CERIAS Tech Report 2004-117

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

by J Chen, C Taskiran, A Albiol, E Delp, C Bouman
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

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

Jau-Yuen Chen, Cüneyt Taşkıran, Alberto Albiol†,
Edward J. Delp and Charles A. Bouman
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
{jauyuen,taskiran,ace,bouman}@ecn.purdue.edu
†Departamento de Comunicaciones
Universidad Politécnica de Valencia
Valencia, Spain
alalbiol@dcom.upv.es

Corresponding Author:
Professor Edward J. Delp
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285
USA
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: DTBS - Image and Video Databases

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

1 Introduction

Applications for digital video are undergoing explosive growth. While these applications will generate and use vast amounts of video data, the technologies for organizing and searching video data are in their infancy. To date, most research in video content management has focused on specific video processing problems which are essential components of any larger video database system. For example, temporal segmentation of video into shots [1, 2, 3], selection of representative keyframes for shots [4, 5], and design of shot similarity measures are problems which are relevant to the design of video database systems. More recently, there have been efforts to extract high level properties of video shots in an effort to better characterize and thereby manage the video. The motion trails features proposed in [6] are used to query video for specific object trajectories, whereas the motion features of [7] are used to label each shot using simple, but useful, categories. The objective of [8] was to use features such as motion and shot length to infer high level labels of the video sequence such as "comedy" and "action".

In separate but related work, a number of techniques have been proposed for browsing or summarizing video sequences. Shahraray and Gibbon exploited closed caption information to automatically generate pictorial transcripts from news sequences [9]. Yeung and Yeo [10] clustered shots into scene transition graphs or video posters, and Rui, Huang and Mehrotra [11] merged shots into scenes to automatically create a table-of-contents. In each case, the objective of this research is subtly different from a video database management, since the systems are designed to operate on individual sequences rather than large databases of video.

Early approaches to video database, such as QBIC, have used what are essentially image database query techniques to search the keyframes of video shots [12]. Zhang *et. al.* [13] proposed a video database system which used features derived from color, and motion. While they mainly focused on search by example, they also proposed a basic browsing capability based on a cluster-based hierarchy. Most recently, Rui, Huang and Mehrotra have proposed an approach to video database which tightly integrates browsing and query methods into a single framework [14].

In this paper, we present an integrated system for managing large databases of video which we call *ViBE* (video indexing and browsing environment) [15, 16]. The *ViBE* system introduces a variety of novel algorithms and techniques for processing, representing, and managing video while keeping the user in the loop. Perhaps the most important objective of this paper is to describe not only how these techniques function, but also how they integrate together into a single

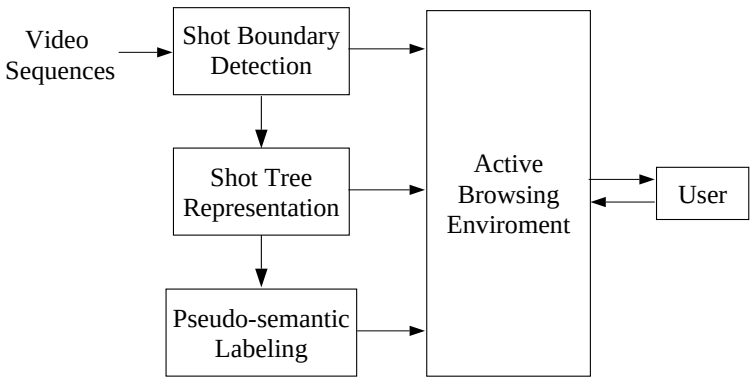


Figure 1: The *ViBE* system

system which can be scaled to large databases and extended to a wide variety of functionalities. Figure 1 illustrates the four major components of *ViBE*: shot boundary detection, hierarchical shot representation, pseudo-semantic shot labeling, and active browsing.

Shot boundary detection is the first step in processing the video data. Our shot boundary detection method incorporates two major innovations: the extraction of a high dimensional compressed-domain feature vector which we call the generalized trace (GT), and the use of a binary regression tree [17] 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 shot tree. The shot tree is a binary tree which contains a set of representative frames from the shot, with the keyframe being the root node of the tree. 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 similarity comparisons, to be obtained efficiently using tree matching algorithms.

Ideally, one would like to query a video database using very high-level semantic descriptions. Unfortunately, automatic labeling of such categories is currently impossible. 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. 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 [18]. 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. We will demonstrate the performance of *ViBE* using a database of MPEG sequences.

2 Shot Boundary Detection

A group of frames from a video sequence that have continuity in some sense is known as a *shot*. Often, a shot is composed of frames which depict the same scene, signify a single camera operation, or contain a distinct event or action. A *scene* is defined as a complete unit of narration which consists of a series of shots or a single shot that takes place in a location and that deals with a single action [19]. The first task in content-based video analysis is to segment shots by detecting and classifying shot boundaries. Although shot boundaries can take a wide variety of forms ranging from cuts to dissolves, fades, wipes, page turns, and special effects, in this paper we shall concentrate on the detection of *cuts* which are abrupt changes of content from one frame to another.

2.1 Previous Work

A large number of techniques have been reported in the literature for temporal segmentation. To detect cuts, some methods have used the difference of pixel values averaged over the entire frame as a similarity feature between frames [20]. Shahraray [1] has proposed dividing a frame into blocks and finding the "best" matching blocks between frames for comparison, similar to the block matching technique of MPEG. Yeo and Liu [21] use the pixel differences of the luminance component of DC frames in an MPEG sequence. The fact remains that the simple frame differencing methods are susceptible to intensity differences caused by motion, illumination changes, and noise.

Other methods have been proposed to address the above problems based on the the use of color histograms of the frames. One approach is to use a test statistic derived from the histograms to determine their similarity. Patel and Sethi [3] have experimented with various statistics and have found that the χ^2 statistic gives the best performance. They use the intensity histograms obtained for the entire frame. The histograms are found using DC coefficients of MPEG video for only

I frames. Tekalp and others [22, 23] use the sum of histogram differences for the Y, U, and V components. Two-class clustering is then used to determine the cut locations.

Idris and Panchanathan [24] use vector quantization to compress a video sequences using a codebook of size 256 and 64-dimensional vectors. The histogram of the labels obtained from the codebook for each frame is used as a frame similarity measure and a χ^2 statistic is used to detect cuts. Using the observation that during a shot transition the locations of appearing edge pixels would be very different from old edge pixel locations, Zabih *et. al.* [2] have proposed a cut detection scheme based on edge detection. Shen *et. al.* [25] have applied this technique to MPEG sequences using multi-level Hausdorff distance histograms. Meng *et. al.* [26] define various ratios of the number of macroblocks to perform cut detection for P and B frames of a MPEG sequence.

A number of researchers have investigated the merits of a video model in the detection of shot boundaries. Aigrain and Joly [27] develop a model for the probability density function of intershot pixel differences in successive frames. They then detect shot transitions by fitting this model to pixel difference data obtained from the sequence. Vasconcelos and Lippman [8] have modeled the time duration between two shot boundaries. Using a Bayesian model and the Weibull prior distribution, they have derived a variable threshold to detect shot boundaries. Knowledge-based approaches include the work of Zhang *et. al.* [28] where anchor person shots are found by examining intrashot temporal variation of frames. Swanberg *et. al.* [29] used a similar approach by template matching.

2.2 The Generalized Trace

Most of the techniques in the literature detect shot boundaries by extracting some form of one-dimensional frame similarity feature from the video data. Usually, this similarity feature is then thresholded using a fixed global threshold. This approach has a number of problems. First, it is difficult to determine a single frame similarity feature which can be used to accurately detect shot boundaries in a wide variety of situations. Second, there is no single threshold value which is appropriate for all video sequences.

In *ViBE*, shot boundary detection is performed by first extracting a set of features from each DC frame. These features are placed in a high dimensional feature vector that we call the *generalized trace* (GT) [31]. The GT is then used in a binary regression tree to determine the probability that each frame is a shot boundary. These probabilities can then be used to detect frames corresponding to shot boundaries. In this paper, we will describe the detection of cut locations. However, we feel that the paradigm of the GT/regression tree can be extended for the detection and classification

of other types of shot boundaries [15].

Our method has a number of advantages. First, the GT feature vector allows a multitude of very different features to be collectively used to detect shot boundaries. This is important since different features may be useful in different shot boundary scenerios. Second, since the regression tree generates probabilities, the detection can be made without requiring the selection of arbitrary thresholds. Moreover, this method is also highly extensible. For example, if we find new features that work well in detecting shot transitions, we can easily incorporate them. In addition, the multidimensional classifier provides a unified framework to investigate various kinds of shot boundaries as compared with trying to derive a different detection scheme for each type of shot boundary.

Our method uses the "DC sequence" extracted from the compressed video sequence. The DC sequence is formed from the DC coefficients of the DCT used in MPEG. While the DC coefficients are directly available for I frames, they must be estimated for P and B frames. We have used the method described in [30] for estimating these DC coefficients.

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

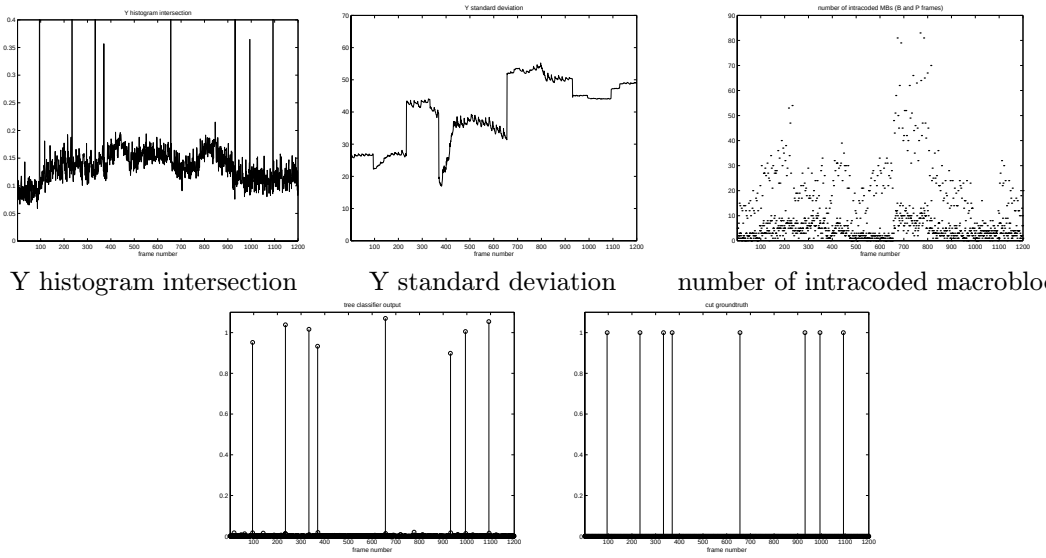
$g_{i,1-3}$ - Histogram intersection [32] of frames i and $i-1$ for the Y, U, and V color components.
$g_{i,4-6}$ - Standard deviation of the Y, U, and V color components for frame i.
$g_{i,7}$ - Number of intracoded macroblocks in frame i.
$g_{i,8}$ - Number of forward predicted macroblocks in frame i.
$g_{i,9}$ - Number of backward predicted macroblocks in frame i.
$g_{i,10-12}$ - Flags which identify the frame type $\{I$, P, or $B\}$ for frame i.

Figure 2 illustrates how some of the components of the GT vary with frame number for a typical sequence. The ground truth information indicates where actual cuts are located.

2.3 Binary Regression Tree

To detect whether a cut has occurred between frames $i-1$ and i, we use the GT and a binary regression tree [17] to estimate the probability of a shot boundary. The binary regression tree provides a piecewise linear approximation to the conditional mean, i.e.,

$$y_i \approx E[\alpha_i | \vec{g}_{i-w} \cdots \vec{g}_i \cdots \vec{g}_{i+w}] \tag{1}$$



Y histogram intersection Y standard deviation number of intracoded macroblocks

Output of the regression tree Ground truth

Figure 2: Examples of features used in the GT, the ground truth for cut locations, and the output of the binary regression tree.

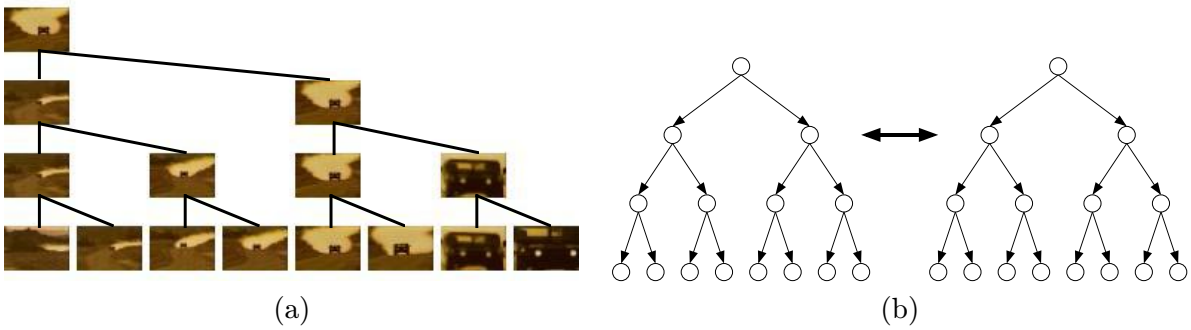
where α_i is the cut indicator function

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

and where y_i is the output of the tree classifier for the i^{th} frame and w controls the window size. The output y_i can be interpreted to be the probability of a cut occurring at frame i [17]. It should be noted that in general the classifier uses more than just $\vec{g}_i$ to determine if a shot boundary has occurred. Our experiments have shown that the use of $\vec{g}_{i-1}$, $\vec{g}_i$ and $\vec{g}_{i+1}$ $(w = 1)$ provides a reasonable balance between complexity and performance.

Cuts are then detected by using the threshold $y_i > Thresh$ where we typically chose $Thresh = 0.20$. This approach is more robust than thresholding the value of the features, as is used in many other shot boundary detection techniques, because the classifier chooses the best features from the GT, and because the threshold is not dependent on the specific video sequence. The detected cut locations are then post-processed to remove cuts that are too close together. In the experiments described in Section 6.2, if two cuts are closer than ten frames, the cut with a smaller y_i is deleted.

The regression tree used in *ViBE* is a variation of the technique proposed in [33]. The difference is that the training and pruning step is used only once. The training process uses two sequences with known shot boundary locations, we shall refer to these sequences as ground truth sequences. One of these sequences is used to build a large tree and the other sequence is then used to prune



(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.

the tree. The growing stage starts with a single terminal node and at each step the terminal node that yields the greatest reduction in the classification error is split. In this way, we produce a tree that overfits the data [33]. The classification tree 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 tree-based approach has a number of advantages when compared to more traditional non-parametric methods such as nearest neighbor or kernel estimator approaches. The regression tree has a simple form which can be compactly stored, and it efficiently classifies data. It also does automatic feature selection and complexity reduction [33].

3 Shot Representation

After the video sequence is segmented into shots, usually a frame is chosen to represent each shot. This is usually known as a keyframe. The objective is to represent each shot with one or several frames that can provide a way to manage the video content. The effectiveness of this approach will then depend on the algorithm used for the selection of keyframes. An easy and straightforward approach is to select one keyframe per shot, usually the n^{th} frame for a fixed number n. To capture the concept of a shot with a great deal of motion, methods for sequentially selecting multiple keyframes have also been proposed based on frame differencing [21, 34]. Other methods for selecting multiple keyframes include the use of motion information [4], and clustering approaches [5, 22].

3.1 Shot Representation Using the Shot Tree

This section describes a novel way to represent a shot based on a tree structure, known as a shot tree, formed by clustering the frames in a shot. The tree structure allows important operations, such as 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 representative groups. For example, if one wants to categorize the shot by one frame, the root node is used. If one wanted to categorize the shot by three frames, the first two levels of the tree are used (Figure 3a). This tree representation is obtained through agglomerative clustering of the shot's frames. We use the color, texture and edge histograms proposed in [18] as the feature vector for each frame in a shot, and the L_1 norm to measure feature distances. The depth of the tree will depend on the application. For browsing, the root node can be used as a keyframe; for similarity measure and classification in the browsing environment, two or three levels of the tree can be used.

Bottom-up (agglomerative) clustering algorithms work by first forming a complete matrix of the distances between the frames and then using this matrix to sequentially group frames[35, 36]. 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 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 is set to 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.

The general algorithm for agglomerative clustering has the following form:

1. $\Psi \leftarrow \{0, 1, \cdots, N-1\}$
2. For each $(i,j) \in \Psi^2$ compute $d_{ij} \leftarrow d(f_i, f_j)$
3. For $k = N$ to $2N-2$ {
   (a) $(i^*, j^*) = \arg \min_{(i,j)\in\Psi^2} d_{ij}$
   (b) Set $c_k \leftarrow c_{i^*} \cup c_{j^*}$ and $n_k \leftarrow n_{i^*} + n_{j^*}$
   (c) $\Psi \leftarrow \{\Psi - \{i^*\} - \{j^*\}\} \cup \{k\}$
   (d) For each $h \in \Psi - \{k\}$ compute d_{hk}
   }

The specific type of agglomerative clustering is defined by the choice of the distance function d_{hk} in step 3.d.

Among agglomerative algorithms, the complete link algorithm has been used to organize keyframes [37], while the flexible algorithm has been proposed in organizing image databases [18]. In *ViBE*, we use Ward's clustering algorithm, also known as the minimum variance method [38], for building the shot tree. This clustering algorithm uses the distance function:

$$d_{hk} = \frac{n_i + n_h}{n_i + n_j + n_h} d_{hi} + \frac{n_j + n_h}{n_i + n_j + n_h} d_{hj} - \frac{n_h}{n_i + n_j + n_h} d_{ij} \tag{2}$$

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 similar groups, therefore allowing automatic detection of subgroups inside a shot.

3.2 Shot Similarity

In *ViBE*, the distance between two shots is a sum of distances which depend on the shot tree, the temporal information, the motion information, and the pseudo-semantic labels. Assume all video sequences in the database are assigned a unique identification number, S_i. Also assume that the j^{th} shot in sequence S_i is denoted as s_{ij}. Then the distance between two shots is given by

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

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

The shot tree dissimilarity, $D_{ST}(s_{ij}, s_{kl})$, 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

set of frames from the shot and a feature vector, z_t, for the cluster of frames. Generally, z_t is the centroid of the feature vectors in the cluster. The distance between the shot trees is then given by the weighted sum of 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.

A shot is more than just a set of frames. The temporal relationship among shots is another useful yet often ignored feature. Yeung, Yeo and Liu [39] has formulated a time-constrained temporal distance. Let b_{ij} and e_{ij} be the frame number of the beginning and ending frame of shot s_{ij}.

We define the "temporal distance" between shots s_{ij} and s_{kl} as the following

$$D_T(s_{ij}, s_{kl}) = \begin{cases} K_{Seq} + K_{Shot}, & \text{if } i \neq k \\ K_{Shot}(\min(\frac{1}{2}, \frac{\min(|b_{ij}-e_{kl}|, |b_{kl}-e_{ij}|)}{2T_f}) + \min(\frac{1}{2}, \frac{|j-l|}{2T_s})), & \text{if } i = k \end{cases} \quad (3)$$

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 shall use the values of $T_f = 3000$ and $T_s = 30$. Notice that 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.

Another feature that we exploit is the number of forward, backward and bidirectionally predicted macroblocks in each P or B frame. For each P or B frame, we first compute

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

for each frame k. We next obtain the histogram $h_{i,j}$ of the values m_k for all the frames k in the shot $s_{i,j}$. We then define the "motion distance" between shots s_{ij} and s_{kl} to be the L_1 norm of the difference between the histograms.

$$D_M(s_{ij}, s_{kl}) = K_{Motion} \| h_{ij} - h_{kl} \|_1$$

The constant K_{Motion} controls the weighting of this component.

The last but perhaps the most important features are the pseudo-semantic labels we will discuss in Section 4. For each shot, we define a label vector, p_{ij}, where each element of the vector takes on continuous values in the range [0,1] indicating the confidence level for the corresponding pseudo-semantic class. The "semantic distance" between shots s_{ij} and s_{kl} is then defined to be the L_1 norm of the difference of these feature vectors p_{ij} and p_{kl}.

$$D_S(s_{ij}, s_{kl}) = K_{Semantic} \| p_{ij} - p_{kl} \|_1$$

The constant $K_{Semantic}$ controls the weighting. We shall describe how these similarity measure are used for browsing in Section 5.

4 Pseudo-Semantic Labeling

The purpose of semantic labeling is to classify or label each frame in a video sequence using a high level semantic description. True semantic labels such as "young girl running", "blue dress", and "park scene" characterizes a scene based on its content. Ideally, such semantic labels might provide the most useful descriptions for indexing and searching databases. Currently, however, automatic extraction of truly semantic features is not possible.

Pseudo-semantic labeling bridges the gap between low-level and truly semantic labels. We are investigating in *ViBE* the use of several pseudo-semantic labels. These include labels such as "face," "indoor/outdoor," "high action," "man made," and "natural." 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 video frame has the associated property with high confidence, and 0 indicates that it does not.

In this paper, we will describe our work on the "face" label. The goal here is to detect whether a frame in the sequence contains a face. Different approaches have been developed in recent years for the face detection problem. Some of the most representative work includes shape-feature approaches [40, 41, 42]. In [43], a hierarchical knowledge-based pattern recognition method is presented. Neural network approaches are used in [44, 45, 46], and template matching approaches are used in [47, 48, 49]. These approaches have tended to focus on still grayscale 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 still 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 [50, 51, 52] both color and spatial information is exploited to detect segmented regions corresponding to faces. However, in [51, 52] the color and spatial information is also used to merge image regions in the segmentation process.

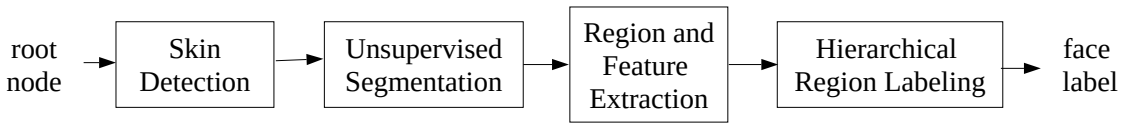


Figure 4: Block diagram illustrating the processing steps used to obtain the pseudo-semantic "face" label for the root node of each shot tree.

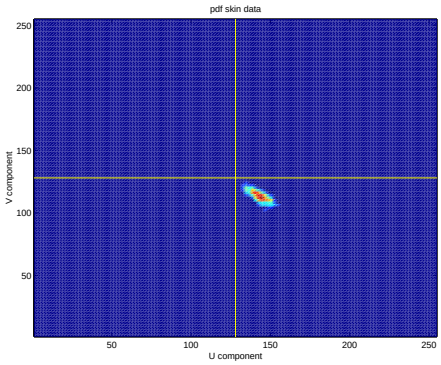


Figure 5: Projection on the UV plane of the histogram of the skin pixels.

4.1 Overview of Face Labeling

The following sections present the details of our approach for determining the face label of a frame. 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 information we have 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. Furthermore, we believe that the general processing strategy is extensible to a wider variety of pseudo-semantic categories.

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 root node for each shot tree. More generally, we could label each frame contained in the shot tree. We should note that all processing is done on full frames extracted from the MPEG data.

Figure 4 illustrates the processing steps that we use to obtain the face label for a given frame. The first step, skin detection, is used to segment regions of the image which potentially correspond to face regions based on pixel color. Figure 5 shows the histogram of a representative set of skin pixels in the UV plane. Notice that the skin tones occupy a relatively small portion of the color space. This property allows us to eliminate many regions from consideration based on color.

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. For example, 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 would like to consider all possible mergings of these initial regions to find the new composite region which best fits our hypothesis for the behavior of face areas. Of course, a search of all possible merges is impractical even for a moderate number of initial regions. Hence we instead use a pair-wise steepest descent method in which all pairs of merged regions are considered to find the merged region which best fits the hypothesis. This pair-wise merging process is recursively performed and results in a single composite region which best fits the face hypothesis.

Figure 6 illustrates the effect of these four processing steps on three example images. The first column shows three original images, each containing a face. The second column shows the result of skin detection with the white pixels representing the regions that have been classified to be skin. The third column is the result of unsupervised segmentation of the skin detected pixels into homogeneous color regions. The fourth column shows the single merged region which best fits the face hypothesis.

4.2 Skin Detection and Unsupervised Segmentation

The objective of skin detection is to extract skin-like regions from the image. This is a challenging problem because in video databases we must deal with substantial variations in illumination conditions and skin-types. We use the YUV luminance-chrominance space to detect the color of skin pixels. This has the advantage of avoiding any color transformation since the YUV color space is also used in the MPEG standard.

The skin detection works by modeling the skin colors using a Gaussian mixture distribution similar to that of [53], and then segmenting the image with a multiscale Bayesian segmentation algorithm known as sequential maximum a posteriori (SMAP) [54]. The Gaussian mixture density models skin pixels with a multimodal distribution. This is important since it allows for a wide variety of possible skin colors or illumination conditions. We use the expectation maximization

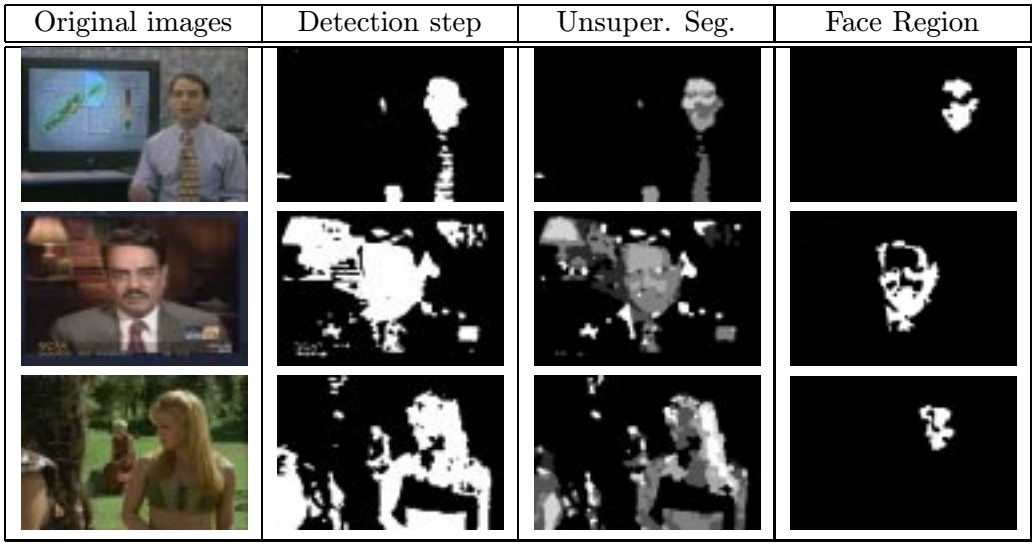


Figure 6: Results of each of the four steps used in obtaining the face label.

(EM) algorithm to estimate the parameters of the Gaussian mixture distribution from a training set of skin pixels [55, 56]. The training set is obtained from more than 200 images randomly selected from our database which were manually segmented into skin and no skin regions.

The next step is unsupervised segmentation of the detected skin pixels into homogeneous regions. The objective of this step is to separate spurious regions from the true face areas. To do this, we again use the EM algorithm to cluster the detected skin pixels using a multivariate Gaussian mixture distribution. Each component of the Gaussian mixture is then treated as a distinct subclass of the skin color space. The number of subclasses is set to be

$$\text{number of subclasses} = 5 + (\#\text{ of detected pixels})/10^4 \ .$$

This insures that detected skin pixels are split into a sufficiently large number of regions. Each pixel of the detected skin region is then individually classified to its highest probability subclass (i.e. the MAP classification), and the resulting segmentation is smoothed using a morphological opening and closing operation with a 5×5 circular kernel. This morphological smoothing eliminates many small spurious regions produced by the MAP classification. As we can see in the third column of Figure 6, the unsupervised segmentation process further partitions the skin detected regions into smaller more homogeneous regions.

4.3 Region Extraction and Hierarchical Region Labeling

The next step is to extract connected regions and features vectors which characterize those regions. For each connected region, we compute a set of features which we call a state vector that describes the color and shape of the region. Let Ω_i be the i^{th} region of pixels, and let x_r be a color pixel in YUV coordinates at position $r \in \Omega_i$. Here $r \in \Omega_i$ is assumed to be a two dimensional column vector $r = [r_1, r_2]^t$ where r_1 and r_2 index the row and column of the pixel. For each region, we will extract a state vector of information defined by $q_i = [N_i, \bar{x}_i, \gamma_i, \mu_i, R_i, max_i, min_i]$. Each feature of q_i describes a characteristic of the color or shape of the region. The elements of the state vector are given by the following expressions:

$$
\begin{aligned}
N_i &= \sum_{r \in \Omega_i} 1 \\
\bar{x}_i &= \tfrac{1}{N_i} \sum_{r \in \Omega_i} x_r \qquad\qquad \gamma_i = \text{Diag}\left\{ \tfrac{1}{N_i} \sum_{r \in \Omega_i} (x_r - \bar{x}_i)(x_r - \bar{x}_i)^t \right\} \\
\mu_i &= \tfrac{1}{N_i} \sum_{r \in \Omega_i} r \qquad\qquad R_i = \tfrac{1}{N_i} \sum_{r \in \Omega_i} (r - \mu_i)(r - \mu_i)^t \\
max_i &= (\max_{r \in \Omega_i} r_1, \max_{r \in \Omega_i} r_2) \quad min_i = (\min_{r \in \Omega_i} r_1, \min_{r \in \Omega_i} r_2)
\end{aligned}
$$

where γ_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 recursively updated for merged regions. For example, let Ω_i and Ω_j be two regions that are merged to form a new region Ω_h. The new state vector for the region h may be expressed as $q_h = U(q_i, q_j)$ where $U(\cdot, \cdot)$ is a function of the two original state vectors.

From the state vectors q_i, we obtain a feature vector v_i which contains the specific features we will use for labeling regions. The elements of v_i are $\bar{x}_i$, γ_i, μ_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 R_i. Let $\lambda_1 > \lambda_2$ be the two eigenvalues of R_i with corresponding eigenvectors e_1 and e_2. Then the three remaining features are $\sqrt{\lambda_1 \lambda_2}/N_i$, the orientation of e_1, and λ_2/λ_1.

A multivariate Gaussian distribution is used to model the behavior of v_i for face areas. Let $\bar{v}$ and Σ_v be the mean and covariance of the vector v_i for regions corresponding to face areas. Then the dissimilarity between v_i and the mean behavior for a face region is given by:

$$C_i = (v_i - \bar{v})^t \Sigma_v^{-1} (v_i - \bar{v}) \tag{4}$$

The smaller the value of C_i the greater the likelihood that the region is a face area. In our

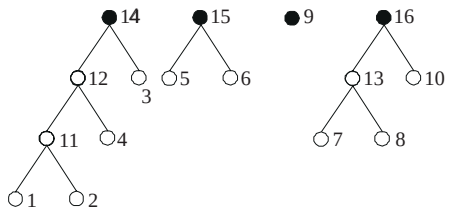


Figure 7: Binary tree structure resulting from skin-like region merging.

experiments, $\bar{v}$ and Σ_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 pair-wise merging of regions to find the new region which minimizes (4). We first form a matrix $C_{i,j}$ defined by

$$C_{i,j} = (v_{i,j} - \bar{v})^t \Sigma_v^{-1} (v_{i,j} - \bar{v}) \tag{5}$$

where $v_{i,j}$ 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 P is the set of entries of the matrix defined by

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

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. Once the two distinct regions Ω_{i^*} and Ω_{j^*} are identified, these two regions are merged to form a new region $\Omega_h = \Omega_{i^*} \cup \Omega_{j^*}$. After merging, the original regions Ω_{i^*} and Ω_{j^*} must be removed from the matrix and replaced by the new region Ω_h, and the entries of $C_{i,j}$ must be updated.

This process of merging is repeated until the set P is empty. At this point, the merging of any two regions will only reduce the quality of the match to the face hypothesis. Figure 7 illustrates how this recursive merging process progresses. 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. Any of these nodes which contains less then 600 pixels are removed, and the region that best fits the face hypothesis is used to compute the face label. This is done using the equation:

$$p = \begin{cases} -\frac{C^*}{40} + 1 & C^* < 40 \\ 0 & C^* > 40 \end{cases} \tag{6}$$

where $C^* = \min_i C_i$ where i indexes the remaining nodes. 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.

5 The *ViBE* Active Browsing Environment

Many approaches for content-based management of video databases have focused on query-by-example methods [57] 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 [18, 58], 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 [18, 58].

In *ViBE*, we adapt this browsing approach to the problem of video databases [16, 15]. The pyramid is organized using the shot dissimilarity measure described in Section 3. Our dissimilarity measure allowed weights to be adjusted which control the relative importance of color, edges, motion, temporal position, and pseudo-semantic labels. To exploit this flexibility, our 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 or pseudo-semantic labels. To do 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 similarity measure. However, it can also be through relevance feedback mechanisms provided in the user interface of the browsing environment [60, 61]. While a number of techniques have used relevance feedback in content-based image retrieval [62, 63, 64], 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 8. 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

Figure 8: An example of a similarity pyramid and its embedded quad-tree.

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 metric of Section 3, and an agglomerative method similar to the general algorithm described in Section 3. However, our clustering method is adapted to use a sparse proximity matrix. For a database with N shots, we only compute the distance between each shot and its M closest matches [59], and then use a clustering technique similar to the flexible method developed by Lance and Williams [65]. 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.

5.2 Browsing Interface

Figure 9 shows the browsing environment presented to the user. The similarity pyramid is shown to the left, and the set of relevant shots, which we call the relevance set, is shown to the right. For a large database, even upper levels of the pyramid will be too large to display on a single screen. Therefore, the user can move along the horizontal or vertical directions in a panning motion to search for image clusters of interest. If a specific cluster appears to be of interest, then the user can choose to move down to the next level by "clicking" on a cluster. 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.

Figure 9: Active Browsing Environment: The top level of the similarity pyramid is shown to the left, and the set of relevant shots (relevance set) is shown to the right. The users can incrementally add or remove shots from the relevance set as they browse through the database.

The relevance set is a set of shots that the user selects as they move through the pyramid. The user can incrementally add or remove shots 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. 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 becomes a buffer or clip board which stores all the shots of interest to the user. Once users have defined a set of relevant shots, they may associate the relevance set with a semantic label. These relevance sets may then be stored and recalled based on the semantic label.

5.3 Pruning and Reorganization

We incorporate relevance feedback into two basic browsing functions: pruning and reorganization. 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, and reorganization changes the structure of the similarity pyramid to facilitate the user's search. In order to perform these basic pruning and reorganization functions, we use statistical estimation techniques based on cross-validation. In [60, 61], we showed that the cross-validation approach gives reliable performance independently of the database content and the specific choice of similarity measures.

The 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 because the similarity pyramid structure allows the user to efficiently browse the resulting shots.

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. To do this, we will assume that the distance function $D_\theta(y, x)$ is parameterized by a vector θ. In this work, θ is a 13 component vector containing the 9 weightings corresponding to the L, a and b components of the color, edge and texture histogram features [18] of the shot tree, and K_{Seq}, K_{Shot}, K_{Motion}, and $K_{Semantic}$ described in Section 3.

Our objective is to find the parameter $\hat{\theta}$ which minimizes a cost function $E(\theta, R)$ that measures the separability of the relevance set R from the rest of the database. The detailed form of $E(\theta, R)$ is given in [60, 61]. With this cost function, we then compute the optimal value of the parameter $\hat{\theta} = \arg\min_\theta E(\theta, R)$. This minimization is done using conjugate gradient optimization. The resulting distance function $D_{\hat{\theta}}(y, x)$ is then used to rebuild the similarity pyramid.

6 Experimental Results

6.1 The Experimental Data Set

At this time, *ViBE* consists of 23 MPEG-1 sequences obtained from several standard broadcast television channels. All sequences used in our database are copyright cleared for use in *ViBE*. These sequences are 10 minutes in length and have been digitized at a rate of 1.5 Mbytes/sec in QCIF format (352×240). The data set used therefore contains nearly four hours of video with more than 400,000 frames. We have obtained ground truth for the shot boundary locations and semantic content for seven sequences in the data set. These ground truth sequences were used for training purposes.

	tree classifier				*sliding window*				*simple thresholding*			
Sequence	Detect	Miss	FA	MC	Detect	Miss	FA	MC	Detect	Miss	FA	MC
news1	171	15	18	7	145	41	10	3	148	38	31	10
news2	46	1	0	0	44	3	1	0	41	6	2	1
news3	40	5	4	0	39	6	1	0	30	15	9	1
news4	65	0	4	3	65	0	4	0	59	6	10	8
news5	128	19	11	6	108	39	17	2	105	42	18	11
movie	121	8	4	0	114	15	1	0	110	19	7	2
drama	94	5	1	1	85	14	3	1	56	43	2	1
Total	665	53	42	17	600	118	37	6	549	169	79	34

Table 1: Results for cut detection using the GT/tree classifier, the sliding window method, and simple thresholding. Detect indicates a correct detection, Miss indicates a miss detection, FA indicates a false alarm and MC indicates a missclassify.

6.2 Scene Change Detection

In the results presented here, the version of the GT/regression tree implemented consists of using the three vectors, $\vec{g}_{i-1}$, $\vec{g}_i$ and $\vec{g}_{i+1}$, for classification. This means that the regression tree uses 36 features. This "windowing" approach has shown to be robust. For all results of our method, we have used a threshold of $y_i > 0.20$ as our detection criteria.

We have compared our method with two other methods. The first comparison method uses a global threshold on the luminance histogram intersection of the frames, i.e, the $g_{i,1}$ component of the GT. Again, if two shots occur closer than 10 frames, the one having a smaller histogram intersection is deleted. The global threshold was chosen experimentally to provide the best performance. We have also implemented a sliding window technique, similar to the one proposed by Yeo and Liu [21], using 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. In this technique, a symmetric window of size $2M + 1$ is placed around the i^{th} frame and a cut is declared from frame $i - 1$ to i if

1. the value of the similarity feature for i is the maximum within the window, and
2. it is also N times the value of the second maximum in the window.

In our experiments we have used $M = 5$ and $N = 1.8$ because these values gave the best over all performance on our data sets.

We used our seven ground truth sequences to examine the performance of the three methods. The results are shown in Table 1. The table shows the number of correct detections (Detect), the number of locations that were missed (Miss), the number of locations that were falsely classified

Figure 10: Face images used in training the pseudo-semantic classifier. The manually segmented face masks are also shown.

as cuts (FA). Since we have ground truth information, we also determined the number of locations that the detection methods labeled as cuts but were actually other types of shot transitions such as dissolves. This is labeled as MC in the table. Our experiment indicates that the GT/regression tree in general performs better for a wide variety of video content.

6.3 Pseudo-Semantic Classification

The algorithm described in Section 4 was tested using the ground truth sequences. The root nodes of the shot trees were extracted for each shot and then ground truth information relative to "face" and "no face" was determined. Finally, the root nodes were extracted from the MPEG sequence at full frame resolution. These images were then used for classification. The algorithm was trained using "face masks" obtained from manually segmented faces from 100 randomly chosen frames that contained faces. A representative set of face masks is shown in Figure 10. The results were evaluated in the following way:

- A False Alarm results if the frame contains no faces, but our algorithm says that there is at least one face.
- A Detection results if the frame contains at least one face, and our algorithm says that there is at least one face.
- A Correct Classification results when a Detection is achieved or when there is no face in the frame and our algorithm finds no face in the frame.

Once we have merged skin-like regions, we apply a threshold to the dissimilarity values given by Equation 4 in the remaining regions. If there exist any value below the threshold, the frame is said to contain at least one face. Thus an optimal threshold can be obtained that maximizes the Classification rate.

Figure 11: The boxes indicate the faces detected (threshold $D_{max} = 20$).

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 only in this section to show the performance of the face labeling stage.

In Figure 11, a bounding box is drawn in each image for regions whose dissimilarity value is less than 20. As shown, we are able to detect faces under many different illumination conditions and poses. In Figure 11g, we can see that two boxes are drawn. In the classical face-detection problem this would have been classified as a false alarm. In our application, we do not care about such false alarms because our objective is to determine whether a face is presented or not. Figures 11m, 11n, 11o and 11p show false alarms where faces are detected in frames containing no human faces. In Table 2, the results for each ground truth sequence are shown separately.

6.4 Active Browsing

Figure 12 and Figure 13 demonstrate the significance of each component of the shot dissimilarity measure. Figure 12 shows the first 15 shots returned for a query using a Boilermaker football shot (upper left hand shot) when searching 23 sequences of video. Figure 12a) shows the result when

Sequence	Shots	Faces	Detect (%)	FA (%)	Correct. Class. (%)
news1	231	76	73.68	16.13	80.51
news2	72	29	93.1	23.25	83.33
news3	78	33	87.87	15.55	85.89
news4	103	42	90.47	13.11	88.34
news5	188	51	76.47	13.86	83.51
movie	142	92	84.78	28	80.28
drama	100	90	94.4	20	93
total	914	413	85.23	16.96	84.02

Table 2: Results for face detection using the ground truth sequences. Faces indicates the number shots containing at least one face. Detect indicates the Detection rate. FA indicates the False Alarm rate and Correct. Class the Correct Classification rate.

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

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

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

Figure 12: Example of search results using different similarity measures.

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

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

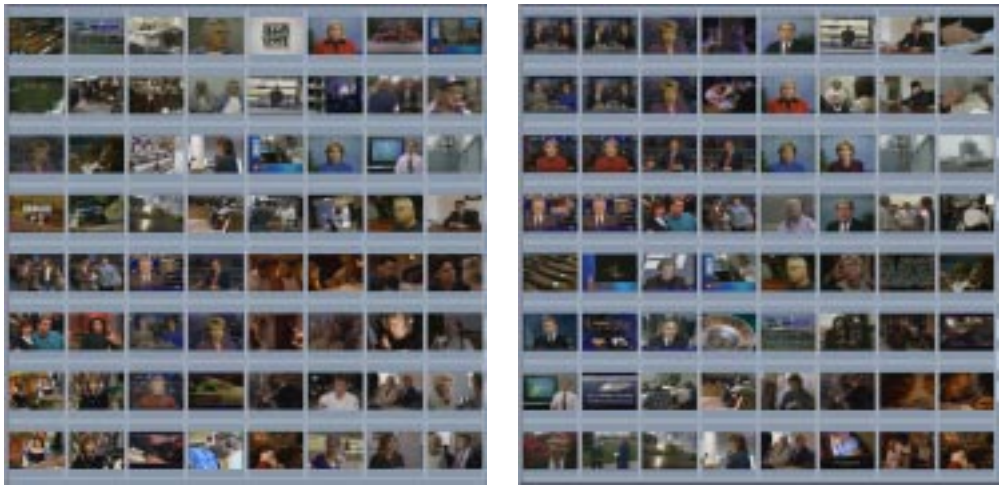
(f) $D_{ST} + D_M + D_{PS}$ (e + psudo-semantic distance)

Figure 13: Example of search results using different similarity measures.

only using D_{ST} the distance between the shot trees. Figure 12b) shows the result when the motion distance D_M is added, and Figure 12c) is the result when temporal distance D_T is added. Notice that the temporal and motion information are important cues in identifying good matching shots.

Figure 13 shows the first 15 shots returned for a query using local weather report shot (upper left hand shot) when searching 7 sequences of video. Figure 13a) shows the result when only using D_{ST} the distance between the shot trees. Figure 13b) shows the result when the motion distance D_M is added, and Figure 13c) is the result when pseudo-semantic distance D_{PS} is added. For this task, the pseudo-semantic distance is very effective at identifying matching shots.

In Figure 14, we show the shots using relevance feedback for similarity pyramid pruning and



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

Figure 14: The pruning result of Figure 9. (a) The pyramid organized with the default weightings (b) The pyramid organized with the optimized distance.

design. Figure 9 shows the original pyramid and the shots that have been selected by a user. Although these shots have dissimilar attributes, they all form a single semantic category, "anchorperson". Notice that the pruned set contains most of the shots which closely match the user selected set (Figure 9). Figure 14b shows the reorganized sub-database where the dissimilarity function is defined by the optimized distance described in Section 5. Notice that the shots in the relevance set are clustered together. These experiments with *ViBE* indicate that the use of the pseudo-semantic label and motion and temporal information can provide powerful aids in querying and browsing the database.

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] 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.

[2] Ramin Zabih, Justin Miller, and Kevin Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proceedings of the ACM International Conference on Multimedia*, San Francisco, CA, November 5-9 1995, pp. 189–200.

[3] 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.

[4] Wayne Wolf, "Key frame selection by motion analysis," in *Proceedings of IEEE Int'l Conference on Acoustic, Speech and Signal Processing*, 1996.

[5] 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 4-7 1998.

[6] 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," *to Appear in IEEE Trans. on Circuits and Systems for Video Technology Special Issue on Image/Video Processing for Interactive Multimedia*, 1998.

[7] 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.

[8] Nuno Vasconcelos and Andrew Lippman, "Towards semantically meaningful feature spaces for the characterization of video content," in *Proceedings of IEEE Int'l Conference on Image Processing*, Santa Barbara, CA, October 1997, vol. II, pp. 542–545.

[9] Behzad Shahraray and David C. Gibbon, "Automatic generation of pictorial transcripts of video programs," in *Proceedings of SPIE Conference on Multimedia Computing and Networking*, San Jose, CA, Febrary 1995, vol. 2417, pp. 338–341.

[10] Minerva M. Yeung and Boon-Lock Yeo, "Video visualization for compact presentation and fast browsing of pictorial content," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 5, pp. 771–785, October 1997.

[11] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Browsing and retrieving video content in a unified framework," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Austin, TX, June 28-July 1 1998, pp. 237–240.

[12] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker, "Query by image content: The QBIC system," *IEEE Computer*, pp. 23–31, September 1995.

[13] HongJiang Zhang, Jianhua Wu, Di Zhong, and Stephen Smoliar, "An integrated system for content-based video retrieval and browsing," *Pattern Recognition*, vol. 30, no. 4, pp. 643–658, April 1997.

[14] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Browsing and retrieving video content in a unified framework," in *Proceedings of IEEE Multimedia and Signal Processing workshop*, Los Angeles, CA, 1998.

[15] 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.

[16] Jau-Yuen Chen, C. 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.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.

[18] Jau-Yuen Chen, Charles A. Bouman, and John Dalton, "Similarity pyramids for browsing and organization of large image databases," in *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases III*, San Jose, CA, January 26-29 1998, vol. 3299.

[19] James Monaco, *How to Read a Film: The Art, Technology, Language, History, and Theory of Film and Media*, Oxford University Press, New York, NY, 1977.

[20] Arun Hampapur, Ramesh Jain, and Terry Weymouth, "Digital video segmentation," in *Proceedings of Second Annual ACM MultiMedia Conference and Exposition*, San Francisco, CA, October 1994, pp. 357–364.

[21] 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.

[22] A. Mufit Ferman and A. Murat Tekalp, "Multiscale content extraction and representation for video indexing," in *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems II*, Dallas, TX, November 3-4 1997, pp. 23–31.

[23] Bilge Gunsel, Yue Fu, and A. Murat Tekalp, "Hierarchical temporal video segmentation and content characterization," in *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems II*, Dallas, TX, November 3-4 1997, vol. 3329, pp. 46–56.

[24] F. Idris and S. Panchanathan, "Indexing of compressed video sequences," in *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases IV*, San Jose, CA, February 1996, vol. 2670, pp. 247–253.

[25] Bo Shen, Donnge Li, and Ishwar K. Sethi, "Cut detection via compressed domain edge extraction," in *IEEE Workshop on Nonlinear Signal and Image Processing*, Mackinac Island, MI, September 1997.

[26] Jianhao Meng, Yujen Juan, and Shih-Fu Chang, "Scene change detection in a MPEG compressed video sequence," in *Proceedings of SPIE Conference on Multimedia Computing and Networking*, San Jose, CA, February 1995, vol. 2417, pp. 180–191.

[27] 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.

[28] HongJiang Zhang, Shuang Yeo Tan, Stephen Smoliar, and Yihong Gong, "Automatic parsing and indexing of news video," *Multimedia Systems*, vol. 6, no. 2, pp. 256–266, 1995.

[29] Deborah Swanberg, Chiao-Fe Shu, and Ramesh Jain, "Knowledge guided parsing in video databases," in *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases*, San Jose, CA, April 1993, vol. 1908, pp. 13–24.

[30] 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, vol. II, pp. 252–255.

[31] Cuneyt Taskiran and Edward J. Delp, "Video scene change detection using the generalized trace," in *Proceedings of IEEE Int'l Conference on Acoustic, Speech and Signal Processing*, Seattle, WA, May 1998, pp. 2961–2964.

[32] Michael J. Swain and Dana H. Ballard, "Color indexing," *Intern. Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

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

[34] Bilge Gunsel and A. Murat Tekalp, "Content-based video abstraction," in *Proceedings of IEEE Int'l Conference on Image Processing*, Chicago, IL, October 4-7 1998.

[35] P.H.A. Sneath and R.R. Sokal, Eds., *Numerical Taxonomy*, Freeman, San Francisco, 1973.

[36] Anil K. Jain and Richard C. Dubes, Eds., *Algorithms for Clustering Data*, Prentice Hall, New Jersey, 1988.

[37] Minerva M. Yeung and Bede Liu, "Efficient matching and clustering of video shots," in *Proceedings of IEEE Int'l Conference on Image Processing*, Washington, DC, October 23-26 1995, vol. I, pp. 338–341.

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

[39] 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.

[40] Sargur N. Srihari Venu Govindaraju and David. B. Sher, "A computational model for face location," in *Proceedings of the third International Conference on Computer Vision*, Osaka, Japan, 1990, vol. 2, pp. 162–177.

[41] T.K. Leung, M.C. Burl, and P. Perona, "Finding faces in cluttered scenes using random labeled graph matching," in *Fifth International Conference on Computer Vision*, Cambridge, MA, June 1995, pp. 637–644.

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

[43] Guangzheng Yang and Thomas S. Huang, "Human face detection in a complex backgroung," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

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

[45] Gilles Burel and Dominique Carel, "Detection and localization of faces on digital images," *Pattern Recognition Letters*, vol. 15, no. 10, pp. 963–67, October 1994.

[46] Kah-Kay Sung and Tomaso Poggio, "Example-based learning for view-based human face detection," Technical Report AI Memo 1521, MIT, 1994.

[47] Baback Moghaddam and Alex Pentland, "Probabilistic visual learning for object detection," in *Proceedings of the fifth International Conference on Computer Vision*, Cambridge, MA, 1995, pp. 786–793.

[48] 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.

[49] Michael S. Lew, "Information theoretic view-based and modular face detection," in *2nd. International Conference on Automatic Face and Gesture Recognition*, Killington, VT, October 1996, pp. 198–203.

[50] N. Herodotou, K. Plataniotis, and A. Venetsanopoulos, "A color segmentation and classification scheme for facial image and video retrieval," in *IX European Signal Processing Conference*, Rhodes, Greece, September 1998.

[51] Veronica Vilaplana, Ferran Marques, Philippe Salembier, and Luis Garrido, "Region-based segmentation and tracking of human faces," in *European Signal Processing*, Rhodes, September 1998, pp. 593–602.

[52] 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.

[53] Stephen McKenna Yogesh Raha and Shaogang Gong, "Tracking and segmenting people in varying lighting conditions using colour," in *Proceedings of International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, April 1998.

[54] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp. 162–177, March 1994.

[55] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[56] Murray Aitkin and Donald B. Rubin, "Estimation and hypothesis testing in finite mixture models," *Journal of the Royal Statistical Society B*, vol. 47, no. 1, pp. 67–75, 1985.

[57] 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.

[58] Jau-Yuen Chen, Charles A. Bouman, and John Dalton, "Hierarchical browsing and search of large image databases," *Submitted to IEEE Trans. on Image Processing*, 1998, 1998.

[59] 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.

[60] Jau-Yuen Chen, Charles A. Bouman, and John Dalton, "Active browsing using similarity pyramids," in *Proceedings of 1998 Asilomar Conference on Signals, Systems, And Computers*, Pacific Grove, CA, November 2-4 1998.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.