A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT
Jun Li, Youdong Ding, Yunyu Shi, Wei Li

A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT

Jun Li 1, Youdong Ding 1, Yunyu Shi 1, Wei Li 2
1. School of Computer Engineering and Science, Shanghai University, Shanghai, China
2. School of Communication and Information Engineering, Shanghai University, Shanghai, China
vbli@163.com
doi:10.4156/jdcta.vol4.issue3.20

Abstract
Shot boundary detection is an important fundamental process toward automatic video indexing, retrieval, editing, etc. After a critical review of most approaches seeking to solve this problem, we propose a novel shot boundary detection. To improve the performance of the algorithm and reduce the computational cost, frames that are clearly not shot boundaries are first removed from the original video. After that, a novel SIFT keypoint matching algorithm based on SVM is proposed, which is used to capture the changing statistics of different kinds of shot transitions so as to identify, not only abrupt transitions, but also gradual transitions (fade, dissolve, wipe) accordingly. At last, our system use different algorithms for different kinds of shot transitions to help us to get a better solution for shot boundary detection problem. Numerical experiments in the evaluation of TRECVID and a variety of film videos demonstrate that our method is capable of accurately detecting shot transitions, and could greatly reduce the computational cost.

Keywords: Shot boundary detection, video retrieval, Support Vector Machine (SVM), Scale Invariant Feature Transform (SIFT)

1. Introduction

Due to the recent progress in the decreasing storage costs and the growing availability of broadband data connection, digital videos are becoming widely used. However, the increasing availability of digital video has not been accompanied by an increase in its accessibility. If we want to find a clip of interest, we have to sequentially browse through the video. This is an extremely time consuming, tedious and labor-intensive process [1] [2]. Therefore, the demand of new technologies and tools for effective and efficient indexing, browsing and retrieval of video data has been exacerbated by recent trends [3] [4].

Content-based video retrieval is an effective and efficient technology for automate the indexing, retrieval and management of video, and therefore has attracted extensive research during the last years. The foundation step of content-based video retrieval is shot boundary detection. A shot is defined as a sequence of frames taken by a single camera with no major changes in the visual content. Shot boundaries can be broadly classified into two types: abrupt transition and gradual transition. Abrupt transition is instantaneous, where frame $f_i$ belongs to one shot and $f_{i+1}$ to the next shot, a clear discontinuity therefore existing [5]. Gradual transition occurs over multiple frames, which is generated via the application of more elaborated editing effects involving several frames, so that $f_i$ frame belongs to one shot, frame $f_{i+N}$ to the second, and the $N-1$ frames in between represent a gradual transformation of $f_i$ into $f_{i+N}$ [5]. Gradual transition can be further classified into fade out/in(FOI) transition, dissolve transition, wipe transition, and others transition, according to the characteristics of the different editing effects [1] [3]. The task of shot boundary detection is to identify the shot boundaries with their location and type in the given video clip(s) [6].

To shot boundary detection, many efforts have been devoted into this area for the past years, and many different methods have been proposed. The initial way to check whether two frames are significantly different is the direct comparison of the pixels in the consecutive frames [7] [8] [9]. If the number of different pixels is large enough, the two processed frames are declared to belong to different shots. The pixel-based method is easy and fast. But it is extremely sensitive,
since it has captured any details of the frame, such as highly sensitive to local motion, camera
motion and minor changes in illumination [1] [3]. To handle these drawbacks, several
ameliorative methods have been proposed, for example luminance/color histogram-based
method and edge-based method.

Histogram-based method uses the statistics of the luminance and color [10]. The advantage
of the histogram-based shot change detection is that it is quite discriminant, easy to compute,
and mostly insensitive to translational, rotational, and zooming camera motions. For these
reasons, it is widely used. The weakness of the histogram-based shot boundary detection is that
it does not incorporate the spatial distribution information of various color, hence it will fail in
the case which similar histograms but different structures [1]. A better tradeoff between pixel
and global color histogram methods can be achieved by block-matching methods [12] [13], in
which each frame is divided into several nonoverlapping blocks and luminance/color histogram
feature of each block are extracted.

The edge information is an obvious choice for characterizing image [1] [10] [14]. The
advantage of this feature is that it is sufficiently invariant to illumination changes and several
types of motion, and it is related to the human visual perception of a scene. Its main
disadvantages are computational cost and noise sensitivity [1].

Most of the methods have used the motion information. The motions, either object motions
or camera motions, exist in almost all video sequences. In order to distinguish shot changes due
to motion from those due to a shot boundary, differential motion based algorithms have been
proposed. Park [15] presents a shot boundary detection based on the combination of two motion
features: the modified displaced frame difference (DFD) and the blockwise motion similarity.
T.LU [16] proposed an algorithm using motion compensation. As the main source of difference
due to the motions can be eliminated by motion compensation, the remaining difference is due
to likely to be a shot boundary.

Another important method has been used is transform coefficients. Dai et al. [17] proposed
an algorithm of video shot detection based on partition in image wavelet entropy. Feng et al.
[18] adopted the wavelet coefficient vectors within a sliding window as the features of the shot
boundary detection system. Yu Wolf [2] presented a hierarchical multiresolution video shot
boundary detection scheme, which used wavelet to decompose every frame into low-resolution
and high-resolution components to detect shot transition.

Thanks to the good function learning and generalization capability, there have been some
efforts turning to the tools of statistical machine learning (e.g., FCM, SVM, KNN) recently. For
example, in [3], [11], and [19], the shot boundary was classified by SVM. In [20], the shot
boundary was detected by K-means. Juan et al. [26] combined the merits of the generalization
ability of the SVM and the comprehensibility of the fuzzy logic.

From above discussion, we can see that many efforts have been undertaken to detect different
kinds of shot boundaries, and a lot success has been achieved. However, so far, there are no
techniques of shot detection have been able to achieve the very ideal result in real applications.
The main challenge is that conventional methods are not robust against camera operations,
object motions, and illumination changes, which often cause erroneous shot boundary detection.
Another challenge is the high computational cost of calculating all the features for each frame.

This paper proposes a new and precise shot boundary detection method. To reduce the
detection time, the algorithm first uses the block color histogram difference between two frames
to remove the frames that are clearly not shot boundaries from the original video. In this
procedure, a new frame sequence named STC is constructed. Next, features are extracted from
each frame of STC to detect shot transitions. The camera operations (e.g. zoom in/out, pan and
tilt), object motions, and the illumination changes often cause false detections on traditional
shot change detection system. SIFT (Scale Invariant Feature Transform) algorithm is invariant
to scaling, translation, rotation, and partially invariant to illumination changes, affine distortion,
addition of noise and even partial occlusion [21]. Hence, SIFT algorithm is adopted to capture
the different characteristics for different kinds of shot transitions. In order to improve the
performance of the algorithm we propose using SVM to address the keypoints matching scheme
in this paper. Furthermore, our system adopts different algorithms for different kinds of shot
transitions, for different shot transitions have different characteristics and it is hard to use one
single algorithm to detect all kinds of shot transitions efficiently. Finally, the experiments are carried out on TRECVID test data and a variety of film videos.

2. Feature Extraction

Illumination change, camera operation (e.g. zoom in/out, pan and tilt), and object motion are the main reasons that cause false detections on traditional shot change detection system. The extracted features should be robust against them in order to increase the efficiency of shot boundary detection. The SIFT algorithm extracts distinctive features which act as descriptors of local image patches. These features are reasonable invariant to scaling, translation, and rotation, and partially invariant to illumination changes, affine distortion, addition of noise and even partial occlusion [21]. Hence, SIFT is chosen to extract the features to detect the shot boundary in the proposed system.

2.1. SIFT

The Scale Invariant Feature Transform (SIFT) algorithm extracts distinctive features which act as descriptors of local image patches. The major stages of computation used to generate the set of image feature, as described in [21], consists of four stages:

1) Detection of Scale-Space Extrema

The SIFT feature algorithm is based upon finding keypoints within the scale space of an image which can be reliably extracted. Therefore, the first stage finds scale-space extrema located in the Difference of Gaussians (DOG) function, \( D(x, y, \sigma) \). DOG can be computed from the difference of two nearby scaled images separated by a multiplicative factor:

\[
D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \ast I(x, y) - L(x, y, \sigma)
\]

\[
G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

where \( L(x, y, \sigma) \) is the scale space of an image, built by convolving the image \( I(x, y) \) with the Gaussian kernel \( G(x, y, \sigma) \).

Once the DOG images have been obtained, a feature point is identified as local minima/maxima by comparing it to its 26 neighbors, 9 above, 9 below, and 8 surrounding. If it is the highest or lowest pixel value, it is selected as a local extrema or "keypoint candidate".

2) Accurate Keypoint Localization

In this step the keypoints are filtered so that only stable keypoints are retained. Performing a detailed fit to the nearby data for location, scale, and ratio of principal curvatures allows the algorithm to filter out points that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

3) Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. This operation is performed for each scale \( \sigma \). For each sample \( L(x, y) \) at a certain scale \( \sigma \), the vector/gradient magnitude, \( m(x, y) \) and orientation \( \theta(x, y) \) are calculated below

\[
f(x) = (L(x+1, y) - L(x-1, y))^2
\]

\[
f(x) = (L(x, y+1) - L(x, y-1))^2
\]

\[
m(x, y) = \sqrt{f(x) + f(x)}
\]

\[
\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)
\]
A keypoint descriptor is created by using clusters of descriptors containing magnitude and orientation information which enables the algorithm to search for the same keypoint in future image frames.

4) keypoint descriptor

The previous operations have assigned an image location, scale, and orientation to each keypoint. The next step is to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations. The orientation histogram has 36 bins covering the 360 degree range of orientations. Each point is added to the histogram weighted by the gradient magnitude \( m(x, y) \). Additional keypoints are generated for keypoint locations with multiple local dominant peaks whose magnitude is within 80% of each other. The dominant peaks in the histogram are interpolated with their neighbors for a more accurate orientation assignment.

The keypoint orientation is also calculated from an orientation histogram of local gradients from the closest smoothed image \( L(x, y) \). This gradient information is rotated to align it with the assigned orientation of the keypoint and then weighted by a Gaussian window. The weighted data is used to create a nominated number of histograms over a set window around the keypoint. Keypoint descriptors use 8 orientation histograms aligned in a \( 16 \times 16 \) grid. The resulting feature vectors are 128 elements.

2.2. SVM and SIFT Based Matching Approach to Similarity of Frames

Having obtained SIFT features from two images, an important remaining issue is how to find the matched keypoint between two images. Traditional, the keypoint matching is computed based on Euclidean distance of their feature vectors. A Keypoint matching is accepted only if its distance is less than \( \text{distRatio} \) times the distance to the second closest match. However, it has several difficulties in achieving satisfactory results. First, the chosen \( \text{distRatio} \) usually highly depends on the genres of videos. Second, a single \( \text{distRatio} \) can not make full use of the contextual information of videos. In order to improve the performance of the algorithm we propose using machine learning methods to address the keypoint matching scheme in this paper. As for the selection of machine learning methods, Support Vector Machine (SVM) is preferred, not only for its solid theoretical foundations but also for its various empirical successes [3]. Therefore, SVM is adopted to match the keypoint between two frames in this paper.

SVM is a kind of machine learning method based on the concept of the structural risk minimization using the Vapnik-Chervonenkis (VC) dimension [22]. It has a good performance in non-linear function estimation. Much of the literature seems to indicate that the performance of SVM is superior to that of traditional learning models [23], [24]. In the following, we will introduce how to apply SVM to implement the Keypoint matching.

1) To train an SVM model for the Keypoint matching, we have annotated a training set consisting of positive examples and negative examples.

\[
F = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (X, Y)
\]

where, \( x_i \in \mathbb{R}^{128} \) is keypoint feature vectors, which is SVM input vector. \( y_i \in \{1, -1\} \) is the output vector. We assume that class labeled 1 corresponds to the correct matches, and class labeled -1 to the incorrect ones.

2) The SVM model requires the solution of the following optimization problem:

\[
\begin{align*}
\text{Min} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \chi_i \\
\text{s.t.} & \quad y_i (w^T \phi(x_i) + b) \geq 1 - \chi_i \\
& \quad \chi_i \geq 0, i = 1, \ldots, n,
\end{align*}
\]

Here training vectors \( x_i \) is mapped into a higher (maybe infinite) dimensional space by the function \( \phi \). Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. \( C > 0 \) is the penalty parameter of the error term. Furthermore, \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \) is called the kernel function. Then the SVM model function is obtained by solving the primal problem.
A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT
Jun Li, Youdong Ding, Yunyu Shi, Wei Li

\[ f(x) = \text{sign}(\sum_{i=1}^{w} w_i K(x_i, x) + b) \] (7)

The keypoint matching is performed by the SVM model. The number of the keypoint matching is regarded as the similarity score of two images, denoted by \(NSKM\).

3. Shot Boundary Detection

The shot boundaries included in film programs typically amount to less than 1% of total frames; it is inefficient and extremely time consuming to apply boundary detection processing to detect all the frames [12]. Therefore, our method first removes the frames that are clearly not shot boundaries from original videos, and detects shot transition only for the remainder parts of the video that are likely to contain shot boundaries. For different shot transitions have different characteristics, it is hard to use one algorithm to capture all the characteristics of all kinds of shot transitions efficiently. Our system uses different algorithms for different kinds of shot transitions to help us to get a better solution for shot boundary detection problem. Detection processing is done in serial to improve the performance of the algorithm. The order of detection processing is cuts, fades, wipes, and dissolves. Whenever a transition is detected, the remaining processing for other types of transition is not performed.

The details of each detection processing are explained in the following sections.

3.1. Extract Shot Transition Candidate Frames

The block color histogram difference is a nice tool to check whether two images are significantly similar. Not only is it fast and easy to compute, but also it requires only linear time in the size of the image and very little code. Therefore, in this proposed paper, frames that are clearly not shot boundaries are determined by the block color histogram difference. The details that extract shot transition candidate frames algorithm based on the block color histogram difference are explained here.

First, frame is decomposed by \(M \times N\) block. The block color histogram difference, \(D_{chd}\), is computed as follows:

\[ D_{chd}(b_{i-1,k}, b_{i,k}) = \begin{cases} 1 & \text{if } B_{chd}(b_{i-1,k}, b_{i,k}) \geq T_{chd} \\ 0 & \text{otherwise} \end{cases} \] (8)

\[ B_{chd}(b_{i-1,k}, b_{i,k}) = \sum_{r=0}^{255} |b_{i-1,k}(r) - b_{i,k}(r)| \] (9)

where \(b_{i,k}\) is the \(k\)th block of the \(i\)th frame, \(b_{i,k}(r)\) represents the number that the hue value equals to \(r\) in \(b_{i,k}\), and \(T_{chd}\) is a threshold.

The next step is to calculate the frame color histogram difference, \(F_{chd}\): \n
\[ F_{chd}(f_{i-1}, f_i) = \sum_{k=0}^{L} D_{chd}(b_{i-1,k}, b_{i,k}) \] (10)

where \(L\) is the number of the block. If \(F_{chd}\) is less than a threshold \(T_{chd}\), it is judged that the frame cannot be a shot boundary, and the remaining processing is skipped. In contrast, if \(F_{chd}\) is above the threshold, it is judged that the frame is shot transition candidate and more precise processing is calculated. In this procedure, a new sequence named \(STC\) is constructed.

\[ STC(f_{i-1}, f_i) = \begin{cases} -1 & \text{if } F_{chd}(f_{i-1}, f_i) \geq T_{chd} \\ 0 & \text{otherwise} \end{cases} \] (11)

where \(STC(f_{i-1}, f_i)\) is equal to 0, denotes the frame cannot be a shot boundary. In contrast, \(STC(f_{i-1}, f_i)\) is equal to -1, denotes the frame is the shot transition candidate.

The following detection processing is done only for the parts of \(STC(f_{i-1}, f_i) = -1\).

3.2. Cut Transition Detection
Cut transition is instantaneous transitions from one shot to the subsequent shot, which just involves two consecutive frames of different shots. And the similarity within the same shot always keeps large magnitudes, while drops to low scores within the different shot. Therefore the cut transition can be detected by the similarity between adjacent frames. In our system, the similarity between adjacent frames is obtained by NSKM (the number of the SIFT keypoint matching), for it is robust to luminance changes, object motion, and camera operations for cut transition. Figure 1 displays clearly the NSKM transition over a video sequences that only contain cut transitions. The variance curve of the NSKM between consecutive frames has an abruptly lowest score in the cut region forms.

\[
\text{Cut}(f_{i-1}, f_i) = \begin{cases} 
1 & \text{if } \text{NSKM}(f_{i-1}, f_i) < T_{cut} \\
0 & \text{otherwise}
\end{cases}
\]

If \( \text{Cut}(f_{i-1}, f_i) \) is lower than a threshold \( T_{cut} \), a cut transition is detected, and \( \text{STC}(f_{i-1}, f_i) \) is labeled \( l \).

\[
\text{STC}(f_{i-1}, f_i) = \begin{cases} 
1 & \text{if } \text{Cut}(f_{i-1}, f_i) = 1 \\
0 & \text{otherwise}
\end{cases}
\]

We carried out several experiments on cut transition detection. Experiments show that NSKM between consecutive frames in the cut region is not more than 4.

### 3.3. Fade Transition Detection

A fade of a video sequence is a shot transition with the first shot gradually disappearing (fade out) before the second shot gradually appears (fade in) [2]. During the FOI, two shots are spatially and temporally well separated by some monochrome frames [5].

One of the important properties of the change during a fade out is that visually the image becomes cloudy and the outline becomes vague [2], until monochrome frame, and during a fade in the image becomes clear and the outline becomes vivid. The more clarity the image is, the more number of the frame SIFT keypoint is. This implies that the number of the frame SIFT keypoint is reduced, along with the image becomes cloudy and the outline becomes vague. When it is the monochrome frame, the number of the frame SIFT keypoint is zero. During a shot appear, with the image becomes clear and the outline becomes vivid, the number of the frame SIFT keypoint is increasing. Figure 2 displays clearly the variance transition of the number of the frame SIFT keypoint over the fade frames. As depicted in Figure 2, variance curve in the fade region forms a clear parabolic shape and the variances in the middle frames over the transition have the zero value, whereas this case seldom appear elsewhere. Thus, the FOI transition can be detected by the variance transition of the number of the frame SIFT keypoints.

The details of the FOI detection are explained in the following section.

First, it is determined whether the current frame is a monochrome frame. The number of the frame SIFT keypoint is zero, as shown in Eq. (14), which is used for the determination.
A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT
Jun Li, Youdong Ding, Yunyu Shi, Wei Li

\[ M_{chk}(f_i) = \begin{cases} 1 & \text{if } F_{kp}(f_i) = 0 \\ 0 & \text{otherwise} \end{cases} \]  \hspace{1cm} (14)

where \( F_{kp}(f_i) \) is the number of the \( i \)th frame SIFT keypoint. If the current frame is not a monochrome frame, processing is stopped. Otherwise, whether the current frame is the starting point of a fade out or the ending point of a fade in is determined. A section of fade in/out is detected based on consecutive monotonic increases/decreases in the number of the frame SIFT keypoint. The following formulas are used for the determination: Eq.(15) is for monotonic increases and Eq.(16) is for monotonic decreases.

\[ \text{IncF}_{kp}(f_{i-1}, f_i) = \begin{cases} 1 & \text{if } F_{kp}(f_{i-1}) < F_{kp}(f_i) \\ 0 & \text{otherwise} \end{cases} \] 

\[ \text{DecF}_{kp}(f_{i-1}, f_i) = \begin{cases} 1 & \text{if } F_{kp}(f_{i-1}) > F_{kp}(f_i) \\ 0 & \text{otherwise} \end{cases} \]  \hspace{1cm} (15) \hspace{1cm} (16)

These equations can be satisfied also when the luminance gradually decreases towards a black scene and then gradually increases from a black scene, and this causes erroneous detection. The similarity within the same shot always keeps large magnitudes, while drops to low scores within the different shot. Thus, if the similarity between the beginning frame of a fade out and the ending frame of a fade in is higher than a threshold, the detected section is considered a false detection. Otherwise, a fade transition is detected, and \( \text{STC} \) value is set to 2, as Eq.(17).

\[ \text{STC}(f_{i-1}, f_i) = 2 \]  \hspace{1cm} (17)

where \( i \) is ranged from the beginning frame of a fade out to the ending frame of a fade in.

![Graph](image.png)

**Figure 2.** A sequence with a fade out/in transitions

### 3.4. Wipe Transition Detection

A wipe is a shot transition that one scene or picture gradually enters across the view while another gradually leaves. During wipe, the appearing and disappearing shots coexist in different spatial regions of the intermediate video frames, and the region occupied by the former grows until it entirely replaces the latter [1]. Visually we see one part of the disappearing scene is gradually replaced by another fragment from appearing scene by a line (a curve or a polygon), and until the appearing entirely replaces the disappearing [2] (see Figure 3).

Reports about wipe transition detection are fewer comparatively cut, fade, and dissolve. Wipe transition detection is especially challenging due to the fact that there are more than 20 different types of wipe that are commonly used in video editing and there is no single rule that applies to all of them. Early reports about wipe are mostly based on the most popular wipe transitions (side-to-side, corner-to-corner, and center-out) [2]. Up to date there is no one a unified effective technique for all kinds of wipe transition detection. However, wipe transition is often used in video editing by video editors for various
applications. Therefore, a unified technique for wipe transition detection is needed. In this section, we propose a new scheme that can be used to detect all kinds of the wipe transitions.

One of the important properties of the change during a wipe is that one portion of the frame match to the starting frame, \( f_{wb} \), and the rest portion of the frame matches to the ending frame, \( f_{we} \). We can model the changing characteristics of a wipe transition as:

\[
\int_i(x,y) = k f_{wb}(x,y) + a b (k - 1) f_{we}(x,y), k \in (1,0)
\]  

(18)

The proposed method tries to detect the wipe transition based on this model and SIFT. The details of the wipe detection are explained in the following section.

First, the starting point of a wipe and the ending point of a wipe are needed to be determined. On a series of frames, where \( STC(f_{i-1}, f_i) = -1 \), the starting frame of this series of frames is regarded as \( f_{wb} \) and ending frame of this series of frames is regarded as \( f_{we} \).

When a shot changes to another shot according to Eq.(18), some SIFT keypoints of frame match to the keypoints extracted from \( f_{wb} \), denote by \( A_i \), and the rest match to the keypoints extracted from \( f_{we} \), denote by \( B_i \). The intersection of set \( A_i \) and \( B_i \) is Null, and the union of set \( A_i \) and \( B_i \) belongs to the keypoints database detected in the \( i \)th frame.

\[
A_i \cap B_i = \text{Null} \quad \text{and} \quad A_i \cup B_i \subset C
\]  

(19)

where \( C \) is the total keypoints detected from the \( i \)th image. During the wipe, the magnitude of \( A_i \) is reducing and \( B_i \) is increasing. For wipe transition, \( STC \) value is set to 3.

\[
STC(f_{i-1}, f_i) = 3
\]  

(20)

where \( i \) is ranged from \( f_{wb} \) to \( f_{we} \).

Figure 4 shows a sample that a middle frame in wipe matches to \( f_{wb} \) and \( f_{we} \) in which lines join the matching keypoints.

![Figure 3. A sample of wipe transition](image)

![Figure 4. A sample of wipe detection](image)

### 3.5. Dissolve Transition Detection

A dissolve in a video sequence is a shot transition with the first shot gradually disappearing while the second shot gradually appears [2]. Early reports about dissolve are mostly based on a single consecutive frames similarity. Due to the fact that during a gradual transition two consecutive frames may be the same or very similar to each other, the similarity between two consecutive frames have not obvious change, as a result, dissolve transition detection that based on a single consecutive frames similarity could not be effectively detected. However, the similarity scores between two frames with time distance \( L \) would have obvious and regularity change, so we are interested in the similarity
between frames that are a specific distance apart from each other. This proposed paper calculates the similarity between two frames also using NSKM, for it is robust against luminance changes, object motions and camera zoom in/out operations over the whole frame. Figure 5 displays clearly the NSKM values between two frames with time distance $L$. As depicted in Figure 5, variance curve in the dissolve region forms a clear parabolic shape and the variances in the middle frames over the transition have the locally lowest value. Thus, the dissolve transition is detected by the similarity score variance between two frames with time distance $L$.

Here, the features calculated with above method are not robust with respect to camera operations because the persistent camera motion will result in continuity signal curve similar to dissolve transition. According to [2] [12], dissolve-type shot transitions can be modeled as Eq. (21).

$$ f(x, y) = (1 - \alpha) f_{eb}(x, y) + \alpha f_{de}(x, y), \quad 0 \leq \alpha \leq 1 $$

(21)

Where $f_{eb}$ represents the starting frame of the dissolve transition, $f_{de}$ represents the ending frame, and $\alpha$ is a constant that varies between 0 and 1. The proposed method tries to use feature based on this model to eliminate false dissolve detections. When a shot changes to another shot according to Eq. (21), each pixel in a frame is either monotonically increasing or monotonically decreasing. Therefore, the total number of monotonically varying pixels within the dissolve transition is calculated, which is used to verify dissolve detection [12]. The detail algorithm is defined as followings.

First, it is determined whether each pixel in a frame is either monotonically increasing or monotonically decreasing.

$$ p_{mono}(f(x, y)) = \begin{cases} 1 & \text{if } f_{i+1}(x, y) < f_i(x, y) \\ 0 & \text{if } f_{i+1}(x, y) = f_i(x, y) \\ -1 & \text{otherwise} \end{cases} $$

(22)

where $f(x, y)$ indicates the pixel value at coordinates $(x, y)$ of the $i$th frame. When $p_{mono}(f(x, y))$ is equal to 1, denotes the pixel value is increasing, 0 means unchanging, and -1 represents decreasing.

Next, monotonically varying of each pixel in the dissolve transitions is calculated:

$$ f_{mono}(x, y) = \begin{cases} 1 & \text{if } |p_{mono}(x, y)| > T_{mono} \\ 0 & \text{otherwise} \end{cases} $$

(23)

$$ sp_{mono}(x, y) = \sum_{i=1}^{f_{de}} p_{mono}(f(x, y)) $$

(24)

$$ T_{mono} = (f_{de} - f_{eb}) \times 0.65 $$

(25)

If $f_{mono}(x, y)$ is equal to 1, denotes the pixel $(x, y)$ is monotonically varying.

Finally, the total number of monotonically varying pixels within the dissolve transitions, denote by $diss$, is calculated as followings.

$$ Diss = \sum_{(x, y) \in F} f_{mono}(x, y) $$

(26)

where, $F$ represents the pixels of the overall frame. If $Diss$ is lower than a threshold $T_{diss}$, the detected section is considered a false detection, otherwise a dissolve transition is detected, and STC value is set to 4.

$$ STC(f_{i-1}, f_i) = 4 $$

(27)

where $i$ is ranged from $f_{de}$ to $f_{eb}$.

4. Experimental Results and Discussions

In this section, we will carry out several experiments on the platform of TRECVID test data [25] and film videos. Totally, there are 40 clips, 150 minutes, which includes cut, dissolve, fade out/in, and wipe gradual transition. All of the experiments are conducted with Matlab.

The performance of a shot transition detection algorithm is usually measured with terms of recall and precision. The recall and precision are defined as following:
Figure 5. A sequence with dissolve transitions

\[
\text{recall} = \frac{N_c}{N_c + N_m} \times 100\% \\
\text{precision} = \frac{N_c}{N_c + N_f} \times 100\% 
\]

(28)

(29)

where, \( N_c \) is the number of correct detections, \( N_m \) is the number of missed detections, \( N_f \) is the number of false detections. A good shot transition detector should have both high precision and high recall.

Figure 7. Example of missed detection

Figure 8. Example of false detection

Firstly, we should decide some parameter in the experiment. For SVM, we use the software Libsvm provided by the National Science Council of Taiwan to do SVM classification [27]. We have tested the
performance on different kernels and find that the "RBF" kernel outperforms others. There are two parameters while using RBF kernels: the penalty parameter $C$ and $r$. It is not known beforehand which $C$ and $r$ are the best for one problem. We adopt the cross-validation method to obtain the $C$ and $r$ in this paper.

J.Bescós [28] has proposed a unified model for shot boundary detection, which is centered on mapping the space of inter-frame distances onto a new space of decision better suited to achieving a sequence-independent thresholding. Yu algorithm [2] presented a hierarchical multiresolution video shot boundary detection scheme based on wavelet. J.Bescós and Yu algorithms are the more successful approaches in lots of algorithms. We thus used them for our comparison study. Table 1 and 2 lists the performance of the proposed algorithm compared with them.

Cut transition is instantaneous transitions from one shot to the subsequent shot. In our system, SIFT algorithm is chosen as the feature to calculate the similarity between adjacent frames to detect cut transition. For it is robust to luminance changes, object motion, and camera operations, the proposed algorithm produces better performance compared with J.Bescós and Yu algorithms. The recall rate of our method is 95.46%. The recall rate of J.Bescós method is 91.16%, and the recall rate of Yu method is 79.70%. The precision rate of our method is 92.24%. The precision rate of J.Bescós method is 92.18%, and the precision rate of Yu method is 79.29%.

Using different features and algorithms for different kinds of gradual transitions to capture the different characteristics help us to get a better solution for gradual transition detections problem. The recall rate of our method is 70.61%. The recall rate of J.Bescós method is 63.69%, and the recall rate of Yu method is 53.34%. The precision rate of our method is 67.38%. The precision rate of J.Bescós method is 62.04%, and the precision rate of Yu method is 59.78%. From the experimental results, we can find that the proposed algorithm produces better performance.

We have investigated the incidence of missed detections and false detections in cut transition detection system. Many missed cut transition detections occurred because the successive abrupt shot changes with similar backgrounds, as shown in Figure 6. False cut detections were mostly caused by fast movements of the camera or object, as shown in Figure 7.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall(%)</th>
<th>Precision(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>95.46</td>
<td>92.24</td>
</tr>
<tr>
<td>J.Bescós method</td>
<td>91.16</td>
<td>92.18</td>
</tr>
<tr>
<td>Yu method</td>
<td>79.70</td>
<td>79.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall(%)</th>
<th>Precision(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>70.61</td>
<td>67.38</td>
</tr>
<tr>
<td>J.Bescós method</td>
<td>63.69</td>
<td>62.04</td>
</tr>
<tr>
<td>Yu method</td>
<td>53.34</td>
<td>59.78</td>
</tr>
</tbody>
</table>

5. Conclusions

We have presented our complete framework of a video shot transition detection methodology. Unlike previous approaches which calculates all frame features to try to detect shot transitions, our method skips the processing of frames that are clearly not shot boundaries, and calculates various features only for the parts of the video that are likely to contain shot boundaries. To get a better solution for shot boundary detection problem, we propose using SVM to compute the keypoint matching scheme. In addition, different features and algorithms are used to capture the different characteristics for different kinds of shot transitions. The testing result of the experiment shows that the method has good accuracy for shot boundary detection.

9. Acknowledgment
10. References


A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT
Jun Li, Youdong Ding, Yunyu Shi, Wei Li


