AT&T RESEARCH AT TRECVID 2006

Zhu Liu, David Gibbon, Eric Zavesky, Behzad Shahraray, Patrick Haffner

AT&T Labs – Research
200 Laurel Avenue South
Middletown, NJ 07748
{zliu, dcg, ezavesky, behzad, haffner}@research.att.com

ABSTRACT

TRECVID (TREC Video Retrieval Evaluation) is sponsored by NIST to encourage research in digital video indexing and retrieval. It was initiated in 2001 as a "video track" of TREC and became an independent evaluation in 2003. AT&T participated in three tasks in TRECVID 2006: shot boundary determination (SBD), search, and rushes exploitation. The proposed 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 machine (SVM) is applied to cut and dissolve detectors to further boost the SBD performance. AT&T collaborated with Columbia University in the search and rushes exploitation tasks. In this paper, we mainly focus on the SBD system and briefly introduce our effort on the search and the rushes exploitation. The AT&T SBD system is highly effective and its evaluation results are among the best.

I. INTRODUCTION

TRECVID started as a video track of TREC (Text Retrieval Conference) in 2001 to encourage research in automatic segmentation, indexing, and content-based retrieval of digital video. Since 2003, it became an independent evaluation. TRECVID 2006 contains three system tasks: shot boundary determination, high-level feature extraction, search (interactive, manually-assisted, and/or fully automatic), and one exploratory task: rushes exploitation. AT&T has been active in the multimedia processing, indexing, and search areas for many years. TRECVID provides us a good opportunity to learn from the other research groups and to share our experience with our colleagues. For the first time, AT&T participated in TRECVID 2006. We submitted results for shot boundary determination (SBD), and we also submitted search and BBC Rush task results in collaboration with Columbia University. For the rushes exploitation task, we demonstrated a content based video browsing and query system, which was built on the MIRACLE (Multimedia information retrieval by content) platform from AT&T Labs and the high level feature extraction technologies developed at Columbia University.

Shot boundary determination has been widely studied for the last decade. Some of the early work can be found in [1-4]. TRECVID further stimulates the interest and effort in a much broader 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 back in the 1990s, 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) [6] was designed to detect all types of scene changes and report camera motions, including panning and tilting.

Thanks to the current computation power (Intel 3.7GHz Xeon CPU) and the non real time requirement, there is a lot of room to extend the existing algorithm. Three major improvements are: 1) Two-dimension motion compensation is utilized. 2) In addition to the intensity values, color information is taken into account. 3) Instead of using a single FSM, multiple FSM-based detectors are adopted to track different types of transitions, 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 transitions.

For the search task, we only participated in the fully automatic search evaluation task, collaborating with Columbia University (CU). The baseline submissions are based on the existing systems built in CU, and query expansions based on the name entities extracted from various external data are built on top of the baseline systems.

We combined several tools from existing systems to better organize the BBC data and showcased it in the MIRACLE system. A large set of the LSCOM visual concepts, as trained on the TV05 dataset by Columbia University, were extracted for the rushes data and an audio sound-type classifier (roughly trained on the BBC06 data) was applied to mark interested audio events, e.g., speech, silence, and noise. We also automatically generated a semi-synchronized script from the scanned documents provided by NIST to provide certain search capability.

This paper is organized as follows. Section II gives a detailed description of the shot boundary determination system. Sections III and IV briefly address our work on the search and rushes exploitation, respectively. Evaluation results are also presented in these sections. Finally, we draw our conclusions in Section V.

II. SHOT BOUNDARY DETERMINATION

2.1 Overview

Fig. 1 shows the high level diagram of the shot boundary determination system. To decode the TRECVID evaluation sequences which are in the MPEG-1 format, we use the open source MPEG decoder, developed by the MPEG Software Simulation Group (MSSG) [5]. Although this codec is not the most efficient choice, it is easy to manipulate and is portable, such that we can run the SBD system on both Windows and Unix platforms.

There are three main components in the system: visual feature extraction, shot boundary detectors, and result fusion. The top level of the algorithm runs in a loop, and every loop processes one image frame. Each new frame is saved in a circular frame buffer, whose size is 256 frames. The extracted visual features are saved in a circular feature buffer with the same size. For ease of developing the algorithm, most parameters that we mention (e.g., the buffer size) in this section can be configured in a control file. To simplify the notation, we will refer to the adopted values directly in the rest of this section. The frame and feature buffers are shared by all shot detectors, such that common features can be reused, and detector specific features can be easily computed. The size of the buffer is determined by the maximum duration of shot transitions. The loop continues until all frames in the MPEG file are processed.

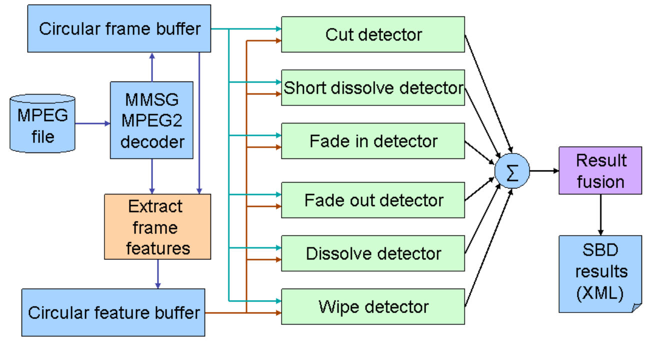


Fig. 1. Overview of the SBD system

Given the wide varieties of shot transitions, it is difficult to handle all of them using one super detector. Our system adopts a "divide and conquer" strategy for SBD. We built six independent detectors, targeting for six dominant shot boundaries in the SBD task. These detectors are cut detector, fast dissolve (less than 5 frames) detector, fade in detector, fade out detector, dissolve detector, and wipe detector. With this architecture, each detector can be tuned separately, and new detectors can be easily plugged in when necessary.

Essentially, each detector is a finite state machine (FSM), whose state is changed by checking the new frame. Each FSM may have a different number of states, but for all FSMs, we intentionally use state 1 as the shot detected state. The advantage is that at the end of each loop, we can easily find out whether new shot boundaries are detected by checking the states of all FSMs. The results of all detectors are merged together in the temporal order.

The top level loop of the algorithm terminates when all frames in the MPEG file are processed. Then, the results generated by the six detectors are fused. Since these results may be overlapped (e.g., a cut overlapped with a fast dissolve), the fusion block needs to merge and clean up the overlapped shot boundaries. Finally, to comply with the TRECVID SBD format, we map all shot boundaries except cuts into gradual.

2.2 Feature Extraction

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

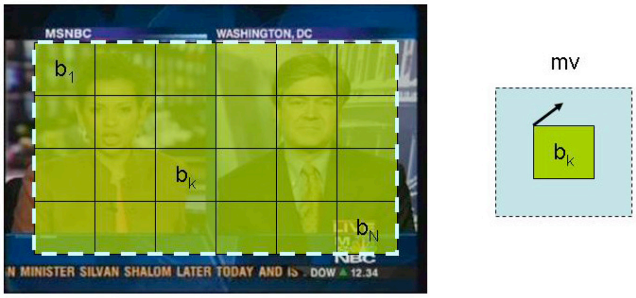


Fig. 2. Visual feature extraction

Fig. 2 illustrates how these visual features are computed. The resolution of the TRECVID evaluation sequences is 240x352 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 normally the center of the image captures more content. 2) The ROI gets rid of the border of the image where usually sliding text or black bands (when showing wide screen content) appear.

Within the ROI, we extract the histogram of red, green, blue, and intensity channels. Based on the histogram, we compute a set of common statistics, including the average, 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. Fig. 3 shows the intensity histogram of a frame. 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%. Then the dynamic range is the difference of these two values. In Fig. 3, the low intensity value L is 16 and the high intensity value H is 175, and consequently, the dynamic range is H - L = 159. Similarly, we compute the dynamic range for red, green, and blue channels.

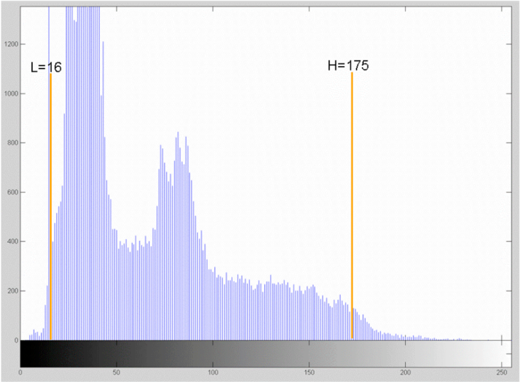


Fig. 3. The computation of histogram dynamic range

For each pixel in the ROI, we compute its discontinuities in the horizontal (respectively, vertical) direction by Sobel operators [7]. If the value is higher than a threshold, the pixel is labeled as horizontal (respectively, vertical) edge pixel. Finally, we use the ratio of the total number of horizontal (respectively, vertical) edge pixels to the size of ROI as an edge based feature.

We compute two sets of inter-frame features, one is based on the current frame (frame c) and the previous frame (frame c-1), and the other is based on the current frame (frame c) and the frame that is N frames away, where N=6 (frame c-6). The first one is to capture frame by frame changes, and the second one is to capture the change over a longer period, which is useful for detecting smooth changes in the video. In this section, we describe how the first set of inter-frame features is computed, and the second set of features can be computed in a similar fashion.

The temporal derivative (delta) of a feature (e.g., histogram mean) is fitted by a second-order polynomial to make it smooth [8]. The delta values of histogram mean, variance, dynamic range are computed. The distance between two single channel (e.g., red) histograms is computed based on the quadratic color histogram distance [9]. Assume **g** and **h** are two histogram vectors and **A**=[a_{ij}] is the similarity matrix, the distance between **g** and **h** is computed by,

$$d(\mathbf{g}, \mathbf{h}) = (\mathbf{g} - \mathbf{h})^t \mathbf{A} (\mathbf{g} - \mathbf{h}), \text{ where } a_{ij} = 1 - |i-j|/255.$$

The overall difference (e.g., histogram mean) is computed as a weighted summation of the difference of channels R, G, and B. The weighting factors are 0.299, 0.587, and 0.114, respectively.

In addition to the red, green, blue (RGB) based histogram distance, we also compute the histogram distance in hue, saturation, value (HSV) space. Basically, the HSV space is vector quantized into 256 bins, where H and V values are assigned 8 levels each, and the S value is assigned 4 levels. We compute the corresponding **A** matrix using the centers of each bin.. Assume the centers of two bins i and j are {h_i, s_i, v_i} and { h_j, s_j, v_j}, then a_{ij} is computed as,

$$a_{ij} = 1 - \frac{1}{\sqrt{5}} \left[(v_i - v_j)^2 + (s_i \cos(h_i) - s_j \cos(h_j))^2 + (s_i \sin(h_i) - s_j \sin(h_j))^2 \right]^{\frac{1}{2}}.$$

Motion features are extracted based on square blocks within the ROI. Specifically, in Fig. 2, we split the ROI (192x288 pixels) into 24 blocks (4 by 6), each with the size 48x48 pixels. Based on our observations, the 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 include the motion vector (mv_k), the 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. mr_k measures how good the match is, the lower value the better. mr_k is low when there is perfect matching and the block has significant texture. After the motion features of all blocks are available, we pick 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. If the percentage is high enough and the motion vector is not (0, 0), we set the motion flag to 1, otherwise we set it to 0. We then sort the arrays of matching error (respectively, matching ratio), and compute the mean, ME_A (resp. MR_A); the median, ME_M (resp. MR_M); the average value (high) of the top N/3 blocks, ME_H (resp. MR_H); and the average value (low) of the bottom N/3 blocks, ME_L (resp. MR_L). Table I summaries all features extracted from one frame.

Table I. List of visual features

Type	Visual feature	Dimension
Intra-frame	Histogram mean (HM_R, HM_G, HM_B, HM_I)	4
	Histogram variance (HV_R, HV_G, HV_B, HV_I)	4
	Histogram skewness (HS_R, HS_G, HS_B, HS_I)	4
	Histogram flatness (HF_R, HF_G, HF_B, HF_I)	4
	Histogram dynamic range (HDR_R, HDR_G, HDR_B, HDR_I)	4
	Edge ratio (Horizontal and Vertical)	2
Inter-frame (1 frame)	Delta histogram mean (DHM_R, DHM_G, DHM_B, DHM_I, DHM_A)	5
	Delta histogram variance (DHVR, DHVG, DHVB, DHVI, DHVA)	5
	Delta histogram dynamic range ($DHDR_R$, $DHDR_G$, $DHDR_B$, $DHDR_I$, $DHDR_A$)	5
	Histogram distance (HD_R, HD_G, HD_B, HD_I, HD_A)	5
	Histogram distance in HSV space	1
	Motion flag and block percentage	2
	Motion vector (horizontal, vertical)	2
	Average (ME_A), low (ME_L), high (ME_H), median (ME_M)of matching error	4
	Average (MR_A), low (MR_L), high (MR_H), median (MR_M) of matching ratio	4
Inter-frame (6 frames)	The same as inter-frame (1 frame)	33
Total		88

2.3 Shot boundary detectors

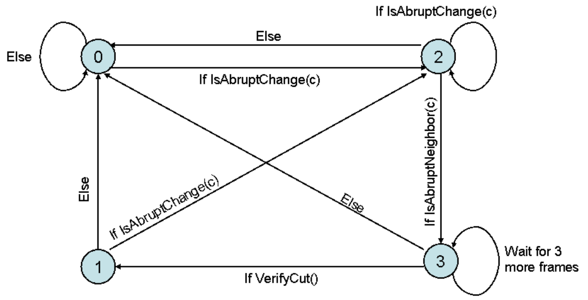
In this section, we describe 6 detectors, which detect 6 common shot boundaries: cut, fast dissolve (less than 5 frames), fade in, fade out, dissolve, and wipe. These 6 types of transitions cover most shot transitions in TRECVID sequences and they can be detected relatively reliably.

Each detector is implemented as a finite sate machine. Before we touch the specific details of each detector, we first introduce their common characteristics. Each FSM has a set of states, labeled from 0 to N, where N is equal to 3 or 4. State 0 is the initial state, waiting for the trigger of a certain event. State 1 is used to flag the detection of certain shot boundaries. The other states are used to represent the intrinsic patterns of various shot boundaries. The state of the FSM is determined by a set of state variables. There are three basic state variables that are common for all FSMs: state_id, which is the state of current FSM, start_frame, which is the last frame of previous shot, end_frame, which is the first frame of the new shot. Some detectors may have an additional state variable to track an adaptive threshold value used for determining the state transitions.

2.3.1 Cut detector

Fig. 4 illustrates the FSM for cut detector and its state variables. There are four states in the cut FSM and one of its state variables, AverageME, is used 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 infinite impulse response (IIR) filter,

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



State Variables
State_ID
Start_frame
End_frame
AverageME

State_ID	Description
0	Initial state
1	Cut detected state
2	Possible cut detected state
3	Cut verification state

Fig. 4. Cut detector

The function IsAbruptChange compares the average matching error (ME_A) of the current frame (frame c) with a threshold, which is 5 times of AverageME. If the current MEA is bigger than the threshold and it is bigger than those of five previous frames, function IsAbruptChange returns true, otherwise, it returns false. If the current frame satisfies the abrupt change criteria, the FSM enters state 2, and the start_frame and end_frame are set to c - 1 and c respectively. Using AverageME to determine the threshold adaptively, instead of a fixed threshold enables the cuts in low intensity frames to be reliably detected.

When the FSM is in state 2, it tests whether the current frame is a valid neighboring frame of a cut. Basically, the function IsAbruptNeighbor compares the matching errors of current frame and the previous frame. If the ME_A of the current frame is smaller, the function returns true, otherwise it returns false. If the current frame is still an abrupt change, the FSM stays at state 2, updating the start_frame and end_frame, otherwise, it returns to state 0. State 3 simply waits for 3 more frames such that we can verify the cut using more neighboring frames. The VerifyCut function compares frame start_frame with all frames from end_frame + 1 to end_frame + 3, and frame end_frame with all frames from start_frame – 4 to start_frame – 1. The similarity is determined by both motion compensated matching errors and pixel-wise image correlation. The purpose is to detect any camera flash related false alarms.

We found that the cases when cut happens only on a portion of a frame are not consistently labeled in TRECVID references. We call these cases local cut, and this occurs, for example when there is a cut in a video that has been inserted into a small window in a static shot. In our system, we use the low matching error ME_L to control whether a local cut is detected or not. Enabling local cut detection leads to higher recall, but lower precision.

Besides the threshold based cut verification method, we also developed a support vector machine (SVM) [10] based cut verification engine. Fig. 5 illustrates the feature extraction method for SVM input. Assume k is the end_frame of a candidate cut, and 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 group is the same statistics on a 21 frame window. The last group of features is based on a 31 frame window. All these features are concatenated together into a 616 dimension feature vector and it is used for SVM input. More details about SVM training can be found in Section 2.3.7.

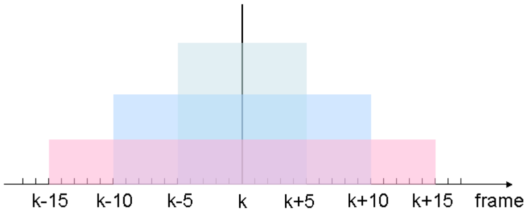
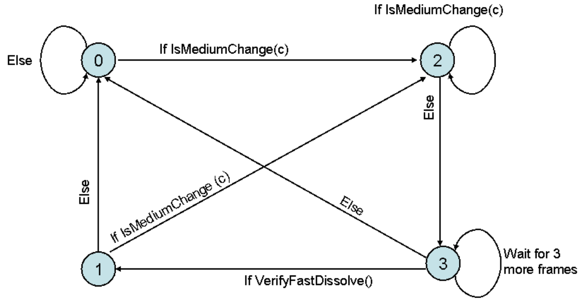


Fig. 5. Feature extraction for cut verification using SVM

2.3.2 Fast dissolve detector

Fig. 6 shows the fast dissolve detector FSM, which contains 4 states, and has 4 state variables.



State Variables
State_ID
Start_frame
End_frame
AverageME

State_ID	Description
0	Initial state
1	Fast dissolve detected state
2	Possible fast dissolve detected state
3	Fast dissolve verification state

Fig. 6. Diagram of fast dissolve detector

The fast dissolve is triggered by a medium change of the matching error, where ME_A is bigger than 2 * AverageME. AveageME is initiated by a value of 5.0, and it is updated using formula (1) whenever the FSM is in state 0. When the FSM changes from state 0 to 2, state variables start_frame and end_frame are set to c - 1 and c. State 2 stays at the same state if the current frame keeps being a medium change, or it jumps to state 3 and updates end_frame to be c. State 3 simply waits for 3 more frames, and then it verifies whether the candidate transition is really a fast dissolve.

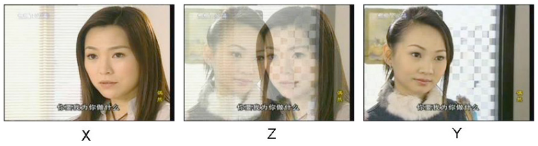
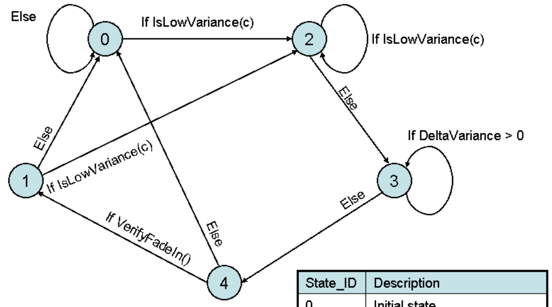


Fig. 7. Typical fast dissolve

Fig. 7 shows a typical fast dissolve, which spans 3 frames. Let X, Y, and Z denote the start_frame, end_frame, and a middle frame of the 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, $Z = \alpha X + (1 - \alpha)Y$, where $0 \leq \alpha \leq 1$. The value of α can be determined by a min square error criteria. 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 VerifyFastDissolve function returns true, otherwise, false.

2.3.3 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. Normally, fade in transitions start from a group of low variance frames and then the variance gradually increases until it becomes stabilized. Fig. 8 shows the diagram of fade in detector FSM. The FSM contains 5 states and 3 state variables. State 2 is activated when the current frame is in low variance mode and the start_frame is set to record the starting of the transition. When the frame is no longer of low variance, the FSM moves to state 3, and stays there as long as the variance increases significantly. Once the variance is stable or starts to decrease, FSM enters state 4, where the verification is conducted.



State Variables
State_ID
Start_frame
End_frame

State_ID	Description
0	Initial state
1	Fade in detected state
2	Low variance state
3	Fade in middle state
4	Fade in verification state

Fig. 8. Fade in detector

The verification code pinpoints the start and end frames of the candidate fade in 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, such that the following error is minimized,

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

r2 is defined as,

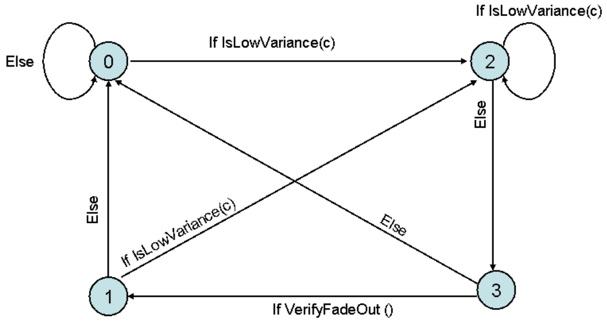
$$r2 = 1 - \frac{E_{reg}}{E_{tot}}, \text{ where } E_{tot} = \sum_{i=1}^{N} (y_i - \bar{y})^2, \qquad (2)$$

and y is the mean of $\{y_i\}$. If the linearity of the STD curve is higher than a certain threshold, the VerifyFadeIn function returns true, otherwise, it returns false.

2.3.4 Fade out detector

The fade out detector is also triggered by low variance frames. The corresponding FSM is shown in Fig. 9. State 2 is the low variance state, and it goes to state 3 when the current frame's intensity variance is no longer low. State 3 pinpoints the starting and ending frame of the transition, and verifies whether the candidate transition is a fade out or not. The verification procedure is similar to that of the fade in detector. The main method is to check the linearity of the standard deviation of the intensity variance.

Very often, fade in and fade out transitions are adjacent, and the overlapped fade in and fade out transitions are merged into a FOI transition in the result fusion stage to be consistent with the TRECVID labeling conventions.



State Variables
State_ID
Start_frame
End_frame

State_ID	Description
0	Initial state
1	Fade out detected state
2	Low variance state
3	Fade out verification state

Fig. 9. Fade out detector

2.3.5 Dissolve detector

Dissolve is the main gradual transition, and we will present more details in this section. It is well known that the intensity variance is a good indicator for detecting dissolve. Assume that the dissolve is a procedure of linearly mixing of two different scenes X and Y, and Z_i is one 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 σ^2_X, σ^2_Y, and σ^2_{Zi}. If we also assume X and Y are independent, then we have,

$$\sigma^2_{Z_i} = \alpha_i^2 \sigma^2_X + (1 - \alpha_i)^2 \sigma^2_Y$$

If $\sigma^2_X = \sigma^2_Y$, the curve for σ^2_{Zi} is a symmetric quadratic function, shown as in Fig. 10 (a). But in typical cases, the curve is more like that shown in Fig. 10 (b), where σ^2_X is not equal to σ^2_Y, and X and Y are not independent. When the variance of either X or Y is small, the variance curve may not contain both the decreasing and increasing patterns such as in Fig. 10 (c) which shows an example where σ^2_Y is small.

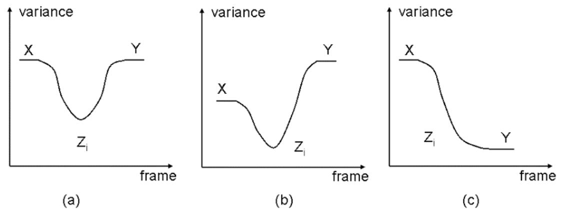
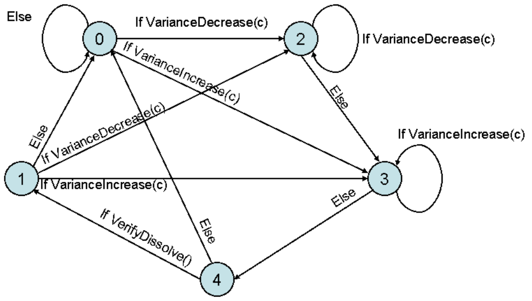


Fig. 10. The variance curves of some typical dissolve transitions

The dissolve detector is designed to capture the characteristic curves shown in Fig. 10. Fig. 11 illustrates the proposed dissolve detector, which has 5 states and 4 state variables. AverageVariance is used for pinpointing the start_frame and end_frame of the dissolve transition. Its initial value is 3.5 and it is updated by following IIR filter in state 0,

AverageVariance = AverageVariance * 0.85 + HV_I * 0.15

State 2 is the variance decreasing state, which corresponds to the decreasing part in Fig. 10 (a), and state 3 is the variance increasing state, which corresponds to the increasing part in Fig. 10 (a). The design of the FSM allows that the cases when either σ^2_X or σ^2_Y is small are reliably handled.



State Variables
State_ID
Start_frame
End_frame
AverageVariance

State_ID	Description
0	Initial state
1	Dissolve detected state
2	Variance decreasing state
3	Variance increasing state
4	Fade in verification state

Fig. 11. Dissolve detector

The verification part is the key component of the FSM, and its main purposes are 1) precisely determines the boundaries of the dissolve, and 2) verifies the candidate dissolve. 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. The following example illustrates the details.

Fig. 12 shows a typical dissolve, and the numbers below the images are their frame numbers. The dissolve starts from frame 20445, and ends at frame 20458. Corresponding variance and delta variance curves are plotted in Fig. 13.

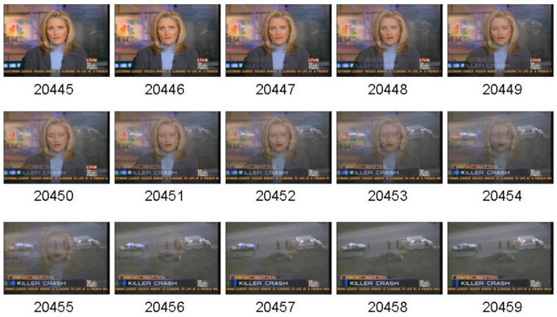


Fig. 12. An example of dissolve

From the variance curve, 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 Start_frame of the candidate dissolve. Similarly, we search from f_{min} forward, and locate the End_frame. The middle of the transition is Middle_frame = (Start_frame + End_frame)/2.

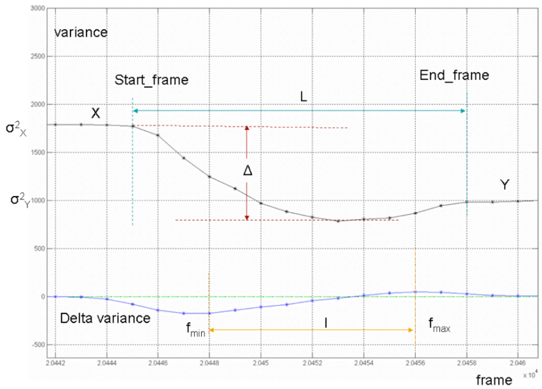


Fig. 13. The curves of variance and delta variance

Then a set of heuristic measurements are taken from the variance and delta variance curves. The height of the variance curve, Δ, is the difference of the maximum and minimum variances within the transition. Ratios hr1 = Δ / min{variance(start_frame), variance(end_frame)} and hr2 = Δ / max{variance(start_frame), variance(end_frame)} are computed to measure the relative height of the variance curve. Knowing that the variance curve is roughly a second order polynomial function, the delta variance should be roughly a linear curve. We do a linear regression for the delta variance within this range, and measure the r2 which is defined in formula (2). As byproducts of the linear regression, we also have the slope s. Similarly, we fit the intensity mean by a linear curve, and compute the corresponding r2 and s. We also use the duration between f_{min} and f_{max}, as one feature, and its ratio to the transition duration, denoted by r = l/L, as another feature.

Then, we measure how well each image in the transition can be estimated from neighboring images. We used five ways to estimate Z_i, i = 3,…, L-2: 1) by Z_{i-2}, 2) Z_{i-1}, 3) by Z_{i+2}, 4) by $(Z_{i-2} + Z_{i+2})/2$, and 5) by $\alpha_i Z_{i-2} + (1-\alpha_i)Z_{i+2}$. The estimation errors are denoted by DEP_i, DEO_i, DEF_i, DEE_i, and DEL_i. The maximum of $\{DEO_i\}$ is Max_DEO. For the fourth estimation, we compute the estimation confidence, DEE_Conf_i, as $sqrt(min(DEP_i, DEF_i) / DEE_i)$. The maximum of $\{DEE_i\}$, Max_DEE, and the average of $\{DEE\_Conf_i\}$, Avg_DEE_Conf, are also used for verification purposes. For the last estimation, we have $\{\alpha_i\}$ and confidence $\{DEL\_Conf_i\}$, which is computed by $sqrt(min(DEP_i, DEF_i) / DEL_i)$. Based on these, we compute the maximum of $\{DEL_i\}$, Max_DEL, the average confidence of $\{DEL\_Conf_i\}$, Avg_DEL_Conf, and the worst $\{\alpha_i\}$, WorstAlpha, which is computed as $max\{abs|\alpha_i-0.5|\}$. For dissolves, ideally, WorstAlpha is 0 (since α_i is 0.5).

To find out the spatial distribution of matching errors, we compute the absolute pixel-wise difference image between frame Middle_frame-2 and frame Middle_frame+2. The average value of the difference image, and its horizontal and vertical centroids are used as extra features, which are denoted as AvgDiff, DiffCH, DiffCV.

Finally, we want to make sure that the dissolve introduces significant content change. We compute the histogram distance, motion matching errors, and pixelwise correlations for 3 pairs of frames: (Start_frame, End_frame), (Start_frame, Middle_frame), and (Middle_frame, End_frame).

Similarly as with the cut detector, we need to differentiate those dissolves that are framewise or partially framewise. We have a flag, GlobalFlag, which is determined by DiffCH, DiffV, and ME_L

between Start_frame and End_frame. If DiffCH and DiffV are small and ME_L is high, GlobalFlag = 1, otherwise, 0.

Table II lists all the features we used for dissolve verification. Some features are omitted (e.g., edge related features) because they are not as promising as the others.

Table II. Features for dissolve verification

Type	Feature	#
Variance	Height, hr1, hr2, r2, s, l, r	7
Intensity	r2, s,	2
Skewness	Height	1
Flatness	Height	1
Delta variance	Max_delta_variance Min_delta_variance	2
Estimation	Max_DEO, Max_DEE, Avg_DEE_Conf, Max_DEL, Avg_DEL_Conf, WorstAlpha	6
Middle_frame	AvgDiff, DiffCH, DiffCV	3
Matching error	Max ME_A, max ME_L, max ME_A,(6 frames apart), max ME_L (6 frames apart), max histogram distance (6 frames apart)	5
Global	Global flag	1
Vertical edge	Max_Delta_Edge, Height, hr1, hr2, r2	5
Horizontal edge	Max_Delta_Edge, Height, hr1, hr2, r2	5
Start_frame & End_frame	$Correlation_{SE}$, RGB histogram distance, HSV histogram distance, ME_M, ME_L. ME_H, MR_M, MR_L, MR_H	9
Start_frame & Middle_frame	$Correlation_{SM}$, RGB histogram distance, HSV histogram distance, ME_M, ME_L. ME_H, MR_M, MR_L, MR_H	9
Middle_frame & End_frame	$Correlation_{ME}$, RGB histogram distance, HSV histogram distance, ME_M, ME_L. ME_H, MR_M, MR_L, MR_H	9
Correlation	$Correlation_{SE}$–0.5($Correlation_{SM}$ +$Correlation_{SM}$)	1
Total		66

The baseline dissolve verification employs a sequence of threshold based criteria relying on these 66 features. A more robust approach is to apply SVM on this feature vector. More details about SVM training can be found in Section 2.3.7.

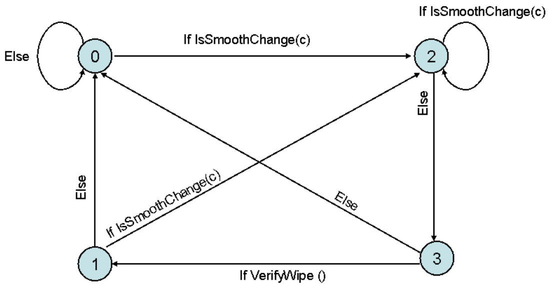
2.3.6 Wipe detector

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 section, we describe our approach for detecting certain kinds of wipes.

Fig. 14 shows the FSM we developed for wipe detection. AverageME is an adaptive state variable updated by formula, AverageME = AverageME * 0.85 + ME_A * 0.15, and its initial value is set to 5.0.

A frame is considered as a smooth change if its matching error ME_A is bigger than 1.5*AverageME and less than 4*AverageME.

State 2 is the smooth change state, and state 3 is the end of smooth change, and it also verifies the candidate transition is a wipe. If the verification logic passed, the detector enters state 1.



State Variables
State_ID
Start_frame
End_frame
AverageME

State_ID	Description
0	Initial state
1	Wipe detected state
2	Smooth change state
3	Wipe verification state

Fig. 14. The Wipe detector

In this system, we only consider one popular type of wipe, which is shown in Fig. 15. The transition starts from frame 21360, ends at frame 21378. Basically, the wipe starts from one scene, and it gradually changes to the second scene, where the dominant portion of every intermediate frame comes from either scene 1 or scene 2. Fig. 16 shows how we measure the fit of this model.

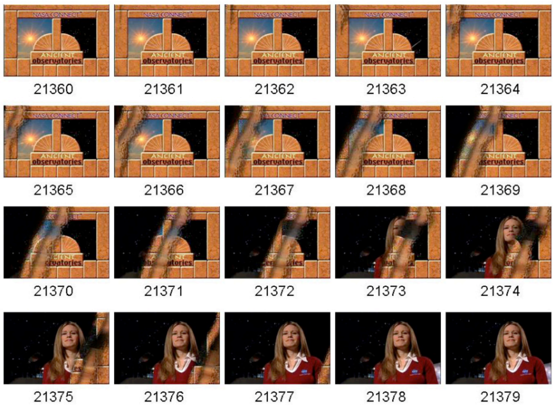


Fig. 15. A sample of wipe transition

In Fig. 16, we denote the starting and ending frames of the candidate wipe transition as X and Y, and one intermediate frame as Z_i, i = 1, ..., L -1, where L is the duration of the transition.

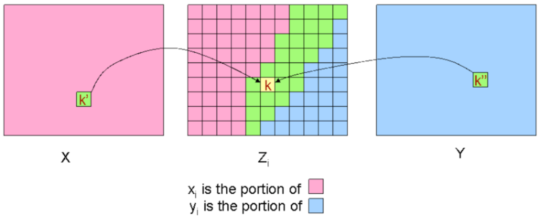


Fig. 16. Illustration of wipe verification

For frame Z_i, we partition it into 8x8 blocks, and for each block, we find the best match with motion compensation from both X and Y. The highlighted block k shows this procedure. If the best matching error is smaller than a preset threshold, it is considered as a valid match. In Fig. 16, for Z_i, we mark those blocks that have valid matches from X in red, and those from Y in blue. The blocks that do not have valid match are painted in green. Then we compute the portion of blocks with valid match from X, denoted as

x_i, which is the ratio of red blocks to the number of all blocks, and the portion of blocks with valid match from Y, denoted as y_i.

Fig. 17 illustrates the ideal patterns of x_i and y_i. We use the linearity of x_i and y_i of the wipe candidate to verify it.

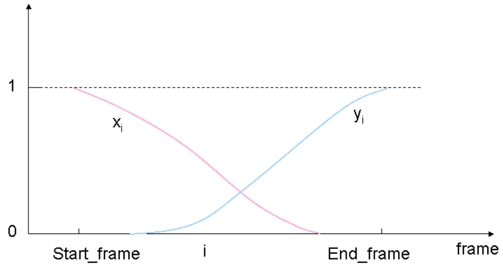


Fig. 17. Curve of x_i and y_i for wipe verification

2.3.7 SVM Models

Support vector machines are now standard for fast and robust classification. While this discriminative classifier greatly reduces training time by analyzing only marginal samples, care must be given to the training parameters and underlying kernel used in an SVM. For 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 training/testing partitions. All features are globally normalized to one before they are analyzed by the SVM.

During the development of our classifier, we performed cross-validation testing across permutations of both the TRECVID2005 and TRECVID2004 datasets. We saw a performance degradation around 10% F1 when combining different data sets, so the final model was constructed with the TRECVID2005 data alone.

2.4 Fusion of Detector Results

Currently, fusion of detector results is conducted at the end, when all frames are processed. Fig. 18 shows a segment of the raw results from all detectors. Because some of the detectors may report the detected transition with delays, we need to sort the list of raw results by their starting frame (preFNum in the figure).

```
<trans type="CUT" preFNum="1572" postFNum="1573" />
<trans type="CUT" preFNum="1604" postFNum="1605" />
<trans type="DISSOLVE" preFNum="1614" postFNum="1624"
/>
<trans type="CUT" preFNum="1672" postFNum="1673" />
<trans type="FADEOUT" preFNum="1728" postFNum="1739"
/>
<trans type="FADEIN" preFNum="1733" postFNum="1743" />
```

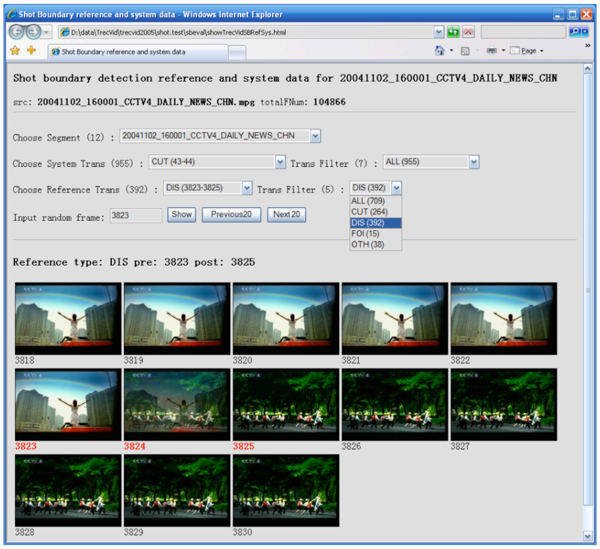
Fig. 18. Segment of raw results

Then we merge the overlapped fade out and fade in transitions, and rename them FOI, as used by the TRECVID reference. For example, the last two transitions in Fig. 18 become <trans type=“FOI” preFNum=“1728” postFNum=“1743” />

The next task is to merge all overlapped transitions with certain priorities assigned to different transitions. Currently, the order of priority we used is (from the highest to the 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.

2.5 System Development Tools

In order to effectively locate the weakness of the algorithm, we developed a set of tools to show the ground truth, the system results, and all false positives and false negatives. Fig. 19 is an interface that shows the ground truth and the system results. Fig. 20 shows the false positives and false negatives.


Shot Boundary reference and system data - Windows Internet Explorer
D:\data\TrecVid\trecvid2005\shot.test\sbeval\showTrecVidSBRefSys.html
Shot Boundary reference and system data
Page
Show
Previous20
Next 20
ALL (709)
CUT (264)
DIS (392)
FOI (15)
OTH (38)
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830

Shot boundary detection reference and system data for 20041102_160001_CCTV4_DAILY_NEWS_CHN

src: 20041102_160001_CCTV4_DAILY_NEWS_CHN.mpg totalFNum: 104866

Choose Segment (12) : 20041102_160001_CCTV4_DAILY_NEWS_CHN

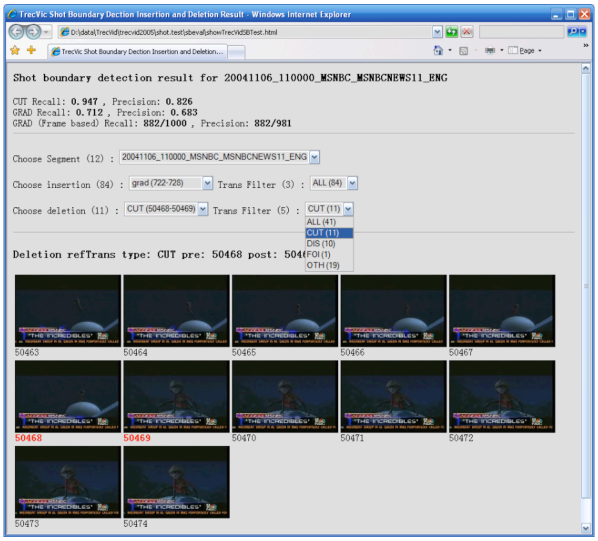
Choose System Trans (955) : CUT (43-44) Trans Filter (7) : ALL (955)

Choose Reference Trans (392) : DIS (3823-3825) Trans Filter (5) : DIS (392)

Input random frame: 3823

Reference type: DIS pre: 3823 post: 3825

Fig. 19. Interface for showing the reference and system results


TrecVic Shot Boundary Dection Insertion and Deletion Result - Windows Internet Explorer
D:\data\TrecVid\trecvid2005\shot.test\sbeval\showTrecVidSBTest.html
TrecVic Shot Boundary Dection Insertion and Deletion...
Page
ALL (41)
CUT (11)
DIS (10)
FOI (1)
OTH (19)
50463
50464
50465
50466
50467
50468
50469
50470
50471
50472
50473
50474

Shot boundary detection result for 20041106_110000_MSNBC_MSNBCNEWS11_ENG

CUT Recall: 0.947 , Precision: 0.826
GRAD Recall: 0.712 , Precision: 0.683
GRAD (Frame based) Recall: 882/1000 , Precision: 882/981

Choose Segment (12) : 20041106_110000_MSNBC_MSNBCNEWS11_ENG

Choose insertion (84) : grad (722-728) Trans Filter (3) : ALL (84)

Choose deletion (11) : CUT (50468-50469) Trans Filter (5) : CUT (11)

Deletion refTrans type: CUT pre: 50468 post: 504

Fig. 20. Interface for showing the system errors

The interface is basically dynamic HTML written in JavaScript. The user can choose different evaluation sequences, and for each chosen sequence, the interface shows all frames of any selected shot, either from the reference or from the system result. The transition filter is also a useful feature, where the user can choose to show certain types of shots in the pull down menu on the left. In Fig. 19, only dissolves are shown for the reference transitions, and the selected dissolve is from 3823 to 3825. Note that these frames are marked in red and bold, and five extra frames on both side of the boundary are also shown. For efficiency, all frames are decoded and saved in advance for fast interactive response. The user can also specify a frame number, and browse 25 frames around that frame. The next 20 frames and the previous 20 frames buttons allow the user to check more neighboring frames around the frame of interest.

Fig. 20 represents a second user interface showing the system errors based on the evaluation tools provided by NIST. The insertion and deletion errors can be displayed separately, and similar to Fig. 19, the transition filter feature helps to show only interested types of errors, e.g., cut or dissolve only.

While developing and adjusting our system, we found that combining these two interfaces were very effective and useful to analyze the weaknesses of the system and to easily figure out the common errors of the system.

Table III. AT&T's 10 submissions for SBD

Run	Block size	Allowing local change	SVM kernel for cut	SVM kernel for Dissolve
1	48x48	Y	N/A	N/A
2		N		
3	32x32	Y		
4		N		
5	48x48	Y	Linear kernel	
6		N		
7		Y	N/A	Linear
8		N		
9		Y		RBF 1
10				RBF 2

2.6 Evaluation Results

Table III shows the 10 runs we submitted for shot boundary determination task. The first two runs are similar, and the only difference is that the first run allows local change for cut and dissolve, and the second one does not. The pair of the third and the fourth runs is similar to the first two, but the block size is different. The major difference between the last six runs and the first four runs is that the SVM is used for verification in runs 5 to 10, either for cut or dissolve. Runs 5 and 6 use linear kernel SVM for cut verification. Runs 7 to 10 use SVM for dissolve verification, with linear, and two RBF kernels trained with different kernel parameters.

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

Run	Category	Performance (%)		
		Recall	Precision	F-Measure
8	**Overall**	**85.5**	**89.2**	**87.3**
	Cut	88.9	90.4	89.6
	Gradual	76.5	85.6	80.8
	Frame based	87.1	91.9	89.4
2	Overall	85.1	87.6	86.3
	Cut	**89.4**	**90.4**	**89.9**
	Gradual	73.6	79.5	76.4
	Frame based	86.9	93.0	89.8
10	Overall	83.8	90.5	87.0
	Cut	86.2	92.2	89.1
	Gradual	**77.5**	**85.8**	**81.4**
	Frame based	87.4	92.3	89.8
9	Overall	82.6	90.9	86.6
	Cut	86.1	92.3	89.1
	Gradual	73.1	86.9	79.4
	Frame based	**88.9**	**92.1**	**90.5**

The best results of AT&T's submissions in different categories are shown in Table IV. For example, run 8 achieves the best overall result, and run 9 achieved the best frame based gradual detection result. The performance of cut is a bit lower than what we expected since better results were achieved on the TRECVID 2005 and 2004 datasets. Gradual transition detectors provide good performance, which enabled the AT&T system to be one of the top contenders. In terms of F-measure, the SVM based dissolve verification boosts the overall performance by 2.5% and the gradual transition performance by 3.4%, which is significant.

The frame based gradual transition performance of AT&T system achieves the best among all the participants, which means that the proposed gradual transition (mainly the dissolve) boundary location approaches are very effective.

III. SEARCH

3.1 Overview

Automated search is an important first line approach for the exploration of complex multi-modal environments. The TRECVID environment provides a formal to evaluate complex query topics on a large database of multilingual video sources. In our experiments, we focused on text-search techniques and evaluated a system that automatically performs query expansion with named entities (NE's); the addition of new proper names of people, organizations, and locations to an initial query topic. Our results show promise but need additional investigation before being deployed in a general framework. While the work in this section was developed at AT&T, it is purely exploratory research for the TRECVID evaluation and may or may not be included in future revisions of the MIRACLE framework.

3.1.1 Related Work

The information retrieval community at large has explored both query expansion and named entity detection independently. Most mature text search engines allow a user to perform query expansion automatically with a method called pseudo relevance feedback (PRF). With PRF, the most frequent words from a query's top scoring documents are identified and added to the initial query topic and then the revised query is evaluated again. Named entity detectors have also been studied for quite some time, with state-of-the-art detectors achieving accuracy above 90% for people, organizations, and location.

In one of the first works that named entities are employed to refine search results, [20] used named entities to re-rank text results by calculating the intersection of search results and a corpus of news articles published in the same time period. Other participants of TRECVID in the manual search category allowed operators to append named entities directly to the initial query to improve search results with domain specific information. In our approach, we combined these earlier methods with two important distinctions: we automatically search over an external document set for the expansion terms and we only consider named entities for expansion.

3.2 Feature Extraction

Basic query expansion is a simple task that has three steps: perform a search on a document dataset, collect the most frequent terms from the highest ranking documents, and finally execute another search using the new terms in addition to the initial query. The following sections describe the datasets, necessary pre-processing, and finally our algorithm for query expansion.

3.2.1 Dataset formulation

In this experiment, we have two types of datasets: internal and external. The internal dataset is derived from the ASR scripts of the TRECVID data. We further partition the ASR scripts into story documents by automatically detected story boundaries. The external dataset is derived from data collected in the same time span as the TRECVID training (October 30, 2004 - December 1, 2004) and testing (November 2, 2005 - December 30, 2005) partitions. No manual annotation was performed for either data set.

3.2.2 External data

For this task, external documents were collected from two sources: MIRACLE and NewsBlaster. The first data source was AT&T' MIRACLE system, which captures several broadcast programs of different genres (news, entertainment, sports, etc.) and performs a complete multimedia indexing on those programs. The most relevant component here is the closed caption (CC) with ASR time alignment. A second data source was [12], a system that collects online news from hundreds of sources and summarizes the content of those sources. This data set is much larger in magnitude because hundreds of unique news articles were collected every day. Table V summarizes a few attributes of each document source.

Table V. Summary of external data sources for QE search task

	MIRACLE	**NewsBlaster**
# train	311 programs	54499 articles
# test	339 programs	302624 articles
Data type	Aligned ASR+CC from recorded broadcast television	Text articles from online news sources
Frequent sources	Nightly News, World News Tonight	washingtonpost.com, dallasnews.com

The motivation for collecting two distinct data sources was to test how much the original medium (broadcast television vs. online news) would affect the performance of expansion. Our intuition was that the NewsBlaster set may offer a richer set of NE expansion choices, but perhaps documents in the MIRACLE set would have much higher relevance because they came from the same domain as the TRECVID data and have a much smaller vocabulary.

3.2.3 Story segmentation

We employed a method developed in [18] for automatic story segmentation, developed by Columbia University. This method uses a probabilistic framework to learn and then identify story boundaries in each TRECVID broadcasts. Both visual and prosodic features are used in the boundary detection process, but we refer to the original paper for more detail. The actual model used in this task was trained labeled boundaries on the TRECVID2005 training set (a subset of the TRECVID2006 training set) and evaluated on all other broadcasts in the TRECVID2006 set. These boundaries were donated to the TREC community at large, so we feel that they are acceptable for a baseline of comparison.

3.3 Document Preprocessing

After the search documents have been created, either from the raw data collection in section 3.2.2 or by automatic story segmentation in section 3.2.3, they are processed for named entities. We used the named entity extraction and co-reference tracking routines included in the open source tool *lingpipe* [19]. The NE model was trained on the MUC6 English corpus. This software choice was important because it provided co-reference identification that was used to further link the found NE's.

3.3.1 Enhanced entities

We refer to enhanced entities as NE's that have additional information like a person's first and last name, a formal or role title (*Mr., President, Senator, etc.*), or a regional location context for people and organizations. With this additional information, we hope to disambiguate NE's across documents and even across data sets. The rules below describe how enhanced entities are constructed from the tagged output of this tool.

- For people and organizations,
  1. Preceding word is of type *NN*, append it to the role (i.e. *Mayor Bloomberg*).
  2. More than one NE in a sentence and the preceding word is of type *NP*, set it as the regional location for this person or organization (i.e. *NYC Mayor Bloomberg*)
  3. More than one NE in a sentence and the word in two positions prior is of type *NP* or *JJ*, prefix it to the role of this NE (i.e. *Senator Clinton of New York*).
- For locations, if the word immediately before the NE is of type *IN* and there are multiple NE’s in the sentence, then set the regional location attribute to this word (i.e. *in Columbus, Ohio*).

We add both the raw NE and the enhanced NE to our lexicon of NE's. At the time of this writing, we require all fields to match for a lexicon match. In future versions of this system, we would like to employ a more systematic matching of NE's in the lexicon so that the goal of disambiguation achieved across more documents.

3.4 Query Topic Processing

Query topic processing is a very rich subject and has entire evaluations within NIST devoted to it alone; the most popular conferences are TREC-9 and TREC-10 or ACQUIANT. We choose to use a very simple query processing approach that consists of NE and keyword identification. Section 3.5.2 discusses a revision to this system that creates class-dependent models for input queries, but only one, global model was evaluated for the submitted TV06 runs. For any single text query, we evaluate its performance by retrieving its source shots and scoring them with the standard AP metric.

3.4.1 Initial query formation

Initial query formation uses a set of cascading rules that stop upon the first identification of NE's, nouns, or verbs. The refined input query is then stemmed and stop words are pruned. This formulation worked well for prior TREC evaluations but we note that it looses strength as query topics get more qualitative.

3.4.2 Expansion candidates

We define an expansion candidate as the initial query with additional named entities that were automatically detected. An expansion candidate is formulated differently depending on parameters that control the depth and richness of the query executed on the external database. We form a single expansion candidate with the following steps.

- Execute the initial query on an external data set (defined in section 3.2.2)
- Keep only the top *K* scoring documents from the external search.
- Accumulate counts for all named entities found in the external search and order by decreasing count. Append the top *N* entities from this list to the initial query.
- Generate an expansion candidate where only people NE's are kept in the last step, where *O* is *true*.

We created different parameter ranges based on limits on the total documents available and the number of NE's present in either data set. In total, there are 100 different parameter settings that are evaluated for a single initial query; *K* as values (5, 10, 25, 50, 100) and *N* as values (1, 2, 3, 4, 5, 10, 50, 100, 150, 200) and *O* as values (true, false).

3.4.3 Predicting best expansion

Our goal for query expansion is to produce the expansion candidate with the highest AP. The primary constraint in choosing the best expansion was that the system needed to offer a gain on the AP score of the initial query, described in 3.4.1. We evaluated the TRECVID2005 topics on the TRECVID2005 test set as training data and found no consistent parameter setting that worked well for all query topics. For example, on the topics that saw the largest AP increase ("*Find shots of Condoleeza Rice*" at 0.168 and "*Find shots of Omar Karami, the former prime minister of Lebenon*" at 0.046), we used different expansion settings of *K*=5, *N*=200, *O*=false and *K*=5, *N*=10, *O*=false respectively. One way to satisfy the AP gain requirement is to automatically select the best parameter settings from the entire set of expansion candidates.

Related work in [13] predicted the gain in AP by labeling images versus automatically acquiring a new set of images for the task of concept detection. We analyzed this approach and its inspiration, which predicted query difficulty [14], to develop our framework. Using an SVM classifier, we created a regression model using the following 32 features that compare the ranked lists of shots generated by the initial query and an expansion candidate.

- Intersection histogram - calculate the number of intersecting documents in the top 10 results of sub-queries executed for each word in an expansion candidate (see [14] for details),
- Distance statistics - the mean, standard deviation, minimum, and maximum of distances between the initial query and the expansion candidate,
- Pearson correlation coefficient,
- Spearman rank coefficient and Student's-t test,
- Fisher score,
- Average dynamic recall,
- Discount cumulated gain.

3.5 IB Reranking

In a method based on the probabilistic framework from section 3.2.3, we also analyzed results after using Columbia University's information bottleneck (IB) reranking tool in [15]. This method uses probabilistic smoothing over all shots and local kernel estimation from a ranked list to reorder shots within a list. Please refer to the original paper for details on the algorithm itself.

3.6 Result Analysis

We configured our submitted runs in a way such that the components in each run built upon each other; refer to Table VI for specific configurations and names of the submitted runs. Note that for runs S2, S4, S5, S6 logic in the algorithm only selected a query expansion candidate if its AP score was predicted to be better than its corresponding baseline.

- S1: Our baseline text run demonstrates that our text search utility (MySQL full-text search) is quite competitive and sufficient for this application.
- S2: Unfortunately our story baseline run was lackluster when compared to S1. This result was much lower than expected and we are currently investigating the underlying cause.
- S3: Our model predicted better performance for three expansion topics: 178 (Cheney), 181, 182.
- S4: Our model only selected topic 179 (Hussein) for expansion. There is marginal improvement in overall MAP, but almost 13% relative AP gain on this topic.
- S5: Story baseline expanded with NE's from NewsBlaster. This run had lower AP than even S2 because the regression model only correctly predicted AP gain on two of the six topics that were enabled.
- S6: Post-processed S4 with the IB reranking tool (section 3.5). This run demonstrates that cross-modality searching is always better than text search alone. The only AP loss was seen on topics where S4 originally achieved less than 0.01 AP.

Table VI. Description of submitted AT&T search runs for TV06

Run	MAP	Components
S1	0.0331	text (baseline
S2	0.0342	story
S3	0.0330	text + MIRACLE
S4	0.0348	story + MIRACLE
S5	0.0298	story + NewsBlaster
S6	0.0383	story + MIRACLE + IB

The story told by score progression is reassuring; as we add multi-modal analysis (text to story), query expansion, and even IB reranking, there is an apparent gain in scores. Fig. 21 displays the performance of the different runs described above.

Finally, we were surprised to see that the NewsBlaster model performed worse than the baseline when chosen for queries containing named entities. This is contrary to traditional behavior of NE expansion tasks, where named queries and sports queries usually experience the most benefit.

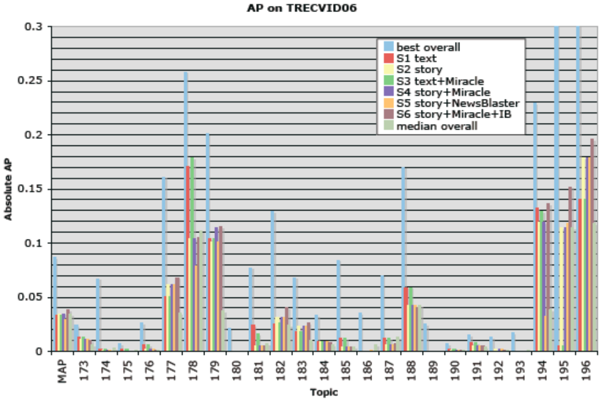


Fig. 21. AP scores for AT&T search submissions.

3.6.1 Prediction accuracy

Prediction accuracy was mixed for the different query topics and story or text baseline searches. While we are still investigating the strength of the regression models, we attribute this volatility to two reasons: data set size differences, more qualitative description in query topics.

First, the training dataset for the regression model (the test partition of TV05) only encompassed one half of a month while the testing dataset (the test partition of TV06) was almost two months. In numerical terms, this equates to 140 programs and 45766 shots for training versus 259 programs and 79484 shots for testing. Not only does this influence the number of results that could be returned in a text search, it dilutes ranking distances and correlation statistics, which are the majority of features for the regression model. We must also concede that training on 24 topics alone (the TV05 topics) is not ideal and could have lead to increased sensitivity to certain types of queries.

Our second intuition for low prediction accuracy was the increase in qualitative requirements fro the TV06 query topics. As described in section 3.4.1, our topic filtering process is quite simple. Unfortunately, logical operators like *or, not, and, except* and numerical operators are not preserved in the formulation of the initial query. For example, "*Find shots of at least one person and at least 10 books*" becomes "*person books*" (topic 190) and the query "*Find shots with a view of one or more tall buildings (more than 4 stories) and the top story visible*" becomes "*view buildings stories story*" (topic 174). Consequently, for topics that had a logical or numerical requirement scored roughly zero AP.

A final point could be argued that the query topics for TV06 contained fewer named entity and sports topics, which seemed to benefit the most from query expansion, but we do not believe that the factor should affect the prediction accuracy of the regression models.

3.6.2 Class dependent models

Recent work by other members in [21] has suggested that a query class dependent strategy may work best for this broad range of topics. If we used a query class strategy with this algorithm, it may also eliminate the need for the regression modeling stage and allow for fixed parameter settings, whose optimal selection has the largest potential for performance degradation.

We are currently investigating this topic and plan to update this report with an in-depth performance analysis soon.

3.6.3 Conclusions

Our experiments demonstrate that named entity query expansion does improve automated search performance within the TRECVID setting. This conclusion is meaningful because it offers a fully automatic method to improve query formulation. At this time, however, the gains from the NE-QE framework are marginal. We plan to investigate how the automatically trained regression models can be improved, why gain prediction failed in some cases, and whether using a strict on/off decision performs better than selecting the expansion candidate with the highest predicted gain.

We also observe that using the IB reranking scheme, S5, (discussed in 3.5) will almost always improve the scores of the expanded query. This is not surprising because the queries themselves are executed only in the text domain and therefore have numerous false positives.

Finally, we hope to analyze performance of this algorithm using general expansion terms; for example, using lexically similar expansions and words within definitions of terms or concepts in a query topic.

IV. BBC RUSHES EXPLOITATION

We combined several tools from existing systems to better organize the BBC data and showcase it in the MIRACLE system. We used our motion-based shot segmentation (not the one used in the SBD task this year) that may help to determine important shot operations like panning and tilting but is semi-resilient to non-steady camera handling. We also used a large set of the LSCOM visual concepts, as trained on the TV05 dataset by Columbia University. On the audio side, we applied an audio sound-type classifier and a speaker segmentation algorithm. Finally, for text search, we have automatically generated a semi-synchronized script from the scanned documents you provided. We will present a demo and prepare background on the techniques and the system for a poster presentation.

4.1 Main Approaches

We applied a multi-modal approach to more intuitively parse and search the BBC data. The BBC data included very little annotation, which is traditionally the strongpoint for text-based search strategies. Additionally, the annotation included was most descriptive of camera actions and events that occurred over a large amount of time, so common approaches that exploit semantic expansion and lexical ontology would be generally weaker for this dataset. In the following sections, we explore user browsing strategies strongly focused on audio and video content, which are easily integrated into MIRACLE's XML-based framework.

4.1.1 Concept Browsing

We used concept models trained at Columbia University that were derived from the LSCOM annotation. These models provided 374 different classifiers based on three core visual features: color moments, Gabor textures, and edge direction histograms. Please refer to Columbia University's TRECVID 2006 report for specific details. Keyframes for the BBC data were derived from the MIRACLE's shot segmentation engine [11]. It should be noted that because of development concerns, this engine is *not* the model that was used in the shot boundary task described in section II.

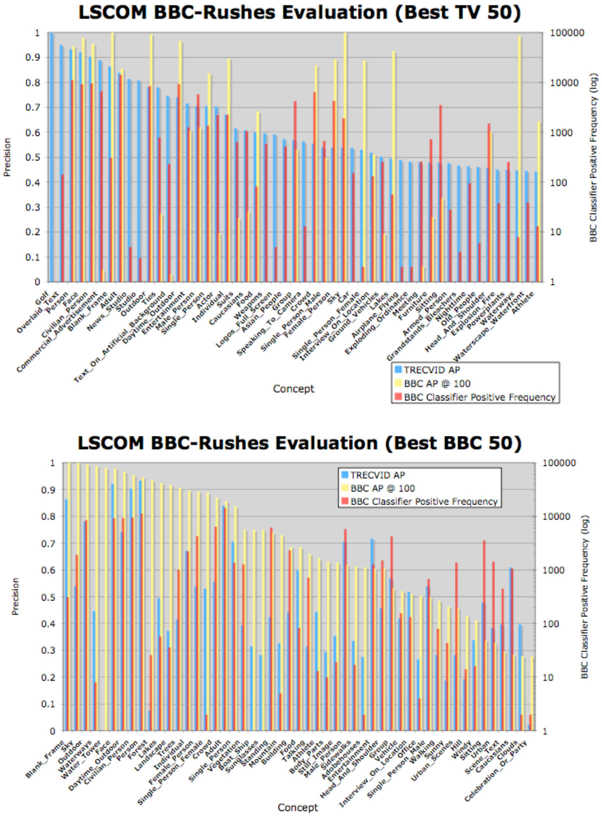


Fig. 22. Concept AP scores on TRECVID2006 and BBC datasets.

Cross-domain application of models is not a trivial task. We applied the concept models described above directly on the keyframes generated for the BBC data. The AP for the best performing BBC and TRECVID concepts are shown in Fig. 22. While we expected different model performance on the BBC and TRECVID datasets, we were pleased to find that some concept models were robust enough to handle the drastically different data. For example, general scene concepts like *outdoor, crowd, person* all worked well in both BBC and TRECVID data sets. However, specific object concepts, like *cars* or *ties* performed very poorly on the BBC dataset.

We attribute the performance loss seen in Fig. 22 between the two datasets to not only a low-level data difference (like color), but also a difference in concept frequencies. The further illustrate this point, we have included the frequency of positive classifications in the BBC data set in both graphs. In future work, we plan to analyze methods to adapt these models to the new BBC data domain with minimal re-labeling of the BBC data.

4.1.2 Visual similarity

We use the features computed for the concept browsing task to compute image similarity within a single BBC video. First, we concatenate all of the visual features to make a single vector of 346 dimensions. Next, distances between all images are computed with a kernelized inner product. The distances are then normalized by the maximum distance between all images in a single video to produce a frame-level similarity between zero and one.

We further leverage frame-level distance to produce a hierarchical scene representation, as seen in Fig. 23. Scenes are constructed by greedily clustering frames by their frame distance until the ratio of the newest frame in a cluster to all other frames in a cluster exceeds some threshold. In future revisions, these automatically formulated scenes can be used as an aide for video summarization.

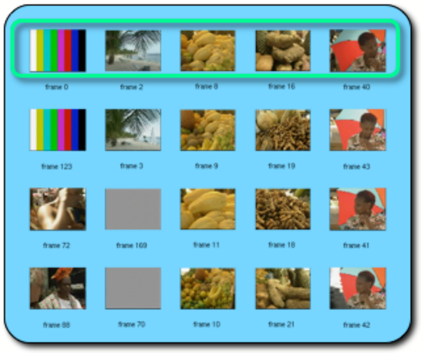


Fig. 23. Scene-level BBC image clustering

4.1.3 Audio sound-type classification

We manually annotated 18 videos from the BBC data. These videos were randomly selected from the development set so as to maximize the diversity of class labels. A total of 15 audio classes were annotated, but we found that the data for most of these classes was too sparse or inconsistent to train accurate sound models. For the actual task and model development only male and female speech were analyzed; all other labels were grouped into a noise class.

We classify the audio sound types at two levels. At the first level, an audio file is classified into speech and non-speech segments. At the second level, speech segments are further classified into male and female speech segments. Although both use the Gaussian mixture model (GMM) classifier, different sets of audio features are adopted for the two tasks.

For speech and non-speech classification, we segment an audio signal into audio clips, which are 3 seconds long on average. Each clip in turn consists of overlapping frames. The features of each audio clip are determined from the sub-features of the associated frames. Each frame is 32 millisecond (ms) long, overlapping with the previous one by 22 ms. Eight features are computed for each frame. They are root mean square volume, zero crossing rate, pitch, frequency centroid, frequency bandwidth, and 3 energy ratios in subbands. We extract 14 features for each audio clip based on frame-level features. The 14 clip-level features are 1) volume standard deviation (VSD), 2) volume dynamic range (VDR), 3) volume undulation (VU), 4) non-silence ratio (NSR), 5) standard deviation of zero crossing rate (ZSTD), 6) 4-Hz modulation energy (4ME), 7) standard deviation of pitch (PSTD), 8) smooth pitch ratio (SPR), LIU et al.: MAJOR CAST DETECTION IN VIDEO USING BOTH SPEAKER AND FACE INFORMATION 7 9) non-pitch ratio (NPR), 10) frequency centroid (FC), 11) frequency bandwidth (BW), 12-14) energy ratio in subbands 1 - 3 (ERSB1, ERSB2, and ERSB3). For detailed description of these features, please refer to [16]. Based on our prior experiments, GMMs with 4 mixtures provide good performance of speech and non-speech classification. In this paper, we assume that the covariance matrix of each Gaussian mixture is diagonal.

For male and female classification, we use Mel-frequency cepstral coefficients (MFCC) [8], because these features have been shown to reflect the speaker characteristics. For each frame, we extract 13 MFCCs, as well as their first and second order temporal delta values. Totally we have 39 features for each frame. Our experiments showed that when the number of GMM mixtures is 2048, the results are reasonably good.

More audio sound types, including noise, music, speech on music, etc., can also be effectively extracted based on the same approach. But given the limit of time and resources, we defer this investigation as the future work.

4.1.4 Speaker segmentation

The audio is first segmented into short segments on phoneme level, where the duration of each segment is in the range of 200 ms to 1 second. Similar to the speaker gender classification, we adopt the same 39 MFCC features, and each speaker is modeled using a GMM model. We employ Bayesian Information Criteria (BIC) [17] to measure how the speaker models fit the data. Fig. 24 shows the diagram of the developed algorithm.

The iterative speaker segmentation contains a loop of speaker splitting procedure. In each iteration, the algorithm first increases the number of speakers (NS), and then evaluates all possible splits: splitting speaker i (i = 1, ..., NS-1) into speakers i and NS. For each possible split, the speaker split procedure is applied, and corresponding BIC value (BIC_{NS}) and speaker labels (L_{NS}) are computed. Among the NS-1 ways of splitting, the one with maximum BIC value is chosen, and its BIC value and speaker labels are kept as the overall BIC value (BIC_{NS}) and speaker labels (L_{NS}) for the current iteration. The iteration terminates when BIC value no longer increases.

The speaker label refinement is also an iterative procedure. For each iteration, a set of GMM's are built for all speakers based on current segment labels. Then all segments are relabeled using the maximum likelihood method based on current speaker models. If the speaker labels converge or the number of iteration reaches a preset value, the refinement iteration stops. Otherwise, a new iteration starts.

The post-processing step merges adjacent segments with the same speaker labels, and smoothes the segments that are too short, e.g., less than 300 ms. Short segments are merged into the longer neighboring segments.

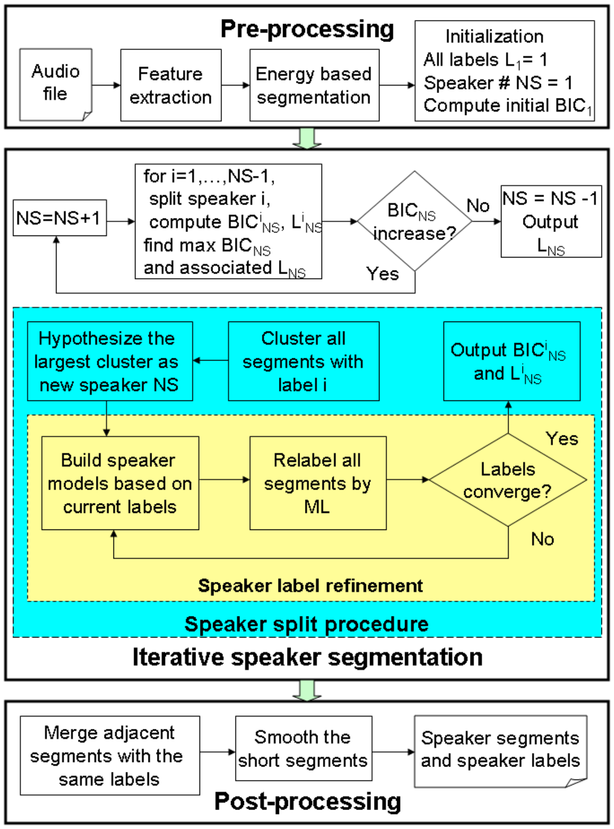

Pre-processing
Audio file
Feature extraction
Energy based segmentation
Initialization
All labels L_1= 1
Speaker # NS = 1
Compute initial BIC_1
NS=NS+1
for i=1,…,NS-1,
split speaker i,
compute BIC^i_{NS}, L^i_{NS}
find max BIC_{NS}
and associated L_{NS}
BIC_{NS} increase?
No
Yes
NS = NS -1
Output L_{NS}
Hypothesize the largest cluster as new speaker NS
Cluster all segments with label i
Output BIC^i_{NS} and L^i_{NS}
Build speaker models based on current labels
Relabel all segments by ML
Labels converge?
Yes
No
Speaker label refinement
Speaker split procedure
Iterative speaker segmentation
Merge adjacent segments with the same labels
Smooth the short segments
Speaker segments and speaker labels
Post-processing

Fig. 24. Speaker segmentation diagram

4.1.5 Text annotation processing

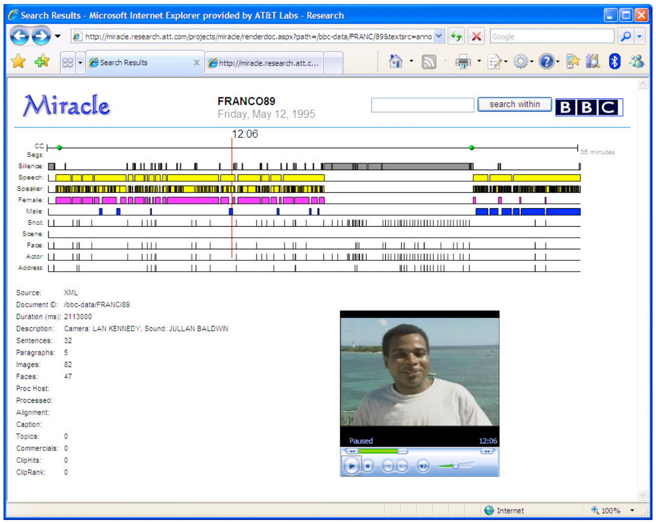
The text data for the BBC rushes passed through several stages of potential degradation. First, they were manual notes that were scanned and passed through an OCR system. After this processing, we analyzed the annotations to create a basic organization of topics and annotation sentences for every line in the OCR output while using manually defined rules to correct common OCR failures and non-English characters.

4.2 Results

Figures 25 and 26 show two interfaces we developed for browsing the BBC rushes data by the extracted content events.

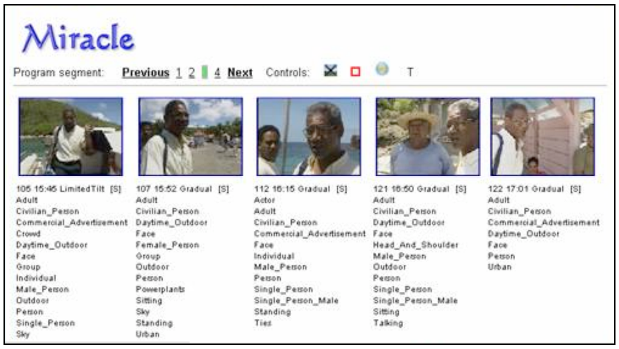
The video data is played back at the top of the interface, and the extracted content events, including speech segments, male and female speaker segments, speaker segments, shot and scene boundaries, as well as face shots are plotted in different color underneath. From this interface, the user is able to efficiently understand the embedded content, and easily jump to the points of interest to see more detailed information.

The second interface (in Fig. 26) demonstrates a frame-level view of a video clip, which gives a detailed description of the detected concepts and, frame time, and detected camera motion.


Search Results - Microsoft Internet Explorer provided by AT&T Labs - Research
http://miracle.research.att.com/projects/miracle/renderdoc.aspx?path=/bbc-data/FRANC/89&textsrc=anno
Search Results
http://miracle.research.att.c...
Miracle
FRANC089
Friday, May 12, 1995
search within
BBC
12:06
CC
Segs
Silence
Speech:
Speaker:
Female:
Male:
Shot:
Scene:
Face:
Actor:
Address:
35 minutes
Paused
12:06
Internet
100%

Source: XML
Document ID: /bbc-data/FRANC/89
Duration (ms): 2113000
Description: Camera: LAN KENNEDY, Sound: JULLAN BALDWIN
Sentences: 32
Paragraphs: 5
Images: 82
Faces: 47
Proc Host:
Processed:
Alignment:
Caption:
Topics: 0
Commercials: 0
ClipHits: 0
ClipRank: 0

Fig. 25. Interface for browsing the BBC content



106 15:46 LimitedTilt [S]
Adult
Civilian_Person
Commercial_Advertisement
Crowd
Daytime_Outdoor
Face
Group
Individual
Male_Person
Outdoor
Person
Single_Person
Sky

107 15:52 Gradual [S]
Adult
Civilian_Person
Daytime_Outdoor
Face
Female_Person
Group
Outdoor
Person
Powerplants
Sitting
Sky
Standing
Urban

112 16:15 Gradual [S]
Actor
Adult
Civilian_Person
Commercial_Advertisement
Face
Individual
Male_Person
Person
Single_Person
Single_Person_Male
Standing
Ties

121 16:50 Gradual [S]
Adult
Civilian_Person
Daytime_Outdoor
Face
Head_And_Shoulder
Male_Person
Outdoor
Person
Single_Person
Single_Person_Male
Sitting
Talking

122 17:01 Gradual [S]
Adult
Civilian_Person
Commercial_Advertisement
Daytime_Outdoor
Face
Person
Urban

Fig. 26. Frame-level interface for BBC content

V. CONCLUSIONS

In this paper, we reported the AT&T system that we developed for NIST TRECVID 2006 evaluation. AT&T participated in three tasks: shot boundary determination, search, and rushes exploitation. In this paper, we described our shot boundary determination system in detail, and briefly introduced our work on the other two tasks. The evaluation results show that the SBD algorithm we proposed is effective and promising.

VI. 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] MPEG Software Simulation Group (MSSG), http://www.mpeg.org/MPEG/MSSG/

[6] J. Hopcroft and J. D. Ullman, “Introduction to Automata Theory, Languages, and Computation,” Addison Wesley, 1979.

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

[8] L. Rabiner and B. Juang, “Fundamentals of Speech Recognition,” Prentice Hall PTR, 1993.

[9] J. Hafner, H. S. Sawhney, W. Equits, M. Flickner and W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 7, July 1995.

[10] V. N. Vapnik, “Statistical Learning Theory,” John Wiley & Sons, 1998.

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

[12] K. McKeown, R. Barzilay, J. Chen, D. Elson, J. Klavans D. Evans, A. Nenkova, B. Schiffman, and S. Sigelman, “Columbia’s newsblaster: New features and future directions (demo),” *NAACL-HLT*, 2003.

[13] L. Kennedy, S. Chang, and I. Kozintsev, “To search or to label? Predicting the performance of search-based automatic image classifiers,” *Multimedia Information Retrieval Workshop (MIR)*, Santa Barbara, CA, USA, 2006.

[14] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow, “Learning to estimate query difficulty: including applications to missing content detection and distributed information,” *SIGIR*, pp. 512-519, New York, NY, 2005.

[15] W. Hsu and S. Chang, “Visual cue cluster construction via information bottleneck principle and kernel density estimation,” *International Conference on Content-Based Image and Video Retrieval (CIVR)*, Singapore, November 2005.

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

[17] S. S. Chen and P. S. Gopalakrishnan, “Speaker, environment and channel change detection and clustering via the Bayesian information criterion,” *DARPA Broadcast News Transcription and Understanding Workshop*, Landsdowne, VA, 1998.

[18] W. Hsu, L. Kennedy, S. Chang, M. Franz, and J. Smith. “Columbia-IBM news video story segmentation in TRECVID” 2004. Advent technical report #207-2005-3, Columbia University, 2005.

[19] “LingPipe Home". http://www.alias-i.com/lingpipe/index.html, October 2006.

[20] T. Chua, S. Neo, H. Goh, M. Zhao, Y. Xiao, G. Wang, S. Gao, K. Chen, Q. Sun, and T. Qi. “Trecvid 2005 by NUS PRIS”. *TREC Video Retrieval Evaluation Online Proceedings*, 2005.

[21] L. Kennedy, P. Natsev, and S. Chang. “Automatic discovery of query class dependent models for multimodal search”. *ACM Multimedia Conference*, Singapore, November 2005.