Automatic Shot Change Detection Algorithm Using Multi-stage Clustering for MPEG-Compressed Videos

Byung Cheol Song and Jong Beom Ra

Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, Republic of Korea

Received January 5, 2000; accepted December 4, 2000

Automatic shot change detection has been recognized as an important research issue for video classification. This paper proposes an automatic clustering-based algorithm for shot change detection in MPEG-compressed videos with a small number of user-defined parameters. For accurate detection of abrupt and gradual shot changes, the proper selection and extraction of features are important. We first propose a fast edge image extraction scheme in the DCT domain on the basis of AC prediction. Then, by using the features extracted from the edge images and DC images, a two-stage clustering-based algorithm is proposed for shot change detection. In the first stage, the algorithm detects abrupt shot changes by employing two-means clustering on the 2-D feature space of histogram and pixel differences between two neighboring DC frames. In the next stage, it subsequently explores gradual shot changes between two adjacent abrupt shot changes by performing a two-step clustering scheme, which uses multiple features such as an edge energy diagram and several frame difference measures. Simulation results show that the proposed algorithm is fast and accurate.

Key Words: video classification; shot change detection; cut, fade, dissolve; feature extraction; MPEG-compressed video; k-means clustering.

I. INTRODUCTION

In many interactive multimedia systems, efficient retrieval of multimedia information in a digital form is becoming an important issue [1]. Among all multimedia types (text, image, graphic, audio, and video), video is the most challenging since all other media information can be combined into a single data stream of video. Thanks to the lowered cost of digital storage media, increased transmission rates, and improved compression techniques, the use of digital videos has increased tremendously in the past few years. However, tools available for video retrieval still remain primitive. To solve this problem, efficient video analysis and summarization techniques are required.
A video sequence is usually composed of many meaningful scenes. Each scene is decomposed again into several shots. A shot is a single sequence of a movie or a TV program captured by one camera without interruption; it consists of consecutive frames, which represent a continuous action in time and space. In general, frame-based video representation is inefficient due to its huge data overhead. So, a shot is considered as the basic unit for video understanding. Shot-based video representation can offer a generic reference for users, like a time-code. Therefore, accurate shot change detection is required for effective video representation.

Shot changes can be divided into two types based on the video production procedure as well as the changing duration: abrupt changes (also referred to as cuts), and gradual changes (such as fades, dissolves, and wipes). Shot change detection can be described as a process for detecting the temporal shot transition in a video sequence based on certain criteria [2–14]. With the advent of automatic video information management systems, many researchers have actively studied the cut detection problem and have developed many successful algorithms, e.g., the algorithms based on frame difference features, motion vector characteristics, and bit-rate comparisons [3–10]. However, successful detection of gradual shot changes such as fades and dissolves is still a hot issue. Zabih et al. proposed an interesting approach that uses edge tracking to detect various shot changes [11]. Observing the ratios of appearing and disappearing edge percentages identifies gradual shot changes. However, their method requires a significant computational cost due to spatial domain processing. To reduce the computational burden, Yeo and Liu extracted DC images directly from MPEG-compressed videos, and detected shot changes by using pixel differences and histogram differences obtained from DC images [12]. However, their scheme does not provide reliable performance in the detection of gradual shot changes in real-world video clips, even though it successfully detects cuts with very high processing speed. Yu and Wolf detected fades/dissolves by using edge spectrum characteristics extracted from wavelet-transformed frames [13]. Recently, Lienhart proved that edge contrast information is more useful for the detection of fades/dissolves than any other well-known features such as edge change ratio, color histogram difference, and variance of pixel intensities [14]. Nevertheless, it is difficult to achieve an acceptable trade-off between recall and precision rates by using a single specific feature such as edge contrast information.

In order to overcome the problems mentioned above, we may need to employ various features including edge information and color histogram, which can be directly extracted from MPEG-compressed videos with high processing speed. Also, since it is difficult to accurately detect fades/dissolves in a single step, locating fades/dissolves via several steps by using various visual features may be desirable for accomplishing reliable search accuracy. In addition, a clustering-based framework as in [10] is very attractive in that a few features can be treated simultaneously in a single step in the framework, which improves detection accuracy.

This paper presents a multi-stage clustering-based algorithm for unsupervised shot change detection in MPEG-compressed videos. The algorithm takes into account all the points mentioned above. First, we propose a fast method to obtain edge images in the DCT domain. Once DC images are obtained from a well-known scheme [15], edge images are extracted from the DC images on the basis of AC prediction. This method drastically reduces computational and memory costs and provides edge images of acceptable quality in comparison to edge images extracted in the spatial domain. Second, we propose a two-stage clustering-based algorithm for shot change detection, which uses several visual features obtained from
DC and edge images. In the first stage, the algorithm detects cuts by performing unsupervised $k$-means clustering ($k = 2$) [16] on the 2-D feature space of the histogram difference and pixel difference features. In the second stage, it subsequently explores fades/dissolves between neighboring cuts by adopting a two-step clustering scheme based on an edge energy diagram and a few frame difference features. The proposed scheme provides reliable performance and reduces the overall computational cost by adopting a kind of hierarchical search structure that makes use of several features.

The rest of the paper is organized as follows. In Section II, a method to extract edge images from an MPEG-compressed video is presented. Section III deals with the proposed shot change detection scheme based on DC images and edge images. Intensive simulation results for the proposed scheme are given in Section IV. Finally, concluding remarks are provided in Section V.

II. FAST FEATURE IMAGE EXTRACTION IN THE DCT DOMAIN

The proposed shot change detection algorithm requires two feature images, DC images and edge images, extracted from an MPEG compressed video. We can directly extract DC images in the DCT domain by using the scheme suggested by Yeo and Liu [15]. Edge images could be also extracted by using AC coefficients [17]. In order to minimize decoding and memory burden, however, this paper presents a new simple algorithm for extracting edge images directly from DC images on the basis of AC prediction. Since the processing as well as decoding of AC coefficients is unnecessary, this algorithm requires a smaller amount of computation and memory than our previous method [17]. In the following subsections, we first review the DC image extraction scheme briefly and then present our scheme.

A. DC Image Extraction Using the First-Order Approximation [15]

Extraction of a DC image from the I-frame is trivial. Its DC term $F(0, 0)$ is related to pixel values $f(i, j)$ by

$$F(0, 0) = \frac{1}{8} \sum_{i=0}^{7} \sum_{j=0}^{7} f(i, j) \tag{1}$$

and is 8 times the average intensity of the block.

For P-frames and B-frames, however, motion vector information must be employed to derive DC images. Suppose $P_{\text{cur}}$ is a current block of interest, and $P_1, \ldots, P_4$ are its four adjacent blocks from which $P_{\text{cur}}$ is derived in the previous frame (see Fig. 1). Let $h_i$ and $w_i$ be the height and width of $P_{\text{cur}} \bigcap P_i$. Then, if the shaded regions in $P_1, \ldots, P_4$ are moved by $(\Delta x, \Delta y), (h_1, w_1) = (\Delta x, \Delta y)$. Due to the linearity of DCT, the DC coefficient of $P_{\text{cur}}$ is given as

$$DC_{P_{\text{cur}}} = \sum_{i=1}^{4} \left( \sum_{m=0}^{7} \sum_{l=0}^{7} w_{ml}(DCT(P_i))_{ml} \right), \tag{2}$$

where $(DCT(P))_{ml}$ denotes the $(m, l)$th component of the DCT of $P$. The weighting factor
$w_{ml}^i$ denotes the contribution of $(DCT(P_i))_{ml}$ to $DC_{P_{cur}}$. Following the approach in [18], $w_{ml}^i$ can be represented as

$$w_{ml}^i = (DCT(S_{i1}))_{hl} \times (DCT(S_{i2}))_{l0},$$  \hspace{1cm} (3)$$

where $S_{i1}$ and $S_{i2}$ are matrices of the form

$$\left( \begin{array}{cc}
0 & 0 \\
I_h & 0
\end{array} \right) \text{ or } \left( \begin{array}{cc}
0 & I_h \\
0 & 0
\end{array} \right).$$

Here $I_n$ is an identity matrix of size $n$. Matrix $S_{ij}$ serves to move the subblock of interest in $P_i$ to the position of $P_{cur}$. An example of the role of $S_{ij}$ on $P_{cur}$ is demonstrated in Fig. 2.

Computation of Eq. (2) may require multiplications up to 256 times per block. To decrease the computational burden, the first-order approximation is newly adopted. Then, the contribution of the four neighboring DC values to the current DC value are obtained from the overlapping ratio of block $P_{cur}$ with each of the blocks $P_1, \ldots, P_4$; i.e.,

$$DC_{P_{cur}}^a = \sum_{i=1}^{4} \frac{h_i w_{i4}}{64} DC_{P_i}.$$

$$\left( \begin{array}{cc}
0 & 0 \\
I_{h_4} & 0
\end{array} \right) \left( \begin{array}{cc}
0 & I_{w_4} \\
0 & 0
\end{array} \right)$$

FIG. 1. Current block $P_{cur}$ and its motion vector.

FIG. 2. Pre- and post-matrix multiplications to move a subblock.
FIG. 3. Computation of the edge block $G^*_5$ of a current block $P_5$. $G^*_5$ is obtained by the addition and subtraction of six shaded blocks.

Note in Eq. (4) that at most four multiplications are needed for obtaining a DC value of each block. Also, for this DC value calculation, only the DC coefficients from the four blocks $P_1, \ldots, P_4$ in the previous frame and motion vector information are needed. This motion vector information can be obtained via minimal decoding of the MPEG-compressed bitstream. Therefore, an approximated DC image can easily be obtained by using Eq. (4).

In this paper, we use these DC images due to their small computational cost. Hereafter, $DC_{P_k}$ means the first-order approximated DC value of block $P_k$, i.e., $DC_{P_k}^{a}$. 

**B. Edge Image Extraction Based on AC Prediction**

Let us consider a scheme for extracting an edge image from a DC image. The edge block $G_i$ of the $i$th block $P_i$ is generated by the following procedure. As shown in Fig. 3, the $x$ component of $G_5$, $G^*_5$, is computed by using only addition and subtraction of six shaded blocks if the Sobel edge operator is employed [19]. Consequently, $G^*_5$ is affected by all nine blocks, from $P_1$ to $P_9$. However, eight blocks, excluding $P_5$, give minor effects on $G^*_5$ since they are used only for the calculation of boundary elements of $G^*_5$. In other words, we can calculate the exact values, up to $6 \times 6$ elements inside $G^*_5$, by using $P_5$ only. Let $\tilde{G}^*_5$ be a modified $G^*_5$ whose boundary values are deliberately filled with zeros. After a few steps of arithmetic operation, $\tilde{G}^*_5$ can be derived as

$$\tilde{G}^*_5 = L_x P_5 R_x.$$  \hspace{1cm} (5)

In Eq. (5), $L_x$ and $R_x$ are sparse $8 \times 8$ matrices and are easily derived from the definition of the Sobel operator. Note that they are independent of $P_5$. From Eq. (5), the DCT form of $\tilde{G}^*_5$ can be described as

$$DCT(\tilde{G}^*_5) = DCT(L_x)DCT(P_5)DCT(R_x).$$  \hspace{1cm} (6)
where

\[
DCT(L_x) = \begin{bmatrix}
3 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\
0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\
0 & -2 & 0 & 2 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\
-1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
DCT(R_x) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}
\]

Similarly, \(DCT(\tilde{G}_y^2)\) is of the form

\[
DCT(\tilde{G}_y^2) = DCT(L_y)DCT(P_5)DCT(R_y).
\]

where

\[
DCT(L_y) = \begin{bmatrix}
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
DCT(R_y) = \begin{bmatrix}
3 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\
0 & 2 & 0 & -2 & 0 & -1 & 0 & 0 \\
-1 & 0 & 2 & 0 & -1 & 0 & -1 & 0 \\
0 & -1 & 0 & 2 & 0 & -1 & 0 & 0 \\
-1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

From Eqs. (6) and (7), the edge energy of \(P_5\) except its boundary is approximated according to Parseval’s theorem [19]. For instance, by using the first 5 AC coefficients of \(DCT(P_5)\),
the block edge energy of $P_5$, $\varepsilon_{P_5}$, is approximated as

$$\varepsilon_{P_5} \approx 2(P_{01}^2 + P_{10}^2) + 4(P_{02}^2 + P_{20}^2) + 3P_{11}^2,$$

where $P_{ij}$ denotes the $(i, j)$th coefficient of $DCT(P_5)$.

Here we employ an AC prediction method, which has already been adopted in JPEG [20], in order to predict the five lower AC components. The basic assumption is that an image can be modeled as a quadratic surface. A quadratic surface,

$$S(x, y) = a_{2,2} x^2 y^2 + a_{2,1} x^2 y + a_{1,2} x y^2 + a_{2,0} x^2 + a_{1,1} x y + a_{0,2} y^2 + a_{1,0} x + a_{0,1} y + a_{0,0},$$

is to be fitted into a $3 \times 3$ array of DC values as depicted in Fig. 4. The coefficients $a_{2,2}, \ldots, a_{0,0}$ are then determined so that block mean values computed from the quadratic surface can match the DC values, $DC_1, \ldots, DC_9$. If DCT is performed on the center block predicted by the quadratic surface model, the five AC coefficients of $DCT(P_5)$ will be derived only from $DC_1, \ldots, DC_9$.

$$
\begin{align*}
P_{01} &= a(DC_4 - DC_5), \\
P_{10} &= a(DC_2 - DC_3), \\
P_{20} &= b(DC_2 + DC_8 - 2DC_3), \\
P_{02} &= b(DC_4 + DC_6 - 2DC_3), \\
P_{11} &= c((DC_1 - DC_3) - (DC_7 - DC_9)),
\end{align*}
$$

where $a = 1.13885/8$, $b = 0.27881/8$, and $c = 0.16213/8$. Equation (10) indicates that AC coefficients can be obtained from the linearly weighted sum of adjacent DC values, and no additional memory is required for AC prediction. By using Eq. (10), we can rewrite Eq. (8) only with DC coefficients. Therefore, we can extract an edge image, or block edge energy distribution, from a DC image. Figure 5 is the block diagram of a modified MPEG decoder for extracting edge images from DC images. Note that this edge image extraction...
can reduce the decoding burden as well as memory size, since it does not need the actual AC coefficients of each block.

Figure 6 shows edge images of reduced size, which are obtained by various methods including the proposed method. The extracted frame is the 250th frame of News2 clip, an MPEG-7 test video clip to be used in Section IV. We can observe that the edge image by the proposed scheme approximates the edge image obtained from actual AC coefficients well, and it retains global edge features given by the edge image obtained in the spatial domain after full decoding. Furthermore, Table 1 demonstrates that the proposed scheme is very fast. Each computation time in Table 1 represents the value averaged for five test clips (used in Section IV) normalized with the computation time for DC extraction only. We can notice that the computational cost for edge image extraction is almost equivalent to that for DC image extraction.

**FIG. 5.** A block diagram of a modified MPEG decoder for extracting edge images as well as DC images.

**FIG. 6.** Original image and its reduced edge images: (a) original image, (b) edge image using a spatial domain operator, (c) edge image using actual AC coefficients, and (d) edge image using predicted AC coefficients. Note that the size of edge images is 1/64 times that of the original image.
### III. SHOT CHANGE DETECTION

The shot change detection algorithm is performed on the basis of two feature images, DC images and edge images, which are extracted from MPEG-compressed data as described in Section II. It should be noticed that the size of feature images is 1/64 times that of the original image and this drastically reduces overall processing time as well as various feature extraction time for shot change detection.

#### A. Cut Detection

Usually, the performance of cut detection relies highly on frame difference features selected to identify shot changes. Frame differences can be defined in terms of pixel values, histograms, motion vectors, pixel statistics, etc. Among them, histogram difference and pixel difference between two adjacent DC frames are the most popular features for hard cut detection [2]. Here, the former is defined as

\[ d^h(n, n-1) = \sum_i |h_n(i) - h_{n-1}(i)|, \]

where \( h_n(i) \) denotes the \( i \)th bin value of a normalized histogram of the \( n \)th DC frame, and the latter is defined as

\[ d^p(n, n-1) = \sum_i \sum_j |p_n(i, j) - p_{n-1}(i, j)|, \]

where \( p_n(i, j) \) denotes an intensity value of \( (i, j) \)th pixel in the \( n \)th DC frame. Histogram-based comparison methods are highly preferred because they are robust to detrimental effects such as camera and object motion and changes in scale and rotation. However, such methods sometimes fail to identify changes between shots having similar color content or intensity distribution. On the other hand, pixel-wise comparison methods can well identify changes between shots having a similar color content or intensity distribution, but they are very sensitive to movements of cameras or objects. Since the adopted pixel difference feature is extracted from DC images, it becomes less sensitive to small object and camera motions. However, it still is not enough for reliable shot change detection.

To overcome the afore-mentioned drawbacks of histogram difference and pixel difference features, we introduce a clustering-based cut detection scheme by jointly using the two features. For their joint usage, each feature is normalized to the values between 0 and 1 and is filtered to remove undesirable noise.

The main assumption for cut detection is as follows:

- Within a single shot, interframe variations are small, which results in a slowly varying feature signal.
- However, an abrupt change across a shot boundary causes a sharp peak in a feature signal.

<table>
<thead>
<tr>
<th>Extraction using predicted AC coefficients</th>
<th>Extraction from actual AC coefficients</th>
<th>Extraction in the spatial domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.03</td>
<td>2.42</td>
<td>7.98</td>
</tr>
</tbody>
</table>
So we can detect cuts by recognizing these peaks. However, the sensitivity of these features to camera motion, object motion, and other noises strongly influences detection performance. In order to remove this phenomenon, a filtering scheme to reduce feature signal values at high activity regions while minimizing effects on those at actual shot changes, is needed [5, 13]. In this paper, we choose an unsharp masking technique, i.e.,

\[ d_f(n, n-1) = \begin{cases} 
  d(n, n-1) - \tilde{d}(n, n-1), & \text{if } d(n, n-1) > \tilde{d}(n, n-1), \\
  0, & \text{otherwise.}
\end{cases} \] (13)

Here, the 1-D frame difference signal \( d(n, n-1) \) can either be \( d^h(n, n-1) \) or \( d^p(n, n-1) \). \( \tilde{d}(n, n-1) \) denotes the low-pass filtering and/or median filtering result of \( d(n, n-1) \), and \( d_f(n, n-1) \) denotes the unsharp masking output, respectively. It is found that the sequential use of low-pass filtering and median filtering for \( \tilde{d}(n, n-1) \) further reduces false alarms than each filtering alone. So, we jointly use low-pass filtering and median filtering. In this paper, we use a simple averaging filter for low-pass filtering and fix the tap sizes of both filters to 5 for simplicity. Figure 7 demonstrates that the adopted filtering suppresses intrashot activity well for a 1-D histogram difference signal of an MPEG-7 test video clip, News1.

After sequentially applying unsharp masking to both histogram difference and pixel difference features, we obtain a scatter plot in the 2-D feature space as shown in Fig. 8 (the data given in Fig. 7 is used). In this figure, it is observed that filtering causes the cluster center of intrashots to move toward the origin, thereby separating the two clusters more clearly. This indicates that more accurate detection of shot changes can be achieved by filtering. Cuts are detected by performing conventional \( k \)-means clustering \((k = 2)\) without any threshold, because cut points are generally located far from \((0, 0)\) on the 2-D feature domain.

The set of detected cuts, however, includes another type of false positives, i.e., flashlights. Usually, in a difference feature signal, the necessary conditions for the presence of flashlight are [12]

1. the maximum peak is close to the second largest peak, and
2. these two peaks are much larger than the average of the remaining signal values.
Based on this condition, we apply the following criterion to all the numbers in the cut cluster:

\[
\begin{align*}
&\text{if } (d_f^{n_{\text{max}1}}(n_{\text{max}1} - 1) > w \cdot c_p^c \& \& d_f^{n_{\text{max}2}}(n_{\text{max}2} - 1) > w \cdot c_p^c) \\
&\Rightarrow \text{flashlight,} \\
&\text{else} \\
&\Rightarrow \text{cut.}
\end{align*}
\]

Here, \(c_p^c\) and \(w\) denote a pixel difference component of the cut cluster center and a constant weight, respectively; and the \(n_{\text{max}1}\)th and \(n_{\text{max}2}\)th frames are the top two frames having maximum differences. In this paper, only the pixel difference feature is used for removing flashlight effects, and \(w\) is heuristically set to 0.4. Note that a threshold, \(w \cdot c_p^c\) is adaptively defined according to input video sequences. This flashlight removal scheme eliminates most of the flashlight effects effectively. A complete block diagram of the proposed cut detection scheme is shown in Fig. 9.

---

**FIG. 8.** Scatter plots of frame comparison measures in the 2-D feature space (a) without filtering and (b) with filtering. Here diamonds denote cluster centers.

**FIG. 9.** Block diagram of the clustering-based cut detection scheme.
B. Gradual Shot Change Detection

Now let us consider a detection scheme of gradual shot changes existing between adjacent cuts. A fade is a shot change where the first shot gradually disappears (fade out) before the second shot gradually appears (fade in); and a dissolve is a shot change where the first shot gradually disappears while the second shot appears. In other words, a fade or a dissolve is a gradual change of the visibility of the details in a shot that causes the overall structure of the shot to change from one to another. The difference between these two kinds of shot changes lies in when the second shot begins to appear. We can easily discriminate dissolves from fades, because the latter tend to have a single color as well as a much lower edge energy in the middle of transition. We can also consider a dissolve as a general form of fade in that the dissolve is a combination of fade-in and fade-out. Thus, this paper focuses on dissolve detection. The proposed scheme for dissolve detection consists of four stages: extraction and filtering of an edge energy diagram, selection of dissolve candidates, two-step clustering, and elimination of false positives (see Fig. 10). The proposed algorithm is a sort of hierarchical search algorithm, which eliminates unreliable dissolve candidates via several steps.

B.1. Extraction and filtering of an edge energy diagram. As mentioned in Section I, edge information is useful for dissolve detection [14, 17]. One of the important properties of dissolves is that as the transition starts, the image becomes cloudy and outlines become vague (see Fig. 11). This implies that the edge energy decreases. As the transition ends, the reverse phenomenon occurs. So U-shaped patterns of edge energy normally occur during dissolves (see Fig. 11c). We will use this phenomenon for dissolve detection.

![Block diagram of the dissolve detection scheme.](image)
First, we obtain an edge energy diagram by computing the sum of edge intensities for each frame throughout the video sequence. Edge energy of the $n$th frame is computed as

$$E(n) = \frac{1}{V \times H} \sum_{i=1}^{V} \sum_{j=1}^{H} \varepsilon_{(i,j)},$$

(14)

where $V$ and $H$ denote the numbers of vertical and horizontal blocks in a frame, respectively, and $\varepsilon_{(i,j)}$ stands for an edge energy value of the $(i, j)$th block in the $n$th frame. Prior to the detection procedure, low-pass filtering and median filtering are performed to remove noises on the edge energy diagram.
B.2. Selection of dissolve candidates. We choose all probable dissolve candidates by recognizing U-shaped patterns on the edge energy diagram. The detection procedure is performed for all $n$ as

$$\text{if } ((\tilde{E}(n-2) > \tilde{E}(n-1) > \tilde{E}) \& \& (\tilde{E}(n-2) - \tilde{E}(n) > \delta) \& \& (\tilde{E}(n+1) > \tilde{E}(n)) \& \& (\tilde{E}(n+2) - \tilde{E}(n) > \delta)), \text{ then } I(n) = 1; \text{ else } I(n) = 0.$$ 

$\tilde{E}(n)$ denotes the filtered edge energy of the $n$th frame, $I(n)$ is an index function, and $\delta$ is a proper threshold. If $I(n)$ is equal to 1, we choose the $n$th frame as a possible dissolve center.

B.3. Two-step clustering. Since it is difficult to accurately detect fades/dissolves at once, we additionally eliminate unreliable dissolve candidates by employing two-step clustering using various visual features.

The first clustering is based on the histogram difference feature and the motion-compensated pixel-difference feature. The histogram difference feature is extracted from a frame pair having a certain distance around each dissolve candidate (see Fig. 12). Let us assume that the $n_{d}$th frame is the center of a dissolve candidate. Then we define the duration of the dissolve candidate as $[n_{d} - \tau, n_{d} + \tau]$. Here, $n_{d} - \tau$ and $n_{d} + \tau$ correspond to the starting and ending frames indicated in Fig. 11c. We decide $\tau$ as

$$\text{THR} = \text{Max}(\tilde{E}(n_{d} - 1) - \tilde{E}(n_{d}), \tilde{E}(n_{d} - 2) - \tilde{E}(n_{d} - 1)), \quad n = 2,$$

$$\text{while}(\text{ind}){\text{if}((\tilde{E}(n_{d} - n - 1) - \tilde{E}(n_{d} - n)) < \text{THR})\{ \tau_i = n, \text{ind} = 1 \}}$$

$$\text{else}\{ \text{if}(n < 10) \quad n + +, \text{else}\{ \tau_i = n; \quad \text{ind} = 1 \} \}$$

}
FIG. 12. Frames corresponding to a dissolve candidate, which are selected for feature extraction in the first clustering step. If the $n_d$th frame is the center of the dissolve candidate obtained in Subsection II.B.1, the $(n_d - 4)$th and $(n_d + 5)$th frames denote the starting and ending frames of the dissolve, respectively.

$\tau_r$ is also determined in a similar fashion. Figure 12 illustrates the frames corresponding a dissolve candidate of $\tau_l = 4$ and $\tau_r = 5$. Based on the determined starting and ending frames of the dissolve candidate, the histogram difference and motion compensated pixel difference features are extracted by

$$D^h(n_d) = d_h^i(n_d - \tau_l, n_d + \tau_r),$$

$$D^{mp}_{1}(n_d) = \sum_{k} \min_{(x,y)} \sum_{t=0}^{1} \sum_{j=0}^{1} |p^k_{n_d+t\tau_l}(i, j) - p^k_{n_d-t\tau_r}(i+x, j+y)|,$$

where $p^k_n(i, j)$ denotes a pixel intensity located at $(i, j)$ in the $k$th macroblock of the $n$th frame. The macroblock size in DC images is $2 \times 2$ pixels. In Eq. (16), the search range of $(x, y)$ is to be related with the distance between the two frames, i.e., $(\tau_l + \tau_r)$, and is set to $\pm(\tau_l + \tau_r)$. Since the motion estimation is performed on DC images, it corresponds to the search range of $\pm 8$ for the adjacent frames. If there is a dissolve around the $n_d$th frame, both $D^h(n_d)$ and $D^{mp}_{1}(n_d)$ have large values. Otherwise, they have small values. Therefore, we can select reliable candidates by performing two-means clustering on the 2-D feature space of $D^h(n_d)$ and $D^{mp}_{1}(n_d)$. Before clustering, the above two features are normalized to the values between 0 and 1.

The second clustering is applied for the dissolve candidates obtained from the first clustering. Note that the number of candidates to be dealt with in the second clustering decreases. We first find the refined center of each dissolve candidate. If the center of a dissolve candidate is the $n_d$th frame, the refined center frame $n'_d$ is determined by

$$n'_d = \arg\min_{n_d-\tau_{\text{max}}/2 \leq n \leq n_d+\tau_{\text{max}}/2} D^d(n),$$

where

$$D^d(n) = \sum_{i} \sum_{j} \left| p_d(i, j) - \frac{p_{n-\tau_{\text{max}}/2}(i, j) + p_{n+\tau_{\text{max}}/2}(i, j)}{2} \right|.$$
Here we limit the search range for $n_d$ to a small range of $\pm \tau_{\text{min}}/2$, because $n'_d$ is mostly equivalent or close to $n_d$. Equation (17) represents a major characteristic of dissolve well. For instance, if there are no object/camera motions and $n'_d$ is a real dissolve center, $D^d(n'_d)$ should be zero. We use $D^d(n'_d)$ as the first feature for the second clustering. As another feature for the second clustering, we adopt the pixel-difference for the same frame-pair as $D^d(n'_d)$, which is defined as

$$D^m_p(n'_d) = \sum_k \min_{(i,j)} \sum_{i=0}^{l} \sum_{j=0}^{l} \left| p^k_{n'_d + \tau_{\text{min}}/2}(i, j) - p^k_{n'_d - \tau_{\text{min}}/2}(i + x, j + y) \right|. \quad (20)$$

Here, the search range of $(x, y)$ is determined according to the distance between the two corresponding frames, i.e., $\tau_{\text{min}}$, as in Eq. (16). If there is a dissolve around the $n'_d$th frame, $D^d(n'_d)$ and $1/D^m_p(n'_d)$ will be small. Therefore, we apply the second clustering on the 2-D feature space of $D^d(n'_d)$ and $1/D^m_p(n'_d)$ to determine a dissolve candidate cluster near the origin. Unlike the 2-D feature space of $D^h(n_d)$ and $D^m_p(n_d)$ in the first clustering, the 2-D feature space of $D^d(n'_d)$ and $1/D^m_p(n'_d)$ has an L-shape distribution because the sample points whose $D^d(n'_d)$ and $1/D^m_p(n'_d)$ are concurrently large rarely exist in real video sequences. Hence, in the second clustering, three-means clustering is employed to extract the dissolve candidate cluster near the origin from the sample points with L-shape distribution.

Figures 15a and 15b show the cluster separation results of dissolve candidates in the News1 video clip after the first and second clustering, respectively. As was mentioned above, we use three-means clustering in Fig. 15b, whereas 2-means clustering is performed in Fig. 15a.

**B.4. Reduction of false positives.** Although unreliable dissolve candidates are mostly removed via the procedures described above, there still exist false positives due to camera motions. So we must distinguish dissolves from camera motions such as zooming and panning. Changes due to camera motions tend to induce successive difference values similar to those of gradual shot changes. This problem cannot be resolved by a threshold-based approach. Rather, it is necessary to detect motion patterns induced by camera motions. In this paper, we adopt a simple motion-based scheme, which is a modified version of Zhang’s algorithm [3].

The feature to be used for detecting camera motions is the motion vector field, which is obtained by the motion vector decoding of an MPEG-compressed video. Figure 13 shows motion vector patterns due to (a) camera panning, (b) zooming out, and (c) zooming in.
FIG. 14. Selection of the P-frame nearest to each dissolve candidate (black box) for extraction a motion vector field. Pointed frames are selected P-frames.

illustrates typical motion vector fields resulted from panning and zooming. In general, shot changes such as dissolves and fades will not introduce such motion vector fields. Therefore, if we examine the motion vector pattern, we can distinguish true shot changes from camera motions. In order to utilize the available motion vector information as much as possible, we examine the motion vector field of a P-frame closest to each dissolve candidate (see Fig. 14). This is because the P-frame consists mostly of intermacroblocks and their motion vectors can be obtained via minimal decoding.

As illustrated in Fig. 13a, during panning, most motion vectors can be represented with a single modal value corresponding to camera motion. This leads to

\[
\sum_{i}^{L} \| \theta_i - \theta_m \| \leq \Theta_p, \tag{21}
\]

where \( \theta_i \) and \( \theta_m \) denote the \( i \)th motion vector and the modal vector, and \( L \) is the total number of motion vectors in a frame. For instance, \( \| \theta_i - \theta_m \| \) is zero when the two motion vectors are exactly the same. Thus, Eq. (21) represents the total variation in all motion vectors from the modal vector. A camera panning is declared if a \( \theta_m \) satisfying Eq. (21) exists. \( \Theta_p \) is empirically defined and set to \( 2 \cdot L \) in this paper.

In the case of zooming, the motion vector field has a minimum point at the focus center. Ideally, if the focus center is located in the center of the frame and there is no object motion, then the mean of all the motion vectors will be zero. However, such a case is not usual, and the computational cost to find the focus center is also heavy. Thus, if we assume that the focus center of a camera zoom exists within a frame boundary, we may apply a simpler

FIG. 15. Scatter plots of frame comparison measures in the 2-D feature space for dissolve detection; (a) the scatter plot after the first clustering and (b) after the second clustering. The News1 video clip is used.
vector comparison technique. In this approach, we compare the vertical components of the macroblock motion vectors for the top and bottom rows of a frame, since these vertical components will have opposite signs during a zoom. In other words, the magnitude of the difference between the vertical components of the top and bottom macroblocks within the same column, will always exceed the magnitude of the vertical component of each macroblock. Then the following condition can be derived [3]:

$$|v_{top}^k - v_{bot}^k| \geq \max(|v_{top}^k|, |v_{bot}^k|).$$

(22)

Let \((u_k, v_k)\) denote a motion vector of the \(k\)th macroblock. The horizontal components of the motion vectors for the left-most and right-most columns can then be examined in the same way. That is, the vectors of two corresponding macroblocks located at the left-most and right-most columns would satisfy the following condition:

$$|u_{left}^k - u_{right}^k| \geq \max(|u_{left}^k|, |u_{right}^k|).$$

(23)

Therefore, we assume that a zoom occurs if the above conditions in Eqs. (22) and (23) are satisfied for the majority of motion vectors of the boundary macroblocks, i.e., \(B\%\).

IV. EXPERIMENTAL RESULTS

We used five various MPEG-1 compressed videos. Three of them were MPEG-7 test sequences; two daily and weekly TV news programs from RTVE (News1 and News2), and one miscellaneous program (Misc2). The remaining two sequences were a video clip extracted from the movie “Jurassic Park” (Jurassic park) and a sports scene collection from a Korean broadcasting company (Sports). The MPEG-7 sequences have a frame size of \(352 \times 288\) (25 Hz), and the others have a frame size of \(352 \times 240\) (30 Hz). Table 2 shows specifications of the test sequences. Note that all the video clips except the Jurassic park sequence include gradual shot changes such as fades/dissolves along with cuts.

We choose the first two news clips among the MPEG-7 test sequences because they have many more dissolves than any other test clips. The clips include lots of sports scenes and dialogue scenes. In general, sports scenes contain many shots with fast object motions, camera panning, etc. Hence the Sports sequence also has fast object motions and camera motions. Most parts of the Jurassic park sequence consist of high action shots under poor lighting conditions, e.g., a running dinosaur under a flash of lightning. In addition, the Misc2 sequence includes several concert scenes where very active dancers and singers perform.

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>Test Video Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (mm:ss)</td>
<td>38:18</td>
</tr>
<tr>
<td>No. of cuts</td>
<td>297</td>
</tr>
<tr>
<td>No. of fades</td>
<td>1</td>
</tr>
<tr>
<td>No. of wipes</td>
<td>9</td>
</tr>
<tr>
<td>No. of dissolves</td>
<td>26</td>
</tr>
<tr>
<td>No. of shots</td>
<td>334</td>
</tr>
<tr>
<td>Average shot length (s)</td>
<td>6.9</td>
</tr>
</tbody>
</table>
A good temporal segmentation method should minimize the number of false detections while maximizing the number of correctly identified shot changes. The two measures, recall and precision, are usually used to evaluate shot change detection methods [7]. In retrieval systems, recall reflects the proportion of relevant material that is retrieved, whereas precision is a measure of how relevant the retrieved or selected information is.

\[
\text{Recall} = \frac{N_{\text{correct}}}{N_{\text{correct}} + N_{\text{missed}}},
\]

\[
\text{Precision} = \frac{N_{\text{correct}}}{N_{\text{correct}} + N_{\text{false positive}}},
\]

where \(N_{\text{correct}}\), \(N_{\text{missed}}\), and \(N_{\text{false positive}}\) denote the numbers of correctly retrieved shot changes, missed ones, and false positives, respectively. Recall and precision jointly rate the performance of a classification/retrieval technique, and a successful method produces recall and precision values that are close to unity.

Recently, Gargi et al. showed that the algorithm by Yeo and Liu [12] provides the best performance in dissolve detection as well as cut detection among the latest shot change detection algorithms [21]. Therefore, to evaluate the shot change detection performance, the proposed algorithm is compared with Yeo and Liu’s algorithm by setting the parameters as recommended in [12]. The two algorithms are implemented on an MPEG-2 software decoder distributed by the MPEG software simulation group in 1996 [22]. This MPEG decoder is modified to extract luminance DC images from the test video bit-streams, and all the features for shot change detection are generated from the decoded DC images.

Table 3 shows that the proposed cut detection scheme provides a recall rate of 96.7% and a precision rate of 99.0% on average. The proposed algorithm has precision rates similar to Yeo and Liu’s algorithm, but it provides much better recall rates. Especially, it is noted that the latter algorithm misses a lot of real cuts for high-action videos such as the Jurassic park sequence. These high recall and precision rates of the proposed cut detection scheme are very desirable because the results of cut detection directly influence those of dissolve detection.

<table>
<thead>
<tr>
<th>Video sequences</th>
<th>Proposed algorithm</th>
<th>Yeo and Liu [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall(%)</td>
<td>Precision(%)</td>
</tr>
<tr>
<td>News1</td>
<td>97.3</td>
<td>99.3</td>
</tr>
<tr>
<td></td>
<td>(289/297)</td>
<td>(289/291)</td>
</tr>
<tr>
<td>News2</td>
<td>93.9</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>(92/98)</td>
<td>(92/92)</td>
</tr>
<tr>
<td>Sports</td>
<td>97.4</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>(151/155)</td>
<td>(151/151)</td>
</tr>
<tr>
<td>Jurassic park</td>
<td>93.8</td>
<td>94.5</td>
</tr>
<tr>
<td></td>
<td>(120/128)</td>
<td>(120/127)</td>
</tr>
<tr>
<td>Misc2</td>
<td>98.0</td>
<td>99.7</td>
</tr>
<tr>
<td></td>
<td>(300/306)</td>
<td>(300/301)</td>
</tr>
<tr>
<td>Average</td>
<td>96.7</td>
<td>99.0</td>
</tr>
<tr>
<td></td>
<td>(952/984)</td>
<td>(952/962)</td>
</tr>
</tbody>
</table>
To determine the proper values of parameters $\delta$ and $B$ in the dissolve detection (see Subsections III.B.2 and 4), we examine the performance of the proposed algorithm for various values. As shown in Table 4, $\delta$ of 200 and $B$ of 70% are found to give the best dissolve detection performance regardless of four test video sequences containing dissolves, and are adopted for the experiment.

Table 5 shows that the proposed dissolve detection scheme provides a recall rate of 84.6% and a precision rate of 71.3% on average, which are considered good performances for practical TV clips according to Linehart [14]. A few missed dissolves occur because several dissolves do not have a U-shaped pattern on the edge energy diagram. On the other hand, false positives sometimes occur due to wipes, missed cuts, and fast motions of large objects. In comparison to Yeo and Liu’s algorithm, the proposed algorithm improves the average recall and precision rates by up to about 20 and 47%, respectively. This indicates that dissolve detection using clustering outperforms dissolve detection using adaptive thresholding. It is also interesting to note that even for the Jurassic park sequence, having no gradual shot changes, both algorithms produce 7 and 9 false alarms, respectively, which may result from fast object motions and camera motions.

Table 6 compares the processing time of the proposed algorithm and Yeo and Liu’s algorithm for the News1 sequence. Here, processing time includes the time consumption for extracting DC images (or edge images) and features such as histograms. We find that for both algorithms, processing is performed two times as fast as the real-time display rate because it takes about 1200 s for the News1 sequence whose running time is 2298 s (see Table 2). The table also shows that most of the processing time is spent on extracting DC images from

| TABLE 4 | Effects of $\delta$ and $B$ on the Overall Performance of Dissolve Detection |
|---|---|---|---|---|---|---|---|
|       | News1 |       | News2 |       | Sports |       | Misc2 |
| $\delta = 100$ | Recall (%) | Precision (%) | Recall (%) | Precision (%) | Recall (%) | Precision (%) | Recall (%) | Precision (%) |
|          | 80.8  | 65.6  | 82.4  | 73.7  | 78.9  | 71.4  | 79.3  | 74.2  |
| $\delta = 200$ | 84.6  | 75.9  | 88.2  | 78.9  | 84.2  | 76.2  | 82.8  | 80.0  |
| $\delta = 300$ | 80.8  | 75.0  | 76.5  | 72.2  | 78.9  | 71.4  | 79.3  | 74.2  |
| $B = 50\%$     | 80.8  | 70.0  | 82.5  | 73.7  | 78.9  | 68.2  | 82.8  | 80.0  |
| $B = 70\%$     | 84.6  | 75.9  | 88.2  | 78.9  | 84.2  | 76.2  | 82.8  | 80.0  |
| $B = 90\%$     | 80.8  | 67.7  | 76.5  | 68.4  | 78.9  | 68.2  | 79.3  | 74.2  |

| TABLE 5 | Recall and Precision Rates of Dissolve Detection for Test Sequences for $\delta = 200$ and $B = 70\%$ |
|---|---|---|---|---|---|---|
|       | Proposed algorithm | Yeo and Liu [12] |
|       | Recall(%) | Precision(%) | Recall(%) | Precision(%) |
| News1  | 84.6 (22/26) | 75.9 (22/29) | 73.1 (19/26) | 55.9 (19/34) |
| News2  | 88.2 (15/17) | 78.9 (15/19) | 76.4 (13/17) | 50.0 (13/26) |
| Sports | 84.2 (16/19) | 76.2 (16/21) | 68.4 (13/19) | 46.4 (13/28) |
| Jurassic park | — | 0 (0/9) | — | 0 (0/7) |
| Misc2  | 82.8 (24/29) | 80.0 (24/30) | 65.5 (19/29) | 51.4 (20/37) |
| Average | 84.6 (77/91) | 71.3 (77/108) | 70.3 (64/91) | 48.5 (64/132) |
TABLE 6
Processing Time Comparison for the News1 Sequence

<table>
<thead>
<tr>
<th></th>
<th>Proposed algorithm</th>
<th>Yeo and Liu [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC image extraction</td>
<td>1138</td>
<td>1138</td>
</tr>
<tr>
<td>Cut detection</td>
<td>11.2</td>
<td>11.9</td>
</tr>
<tr>
<td>Dissolve detection</td>
<td>66.8</td>
<td>16.1</td>
</tr>
<tr>
<td>Total</td>
<td>1216</td>
<td>1166</td>
</tr>
</tbody>
</table>

Note. Running time for News 1 sequence is 2298 s.

compressed bit-streams, which is commonly needed for compressed domain shot change detection. The remaining processing time is relatively negligible for both algorithms.

V. CONCLUDING REMARKS

This paper proposed an unsupervised clustering-based algorithm for shot change detection in MPEG-compressed videos. Edge images as well as DC images were extracted by using fast and efficient algorithms. Then, multiple features obtained from those DC and edge images were utilized for the accurate detection of shot changes. In order to perform the detection, we presented a two-stage clustering-based scheme. First, we detect abrupt shot changes such as cuts by employing a two-means clustering scheme on the 2-D feature space of histogram difference and pixel difference. Subsequently, we explore dissolves between adjacent cuts by performing a two-step clustering for multiple features such as an edge energy diagram and several frame difference features. The adopted hierarchical search structure makes the overall processing speed fast. Simulation results show that the proposed algorithm also provides prospective detection accuracy.

REFERENCES


**BYUNG CHEOL SONG** received the B.S., M.S., and Ph.D. degrees in electrical engineering from KAIST in 1994, 1996, and 2001, respectively. Since 2001, he has been working for Samsung Electronics. His research interests include image/video compression and representation.

**JONG BEOM RA** received the B.S. degree in electronic engineering in 1975 from Seoul National University, and the M.S. and Ph.D. degrees in electrical engineering from KAIST in 1977 and 1983, respectively. From 1983 to 1987, he served as an associate research scientist in Columbia University in New York. Since 1987, he has been working as a professor in the department of electrical engineering, KAIST. His research interests are digital image processing, video signal processing, 3-D visualization, and medical imaging such as MRI.