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

Cuneyt Taskiran[*], Jau-Yuen Chen[*], Alberto Albiol[†],
Luis Torres[‡], Charles A. Bouman[*] and Edward J. Delp[*]
[*]School of Electrical and Computer Engineering
Purdue University West Lafayette, IN 47907-1285
{taskiran, bouman, ace}@ecn.purdue.edu
[*]Epson Palo Alto Laboratory
3145 Porter Drive, Palo Alto CA 94304
jauyuen@erd.epson.com
[†]Departamento de Comunicaciones
Universidad Politécnica de Valencia, Spain
alalbiol@dcom.upv.es
[‡]Universidad Politécnica de Catalunya, Spain
luis@gps.tsc.upc.es

Corresponding Author:
Professor Edward J. Delp
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University West Lafayette, IN 47907-1285
Telephone: +1 765 494 1740
Fax: +1 765 494 0880
Email: ace@ecn.purdue.edu

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.

EDICS: 4-KEEP: Indexing, Searching, Retrieving, Query, and Archiving Databases

---

[*]Please address all correspondence relative to this manuscript to Professor Edward J. Delp.

1 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 [47]. Although some frameworks to define systematic methods for multimedia indexing exist [1], currently there is no accepted universal set of features to describe multimedia information. The MPEG-7 standard [2] 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 to describe video programs and the devices that consume these programs [9].

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 [3], which allows region-based queries with motion information, the Virage video Engine [4], CueVideo from IBM [5], a home movie library management system from Intel [6], and Delft University's DANCERS system [7]. One of the largest efforts is the Informedia system [8].

In this paper, we present an integrated system for managing large databases of video which we call *ViBE* (video indexing and browsing environment) [11, 12]. The *ViBE* system introduces a variety of novel algorithms and techniques for processing, representing, and managing video while keeping the 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 which can be scaled to large databases and extended to a wide variety of functionalities. Figure 1 illustrates the major components of the *ViBE* system.

Shot boundary detection is the first step in processing the 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) [13], are extracted from compressed video. We then use a classifier based on a regression tree [14] 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 operation.

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 [15]. 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.

2   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. Generalization of our technique to other types of shot boundaries may be found in [12].

2.1   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 [16, 17, 18, 19, 20].

Pixelwise frame differences [21, 22], where the differences in intensity or color values of corresponding pixels in two successive frames are used, is one of the oldest methods. Block-based versions of this technique [23] 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 [24]. Various distance measures between histograms have been investigated [25] 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 [26].

Processing in compressed domain offers many advantages and many methods have been pro-

posed 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 [27, 28] and macroblock types [29, 30].

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 [31], shot lengths [32], and interesting events in soundtracts [33] have been proposed. Hidden Markov Models, built using information from both images and audio, have also been used successfully to detect shots [34, 35].

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 may vary considerably from sequence to sequence.

2.2 The 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 [36] 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) [13]. 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

$$g_{i,1} = \frac{1}{2M_1} \sum_{k=1}^{64} |H_i^Y(k) - H_{i+1}^Y(k)| \tag{1}$$

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

4

where M_1 and M_2 are the number of luminance and chrominance blocks, respectively, and which in our case 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

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

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.

2.3   Binary Regression Tree

We use the GT of a video sequence and a binary regression tree [14] to estimate the conditional probability that a given frame from the sequence belongs to a shot transition. 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 method of building regression trees was proposed in [37] where the data set is divided into two roughly equal sets. One set is used to build a large tree that overfits the data. Then the other set is used to prune the tree back to maximize some desirability criteria. This procedure is then iterated, successively interchanging the roles of the first and second ground truth sequences 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*. 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 GT/regression tree method offers a number of advantages. First, the GT feature vector allows a multitude of different features to be collectively used to detect shot transitions. This is important since different features may be useful in detecting different types of shot transitions. The regression tree performs automatic feature selection and complexity reduction [37]. Second, 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. Moreover, the method is highly extensible. New features can easily be incorporated into the existing system.

2.3.1 Detection of Cuts

A schematic diagram of the steps in cut detection is shown in Figure 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 groundtruth sequences, it is used to process the GT from the sequence whose cuts are to be detected. 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

$$\alpha_i = \begin{cases} 1 & \text{if a cut occurs between frames } i \text{ and } i + 1 \\ 0 & \text{if no cut occurs} \end{cases}$$

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 $i + 1$ [14].

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 6.2, if two candidate cut locations are closer than

6

10 frames, the candidate cut with a smaller value of y_i is deleted.

3 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 [9]. 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 [26, 38, 39, 40]. The features that have been used in clustering include 2D color histograms of frames [39, 38] and color histograms averaged over the whole shot [26]. In *ViBE* we obtain the keyframes for a shot using a tree representation as described in the next section.

3.1 The 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 Figure 3a. 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 Figure 3a.

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.

To build the tree we first extract a high dimensional feature vector containing color, texture and edge histogram features proposed in [41] from each DC frame in a shot, and use the L_1 norm to measure feature distances. Let c_i be the set of all frames in cluster i, and let n_i be the number of frames in c_i. The proximity matrix $[d_{ij}]$ defines the pairwise distances between clusters c_i and c_j. Let the shot contain frames $f_j, j = 1 \cdots N$. Initially we let $c_i = \{f_i\}, i = 1 \cdots N$, be the disjoint clusters each of size $n_i = 1$, i.e., one frame per cluster. The proximity matrix $[d_{ij}]$ is then initialized using the L_1 distance between the frames f_i and f_j. 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 distance from c_k to each of the remaining clusters is updated. In *ViBE*, we use Ward's clustering algorithm, also known as the minimum variance method [42] where the distances between the newly formed cluster, k, and all the other clusters, $h \neq k$ are updated according to the rule

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

This pairwise clustering generates a binary tree structure which is used as the representation for the shot. Figure 3a 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.

3.2   The 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 each P or B frame, we first compute, for each frame k, the motion feature

$$m_k = (\# \text{ forward MB}) + (\# \text{ backward MB}) + 2(\# \text{ forward-backward MB}) \tag{5}$$

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 4. 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.

3.3   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 [43], and color, texture, and motion features derived from keyframes [44]. In order to define a shot distance measure that is both robust and comparable 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 j^{th} shot in sequence S_i. Then the distance between two shots is defined as

$$D(s_{ij}, s_{kl}) = D_{ST}(s_{ij}, s_{kl}) + D_T(s_{ij}, s_{kl}) + D_M(s_{ij}, s_{kl}) + D_{PS}(s_{ij}, s_{kl}) \tag{6}$$

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

The *shot tree distance*, $D_{ST}(s_{ij}, s_{kl})$, is measured by obtaining the distance between the two corresponding shot trees as in Figure 3b. Each node, t, of the shot tree is associated with a cluster of frames from the shot and a feature vector, $\mathbf{z}_t$, corresponding to this cluster. The feature vector $\mathbf{z}_t$ includes sequence number, shot number, frame numbers of the beginning and last frame of the shot, the motion histogram, and the tree representation which consists of color, texture, and edge histograms for each frame in the tree structure. We take $\mathbf{z}_t$ to be the centroid of the feature vectors in the cluster. The distance between two shot trees is then given by the weighted sum of

9

the distances between nodes for the best mapping between the shot trees. In this work, we only use trees of depth three, so the optimum mapping can be found by simply checking the 8 distinct mappings between the two trees.

Similar to [45], we define the *temporal distance* between shots s_{ij} and s_{kl} as the following

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

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

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

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

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

We shall describe how these distance measures are used for browsing in Section 5.

4 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 possible. 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 [46]. This is the *semantic gap problem* which is central 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 [47].

Various approaches have been proposed to process low-level features to obtain higher level labels, hence bridging the semantic gap to some degree. In [48] 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 [49]. 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. We are investigating several pseudo-semantic labels. including labels such as "face", "indoor/outdoor", and "high action". 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.

The task of pseudo-semantic labeling is greatly reduced by the shot tree representation described in Section 3. 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 for accurate face labeling, 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 In this paper, because of space constraints we will only describe our work on the "face" label. The goal here is to detect whether a frame in the sequence contains a face. Deriving the other labels follow a similar process.

Different approaches have been developed in recent years for the face detection problem. Some of the most representative methods used include shape-feature approaches [50], neural networks [51], and template matching [52]. 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 [53, 54] both color and spatial information are exploited to detect segmented regions corresponding to faces. The following sections present the details of our approach for determining

the "face" label for a frame.

4.1 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.

Figure 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 [55, 56]. 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. 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.

4.2 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 *non-skin class*, H_1 is

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

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

non-skin classes. In order to achieve a 90% detection rate, a value of $\lambda = 2.8$ was chosen yielding a 20.4% false alarm rate. Figure 5b shows an example where, 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 [57] to cluster the skin detected pixels in the color space. Figure 5c shows the results of the unsupervised segmentation using the regions found in Figure 5b. The algorithm is able to successfully separate the face region from the background.

4.3   Region Extraction and Hierarchical Region Labeling

The next step is to extract connected regions and features vectors which characterize those regions. Let Ω_i be the ith region of pixels, and let $x_{\mathbf{r}}$ be a color pixel in $C_B C_R$ coordinates at position $\mathbf{r} \in \Omega_i$. Here $\mathbf{r} \in \Omega_i$ is assumed to be a two dimensional 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 $\mathbf{v}_i = [N_i, \bar{\mathbf{x}}_i, \gamma_i, \mu_i, \mathbf{max}_i, \mathbf{min}_i]^t$, where N_i is the area of the region, $\bar{x}_i$ and γ_i are vectors containing the mean and variance for each of the Y, C_B, and C_R color components respectively, μ_i is the centroid of the region, and $\mathbf{max}_i$ and $\mathbf{min}_i$ are the corner coordinates of the bounding box of the region. An important property of the state vector is that it can be easily updated recursively for merged regions.

A multivariate Gaussian distribution is used to model the behavior of $\mathbf{v}_i$ for face areas. Let $\bar{\mathbf{v}}$ and $\mathbf{\Sigma}_v$ 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:

$$C_i = (\mathbf{v}_i - \bar{\mathbf{v}})^t \mathbf{\Sigma}_v^{-1} (\mathbf{v}_i - \bar{\mathbf{v}}) \tag{11}$$

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 $\mathbf{\Sigma}_v$ 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 Equation 11. We first form a

distance matrix $\mathbf{C}$ defined by

$$C_{i,j} = (\mathbf{v}_{i,j} - \bar{\mathbf{v}})^t \mathbf{\Sigma}_v^{-1} (\mathbf{v}_{i,j} - \bar{\mathbf{v}}) \tag{12}$$

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, $C_{i^*,j^*} = \min_{(i,j) \in P} C_{i,j}$, where the set P is defined by

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

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. Figure 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:

$$p = \begin{cases} 1 - \frac{C^*}{\tau_{face}} & C^* \leq \tau_{face} \\ 0 & C^* > \tau_{face} \end{cases} \tag{13}$$

where $C^* = \min_i C_i$ where i indexes the remaining nodes. We have used the value $\tau_{face} = 40$ 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. Figure 5d shows the region which best fits the face behaviour after the merging process.

5 The *ViBE* Active Browsing Environment

Many approaches for content-based management of video databases have focused on query-by-example methods [58] 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 research [15, 41], 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 [15, 41].

In *ViBE*, we adapt this browsing approach to the problem of video databases [11]. The similarity pyramid is organized using the shot distance measure given in Equation 6. Our distance measure allowes 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 [60]. While a number of techniques have used relevance feedback in content-based image retrieval [61, 62, 63], we believe that the use of relevance feedback for browsing is a very different problem which requires a fundamentally different approach.

5.1   Similarity Pyramids for Video Browsing

The structure of a similarity pyramid is illustrated in Figure 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 quad-tree structure, and then mapping the quad-tree's clusters onto the 2-D grid at each level of the pyramid. The shots are clustered using the distance measure given in Equation 6 and an agglomerative method similar to the general algorithm described in Section 3.1. 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 to the flexible method developed by Lance and Williams [64]. The result of this clustering is a binary tree which is subsequently converted to a quad-tree, and then remapped to the pyramid structure in a way which best preserves the organization of the

database [65].

5.2 Browsing Interface

Figure 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 [6]. 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.

5.3 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

$$D(y, R) \leq T \tag{14}$$

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

$$D(y, R) = \min_{x \in R} d_{\boldsymbol{\theta}}(y, x) \tag{15}$$

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 [15] of the shot tree, and the weights K_{Seq}, K_{Shot}, K_{Motion}, and $K_{Semantic}$ which were defined in Section 3.

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 [60] 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 Equation 14. For example, if the relevance set contains 10 images and $D_{(10)}$ is used as the threshold, then on average $\frac{10}{11}$ or 91% of the desired shots will be retained in after the pruning. An important result proved in [60] 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.

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.

5.4   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 [60]. With this cost function,

we then 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 Equation 15 to rebuild the similarity pyramid. The cross-validation approach gives reliable performance independently of the database content and the specific choice of similarity measures [60].

6   Experimental Results

6.1   The Experimental Data Set

At this time *ViBE* contains more than 100 hours of MPEG-1 sequences, which were recorded from miscallenous television programs. The sequences have been digitized at a rate of 1.5 Mbits/sec in SIF format (352×240). 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 the 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 (5 sequences) which consist mostly of dialogue with little camera motion; talk shows (7 sequences), which contain a large number of dissolves and camera motion at the start of the programs; sports (5 sequences), containing miscallenous 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 (4 sequences); and movies (3 sequences) which contain sequences from the films *Before He Wakes* and *Lawrence of Arabia*; and CSPAN (5 sequences) obtained from CSPAN and CSPAN-2. 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 1.

6.2   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

    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

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 [27] 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, and

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 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 10 frames.

The results of these experiments are shown in Table 2. 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.

6.3 Pseudo-Semantic Classification

We have tested the face labeling algorithm described in Section 4 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 in the following way: 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 Equation 11 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 Figure 9, a bounding box is drawn in each image for regions with dissimilarity value, C, defined by Equation 11 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 Figures 9g and 9m-9p. Results for the face detection algorithm for a number of sequences are given in Table 3.

6.4 Active Browsing

Figures 10a-f demonstrate the significance of each component of the shot dissimilarity measure. Figures 10a-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. Figures 10a shows the result when only the D_{ST} distance is used to compute the the distance between shot trees. Figures 10b and 10c 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.

Figures 10d-f show the first 15 shots returned for a query using a local weather report shot (upper left hand shot) when searching using 7 sequences of video. Figure 10d shows the result when only using D_{ST} as the distance between shot trees. Figure 10e shows the result when the motion

distance, D_M, is added, and Figure 10f 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 Figure 11, we show the shots obtained using relevance feedback for similarity pyramid pruning and redesign using the original pyramid and relevance set shown in Figure 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. Figure 11b shows the reorganized sub-database where the dissimilarity function is defined by the optimized distance described in Section 5.4. Notice that the shots in the relevance 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.

7   Conclusion

In this paper, we have presented a new paradigm for video database management known as *ViBE*. *ViBE* introduces a variety of novel algorithms and techniques for processing, representing, and managing digital video in an integrated environment. We have introduced a new way of examining scene change detection through the use of the Generalized Trace and regression trees. We have described a powerful way of representing each shot using a binary tree and have discussed similarity measures that exploit information in the shot tree and the shot boundaries. 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] Alejandro Jaimes and Shih-Fu Chang, "A conceptual framework for indexing visual information at multiple levels," in *Proceedings of SPIE Conference on Internet Imaging*, San Jose, CA, January 2000, vol. 3964.

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

[3] Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong, "A fully automated content-based video search engine supporting spatio-temporal queries," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 602–615, September 1998.

[4] A. Hampapur, A. Gupta, B. Horowitz, C. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain, "Virage video engine," in *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, 13-14 February 1997, vol. 3022, pp. 188–197.

[5] Savitha Srinivasan, Dulce Ponceleon, Arnon Amir, and Dragutin Petkovic, "What is in that video anyway?: In search of better browsing," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 1999, pp. 388–392.

[6] Bob Davies, Rainer Lienhart, and Boon-Lock Yeo, "The video document," in *Proceedings of the SPIE Conference on Multimedia Storage and Archiving Systems IV*, 20-22 September 1999, vol. 3846, pp. 22–34.

[7] A. Hanjalic, R.L. Lagendijk, and J. Biemond, "DANCERS: Delft advanced news retrieval system," in *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2001*, San Jose, CA, January 2001.

[8] Howard D. Wactlar, "Informedia - search and summarization in the video medium," in *Proceedings of the IMAGINA 2000 Conference*, Monaco, January 31 - February 2 2000.

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

[10] Cuneyt Taskiran, Jau-Yuen Chen, Charles A. Bouman, and Edward J. Delp, "A compressed video database structured for active browsing and search," in *Proceedings of IEEE Int'l Conference on Image Processing*, Chicago, IL, October 4-7 1998.

[11] Jau-Yuen Chen, Cuneyt Taskiran, Alberto Albiol, Charles A. Bouman, and Edward J. Delp, "ViBE: A video indexing and browsing environment," in *Proceedings of the SPIE Conference on Multimedia Storage and Archiving Systems IV*, Boston, MA, September 1999, vol. 3846, pp. 148–164.

[12] Cuneyt Taskiran, Charles Bouman, and Edward J. Delp, "The ViBE video database system: An update and further studies," in *Proceedings of the SPIE/IS&T Conference on Storage and Retrieval for Media Databases 2000*, San Jose, CA, January 26-28 2000, pp. 199–207.

[13] Cuneyt Taskiran and Edward J. Delp, "Video scene change detection using the generalized trace," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, WA, May 12-15 1998.

[14] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.

[15] Jau-Yuen Chen, Charles A. Bouman, and John C. Dalton, "Similarity pyramids for browsing and organization of large image databases," in *Proceedings of SPIE Conference on Human Vision and Electionic Imaging III*, San Jose, CA, January 26-29 1998, vol. 3299.

[16] Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer, "Fast scene change detection using direct feature extraction from MPEG compressed videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 1–13, February 2000.

[17] Rainer Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, San Jose, CA, January 1999, vol. 3656, pp. 290–301.

[18] C. O'Toole, A. Smeaton, N. Murphy, and S. Marlow, "Evaluation of automatic shot boundary detection on a large video test suite," in *Proceedings of the 2nd UK Conference on Image Retrieval*, Newcastle, UK, 25-26 February 1999.

[19] John Boreczky and Lawrence Rowe, "Comparison of video shot boundary detection techniques," in *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases IV*, San Jose, CA, February 1996, vol. 2670, pp. 170–179.

[20] Apostolos Dailianas, Robert B. Allen, and Paul England, "Comparison of automatic video segmentation algorithms," in *Proceedings of SPIE Photonics East'95: Integration Issues in Large Commercial Media Delivery Systems*, Philadelphia, PA, October 1995, pp. 1–16.

[21] Edoardo Ardizzone and Marco La Cascia, "Multifeature image and video content-based storage and retrieval," in *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems*, Boston, MA, November 18-19 1996, vol. 2916, pp. 265–276.

[22] 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., pp. 113–127. Austin, TX, 1995.

[23] Behzad Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proceedings of SPIE Conference on Digital Video Compression: Algorithms and Technologies*, San Jose, CA, February 1995, vol. 2419, pp. 2–13.

[24] Nilesh V. Patel and Ishwar K. Sethi, "Video shot detection and characterization for video databases," *Pattern Recognition*, vol. 30, no. 4, pp. 583–592, April 1997.

[25] Wenyi Zhao, J. Wang, D. Bhat, K. Sakiewicz, N. Nandhakumar, and W. Chang, "Improving color-based video shot detection," in *IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, 7-11 June 1999, pp. 752–756.

[26] A. M. Ferman and A. M. Tekalp, "Efficient filtering and clustering methods for temporal video segmentation and visual summarization," *Journal of Visual Communicationa and Image Representation*, vol. 9, pp. 336–352, 1998.

[27] Boon-Lock Yeo and Bede Liu, "Rapid scene analysis on compressed video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, December 1995.

[28] Seong-Whan Lee, Young-Min Kim, and Sung Woo Choi, "Fast scene change detection using direct feature extraction from MPEG compressed videos," *IEEE Transactions on Multimedia*, vol. 2, no. 4, pp. 240–254, December 2000.

[29] Jongho Nang, Seungwook Hong, and Youngin Ihm, "An efficient video segmentation scheme for MPEG video stream using macroblock information," in *Proceedings of the 7th ACM International Multimedia Conference*, Orlando, FL, October 30 - 5 November 1999.

[30] Soo-Chang Pei and Yu-Zuong Chou, "Efficient MPEG compressed video analysis using macroblock type information," *IEEE Transactions on Multimedia*, vol. 1, no. 4, pp. 321–333, December 1999.

[31] Philippe Aigrain and Philippe Joly, "The automatic real-time analysis of film editing and transition effects and its applications," *Computation and Graphics*, vol. 18, no. 1, pp. 93–103, 1994.

[32] Nuno Vasconcelos and Andrew Lippman, "A bayesian video modeling framework for shot segmentation and content characterization," in *CVPR'97: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Purto Rico, October 4-7 1997.

[33] Milind R. Naphade and Thomas S. Huang, "Stochastic modeling of soundtrack for efficient segmentation and indexing of video," in *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2000*, San Jose, CA, January 2000, vol. 3972, pp. 168–176.

[34] John Boreczky and Lynn D. Wilcox, "A hidden markov model framework for video segmentation," in *Proceedings of IEEE Int'l Conference on Acoustic, Speech and Signal Processing*, Seattle, WA, May 1998, pp. 3741–3744.

[35] J. Huang, Z. Liu, and Y. Wang, "Joint video scene segmentation and classification based on Hidden Markov Model," in *IEEE International Conference on Multimedia and Expo (ICME 2000)*, New York, NY, July 30 - August 2 2000.

[36] Ke Shen and Edward J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proceedings of IEEE Int'l Conference on Image Processing*, Washington, D.C., October 26-29 1995, pp. 252–255.

[37] Saul Gelfand, C. Ravishankar, and Edward Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 163–174, February 1991.

[38] Yueting Zhuang, Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *Proceedings of IEEE Int'l Conference on Image Processing*, Chicago, IL, October 1998, pp. 866–870.

[39] Mark S. Drew and James Au, "Video keyframe production by efficient clustering of compressed chromaticity signatures," in *Proceedings of the Eighth ACM Multimedia Conference*, Los Angeles, CA, October 30 - November 4 2000, pp. 365–367.

[40] Konstantin Y. Kupeev and Zohar Sivan, "An algorithm for efficient segmentation and selection of representative frames in video sequences," in *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2001*, San Jose, CA, January 2001, vol. 4315, pp. 24–26.

[41] Jau-Yuen Chen, Charles A. Bouman, and John C. Dalton, "Hierarchical browsing and search of large image databases," *IEEE Trans. on Image Processing*, vol. 9, no. 3, pp. 442–455, March 2000.

[42] Jr. Joe H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.

[43] Milind R. Naphade, Minerva M. Yeung, and Boon-Lock Yeo, "A novel scheme for fast and efficient video sequence matching using compact signatures," in *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2000*, San Jose, CA, January 2000, vol. 3972, pp. 426–431.

[44] A. K. Jain, A. Vailaya, and W. Xiong, "Query by video clip," *Multimedia Systems*, vol. 7, pp. 369–384, 1999.

[45] Minerva M. Yeung, Boon-Lock Yeo, and Bede Liu, "Extracting story units from long programs for video browsing and navigation," in *IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 17-21 1996, pp. 296–305.

[46] Simone Santini and Ramesh Jain, "Beyond query by example," in *IEEE Second Workshop on Multimedia Signal Processing*, Redondo Beach, CA, December 1998, pp. 3–86.

[47] Arnold W. M. Smeulders, Marcel Worring, Simaon Santini, Amarnath Gupta, and Ramesh Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, December 2000.

[48] Niels Haering, Richard J. Qian, and M. Ibrahim Sezan, "A semantic event-detection approach and its application to detecting hunts in wildlife video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 857–868, September 2000.

[49] Milind R. Naphade and Thomas S. Huang, "A probabilistic framework for semantic indexing and retrieval in video," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, New York, NY, July 31-August 2 2000.

[50] Kin Choong Yow and Roberto Cipolla, "Feature-based human face detection," *Image and Vision Computing*, vol. 15, no. 9, pp. 713–735, 1997.

[51] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade, "Human face detection in visual scenes," *Advances in Neural Information Processing Systems*, vol. 8, pp. 875–881, 1996.

[52] Antonio J. Colmenarez and Thomas S. Huang, "Face detection with information-based maximum discrimination," in *Proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, PR, 1997, pp. 782–787.

[53] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions, merging and wavelet packet analysis," *IEEE. Transactions on multimedia*, vol. 1, no. 3, pp. 264–277, September 1999.

[54] Alberto Albiol, Charles A. Bouman, and Edward J. Delp, "Face detection for pseudo-semantic labeling in video databases," in *Proceedings of the IEEE International Conference on Image Processing*, Kobe, Japan, October 25-28 1999.

[55] Hualu Wang and Shih-Fu Chang, "A highly efficient system for automatic face recognition in MPEG video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615–628, August 1997.

[56] Ming-Hsuan Yang and Narendra Ahuja, "Detecting human faces in color images," in *Proceedings of IEEE Int'l Conference on Image Processing*, Chicago, IL, October 4-7 1998, pp. 127–130.

[57] S. Beucher and F. Meyer, ," in *Mathematical Morphology in Image Processing*, E.R. Dougherty, Ed., chapter 12. The morphological Approach to Segmentation: The Watershed Transformation, pp. 433–481. Marcel Dekker Inc., 1993.

[58] Di Zhong and Shih-Fu Chang, "Spatio-temporal video search using the object based video representation," in *Proceedings of IEEE Int'l Conference on Image Processing*, Santa Barbara, CA, October 1997, pp. 21–24.

[59] Jau-Yuen Chen, Cuneyt Taskiran, Edward J. Delp, and Charles A. Bouman, "ViBE: A new paradigm for video database browsing and search," in *IEEE Workshop on Content-Based Image and Video Databases*, Santa Barbara, CA, June 21 1998, pp. 96–100.

[60] Jau-Yuen Chen, Charles A. Bouman, and John Dalton, "Active browsing using similarity pyramids," in *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases VII*, San Jose, CA, January 24-29 1999, vol. 3656, pp. 144–154.

[61] Ingemar J. Cox, Matt L. Miller, Stephen M. Omohundro, and Peter N. Yianilos, "Pichunter: Bayesian relevance feedback for image retrieval," in *Proceedings of International Conference on Pattern Recognition*, Vienna, Austria, August 1996, vol. 3, pp. 361–369.

[62] Leonid Taycher, Marco La Cascia, and Stan Sclaroff, "Image digestion and relevance feedback in the Imagerover WWW search engine," in *Proceedings of International Conference on Visual Information*, San Diego, CA, December 15-17 1997.

[63] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Relevance feedback techniques in interactive content-based image retrieval," in *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases VI*, San Jose, CA, January 26-29 1998, vol. 3312, pp. 25–36.

[64] G. N. Lance and W.T. Williams, "A general theory of classificatory sorting strategies. i. hierarchical systems," *Computer Journal*, vol. 9, pp. 373–380, 5 1966.

[65] Jau-Yuen Chen, Charles A. Bouman, and Jan P. Allebach, "Fast image database search using tree-structured VQ," in *Proceedings of IEEE Int'l Conference on Image Processing*, Santa Barbara, CA, October 26-29 1997, vol. 2, pp. 827–830.
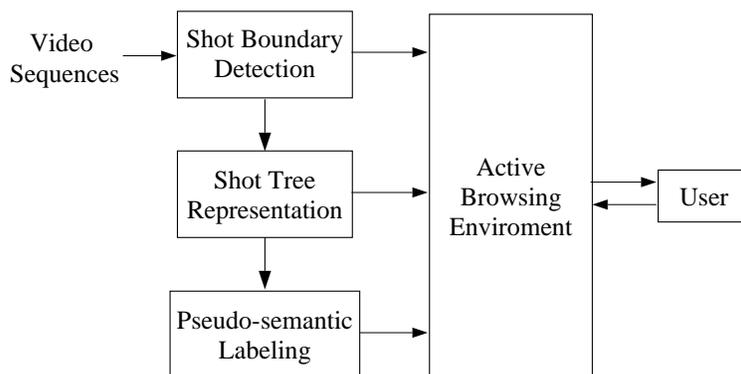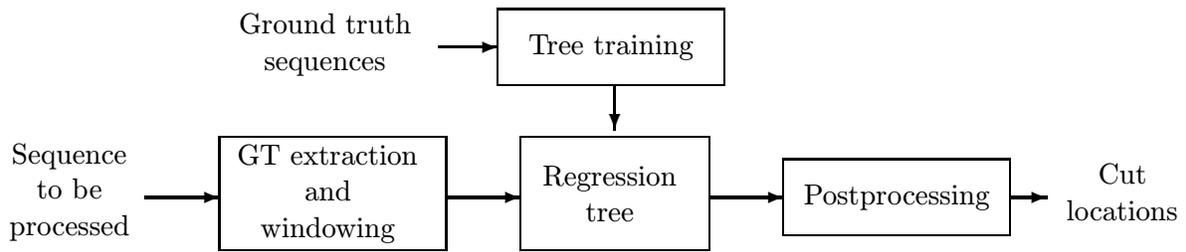
Figure 1: Major components of the *ViBE* system

Figure 2: Schematic diagram for cut detection.

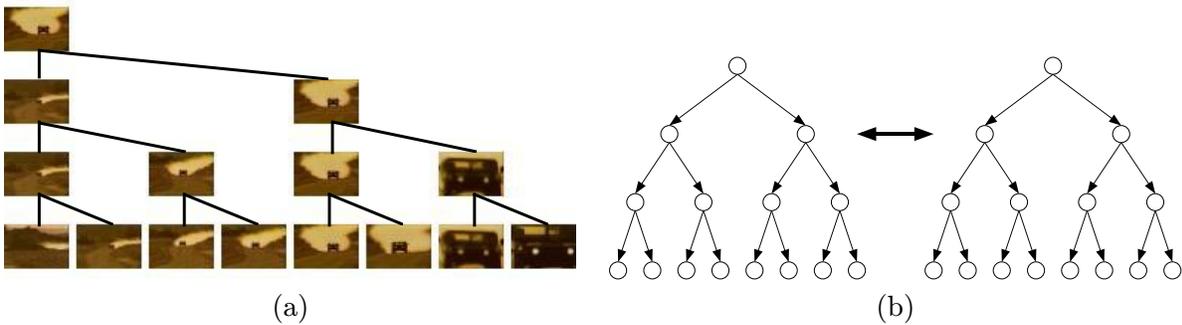(a)   (b)

Figure 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.
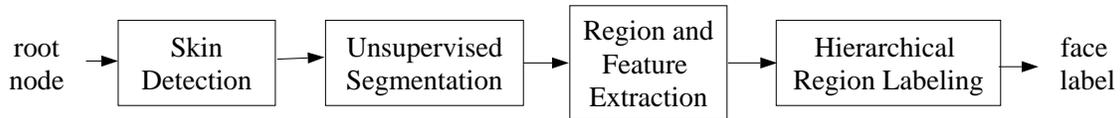

Figure 4: Block diagram illustrating the processing steps used to obtain the pseudo-semantic "face" label for the root node of each shot tree.

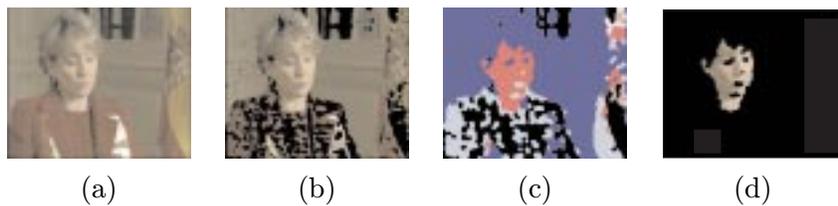(a)   (b)   (c)   (d)

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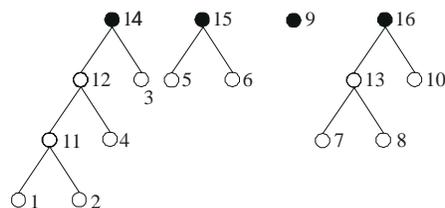

Figure 6: Binary tree structure resulting from skin-like region merging.

26

Figure 7: An example of a similarity pyramid and its embedded quad-tree.

Figure 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.

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 1: The number of different types of shot boundaries for different programe genres used. The mean shot length and standard deviation in frames is also given.

Figure 9: The boxes indicate the faces detected for $C \leq= 20$.

(a) D_{ST} (shot tree distance)

(d) D_{ST} (shot tree distance)

(b) $D_{ST} + D_M$ (a + motion distance)

(e) $D_{ST} + D_M$ (d + motion distance)

(c) $D_{ST} + D_M + D_T$ (b + temporal distance)

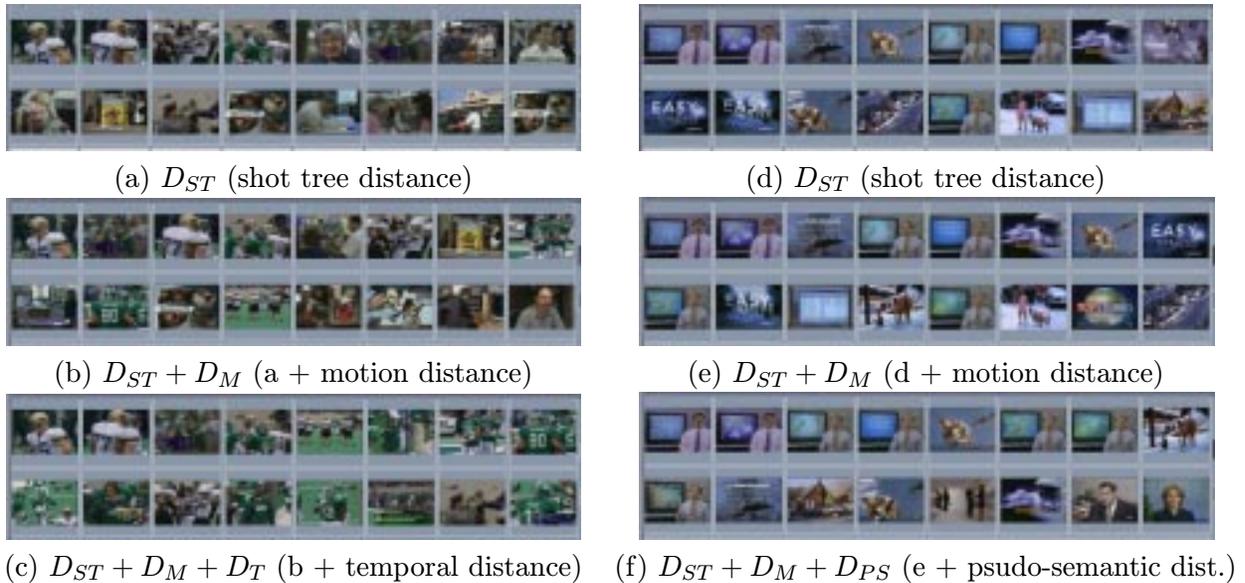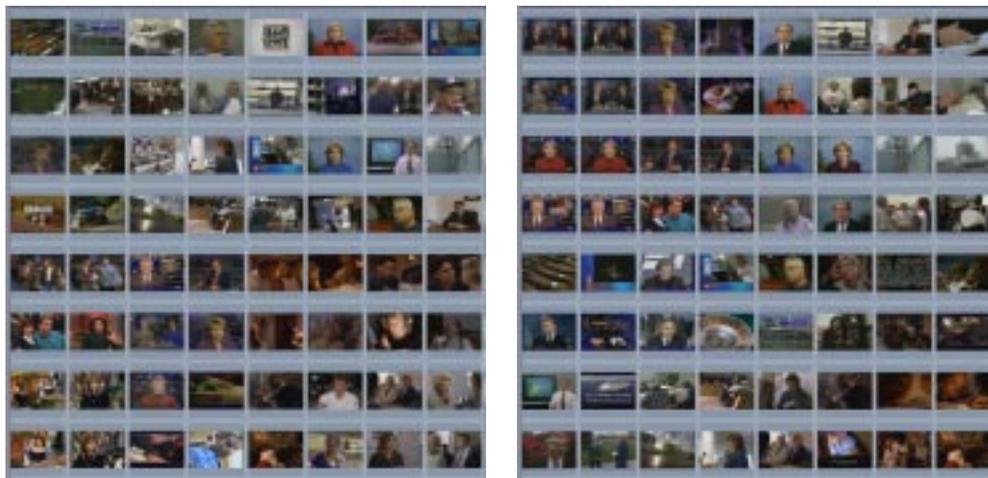(f) $D_{ST} + D_M + D_{PS}$ (e + psudo-semantic dist.)

Figure 10: Example of search results using different similarity measures.

(a) organized with default weightings   (b) organized with optimized distance

Figure 11: The pruning result of Figure 8. (a) The pyramid organized with the default weights (b) The pyramid organized using the optimized distance measure.

Genre	tree classifier				sliding window				simple thresholding			
	D	M	FA	MC	D	M	FA	MC	D	M	FA	MC
soap	0.941	0.059	13.3	0	0.916	0.084	99	0	0.852	0.145	24	0
talk	0.942	0.058	32.3	7.5	0.950	0.050	45	1	0.968	0.032	171	15
sports	0.939	0.051	82.5	34.8	0.785	0.215	59	1	0.925	0.075	251	73
news	0.958	0.042	38.0	0.75	0.886	0.114	61	0	0.926	0.074	212	1
movies	0.821	0.179	43.3	2	0.856	0.144	25	0	0.816	0.184	25	3
cspan	0.915	0.085	54.3	8.5	0.994	0.006	40	0	0.943	0.057	3	20

Table 2: Results for cut detection using the GT/tree classifier, the sliding window method, and simple thresholding. D and M indicate the average detection rate and missed detection, respectively, for each class, and FA is the total number of false alarms. MC gives the total number of misclassifies where the a shot boundary other than a cut was detected at a cut location.

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

Table 3: 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.