, frame i is labeled as 1 if there is a cut between frames i and $i+1$ and 0, otherwise. One ground truth sequence is used to build a large tree which overfits the data. The tree obtained is then pruned in a bottom-up fashion using the second ground truth sequence where we remove nodes whose deletion decreases the classification error.

The main advantage of the GT/regression tree method is that it lets us collectively use nominal features together with real-valued features. Note that the last three features of the GT feature vector jointly specify the type of the frame being checked for a shot boundary. Therefore, these features are nominal, that is, there is no natural distance measure defined between them. Authors who have applied MPEG macroblock type information to the shot boundary task [30], [31] have used *ad hoc* rule-based systems that deal with different kinds of frame types separately. On the other hand, authors who have used color histogram and related techniques do not usually incorporate the additional information contained in the MPEG macroblock types. Our GT/regression tree approach enables us to methodically combine real-valued and nominal valued features in our feature vector and collectively use the information provided by them.

The GT/regression tree method also offers some other advantages: the output of the regression tree is normalized in the range $[0,1]$ and approximates the probability that the frame under question belongs to a shot transition, which allows consistent and meaningful thresholding for different program types. Other thresholding methods may need different types of theresholds for different program types. Furthermore, the method is highly extensible. New features can easily be incorporated into the existing system.

1) Detection of Cuts: A schematic diagram of the steps in cut detection is shown in Fig. 2. To train the regression tree, we use known cut locations in the ground truth sequences and ignore gradual transitions. After the tree has been trained using two ground truth sequences, it is used to process the GT from the sequence whose cuts are to be detected. In order to make the detection more robust, instead of using the feature vectors derived from each frame in the regression tree classifier, we combine the feature vectors from a window of frames centered at the frame we want to classify, and use this agglomerate feature vector as the input to the regression tree classifier. Therefore, to detect if a cut has occurred between frames i and $i+1$ we place a window of length $2W+1$ frames centered around frame i and the GT vectors for all the frames in this window are concatenated into one large feature vector. These agglomerate vectors are then used by the regression tree which provides a piecewise linear approximation to the conditional mean, i.e.,

$$y_i \approx E[\alpha_i|\mathbf{g}_{i-W},\cdots\mathbf{g}_i,\cdots\mathbf{g}_{i+W}] \tag{3}$$

where α_i is the cut indicator function defined as

$$\alpha_i = \begin{cases} 1, & \text{if a cut occurs between frames } i \text{ and } i+1 \\ 0, & \text{otherwise} \end{cases}$$

and y_i is the output of the regression tree for frame i. The output y_i can be interpreted to be the probability of a cut occurring between frames i and $i+1$ [15].

Candidate cut locations are then determined by thresholding the output of the regression tree; if $y_i \geq \tau$, we decide that there is a cut between frames i and $i+1$. This approach is more robust than thresholding the value of a frame similarity feature, because the classifier chooses the best features from the GT and the threshold is not dependent on the specific video sequence.

The detected candidate cut locations are then post-processed to remove cuts that are too close together. In the experiments described in Section VI-B, if two candidate cut locations are

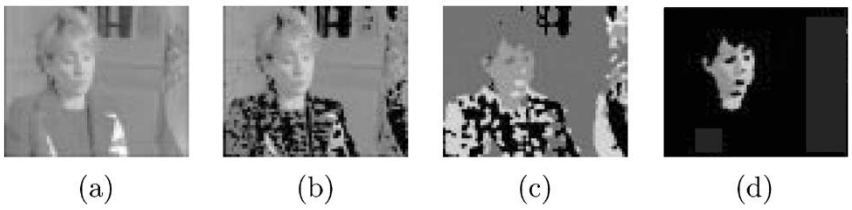

Fig. 5. (a) Original image. (b) Skin-like detected pixels. (c) Homogeneous skin-like regions. (d) Detected face.

closer than ten frames, the candidate cut with a smaller value of y_i is deleted.

III. Shot Representation

The shots that are obtained at the completion of the temporal segmentation step are processed to obtain a representation for each shot. This representation is both used in computing similarity between shots for purposes of search and database reorganization, and in providing information for users to browse the database rapidly and efficiently. The representation used in *ViBE* includes a tree of representative frames extracted from the shot, the beginning and end frame numbers, the unique sequence number of the sequence the shot belongs to, and a motion histogram to quantify the motion content of the shot.

One of the most common ways to represent and visualize video shots is to extract one or more frames from each shot and use these as an abstract representation of the shot. These frames are known as *representative frames* or *keyframes* which may be used in a table of contents representation for video [4]. For shots with slow changing content, one keyframe per shot is adequate. When shots contain semantic changes or fast motion, however, more than one keyframe per shot is required.

The simplest method for keyframe selection is to pick from every shot one or more frames in prescribed locations, e.g., the first frame of the shot. This method will not generate a faithful representation of the shot for most video content. The most common approach is to cluster frames from a shot based on low level properties such as color and texture, and choosing the frames closest to cluster centroids as keyframes [27], [41]–[43]. The features that have been used in clustering include two-dimensional (2-D) color histograms of frames [41], [42], and color histograms averaged over the whole shot [27]. In *ViBE*, we obtain the keyframes for a shot using a tree representation as described in Section IV.

A. Shot Tree Representation

This section describes a novel way to represent a shot based on a tree structure, which we call the *shot tree*, formed by clustering the frames in a shot. This tree structure allows important operations, such as shot similarity comparisons, to be obtained efficiently using tree matching algorithms. A typical shot tree is shown in Fig. 3(a). The frames in the shot are organized by the tree in such a way that the root node is the most "representative" frame in the shot. As one progresses down the tree, frames are organized into similarity groups.

The number of levels considered for the tree depends on the application. For example, if one wants to categorize the shot by one frame, the root node is used. If more detail is desired, additional levels may be used. For browsing, it is generally sufficient to use the root node as the keyframe; for shot similarity measurement and classification in the browsing environment, two or three levels of the tree can be used, as shown in Fig. 3(a).

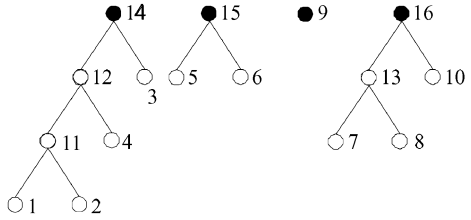

Fig. 6. Binary tree structure resulting from skin-like region merging.

The shot tree may be obtained by employing either top-down or bottom-up (agglomerative) clustering of the shot's frames. While top-down clustering is fast, it tends to produce poor clusters at lower levels of the tree. This is due to the fact that once an image is placed in an undesirable cluster, it is constrained to remain in that branch of the tree. In this paper, the shot tree representations are obtained through agglomerative clustering, which we describe next.

To build the shot tree for a given shot, we first extract a high dimensional feature vector containing color, texture and edge histogram features from each DC frame in the shot. We use exactly the same features described in detail in [44]. In particular we use color histograms, texture features formed by histogramming the magnitude of the local image gradient of each color component, and edge features formed by thresholding the edge gradient and then computing the angle of the edge for all points that exceed a certain threshold. We denote the feature vector for frame j by f_j. Let c_i be the set of all frames in cluster i, and let n_i be the number of frames in c_i. Each element of the proximity matrix $[d_{ij}]$ is then defined as the distance between clusters c_i and c_j. Let the given shot contain frames $f_j, j = 1 \cdots N$. Initially, the element d_{ij} is set equal to the L_1 distance between feature vectors f_i and f_j, that is, each frame is placed in a separate cluster. Then, each iteration of agglomerative clustering combines the two clusters, c_i and c_j, with the minimum distance. The new cluster formed by joining c_i and c_j is denoted by c_k, and the distances from c_k to each of the remaining clusters are updated. In *ViBE*, we use Ward's clustering algorithm, also known as the minimum variance method [45], where the distances between the newly formed cluster, k, and all the other clusters, $h \neq k$, are updated according to the rule

$$d_{hk} = \frac{n_i + n_h}{n_i + n_j + n_h} d_{hi} + \frac{n_j + n_h}{n_i + n_j + n_h} d_{hj} - \frac{n_h}{n_i + n_j + n_h} d_{ij}. \quad (4)$$

This pairwise clustering generates a binary tree structure which is used as the representation for the shot. Fig. 3(a) illustrates such a representation for a shot from an action sequence. In this example, the shot contains significant changes which can not be captured by a single keyframe; hence the tree structure can better represent the diverse aspects of the shot. The tree structure hierarchically organizes frames into similarity groups, therefore allowing automatic detection of subgroups inside a shot.

B. Motion Histogram and Pseudo-Semantic Representations

In order to quantify the motion content of a shot, we use the motion histogram derived from motion vectors of macroblocks of the frames. For the kth P or B frame in the shot, we first compute, the motion feature

$$m_k = (\#\text{forward MB}) + (\#\text{backward MB}) + 2(\#\text{forward–backward MB}). \quad (5)$$

Then we obtain the histogram $\mathbf{h}_{ij}$ of the values m_k for all the frames k in the shot s_{ij}.

A very important part of the shot representation is based on the shot pseudo-semantic labels we will discuss in Section IV. For each shot, we define a label vector, $\mathbf{p}_{ij}$, where each element of the vector takes on continuous values in the range $[0, 1]$, indicating the confidence level of assigning the corresponding pseudo-semantic label to the shot.

C. Shot Similarity

Determining shot similarity is important for searching and reorganizing the video database. Methods have been proposed that use a version of histogram intersection accumulated for the whole shot [46], and color, texture, and motion features derived from keyframes [47]. In order to define a shot distance measure that is both robust and comparable to human notions of similarity as much as possible we use a weighted sum of distances which depend on the shot tree, the temporal information, the motion information, and the pseudo-semantic shot labels.

Let S_i be the unique sequence number in the database and s_{ij} be jth shot in sequence S_i. Then the distance between two shots is defined as

$$\begin{aligned} D(s_{ij}, s_{kl}) = & D_{ST}(s_{ij}, s_{kl}) \\ & + D_T(s_{ij}, s_{kl}) \\ & + D_M(s_{ij}, s_{kl}) \\ & + D_{PS}(s_{ij}, s_{kl}) \end{aligned} \quad (6)$$

where D_{ST}, D_T, D_M, and D_{PS} are the shot tree, temporal, motion, and pseudo-semantic distance components which are defined as follows.

The *shot tree distance*, $D_{ST}(s_{ij}, s_{kl})$, is measured by obtaining the distance between the two corresponding shot trees as illustrated in Fig. 3(b). Since the shot trees are obtained by clustering all the frames from a given shot, in general they will have many levels. For the purposes of shot similarity calculations, we only use the first three levels of each shot tree This means that each shot is represented by a tree with seven nodes, as illustrated in Fig. 3(b). Each node, t, of the shot tree is associated with a cluster of frames from the shot which form the children of t. We also associate each node with a feature vector, $\mathbf{z}_t$, which is the average of the feature vectors of the frames in the cluster associated with t. The distance between two tree nodes is taken to be the L_1 distance between the feature vectors associated with them. The distance between two shot trees is then given by the weighted sum of the distances between nodes using the best mapping between the shot trees. The best mapping may be found by simply checking the eight distinct mappings between the two trees.

Similar to [48], we define the *temporal distance* between shots s_{ij} and s_{kl} as shown in (7) at the bottom of the page, where b_{ij} and e_{ij} are the beginning and end frame of shot numbers for s_{ij}. Here we assume that if two shots are farther apart than T_f frames or T_s shots, we will not consider them similar in a temporal sense. We have used the values $T_f = 3000$ and $T_s = 30$. The constants K_{Seq} and K_{Shot} can be used to control the relative importance of shot and sequence matching in the overall distance function.

The *motion distance* between shots s_{ij} and s_{kl} is defined to be the L_1 norm of the difference between their motion histograms

$$D_M(s_{ij}, s_{kl}) = K_{\text{Motion}} \|\mathbf{h}_{ij} - \mathbf{h}_{kl}\|_1. \quad (8)$$

The constant K_{Motion} controls the weighting of this component. Similarly, the *pseudo-semantic distance* is defined as the L_1 norm of the difference of the shot pseudo-semantic feature vectors as

$$D_S(s_{ij}, s_{kl}) = K_{\text{Semantic}} \|\mathbf{p}_{ij} - \mathbf{p}_{kl}\|_1. \quad (9)$$

We shall describe how these distance measures are used for browsing in Section V.

IV. Pseudo-Semantic Labeling of Shots

Users often desire to search a multimedia library by issuing queries in terms of keywords describing events, objects, and locations such as "young girl running," "blue dress," and "park scene." Ideally, such content labels might provide the most useful descriptions for indexing and searching video libraries. Currently, however, automatic extraction of such high-level semantic features is not feasible. On the other hand, if the search is based on indexes that are semantically too primitive, such as low-level motion and color features, not only the user may be frustrated by results that seem to be totally irrelevant to the query, but the searching and browsing process will not be intuitive [49]. This is the *semantic gap problem* which is central

$$D_T(s_{ij}, s_{kl}) = \begin{cases} K_{\text{Seq}} + K_{\text{Shot}}, & \text{if } i \neq k \\ K_{\text{Shot}} \left(\min\left(\frac{1}{2}, \frac{\min(|b_{ij} - e_{kl}|, |b_{kl} - e_{ij}|)}{2T_f}\right) + \min\left(\frac{1}{2}, \frac{|j-l|}{2T_s}\right)\right), & \text{if } i = k \end{cases} \quad (7)$$

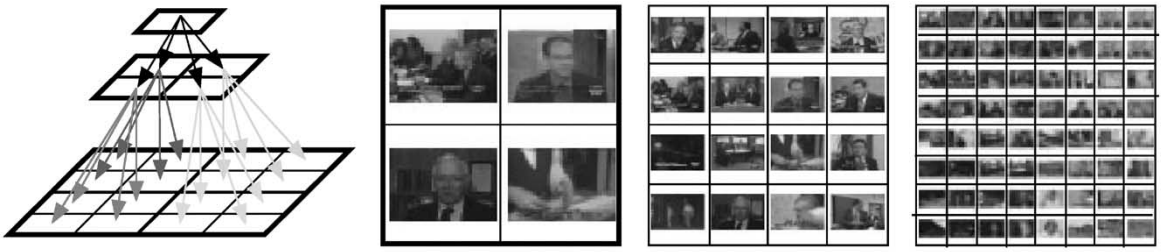

Fig. 7. An example of a similarity pyramid and its embedded quadtree.

in multimedia retrieval. It is caused by the discrepancy between the user's conceptual model of the data and the capabilities of current state of the art in data analysis [1].

Various approaches have been proposed to process low-level features to obtain higher level labels, hence bridging the semantic gap to some degree. In [50] low-level features such as color, texture, and motion blobs are extracted from video and are classified using a neural network which are then processed using a domain-specific inference process. Binary random labels, which the authors call "multijects," are introduced in [51]. These labels are derived using Gaussian mixture models and hidden Markov models which process features extracted from dominant regions in frames. Bayesian networks are used to model the interactions between different multijects.

In *ViBE*, we use a number of shot labels, which we call *pseudo-semantic labels*, to solve the semantic gap problem in searching and browsing. Since pseudo-semantic labels are generally associated with some level of uncertainty, each label is assumed to be a continuous value in $[0, 1]$ where 1 indicates that the shot has the associated property with high confidence, and 0 indicates that it does not. We are investigating several pseudo-semantic labels including labels such as "face," "indoor/outdoor," and "high action." In this paper we will only describe our work on the "face" label. Representative methods which attempt to derive a "indoor/outdoor" label from images include [52] where shots are classified as interior or exterior based on the variation of luminance caused by natural and artificial lighting, [53] where color, texture, and DCT coefficient features are utilized to perform a similar task, and [54] where a Bayesian approach was taken using edge and color features where the class-conditional distributions of features are estimated using vector quantization.

The task of pseudo-semantic labeling is greatly reduced by the shot tree representation described in Section III. For the work presented in this paper, we only label the frame corresponding to the root node for each shot tree. More generally, children of the root node could also be labeled and a voting procedure could be used to determine the shot label. Since the resolution of the DC frames is not adequate to accurately derive most pseudo-semantic labels, the root nodes of the shot trees are extracted from the MPEG stream and are decompressed. Note however, that these this decompression is performed for only a very small percentage of all frames in a video sequence The goal here is to detect whether a frame in the sequence contains a face. Derivation of other labels follows a similar process.

Different approaches have been developed in recent years for the face detection problem. Face detection can be performed either in the compressed or spatial domain. Some of the most representative methods in the spatial domain include shape-feature approaches [55], neural networks [56], and template matching [57]. These methods have tended to focus on still gray scale images. While they report good performance, they are often computationally expensive. This is especially true of template matching and neural network approaches, since they require processing for each possible position and scaling of the image. Also, the results may be excessively sensitive to the rotation and pose of the face. Recently, a number of authors have used color and shape as important cues for face detection. In [58], [59] both color and spatial information are exploited to detect segmented regions corresponding to faces.

Compressed domain face detection techniques have the main advantage of a lower computational load when dealing with compressed video sequences, since it is not necessary to fully decode the frames to perform the detection. This is usually accomplished at the expense of lower performance. One of the most representative compressed domain techniques is [60], where the skin color and the energy distribution of the DCT coefficients were proposed as the basis to detect a face. More recently, an improvement to the previous technique was proposed in [61], where a spatial domain technique [62] was adapted to work directly in DCT domain. Sections V and VI present the details of our approach for determining the "face" label for a frame.

A. Derivation of the Face Label

Our method is designed to be robust for variations that can occur in face illumination, shape, color, pose, and orientation. To achieve this goal, our method integrates a priori information regarding face color, shape, position, and texture to identify the regions which are most likely to contain a face. It does this in a computationally efficient way which avoids explicit pattern matching.

Fig. 4 illustrates the processing steps that we use to obtain the face label for a given frame. Under normal illumination conditions skin colors occupy a small portion of the YC_BC_R color space [60], [63]. The first step is to use this property to segment regions of the image which potentially correspond to face regions. Once the pixels of interest are identified, unsupervised segmentation is used to separate these pixels into smaller regions which are homogeneous in color. This is important because the skin detection will produce nonhomogeneous regions often containing more than a single object, e.g., skin colored materials may be erroneously included in the skin detection.

Fig. 8. Active browsing environment. The top level of the similarity pyramid is shown on the left, and the set of relevant shots (relevance set) is shown on the right. Users can incrementally add or remove shots from the relevance set as they browse through the database.

These undesired regions are separated from true faces using the subsequent unsupervised segmentation step. The next step extracts regions using connected components analysis. For each connected component, a set of features is extracted which characterizes the region in terms of its color and shape. The final step is to label face regions. Face areas will often be divided up into two or more regions in the unsupervised segmentation process. Therefore, to improve detection accuracy, we use a pairwise steepest descent method in which all pairs of merged regions are considered to find the merged region which best fits the face hypothesis. This pairwise merging process is recursively performed and results in a single composite region.

B. Skin Detection and Unsupervised Segmentation

Let $\mathbf{x}$ be the chrominance vector of a pixel in the C_BC_R space. The decision rule to classify a pixel into the *skin class*, H_0, or the *nonskin class*, H_1 is given by

$$\lambda(\mathbf{x}) = \frac{p(\mathbf{x}|H_0)}{p(\mathbf{x}|H_1)} \underset{H_0}{\overset{H_1}{\gtrless}} \lambda \tag{10}$$

where $p(\mathbf{x}|H_0)$ and $p(\mathbf{x}|H_1)$ are the conditional probability density functions given the classes H_0 and H_1, respectively, and λ is a suitable threshold chosen to satisfy the constraint of a fixed false alarm using the Neyman–Pearson test. We estimate $p(\mathbf{x}|H_0)$ and $p(\mathbf{x}|H_1)$ using the normalized histograms obtained from more than 200 images which were manually segmented into *skin* and *nonskin* classes. In order to achieve a 90% detection rate, a value of $\lambda = 2.8$ was chosen yielding a 20.4% false alarm rate. Fig. 5(b) shows an example where the pixels classified as nonskin are set to black. It can be seen that in addition to face pixels, many objects in the background with skin-like colors are also selected.

Once the pixels of interest are identified, unsupervised segmentation is used to separate these pixels into smaller regions which are homogeneous in color. The objective of this step is to separate spurious regions from the true face areas. We use a novel approach for the unsupervised segmentation stage using the watershed algorithm [64] to cluster the skin detected pixels in the color space. Fig. 5(c) shows the results of the unsupervised segmentation using the regions found in Fig. 5(b). The algorithm is able to successfully separate the face region from the background.

C. Region Extraction and Hierarchical Region Labeling

The next step is to extract connected regions and feature vectors that characterize those regions. Let Ω_i be the i^{th} region of pixels, and let $\mathbf{x_r}$ be a color pixel in YUV coordinates at position $\mathbf{r} \in \Omega_i$. Here $\mathbf{r} \in \Omega_i$ is assumed to be a 2-D column vector $\mathbf{r} = [r_1, r_2]^t$ where r_1 and r_2 index the row and column of the pixel. For each region, we extract a state vector containing color and shape features, defined by $\mathbf{u}_i = [N_i, \bar{\mathbf{x}}_i, \boldsymbol{\gamma}_i, \boldsymbol{\mu}_i, \mathbf{R}_i, \max_i, \min_i]^t$, where N_i is the number of pixels in the region, $\max_i$ and $\min_i$ are the corner coordinates of the bounding box of the region, and the other features are defined by

$$\begin{aligned}
\bar{\mathbf{x}}_i &= \frac{1}{N_i} \sum_{\mathbf{r}\in\Omega_i} \mathbf{x}_\mathbf{r}, \\
\boldsymbol{\gamma}_i &= \mathrm{Diag}\left\{ \frac{1}{N_i} \sum_{\mathbf{r}\in\Omega_i} (\mathbf{x}_\mathbf{r} - \bar{\mathbf{x}}_i)(\mathbf{x}_\mathbf{r} - \bar{\mathbf{x}}_i)^t \right\} \\
\boldsymbol{\mu}_i &= \frac{1}{N_i} \sum_{\mathbf{r}\in\Omega_i} \mathbf{r}, \\
\mathbf{R}_i &= \frac{1}{N_i} \sum_{\mathbf{r}\in\Omega_i} (\mathbf{r} - \boldsymbol{\mu}_i)(\mathbf{r} - \boldsymbol{\mu}_i)^t
\end{aligned} \tag{11}$$

where $\boldsymbol{\gamma}_i$ is a vector containing the variance for each of the Y, U, and V color components. An important property of the state vector is that it can be easily updated recursively for merged regions.

From the state vectors $\mathbf{u}_i$, we obtain a feature vector $\mathbf{v}_i$ which contains the specific features that we use for labeling regions. The elements of $\mathbf{v}_i$ are $\bar{\mathbf{x}}_i$, $\boldsymbol{\gamma}_i$, $\boldsymbol{\mu}_i$, the area of the bounding box derived from $\max_i$ and $\min_i$ and normalized by N_i, and three more features obtained from the eigenvalues and eigenvectors of $\mathbf{R}_i$. Let $\lambda_1 > \lambda_2$ be the two eigenvalues of $\mathbf{R}_i$ with corresponding eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. Then the three remaining features are chosen to be $\sqrt{\lambda_1\lambda_2}/N_i$, the orientation of $\mathbf{e}_1$, and λ_2/λ_1.

A multivariate Gaussian distribution is used to model the behavior of $\mathbf{v}_i$ for face areas. Let $\bar{\mathbf{v}}$ and $\boldsymbol{\Sigma}_v$ be the mean and covariance of the vectors $\mathbf{v}_i$ for regions corresponding to face areas. Then the distance between $\mathbf{v}_i$ and the mean behavior for a face region is given by

$$C_i = (\mathbf{v}_i - \bar{\mathbf{v}})^t \boldsymbol{\Sigma}_v^{-1} (\mathbf{v}_i - \bar{\mathbf{v}}). \tag{12}$$

The smaller the value of C_i the greater the likelihood that the region is a face area. In our experiments, $\bar{\mathbf{v}}$ and $\boldsymbol{\Sigma}_v$ are extracted as the sample statistics from a set of 100 manually segmented frames, each containing at least one face.

As discussed earlier, the unsupervised segmentation is likely to partition face areas into more than one connected region. Therefore, we must merge regions to form a new region which best matches the behavior expected for face areas. We do this by searching each pairwise merging of regions to find a new region which minimizes the distance given in (12). We first form a distance matrix $\mathbf{C}$ defined by

$$C_{ij} = (\mathbf{v}_{i,j} - \bar{\mathbf{v}})^t \boldsymbol{\Sigma}_v^{-1} (\mathbf{v}_{i,j} - \bar{\mathbf{v}}) \tag{13}$$

where $\mathbf{v}_{i,j}$ is the feature vector computed by merging regions Ω_i and Ω_j. We then search for the minimum entry of the matrix, $C_{i^*j^*} = \min_{(i,j)\in P} C_{ij}$, where the set P is defined by

$$P = \{(i,j) : C_{ij} < C_{ii} \text{ and } C_{ij} < C_{jj}\}$$

and merge the two regions Ω_{i^*} and Ω_{j^*}, updating the entries of $\mathbf{C}$. The restriction to the set P insures that the minimum occurs between a pair of distinct regions which, when merged, will match the face hypothesis better then any existing individual region. This process of merging is repeated until the set P is empty. Fig. 6 illustrates this recursive merging process. Each node represents a region of the image with the internal nodes representing regions that result from merging. The merging process terminates in a set of nodes, in this example nodes 9, 14, 15, and 16.

Once the above procedure terminates, any of the remaining nodes with less then 600 pixels are removed. The face label for the frame is then computed using the region that best fits the face hypothesis by the equation:

$$p = \begin{cases} 1 - \frac{C^*}{\tau_{\text{face}}} & C^* \le \tau_{\text{face}} \\ 0 & C^* > \tau_{\text{face}} \end{cases} \tag{14}$$

where $C^* = \min_i C_i$ and i indexes the remaining nodes. We have used the value $\tau_{\text{face}} = 40$ in our experiments. Notice that $p \in [0,1]$ is larger if the frame is more likely to contain a face, and smaller if it is less likely. Fig. 5(d) shows the region which best fits the face behavior after the merging process.

V. *ViBE* Active Browsing Environment

Many approaches for content-based management of video databases have focused on query-by-example methods [65], in which an example shot is presented and the database is searched for shots with similar content. However, query-by-example techniques tend to quickly converge to a small set of shots that may not be of interest to the user.

In previous work [16], [44], we introduced the similarity pyramid as a structure for organizing and browsing large image databases. The similarity pyramid uses a tree structure to organize the shots in the database for efficient access. To construct these trees, we apply a bottom-up or agglomerative clustering method because our experiences have shown that bottom-up methods tend to yield better results [16], [44].

In *ViBE*, we adapt this browsing approach to the problem of video databases [12]. The similarity pyramid is organized using the shot distance measure given in (6). Our distance measure allows weights to be adjusted which control the relative importance of color, edges, motion, temporal position, and pseudo-semantic labels. Exploiting this flexibility, the *ViBE* browsing environment allows dynamic reorganization based on the user's needs. For example, the database may be organized to group together shots from the same sequence, or from similar times, or containing similar content. In order to achieve this, the browsing environment is designed to allow user feedback. This feedback can be through the direct selection of parameters that control the relative importance of sequence number, time, content, and labeling in the distance measure. However, it can also be through relevance feedback mechanisms provided in the user interface of the browsing environment [66]. While a number of techniques have used relevance feedback in content-based image retrieval [67]–[69], we believe that the use of relevance feedback for browsing is a very different problem which requires a fundamentally different approach.

A. Similarity Pyramids for Video Browsing

The structure of a similarity pyramid is illustrated in Fig. 7. The similarity pyramid organizes large video databases into a three-dimensional pyramid structure. Each level of the similarity pyramid contains clusters of similar shots organized on a 2-D grid. As users move down the pyramid, the clusters become smaller, with the bottom level of the pyramid containing individual shots. Each cluster of the pyramid is represented by a single key frame chosen from the cluster; so as the user moves through the pyramid they have a sense of database content at various scales. In addition, users can pan across at a single level of the pyramid to see shots or shot clusters that are similar.

The similarity pyramid is created by first hierarchically clustering the shots, reorganizing them into a quadtree structure, and then mapping the quadtree's clusters onto the 2-D grid at each level of the pyramid. The shots are clustered using the distance measure given in (6) and an agglomerative method similar to the general algorithm described in Section III-A. However, we modify the clustering method to exploit the sparse nature of the proximity matrix. For a database with N shots, we only compute the distance between each shot and its M closest matches, and then use a clustering technique similar

TABLE I
NUMBER OF DIFFERENT TYPES OF SHOT BOUNDARIES FOR DIFFERENT PROGRAME GENRES USED. THE MEAN SHOT LENGTH AND STANDARD DEVIATION IN FRAMES IS ALSO GIVEN

Genre	# frames	# cuts	# dissolves	# fades	# others	avg. shot length
soap	67582	337	2	0	0	196 ± 17
talk	107150	331	108	1	6	239 ± 97
sports	78051	173	45	0	29	299 ± 40
news	58219	297	7	0	6	182 ± 10
movies	54160	262	15	6	1	760 ± 270
cspan	90269	95	19	0	0	206 ± 81
TOTAL	455431	1495	196	7	42	

TABLE II
RESULTS FOR CUT DETECTION USING THE GT/TREE CLASSIFIER (GT), SLIDING WINDOW (SW), AND SIMPLE THRESHOLDING (ST) METHODS. FOR THE SW AND ST METHODS THE PERFORMANCE FOR EACH GENRE IS COMPUTED AS THE MEAN PERFORMANCE OVER ALL SEQUENCES IN THAT GENRE. FOR THE GT METHOD, THE MEAN IS TAKEN OVER THE FOUR TRAINING INSTANCES, AS DESCRIBED IN SECTION VI-B. *D* INDICATES THE DETECTION RATE AND *FA* INDICATES THE NUMBER OF FALSE ALARMS, FOR EACH PROGRAM GENRE. *MC* ARE THE NUMBER OF MISCLASSIFICATIONS WHERE A CUT WAS DECLARED AT A POSITION WHERE A SHOT BOUNDARY OTHER THAN A CUT WAS PRESENT

	tree classifier			*sliding window*			*simple thresholding*		
Genre	D	FA	MC	D	FA	MC	D	FA	MC
soap	0.941	13.3	0	0.916	99	0	0.852	24	0
talk	0.942	32.3	7.5	0.950	45	1	0.968	171	15
sports	0.939	82.5	34.8	0.785	59	1	0.925	251	73
news	0.958	38.0	0.75	0.886	61	0	0.926	212	1
movies	0.821	43.3	2	0.856	25	0	0.816	25	3
cspan	0.915	54.3	8.5	0.994	40	0	0.943	3	20

to the flexible method developed in [70]. The result of this clustering is a binary tree which is subsequently converted to a quadtree, and then remapped to the pyramid structure in a way which best preserves the organization of the database [71].

B. Browsing Interface

Fig. 8 shows the browsing environment presented to the user. It consists of three blocks: the display space on the left, which is used to browse through keyframes from shots in the database, the relevance space on the right where the set of relevant shots as judged by the user, which we call the *relevance set*, are displayed and the query space in the middle where users access various querying functionalities. The grid representation of keyframes extracted from shots is a very efficient method for the user to get a quick overview of the material in the database and have been used in other video database management interfaces [8]. For a large database, even the upper levels of the similarity pyramid will be too large to display entirely in the display space. Therefore, the user can move along the horizontal or vertical directions in a panning motion to search for shot clusters of interest. If a specific cluster appears to be of interest, then the user can choose to move a level deeper in the pyramid by clicking on the cluster using the mouse. The next level of the pyramid is then presented with the corresponding group of four children clusters centered in the view. Alternatively, the user may desire to backtrack to a higher level of the pyramid. In this case, the previous level of the pyramid is presented with proper centering.

The relevance set is a set of shots that the users deem to be interesting and select as they move through the pyramid. Shots may be incrementally added or removed from the relevance set at any time during the browsing process. For browsing, the user's objective may be to locate all shots from a desired class for example anchorperson shots. In this case, the relevance set may contain dissimilar groupings of shots that represent the variations that can occur among shots in the class. As the user browses through the database, the relevance set also serves as a clipboard which stores all the shots of interest to the user. Once a set of relevant shots has been selected, it may be associated with a semantic label. These relevance sets may then be stored and recalled using this semantic label. We incorporate relevance feedback into two basic browsing functions, pruning and reorganization, which are discussed next.

C. Pruning of the Database

Pruning removes shots from the database that are not likely to be of interest to the user while retaining all or most potentially desirable shots. Let $R = \{x_1, \ldots, x_M\}$ be the set of M relevant keyframes that a user selects for the relevance set. We would like to retain all keyframes y in the database such that

$$D(y, R) \leq T \tag{15}$$

where T is a threshold and the distance measure $D(y, R)$ is defined as

$$D(y, R) = \min_{x \in R} d_{\boldsymbol{\theta}}(y, x). \tag{16}$$

The distance $d_{\boldsymbol{\theta}}(y, x)$ is a simple histogram-based distance function characterized by the parameter vector $\boldsymbol{\theta}$ which is a 13 component vector containing weights corresponding to the L, a, and b components of the color, edge, and texture histogram features [16] of the shot tree, and the weights K_{Seq}, K_{Shot}, K_{Motion}, and K_{Semantic} which were defined in Section III.

The selection of the threshold T is done as follows: Let $R_i = R - \{x_i\}$ be the set of relevance keyframes with the keyframe x_i removed. We define $D_i = D(x_i, R_i)$ and denote the order statistics of D_i as $D_{(n)}$ so that $D_{(n)} \leq D_{(n+1)}$. It can be shown [66] that, if we choose the threshold as $T = D_{(n)}$, then the pruned database will contain on the average $n/(M+1)$ of the keyframes that satisfy (15). For example, if the relevance set contains 10 images and $D_{(10)}$ is used as the threshold, then on average $10/11$ or 91% of the desired shots will be retained in after the pruning. An important result proved in [66] is that the above result does not depend on the statistics of the database or the properties of the distance measure, $d_{\boldsymbol{\theta}}(y, x)$, used.

Fig. 9. Boxes indicate the faces detected for $C \leq 20$.

TABLE III
RESULTS FOR FACE DETECTION USING THE GROUND TRUTH SEQUENCES. *FACES* INDICATES THE NUMBER OF SHOTS CONTAINING AT LEAST ONE FACE. *DETECT* AND *FA* INDICATE THE DETECTION AND FALSE ALARM RATES, RESPECTIVELY, AND *CORRECT* INDICATES THE CORRECT CLASSIFICATION RATE

Sequence	Shots	Faces	Detect (%)	FA (%)	Correct (%)
news1	231	76	73.7	16.1	80.5
news2	72	29	93.1	23.3	83.3
news3	78	33	87.9	15.6	86.0
news4	103	42	90.5	13.1	88.3
news5	188	51	76.5	13.9	83.5
movie	142	92	84.8	28.0	80.3
drama	100	90	94.4	20.0	93.0
total	914	413	85.2	17.0	84.0

Pruning is useful because it reduces the size of the database, and therefore makes browsing easier and more effective. While traditional query methods attempt to find shots that are likely to be of interest, pruning retains all shots which are of possible interest, but at the expense of retaining many questionable shots. Intuitively, pruning attempts to achieve high recall, but at the expense of precision; whereas traditional queries tend to emphasize precision over recall. The reduced precision of pruning is acceptable in *ViBE* because the similarity pyramid structure allows users to efficiently browse the resulting shots.

D. Reorganization of the Database

The objective of reorganization is to dynamically rebuild the similarity pyramid based on the relevance information. As with pruning, reorganization makes the browsing process more efficient by moving the desired shots nearer to one another in the similarity pyramid structure.

Our objective is to find an estimate for the parameter vector, $\hat{\boldsymbol{\theta}}$. This is done by minimizing a cost function $E(\boldsymbol{\theta}, R)$ which uses cross-validation to estimate the separability of the relevance set R from the rest of the database. The detailed form of $E(\boldsymbol{\theta}, R)$ is given in [66]. With this cost function, we compute the optimal value of the parameter $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}, R)$ using conjugate gradient optimization. The resulting distance function $d_{\hat{\boldsymbol{\theta}}}(y, x)$ is then used in (16) to rebuild the similarity pyramid. The cross-validation approach gives reliable performance independently of the database content and the specific choice of similarity measures [66].

VI. EXPERIMENTAL RESULTS

A. Experimental Data Set

At this time *ViBE* contains more than 100 hours of MPEG-1 sequences, which were recorded from miscellaneous television programs. The sequences have been digitized at a rate of 1.5 Mbits/s in SIF format (352×240 pixels). All sequences used in our database are copyright cleared for use in *ViBE*. We have selected 29 sequences from our database which we believe represent a number of different program genres. Commercials in these sequences, if they exist, were edited out. The locations and types of all the shot transitions in these sequences were recorded by a human operator.

These 29 sequences are classified into six program genres as follows: Soap operas (five sequences) which consist mostly

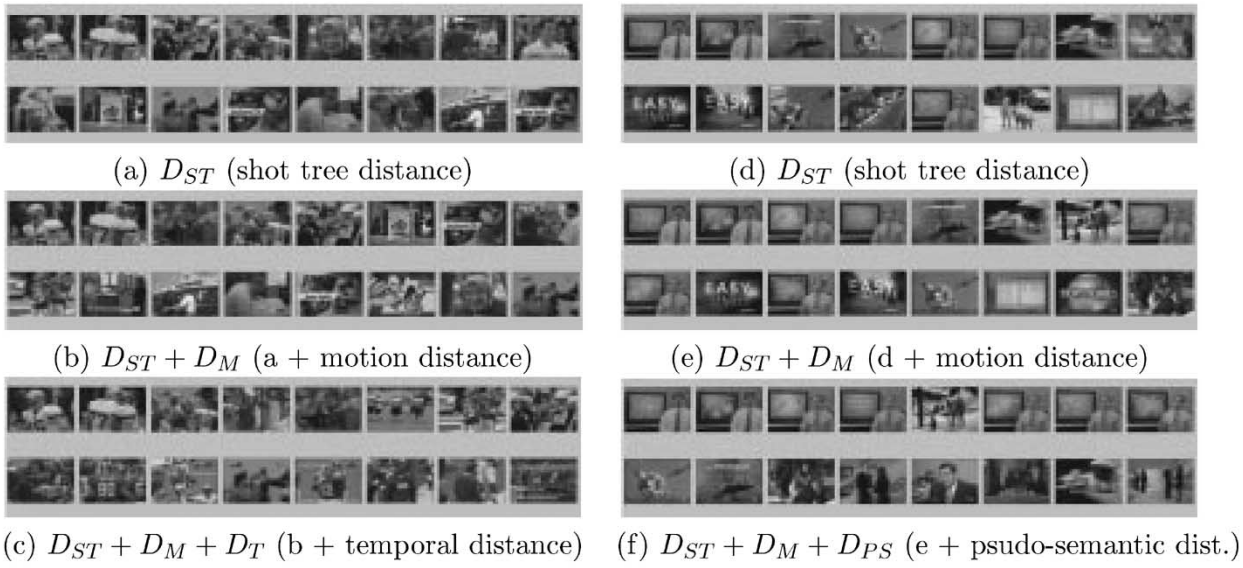

Fig. 10. Example of search results using different similarity measures.

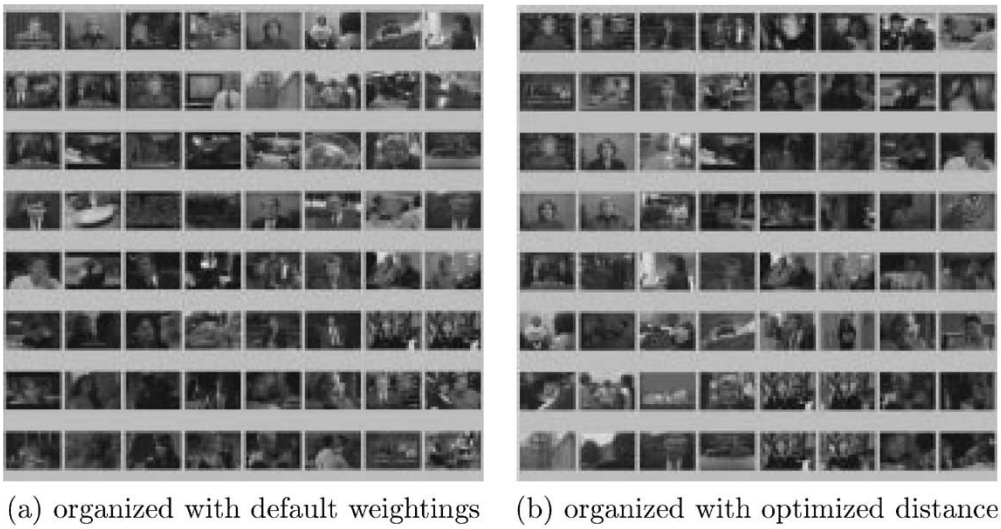

Fig. 11. Pruning result of Fig. 8. (a) Pyramid organized with the default weights. (b) Pyramid organized using the optimized distance mpeasure.

of dialogue with little camera motion; talk shows (seven sequences), which contain a large number of dissolves and camera motion at the start of the programs; sports (five sequences), containing miscellaneous sequences from football, basketball, and car racing programs with a lot of special effects shot transitions. rapidly moving objects, and a high degree of camera motion; news (four sequences); movies (three sequences) which contain sequences from the films *Before He Wakes* and *Lawrence of Arabia*; and cspan (five sequences) obtained from C-SPAN and C-SPAN2 networks. containing very long shots with very little camera or object motion.

With the exception of the sequences from *Lawrence of Arabia* we never use more than one sequence from a given airing of a particular program in order to achieve maximum content variation. Statistical information about these sequences is summarized in Table I.

B. Cut Detection Experiments

To get an honest estimate of the performance of our cut detection system, we have used the following procedure which is similar to a cross-validation procedure shown at the bottom of the next page.

Our experiments have shown that using a window size of $W = 1$, which means that the regression tree uses 36 features, provides a reasonable balance between complexity and performance for cut detection so we have used this value for our experiments. For all results of our method, we have used a threshold of $\tau = 0.35$ as our detection criteria.

We have compared our method with the sliding window technique proposed in [28] and often used in cut detection. In this technique, a symmetric window of size $2m+1$ is placed around the ith frame and a cut is declared from frame i to $i+1$ if

1) the value of the feature value for i is the maximum within the window;
2) it is also n times the value of the second maximum in the window.

We have used the sum of the histogram intersections of the Y, U, and V components, i.e., $g_{i,1} + g_{i,2} + g_{i,3}$ as the frame similarity feature. We chose the values $m = 7$ and $n = 2$ because these values gave the best over all performance on our data sets. We should point out that the sliding window technique does not use all the features in the feature vector and so has a distinct disadvantage. Nevertheless, we use only the sum of the color histogram intersections as the feature for the sliding window algorithm in order to be able to compare our results with those of a large number of authors. who have used only this feature. We have also compared our results with a global thresholding technique which uses the sum of the histogram intersections of the Y, U, and V components, i.e., it uses $g_{i,1} + g_{i,2} + g_{i,3}$ as the frame similarity feature. This simple method is included here as a baseline, to serve as an indication of the relative difficulty in detecting the cuts in various video genres. A global threshold value of 0.45 was found to give best results and this value was used for all of our experiments. Again, we remove the cut with a lower feature value if two cuts are closer than ten frames.

The results of these experiments are shown in Table II. From this table we can make the following observations: The GT/regression tree method gives a very consistent performance for video sequences with diverse content whereas the sliding window method runs into problems with some types of sequences. This is most clear with the *sports* and *news* genres, where the the detection rate of the sliding window detector is low. Our experiments have shown that the genres of the sequences used to train the regression tree do not affect the performance of the system.

C. Pseudo-Semantic Classification

We have tested the face labeling algorithm described in Section IV using seven video sequences containing various program content. The root nodes of the shot trees were determined for each shot and the corresponding frames were extracted from the MPEG sequence at full frame resolution. Then ground truth information relative to "face" and "no face" was marked by hand for each keyframe so obtained. The algorithm was trained using "face masks" obtained from manually segmented faces from 100 randomly chosen frames that contained faces. The results were evaluated as follows. A false alarm is declared if the frame contains no faces, but our algorithm detects at least one face; a detection is declared if the frame contains at least one face, and our algorithm detects at least one face; and a correct classification is declared if there is no face in the frame and our algorithm detects no faces.

Once we have merged skin-like regions, we apply a threshold to the dissimilarity values given by (12) in the remaining regions. If any value below the threshold exists, the frame is said to contain at least one face. Thus, an optimal threshold can be obtained that maximizes the classification rate. It is important to emphasize that no thresholds are used when similarity measurements are performed. Our algorithm produces confidence values that are then used by the browsing environment. Thresholds are used in this section only to show the performance of the face labeling stage.

In Fig. 9(a), bounding box is drawn in each image for regions with dissimilarity value, C, defined by (12) is less than 20. As shown, we are able to detect faces under many different illumination conditions and poses. Some false alarms, where faces are detected in frames containing no human faces, are seen in Fig. 9(g) and (m)–(p). Results for the face detection algorithm for a number of sequences are given in Table III.

D. Active Browsing

Fig. 10(a)–(f) demonstrate the significance of each component of the shot dissimilarity measure. Fig. 10(a)–(c) show the first 15 shots returned for a query using a football shot (upper left hand shot) when searching using 23 sequences of video. Fig. 10(a) shows the result when only the D_{ST} distance is used to compute the the distance between shot trees. Fig. 10(b) and (c) show the result when the motion distance, D_M, and the temporal distance, D_T, are added, respectively. We observe that the temporal and motion information are important cues in identifying good matching shots.

Fig. 10(d)–(f) show the first 15 shots returned for a query using a local weather report shot (upper left hand shot) when searching using seven sequences of video. Fig. 10(d) shows the result when only using D_{ST} as the distance between shot trees. Fig. 10(e) shows the result when the motion distance, D_M, is added, and Fig. 10(f) is the result when the pseudo-semantic distance, D_{PS}, is added. For this search, the pseudo-semantic distance is very effective at identifying matching shots.

In Fig. 11, we show the shots obtained using relevance feedback for similarity pyramid pruning and redesign using the original pyramid and relevance set shown in Fig. 8. Although the shots in this relevance set have dissimilar attributes, they all form a single semantic category, "anchorperson." The pruned set contains most of the shots which closely match the ones in the relevance set. Fig. 11(b) shows the reorganized sub-database where the dissimilarity function is defined by the optimized distance described in Section V-D. Notice that the shots in the rel-

---

for each genre $G \in \{soap, talk, sports, news, movies, cspan\}$
  for $i = 1$ to 4.
    randomly choose two sequences, S_1 and S_2, both not in G
    train the regression tree using S_1 and S_2
    process all the sequences in G using this tree
    average the cut detection performance of the tree over the sequences in G
  average the four sets of values to get the performance for genre G

evance set are clustered together. These experiments with *ViBE* indicate that the use of the pseudo-semantic labels and motion and temporal information can provide powerful aids in querying and browsing digital video libraries.

VII. Conclusion

In this paper, we presented a new paradigm for video database management known as *ViBE* introduces a variety of novel algorithms and techniques for processing, representing, and managing digital video in an integrated environment. We introduced a new way of performing shot boundary detection through the use of the Generalized Trace and regression trees. We described a powerful way of representing each video shot using a binary tree and discussed shot similarity measures that exploit information in the shot tree and the shot boundaries. We also showed that the use of pseudo-semantic labels provide a novel way of describing shots that can be used in our browsing environment.

Future work includes populating *ViBE* with more video sequences, extending the GT/regression tree approach to other types of shot transitions, the description of more pseudo-semantic labels and the use of this information in our browsing environment. Our experiments have demonstrated that the use of pseudo-semantic labels and motion information obtained from shot boundary detection provide powerful cues in aiding browsing.

References

[1] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 1349–1380, Dec. 2000.

[2] A. Jaimes and S.-F. Chang, "A conceptual framework for indexing visual information at multiple levels," in *Proc. SPIE Conf. Internet Imaging*, vol. 3964, San Jose, CA, Jan. 2000.

[3] "MPEG-7: Requirements document (v12) w3548," in MPEG Meeting, Beijing, China, July 2000, IsO/JTC1/SC29/WG11.

[4] P. Salembier, R. Qian, N. O'Connor, P. Correia, I. Sezan, and P. van Beek, "Description schemes for video programs, users and devices," *Signal Process.: Image Commun.*, vol. 16, no. 1–2, pp. 211–234, Sept. 2000.

[5] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatio–temporal queries," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 602–615, Sept. 1998.

[6] A. Hampapur, A. Gupta, B. Horowitz, C. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain, "Virage video engine," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases V*, vol. 3022, San Jose, CA, Feb. 13–14, YEAR???, pp. 188–197.

[7] S. Srinivasan, D. Ponceleon, A. Amir, and D. Petkovic, "What is in that video anyway?: In search of better browsing," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, Florence, Italy, June 1999, pp. 388–392.

[8] B. Davies, R. Lienhart, and B.-L. Yeo, "The video document," in *Proc. SPIE Conf. Multimedia Storage and Archiving Systems IV*, vol. 3846, Sept. 20–22, 1999, pp. 22–34.

[9] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "DANCERS: Delft advanced news retrieval system," in *Proc. SPIE Conf. Storage and Retrieval for Media Databases 2001*, San Jose, CA, Jan. 2001, p. 301.

[10] H. Lee, A. Smeaton, C. O'Toole, N. Murphy, S. Marlow, and N. O'Connor, "The fschlr digital video recording, analysis, and browsing system," in *Proc. Content-Based Multimedia Information Access Conf. (RIAO 2000)*, Paris, France, Apr. 2000, pp. 1390–1399.

[11] H. D. Wactlar, "Informedia—search and summarization in the video medium," in *Proc. IMA GINA Conf.*, Monaco, Jan. 31–Feb. 2 2000, p. 2000.

[12] J.-Y. Chen, C. Taskiran, A. Albiol, C. A. Bouman, and E. J. Delp, "ViBE: A video indexing and browsing environment," in *Proc. SPIE Conf. Multimedia Storage and Archiving Systems IV*, vol. 3846, Boston, MA, Sept. 1999, pp. 148–164.

[13] C. Taskiran, C. Bouman, and E. J. Delp, "The vibe video database system: An update and further studies," in *Proc. SPIE/IS&3T Conf. Storage and Retrieval for Media Databases 2000*, San Jose, CA, Jan. 26–28, 2000, pp. 199–207.

[14] C. Taskiran and E. J. Delp, "Video scene change detection using the generalized trace," presented at the IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Seattle, WA, May 12–15, 1998.

[15] L. Breiman, J. F. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[16] J. Y. Chen, C. A. Bouman, and J. C. Dalton, "Similarity pyramids for browsing and organization of large image databases," in *Proc. SPIE Conf. Human Vision and Electionic Imaging III*, vol. 3299, San Jose, CA, Jan. 26–29, 1998.

[17] U. Gargi, R. Kasturi, and S. H. Strayer, "Fast scene change detection using direct feature extraction from MPEG compressed videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1–13, Feb. 2000.

[18] R. Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases VII*, vol. 3656, San Jose, CA, Jan. 1999, pp. 290–301.

[19] C. O'Toole, A. Smeaton, N. Murphy, and S. Marlow, "Evaluation of automatic shot boundary detection on a large video test suite," presented at the 2nd UK Conf. Image Retrieval, Newcastle, U.K., Feb. 25–26, 1999.

[20] J. Boreczky and L. Rowe, "Comparison of video shot boundary detection techniques," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases IV*, vol. 2670, San Jose, CA, Feb. 1996, pp. 170–179.

[21] A. Dailianas, R. B. Allen, and P. England, "ComPArison of automatic video segmentation algorithms," in *Proc. SPIE Photonics East'95: Integration Issues in Large Commercial Media Delivery Systems*, Philadelphia, PA, Oct. 1995, pp. 1–16.

[22] E. Ardizzone and M. L. Cascia, "Multifeature image and video content-based storage and retrieval," in *Proc. SPIE Conf. Multimedia Storage and Archiving Systems*, vol. 2916, Boston, MA, Nov. 18–19, 1996, pp. 265–276.

[23] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances," in *Visual Databases II*, E. Knuth and L. M. Wegner, Eds. Austin, TX, 1995, pp. 113–127.

[24] B. Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proc. SPIE Conf. Digital Video Compression: Algorithms and Technologies*, vol. 2419, San Jose, CA, Feb. 1995, pp. 2–13.

[25] N. V. Patel and I. K. Sethi, "Video shot detection and characterization for video databases," *Pattern Recognit.*, vol. 30, no. 4, pp. 583–592, Apr. 1997.

[26] W. Zhao, J. Wang, D. Bhat, K. Sakiewicz, and N. Nandhaknmar, "Improving color based video shot detection," in *IEEE Int. Conf. Multimedia Computing and Systems*, Florence, Italy, June 7–11, 1999, pp. 752–756.

[27] A. M. Ferman and A. M. Tekalp, "Efficient filtering and clustering methods for temporal video segmentation and visual summarization," *J. Vis. Commun. Image Represent.*, vol. 9, pp. 336–352, 1998.

[28] B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 533–544, Dec. 1995.

[29] S.-W. Lee, Y.-M. Kim, and S. W. Choi, "Fast scene change detection using direct feature extraction from MPEG compressed videos," *IEEE Trans. Multimedia*, vol. 2, pp. 240–254, Dec. 2000.

[30] J. Nang, S. Hong, and Y. Ihm, "An efficient video segmentation scheme for MPEG video stream using macroblock information," presented at the 7th ACM Int. Multimedia Conf., Orlando, FL, Oct. 30–Nov. 5, 1999.

[31] S.-C. Pei and Y.-Z. Chou, "Efficient MPEG compressed video analysis using macroblock type information," *IEEE Trans. Multimedia*, vol. 1, pp. 321–333, Dec. 1999.

[32] P. Aigrain and P. Joly, "The automatic real-time analysis of film editing and transition effects and its applications," *Comput. Graph.*, vol. 18, no. 1, pp. 93–103, 1994.

[33] N. Vasconcelos and A. Lippman, "A bayesian video modeling framework for shot segmentation and content characterization," presented at the IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'97), San Juan, PR, Oct. 4–7, 1997.

[34] M. R. Naphade and T. S. Huang, "Stochastic modeling of soundtrack for efficient segmentation and indexing of video," in *Proc. SPIE Conf. Storage and Retrieval for Media Databases 2000*, vol. 3972, San Jose, CA, Jan. 2000, pp. 168–176.

[35] J. Boreczky and L. D. Wilcox, "A hidden markov model framework for video segmentation," in *Proc. IEEE Int. Conf. Acoustic, Speech and Signal Processing*, Seattle, WA, May 1998, pp. 3741–3744.
[36] J. Huang, Z. Liu, and Y. Wang, "Joint video scene segmentation and classification based on hidden markov model," in IEEE Int. Conf. Multimedia and Expo (ICME 2000), New York, NY, July 30-Aug. 2, 2000.
[37] K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proc. IEEE Int. Conf. Image Processing*, Washington, DC, Oct. 26–29, 1995, pp. 252–255.
[38] H. Cheng and C. A. Bouman, "Multiscale bayesian segmentation using a trainable context model," *IEEE Trans. Image Processing*, vol. 10, no. 4, pp. 511–525, Apr. 2001.
[39] C. B. Atkins, C. A. Bouman, and J. P. Allebach, "Optimal image scaling using pixel classification," in *Proc. IEEE Int. Conf. Image Processing*, Thessaloniki, Greece, Oct. 7–10, 2001, pp. 864–867.
[40] S. Gelfand, C. Ravishankar, and E. Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 163–174, Feb. 1991.
[41] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *Proc. IEEE Int. Conf. Image Processing*, Chicago, IL, Oct. 1998, pp. 866–870.
[42] M. S. Drew and J. Au, "Video keyframe production by efficient clustering of compressed chromaticity signatures," in *Proc. Eighth ACM Multimedia Conf.*, Los Angeles, CA, Oct. 30-Nov. 4, 2000, pp. 365–367.
[43] K. Y. Kupeev and Z. Sivan, "An algorithm for efficient segmentation and selection of representative frames in video sequences," in *Proc. SPIE Conf. Storage and Retrieval for Media Databases 2001*, vol. 4315, San Jose, CA, Jan. 2001, pp. 24–26.
[44] J.-Y. Chen, C. A. Bouman, and J. C. Dalton, "Hierarchical browsing and search of large image databases," *IEEE Trans. Image Processing*, vol. 9, pp. 442–455, Mar. 2000.
[45] J. H. Ward, Jr., "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, pp. 236–244, 1963.
[46] M. R. Naphade, M. M. Yeung, and B.-L. Yeo, "A novel scheme for fast and efficient video sequence matching using compact signatures," in *Proc. SPIE Conf. Storage and Retrieval for Media Databases*, vol. 3972, San Jose, CA, Jan. 2000, pp. 426–431.
[47] A. K. Jain, A. Vailaya, and W. Xiong, "Query by video clip," *Multimedia Syst.*, vol. 7, no. 5, pp. 369–384, 1999.
[48] M. M. Yeung, B.-L. Yeo, and B. Liu, "Extracting story units from long programs for video browsing and navigation," in *IEEE Int. Conf. Multimedia Computing and Systems*, Hiroshima, Japan, June 17–21, 1996, pp. 296–305.
[49] S. Santini and R. Jain, "Beyond query by example," in *IEEE Second Workshop on Multimedia Signal Processing*, Redondo Beach, CA, Dec. 1998, pp. 3–86.
[50] N. Haering, R. J. Qian, and M. I. Sezan, "A semantic event-detection approach and its application to detecting hunts in wildlife video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 857–868, Sept. 2000.
[51] M. R. Naphade and T. S. Huang, "A probabilistic framework for semantic indexing and retrieval in video," presented at the IEEE Int. Conf. Multimedia and Erpo, New York, July-Aug. 31–2, 2000.
[52] W. Mahdi, M. Ardabilian, and L. Chen, "Automatic video content parsing based on exterior and interior shots classification," in *SPIE Conf. Multimedia Storage and Archiving Systems*, vol. 3846, Boston, MA, Sept. 20–22, 1999, pp. 46–55.
[53] M. Szummer and R. W. Picard, "Indoor-outdoor image classification," in *IEEE Int. Workshop on Content-Based Access of Video and Image Databases*, Bombay, India, Jan. 1998, pp. 42–51.
[54] A. Vailaya, M. Figueiredo, A. Jain, and H. J. Zhang, "A bayesian framework for semantic classifi cation of outdoor vacation images," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases VII*, vol. 3656, San Jose, CA, Feb. 1999, pp. 415–426.
[55] K. C. Yow and R. Cipolla, "Feature-based human face detection," *Image Vis. Comput.*, vol. 15, no. 9, pp. 713–735, 1997.
[56] H. A. Rowley, S. Baluja, and T. Kanade, "Human face detection in visual scenes," *Adv. Neural Inform. Process. Syst.*, vol. 8, pp. 875–881, 1996.
[57] A. J. Comenarez and T. S. Huang, "Face detection with information-based maximum discrimination," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, San Juan, PR, 1997, pp. 782–787.
[58] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions, merging and waveletpacket analysis," *IEEE Trans. Multimedia*, vol. 1, pp. 264–277, Sept. 1999.
[59] A. Albiol, C. A. Bouman, and E. J. Delp, "Face detection for pseudo-semantic labeling in video databases," in *Proc. IEEE Int. Conf. Image Processing*, Kobe, Japan, Oct. 25–28, 1999.
[60] H. Wang and S.-F. Chang, "A highly efficient system for automobile facerecognition in MPEG video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 615–628, Aug. 1997.
[61] H. Luo and A. Eleftheriadis, "On face detection in the compressed domain," in *Proc. ACM Multimedia Conf.*, Los Angeles, CA, Nov. 2000, pp. 285–294.
[62] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 39–51, Jan. 1998.
[63] M. H. Yang and N. Ahuja, "Detecting human faces in color images," in *Proc. IEEE Int. Conf. Image Processing*, Chicago, IL, Oct. 4–7, 1998, pp. 127–130.
[64] S. Beucher and F. Meyer, "The morophological approach to segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, E. R. Dougherty, Ed. New York: Marcel Dekker, 1993, ch. 12, pp. 433–481.
[65] D. Zhong and S.-F. Chang, "Spatio–temporal video search using the object based video represen-tation," in *Proc. IEEE Int. Conf. Image Processing*, Santa Barbara, CA, Oct. 1997, pp. 21–24.
[66] J.-Y. Chen, C. A. Bouman, and J. Dalton, "Active browsing using similarity pyramids," in *Proc. SPIE/IS&4T Conf. Storage and Retrieval for Image and Video Databases VII*, vol. 3656, San Jose, CA, Jan. 24–29, 1999, pp. 144–154.
[67] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, "Pichunter: Bayesian relevance feedback for image retrieval," in *Proc. Int. Conf. Pattern Recognition*, vol. 3, Vienna, Austria, Aug. 1996, pp. 361–369.
[68] L. Taycher, M. L. Cascia, and S. Sclaroff, "Image digestion and relevance feedback in the Imagerover WWW search engine," presented at the Int. Conf. Visual Information, San Diego, CA, Dec. 15–17, 1997.
[69] Y. Rui, T. S. Huang, and S. Mehrotra, "Relevance feedback techniques in interactive content-based image retrieval," in *Proc. SPIE/IS&4T Conf. Storage and Retrieval for Image and Video Databases VI*, vol. 3312, San Jose, CA, Jan. 26–29, 1998, pp. 25–36.
[70] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies. i. Hierarchical systems," *Comput. J.*, vol. 9, no. 5, pp. 373–380, 1966.
[71] J. Y. Chen, C. A. Bouman, and J. P. Allebach, "Fast image database search using tree-structured VQ," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, Santa Barbara, CA, Oct. 26–29, 1997, pp. 827–830.

Cuneyt Taskiran (S'03) received the B.S. and M.S. degrees, both in electrical engineering from Bogazici University, Istanbul, Turkey, in 1990 and 1993, respectively. Currently, he is pursuing the Ph.D. degree in the Electrical and Computer Engineering Department and the M.A. degree in the Linguistics Department, both at Purdue University, West Lafayette, IN.

During his Ph.D. studies he has worked at Sharp Labs of America, Camas, WA, and IBM Almaden Research Center, San Jose, CA. At Sharp Labs, he worked on text analysis algorithms to automatically fetch and filter business news from the Internet and investigated the use of this for a prospective home audio-visual system, which has been patented. At IBM Almaden, he worked with the CueVideo team to develop video summarization techniques for rapid navigation in large video databases and integrated this system into CueVideo for video management. Currently, he is developing video analysis tools for C-SPAN Archives. His research interests include context-based multimedia retrieval, video understanding and analysis using stochastic models, and statistical text analysis for natural language processing applications.

Jau-Yuen Chen was born in Pingtong, Taiwan, R.O.C., on January 25, 1963. He received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, in 1985 and 1987, respectively, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1999.

From 1987 to 1994, he was an Assistant Scientist at the Chung-Shan Institute of Science and Technology, Taiwan. He is currently with Epson, Palo Alto, CA.

Alberto Albiol (M'03) received a degree in telecommunications engineering in 1995 and the Ph.D. degree in 2003, both from the Politechnic University of Valencia, Spain.

From 1995 to 1997, he was with the University of Zaragoza, Spain, as an Associate Professor, and in 1997, he joined the Communications Deparment of the Politechnic University of Valencia, where he is currently a Lecturer Professor. In 1998, he was a Visiting Scholar at Purdue University, West Lafayette, IN, where he worked in the VIPER Laboratory. His research interests include digital image and video processing and multimedia applications.

Dr. Albiol has served on technical conference committees and as a reviewer for many international journals, and has also participated in several Spanish and European projects in the fields of image and video analysis and video indexing.

Luis Torres (M'91–SM'00) received a degree in telecommunication engineering from the Telecommunication School of the Technical University of Catalonia, Barcelona, Spain, in 1977, and the Ph.D. degree from the University of Wyoming, Laramie, in 1986.

He is a Professor at the Technical University of Catalonia, where he teaches telecommunication systems and advanced video coding courses. He has been responsible of national and international projects in very low bit-rate video coding applications, face recognition and MPEG-4 and MPEG-7 standardization activities. His main interests are image and video coding, image and video analysis and face recognition. He publishes actively in numerous conference and journal papers. He is also co-editor with Murat Kunt of the book *Video Coding: The Second Generation Approach* (Norwell, MA: Kluwer, January 1996).

Dr. Torres is currently serving as Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, is a member of the IEEE Image and Multidimensional Digital Signal Processing Committee and was the General Chair of the International Conference on Image Processing (ICIP), held in Barcelona, in September 2003. He has served in technical conference committees and as a reviewer in many international journals and also served as coding expert for the European Commission evaluating technical proposals and designing research European programs.

Charles A. Bouman (S'86–M'89–SM'97–F'01) received the B.S.E.E. degree from the University of Pennsylvania, Philadelphia, in 1981, and the M.S. degree in electrical engineering from the University of California at Berkeley in 1982. In 1987 and 1989, respectively, he received the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, under the support of an IBM graduate fellowship.

From 1982 to 1985, he was a Staff Member in the Analog Device Technology Group at the Lincoln Laboratory, Massachusetts Institute of Technology. In 1989, he joined the faculty of Purdue University, West Lafayette, IN, where he is a Professor in the School of Electrical and Computer Engineering. His research interests include statistical image modeling and analysis, multiscale processing, and the display and printing of images. He is particularly interested in the applications of statistical signal processing techniques to problems such as document image segmentation and compression, tomographic reconstruction, optical inverse problems, and high quality printing and rendering. His research has resulted in new methods for image rendering, halftoning, and display that have been widely used in commercial products. He has authored over 30 full journal publications, over 90 conference publications, and is an inventor on five issued patents. He has performed research for numerous government and industrial organizations including National Science Foundation, U.S. Army, Hewlett-Packard, Xerox, NEC Corporation, Apple Computer, and Eastman Kodak.

Prof. Bouman is a member of the SPIE and IS&T professional societies. He has been an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, and is currently an Associate Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and is Vice President of Publications for the IS&T Society. He was a member of the ICIP 1998 organizing committee, is currently a chair for the SPIE conference on Visual Communications and Image Processing 2000 (VCIP), and is a member of the IEEE Image and Multidimensional Signal Processing Technical Committee.

Edward J. Delp (S'70–M'79–SM'86–F'97) was born in Cincinnati, OH. He received the B.S.E.E. (cum laude) and M.S. degrees from the University of Cincinnati and the Ph.D. degree from Purdue University, West Lafayette, IN. In May 2002, he received an Honorary Doctor of Technology degree from the Tampere University of Technology, Tampere, Finland.

From 1980 to 1984, he was with the Department of Electrical and Computer Engineering at The University of Michigan, Ann Arbor. Since August 1984, he has been with the School of Electrical and Computer Engineering and the Department of Biomedical Engineering at Purdue University. In 2002, he received a chaired professorship and currently is The Silicon Valley Professor of Electrical and Computer Engineering and Professor of Biomedical Engineering. His research interests include image and video compression, multimedia security, medical imaging, multimedia systems, communication and information theory. During the summers of 1998 and 1999–2003, he was a Visiting Professor at the Tampere International Center for Signal Processing at the Tampere University of Technology.

Dr. Delp is a Fellow of the SPIE and of the Society for Imaging Science and Technology (IS&T). In 2000, he was selected a Distinguished Lecturer of the IEEE Signal Processing Society. In 1990, he received the Honeywell Award, and in 1992, the D. D. Ewing Award, both for excellence in teaching. In 2001, he received the Raymond C. Bowman Award for fostering education in imaging science from the IS&T. In 2002, he was awarded a Nokia Fellowship.