Scene Detection in Videos Using Shot Clustering and Sequence Alignment

Vasileios T. Chasanis, Aristidis C. Likas, and Nikolaos P. Galatsanos

***Abstract*—Video indexing requires the efficient segmentation of video into scenes. The video is first segmented into shots and a set of key-frames is extracted for each shot. Typical scene detection algorithms incorporate time distance in a shot similarity metric. In the method we propose, to overcome the difficulty of having prior knowledge of the scene duration, the shots are clustered into groups based only on their visual similarity and a label is assigned to each shot according to the group that it belongs to. Then, a sequence alignment algorithm is applied to detect when the pattern of shot labels changes, providing the final scene segmentation result. In this way shot similarity is computed based only on visual features, while ordering of shots is taken into account during sequence alignment. To cluster the shots into groups we propose an improved spectral clustering method that both estimates the number of clusters and employs the fast global k-means algorithm in the clustering stage after the eigenvector computation of the similarity matrix. The same spectral clustering method is applied to extract the key-frames of each shot and numerical experiments indicate that the content of each shot is efficiently summarized using the method we propose herein. Experiments on TV-series and movies also indicate that the proposed scene detection method accurately detects most of the scene boundaries while preserving a good tradeoff between recall and precision.**

***Index Terms*—Global k-means, key-frames, scene detection, sequence alignment.**

I. Introduction

In recent years the extended use of videos in several applications such as internet-TV and video on demand, as well as the thousand TV-series and movies produced every year has led to a significant increase in the availability and the amount of video information. Video indexing, retrieval and analysis seem quite difficult due to this huge amount of data constantly produced. Video scene segmentation provides the most efficient solution so far. However, to proceed with scene segmentation, low level segmentation of the video must be first applied.

The smallest physical segment of a video is the shot and is defined as an unbroken sequence of frames recorded from the same camera. The visual content of each shot of the video can be represented by one or multiple frames, called key-frames. The number of key-frames cannot be predetermined because due to content variation it may be different for each shot. For example for a static shot where there is little object motion, one key-frame may represent the shot quite adequately, whereas when there is high camera and object motion, more key-frames are needed for a good representation. Several approaches have been proposed for key-frame extraction. In [22] the authors detect multiple frames using unsupervised clustering based on the visual variations in shots. A main drawback of this algorithm is the determination of the appropriate number of key-frames to represent each shot which depends on the threshold parameter that controls the density of the clusters. A variant of this algorithm is presented in [11] where the final number of key-frames depends on a threshold parameter which defines two frames to be similar.

Proceeding further towards the goal of video indexing and retrieval requires the grouping of shots into scenes. A scene can be regarded as a series of semantically correlated shots. The term scene usually refers to a group of shots taken in the same physical location describing objects or events. A more compact representation of a video could be the merging of scenes into logical story units that correspond to chapters describing the different subthemes of a movie.

Several approaches have been proposed for the scene segmentation problem. In [11] the authors transform this task into a graph partitioning problem. A shot similarity graph is constructed, where each node represents a shot and the edges between shots depict their similarity based on color and motion information. Then the normalized cuts [13] method is applied to partition the graph. In [4], a method is proposed for detecting boundaries of the logical story units by linking similar shots and connecting overlapping links. For each shot, all key frames are merged into a larger image and the similarity between shots is computed by comparing these shot images. A similar approach is presented in [17], where a scene transition graph is constructed to represent the video and the connectivity between shots. Then, this transition graph is divided into connected subgraphs representing the scenes. A different approach is presented in [10] where a two-pass algorithm is proposed. In the first pass shots are clustered by computing backward shot coherence, a similarity measure of a given shot with respect to the previously seen shots, while in the second pass oversegmented scenes are merged based on the computation of motion content in scenes. Another method that uses Markov chain Monte Carlo to determine scene boundaries is proposed in [20]. Two processes, diffusions and jumps, are used to update the scene

Manuscript received March 24, 2008; revised August 31, 2008. First published December 22, 2008; current version published January 08, 2009. This work, the research project (PENED), was supported by the E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%). The associate editor coordinating the review if this manuscript and approving it for publication was Dr. Marcel Worring.

V. T. Chasanis and A. C. Likas are with the Department of Computer Science, University of Ioannina, Ioannina, Greece 45110 (e-mail: vchasani@cs.uoi.gr; arly@cs.uoi.gr).

N. P. Galatsanos is with the Department of Electrical and Computer Engineering, University of Patras, 26500 Rion, Greece (e-mail: ngalatsanos@upatras.gr).

Digital Object Identifier 10.1109/TMM.2008.2008924

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

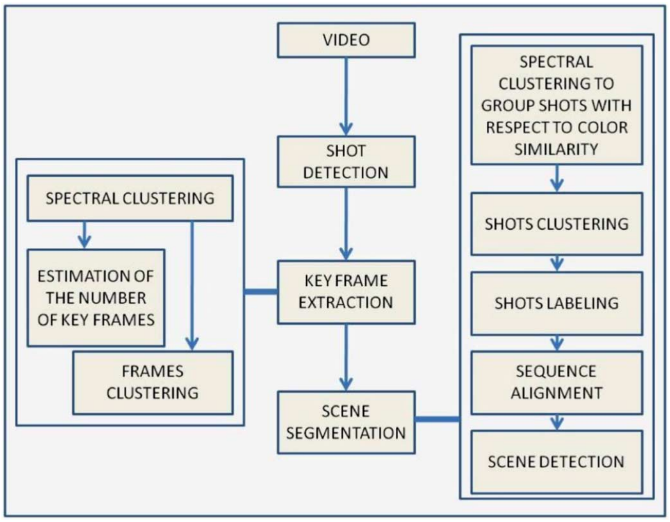

Fig. 1. Main steps of our method.

boundaries that are initialized at random positions. Diffusions are the operations that adjust the boundaries between adjacent scenes, while jump operations merge or split existing scenes.

Most of the above approaches, calculate shot similarity based on visual similarity. Furthermore, they consider the temporal distance of shots as an extra feature that is taken into account when computing the similarity between two shots for shot clustering into scenes. Due to the absence of prior knowledge concerning the video content and the duration of scenes, it is difficult to determine an appropriate weight parameter that will account for the contribution of the temporal distance in the computation of the overall similarity between shots.

One of the novelties of our approach is that shots are clustered into groups using an improved version of the typical spectral clustering method [8] that uses the fast global k-means algorithm [6] in the clustering stage after the eigenvector computation. In addition, we employ a criterion for estimating the number of groups based on the magnitude of the eigenvalues of the similarity matrix. The resulted groups of shots are not the final scene boundaries, but this clustering procedure is a preprocessing step towards the final detection of scene boundaries. Another novelty of our method is that shot similarity is computed based only on visual features, because incorporating time distance in a shot similarity metric requires a priori knowledge of the scene duration. Thus, it is a quite difficult task to determine a distance parameter that defines whether two shots are related or not. In our method cluster labels are assigned to shots according to their visual content and then, sequences of shot labels are compared to identify changes in the patterns of successive labels. In that way time distance between shots is not taken into account since our method locally searches for changes in patterns of shot labels ignoring the relation between shots with respect to time distance.

Typically the sequence of shots in a video follows specific production rules. The most common is known as the 180° rule, where the director draws a line in the physical setting of a scene and all cameras are placed on the same side of this line [14]. This production rule produces repeating shots of one person, a group of persons or the same setting which is commonly seen in movies, documentaries and TV-series. The most common patterns of repetitive shots are two. The first one is a dialogue between two or more persons, where the camera switches from one person to another, thus producing a sequence of shots like $ABABCBCABABC$, where A, B and C are the shot labels for three different persons. Another common pattern is a sequence of shots like $A_1A_2A_1A_3A_3A_2$ where A_1, A_2 and A_3 are captions of three different cameras providing views of the same physical setting from different angles. When a scene changes it is expected that a change in such patterns will occur. For example, if two dialogues take place in different scene, it is expected that a sequence of shots like $ABABCBDEDFEDEF$ is produced where $ABABCB$ corresponds to the first scene and $DEDFEDEF$ corresponds to the second scene. To identify the change in pattern, a comparison of successive non-overlapping windows of shot labels is performed. Thus, we need to define a proper measure to define whether two sequences are related (share the same patterns of shots) or not. A very efficient category of algorithms that compare sequences in order to define whether two sequences are related or not are the sequence alignment algorithms that are successfully used in biological applications [5].

In our approach, to compare sequences we use the Needleman–Wunsch global sequence alignment algorithm [7], which performs global alignment on two sequences and is guaranteed to find the alignment with the maximum score. This algorithm requires the definition of a substitution matrix in order to implement the alignment. This matrix represents the rate at which one character in a sequence changes to another character over time. In our method the substitution matrix is formulated based on criteria that are adapted to the problem of scene detection. Color similarity between clusters of shot labels and probability of existence of a pair of successive shot labels are the two components that contribute to the substitution matrix. The score of each alignment is given through a scoring function which takes into account matches, mismatches and gaps of shot labels. When an alignment gives a low score, a change in the patterns of shot labels is implied and suggests a scene boundary. The proposed two-stage approach (shot clustering, sequence alignment) achieves high correct detection rates while preserving a good trade off between the number of missed scenes and the number of false detected scenes.

Another novelty of the proposed method is that the key-frame extraction problem is treated using an improved spectral clustering algorithm (also employed for shot clustering) which estimates the number of key-frames using the eigenvalues of the similarity matrix corresponding to pairs of shot frames.

In Fig. 1 we summarize the main steps of our approach and the algorithms employed in these steps. The video is segmented into shots and the spectral clustering algorithm is employed to extract the key-frames of the corresponding shots. Next, shots are grouped with respect to their visual similarity and labeled according to the group they are assigned. Finally, a sequence alignment algorithm is implemented to identify high dissimilarities between successive windows of shot labels. Scene boundaries are considered to be the points of high dissimilarity.

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

The rest of the paper is organized as follows: In Section II, the procedure for extracting key-frames of shots and for computing shot similarity is described. In Section III, the proposed scene detection algorithm is presented. In Section IV, we present numerical experiments and compare our method with two other methods proposed in [11] and [17]. Finally, in Section V, we conclude our work and provide suggestions for further study.

II. Key-Frame Extraction and Shot Similarity

The first level of video segmentation is shot detection. We implemented the most widely used method for shot detection [21] that is based on color histograms. For each frame a 16-bin HSV normalized histogram is used [11], with eight bins for hue and four bins for each of saturation and value.

A. Spectral Clustering of Video Frames

To perform key-frame extraction the video frames of a shot are clustered into groups using an improved spectral clustering algorithm. The medoid of each group, defined as the frame of a group whose average similarity to all other frames of this group is maximal, is characterized as a key-frame. The main steps of the typical spectral clustering algorithm [8] are described next. Suppose there is a set of objects $S = s_1, s_2, \dots, s_N$ to be partitioned into K groups.

1) Compute similarity matrix $A \in \mathbb{R}^{N\times N}$ for the pairs of objects of the data set S.
2) Define D to be the diagonal matrix whose (i,i) element is the sum of the elements of A's i-th row and construct the Laplacian matrix $L = I - D^{-1/2}AD^{-1/2}$.
3) Compute the K principal eigenvectors $x_1, x_2, \dots, x_K$ of matrix L to build an $N \times K$ matrix $X = [x_1 \ x_2 \ \dots \ x_K]$.
4) Renormalize each row of X to have unit length and form matrix Y so that:

$$y_{ij} = x_{ij} \Bigg/ \left(\sum_j x_{ij}^2\right)^{1/2}. \qquad (1)$$

5) Cluster the rows of Y into K groups using k-means.
6) Finally, assign object s_i to cluster j if and only if row i of matrix Y has been assigned to cluster j.

In what concerns our key-frame extraction problem, suppose we are given a data set $H = H_1, \dots, H_N$ where H_n is the feature vector (color histogram) of the n-th frame. The distance function we consider is the Euclidean distance between the histograms of the frames. As a result each element of similarity matrix A is computed as follows:

$$a(i,j) = 1 - \frac{1}{\sqrt{2}}\sqrt{\sum_{h\in \text{bins}} (H_i(h) - H_j(h))^2}. \qquad (2)$$

In our method, in the fifth step of the spectral clustering algorithm instead of using the typical k-means approach, we have used the fast version of the very efficient global k-means algorithm [6]. Global k-means in an incremental deterministic clustering algorithm that overcomes the important initialization problem of the typical k-means approach. This initialization problem has been found to be severe in the case of frame clustering, significantly affecting the quality of the key-frames. Using the global k-means, the obtained key frames usually provide a sensible representation of shot content. Next we briefly review the global k-means algorithm. Suppose we are given a data set $X = x_1, \dots, x_N$, $x_n \in R^d$ to be partitioned into K disjoint clusters $C_1, C_2, \dots, C_K$.

This algorithm is incremental in nature. It is based on the idea that the optimal partition into K groups can be obtained through local search (using k-means) starting from an initial state with i) the $K-1$ centers placed at the optimal positions for the $(K-1)$-clustering problem and ii) the remaining K-th center placed at an appropriate position within the dataset. Based on this idea, the K-clustering problem is incrementally solved as follows. Starting with $k=1$, find the optimal solution which is the centroid of the data set X. To solve the problem with two clusters, the k-means algorithm is executed N times (where N is the size of the data set) from the following initial positions of the cluster centers: the first cluster center is always placed at the optimal position for the problem with $k=1$, whereas the second center at execution n is initially placed at the position of data x_n. The best solution obtained after the N executions of k-means is considered as the solution for $k=2$. In general if we want to solve the problem with k clusters, N runs of the k-means algorithm are performed, where each run n starts with the $k-1$ centers initially placed at the positions corresponding to the the solution obtained for the $(k-1)$-clustering problem, while the k-th center is initially placed at the position of data x_n. A great benefit of this algorithm is that it provides the solutions for all k-clustering problems with $k \le K$.

The computational cost of the global k-means algorithm can be reduced without significant loss in the quality of the solution using the fast global k-means algorithm [6]. This method computes an upper bound E_n of the final clustering error obtained by initializing a new cluster center at position x_n. The initial position of the new cluster center is selected as the point x_i that minimizes E_n and k-means runs only once for each k. The application of fast global k-means requires a single execution of k-means for each value (m) of the number of clusters: $m = 1, \dots, k$.

B. Estimation of the Number of Clusters Using Spectral Clustering

As already mentioned in the Introduction, the number of key-frames cannot be predetermined due to the different content of each shot. In our approach we attempt to estimate the number of the key-frames using results from the spectral graph theory.

Assume we wish to partition dataset S into K disjoint subsets $(S_1, \dots, S_K)$, and let $X = [X_1, \dots, X_K] \in \mathbb{R}^{N\times K}$ denote the partition matrix, where X_j is the binary indicator vector for set S_j such that:

$$\begin{aligned} X(i,j) &= 1\text{: if } i \in S_j \\ X(i,j) &= 0\text{: otherwise.} \end{aligned} \qquad (3)$$

This clustering problem can be defined as [18]:

$$\begin{aligned} &\max_X \ \text{trace}(X^T L X) \\ &\text{s.t. } X^T X = I_K \quad \text{and} \quad X(i,j) \in \{0,1\}, \end{aligned} \qquad (4)$$

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

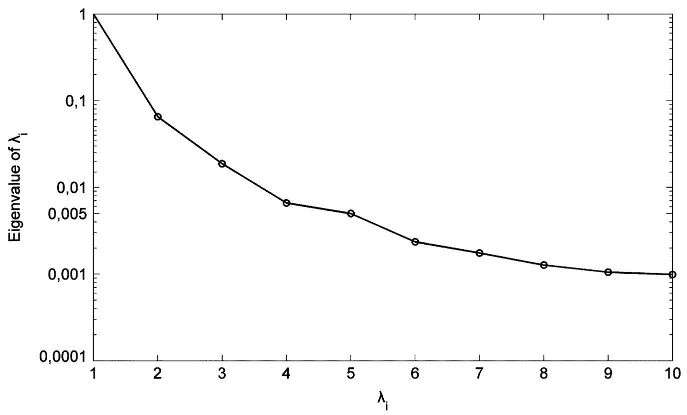

Fig. 2. Eigenvalues and selection of k.

where L is the Laplacian matrix defined in Section II-A. The spectral clustering algorithm (for K clusters) provides solution to the following relaxed optimization problem:

$$\begin{aligned} &\max_{Y} \operatorname{trace}(Y^T L Y) \\ &\text{s.t. } Y^T Y = I_K. \end{aligned} \tag{5}$$

Relaxing Y into the continuous domain turns the discrete problem into a continuous optimization problem. The optimal solution is attained at $Y = U_K$, where the columns u_i of $U_k, i = 1, \ldots, K$, are the eigenvectors corresponding to the ordered top K largest eigenvalues λ_i of L. Since it holds that [19]:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_K = \max_{Y^T Y = I_K} \operatorname{trace}(Y^T L Y) \tag{6}$$

the optimization criterion that also quantifies the quality of the solution for K clusters and its corresponding difference for successive values of K are respectively given by:

$$\begin{aligned} \operatorname{sol}(K) &= \lambda_1 + \lambda_2 + \cdots + \lambda_K \\ \operatorname{sol}(K+1) - \operatorname{sol}(K) &= \lambda_{K+1}. \end{aligned} \tag{7}$$

When the improvement in this optimization criterion (i.e., the value of the λ_{K+1} eigenvalue) is below a threshold, improvement by the addition of cluster $K+1$ is considered negligible, thus the estimate of the number of clusters is assumed to be K. The threshold value that is used in all our experiments was fixed to $\mathrm{Th} = 0.005$ with very good results. In Fig. 2 we provide an example of the eigenvalues of a matrix L for a key-frame extraction problem with five clusters (key-frames).

Summarizing, to extract the appropriate key-frames for a shot, we compute the corresponding Laplacian matrix L and analyze its eigenvalues to select the number of key-frames k_f. After we have determined k_f, we proceed with the steps 4–6 of the spectral clustering algorithm. In our implementation the fast global k-means is employed in step 5, instead of k-means.

C. Shot Similarity

As explained earlier, shots that belong to the same scene often have similar color content. As suggested in [11] the visual similarity between a pair of shots i and j can be computed as the maximum color similarity (ColSim) among all possible pairs of their key-frames:

$$\operatorname{VisSim}(i,j) = \max_{p \in K_i, q \in K_j} \operatorname{ColSim}(p,q) \tag{8}$$

where K_i and K_j are the sets of key-frames of shots i and j respectively, and the color similarity (ColoSim) between two frames f_i, f_j is defined as the histogram intersection [15]:

$$\operatorname{ColSim}(i,j) = \sum_{h \in \text{bins}} \min(H_i(h), H_j(h)) \tag{9}$$

where H_i, H_j are the HSV normalized color histograms of frames f_i and f_j respectively.

III. Scene Detection

Scene detection is a quite difficult task, because a scene is a group of shots that are i) semantically correlated and ii) continuous in time. The semantic correlation between two shots cannot actually be described with low-level features. However low-level features such as color, give useful information about the connection between shots and the physical setting where the scene takes place. On the other hand, taking into account the contribution of temporal distance in the computation of the overall similarity between shots is difficult, due to the absence of prior knowledge about the scene duration.

A. Shots Clustering

In order to perform scene detection, clustering of shots into groups, taking into account visual similarity (VisSim) and time adjacency is required. Suppose there is a set $V = v_1, v_2, \ldots, v_N$ of N shots, ordered in time, to be segmented. In order to implement shot grouping, an $N \times N$ similarity matrix A must be specified. In [9], [11] both visual similarity and time distance are combined in a single similarity metric (see Section IV.C.3). On the contrary, in our method we have considered only visual similarity (8):

$$a(i,j) = \operatorname{VisSim}(v_i, v_j), \quad v_i, v_j \in V \tag{10}$$

for shot clustering, while ordering of shots is taken into account at a later processing stage.

After the similarity matrix A has been computed, the modified spectral clustering algorithm is used to group shots into clusters. The main steps of this algorithm have been presented earlier. The selection of the number of shot clusters is done in a way similar to the key-frame extraction problem. However it is worth mentioning that the number of shot clusters is not equal to the number of scenes in the video. Our aim is to estimate the principal color distributions over the video shots and group all shots according to that color distribution that they fit most. Following the same approach used for key-frame extraction, the analysis of the eigenspectrum of the Laplacian matrix

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

$$\begin{array}{ccccccccccccccccccccc} V_{01} & V_{02} & V_{03} & V_{04} & V_{05} & V_{06} & V_{07} & V_{08} & V_{09} & V_{10} & V_{11} & V_{12} & V_{13} & V_{14} & V_{15} & V_{16} & V_{17} & V_{18} & V_{19} & V_{20} & V_{21} \\ C_1 & C_1 & C_1 & C_1 & C_1 & C_2 & C_2 & C_2 & C_2 & C_3 & C_5 & C_3 & C_5 & C_3 & C_5 & C_3 & C_4 & C_4 & C_2 & C_4 & C_4 \end{array}$$

Fig. 3. Video sequence of labels.

L provides an estimate of the number of clusters K. Then shots are clustered into K groups with respect to their visual content (color histogram similarity), while the final number of scenes will be extracted at a later step of our algorithm.

Once the spectral clustering algorithm has provided a partition of the shots into K clusters $\{C_1, C_2, \ldots, C_K\}$, a label is assigned to each shot according to the cluster it belongs, thus producing a symbolic sequence of labels. In this way, the sequence of shots is transformed into a new sequence of labels that illustrates the visual similarity between shots. An illustrative example is given in Fig. 3:

To each shot V_t (the index t implies time) a label from the set $\{C_1, C_2, C_3, C_4, C_5\}$ is assigned to. Typically, during a scene there exists a sequence of similar shot labels (different captions of the same person/place) or a sequence of repetitive label patterns (rotation of different camera captions, e.g., dialogue). We consider that a scene change occurs when the pattern of symbols changes. In our example, distinct scenes correspond to shots with time indices 1–5, 6–9, 10–16 (repetitive pattern C_3C_5) and 17–21. In practice, due to the presence of noise (shot V_{19} with label C_2), it is not trivial to discriminate patterns of symbols. In the proposed approach we treat this problem using a sequence alignment algorithm as it will be explained next.

B. Scene Segmentation Through Sequence Alignment

As already mentioned in the introduction, videos such as movies, documentaries and TV-series, follow some production rules. These rules result in the generation of patterns of shots inside a scene. Different scenes share different patterns of shots (different subsequences of labels). Thus, it is expected to detect scene changes in cases where the pattern of shot labels changes. In order to find the points in the sequence of shot labels where the pattern of symbols changes, we compare successive non-overlapping windows of shot labels using a sequence alignment algorithm. More specifically, given the set V of N shots, the subsequences of the original video sequence to be compared at each iteration i are formulated as:

$$\begin{aligned} X_1^i &= L_i L_{i+1} \ldots L_{i+w-1} \\ X_2^i &= L_{i+w} L_{i+w+1} \ldots L_{i+2w-1} \quad i = 1, \ldots, N-2w \end{aligned} \tag{11}$$

where w is the length of the window used and $L_i, i = 1, \ldots, N$ are the shot labels. In Fig. 4 the first three subsequences of the video sequence in Fig. 3 are shown, using a window of length 4. In iteration 1 the first subsequence containing shots $V_1 - V_4$ will be compared with subsequence containing shots $V_5 - V_8$. In next iteration the two subsequences under comparison are those containing shots $V_2 - V_5$ and $V_6 - V_9$ respectively.

A well established approach to compare sequences of symbols is the sequence alignment algorithm. Significant similarity between sequences may imply that the sequences belong to the same scene. Our interest however, focuses on cases of high dissimilarity that is a strong indication of a scene boundary. The sequence alignment algorithm we used in our approach is the Needleman–Wunsch algorithm [7] which is commonly used in bioinformatics to align protein or nucleotide sequences. This algorithm performs global alignment on two sequences and is guaranteed to find the alignment with the maximum score. The input consists of two sequences of length w as described in (12). Let us denote

$$\overbrace{\begin{array}{ccccccccccccccccccccc} V_{01} & V_{02} & V_{03} & V_{04} & V_{05} & V_{06} & V_{07} & V_{08} & V_{09} & V_{10} & V_{11} & V_{12} & V_{13} & V_{14} & V_{15} & V_{16} & V_{17} & V_{18} & V_{19} & V_{20} & V_{21} \\ C_1 & C_1 & C_1 & C_1 & C_1 & C_2 & C_2 & C_2 & C_2 & C_3 & C_5 & C_3 & C_5 & C_3 & C_5 & C_3 & C_4 & C_4 & C_2 & C_4 & C_4 \end{array}}^{OriginalSequence}$$

$$\text{Iteration 1} \rightarrow \underbrace{(C_1C_1C_1C_1)}_{X_1^1}\underbrace{(C_1C_2C_2C_2)}_{X_2^1}$$

$$\text{Iteration 2} \rightarrow \underbrace{(C_1C_1C_1C_1)}_{X_1^2}\underbrace{(C_2C_2C_2C_2)}_{X_2^2}$$

$$\text{Iteration 3} \rightarrow \underbrace{(C_1C_1C_1C_2)}_{X_1^3}\underbrace{(C_2C_2C_2C_3)}_{X_2^3}$$

Fig. 4. Subsequences to be compared.

$$X_1 = L_1 L_2 \ldots L_w \quad \text{and} \quad X_2 = M_1 M_2 \ldots M_w. \tag{12}$$

The labels $L_i, M_i, i = 1, \ldots, w$ belong to some alphabet of K symbols, where K is the number of cluster labels generated from the spectral clustering of shots. To align these sequences a $w \times w$ matrix N is constructed where the value $N(i,j)$ is the score of the best alignment between the segment $X_1(1 \ldots i)$ and the segment $X_2(1 \ldots j)$ [5]. There are three possible ways to obtain the best score of an alignment up to $X_1(i)$, $X_2(j)$: a) $X_1(i)$ could be aligned to $X_2(j)$, b) $X_1(i)$ could be aligned to a gap and c) $X_2(j)$ could be aligned to a gap. The best score will be the largest of these three options:

$$N(i,j) = \begin{cases} N(i-1, j-1) + S(X_1(i), X_2(j)) \\ N(i-1, j) - d \\ N(i, j-1) - d \end{cases} \tag{13}$$

where S is a substitution matrix and d is a gap penalty. The definition and calculation of these quantities are given below. The traceback from $N(w,w)$ to $N(0,0)$ defines the optimal alignment of X_1 and X_2. The time complexity for aligning two sequences of length w is $O(w^2)$. A typical example of a sequence alignment over an alphabet $\{C_1, C_2, C_3, C_4\}$ is given in Fig. 5. The output of the alignment algorithm is an alignment matrix. The columns of this matrix that contain the same label in both rows are called matches (M), while columns containing different letters are called mismatches (m). The columns of the alignment containing one space are called gaps (G). A gap in an alignment is defined as a contiguous sequence of spaces in one of the rows of the alignment matrix [5]. By inserting one or more gaps, the algorithm succeeds in aligning symbols that occur in different positions.

The sequence alignment algorithm requires a substitution matrix S and a gap cost function δ. In our problem, the elements $s(i,j)$ of the substitution matrix S express how similar are shot labels C_i and C_j in terms of color and position. The color similarity between shot labels can be defined from the similarity of their respective clusters. In what concerns position, it can be

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

$$
\begin{array}{lll}
Seq_1 & : & C_1C_2C_1C_2C_3C_4C_1C_1C_3C_4C_4 \\
Seq_2 & : & C_4C_1C_2C_1C_2C_2C_3C_4C_4C_1C_3C_4
\end{array}
$$

Output : (Alignment matrix)

Seq_1	_	C_1	C_2	C_1	C_2	_	C_3	C_4	C_1	C_1	C_3	C_4	C_4
Seq_2	C_4	C_1	C_2	C_1	C_2	C_2	C_3	C_4	C_4	C_1	C_3	C_4	_
Type	G	M	M	M	M	G	M	M	m	M	M	M	G

Fig. 5. Alignment matrix of a sequence alignment example.

$$
\begin{array}{lll}
Seq_1 & : & C_1C_2C_1C_2C_3C_4C_1C_1C_3C_4C_4 \\
Seq_2 & : & C_4C_1C_2C_1C_2C_2C_3C_4C_4C_1C_3C_4
\end{array}
$$

Seq_1	_	C_1	C_2	C_1	C_2	_	C_3	C_4	C_1	C_1	C_3	C_4	C_4
Seq_2	C_4	C_1	C_2	C_1	C_2	C_2	C_3	C_4	C_4	C_1	C_3	C_4	_
Type	G	M	M	M	M	G	M	M	m	M	M	M	G
Score	-1	1	1	1	1	-1	1	1	-1	1	1	1	-1

Fig. 6. Scoring function of the sequence alignment example.

observed that during a scene, repetitive patterns of labels frequently occur. This increases the possibility that a shot label i can be aligned with a shot label j and the opposite with high score, when shot labels i and j belong to the same pattern, thus the similarity between shot labels, as far as position is concerned, can be expressed through the possibility that a shot label i precedes or follows a shot label j. As a result, the substitution matrix S is defined as the combination of two different similarity metrics. Next, we define these similarity metrics, one for color similarity and one for position similarity, and how they are combined to formulate matrix S.

For color similarity, for each cluster i we compute the medoid m_i, defined as the shot of a cluster, whose average similarity to all the other shots of this cluster is maximal. Then, the visual similarity between shot clusters can be computed from the visual similarity between the corresponding medoids, thus producing a cluster similarity matrix (CSM):

$$\mathrm{CSM}(i,j) = \mathrm{VisSim}(m_i, m_j) m_i, m_j \in M \tag{14}$$

where VisSim is given from (8) and M is the set of the medoids of the clusters. Next we compute a pair probability matrix (PPM) which represents the probability (frequency) of existence of a pair of sequential labels in the video. There are $N-1$ pairs of successive labels in a video containing N shots and the PPM matrix is given from the following equation:

$$\mathrm{PPM}(i,j) = \frac{1}{N-1}\{\#\,\mathrm{pairs}(L_1 = C_i, L_2 = C_j)\} \tag{15}$$

where L_1, L_2 are the first and the second label of a pair respectively and $i,j = 1,\ldots,K$. The final substitution matrix S is computed as follows:

$$S(i,j) = \begin{cases} A(i,j) + B(i,j), & i = j \\ -\alpha(1 - A(i,j)) - \beta(1 - B(i,j)), & i \neq j \end{cases}, \tag{16}$$

where A and B are the CSM and PPM matrices respectively and α, β with $\alpha + \beta = 1$, are weights controlling the contribution of each matrix element. Each entry (i,j) of the matrix represents the score of alignment of the ith and jth symbols in the alphabet. The diagonal elements of matrix S account for match operations, while the non-diagonal elements account for the mismatch operations during the alignment procedure. To represent the cost of having a gap of length l we consider the linear gap model $\delta(l) = -ld$, where d is a nonnegative constant called the linear gap penalty and is set to 1.

After the formulation of the substitution matrix, the sequence alignment algorithm computes the score for the best alignment in each iteration (Fig. 4). The evaluation of the alignment is based on the number of matches, mismatches and gaps between the sequences. A scoring function [5] is defined as:

$$F = (\text{score of matches}) - (\text{score of mismatches}) - (\text{score of gaps}). \tag{17}$$

In Fig. 8 we illustrate the computation of this scoring function for the previous sequence alignment example using a similarity matrix with score $+1$ for matches (M), -1 for mismatches (m) and a linear gap (G) function with $d = 1$.

We apply the above sequence alignment procedure to all pairs of subsequences $(X_1^i, X_2^i), i = 1, \ldots, N - 2w$. The values of the scoring function are stored in a score sequence SC. In Fig. 7 an example of the score sequence values is shown. At the scene boundaries a change in the pattern of labels occurs, thus it is expected to observe a low score value. In other words, low score values are considered as indicators of the possibility for scene change. The global minimum of the score sequence corresponds to the most dissimilar subsequences in the video, thus to the most certain scene boundary. Since there are many local minima in the score sequence, it is expected that those with value close to the global minimum to correspond to the most probable scene boundaries. To locate these boundaries we first find the global minimum value in sequence SC. Then, the local minima of the sequence SC that are less than a percentage of the global minimum value are characterized as scene boundaries. In our experiments, a percentage equal to 80% was used providing very good results.

IV. Experiments

In this section we present numerical experiments for the key-frame extraction problem and the scene detection problem, and we compare our methods with existing approaches.

A. Data

To evaluate the performance of our key-frame extraction algorithm we use seven frame sequences (Dataset A) taken from TV-series and sports (Table I), which contain high camera and object motion. The first frame sequence describes an action of a comedy movie that takes place in an office. The next three sequences describe three attempts in a NBA Slam Dunk Contest and the other three a goal attempt in a football match taken from three individual cameras.

For the scene detection problem, the video sequences (Dataset B) used for our data set were taken from TV-series and movies. The majority of the videos are drama and comedy films, while the rest are action films. Ten videos were used consisting of 5051 shots and 177 scenes (Table II). On average there were 505 shots and 18 scenes per video and the total duration of the videos of the test set was approximately 5 h

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

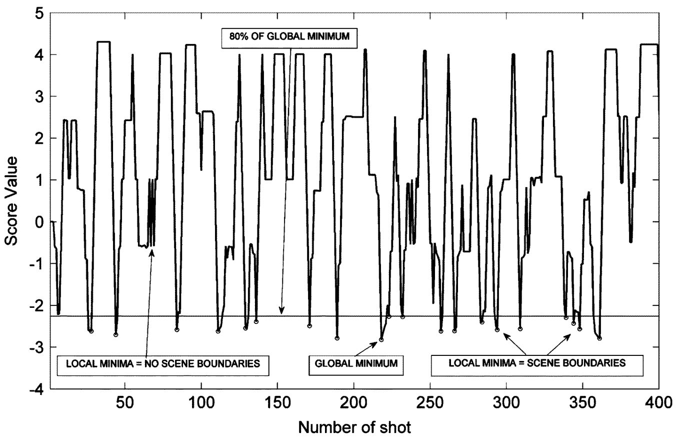

Fig. 7. Scoring function of a sequence alignment example.

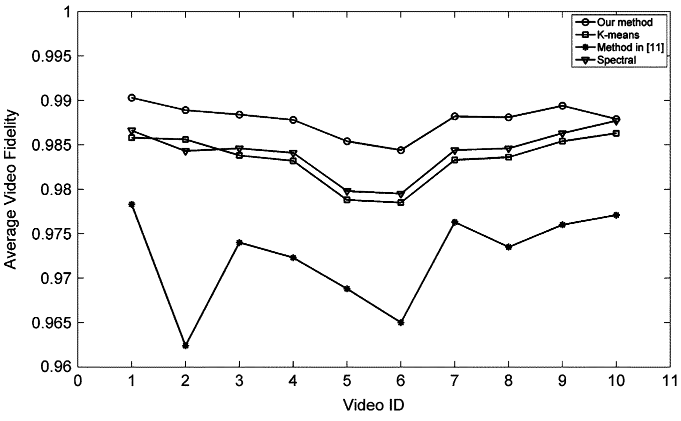

Fig. 8. Comparative results of the tested key-frame extraction algorithms using Average Video Fidelity measure on dataset B.

TABLE I
DATASET A CHARACTERISTICS

Frame Sequence	No. Frames	Genre
F_1	633	Comedy
F_2	144	Basketball
F_3	145	Basketball
F_4	146	Basketball
F_5	225	Football
F_6	300	Football
F_7	172	Football

(293 min). The ground truth for this data set was manually defined by a human observer of our research team. Each scene was as a group of shots taken in the same physical location describing objects or events.

B. Key-Frame Extraction Experiments

A difficult issue of the key-frame extraction problem is related to the evaluation of the extracted key-frames, since it is rather subjective which frames are the best representatives of the content of a shot. There are several quality measures that can be used to evaluate the efficiency of the algorithms. In [3], two quality measures are used. The first is the Fidelity measure proposed in [2] and the second is the Shot Reconstruction Degree measure proposed in [16].

TABLE II
DATASET B CHARACTERISTICS

Video	Duration(min)	Shots	Scenes	Genre
V_1	22	404	15	comedy
V_2	31	591	18	comedy
V_3	30	587	16	comedy
V_4	23	437	13	comedy
V_5	27	633	14	drama
V_6	26	454	17	drama
V_7	32	377	15	comedy
V_8	45	608	25	drama
V_9	31	714	25	action
V_{10}	26	246	19	action

1) Average Shot Fidelity: The Fidelity measure compares each key-frame with other frames in the shot. Given the frame sequence $F = \{F_1, F_2, \ldots, F_N\}$ and the set of key-frames $\mathrm{KF} = \{\mathrm{KF}_1, \mathrm{KF}_2, \ldots, \mathrm{KF}_{N_{\mathrm{kf}}}\}$ the distance between the set of key-frames KF and a frame F_n is defined as

$$d(F_n, \mathrm{KF}) = \min_j \mathrm{Diff}(F_n, \mathrm{KF}_j), \quad j = 1, \ldots, N_{\mathrm{kf}} \tag{18}$$

where N_{kf} is the number of key-frames and $\mathrm{Diff}(F_i, F_j)$ a distance measure between two frames F_i and F_j.

The Average Shot Fidelity (ASF) measure is computed using the average of the minimal distances between the key frame set and the video shot and is given from the following equation:

$$\mathrm{ASF}(F, \mathrm{KF}) = 1 - \frac{1}{N}\sum_{n=1}^{N} d(F_n, \mathrm{KF}). \tag{19}$$

2) Shot Reconstruction Degree: Given the set of key-frames, the whole frame sequence of a shot can be reconstructed using an interpolation algorithm. The better the reconstructed video sequence approximates the original sequence, the better the set of key-frames summarizes the video content. More specifically, given the frame sequence F, the set of key-frames KF and a frame interpolation algorithm IA(), we can reconstruct any frame from a pair of key-frames in KF [16]:

$$\tilde{F}_n = \mathrm{IA}(KFn_j, KFn_{j+1}), \quad n_j \le n < n_{j+1}. \tag{20}$$

The Shot Reconstruction Degree (SRD) measure is defined as follows:

$$\mathrm{SRD}(F, KF) = \sum_{n=0}^{N-1} (\mathrm{Sim}(F_n, \tilde{F}_n)) \tag{21}$$

where $\mathrm{Sim}(\,)$ is given from the following equation:

$$\mathrm{Sim}(F_n, \tilde{F}_n) = \log(\mathrm{MaxDiff}/\mathrm{Diff}(F_n, \tilde{F}_n)) \tag{22}$$

where $\mathrm{Diff}(F_i, F_j)$ is a distance measure between two frames F_i and F_j and MaxDiff the largest possible value that the frame difference measure can assume.

3) Comparison: We compare the proposed approach to three other methods. The first one is the simple k-means algorithm applied on the histogram vectors. For each shot we perform 20 runs of the k-means algorithm keeping as final solution one with the minimum clustering error. The number of clusters in k-means algorithm is assumed to be the same as selected using

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

TABLE III
COMPARATIVE RESULTS OF THE TESTED KEY-FRAME EXTRACTION ALGORITHMS USING AVERAGE SHOT FIDELITY MEASURE ON DATASET A

ASF	Algorithm			
Frame Seq.	Our method	K-means	Method in [11]	Spectral
F_1	0.973	0.9549	0.9616	0.9619
F_2	0.9437	0.9278	0.8913	0.9235
F_3	0.9506	0.9344	0.9268	0.9253
F_4	0.9557	0.948	0.9405	0.9462
F_5	0.9673	0.9467	0.955	0.9625
F_6	0.9558	0.931	0.9424	0.9318
F_7	0.9782	0.9654	0.9672	0.9675

TABLE IV
COMPARATIVE RESULTS OF THE TESTED KEY-FRAME EXTRACTION ALGORITHMS USING SRD MEASURE ON DATASET A

SRD	Algorithm			
Frame Seq.	Our method	K-means	Method in [11]	Spectral
F_1	1859.66	1533.34	1693.1	1620.6
F_2	424.72	369.87	292.43	362.64
F_3	502.76	430.78	374.23	431.32
F_4	528.09	356.46	340.89	393.02
F_5	843.10	808.2	758.23	780.33
F_6	855.44	753.75	813.1	791.2
F_7	707.92	648.71	642.97	663.15

the proposed estimation algorithm (Section II-B). The second technique used for comparison is presented in [11], as a variant of the method proposed in [22]. Initially, the middle frame of the video sequence is selected as the first key-frame and added to the empty set of key-frames KF. Next, each frame in the video sequence is compared with the current set of key-frames. If it differs from every key-frame in the current set, then it is added into the set as a new key-frame. This algorithm uses a threshold to discriminate whether two frames are similar or not. In our experiments this threshold parameter is set to such a value that the number of key-frames extracted is the same as in our algorithm. Finally, the third technique is the typical spectral clustering algorithm [8], described in Section II-A and employing the simple k-means algorithm.

To evaluate the results of the extracted key-frames we use the metrics mentioned above. More specifically in Tables III–IV we present the performance results for the ASF and SRD measures, respectively. To compute the SRD we use a simple linear interpolation algorithm on the frame's features [3]. The dataset A, which contains high camera and object motion, is used to show the effectiveness of our algorithm in cases where many key-frames are required to represent the shot. It is clear that our approach provides the best summarization of each shot compared to the other methods and the best reconstruction of the original video sequence from the extracted key-frames.

We also tested our key-frame extraction algorithm and compared it with the other methods using dataset B (Tv-series and movies). The measures we used are : i) Average Video Fidelity, which is the mean of Average Shot Fidelities of each video and ii) Average SRD, which is the mean of the SRD of the shots of each video. In Figs. 8 and 9, we present the Average Video Fidelity and the Average SRD respectively. It is obvious that our key-frame extraction algorithm provides better shot reconstruction and representation than the other three methods.

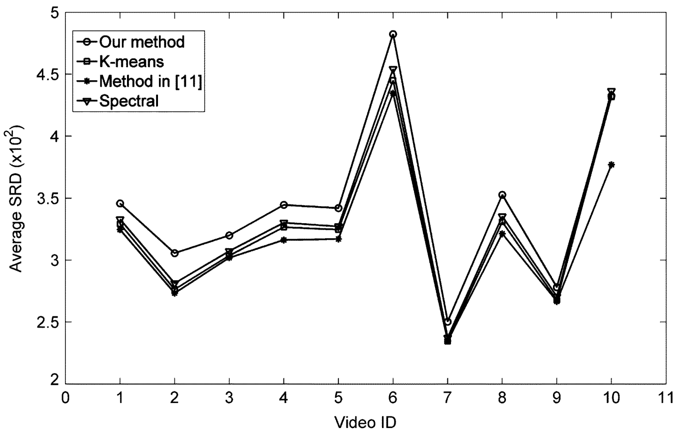

Fig. 9. Comparative results of the tested key-frame extraction algorithms using Average SRD measure on dataset B.

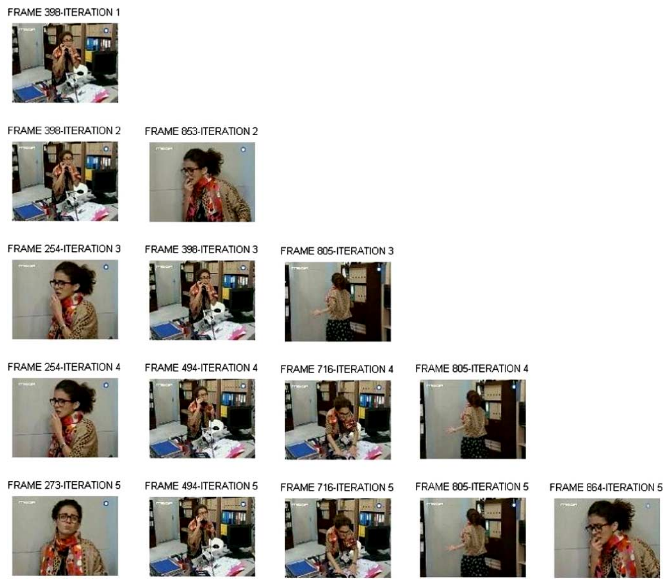

Fig. 10. Key-frame extraction using the proposed approach of a shot with object and camera motion ($N_{\mathrm{kf}} = 5$).

4) Representation: As already mentioned (Section II-A), a great benefit of the fast global k-means algorithm is that it provides the solutions for all intermediate k-clustering problems with $k \leq K$. In Fig. 10 we give an example of the extracted key-frames of a video shot with object and camera motion. Moving from the top to the bottom of this figure we show all intermediate solutions until the selected number of key-frames $N_{\mathrm{kf}} = 5$ is reached. The shot that we used contains 633 frames (frame sequence F_1). It shows a woman in an office setup. This shot can be semantically divided into five subshots. a) The woman stands against a door eavesdropping and then rushes to her office to pick up the phone that is ringing; b) she talks on the phone, c) lays the receiver of the phone down with a visible effort not to make any noise, d) she rushes back to the door, and e) she continues eavesdropping.

In Fig. 11 we provide the key-frames extracted performing the simple k-means algorithm, the algorithm in [11] and the typical spectral clustering algorithm. All algorithms fail to provide a solution adequately describing the visual content of the shot,

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

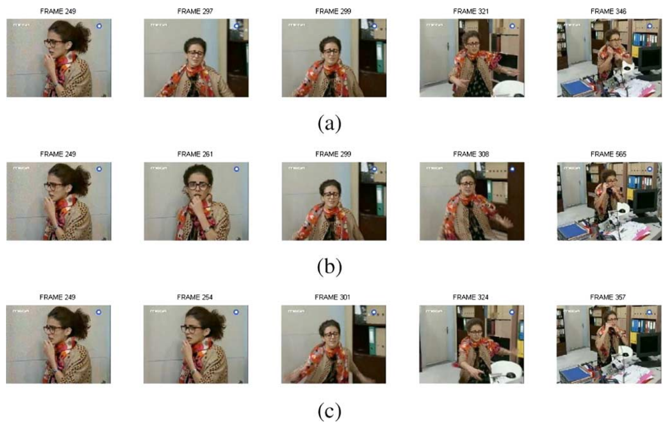

Fig. 11. Results for the key-frame extraction algorithms used for comparison with ($N_{\rm kf} = 5$). (a) K-means. (b) Method in [11]. (c) Spectral Clustering employing simple k-means.

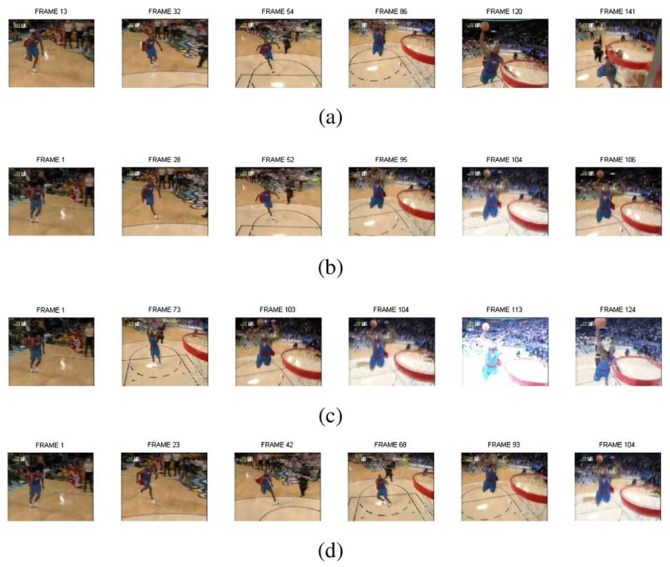

Fig. 12. Key-frame extraction algorithms in comparison in basketball sequence. (a) Our method. (b) K-means. (c) Method in [11]. (d) Spectral Clustering employing simple k-means.

whereas our approach provides a sensible solution. More specifically, they do not produce any frames for subshots (c), (d), and (e) and instead produce multiple frames for subshot (a). In contrast the proposed approach produces key frames for all subshots.

In Fig. 12 we provide the key-frames for these four algorithms for a video shot describing a slam dunk attempt (frame sequence F_3). It becomes clear that our algorithm summarizes the attempt from the beginning to the end, whereas the other three fail to describe the end of the action.

C. Scene Detection Experiments

1) Performance Criteria: To evaluate the performance of our method we used the following criteria [1]:

$$\text{Recall} = \frac{N_c}{N_c + N_m}, \quad \text{Precision} = \frac{N_c}{N_c + N_f}$$
$$F_1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{23}$$

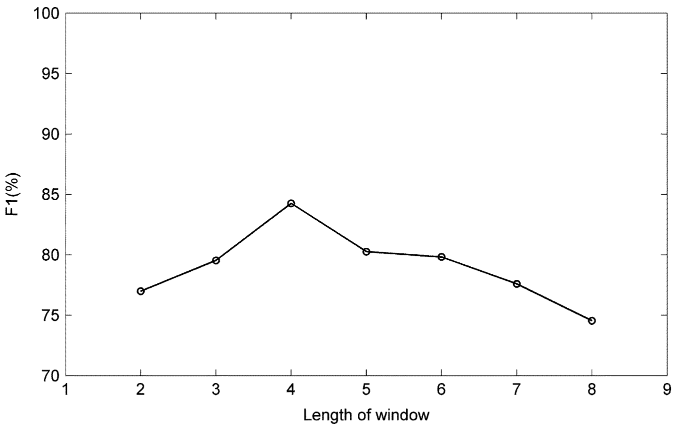

Fig. 13. Average performance results for different values of the window parameter.

where N_c stands for the number of correct detected scene boundaries, N_m for the number of missed ones and N_f the number of false detections.

2) Results: In Fig. 13, the average performance of our algorithm on all videos is presented, varying the length of the window w, (which defines the length of the sequences to be aligned) from 2 to 8. It can be observed that even for $w = 8$, the algorithm yields very good results. We believe that the choice of $w = 4$ is preferable because, apart from reducing the possibility of missing a scene with a small number of shots, it is sufficiently large for a reliable comparison during the sequence alignment algorithm.

To detect the final scene boundaries, as already mentioned in Section III-B, we select the local minima of the SC sequence that are less than a percentage Th of its global minimum. In Fig. 14, the average F_1 values (for $w = 4$) for all videos are presented, for Th varying from 0.7 to 0.95. It can be observed that for any Th from 0.7 to 0.85 our algorithm provides very good result achieving the best performance for $Th = 0.8$. In Fig. 15 we present the F_1 values for $w = 4$ and $Th = 0.8$ varying the weight parameter α, which controls the contribution of the matrices CSM and PPM, from 0 to 1. The best performance is achieved for $\alpha = 0.5$. It can be observed that for $\alpha = 1$ the performance is very low, thus indicating that the use of the PPM matrix is beneficial. In Table V we present the recall, precision and F_1 values for $w = 4, Th = 0.8$ and $\alpha = 0.5$. It can be observed that our approach achieves high correct detection rate while keeping small the number of false detections.

To demonstrate the efficiency of the string comparison method, we also implemented another approach where subsequences are simply considered as *sets* of labels and their similarity is measured using the similarity of the corresponding histograms of labels. In Fig. 16 we present the F_1 values comparing the set comparison and our method (string comparison using sequence alignment). It is clear that the structure of the label sequence assists in the scene detection problem.

3) Comparison: To compare the effectiveness of our approach, we have also implemented two other methods. The first one is proposed in [11]. This method computes both color and motion similarity between shots and the final similarity value is

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

TABLE V
COMPARATIVE RESULTS OF THE TESTED SCENE DETECTION ALGORITHMS USING RECALL, PRECISION AND F_1 MEASURES

	Our method(w = 4, α = 0.5, Th = 0.8)			Method in [11]			Method in [17]		
Video	Recall(%)	Precision(%)	F_1(%)	Recall(%)	Precision(%)	F_1(%)	Recall(%)	Precision(%)	F_1(%)
V_1	86.67	92.85	89.70	86.67	61.90	72.22	60.00	81.82	69.23
V_2	100.00	90.00	94.74	83.33	62.50	71.43	72.22	68.42	70.27
V_3	87.50	73.68	80.00	81.25	52.00	63.41	87.50	70.00	77.78
V_4	76.92	83.33	80.00	92.31	60.00	72.73	76.92	71.43	74.07
V_5	85.71	92.31	88.89	92.86	63.16	75.18	78.57	64.71	70.97
V_6	82.35	93.33	87.50	76.47	61.90	68.42	70.59	66.67	68.57
V_7	86.67	81.25	83.87	86.67	61.90	68.42	80.00	75.00	77.42
V_8	76.00	95.00	84.44	80.77	70.00	75.00	71.43	74.07	72.73
V_9	72.00	75.00	73.47	80.77	55.26	65.63	64.00	59.26	61.54
V_{10}	70.00	93.33	80.00	75.00	75.00	75.00	68.42	72.22	70.27
Mean	82.38	87.01	84.26	83.61	62.36	70.74	72.97	70.36	71.29
Standard deviation	8.46	7.64	5.85	5.72	6.19	3.92	7.65	5.86	4.45

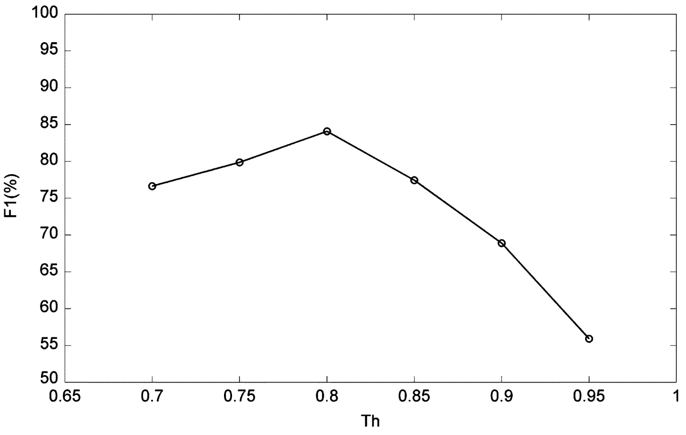

Fig. 14. Average performance results for different values of the Th parameter and $w = 4$.

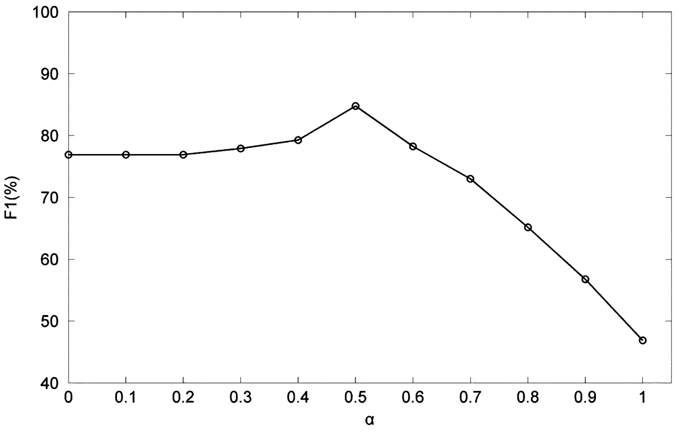

Fig. 15. Average performance results for different values of the a parameter and $w = 4, Th = 0.8$.

weighted by a decreasing function of the temporal distance between shots given by the following equation:

$$w_t(i,j) = e^{-\frac{1}{d}|\frac{m_i - m_j}{\sigma}|^2} \tag{24}$$

where m_i and m_j are the time indices of the middle frames of the two shots under consideration and σ the standard deviation of the shots duration in the entire video. The parameter d plays a critical role in the final number of scenes produced by the algorithm. The final shot similarity matrix defines a weighted

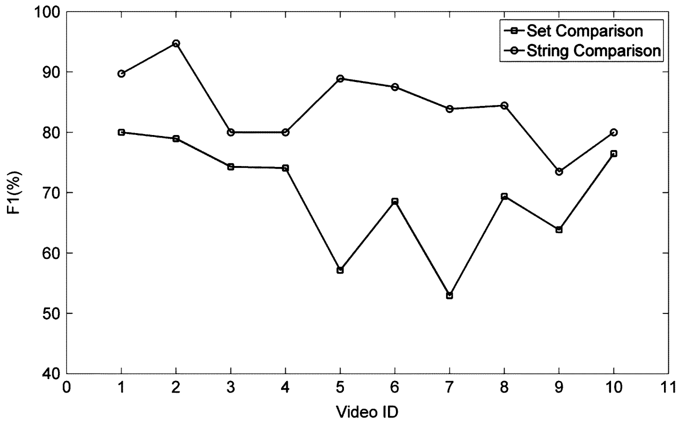

Fig. 16. Scene detection results (using F_1 measure) when subsequences are considered as i) strings (compared using sequence alignment) and ii) sets of labels (compared using histogram similarity).

undirected graph where each node represents a shot and the edges are the elements of the matrix. To partition the video into scenes, an iterative application of Normalized cuts method [13] was used that divides the graph into subgraphs. It must be noted that the implementation of the Normalized cuts method in this approach does not require the computation of eigenvectors, because scenes are composed of shots which are time continuous. Thus a cut can be made along the diagonal of the shot similarity matrix. The *Ncut* algorithm is applied recursively as long as the *Ncut* value is below some stopping threshold T. We have implemented and tested this method using the same video set for different values of the threshold parameter T and the parameter d (24). Determination of optimal values for these parameters is a tedious task. In our comparisons we found distinct values for each video that provide the best performance. The recall, precision and the F_1 values of the experiments are presented in Table V.

The second method has been proposed in [17]. This method clusters shots into groups taking into account the visual characteristics and temporal dynamics of video. Then, a *scene transition graph* which is a graphical representation of the video is constructed. The nodes of this graph represent the shots and the edges the transitions between the shots. To find the scenes, this graph is partitioned into connected subgraphs. The above algorithm depends on two parameters. The first one is the parameter

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

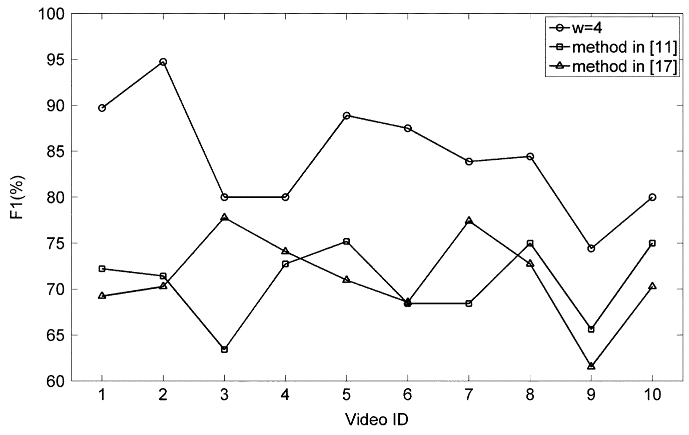

Fig. 17. Scene detection results (using F_1 measure) comparing three scene detection algorithms.

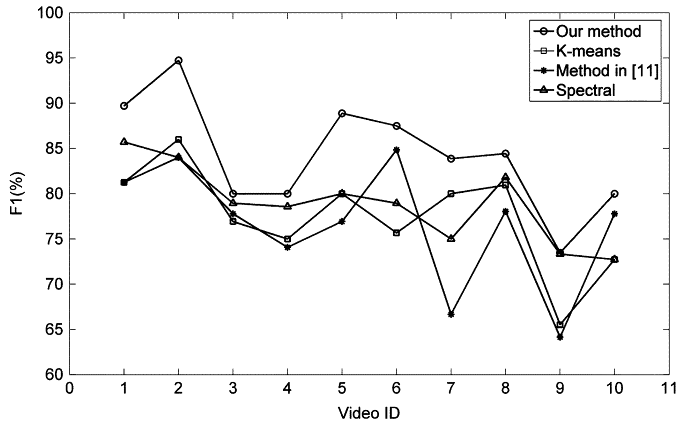

Fig. 18. Scene detection results (using F_1 measure) comparing four key-frame extraction algorithms.

δ which defines the minimum separation between any two resulting clusters and controls the final number of clusters. The second parameter is T that defines two shots to belong in different clusters if they are not close to each other. After the initial segmentation, the segmented scenes are refined by adjusting the threshold parameter T to reflect the duration of scenes. Determination of optimal values for the parameters δ and T is a tedious task. To test the performance of this algorithm we executed multiple runs using different values for the parameters δ and T. In our comparisons we used distinct values for each video that provide the best performance. The recall, precision and the F_1 values of the experiments are presented in Table V.

In Fig. 17, the F_1 values of the three examined methods are graphically presented. It is clear that our algorithm provides the best F_1 value for all videos, and in general our method outperforms the other approaches.

Finally, to show that a sensible representation of a shot by its key-frames contributes to the scene detection problem, we carried out the following experiment. We implemented our scene detection algorithm using as key-frames for the shots those extracted from our method and the other three methods mentioned in Section IV-B. The F_1 values of the four examined methods are presented in Fig. 18. It is obvious that the better the shot is represented by its key-frames, the better our algorithm detects scene boundaries.

All three algorithms were implemented in Matlab. Considering the scene detection problem for the first video sequence, our algorithm and the method in [11] took approximately the same time to identify the scene boundaries, whereas the method in [17] took approximately five times more than the first two.

V. Conclusion

In this paper a new method for video scene segmentation has been proposed. First key-frames are extracted using a spectral clustering method employing the fast global k-means algorithm in the clustering phase and also providing an estimate for the number of the key-frames. Then shots are clustered into groups using only visual similarity as a feature and they are labeled according to the group they are assigned. Shot grouping is achieved using the same spectral clustering method proposed for key-frame extraction.

After shot grouping, shots are labeled according to the cluster they are assigned. Since a typical scene contains a sequence of similar shot labels or a sequence of repetitive label patterns of two or more different groups of shots, when a change in the pattern occurs, we consider that a scene boundary also occurs. To identify such changes, we considered windows of shot sequences which are compared using the "Needleman-Wunsch" sequence alignment algorithm [7]. Thus our approach treats time adjacency in a distinct processing phase while existing methods use temporal distance between shots in the definition of the similarity matrix that is subsequently used as input to the clustering procedure. The presented experimental results on several videos indicate that the proposed method accurately detects most scene boundaries, while providing a good trade off between recall and precision.

A drawback of most algorithms including our own is the over-segmentation that occurs in cases where there is a continuous change in the visual content of shots in a scene. In future work, we will try to improve the performance of our method in order to treat more effectively the case of videos where the shot visual content changes continuously.

References

[1] A. Del Bimbo, *Visual Information Retrieval*. San Francisco, CA: Morgan Kaufmann, 1999.
[2] H. S. Chang, S. Sull, and S. U. Lee, "Efficient video indexing scheme for content-based retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 8, pp. 1269–1279, 1999.
[3] C. Gianluigi and S. Raimondo, "An innovative algorithm for key frame extraction in video summarization," *Journal of Real-Time Image Proceesing*, vol. 1, no. 1, pp. 69–88, 2006.
[4] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "Automated high-level movie segmentation for advanced video-retrieval systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 4, pp. 580–588, June 1999.
[5] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*. Cambridge, MA: MIT Press, 2004.
[6] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.
[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Molec. Biol.*, vol. 48, pp. 443–453, 1970.

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Neural Info. Processing Systems*, 2001, (NIPS 2001).

[9] J.-M. Odobez, D. Gatica-Perez, and M. Guillemot, "Spectral structuring of home videos," in *Proc. Int. Conf. Image and Video Retrieval*, 2003, pp. 310–320.

[10] Z. Rasheed and M. Shah, "Scene detection in hollywood movies and TV shows," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2003, (CVPR 2003).

[11] Z. Rasheed and M. Shah, "Detection and representation of scenes in videos," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1097–1105, Dec. 2005.

[12] I. K. Sethi and N. Patel, "A statistical approach to scene change detection," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases III*, 1995, vol. 2420, pp. 329–339.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[14] H. Sundaram and S. F. Chang, "Computable scenes and structures in films," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 482–491, 2002.

[15] M. J. Swain and D. M. Ballard, "Colour indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[16] L. Tieyan, X. Zhang, J. Feng, and K. T. Lo, "Shot reconstruction degree: A novel criterion for key frame selection," *Pattern Recognit. Lett.*, vol. 25, pp. 1451–1457, 2004.

[17] M. Yeung, B. Yeo, and B. Liu, "Segmentation of videos by clustering and graph analysis," *Comput. Vis. Image Understand.*, vol. 71, no. 1, pp. 94–109, July 1998.

[18] E. P. Xing and M. I. Jordan, On Semidefinite Relaxation for Normalized k-Cut and Connections to Spectral Clustering Computer Science Division, Univ. California, Berkeley, Tech. Rep. CSD-03-1265, 2003.

[19] H. Zha, C. Ding, M. Gu, X. He, and H. Simon, "Spectral relaxation for k-means clustering," in *Neural Info. Processing Systems*, 2001, (NIPS 2001).

[20] Y. Zhai and M. Shah, "Video scene segmentation using Markov chain Monte Carlo," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 686–697, 2006.

[21] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Syst.*, vol. 1, no. 1, pp. 10–28, 1993.

[22] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *Proc. IEEE Int. Conf. on Image Processing*, 1998, pp. 866–870.

Vasileios T. Chasanis received the Diploma degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2004. He is currently pursuing the Ph.D. degree at the Department of Computer Science, University of Ioannina, Greece.

His research interests are in the areas of machine learning for video analysis and summarization, multimedia information retrieval and real time video surveillance.

Aristidis C. Likas (S'91–M'96–SM'03) received the Diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering both from the National Technical University of Athens, Greece.

Since 1996, he has been with the Department of Computer Science, University of Ioannina, Greece, where he is currently an Associate Professor. His research interests include neural networks, machine learning, multimedia data analysis and bioinformatics.

Nikolaos P. Galatsanos received the Diploma of Electrical Engineering from the National Technical University of Athens, Greece, in 1982. He received the M.S.E.E. and Ph.D. degrees from the Electrical and Computer Engineering Department, University of Wisconsin-Madison, in 1984 and 1989, respectively.

He was on the faculty of the Electrical and Computer Engineering Department at the Illinois Institute of Technology, Chicago (1989–2002,) and of the Computer Science Department at the University of Ioannina, Greece (2002–2008). Presently, he is a Professor with the Department of Electrical and Computer Engineering, University of Patras, Greece. His research interests center on image processing and statistical learning problems for medical imaging and visual communications applications.

Dr. Galatsanos has served as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE *Signal Processing Magazine*. He has coedited the book *Image Recovery Techniques for Image and Video Compression and Transmission* (Kluwer Academic, October 1998).

Authorized licensed use limited to: University of Ioannina. Downloaded on January 21, 2009 at 06:27 from IEEE Xplore. Restrictions apply.