LETTER

Dissolve Detection Using Intensity Change Information of Edge Pixels

Chul-Hyun KWON(a), Student Member, Doo-Jin HAN†, Hyun-Sool KIM†, Myung-Ho LEE†, and Sang-Hui PARK†, Nonmembers

SUMMARY Shot transition detection is a core technology in video browsing, indexing systems and information retrieval. In this paper we propose a dissolve detection algorithm using the characteristics of edge in MPEG compressed video. Using the intensity change information of edge pixels obtained by Sobel edge detector, we detect the location of a dissolve and its precise duration. We also present a new reliable method to eliminate the false dissolves. The proposed algorithm is tested in various types of videos, and the experimental results show that the proposed algorithm is effective and robust.

key words: shot transition detection, dissolve detection

1. Introduction

The integration of information from various sources and the emergence of digital libraries have increased the importance of browsing multimedia information in the form of still images and videos in interactive multimedia system. In such circumstances, shot transition detection is a core technology for content-based video retrieval such as video indexing and browsing. A shot is defined as a continuous sequence of frames taken by a single camera. When dividing video into shots, the key is to differentiate shot transition from the normal changes in a shot. Shot transition can be divided into abrupt transition and gradual transition such as fade and dissolve. Generally gradual transitions are more difficult to detect than abrupt transitions. It is particularly difficult to detect dissolves between sequences involving intensive motion. So early studies mostly focused on abrupt transition detection, and provided some level of completed results. Recent studies have focused on detecting gradual shot transition. Also, early studies on shot transition detection focused on uncompressed video. The studies suggested methods such as pair-wise pixel comparison, block-based comparison, histogram comparison, clustering-based division, feature-based division and model driven division[1]–[5]. As nowadays video is increasingly stored and transferred in a compressed format (MPEG), it is better to detect shot transition directly in compressed video. There are various methods of detecting shot transition in compressed video based on the type of information used, such as 1) DCT coefficients 2) DC terms 3) DC terms, MB coding mode and motion vector 4) DCT coefficients, MB coding mode and motion vector 5) MB coding mode and motion vector and 6) MB coding mode and bit rate information[5]–[8]. We propose a dissolve detection algorithm using the characteristics of edge based DC information in compressed video. It is important to detect dissolves because dissolve is the most common gradual shot transition. Zabih et al.'s algorithm is a well known method of detecting shot transitions using edge. Their algorithm detects shot transitions by considering the entering edge pixels and the exiting edge pixels between consecutive frames[3]. In this paper we propose a simpler and better algorithm. We use intensity change information of edge pixels obtained by Sobel edge detector. We first detect dissolve candidates without comparing between neighboring frames. And we detect final dissolves using a new reliable method to eliminate the false dissolves from detected dissolve candidates.

2. Dissolve Detection

2.1 DC Image Sequence

We first reconstruct a spatially-reduced image sequence, called DC image sequence from MPEG video data. The resulting DC sequence is reduced to 1/64. Though DC sequence is greatly reduced, it captures meaningful global image features. While the reconstruction of such reduced image sequence is a trivial task for I-type frames, it is not straightforward for the P-type frames and B-type frames. We use Yeo and Liu's algorithm in reconstructing DC image sequence from MPEG video[7].

2.2 Mathematical Model for Dissolve

In video editing, a shot A is often combined with another shot B in a video. Considering the characteristic of such combined signals, sometimes the signal of shot A changes from 100% to 0%, while the signal of shot B changes form 0% to 100%. This is the process of dissolve. Mathematically, dissolve can be expressed as follows.

\[ S_n(x,y) = \begin{cases} 
  f_n(x,y) & 0 \leq n < L_1 \\
  A(n)f_n(x,y) + B(n)g_n(x,y) & L_1 \leq n \leq (L_1 + F) \\
  g_n(x,y) & (L_1 + F) < n \leq L_2 
\end{cases} \tag{1} \]

\[ A(n) = 1 - \left( \frac{n - L_1}{F} \right)^d, \quad B(n) = \frac{n - L_1}{F} \]


The authors are with the Department of Electrical and Electronics Engineering, Yonsei University, Seoul 120–749, Korea.
a) E-mail: wall@naver.com
DOI: 10.1093/ietisy/e91-d.1.153

Copyright © 2008 The Institute of Electronics, Information and Communication Engineers
Where, $S_n(x, y)$ is the signal of resultant video signal, $f_n(x, y)$ is shot A, $g_n(x, y)$ is shot B, $L_1$ is the length of shot A, $F$ is the length of dissolve sequence and $L_2$ is the length of the total sequence [9].

### 2.3 Dissolve Detection

The edge is the part where the intensity of pixel changes drastically. In this paper, DC sequence goes through Sobel edge detector with $3\times3$ mask to get the edge function $E_n(x, y)$. Then, according to threshold applied, the edges of each frame are determined. Let the number of edge pixels in a frame $NEP(n)$. If the threshold gets higher, $NEP(n)$ decreases. On the other hand, if the threshold gets lower, $NEP(n)$ increases. Generally, $NEP(n)$ in a shot changes smoothly. Although the threshold changes, $NEP(n)$ in a shot keeps changing smoothly. However, $NEP(n)$ changes drastically when abrupt transitions occur. Though not as drastic as abrupt transitions, $NEP(n)$ also changes somewhat during dissolving. But it is difficult to detect shot transitions using only this information, especially in dissolve. At this point, the mathematical model of dissolve defined in Eq. (1) should be reconsidered. Figure 1 shows an example of a dissolve. A typical dissolve becomes more complicated as it moves towards the center frame of dissolve. At the same time, the overall edge of frame becomes blurry because both shot A and shot B represent only 50% of their signals at the center frame of dissolve as shown Fig. 1 (b). In other words, as moving towards the center frame of dissolve, the intensity of almost all pixels in a frame changes smoothly. Thus the values of $E_n(x, y)$ are not large overall as moving towards the center frame of dissolve. As threshold increases, $NEP(n)$ decreases drastically as moving towards the center frame of dissolve. This indicates that edge detection in dissolve is greatly influenced by threshold. We use this property to detect dissolves.

We use two thresholds, the lower threshold $T_1$ and the upper threshold $T_2$, to detect edge of DC frames. Our proposed algorithm is not to merely count the number of edge pixels exceeding each threshold but to consider the value of $E_n(x, y)$, which contains the information on the intensity change of edge pixels in a frame. This is shown in Eq. (2). It adds all differences between the threshold and $E_n(x, y)$ exceeding the threshold to consider the intensity change of edge pixels in DC frames.

$$D(n) = \sum_{x=1}^{w} \sum_{y=1}^{h} \max(E_n(x, y) - T, 0) \quad 0 \leq n < N$$ (2)

Where, $T$ is threshold, $h$ is the number of pixels in the vertical direction of DC frame, $w$ is the number of pixels in the horizontal direction of DC frame and $N$ is the total number of frames. By doing so, good performance on various types of videos can be obtained while applying fixed $T_1$ and $T_2$. Let $D(n)$ is applied with $T_1$ and $T_2$. Let $D_1(n)$ and $D_2(n)$ respectively. In order to minimize the influence of noise, a smoothing process is applied to calculate $D_1(n)$ and $D_2(n)$. (When there is a large difference in $D(n)$ between neighboring frames, the smoothing process is omitted.) Then, the ratio of $D_1(n)$ and $D_2(n)$, $R(n)$ is calculated as follows.

$$R(n) = \frac{D_1(n)}{D_2(n)} \quad 0 \leq n < N$$ (3)

Although $D_1(n)$ and $D_2(n)$ change respectively in a shot, the pattern of change is almost identical, thus $R(n)$ doesn’t change much. The flat area in Fig. 2 (c) shows such stability. However, during dissolving, $D_2(n)$ decreases more steeply than $D_1(n)$ as moving towards the center frame of dissolve. As a result, $R(n)$ in dissolve forms the shape of a hill, as shown in Fig. 2 (c). Therefore the candidates of dissolve can be detected at outstanding local maxima. The duration of a dissolve is from the frame where the hill starts to the frame where it ends. If the duration of a candidate is below $\theta_1 (= 5)$ frames, we regard it as non-dissolve and eliminate it.

![Fig. 1](image1.png) ![Fig. 2](image2.png)

**Fig. 1** An example of dissolve: (a) first frame, shot A = 100%, shot B = 0 (b) center frame, shot A = 50%, shot B = 50% (c) last frame, shot A = 0, shot B = 100%.

**Fig. 2** (a) $D_1(n)$, (b) $D_2(n)$, (c) $R(n)$. 
2.4 Elimination of False Dissolves

The proposed method \( R(n) \) detects dissolve candidates using the property that the overall edges of a frame become blurry as moving towards the center frame of dissolve. When a camera is out of focus instantly, or when a camera or objects move fast, frames are blurred and sometimes show similar edge characteristics with dissolve, so they are detected as dissolve candidates (false dissolves). To solve the problem of these false dissolves, a lot of methods were proposed before. While these methods eliminate a lot of false dissolves, they also eliminate a lot of correct dissolves. To eliminate correctly false dissolves from candidates, we propose the new and robust method using the pair-wise pixel comparison method [7]. The difference between two frames \( F, G \) is denoted by \( PD(F, G) \).

\[
PD(F, G) = \sum_{x,y} |f_{x,y} - g_{x,y}|
\]

(4)

\( F = \{f_{x,y}\}, G = \{g_{x,y}\} \)

A dissolve is the process that \( f_{L_1}(x, y) \) changes gradually into \( g_{L_1+1}(x, y) \) as shown Eq. (1). If \( f_n(x, y) \) and \( g_n(x, y) \) are constant without any motion during dissolving, the total sum of pixel differences between all consecutive frames during dissolving is as follows.

\[
\sum_{n=L_1}^{L_1+F-1} |S_n(x, y) - S_{n+1}(x, y)| = \left| f_{L_1}(x, y) - \left(1 - \frac{1}{F}\right)f_{L_1+1}(x, y) - \frac{1}{F}g_{L_1+1}(x, y) \right| + \]

\[
\vdots
\]

(5)

\[
= \frac{1}{F}f_{L_1}(x, y) + \frac{F-1}{F}g_{L_1+1}(x, y) - g_{L_1+1}(x, y)
\]

(6)

\[
= \left| f_{L_1}(x, y) - g_{L_1+1}(x, y) \right|
\]

If \( f_n(x, y) \) and \( g_n(x, y) \) are constant, \( L_1 \leq n \leq L_1 + F \)

Finally, during dissolving the total sum of pixel differences between all consecutive frames is equal to that between the first frame and the last frame of dissolve. Therefor, we can get the following equations as follows.

\[
\sum_{n=L_1}^{L_1+F-1} PD(S_n, S_{n+1}) = PD(f_{L_1}, g_{L_1+1})
\]

(6)

\[
However, this is only when \( f_n(x, y) \) and \( g_n(x, y) \) are constant without any motion during dissolving. In general, as \( f_n(x, y) \) and \( g_n(x, y) \) move during dissolving, \( \sum_{n=L_1}^{L_1+F-1} PD(S_n, S_{n+1}) \) becomes larger. So

\[
\sum_{n=L_1}^{L_1+F-1} PD(S_n, S_{n+1}) \geq PD(f_{L_1}, g_{L_1+1})
\]

(7)

During dissolving, the motions are so small that their difference is not large. But, \( \sum_{n=L_1}^{L_1+F-1} PD(S_n, S_{n+1}) \) is much larger than \( PD(f_{L_1}, g_{L_1+1}) \) in false dissolves with motions of object or camera like close-up, zoom-in, zoom-out, panning, and so on. To detect false dissolves from candidates, we first calculate \( R_{diff} \) as follows.

\[
R_{diff} = \frac{PD(f_{L_1}, g_{L_1+1})}{\sum_{n=L_1}^{L_1+F-1} PD(S_n, S_{n+1})}
\]

(8)

If \( R_{diff} \) of a candidate is larger than threshold \( \theta_f \), it is regarded as a dissolve. Otherwise, a candidate is regarded as a false dissolve and eliminated from candidates. We choose \( \theta_f \) considering the general movement during dissolving. We show some examples in Fig. 3. Figure 3 (a) is a false dissolve detected by method \( R(n) \) that is blurred by fast object motion. Figure 3 (b) is a false dissolve that is blurred because a camera moves up and down and is out of focus. They both are not dissolves so that they should be eliminated from candidates. As their \( R_{diff} \) both are below threshold \( \theta_f \), they are eliminated from candidates. A lot of false dissolves among candidates are eliminated by method \( R_{diff} \). But sometimes real dissolves are eliminated by method \( R_{diff} \). Figure 3 (c) is a dissolve which has very large objects motion during dissolving. Therefore its \( R_{diff} \) is smaller than threshold \( \theta_f \) so that it is defined a false dissolve and eliminated from candidates. Finally it is classified a missed dissolve.

3. Results

In this section, we evaluate the performance of our proposed algorithm. To test the effectiveness of the proposed algorithm, the algorithm was used on 90 minutes MPEG-2 video
at a resolution of 352*240. We chose a video containing a lot of dissolves because the goal of our algorithm is to detect dissolves. The video is clipped from English BBC Broadcasting and American CNN Broadcasting. Overall, there are 184 dissolves with various lengths in the test data set. To achieve optimal results, we chose thresholds through experiment. From the range of 20 ~ 70 for the lower threshold \( T_1 \), and from the range of 150 ~ 250 for the upper threshold \( T_2 \), we select each threshold showing optimal results. As a result, we set 45 for \( T_1 \) and 210 for \( T_2 \). And we set 0.5 for \( \theta_f \) to eliminate false dissolves with motions of objects or camera. Also, instead of using all continuous frames, frames are processed with a fixed interval (= 2). The performance of a shot transition detection algorithm is usually expressed in terms of recall and precision. The recall and precision are defined as

\[
Recall = \frac{Hits}{Hits + Missed} \times 100
\]

\[
Precision = \frac{Hits}{Hits + False} \times 100
\]

(9)

The proposed algorithm is compared with Zabih et al.’s algorithm. Their algorithm first detects intensity edges in each frame and then it goes through the process of dilation considering the movement of video. And it counts the entering pixels and the exiting pixels between consecutive frames and calculates edge change fraction. A shot transition is declared if edge change fraction is high [3]. On the other hand, the algorithm proposed in this paper detects intensity edges in each DC frame using Sobel edge detector and then only uses the intensity change information of edge pixels exceeding threshold without the process of dilation and comparison between neighboring frames. And we raise precision using the method that eliminates false dissolves considering the property of dissolve. Figure 4 shows the distribution of \( R_{diff} \) to dissolves and non-dissolves detected by method \( R(n) \) in our experiments. 97.2% of \( R_{diff}s \) to dissolves are over \( \theta_f \), while 83.2% of \( R_{diff}s \) to non-dissolves are below \( \theta_f \). The experimental results are summarized in Table 1. The result of dissolve detection using Zabih et al.’s method is 71.7% for recall and 34.5% for precision. This algorithm don’t eliminate false dissolves effectively so that it shows too low precision. The result using just method \( R(n) \) without method \( R_{diff} \) is 96.7% for recall and 63.7% for precision. It shows much better recall but still low precision due to many false dissolves. While the result of our proposed algorithm is 93.5% for recall and 91.0% for precision. We could achieve better precision because the method proposed to eliminate false dissolves is effective. Our proposed algorithm couldn’t eliminate false dissolve with instant blurring and little motion, which has similar characteristic with dissolve in all aspects. But our proposed algorithm detects most of the dissolves correctly and eliminates a lot of false dissolves. We performed Zabih et al.’s algorithm and our algorithm on a T2500 (2.0GHz) with 1024 MB of RAM. Zabih et al.’s algorithm requires about 0.022 seconds to process one frame and our proposed algorithm requires about 0.008 seconds to process one frame. Our proposed algorithm can achieve 2.8 times faster than Zabih et al.’s algorithm.

4. Conclusion

In this paper we presented a dissolve detection algorithm in MPEG compressed video and gave experimental results to evaluate the performance. Using the property that the overall edges of a frame become blurry as moving towards the center frame of dissolve, our algorithm detects the location of a dissolve as well as the duration of a dissolve without the procedure of comparing between neighboring frames. And using the proposed method that eliminates the false dissolves, we achieved outstanding precision compared with existing algorithms. Experimental results show that the proposed algorithm is effective in detecting dissolves using edge. But the proposed method eliminates dissolves with very large motions of objects or camera. For future work, we plan to research a method to discriminate between dissolves with large motion and non-dissolves with large motion.

References


