Available online at www.sciencedirect.com

SCIENCE DIRECT®

Image and Vision Computing 24 (2006) 131–142

www.elsevier.com/locate/imavis

A motion-based scene tree for compressed video content management

Haoran Yi, Deepu Rajan *, Liang-Tien Chia

Center for Multimedia and Network Technology, School of Computer Engineering, Nanyang Technological University, Singapore 639798

Received 24 July 2004; received in revised form 29 July 2005; accepted 9 September 2005

Abstract

This paper describes a fully automatic content-based approach for browsing and retrieval of MPEG-2 compressed video. The first step of the approach is the detection of shot boundaries based on motion vectors available from the compressed video stream. The next step involves the construction of a scene tree from the shots obtained earlier. The scene tree is shown to capture some semantic information as well as provide a construct for hierarchical browsing of compressed videos. Finally, we build a new model for video similarity based on global as well as local motion associated with each node in the scene tree. To this end, we propose new approaches to camera motion and object motion estimation. The experimental results demonstrate that the integration of the above techniques results in an efficient framework for browsing and searching large video databases.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Shot boundary detection; Video indexing; Video browsing; Video similarity; Video retrieval

1. Introduction

State-of-the-art video compression and communication technologies have enabled a large amount of digital video to be available online. Storage and transmission technologies have advanced to a stage so that they can accommodate the demanding volume of video data. Encoding technologies such as MPEG, H.263 and H.264 [1–4] provide for access to digital videos within the constraints of current communications infrastructure and technology. Even production of digital videos has become available to the masses with introduction of high performance, low-cost digital capture and recording devices. As a result, a huge volume of digital video content is available in digital archives on the World Wide Web, in broadcast data streams, and in personal and professional databases. Such a vast amount of content information calls for effective and efficient techniques for finding, accessing, filtering and managing video data. While search engines and database management systems suffice for text documents, they simply cannot handle the relatively unstructured, albeit information rich, video content. Hence, building a content-based video indexing system turns out to be a difficult problem. However, we can identify three tasks that are fundamental to building an efficient video management system: (i) the entire video sequence must be segmented into shots, where a shot is defined as a collection of frames recorded from a single camera operation. This is akin to a tuple, which is the basic structural element for retrieval in conventional text-based database management system. (ii) Even though a shot determines a physical boundary in a video sequence, it does not convey any meaningful semantics within it. Hence, shots that are related to each other must be grouped together into a scene [8–10]. (iii) Finally, a robust retrieval method depends on a model that captures similarity in the semantics of video content.

In this paper, we address the above fundamental tasks to provide an integrated approach for managing *compressed* video content. Since video is mostly available in compressed form, there is a need to develop algorithms to process compressed video directly without paying for overheads to decode them before processing. Tasks such as restoration, resolution enhancement, tracking, etc. in compressed videos have been reported recently [11–14]. Our objective is to develop a fully automatic technique for content-based organization and management of MPEG compressed videos. To this end, the paper

(1) describes a novel shot boundary detection algorithm that is capable of detecting both abrupt and gradual shot changes like dissolve, fade-in/fadeout;
(2) describes a scene tree that acts as an efficient hierarchical structure to facilitate browsing;

* Corresponding author. Tel.: +65 6790 4933; fax: +65 6792 6559.
E-mail addresses: pg03763623@ntu.edu.sg (H. Yi), asdrajan@ntu.edu.sg (D. Rajan), asltchia@ntu.edu.sg (L.-T. Chia).

0262-8856/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.09.019

(3) presents a video similarity model that enables efficient indexing of video content based on motion and which is shown to be useful in video retrieval.

As noted earlier, these three tasks provide for an integrated approach to browsing and retrieval in large video databases. This is in contrast to video abstraction [5] and video summarization [6], which provides a linear (sequential) representation of the sequence, but is insufficient for interactive browsing and retrieval of video. These processes entail extraction of 'important' parts within a video, while discarding the rest, akin to text summarization in documents. In doing so, relevant information needed for retrieval may also be lost. Furthermore, they do not support nonlinear access to the video content. Hence, a hierarchical structure that facilitates non-linear access is needed to build a video database search engine as in Ref. [7]. However, the authors in Ref. [7] use a fixed hierarchical structure as opposed to the adaptive structure proposed in this paper. Moreover, their structure lacks temporal information of the video sequence since it is obtained by clustering shots using only visual similarity.

The remainder of this paper is organized as follows. In Section 2, we describe a novel shot boundary detection algorithm in the compressed domain. The procedure for building an adaptive scene tree is described in Section 3. In Section 4, the motion-based indexing and retrieval techniques are discussed. The experimental results are presented in Section 5. Finally, we give concluding remarks in Section 6.

2. Shot boundary detection algorithm

There is a rich literature of algorithms for detecting shot boundaries in video sequences. They can be broadly classified into methods that use one or more of the following features extracted from the video frames: (i) pixel differences; (ii) statistical differences; (iii) histogram; (iv) compression differences; (v) edge tracking and (vi) motion vectors [15]. In the initial phase of research in shot boundary detection (SBD), the data mostly consisted of uncompressed video [16–18]. However, with the emergence of the MPEG compression standard, it has become prudent to develop algorithms to parse MPEG video directly. Several methods that detect shot boundaries directly from compressed video have also been reported [19–22]. Most of these methods use either the DC coefficients or the bit-rate of different MPEG picture types (I, P, or B) in their algorithms; the methods using DC coefficients involve significant decoding of the MPEG-compressed video, while the methods using bit-rate do not yield satisfactory detection. Moreover, these methods are incapable of handling both hard cuts and gradual transitions (i.e. fade, dissolve, etc.) simultaneously. Lastly, most of them involve tight thresholds that are largely dependent on the video sequence. In this section, we describe a method of detecting not only hard cuts but also gradual transitions [23]. We exploit the number of macroblock (MB) types in each of the I-, P-, and B-frames to derive the SBD algorithm.

2.1. Abrupt shot change detection

We note that not all the macroblocks (MBs) in a P-frame are forward motion predicted and not all the macroblocks in a B-frame are bidirectionally predicted [22]. Each MB in the P-frame is either forward predicted, skipped or intracoded. Similarly, each MB in the B-frame is either forward predicted, backward predicted, bi-directionally predicted, skipped or intracoded. However, a skipped MB is encoded in the same way as the first MB in the slice to which it belongs [24]. Hence, we are left with basically four types of MBs which we abbreviate as *In*, *Fw*, *Bk* and *Bi* for intracoded, forward predicted, backward predicted and bidirectionally predicted, respectively.

The number of each type of MBs in a frame is an indication of the similarity/dissimilarity of that frame with its neighboring frames. Since only the B-frame has all the four types of MBs, we use it to compare the dissimilarity through a frame dissimilarity ratio (FDR), which is defined as

$$\mathrm{FDR}_n = \begin{cases} \dfrac{Fw_{n-1}}{Bi_{n-1}} & \text{for Reference frame} \\ \max\left(\dfrac{Fw_n}{Bi_n}, \dfrac{Bk_n}{Bi_n}\right) & \text{for } B \text{ frame} \end{cases} \tag{1}$$

where the Reference frame refers to either an I-frame or a P-frame and the index n denotes the frame number. If a shot change takes place at a reference frame, most of the MBs in the previous frame (which has to be a B-frame) are predicted from the previous reference frame. In other words, *Fw* in the previous frame will be high resulting in a high FDR. On the other hand, if the shot change takes place in a B-frame, all the frames lying between the previous and the following reference frames are either forward predicted or backward predicted. If the number of bi-directionally predicted MBs is small, it once again results in a high FDR for these B-frames. The advantage of using only the B-frames for comparison is that we can avoid the need for normalizing the FDR as well as choosing a different threshold on the FDR for each type of frame. However, we observe that if a shot change takes place at a B-frame, all the B-frames lying between the previous and the next reference frames will have high FDRs. Consider the following frame structure in an MPEG bit stream: … $I_1B_2B_3B_4P_5B_6B_7B_8P_9$ …. If the shot change takes place at B_3, FDRs for B_2, B_3 and B_4 will be very high. In order to determine the exact location of the shot boundary, we observe that B_2 is mostly forward predicted while B_3 and B_4 are mostly backward predicted. Thus, at the shot boundary there is a change in the dominant MB type of the B-frame. So, we define a dominant MB change (DMBC) for frame n as

$$\mathrm{DMBC}_n = \begin{cases} 1 & \\ 0 & \text{if } (Bk_n - Fw_n)(Bk_{n-1} - Fw_{n-1}) > 0 \\ 1 & \text{if } (Bk_n - Fw_n)(Bk_{n-1} - Fw_{n-1}) \le 0, \end{cases} \tag{2}$$

where the first line in the above equation applies to the reference frames and the other two lines apply to the B-frames. The modified FDR (MFDR) for frame n is then defined as

$$\mathrm{MFDR}_n = \mathrm{FDR}_n \times \mathrm{DMBC}_n \tag{3}$$

While the MFDR is the same as the FDR for a reference frame, there will be one or two consecutive high MFDRs for a shot change in a B-frame. If there is only one high MFDR, the corresponding frame indicates the beginning of a new shot. If there are two consecutive high MFDRs, the former indicates the end of a shot and the latter indicates the beginning of a new shot. In the example frame structure considered earlier, the MFDR is retained for frames B_2 and B_3, while it drops to zero for frame B_4.

An adaptive threshold mechanism based on a sliding window as proposed in Ref. [25] is used to detect hard cuts from the MFDR. The MFDR values are compared over a window whose length l is set to be less than the duration of a shot, typically 10 frames. The presence of an abrupt change is detected at each window position, in the middle of the window, using the following adaptive threshold:

$$T = \frac{\max(\mathrm{MFDR}_n, \ldots, \mathrm{MFDR}_{n+l})}{\max(T_g,\ \mathrm{submax}(\mathrm{MFDR}_n, \ldots, \mathrm{MFDR}_{n+1}))} \geq \alpha \tag{4}$$

Here, max and submax refer to the largest and the third largest values, respectively, of their arguments, the parameter α can be thought of as a shape parameter of the boundary pattern characterized by an isolated sharp peak in a series of discontinuity values [26] and T_g is a global threshold which prevents the detected MFDR to be very small. At first glance, it would appear that the presence of α and T_g imposes a tight constraint diluting the adaptive nature of the thresholding process. It is interesting to note, though, that an accurate choice of α and T_g is not crucial to the success of the algorithm. We have observed that an empirical choice of $\alpha = 5$ and $T_g = 1$ works well with a wide variety of video sequences.

2.2. Detection of gradual transitions

We consider two types of gradual transitions, viz., fade and dissolve. A fade is characterized by a gradual darkening of a frame and its replacement by another image, which either fades in or appears abruptly [27]. A fade-in occurs when the image gradually appears from a blank screen and a fade-out occurs when the image disappears gradually, leaving a blank screen. A dissolve occurs when one image fades away while another image appears. It can be viewed upon as a linear transition from the ending frame in one shot to the starting frame in the next shot with pictures from both the shots occupying the region of transition. From this perspective, fade-in and fade-out can be considered as special instances of a dissolve in which case, the former is a dissolve that begins with a blank frame and the latter is a dissolve that ends in a blank frame.

We use the DMBC defined in Eq. (2) to detect dissolves. Consider the following frame structure from an MPEG video: … $I_1B_2B_3P_4B_5B_6P_7$ …. If a dissolve takes place in this sequence, the dominant MB type in frame B_2 will be Fw since B_2 is nearer to the reference frame I_1, and that in frame B_3 will be Bk because it is nearer to P_4. Thus, we observe that during a dissolve the DMBC changes rapidly during a dissolve and consequently a high local sum of the DMBC indicates the presence of a dissolve. If w is the length of a window centered around frame n, then the local sum of the DMBC for that frame is given by

$$F_n = \sum_{n-w}^{n+w} \mathrm{DMBC}_k \tag{5}$$

If F_n is greater than a threshold τ, then frame n belongs to the set of frames that makes up the dissolve. It is heuristically found that the value of τ can be chosen to be 0.8 times the width of the window for several kinds of video sequences. The width of the window is taken to be less than the width of a dissolve, which is typically 10 frames. Thus, we note that the presence of the threshold τ does not render the algorithm inflexible over different types of sequences, as we show in our experiments. A further refinement of the detected gradual transitions can be achieved by merging those transitions whose temporal distance is less than the width of a typical dissolve.

3. Scene tree building algorithm

The objective of content-based indexing of videos is to facilitate easy browsing and retrieval. Ideally, a nonlinear browsing capability is desirable as opposed to standard techniques like fast forward or fast reverse. This can be achieved, preferably, using a structure that represents video information as a hierarchy of various semantic levels. Such a multi-layer abstraction makes it not only more convenient to reference video information but also simplifies video indexing and storage organization. Several multilevel structures have been proposed in the literature [28,9,29,7]. However, they use a fixed structure to describe the video content, i.e. shot and scene. The underlying theme is to group together frames that are ‘similar’ to each other where similarity could be defined in such primitive terms as temporal adjacency [30] or in terms of video content. The latter results in the entity called *scene*, which should convey semantic information in the video being viewed. Hence, our objective is to build a browsing hierarchy whose shape and size are determined only by the semantic complexity of the video. We call such a hierarchical structure as an *adaptive scene tree*. The scene tree algorithm described in this section is motivated by the work of Oh and Hua [31] who build a scene tree for uncompressed videos. Our algorithm is an extension and an improvement over [31] that works for compressed videos.

The main idea of the scene tree building algorithm is to sequentially compare the similarity of the key frame from the current shot to the key frames from the previous w shots. Based on a measure of similarity, the current shot is either appended to the parent node of the previous shot or appended to a newly created node. The result of the scene tree building algorithm is a browsing tree whose structure is adaptive, i.e. the number of levels of the tree are larger for complex content and smaller for simple content. The details of the algorithm, which takes in a sequence of video shots and outputs the scene tree is shown below:

(1) Initialization: create a scene node SN_i^0 at the lowest level (i.e., level 0) of the scene tree for each $shot_i$. The subscript indicates the shot (or scene) from which the scene node is derived and the superscript denotes the level of the scene node in the scene tree;
(2) Initialize $i \leftarrow 3$.
(3) Check if $shot_i$ is similar to shots $shot_{i-1}$, …, $shot_{i\text{-w}}$ (in descending order) using a function *isSimilar*(), which will be described later. The comparisons stop when a similar shot, say $shot_j$, is found. If no related shot is found, a new empty node is created and connected to SN_i^0 as its parent node; then proceed to step 5.
(4) For scene nodes SN_{i-1}^0 and SN_j^0,
    (a) If SN_{i-1}^0 and SN_j^0 do not currently have a parent node, we connect all scene nodes, SN_i^0 through SN_j^0 to a new empty node as their parent node.
    (b) If SN_{i-1}^0 and SN_j^0 share an ancestor node, we connect SN_i^0 to this ancestor node.
    (c) If SN_{i-1}^0 and SN_j^0 do not currently share an ancestor node, we connect SN_i^0 to the current oldest ancestor of SN_{i-1}^0, and then create a new empty node and connect the oldest ancestor of all nodes from SN_j^0 to SN_{i-1}^0 if$(Bk_n - Fw_n)(Bk_{n-1} - Fw_{n-1}) \leq 0$, to it as its children.
(5) If there are more shots, we set $i \leftarrow i+1$, and go to step 3. Otherwise, connect all the nodes currently without a parent to a new empty node as their parent.
(6) For each scene node at the bottom of the scene tree (it represents a shot), we select the key frame as its representative frame by choosing the I frame in a shot whose DC value is closest to the average of the DC values of all the I frames in the shot. We then traverse all the nodes in the scene tree from the bottom to the top. For each empty node visited, we identify the child node, which contains the largest number of frames and assign its representative frame as the representative frame for this node.

We now return to the function *isSimilar*() used in Step 3 to compute the similarity between key frames. The first step is to divide each frame into a fixed background area and a fixed object area, as shown in Fig. 1. The fixed background area is defined as the left, right and top margins of the frame whose width is chosen as 10(of the frame width. The remaining area of the frame is called a fixed object area. Such a partitioning of the frame was proposed in Ref. [31], but the authors used it to develop a SBD algorithm. Operating on the HSV color space, we compute the color histograms of the object and background areas of two key frames k_1 and k_2. The H and S are quantized into 8 bins and 4 bins, respectively, while we ignore the V values since it corresponds to luminance. The similarity

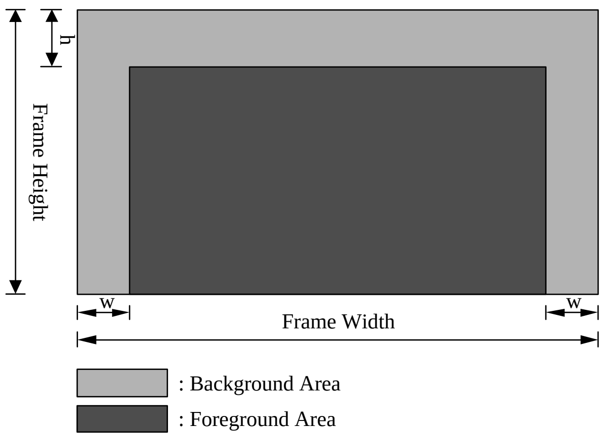

Fig. 1. Background and foreground areas for computing the function *isSimilar*().

measure is defined as

$$\mathrm{Sim} = w_1 \times |\mathrm{hist}_b(k_1, k_2)| + w_2 \times |\mathrm{hist}_o(k_1, k_2)| \tag{6}$$

where $hist_b(.)$ and $hist_o(.)$ are the Euclidean distances of histograms of the background and object areas, respectively, between frames k_1 and k_2 and w_1, w_2 are weightings for the background and object areas, respectively. We choose $w_1 = 0.7$ and $w_2 = 0.3$ so as to give more weight to the background since a change in background area is a stronger indication of a change in the scene. The value of *Sim* in Eq. (6) is compared with a threshold to determine if two key frames are similar.

Although the above adaptive scene tree building algorithm is motivated by Ref. [31], there are some important differences, which we highlight now. As mentioned earlier, our algorithm works directly on MPEG-2 compressed video in the sense that we need only partially decode the video to extract the motion vectors and DC images. Moreover, a full decoding is required only for one frame per shot that represents the key frame. Secondly, the similarity between shots is ascertained through a meaningful predefined window, w, as opposed to all the shots from the beginning of the video. We now compare the computational complexity of our algorithm to Ref. [31]. The shot similarity determination can be done in $O(w \times s)$, where w is the length of the window and s is the number of shots. Generally, the number of frames in a shot, f is much larger than s and s is larger than $w(f >> s > w)$. The scene tree construction algorithm involves traversal in step 4 and step 6. Therefore, the worst-case computational complexity of building the tree is of $O(s \times \log(s))$. Thus, the total computational complexity of our algorithm is of $O(s \times \log(s))$, while that in Ref. [31] is $O(s \times f^2)$.

4. Motion-based indexing and retrieval

In this section, we develop a model to compute the similarity between two video sequences. The model is based on motion content within the video sequences. The motion content itself is represented by three components, viz., the camera motion (CM), the object motion (OM), and the total motion (TM). Each of CM, OM and TM are matrices and it is these matrices that serve as indices for video retrieval.

4.1. Motion extraction

Recall that all the videos considered in this paper are compressed according to MPEG-2 format. Hence, the MV for frame n is obtained as

$$\mathrm{MV}_n = \begin{cases} -Bk_{n-1} & \text{for } I \text{ frame} \\ Fw_n - Fw_{n-1} & \text{for } P \text{ frame} \\ (Fw_n - Fw_{n-1} + Bk_n - Bk_{n-1})/2 & \text{for } B \text{ frame} \end{cases} \qquad (7)$$

Refer to Section 2.1 for meaning of the notations. The MV fields (MVFs) obtained are then smoothened using a spatial (3×3×1) median filter followed by a temporal (1×1×3) median filter. The smoothened MVFs are then used to calculate the camera motion and object motion. The total motion component at each MB is just the smoothened motion vector at its center.

We use a six parameter affine model to estimate the camera motion, which is modeled as

$$mv_x(i,j) = p_1 i + p_2 j + p_3 \qquad mv_y(i,j) = p_4 i + p_5 j + p_6, \qquad 8$$

where $mv_x(i,j)$ and $mv_y(i,j)$ are the x and y components of the motion vector for an MB centered at (i,j), ps are the affine parameters. Consider a group of MBs G whose affine parameters $P=\{p_1,\Lambda,p_6\}$ need to be determined. We will explain the significance of G shortly, when we present the iterative algorithm for camera motion estimation. Using the method of least squares, from Eq. (8), P is obtained by minimizing

$$\sum_G (mv_x(i,j) - p_1 i - p_2 j - p_3)^2 + (mvy(i,j) - p_4 i - p_5 j - p_6)^2 \qquad (9)$$

The two terms inside the summation can be minimized independently. Considering the first term only and differentiating it with respect to p_1, p_2 and p_3 and setting the resulting equations to zero, we get

$$\sum_{(i,j)\varepsilon G} (mv(i,j) - p_1 i - p_2 j - p_3) i = 0 \qquad (10)$$

$$\sum_{(i,j)\varepsilon G} (mv(i,j) - p_1 i - p_2 j - p_3) j = 0 \qquad (11)$$

$$\sum_{(i,j)\varepsilon G} (mv(i,j) - p_1 i - p_2 j - p_3) = 0 \qquad (12)$$

If the origin of the frame is taken to be at its center (instead of the top left corner as is conventionally done), $i' = i-(V+1)/2$ and $j' = j-(U+1)/2$, $U\times V$ is the size of the motion vector field, then $\sum_G i' = 0$ and $\sum_G j' = 0$ and the affine parameters can be easily shown to be

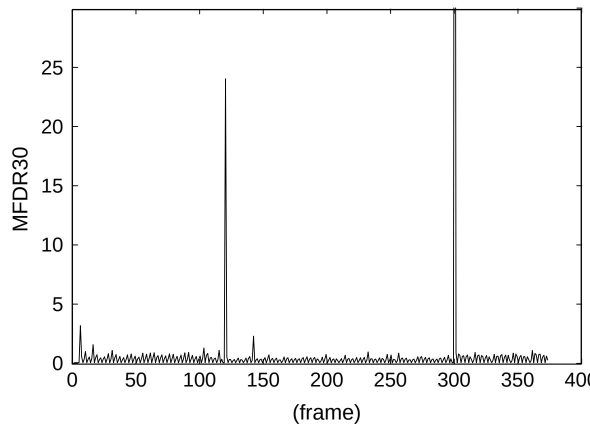

Fig. 2. MFDR for a News Sequences.

$$p_1 = \frac{IX\cdot YY - JX\cdot XY}{A}, \qquad p_2 = \frac{JX\cdot XX - IX\cdot XY}{A},$$
$$p_3 = \frac{\sum_G mv_x(i,j)}{g}, \qquad p_4 = \frac{IY\cdot YY - JY\cdot XY}{A}, \qquad (13)$$
$$p_5 = \frac{JY\cdot XX - IY\cdot XY}{A}, \qquad p_6 = \frac{\sum_G mv_y(i,j)}{g}$$

where

$$IX = \sum_G i'\cdot mv_x(i',j'), \qquad JX = \sum_G j'\cdot mv_x(i',j'),$$
$$IY = \sum_G i'\cdot mv_y(i',j'), \qquad JY = \sum_G j'\cdot mv_y(i',j'),$$
$$XX = \sum_G i'^2, \qquad YY = \sum_G j'^2 \qquad XY = \sum_G i'\cdot j',$$
$$A = XY\cdot XY - XX\cdot YX,$$

and g is the number of macroblocks in the group G. $mv(i',j') = mv(i,j)$ since the they refer to the same MB.

The algorithm to estimate the affine parameters starts by labelling all the MBs in a frame as 'inliers'. Then the

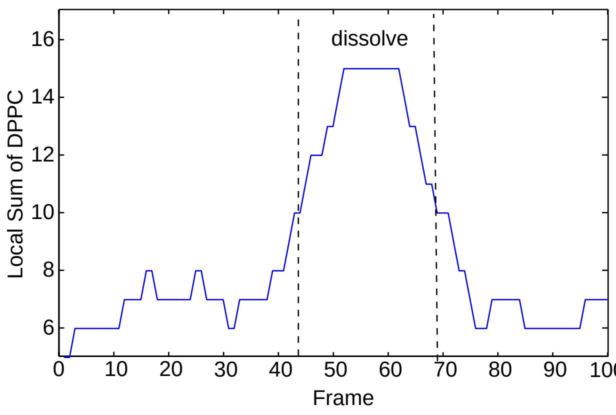

Fig. 3. Local Sum of DMBC in a Dissolve.

Table 1
Summary of SBD results

Video (frames)	Cut (correct, false)	Gradual (correct, false)	Recall/precision cut	Recall/precision gradual
News (32640)	43 (42,4)	12 (11,2)	97.67/91.3	91.67/84.62
Movie (29010)	17 (16,2)	3 (3,1)	94.12/88.89	100/75
Cartoon (1498)	13 (13,2)	0 (0,0)	100/86.67	–
Soccer (21310)	32 (31,2)	10 (9,3)	96.88/93.94	90/75.00
Total (84458)	105 (102,10)	25 (23,6)	97.41/91.75	92.00/79.31

parameters are estimated for all the inliers using Eq. (13). A new set of motion vectors are reconstructed for each inlier using the estimated parameters. If the magnitude of the residual motion vector R_{mv}, calculated as the difference between the original motion vector and the reconstructed one, is greater than an adaptive threshold $T=\max(\mathrm{median}(R_{\mathrm{mv}}), \beta)$ for a particular MB, then that MB and the one situated diagonally opposite it are marked as 'outliers'. The role of β is to prevent the rejection of a large number of MBs if the median of the residuals is very small. We choose β to be 1. Note that the diagonally opposite MB is also marked as outliers due to the shifting of the co-ordinate axes to the center of the frame. After each iteration, some MBs are marked as outliers; these MBs correspond to areas in which the motion is associated with

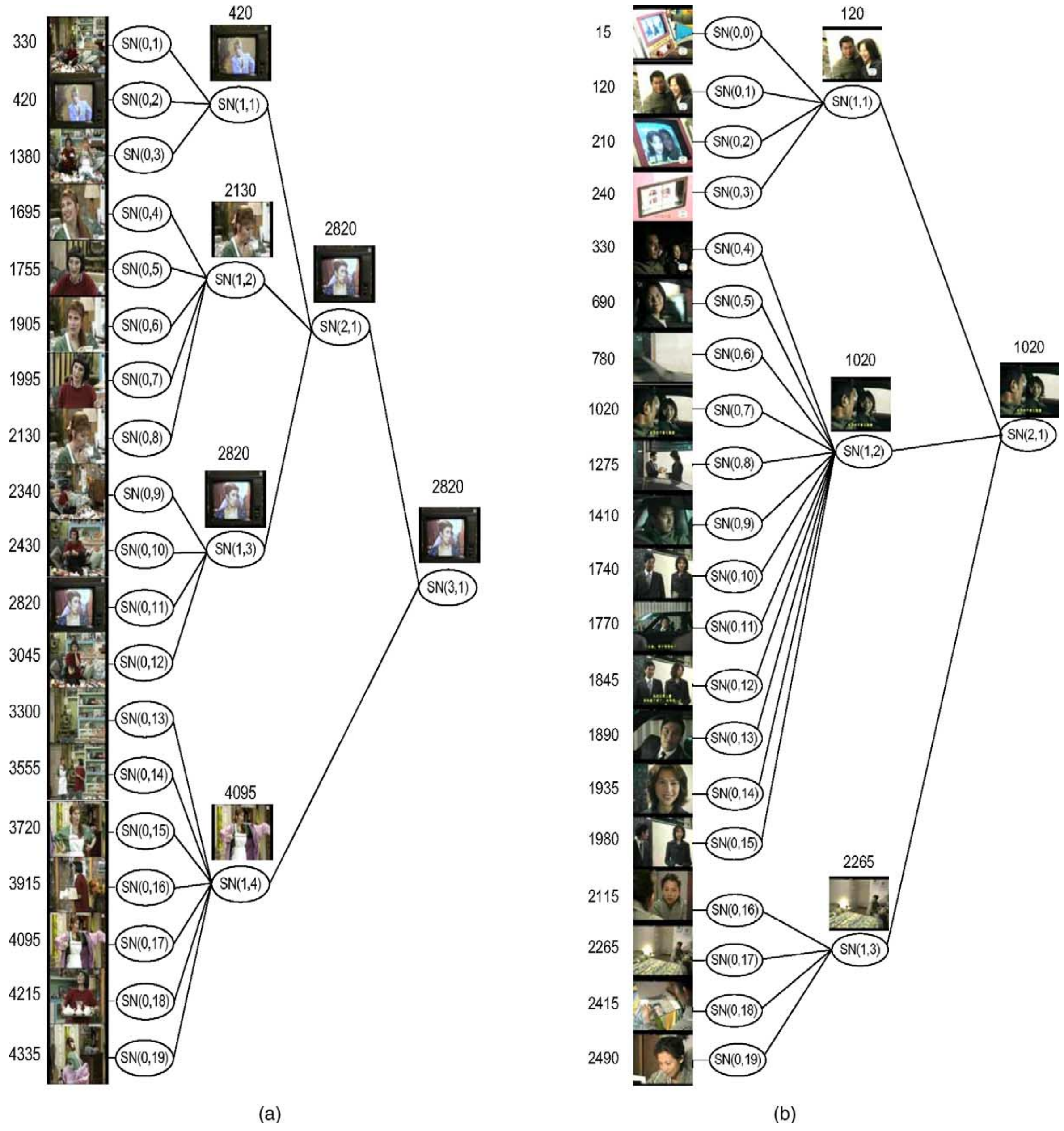

Fig. 4. (a) Scene tree of MPEG-7 test video, (b) scene tree of TV program 'Opening The New Era'.

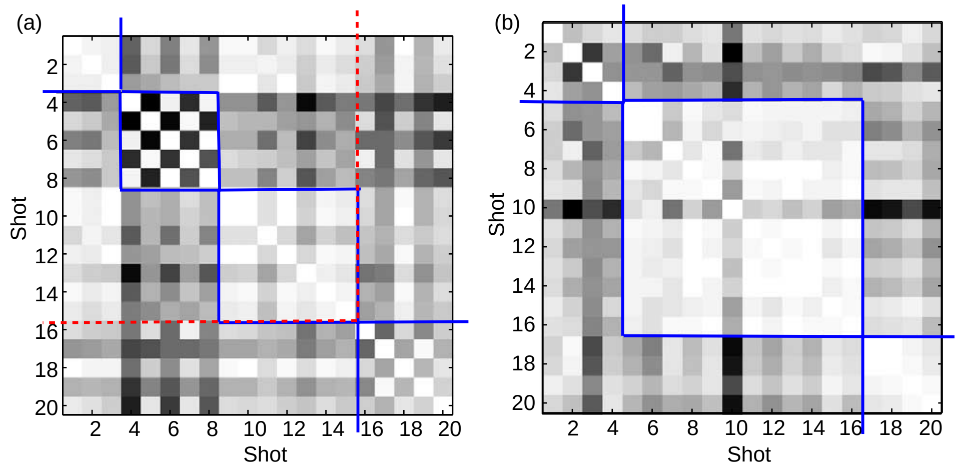

Fig. 5. Shot pairwise similarity matrix for (a) MPEG-7 test video and (b) Opening The New Era video in Fig. 4.

moving objects. The set of macroblocks marked as 'inliers' at any iteration constitute G. The steps in the algorithm to estimate the camera motion is shown in Algorithm 1.

Algorithm 1. Camera Motion Estimation

```
1. Mark all the MBs as inliers.
2. Estimate the motion affine parameters for inliers (Eq. (13)).
3. Reconstruct the global motion vector at each macroblock
   with the estimated affine parameters.
4. Calculate residual motion vector ($R_{\text{mv}}$) as the difference
   between the original and reconstructed motion vector.
5. If $R_{\text{mv}}$ is greater than max (median($R_{\text{mv}}$), $\beta$), then mark this
   MB and its opposite diagonal MB as outliers. Median($R_{\text{mv}}$)
   is the median of all the $R_{\text{mv}}$'s.
6. Go to step 2 until there are no more new outliers or more
   than two thirds of the MBs are marked as outliers.
7. If more than two thirds of the MBs are outliers, the affine
   parameters are set to zero.
```

Since the computed camera motion vector should not be greater than the total motion vector at a MB,

$$|\text{cm}(i,j)| = \min(|\text{mv}(i,j)|, |\text{cm}(i,j)|) \tag{14}$$

where cm(i, j) and mv(i, j) are the camera motion vector and total motion vector, respectively, at the MB centered at (i, j). For the same reason

$$|\text{om}(i,j)| = \max(0, |\text{mv}(i,j) - \text{cm}(i,j)|) \tag{15}$$

where om(i, j) is the object motion vector at the MB centered at (i, j). The CM, OM and TM matrices are now formed for a shot by accumulating |cm(i, j)|, |om(i, j)| and |mv(i, j)|, respectively, over all the frames in the shot, i.e., $\text{CM}(i,j) = \sum_{l=1}^{n} \text{cm}_l(i,j)$, $\text{OM}(i,j) = \sum_{l-1}^{n} \text{om}_l(i,j)$, and $\text{TM}(i,j) = \sum_{l=1}^{n} \text{mv}_l(i,j)$, where n is the number of frames in the shot. These matrices serve as indices for retrieval, as explained in the next sub-section.

4.2. Video similarity measure

Each shot is characterized by the three matrixes CM, OM and TM. In order to compute the similarity between two shots, we assume that the matrices belong to the Hilbert sequence space l^2 with metric defined by

$$d(M_x, M_y) = \left[\sum_{u=1}^{U}\sum_{v=1}^{V} |M_x(u,v) - M_y(m,n)|^2\right]^{1/2} \tag{16}$$

where M_x and M_y are the motion matrices of order $U \times V$. However, instead of comparing matrices, we reduce the dimension of the feature space by projecting the matrix elements along the rows and the columns to form one dimensional feature vectors $M_x^r(v) = \sum_{u=1}^{U} M_x(u,v)$ and $M_x^c(u) = \sum_{v=1}^{V} M_x(u,v)$. The metric in Eq. (16) can then be rewritten as

$$d(M_x, M_y) = \left[\sum_{v=1}^{V} \frac{1}{U} |M_x^r(v) - M_y^r(v)|^2 + \sum_{u=1}^{U} \frac{1}{V} |M_x^c(u) - M_y^c(u)|^2\right]^{1/2}. \tag{17}$$

This leads us to the distance function between two shots s_x and s_y to be defined as

$$\hat{d}(s_x, s_y) = \omega_C d(\text{CM}_x, \text{CM}_y) + \omega_O d(\text{OM}_x, \text{OM}_y) + \omega_T d(\text{TM}_x, \text{TM}_y), \tag{18}$$

where the d(·)s are computed according to Eq. (17) and the ωs are the weights for each motion component. The weights are assigned in such a way that there is equal contribution from each of the components to the distance function.

5. Experimental results

Our experiments were designed to assess the performance of the proposed techniques for SBD, scene tree construction, motion estimation and motion based indexing and retrieval of shots.

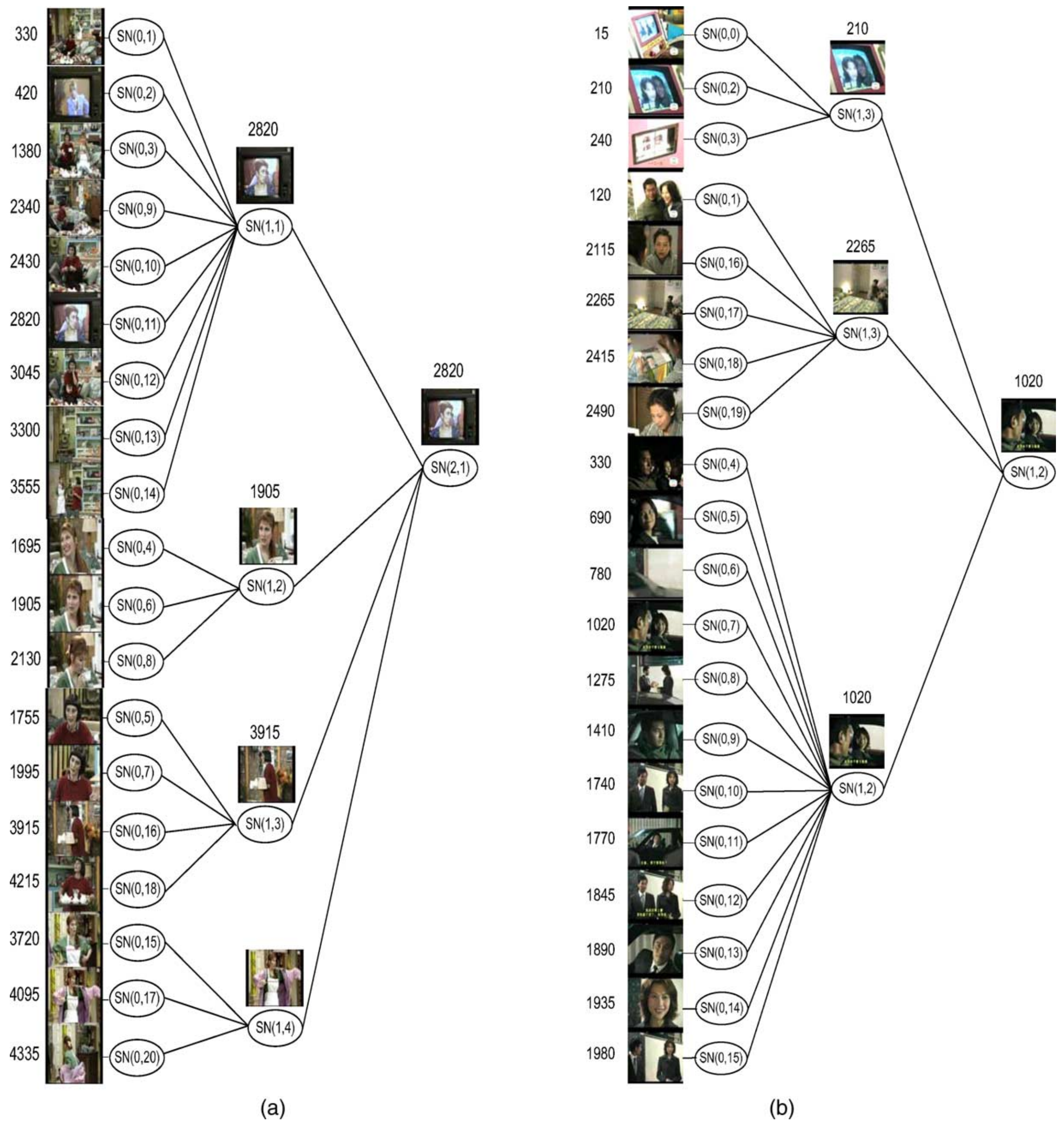

Fig. 6. Video decomposition by clustering for (a) MPEG-7 test video and (b) TV program Opening The New Era [7].

5.1. Shot boundary detection

We have tested our algorithm on a wide variety of video sequences like soccer, news, movie and cartoons, which were encoded in the MPEG-2 format with a GOP length of 12 frames. The encoding pattern was IBBPBBPBBPBB. There were a total of 105 hard cuts and 25 gradual transitions of which 20 were dissolves, 3 were fade-ins and 2 were fade-outs. Fig. 2 shows the MFDR, (see Eq. (3)) for 373 frames of a 'News' video. As expected, the hard cuts at frames 120 and 300 have a very high MFDR, which enables easy detection of the shot boundary. As for gradual transitions, local sum of DMBC, (see Eq. (2)) for a 'Movie' sequence is shown in Fig. 3. We note that the beginning of the dissolve takes place at frame 43 and ends at frame 68. Table 1 summarizes the results of shot boundary detection. In the experiment, the global threshold T_g in Eq. (4) for detecting hard cuts was set to 1. For detecting gradual transitions, the length of the window over which the local sum is computed is set to 12, with τ (the threshold for F_n in Eq. (5)) set to 10. The video sequences are chosen so as to include various kinds of camera and object motions. We use *precision* (the ratio of the number of shot changes detected correctly to the actual number of shot changes) and *recall* (the ratio of the number of shot changes detected correctly over the total number of shot changes detected) to measure the performance of the SBD algorithm.

The recall rates for hard cuts are consistently above 95(over all the sequences reinforcing the fact that the choice of the

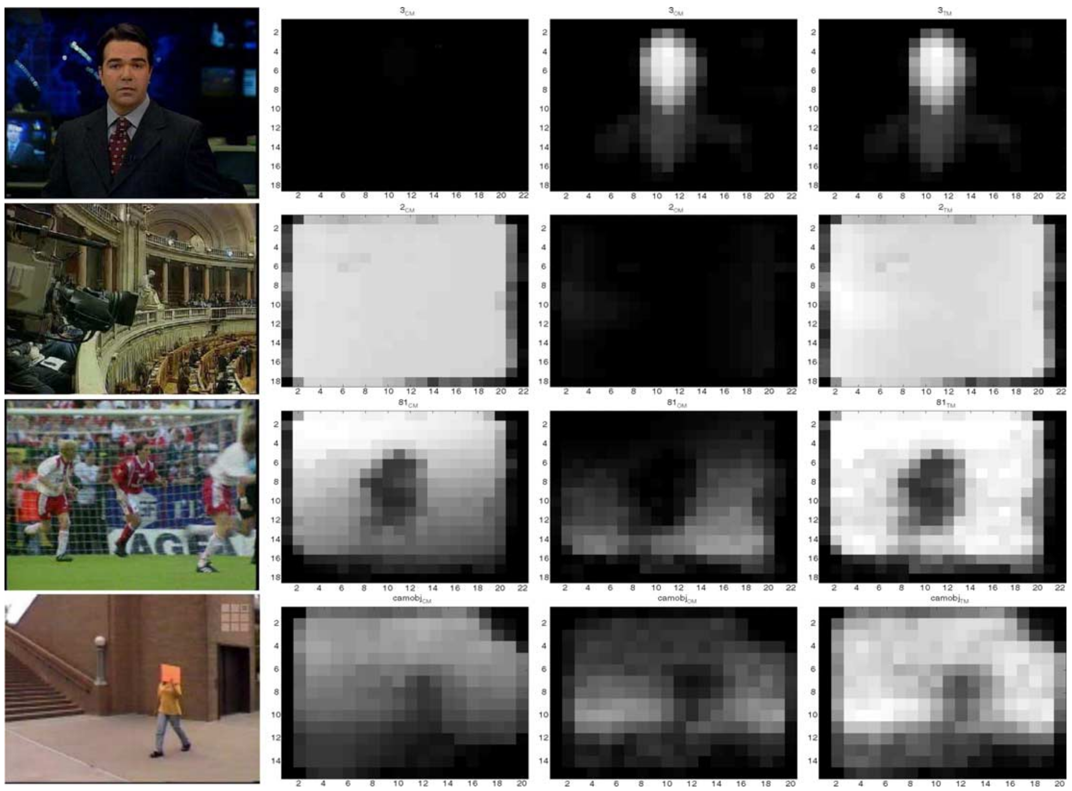

Fig. 7. Motion estimation. Each row consists of a key frame of the shot, the CM image, the OM image and the TM image of News anchor person (first row), gallery (second row), soccer (third row) and moving person (fourth row).

thresholds are not critical to the success of the method. The missed cuts are mostly due to large motion at the shot boundary. The overall recall rate for gradual transitions is 92 (only 2 of the 25 transitions were missed). Note that τ is chosen the same for all the sequences. The false detection of hard cuts is attributed to the sudden change in brightness of the frame due to flash in the News sequence, a lightning bolt in the Movie sequence and the sudden appearance of text in the 'Cartoon' sequence. We must mention that the News sequence has 45 instances of flashes, which were potential candidates for shot boundaries, out of which only 4 were falsely detected, and yielding the precision rate of 91.3(. However, the overall precision obtained for hard cuts is 91.75(. The false detection of gradual transitions is mainly due to the undesirable shaking of the camera, which pulls down the camera motion characterization parameters to near zero. This is evident from the relatively low precision obtained for 'Soccer' and Movie sequence where such jerky movements of the camera are present.

5.2. Adaptive scene tree construction

To evaluate the scene tree building algorithm, we run the algorithm described in Section 3 for various videos. Since we cannot quantify the effectiveness of the algorithm to produce the scene trees, we show two examples of scene trees in Fig. 4 and describe the main events occurring in the video sequences from which the tree is constructed. The scene tree in Fig. 4(a) is built from a 3 min video clip available from the MPEG-7 test video set (CD 21 Misc2.mpg). In this sequence, two women are watching the TV, and talking to each other (the first 8 leaf nodes). Then one of the women leaves while the other continues watching the TV (next 4 leaf nodes). After a while, this woman also leaves. Then both the women appear together. One of them puts on a coat and leaves while the other starts cleaning the table (the last 4 leaf nodes). The scene tree in Fig. 4(b) is built from a 2-min video clip from the TV program 'Opening The New Era'. This video sequence consists of three scenes. A man and a woman are taking a picture together. Then the man drives the woman to her home. The woman goes inside her home and reads the newspaper. If we traverse the scene trees from the top to bottom, which in essence is a nonlinear browsing of the video, we get the above stories. The scene trees give a hierarchical view of the scenes as well as provide a summary of the video sequence.

Fig. 5 shows the pairwise shot similarity matrices for the two video clips in Fig. 4. The brighter the pixel is, the higher the similarity between the two shots is. The similarity between any two shots is computed using Eq. (6). The similarity matrix in Fig. 5(a) represents the MPEG-7 test video shown in Fig. 4(a). The blue lines create four clusters along the diagonal of the similarity matrix, which correspond to the four scene nodes (*SN* (1, 1), *SN* (1, 2), *SN* (1, 3), *SN* (1, 4)) at the first level in the scene tree. The clusters formed by these scene nodes have very high intra cluster similarity and relatively low inter cluster similarity except for cluster *SN* (1, 2). The cluster

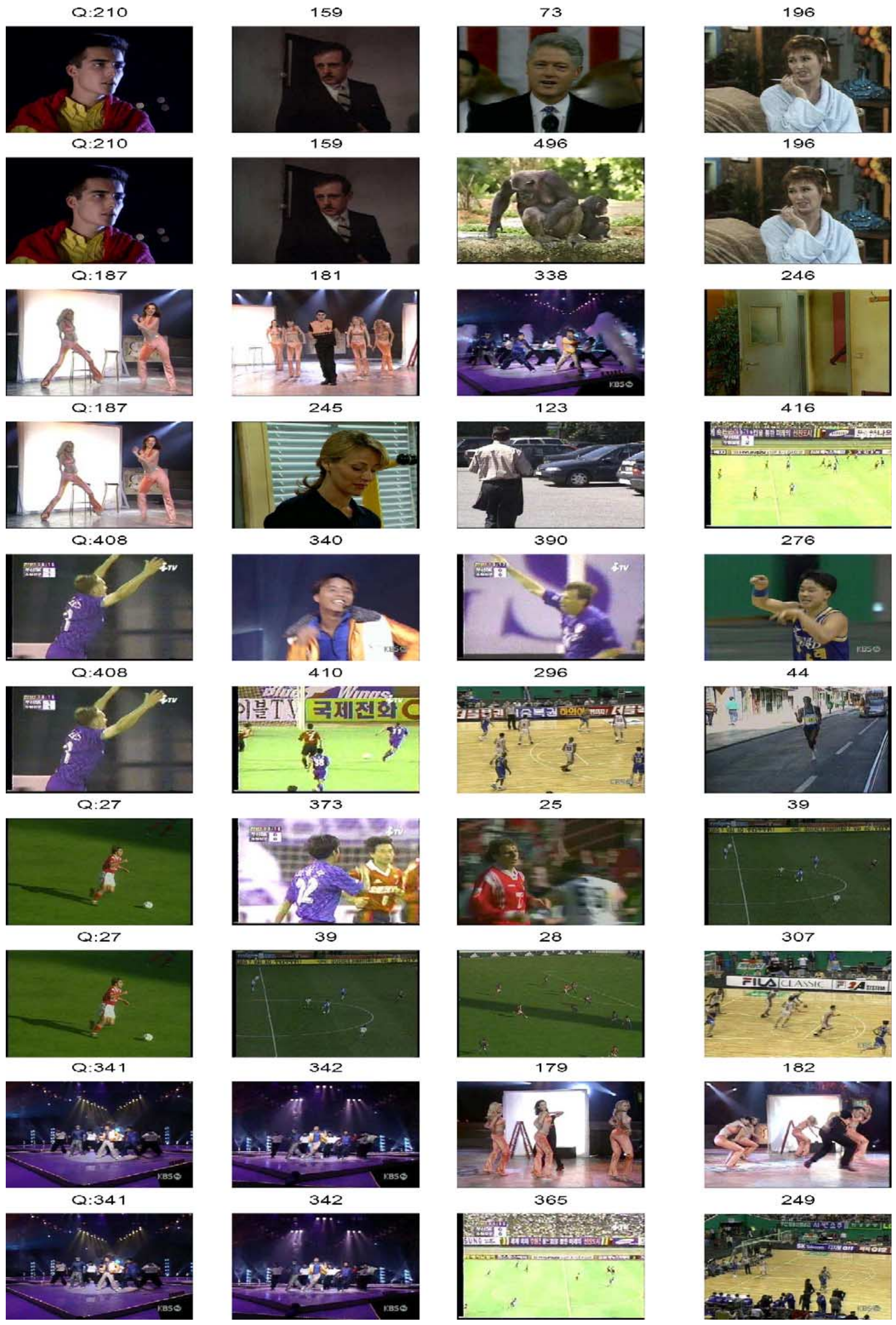

Fig. 8. Motion-based video retrieval. The odd rows use CM, OM and TM matrices for retrieval; the even rows use only the TM matrix for retrieval.

formed by *SN* (1, 2) displays a check-board pattern in the intra-cluster similarity matrix due the alternative appearance of the two women in the talking scene. The largest partition created by the red dotted lines encloses the clusters representing, *SN* (1, 1), *SN* (1, 2), *SN* (1, 3), so that it corresponds to scene node *SN* (2, 1) at the second level. Fig. 5(b) shows the shot similarity matrix computed from the 'Opening The New Era' video in Fig. 4(b). The three blue lines create clusters corresponding to the scene nodes (*SN* (1, 1), *SN* (1, 2), *SN* (1, 3)) at the first level in the scene tree in Fig. 4(b).

Fig. 6 shows the video decomposition scheme proposed in Ref. [7], in which shot classes are found by clustering similar shots using GA algorithm. In our implementation, we use K-means clustering for finding the shot clusters with the

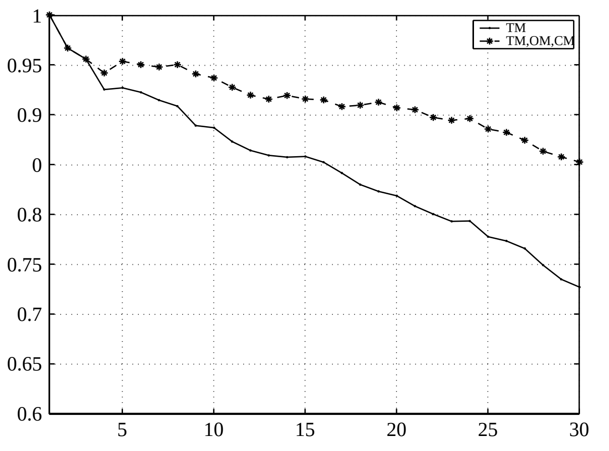

Fig. 9. Precision as a function of the number of video clips retrieved. Solid line correspond to retrievals using the TM, OM and CM and dashed line correspond to retrievals using TM only.

pairwise similarity matrix given by Eq. (6). In Fig. 6(a), the shots are decomposed into four clusters that contain shots relating to (1) watching TV, (2) a woman talking, (3) the other woman talking and (4) a women leaving. Although these four clusters of shots characterize some of the major events in the video, it does not preserve the temporal order of the events as does the scene tree in Fig. 4(a). Fig. 6(b) shows three representative shot classes obtained from the clustering procedure of Ref. [7]. The clusters are: (1) photo taking machine, (2) indoor shots and (3) outdoor shots. But this decomposition scheme does not describe the story of the video properly. For example, *SN* (0, 1), the photo taking shot is grouped as an indoor shot together with the shots, in which the woman is reading a news paper in her room. This decomposition lacks temporal organization of events as well. However, the scene tree decompositions of the video content in Fig. 4(b) do not have these limitations.

5.3. Motion estimation

In this subsection, we evaluate the proposed algorithm for motion estimation. We consider video sequences that are compressed using the MPEG-2 standard. Fig. 7 shows four examples of such videos. Each row consists of the key frame in a particular shot, followed by camera motion, object motion and total motion. The motion information contained in the CM, OM and TM matrices are represented as monochrome images for visualization purposes; the brighter pixels indicate higher motion activity. The first row in Fig. 7 shows the shot of a news anchor person in which the camera is stationary. This is reflected in the black CM image while the motion of the object (face of the person) is clear in the OM image. Thus the total motion consists essentially of the object motion only. Similarly in the second row, which shows a scene with a moving camera while the objects in the scene are stationary, the OM image is black (indicating no motion) while the CM image is uniformly bright. The third row illustrates the case of both object motion and camera motion contributing to the total motion in the shot from a soccer game. We can clearly see the zooming in of the camera in the CM image while object motion is indicated by some bright pixels in the OM image. Finally, we show a shot of the camera tracking a person and its corresponding motion images, which are all mostly bright implying that the motion contains both camera as well as object motion. We see that the proposed motion estimation algorithm has been able to extract the motion information quite well.

5.4. Motion-based retrieval

We noted earlier that the video retrieval process is implicitly contained within the process of browsing the adaptive scene tree. However, the retrieval method proposed in this paper can also be considered as a ‘standalone’ process. To evaluate the performance of our proposed method for motion indexing and retrieval, we build a database consisting of video shots using the SBD algorithm on the 643 video sequences of the MPEG-7 test set. The lengths of the video clips range from 5 to 30 s.

Using the CM, OM and TM matrices as indices and the video similarity measure developed in Section 4, we retrieve the top *N* video shots from the database. In Fig. 8, we show the top three results of the query shown in the first column. We compare the retrieval results when only TM is used with the case when all the three motion matrices, viz., CM, OM and TM, are used to retrieve. The odd rows in Fig. 8 show the retrieval result with the motion feature extracted from CM, OM and TM matrices while the even rows are the retrieval results when only TM is considered. The first column is the key frame from the query video shot. The second, third and fourth columns are the first, second and third retrieval results. From these examples, we observe that both models retrieve video clips, which have similar motion content. Furthermore, we can see that video similarity model with CM, OM and TM gives better retrieval result in both the motion and semantic sense than the model that uses only TM.

We randomly choose 50 video shots from the video database as queries and retrieve the top 30 videos. Using *precision* as the measure for performance, where *precision* is defined as the ratio of the number of relevant shots retrieved to the total number of shots retrieved, we plot *precision* as a function of the number of retrieved video clips as shown in Fig. 9. The average precision (computed as average of the precision with the number of top return video clips varying from 1 to 30) when CM, OM and TM are considered is 91(compared to 85(when only TM is used as an index.

6. Conclusion

In this paper, we have presented a fully automatic content-based approach to organizing and indexing *compressed* video data. The three main steps in our approach are: (i) shot boundary detection using motion prediction information of MPEG-2, (ii) development of a browsing hierarchy called adaptive scene tree, and (iii) video indexing based on camera motion, object motion and total motion. The SBD algorithm can detect both abrupt cuts and gradual transitions. Unlike existing schemes for building browsing hierarchies, our technique builds a scene tree automatically from the visual content of the video. The size and shape of the tree reflect the semantic complexity of the video sequence. A video similarity measure based on the CM, OM and TM matrices is shown to give good retrieval results. The estimation of these matrices using the affine parameter model is also shown to perform well.

The hunt for highly discriminating features is a continuous one. From our study, we find that the CM, OM and TM matrices have excellent discriminatory characteristics. However, during computation of video similarity, the conversion of the matrices to a one-dimensional vector might result in loss of valuable information. A more elegant and robust use of these matrices is one of the future directions for this research. To check the effectiveness of the scene tree building algorithm, it will be useful to conduct user studies in the form of presenting the summarized video to several users and asking them to narrate the story as they understand it. However, it remains to be seen if a meaningful scene tree can be built for sports videos.

References

[1] International Organization for Standardization, Overview of the MPEG-7 Standard, ISO/IEC/JTC1/SC29/WG11 N4031 Edition, March 2001.

[2] International Organization for Standardization, MPEG21 Overview (CODING OF MOVING PICTURES AND AUDIO), ISO/IEC JTC1/SC29/WG11/N4318 Edition, July 2001.

[3] ITU-T, Video Coding for Low Bitrate Communication ITU-T Recommendation, H.263 Edition, February 1998.

[4] ITU-T, Joint Final Committee Draft (JFCD) of Joint Video Specification (ITUT Rec. H.264 - ISO/IEC 14496-10 AVC), H.264 Edition, July 2002.

[5] A. Divakaran, R. Regunathan, K.A. Peker, Video summarization using descriptors of motion activity: A motion activity based approach to key-frame extraction from video shots, Journal of Electronic Imaging 10 (2001) 909–916.

[6] B. Yu, W.-Y. Ma, K. Nahrstedt, H. Zhang, Video summarization based on user log enhanced link analysis, in: ACM Multimedia 2003, Berkeley, CA, USA, 2003, pp. 382–391.

[7] N.D. Doulamis, Optimal content-based video decomposition for interactive video navigation, IEEE Transaction on Circuits and Systems for Video Technology 14 (6) (2004) 757–775.

[8] E. Katz, The Film Encyclopedia, Harper Collins, New York, 1994.

[9] J.R. Kender, B.-L. Yeo, Video scene segmentation via continuous video coherence, in: Proceedings of IEEE Internation Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, 1998, pp. 367–373.

[10] B.T. Truong, S. Vekatesh, C. Dorai, Scene extraction in motion picture, IEEE Transactions on Circuits and Systems for Video Technology 13 (1) (2003) 5–15.

[11] M.A. Robertson, R.L. Stevenson, Restoration of Compressed Video Using Temporal Information, in: Proceedings of SPIE conference on Visual Communications and Image Processing, San Jose, CA, vol. 4310, 2001, pp. 21–29.

[12] B. Gunturk, Y. Altunbasak, R. Mersereau, Super-resolution reconstruction of compressed video using transform-domain statistics, IEEE Transactions on Image Processing 13 (1) (2004) 33–43.

[13] D. Schonfeld, D. Lelescu, VORTEX: Video retrieval and tracking from compressed multimedia databases—multiple object tracking from MPEG- 2 bitstream, Journal of Visual Communications and Image Representation, Special Issue on Multimedia Database Management 11 (2000) 154–182.

[14] H. Wang, A. Divakaran, A. Vetro, S.-F. Chang, H. Sun, Survey of compressed-domain features used in audio-visual indexing and analysis, Journal of Visual Communication and Image Representation 14 (2) (2003) 50–183.

[15] R. Lienhart, Comparison of automatic shot boundary dtection algortihms, in: Proceedings of SPIE conference on Storage and Retrieval for Image and Video Databases VII, vol. 3656, 1999, pp. 290–301.

[16] H. Zhang, A. Kankanhalli, S. Smoliar, Automatic partitioning of full-motion video, Multimedia Systems 1 (1993) 10–28.

[17] R. Zabih, J. Miller, K. Mai, A feature-based algorithm for detecting and classifying scene breaks, in: Proceedings of ACM Multimedia 95, San Francisco, CA, vol. 1, 1995, pp. 10–28.

[18] I.K. Sethi, N.V. Patel, Statistical approach to scene change detection, in: Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases, San Jose, CA, vol. 2420, 1995, pp. 329–338.

[19] J. Feng, K.-T. Lo,. M.H, Scene change detection algorithm for image video sequence, in: Proceedings of IEEE International Conference on Image Processing, vol. 1, 1996, pp. 821–824.

[20] B.L. Yeo, B. Liu, Rapid scene analysis on compressed video, IEEE Transactions On Circuits and Systems for Video Technology 5 (1995) 553-544.

[21] B.L. Yeo, B. Liu, On the extraction of DC sequence from MPEG compressed video, in: Proceedings of IEEE International Conference on Image Processing, Washington, DC, 1995, pp. 2260–2263.

[22] J. Meng, Y. Juan, S.-F. Chang, Scene change detection in a MPEG video sequence, in: Proceedings of SPIE Conference on Multimedia Computing and Networking, San Jose, CA, vol. 2417, 1995, pp. 180–191.

[23] H. Yi, D. Rajan, L.-T. Chia, A unified approach to detection of shot boundaries and subshots in compressed video, in: Proceedings of IEEE International Conference on Image Processing, Barcelona, Spain, 2003.

[24] B.G. Haskell, A. Puri, A.N. Netravali, Digital video: An Introduction to MPEG-2, Chapman I & Hall, New York, 1997.

[25] B.L. Yeo, B. Liu, Rapid scene analysis on compressed video, IEEE Transactions On Circuits and Systems for Video Technology 5 (1995) 553-544.

[26] A. Hanjalic, R.L. Lagendijk, J. Biemond, Automated segmentation of movies into logical story units, IEEE Transactions on Circuits and Systems for Video Technology 9 (4) (1999) 580–588.

[27] A. Daniel, Grammar of the Film Language, Focal Press, London, 1976.

[28] C. Saraceno, R. Leonardi, Identification of story units in audio-visual sequences by joint audio and video processing, in: Proceeding of International Conference on Image Processing, Chicago, IL, USA, 1998, pp. 358–362.

[29] Z. Rasheed, M. Shah, Scene boundary detection in hollywood movies and TV show, in: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, Madison, WI, 2003.

[30] J. Nam, A.H. Tewfik, Combined audio and visual streams analysis for video sequence segmentation, in: Proceedings of ICASSP-97, Munich, Germany, vol. 4, 1997, pp. 2665–2668.

[31] J. Oh, K.A. Hua, An efficient and cost-effective technique for browsing and indexing large video databases, in: Proceedings of 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, 2000, pp. 415–426.