Shot Boundary Detection at TRECVID 2007

Yoshihiko Kawai † Hideki Sumiyoshi † Nobuyuki Yagi †

†Science and Technical Research Laboratories, NHK
1–10–11 Kinuta, Setagaya–ku, Tokyo, Japan

Abstract

Shot boundary detection is one of the most fundamental processes in video analysis, and it requires high detection accuracy and high-speed processing. This paper proposes a method of shot boundary detection based on multiple features. The proposed method enables precise, high-speed detection by omitting the processing of frames that are clearly not shot boundaries, and by analyzing various features only for the parts of the video that are likely to contain shot boundaries. An experiment using TRECVID 2007 test data resulted in a recall rate of 90.5% and a precision rate of 94.4%. About 425 minutes of test data was processed in 3 minutes 28 seconds (excluding the MPEG1 decoding time), 1/123 of the real time.

1 Introduction

A shot is a basic unit of video, and dividing video into shots is the first step in video analysis. It is necessary to detect shot boundaries, which are the connection points between one shot and another, to divide video into shots. Shot boundaries can be broadly classified into two types: abrupt transitions and gradual transitions. Abrupt transitions are instantaneous transitions from one shot to the subsequent shot. This is also called a cut. Gradual transitions include dissolves and fades, in which one frame is transformed into another as the proportion of each frame is gradually changed; wipes, in which the boundary between shots moves spatially; and special effects, in which shapes and positions are transformed three-dimensionally during the transition.

Shot boundaries can be detected on the basis of a characteristic that the similarity between adjacent frames within the same shot is high, but is low if a shot boundary exists between the frames. Various methods have been proposed to measure the degree of similarity between frames, such as the differences of luminance histogram [1, 2, 3], edge variation [4], and mutual information between pixels [5]. Distributions of pixel values [6] and DCT coefficients [7] have also been proposed for gradual transitions. In recent years, methods that combine some of these features in order to increase the detection accuracy have been proposed [8, 9, 10, 11]. These methods calculate multiple features for each frame, and then classify them as either a "shot boundary" or "not shot boundary", using classification algorithm such as fuzzy c-means or support vector machine. One problem with conventional methods that make use of multiple features is the high computational cost of calculating all the features for each frame. The computing cost is particularly high in cases where complex features are used, or where a large number of features are used. Since the shot boundaries included in TV programs typically amount to less than 1% of total frames, it is inefficient to apply boundary detection processing indiscriminately to all the frames.

This paper proposes a high-speed, high-precision shot boundary detection method. 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. This enables high detection accuracy with low computational cost.

2 Shot boundary detection

An overview of our method is shown in Figure 1. First, frame images are extracted from the input video in the decoder. Next, the extracted frame images are analyzed to determine the shot boundaries. Processing for abrupt transitions and gradual transitions is done in parallel to reduce missed detections. A gradual transition is analyzed in the order of fades, dissolves, and long dissolves. Whenever a gradual transition is detected, the remaining processing for other types of gradual transition is not performed. To avoid missing long dissolves, these are handled separately from normal dissolves. Furthermore, fea-

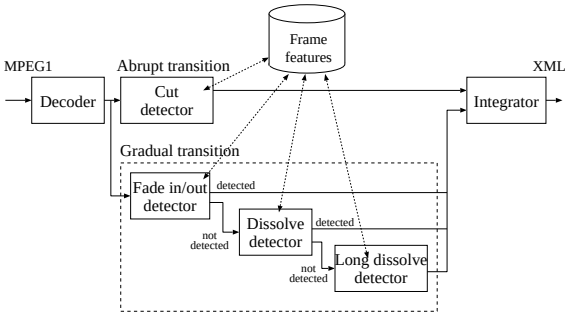


Figure 1: Overview of proposed method

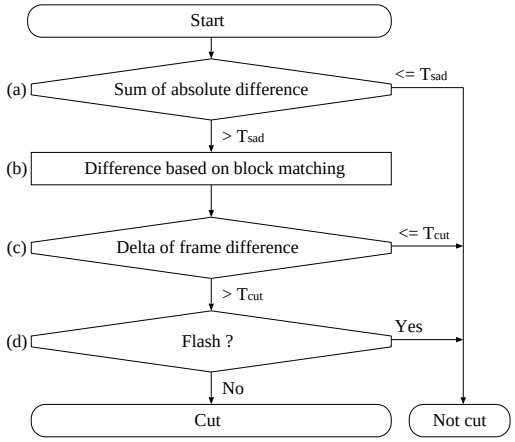


Figure 2: Diagram of cut detection

tures of frames are shared by all the detection processing and the system ensures that the same type of feature is not calculated redundantly. The final detection results are produced at the integrator, by integrating the overlapping transitions.

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

3 Cut detector

Cut transitions are detected based on the differences between adjacent frames. Figure 2 shows a diagram for detecting cuts. First, frames that show no significant change are determined by using the sum of the absolute differences between pixel R, G, and B values, d_{sad} (Figure 2(a)). The calculation formula of d_{sad} is shown in Eq. (1).

$$d_{sad}(f_{i-1}, f_i) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} |f_{i-1}(\boldsymbol{r}) - f_i(\boldsymbol{r})| \qquad (1)$$

$f_i(\boldsymbol{r})$ indicates the pixel value at coordinates $\boldsymbol{r}$ of the ith frame. $\mathbf{F}$ represents the pixels of the overall frame, and $|\mathbf{F}|$ represents the total number of pixels

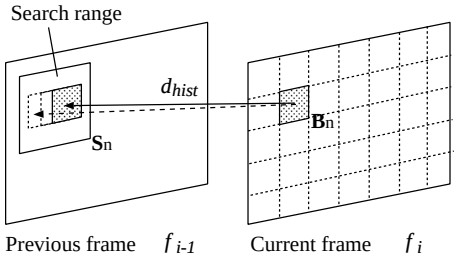


Figure 3: Block matching

in a frame. The formula described here has been simplified to avoid undue complexity, but in practice, the sum of differences for each of the R, G, and B values is required. If d_{sad} is less than a threshold T_{sad}, it is judged that the frame cannot be a shot boundary, and the remaining processing is skipped.

In contrast, if d_{sad} is above the threshold, a more precise frame difference is calculated (Figure 2(b)). Here, the frame difference based on block matching, d_{bm} is used, which is robust against camera operations (e.g. zoom in/out, pan and tilt) or object motions. A detailed explanation of d_{bm} is given in a later section.

Next, it is determined whether a cut transition exists between frames based on the frame difference d_{bm} (Figure 2(c)). Our method detects a cut when the increase in frame difference d_{bm} exceeds a certain threshold, because d_{bm} can vary significantly due to strong movements of the camera or the object, resulting in false detections. Equation (2) shows the condition for this decision. If Eq. (2) is true, a cut is detected between frame $i-1$ and frame i. Otherwise, the processing is terminated here.

$$d_{bm}(f_{i-1}, f_i) - d_{bm}(f_{i-2}, f_{i-1}) > T_{cut} \qquad (2)$$

Finally, whether a detected transition is a false detection caused by the flash illumination of a still camera is determined (Figure 2(d)).

The processes shown in Figure 2(b) and Figure 2(d) are explained in detail in **3.1** and **3.2**, respectively.

3.1 Difference based on block matching

Figure 3 shows an outline of block matching. In the block matching method, the current frame is divided into non-overlapping block regions, and then the previous frame is searched to find the position of minimum inter-block cost for each block of the current frame. The frame difference is calculated by adding up the total number of blocks for which

the cost is higher than a threshold. The calculation formula of d_{bm} is shown in Eq. (3).

$$d_{bm}(f_{i-1}, f_i) = \frac{1}{N} \sum_{n=1}^{N} 1 \quad if\ \lambda_n(f_{i-1}, f_i) > T_\lambda \tag{3}$$

N represents the total number of blocks, and λ_n represents the minimum value of the inter-block cost for the nth block. T_λ is the threshold. The sum of squared differences or the sum of absolute differences is generally used as the inter-block cost to achieve motion vectors for video encoding. In contrast, our method adopts the sum of absolute histogram differences which is more robust to camera motions. λ_n is calculated using Eq. (4).

$$\lambda_n(f_{i-1}, f_i) = \min_{\boldsymbol{v} \in \mathbf{S}_n} \left\{ d_{hist}\left(f_{i-1}(\boldsymbol{r}+\boldsymbol{v}), f_i(\boldsymbol{r})\right), \boldsymbol{r} \in \mathbf{B}_n \right\} \tag{4}$$

Here, $\mathbf{S}_n$ indicates the search range, and $\mathbf{B}_n$ indicates the pixels contained within the nth block region. d_{hist} represents the sum of absolute histogram differences. d_{hist} is calculated by creating a frequency histogram of pixel values for each block region and then computing the sum of the absolute differences for each bin. Equation (5) shows the formula for calculating d_{hist}.

$$d_{hist}(f_{i-1}, f_i) = \frac{1}{N_c} \sum_{c=1}^{N_c} |H_{i-1}(c) - H_i(c)| \tag{5}$$

N_c represents the total number of histogram bins, and $H_i(c)$ represents the number of pixels in frame i contained in the cth bin.

The most computationally expensive processing in block matching is searching for block position $\boldsymbol{v}$, as shown in Eq. (4). The proposed method attempts to speed up block matching by reducing the amount of calculation in this search processing. Various block search algorithms have been proposed to speed up block matching, and our method uses the dual cross search algorithm (DCS) [12], which is one of the fastest. Additionally, the proposed method tries to further boost its speed by cutting off the search processing using a threshold T_λ. In a preliminary experiment, we found that the calculation amount for searching $\boldsymbol{v}$ could be reduced by a factor of 1/600 compared to that with a simple full search algorithm.

3.2 Flash detection

The sudden change in luminance resulting from flash illumination can cause erroneous detection. The proposed method therefore needs to ensure that detected cut transitions are not false detections due to flash illumination from a still camera.

When a flash is used, the luminance for the first several frames increases, after which the luminance returns to the previous level. Thus, the proposed method first produces the image f'_i, each pixel of which has the minimum value of corresponding pixels in the following N_f frames. The calculation is shown in Eq. (6).

$$f'_i(\boldsymbol{r}) = min\left(f_i(\boldsymbol{r}), f_{i+1}(\boldsymbol{r}), .., f_{i+N_f}(\boldsymbol{r})\right) \tag{6}$$

If f_i is a frame whose luminance has increased due to flash illumination, frames f_{i-1} and f'_i will have similar image features. Thus, if the frame difference $d(f_{i-1}, f'_i)$ is lower than a threshold T_{cut}, the potential shot boundary is judged to be a false detection. Equation (7) shows the formula to calculate the frame difference d.

$$d(f_{i-1}, f_i) = \begin{cases} 0 & if\ d_{sad}(f_{i-1}, f_i) < T_{sad} \\ d_{bm}(f_{i-1}, f_i) & otherwise \end{cases} \tag{7}$$

The calculation amount is reduced by omitting computationally intensive block matching when d_{sad} is less than a threshold, in the same way as for cut determination.

4 Fade in/out detector

A fade in is a transition where the luminance of a frame gradually increases from a black frame. Conversely, a fade out is a transition where the luminance gradually decreases towards a black frame. The proposed method tries to detect fades by focusing on this black frame. A diagram of the method is shown in Figure 4.

First, it is determined whether the current frame is a black frame (Figure 4(a)). The mean luminance, as shown in Eq. (8), and the number of pixels whose luminance is less than a threshold T_{black}, as shown in Eq. (9), are used for the determination.

$$\overline{f_i} = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} f_i(\boldsymbol{r}) \tag{8}$$

$$black(f_i) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} 1 \quad if\ f_i(\boldsymbol{r}) < T_{black} \tag{9}$$

If the current frame is not a black frame, processing is stopped. Otherwise, whether the current frame is the end point of a fade out or the starting point of a fade in is determined.

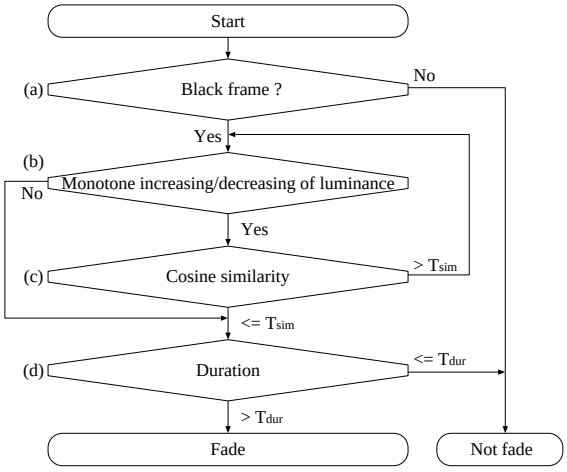


Figure 4: Diagram of fade in/out detection

A section of fade in/out is detected based on consecutive monotonic increases/decreases in luminance (Figure 4(b)). The following formulas are used for the determination: Eq. (10) is for monotonic increases and Eq. (11)) is for monotonic decreases. If the result of one of these formulas exceeds a threshold T_{fade}, a monotonic increase/decrease is detected.

$$inc(f_{i-1}, f_i) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} 1 \quad if \ \ f_i(\boldsymbol{r}) > f_{i-1}(\boldsymbol{r}) \tag{10}$$

$$dec(f_{i-1}, f_i) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} 1 \quad if \ \ f_i(\boldsymbol{r}) < f_{i-1}(\boldsymbol{r}) \tag{11}$$

These equations can be satisfied also when low luminance objects appear in the frame, and this causes erroneous detection. Thus, if the similarity between adjacent frames within the fade section is less than a threshold T_{sim}, a false detection is judged to have occurred (Figure 4(c)). The similarity is calculated using Eq. (12).

$$sim(f_{i-1}, f_i) = \frac{\sum_{\boldsymbol{r} \in \mathbf{F}} f_{i-1}(\boldsymbol{r}) \cdot f_i(\boldsymbol{r})}{\sqrt{\sum_{\boldsymbol{r} \in \mathbf{F}} f_{i-1}^2(\boldsymbol{r}) \sum_{\boldsymbol{r} \in \mathbf{F}} f_i^2(\boldsymbol{r})}} \tag{12}$$

This similarity is calculated based on the cosine of frame images, which is insensitive to luminance changes over the whole frame.

Finally, if the duration of the fade section is greater than a threshold T_{dur}, a fade transition is detected (Figure 4(d)).

5 Dissolve detector

Dissolve-type shot transitions can be modeled as Eq. (13).

$$f_i(\boldsymbol{r}) = (1-\alpha) \cdot f_b(\boldsymbol{r}) + \alpha \cdot f_e(\boldsymbol{r}), \quad 0 \le \alpha \le 1 \tag{13}$$

Here, f_b represents the start frame of the dissolve, f_e represents the end frame, and α is a constant that varies between 0 and 1. The proposed method tries to detect dissolve sections using two kinds of feature based on this model.

The first feature is based on monotonic luminance changes. When a shot changes to another shot according to Eq. (13), each pixel in a frame is either monotonically increasing or monotonically decreasing. Therefore, the total number of monotonically varying pixels is calculated by comparison with previous or subsequent frames, and this value is used as the first feature. The first feature is defined as Eq. (14) and Eq. (15).

$$diss(f_{i-1}, f_i, f_{i+1}) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} mono(f_{i-1}(\boldsymbol{r}), f_i(\boldsymbol{r}), f_{i+1}(\boldsymbol{r})) \tag{14}$$

$$mono(v1, v2, v3) = 1 \quad if \quad (v1 > v2 > v3) \ \ or \ \ (v1 < v2 < v3) \tag{15}$$

Another feature is the difference relative to an ideal dissolve. The relationship given in Eq. (16) must be satisfied during an ideal dissolve section.

$$f_i(\boldsymbol{r}) = \frac{f_{i-1}(\boldsymbol{r}) + f_{i+1}(\boldsymbol{r})}{2} \tag{16}$$

The difference shown in Eq. (17) is calculated and utilized as another feature.

$$err(f_{i-1}, f_i, f_{i+1}) = \frac{1}{|\mathbf{F}|} \sum_{\boldsymbol{r} \in \mathbf{F}} \left| \frac{f_{i-1}(\boldsymbol{r}) + f_{i+1}(\boldsymbol{r})}{2} - f_i(\boldsymbol{r}) \right| \tag{17}$$

A diagram of dissolve detection is shown in Figure 5. First, the sum of absolute differences between the current frame i and the N_dth frame prior to the current frame, $d_{sad}(f_{i-N_d}, f_i)$ (Figure 5(a)). If d_{sad} is less than a threshold T_{sad}, it is judged that there are no shot transitions within the section, and the process is terminated. Otherwise, the dissolve section will be searched based on the two kinds of feature described above.

Here, the features calculated with Eq. (16) and Eq. (17) are not robust with respect to camera operations and object motions because these features consider only three continuous frames. Thus,

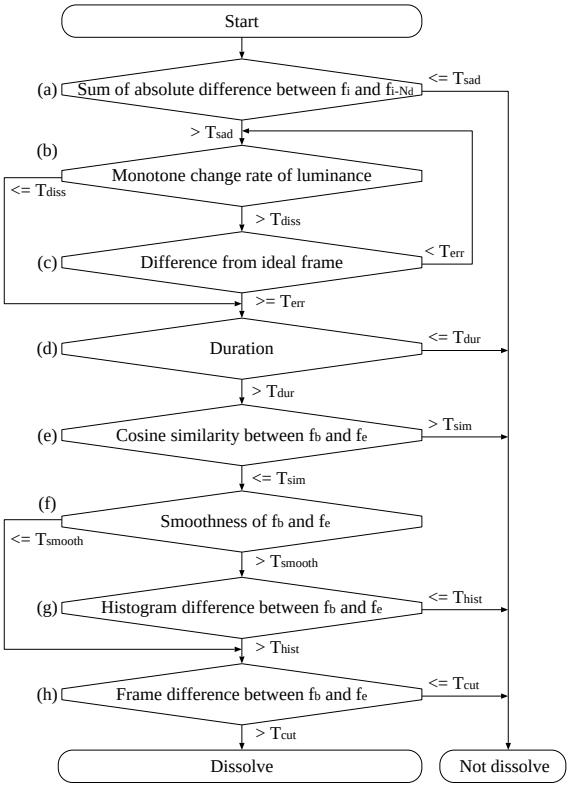


Figure 5: Diagram of dissolve detection

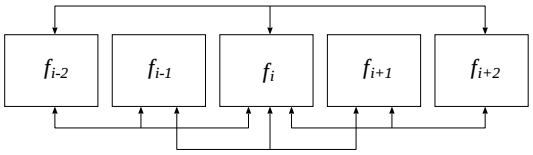


Figure 6: Dissolve detection based on monotonic luminance change

our method calculates features based on four points, making use of five continuous frames, as shown in Figure 6. The calculation formula is shown in Eq. (18) and Eq. (19). Continuous frames that satisfy both of the equations are judged to be dissolve sections (Figure 5(b) and (c)).

$$diss(f_{i-2}, f_{i-1}, f_i) + diss(f_{i-1}, f_i, f_{i+1}) + diss(f_i, f_{i+1}, f_{i+2}) + diss(f_{i-2}, f_i, f_{i+2}) > T_{diss} \quad (18)$$

$$err(f_{i-2}, f_{i-1}, f_i) + err(f_{i-1}, f_i, f_{i+1}) + err(f_i, f_{i+1}, f_{i+2}) + err(f_{i-2}, f_i, f_{i+2}) < T_{err} \quad (19)$$

Figure 7: Example of frame which causes false detection

The next step is to verify the detected dissolve sections. First, whether the duration of the section is sufficiently long is checked (Figure 5(d)). If the duration is less than a threshold T_{dur}, detection processing is terminated.

Next, whether luminance variation has occurred is checked. A smooth variation of luminance over time, due for example to changes in camera aperture, can satisfy Eq. (18) and Eq. (19), and cause false detections. The similarity between the beginning frame f_b, and ending frame f_e of the dissolve section is therefore calculated, and if the similarity is higher than a threshold, the detected section is considered a false detection (Figure 5(e)). The similarity is calculated using Eq. (12), because this equation is robust against luminance variation.

In addition, when images that show smooth spatial variation in luminance, like that in Figure 7, move in a particular direction, Eq. (18) and Eq. (19) can be satisfied, resulting in false detection. Equation (20) is therefore used to determine whether f_b and f_e are images that have smooth backgrounds (Figure 5(f)).

$$smooth(f_i) = \frac{1}{4|\mathbf{F}|} \sum_{(x,y)\in\mathbf{F}} \{s_1(f_i) + s_2(f_i) + s_3(f_i) + s_4(f_i)\} \quad (20)$$

$$s_1(f_i) = mono(f_i(x-1,y), f_i(x,y), f_i(x+1,y)) \quad (21)$$

$$s_2(f_i) = mono(f_i(x,y-1), f_i(x,y), f_i(x,y+1)) \quad (22)$$

$$s_3(f_i) = mono(f_i(x-1,y-1), f_i(x,y), f_i(x+1,y+1)) \quad (23)$$

$$s_4(f_i) = mono(f_i(x-1,y+1), f_i(x,y), f_i(x+1,y-1)) \quad (24)$$

If $smooth(f_b)$ or $smooth(f_e)$ are above the threshold T_{smooth}, the difference between the histograms of f_b and f_e is calculated using Eq. (5), and if this value is lower than a threshold T_{hist}, the detected section is judged to be a false detection (Figure 5(g)).

Finally, the difference between frames f_b and f_e is calculated using Eq. (7), and if the variation of frame difference is greater than a threshold T_{cut}, a dissolve transition is detected (Figure 5(h)).

Table 2: Evaluation result of shot boundary detection

Run	Overall			Cut			Gradual			Frame based		
	R	P	F	R	P	F	R	P	F	R	P	F
4	0.937	0.859	0.896	**0.961**	**0.939**	**0.968**	0.680	0.369	0.478	0.662	0.872	0.753
1	0.921	0.913	0.917	0.946	0.951	0.948	0.650	0.554	0.598	0.652	0.879	0.749
2	**0.905**	**0.944**	**0.924**	0.933	0.965	0.949	0.607	0.691	0.646	0.683	0.899	0.776
3	0.888	0.960	0.923	0.916	0.975	0.945	**0.578**	**0.768**	**0.660**	0.706	0.887	0.786
5	0.867	0.966	0.914	0.896	0.980	0.936	0.544	0.767	0.637	**0.709**	**0.910**	**0.797**

Table 1: Threshold settings

Threshold	Run				
	4	1	2	3	5
T_{sad}	15				
T_{cut}	30	35	40	45	50
T_{λ}	30				
N_f	3				
T_{black}	13				
T_{fade}	55				
T_{dur}	5				
T_{sim}	90				
N_d	10				
T_{diss}	30	35	40	45	50
T_{err}	70				
T_{smooth}	40				
T_{hist}	40				
N_l	2				

6 Long dissolve detector

The pixel value may not change between adjacent frames when the dissolve duration is very long. Such long dissolves will not be detected by the method described in the previous section, because Eq. (18) and Eq. (19) will not be satisfied. To ensure long dissolves are detected, the frame interval in Eq. (18) and Eq. (19) is expanded by a factor of N_l. Equation (25) and Eq. (26) show the formulas.

$$diss(f_{i-2N_l}, f_{i-N_l}, f_i) + diss(f_{i-N_l}, f_i, f_{i+N_l}) + diss(f_i, f_{i+N_l}, f_{i+2N_l}) + diss(f_{i-2N_l}, f_i, f_{i+2N_l}) \quad (25)$$

$$err(f_{i-2N_l}, f_{i-N_l}, f_i) + err(f_{i-N_l}, f_i, f_{i+N_l}) + err(f_i, f_{i+N_l}, f_{i+2N_l}) + err(f_{i-2N_l}, f_i, f_{i+2N_l}) \quad (26)$$

All other processing for long dissolves is done in the same way as for normal dissolves.

7 Experiment

We carried out an experiment with TRECVID 2007 test data using multiple combinations of thresholds. The threshold combinations were determined based on trials for several TV program genres (e.g. drama, baseball, soccer and documentary) broadcast in Japan, and these thresholds were used in Run 1. We varied these thresholds, and carried out a total of five runs. The thresholds used in the experiment are shown in Table 1.

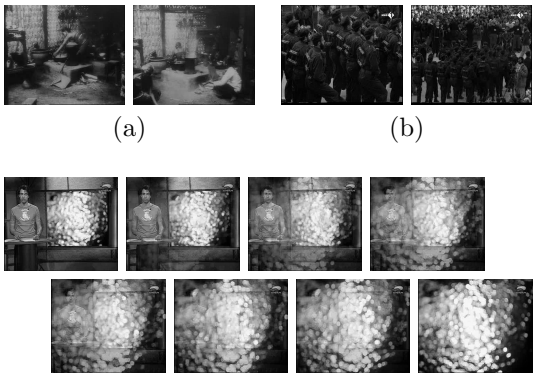
The number of frame divisions for block matching was $10 \times 10 = 100$ blocks, and the number of histogram bins used for calculating the inter-block cost was 16 for each R, G and B color. For detection processing, we used frame images of 176×144 pixels, obtained by 1/2 down-sampling.

7.1 Evaluation result

The evaluation results for the five runs are shown in Table 2. In the table, R stands for recall rate, P for precision rate, and F for F-measure. In the overall results, Run 2 achieved the highest F-measure. The recall rate of 90.5% and precision rate of 94.4% represent extremely high detection accuracy. For cuts, the best result was from Run 4, while for gradual transitions the best result was from Run 3. For frame-based comparative results in gradual transitions, Run 5 was the best. Run 1 showed average accuracy for each performance measure.

We also investigated the incidence of missed detections and false detections. For cuts, the recall rate was very low (about 50%) for monochrome film video included in the test data. Many missed detections occurred because the changes between frames at the shot boundary were small. Figure 8(a) shows an example. Missed detections also occurred at transitions between similar shots, and dark shots such as night scenes, as shown in Figure 8(b). We have to consider dynamically adjusting the number of bins for the histogram difference and thresholds according to the input video to avoid missed detections. False detections were caused by the appearance of large graphic overlays (Figure 9(a)) or objects passing in front of the camera (Figure 9(b)).

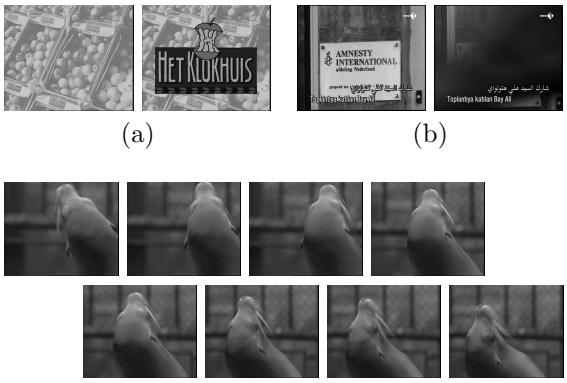
For gradual transitions, missed detections were mostly caused by movements of the camera or objects during the transition. The dissolve model de-



(a) (b)

(c)

Figure 8: Example of missed detections



(a) (b)

(c)

Figure 9: Example of false detections

scribed by Eq. (13) assumes that the frames within the transition section are still. Therefore, if there is camera movement or object motion, the conditions for dissolve detection are not satisfied. An example of such a missed detection is shown in Figure 8(c). Dissolve detection failed due to the variation in the luminance caused by the change in the water surface. Our system also failed to detect wipes and special effects, which are not targeted by the proposed method. Figure 9(c) shows an example of false detections caused by object motion. We have to consider a method that can distinguish dissolves from object motion by using, for example, edge or frequency features.

7.2 Processing time

The computer used for the experiment had an Intel Core 2 Duo E6600 2.40 GHz processor and 2 GB of RAM. The processing times for about 425 minutes of test data are shown in Table 3. The "Segmentation" column represents the time required for shot boundary detection, while the "Decode" column represents the processing time needed for decoding MPEG1 video and extracting frames. The "Total" column shows the total processing time, which is equal to the sum of the segmentation time and the decode time. The " Avg. " row shows the mean processing time for the five runs, and the ratio to real time.

Table 3: Processing time (mm:ss)

Run	Segmentation	Decode	Total
4	03:35	24:42	28:17
1	03:29	24:36	28:05
2	03:26	24:37	28:03
3	03:26	24:39	28:05
5	03:24	24:39	28:03
Avg.	03:28 (1/123)	24:39 (1/17)	28:07 (1/15)

The average total processing time was about 28 minutes, or 1/15 of real time. Our system could greatly reduce the computational cost by skipping the processing of frames that are clearly not boundary shots. Additionally, the segmentation time which does not include the time for MPEG1 decoding amounted to a mere 3 minutes 28 seconds, only about 1/123 of real time. Further speed improvements should be attainable by optimizing the video decoding processing.

8 Conclusion

We proposed a method for detecting shot boundaries that utilizes multiple features. The proposed method enabled accurate and high-speed shot boundary detection by omitting the processing of frames that are clearly not shot boundaries and calculating detailed features only for parts of the video that are likely to contain transitions. In the experiments with TRECVID 2007 test data, a recall rate of 90.5% and a precision rate of 94.4% were achieved. In addition, our system could process about 425 minutes of test data in 3 minutes 28 seconds, only 1/123 of the real time.

Future work will include methods to determine thresholds for each detection process. Some kinds of machine learning algorithms will be considered to determine optimal combination of thresholds. We also have to consider a method for use with video data that has small luminance changes, such as monochrome video, and a detection method for dissolves which include camera or object motions. Furthermore, we will extend the method to detect other kinds of transitions, such as wipes and special effects.

References

[1] H.Zhang and S.S.A.Kankanhalli, "Automatic partitioning of full-motion video," ACM Multimedia Systems, vol.1, no.1, pp.10–28 (1993).

[2] J.S.Boreczky and L.A.Rowe, "Comparison of video shot boundary detection techniques," in Proc. SPIE, vol.2664, pp.170–179 (1996).

[3] B.T.Truong, C.Dorai and S.Venkatesh, "New enhancements to cut, fade, and dissolve detection processes in video segmentation," in Proc. ACM Multimedia, pp.219–227 (2000).

[4] R.Zabih, J.Miller and K.Mai, "A feature-based algorithm for detecting and classifying production effects," Multimedia Systems, vol.7, no.2, pp.119–128 (1999).

[5] Z.Cernekova, I.Pitas and C.Nikou, "Information Theory-Based Shot Cut/Fade Detection and Video Summarization," IEEE Trans. Circuits and Systems for Video Tech., vol.16, no.1, pp.82–91 (2006).

[6] J.Nam and A.H.Tewfik, "Detection of Gradual Transitions in Video Sequences Using B-Spline Interpolation," IEEE Trans. Multimedia, vol.7, no.4, pp.667–679 (2005).

[7] R.A.Joyce and B.Liu, "Temporal Segmentation of Video Using Frame and Histogram Space," IEEE Trans. Multimedia, vol.8, no.1, pp.130–140 (2006).

[8] X.Gao and X.Tang, "Unsupervised Video-Shot Segmentation and Model-Free Anchorperson Detection for News Video Story Parsing," IEEE Trans. Circuits and Systems for Video Technology, vol.12, no.9, pp.765-776 (2002).

[9] C-W.Ngo, "A robust dissolve detector by support vector machine," in Proc. ACM Int. Conf. Multimedia, pp.283–286 (2003).

[10] H.Feng, W.Fang, S.Liu and Y.Fang, "A New General Framework for Shot Boundary Detection Based on SVM," in Proc. IEEE ICNN&B, vol.2, pp.1112–1117 (2005).

[11] K.Matsumoto, M.Naito, K.Hoashi and F.Sugaya, "SVM-Based Shot Boundary Detection with a Novel Feature," In Proc. IEEE Int. Conf. Multimedia and Expo, pp.1837–1840 (2006).

[12] X.-Q.Banh and Y.-P.Tan, "Adaptive Dual-Cross Search Algorithm for Block-Matching Motion Estimation," IEEE Trans. Consumer Electronics, vol.50, no.2, pp.766–775 (2004).