# A FAST, COMPREHENSIVE SHOT BOUNDRAY DETERMINATION SYSTEM

*Zhu Liu, David Gibbon, Eric Zavesky*[*]*, Behzad Shahraray, and Patrick Haffner*

AT&T Labs Research
{zliu, dcg, behzad, haffner}@research.att.com

[*]Columbia University, Electrical Engineering
emz@ee.columbia.edu

## ABSTRACT

The proposed shot boundary determination (SBD) algorithm contains a set of finite state machine (FSM) based detectors for pure cut, fast dissolve, fade in, fade out, dissolve, and wipe. Support vector machines (SVM) are applied to the cut and dissolve detectors to further boost performance. Our SBD system was highly effective when evaluated in TRECVID 2006 (TREC Video Retrieval Evaluation) and its performance was ranked highest overall.

## 1. INTRODUCTION

Shot boundary determination has been widely studied for the last decade. Some of the early work can be found in [1-4]. TRECVID [5] further stimulates the interest and effort in automatic segmentation, indexing, and content-based retrieval of digital video in a broad research community. New systems and algorithms have been constantly reported from all TRECVID participants over the years, e.g., IBM, Tsinghua University, Columbia University, CMU, KDDI, etc.. Researchers at AT&T started to tackle multimedia content processing and indexing in the early 1990's, and Shahraray reported a scene change detection algorithm in 1995 [3]. With the limited computation power (90M CPU) and system memory (8M) available at that time, as well as the constraints of real time and low latency, the original algorithm was designed to be effective and highly efficient. The adopted visual features were intensity histogram and image matching with 1 dimensional motion compensation by projection. A single finite sate machine (FSM) was designed to detect all types of scene changes and report camera motions, including panning and tilting. An improved version of this algorithm is adopted in the MIRACLE system, a video search engine, at AT&T [8].

Thanks to current computation power, there is a lot of room to extend the existing algorithm. Three major improvements are: 1) Two-dimension motion compensation, 2) utilizing color information in addition to intensity values, 3) instead of using a single FSM, multiple FSM-based detectors are adopted to track different types of shot boundaries, e.g., cut, fade in/out, dissolve, wipe, etc.. The new architecture is more flexible and modularized: each detector is independently designed and adjusted, and additional detectors can be easily plugged in to capture any new types of shot boundaries.

In this paper, we report the AT&T SBD system evaluated in TRECVID 2006. The paper is organized as follows. Section 2 gives an overview of the SBD system. Section 3 describes the adopted visual features and Section 4 illustrates the six shot boundary detectors. Result fusion is briefly addressed in Section 5. Evaluation results are presented and discussed in Section 6, and finally we draw our conclusions in Section 7.

## 2. OVERVIEW

There are three main components in our SBD system: visual feature extraction, shot boundary detectors, and result fusion. Fig. 1 shows the high level diagram of the SBD system. The top level of the algorithm runs in a loop, and every loop processes one video frame. Each new frame and the associated visual features are saved in circular buffers. The loop continues until all frames in the MPEG file are processed.

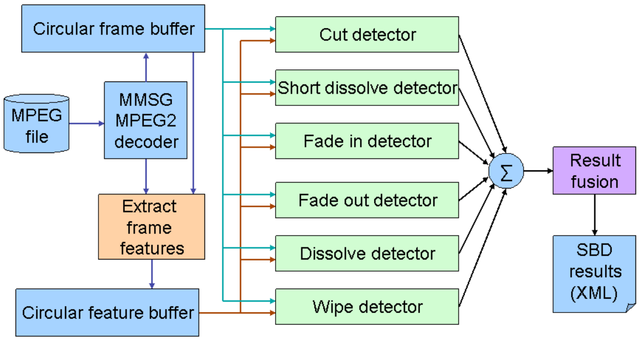


Fig. 1. Overview of the SBD system

Given the wide varieties of shot transitions, it is difficult to handle all of them using a single detector. Our system adopts a "divide and conquer" strategy. We devised six independent detectors, targeting for six dominant types of shot boundaries in the SBD task. They are cut, fade in, fade out, fast dissolve (less than 5 frames), dissolve, and wipe. Essentially, each detector is a finite state machine (FSM), which may have different number of states. Finally, the results of all detectors are fused and the overall SBD result is generated in the necessary format.

## 3. FEATURE EXTRACTION

For each frame, we extract a set of visual features, which can be classified into two types: intra-frame and inter-frame visual features. The intra-frame features are extracted from a single, specific frame, and they include color histogram, edge, and related statistical features. The inter-frame features rely on the current frame and one previous frame, and they capture the motion compensated intensity matching errors and histogram changes.

Fig. 2 illustrates how these visual features are computed. The resolution of the TRECVID evaluation sequences is 352x240 pixels. The visual features are extracted from a central portion of the picture, which we called the region of interest (ROI). The ROI is marked by a dashed rectangle in Fig. 2, overlaid on the original image. The choice of the ROI size is based on two considerations: 1) The ROI covers the majority of the image and effectively eliminates the letterbox for wide screen content. 2) The ROI avoids the border effect in the following feature extraction steps.

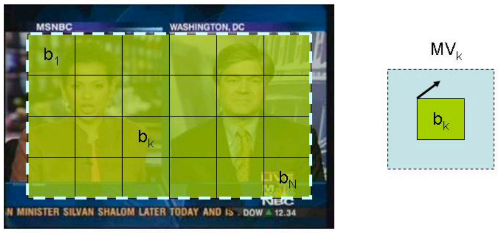


Fig. 2. Visual feature extraction

Within the ROI, we extract the histogram of red, green, blue, and intensity channels and compute a set of common statistics, including the mean, the variance, the skewness (the 3rd order moment), and the flatness (the 4th order moment). We also extract a visual feature called histogram dynamic range, which roughly measures how wide the histogram spreads. To compute the intensity dynamic range, we first search the histogram from both ends, until the accumulated mass of both sides is more than 2%. The dynamic range is the difference of these two values.

For each pixel in the ROI, we compute its discontinuities in the horizontal (with respect to vertical) direction by Sobel operators [6]. If the value is higher than a preset threshold, the pixel is labeled as horizontal (resp. vertical) edge pixel. Finally, we use the ratio of the total number of horizontal (resp. vertical) edge pixels to the size of ROI as an edge based feature.

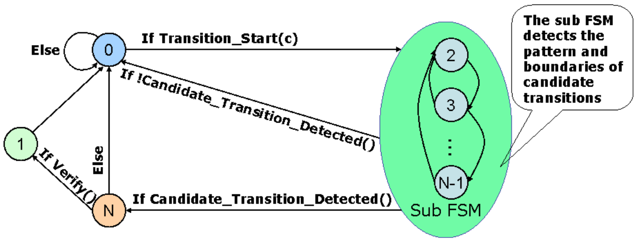
The temporal derivative (delta) of a feature (e.g., histogram mean) is fitted by a second-order polynomial to make it smooth. The delta values of histogram mean, variance, and dynamic range are computed as additional visual features.

Motion features are extracted based on smaller blocks within the ROI. Specifically, in Fig. 2, we split the ROI (288x192 pixels) into 24 blocks (6 by 4), each with the size 48x48 pixels. Based on our observations, motion information extracted from bigger block sizes (e.g., 48x48) is more reliable than those from smaller sizes (e.g., 8x8). The search range of motion vector for each block is set to 32x32. It could be either an exhaustive search for better accuracy or a hierarchical search for higher efficiency. The motion features for each block, e.g., block *k*, include the motion vector ($MV_k$), the best matching error ($ME_k$), and the matching ratio ($MR_k$). The matching ratio is the ratio of the best matching error with the average matching error within the searching range, and it measures how good the matching is. The value is low when the best matching error is small and the block has significant texture. Based on the motion features of all blocks, we select the dominant motion vector and its percentage (the ratio of the number of blocks with this motion vector to the total number of blocks) as frame level features. We then rank all $ME_k$ (resp. $MR_k$), and compute the order statistics, including the mean, $ME_A$; the median, $ME_M$; the average of the top 1/3, $ME_H$; and the average of the bottom 1/3, $ME_L$ (resp. $MR_A$, $MR_M$, $MR_H$, $MR_L$). These features are effective in differentiating the localized visual changes (e.g., foreground changes only) from the frame wised visual changes. For example, high $MR_H$ with low $MR_A$ indicates a localized transition.

Totally, we extract 88 visual features for each frame. Interested readers can find more details in [9].

## 4. SHOT BOUNDARY DETECTORS

Fig. 3 illustrates the general FSM structure for all shot boundary detectors. State 0 is the initial state. When the transition start event is detected, the detector enters the sub FSM, which detects the target transition pattern, and locates the boundaries of the candidate transition. If the sub FSM fails to detect any candidate transition, it returns to state 0, otherwise, it enters state N. State N further verifies the candidate transition with more strict criteria, and if the verification succeeds, it transfers to state 1, which indicates that a transition is successfully detected, otherwise, it returns to the initial state. Although the six detectors share the same general FSM structure, their intrinsic logic and complexity is quite different. In the rest of this section, we briefly discuss all the individual detectors. For more details, please refer to [9].



Else
0
If Transition_Start(c)
If !Candidate_Transition_Detected()
1
If Verify()
Else
N
If Candidate_Transition_Detected()
2
3
⋮
N-1
Sub FSM
The sub FSM detects the pattern and boundaries of candidate transitions

| State | Description |
|---|---|
| 0 | Initial state |
| 1 | Transition detected state |
| 2 ~ N-1 | Candidate transition detection states |
| N | Verification state |

| State Variables |
|---|
| Current state |
| Transition start frame |
| Transition end frame |
| Detector specific variables |

Fig. 3. General FSM for transition detectors

### 4.1. Cut detector

Cut detector uses a state variable, *AverageME*, to track the average value of matching errors. Its initial value is set to 5.0, and it is updated whenever the state is 0 with the following IIR filter,

$$AverageME = AverageME * 0.85 + ME_A * 0.15 \qquad (1)$$

If the current mean matching error, $ME_A$, is larger than 5 times of *AverageME*, the sub FSM is activated. The main roles of the sub FSM are to check whether the candidate boundary has the local maximum matching error, and to introduce a 3-frame delay for verification. The *Verify()* function compares all pairs of frames in the neighborhood (within 3 frames) of the boundary, such that false cuts introduced by camera flashes can be effectively removed.

We also developed a cut verification engine based on a support vector machine (SVM) [7]. Assuming k is the end frame of a candidate cut, we extract four groups of features. The first group is the original visual features (88 dimensions) of frame *k*. The second group is the mean and the standard deviation of all features within an 11-frame window centered at *k*. The third and the last group of features are the same statistics on a 21-frame window and a 31-frame window. All these features are concatenated into a 616-dimension feature vector as SVM input. More details of SVM training are shown in Section 4.7.

### 4.2. Fade in detector

Fade in can be reliably detected using the intensity histogram variance. Low variance (not necessarily low intensity) is a strong indicator for the beginning of fade in. Fade in transitions often start from a group of low variance frames and then the variance gradually increases until it becomes stabilized.

The verification algorithm pinpoints the starting and the ending frames of the candidate transition based on the variance value, and it then measures the linearity of the standard deviation (STD) of the intensity (the square root of intensity variance). We use r2 as a measure of linearity in linear regression. Assume we have a set of pairs: $\{x_i, y_i\}$, $1 \le i \le N$. By min square error, we get the optimal a and b, which minimize the error $E_{reg}$,

$$E_{reg} = \sum_{i=1}^{N} (y_i - ax_i - b)^2 \cdot$$

r2 is defined by,

$$r2 = 1 - \frac{E_{reg}}{E_{tot}}, \text{ where } E_{tot} = \sum_{i=1}^{N} \left( y_i - \overline{y} \right)^2 . \qquad (2)$$

If the linearity of the STD curve is higher than a preset threshold, the *Verify()* function returns true, otherwise, it returns false.

### 4.3. Fade out detector

Similar to the fade in detector, the fade out detector is also triggered by low variance frames. The verification algorithm checks the linearity of the standard deviation of the intensity. Very often, fade out and fade in transitions are adjacent, and the overlapped fade out /in transitions are merged into a single FOI transition in result fusion step.

### 4.4. Fast dissolve detector

Fast dissolve is triggered by a medium change of the matching error, where $ME_A$ is bigger than 2**AverageME*. Let X, Y, and Z denote the starting frame, the ending frame, and a middle frame within a fast dissolve transition. We require that the duration of the fast dissolve transition be less than 5 frames, so it is reasonable to assume that there is no motion involved in the transition. With this assumption, Z can be written as a linear combination of X and Y, Z = αX + (1 - α)Y, where $0 \le \alpha \le 1$. The value of α can be determined by a minimum square error criterion. If the fitting error is smaller than a preset threshold and $0.2 \le \alpha \le 0.8$ for all middle frames of the transition, then the *Verify()* function returns true.

### 4.5. Dissolve detector

Dissolve is a procedure of linearly mixing two different scenes X and Y. Assuming $Z_i$ is an intermediate frame, then we can use the following formula to represent $Z_i$,

$$Z_i = \alpha_i X + (1 - \alpha_i) Y ,$$

where $\{\alpha_i\}$ are a set of monotonically increasing values that are in the range of [0, 1]. Let the variances of X, Y, and $Z_i$ be $\sigma^2_X$, $\sigma^2_Y$, and $\sigma^2_{Zi}$. If we also assume X and Y are independent, then we have,

$$\sigma^2_{Z_i} = \alpha_i^2 \sigma_X^2 + \left(1 - \alpha_i\right)^2 \sigma_Y^2$$

If $\sigma^2_X = \sigma^2_Y$, the curve for $\sigma^2_{Zi}$ is a symmetric quadratic function, shown as in Fig. 4a. But in typical cases, the curve is more like that shown in Fig. 4b, where $\sigma^2_X$ is not equal to $\sigma^2_Y$, and X and Y are not independent. When the variance of either X or Y is small, the variance curve may only contain either the decreasing or the increasing pattern, such as illustrated in Fig. 4c.

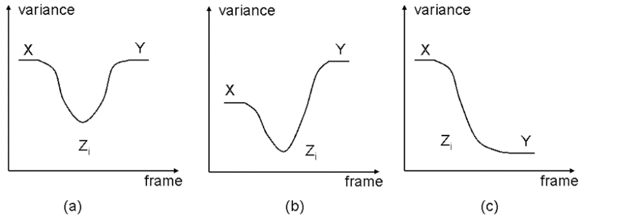


Fig. 4. The variance curves of some typical dissolve transitions

The sub FSM of the dissolve detector is designed to capture the characteristic curves shown in Fig. 4. A state variable, *AverageVariance*, is used for pinpointing the starting and ending frame of the dissolve transition. Its initial value is set to 3.5 and it is updated by following IIR filter in state 0,

$$AverageVariance = AverageVariance * 0.85 + HV_I * 0.15 \qquad (3)$$

where $HV_I$ is the intensity histogram variance.

Verification is a key component of this FSM. The main challenge is that the variance curve may not be smooth due to motion or camera flashes in the original sequences X and/or Y. For verification purposes, we extract a set of heuristic features based on the entire transition. In this section, we only present a few interesting features, for more details, please refer to [9].

From the variance curve, shown in Fig. 5, we first pinpoint the starting and ending frames. To do that, we start from the minimum variance frame in the candidate transition, and then search forward and backward for the maximum absolute delta variance frames, which are $f_{min}$ and $f_{max}$ in the figure. Then from $f_{min}$, we further search backward until the delta variance of the current frame is less than half of the delta variance of the next frame or 2**AverageVariance*. This frame is set as the starting frame of the candidate dissolve. Similarly, we search from $f_{min}$ forward, and locate the ending frame.

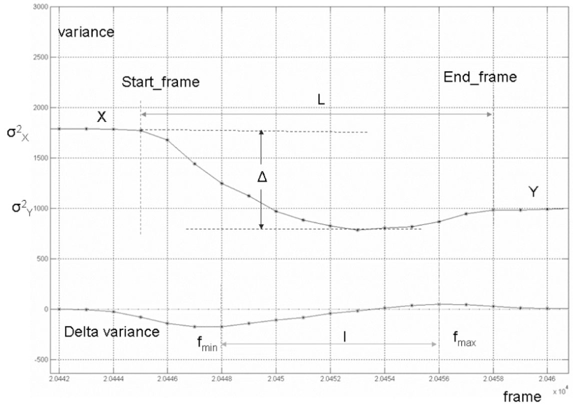


Fig. 5. The curves of variance and delta variance

Then a set of heuristic features are extracted for verification purpose. For example, the height of the variance curve, Δ, is the difference of the maximum and minimum variances within the transition. Knowing that the delta variance is roughly a linear curve between $f_{min}$ and $f_{max}$, we do a linear fitting for the delta variance. We also compute the estimation error for each image in the transition from its neighboring images, and the matching error between the starting and ending frames of the transition.

The baseline dissolve verification employs a sequence of threshold-based criteria relying on these features. A more robust approach is to apply SVM on these features, and we discuss this more in Section 4.7.

### 4.6. Wipe detector

Wipe is the most ill defined transition. 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. In this system, we only consider one common type of wipe, where the first scene gradually changes to the second scene, and for a certain intermediate frame, part of the frame comes from the first scene, and part of it comes from the second scene.

A wipe is triggered by a smooth change, when the matching error $ME_A$ is bigger than 1.5**AverageME* and less than 4**AverageME*. In Fig. 6, we denote the starting and the ending frames of the candidate wipe transition as X and Y, and an intermediate frame as $Z_i$, *i = 1, ..., L -1*, where *L* is the duration of the transition. We partition frame $Z_i$ into 8x8 blocks, and find the best match with motion compensation from both X and Y for each block. When the matching error is too high, the block does not come from either X or Y. Then we compute the portion of blocks with match from X, denoted as $x_i$, and the portion of blocks with match from Y, denoted as $y_i$. Finally, we measure the linearity of $x_i$ and $y_i$ curves to verify the wipe transition.

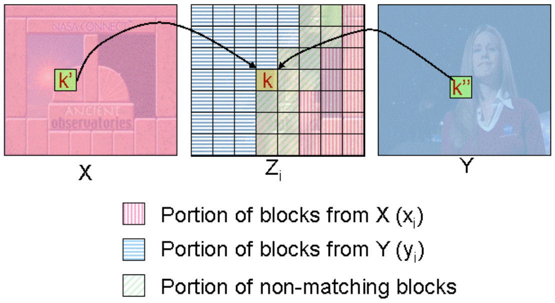

k'
k
k''
X
Z_i
Y
Portion of blocks from X (x_i)
Portion of blocks from Y (y_i)
Portion of non-matching blocks

Fig. 6. Illustration of wipe verification

### 4.7. SVM Models

Support vector machines are now a standard for fast and robust classification. While this classifier greatly reduces training time by analyzing only marginal samples, care must be given to the training parameters and underlying kernel chosen. In our experiments, we evaluated both linear and radial basis functions in a 3-fold validation process. We searched 7 linear settings and 70 RBF settings with random subsets of our training set split into 80/20 percent training/testing partitions. All features are globally normalized with a sigmoid before feeding the SVM.

## 5. FUSION OF DETECTOR RESULTS

Fusion of detector results occurs when all frames are processed. We first sort the list of raw results by their starting frames and then merge all overlapped transitions with different priorities assigned to each transition type. Currently the order used is (from highest to lowest) FOI, dissolve, fast dissolve, cut, and wipe. The final step is to map the system types into two categories: cut and gradual. All shot boundaries except cuts are mapped into gradual.

## 6. EVALUATION RESULTS

In the TRECVID SBD evaluation, each group can submit up to 10 runs. Fig. 7 shows the overall performance of all participants, with AT&T's runs plotted with squares. In term of F-measure, our system achieved the best overall performance. Table I shows the four best submissions for AT&T's SBD system in TRECVID2006.

Table I. The best runs of AT&T's submissions

| Category | Performance (%) | | | Report localized changes | SVM Verification Kernel |
|---|---|---|---|---|---|
| | Recall | Precision | F-Measure | | |
| **Overall** | **85.5** | **89.2** | **87.3** | No | Linear SVM |
| Cut | 88.9 | 90.4 | 89.6 | | |
| Gradual | 76.5 | 85.6 | 80.8 | | |
| Frame | 87.1 | 91.9 | 89.4 | | |
| Overall | 85.1 | 87.6 | 86.3 | No | None |
| **Cut** | **89.4** | **90.4** | **89.9** | | |
| Gradual | 73.6 | 79.5 | 76.4 | | |
| Frame | 86.9 | 93.0 | 89.8 | | |
| Overall | 83.8 | 90.5 | 87.0 | Yes | RBF 2 |
| Cut | 86.2 | 92.2 | 89.1 | | |
| **Gradual** | **77.5** | **85.8** | **81.4** | | |
| Frame | 87.4 | 92.3 | 89.8 | | |
| Overall | 82.6 | 90.9 | 86.6 | Yes | RBF 1 |
| Cut | 86.1 | 92.3 | 89.1 | | |
| Gradual | 73.1 | 86.9 | 79.4 | | |
| **Frame** | **88.9** | **92.1** | **90.5** | | |

Among these results we varied the usage of local changes and the inclusion of an SVM verification stage. The SVM based dissolve verification boosts the overall performance by 2.5% and gradual transition performance by 3.4%, a significant improvement when the initial performance is already high. The frame based gradual transition performance of all our 10 runs leads the other systems by more than 3.5%, meaning the proposed gradual transition (mainly the dissolves) boundary location approaches are very accurate. Also, on an Intel 3.7GHz Xeon machine, all of the proposed system runs faster than 0.4x real time.

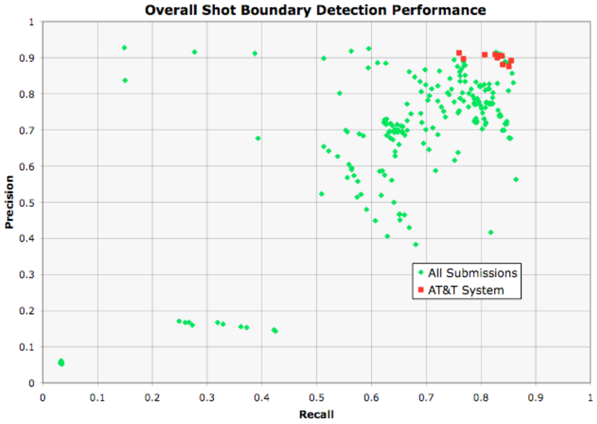


Fig. 7. SBD overall performance in TRECVID2006

## 7. CONCLUSIONS

In this paper, we described a system developed for the shot boundary determination task in TRECVID 2006. The evaluation results show that our proposed SBD algorithm is effective and robust enough to detect several different types of cuts and gradual transitions. We also demonstrated that with a simple fusion of FSM's and optional SVM verification, we achieved very high performance at execution times faster than real time.

## 8. REFERENCES

[1] H. J. Zhang, A. Kankanhalli, S. W. Smoliar, "Automatic Partitioning of Full-motion Video," ACM Multimedia System, Vol. 1, No. 1, pp. 10-28, 1993.

[2] B. L. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video," IEEE Transactions on Circuits and Systems for Video Technologies, 5(6), pp. 533-544, 1995.

[3] B. Shahraray, "Scene Change Detection and Content-based Sampling of Video Sequences," in Digital Video Compression: Algorithms and Technologies 1995, *Proc. SPIE 2419*, February 1995.

[4] Y. Wang, Z. Liu, and J. Huang, "Multimedia Content Analysis Using Audio and Visual Information," *IEEE Signal Processing Magazine*, pp.12-36, Nov. 2000.

[5] W. Kraaij, P. Over, A. Smeaton, "TRECVID 2006 - An Introduction," TRECVID 2006 Workshop, Gaithersburg, MD, Nov. 13-14, 2006.

[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, 1993.

[7] V. N. Vapnik, "Statistical Learning Theory," John Wiley & Sons, 1998.

[8] D. Gibbon, Z. Liu, and B. Shahraray, "The MIRACLE video search engine," *IEEE CCNC*, Jan. 2006.

[9] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, P. Haffner, "AT&T Research at TRECVID 2006," TRECVID 2006 Workshop, Gaithersburg, MD, Nov. 13-14, 2006.