NTU TRECVID-2007 Fast Rushes Summarization System

Chen-Ming Pan Yung-Yu Chuang Winston H. Hsu
National Taiwan University
{pansack, cyy, winston}@cmlab.csie.ntu.edu.tw

ABSTRACT

Rushes are the raw materials used to produce a video. They often contain redundant and repetitive contents. Rushes summarization aims to provide a quick overview for a rushes video. As part of TRECVID 2007, NIST initiates a rushes summarization task. This paper reports on the design of NTU rushes summarization system for this task. Our system consists of three components, shot segmentation, redundant shot detection and summary creation. To tackle the bulky rushes, we focus on efficient but effective feature representations (local color histograms and compressed-domain motion vectors) and summarization methods. In addition, we proposed a novel approach to detect clapper shots which are not only relevant to concise summarizes but also essential for indexing numerous camera takes in the rushes. Even practically efficient and requiring only 40% of the video time for computation, the proposed system achieved satisfying results in TRECVID 2007 rushes summarization task.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Information Storage and Retrieval – Content Analysis and Indexing.

General Terms

Algorithm, Design.

Keywords

Rushes summarization, Shot detection, Clapper board.

1. INTRODUCTION

Rushes are the raw materials used to produce a video. Because of errors in shooting (e.g. an actor gets his lines wrong or a plane flies over), search for better quality and other reasons, many takes for the same scene are often made. In addition, there are long segments of the same scene with fixed or barely-moving cameras [1]. Hence, rushes often contain redundant and repetitive content. As part of TRECVID 2007 effort, NIST initiates a task of making summarization for rushes so that users can understand and utilize rushes videos more easily. Details for this task can be found in the overview paper of the workshop [4].

Our goal is to be able to create summary for rushes videos efficiently. To reach this goal, instead of exploring time-consuming high-level features, we investigate and leverage efficiently computable feature representations and effective methods for rushes summarization. Specifically, the proposed system benefits from local color histograms computed from frames and motion vectors extracted directly from compressed domain. These two features are used at various stages of our system. Local color histograms are used for shot segmentation, shot clustering and junk shot detection. Motion vectors are used for shot segmentation, representa-

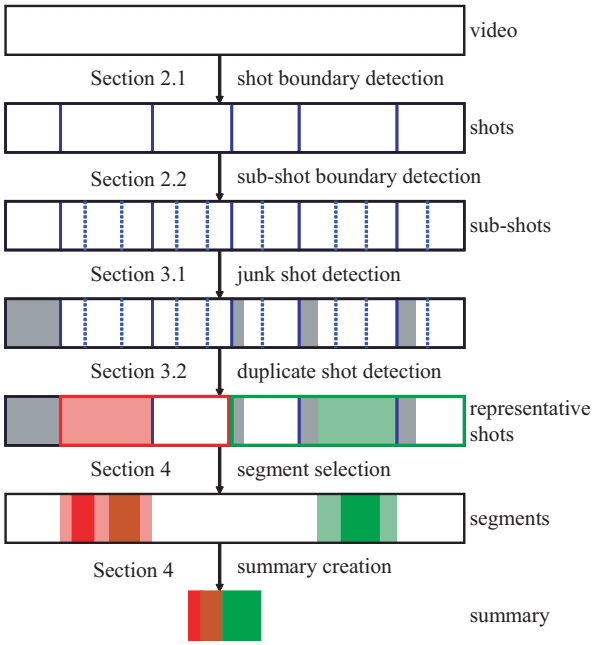
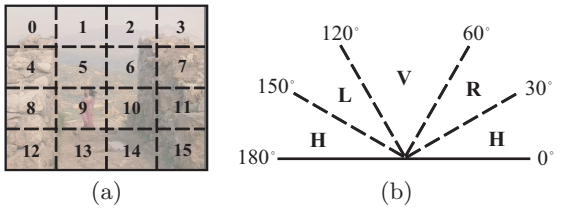


Figure 1: Overview of our rushes summarization system. Our system consists of three components, shot segmentation (shot and sub-shot boundary detection), redundant shot detection (junk and duplicate shot detection) and summary creation (segment selection and summary creation).

TVS'07, September 28, 2007, Augsburg, Bavaria, Germany.



(a) (b)

Figure 2: Block layout and line categories. On the left is the layout of the local color histogram for a frame (a). On the right, we show four line categories for clapper shot detection (b).

tive segment selection and clapper shot detection. We found that local color histograms are useful for comparing the similarity between frames and motions are often good indicators for important shots. As a result, the proposed system achieved satisfying results in the rushes summarization task while only requiring 40% of the video time for computation.

Figure 1 gives an overview for our rushes summarization system. The input video is first segmented into shots. Shot boundaries are detected by comparing the local histograms of neighboring frames (Section 2.1). A shot is further divided into sub-shots to ensure visual similarity between frames within a sub-shot (Section 2.2). Our system then attempts to remove redundant shots. There are two types of redundant shots, junk shots (Section 3.1) and duplicated shots (Section 3.2). Junk shots includes short shots, clapper shots and shots with color bars or single colors. Because many takes of the same scene can be taken in a rushes video, we treat those retaken shots as duplicate shots. A clustering algorithm is used to group repetitive shots together and a representative shot is selected for each group. For each representative shot, we selected multiple segments on basis of motion content. The selected segments are then squeezed into a video summary whose duration is 4% of the original video's duration according to their importance (Section 4).

In the following three sections, we will describe the components of shot segmentation, redundant shot detection and summary creation. These components often involve selection of parameters. These parameters are selected empirically from training rushes videos provided by NIST and fixed during summary creation for test videos. To evaluate the effectiveness of these components, in the following sections, we use five test sequences, MRS044500.mpg, MRS145918.mpg, MRS157444.mpg, MRS157464.mpg and MRS157475.mpg, to report performance. We used only five sequences because of limited time. These five sequences are randomly selected from 42 test videos.

2. SHOT SEGMENTATION

As the first step, our system segments the input video into shots. For example, in Figure 1, the video is divided into six shots. Each shot is further divided into several visually similar units that we called sub-shots. Sub-shots are used mainly for shot clustering. We will discuss the reasons for adding the sub-shot level in Section 2.2.

2.1 Shot boundary detection

A shot is a sequence of frames continuously taken from a camera. Therefore, adjacent frames of the same shot should exhibit strong temporal continuity. Discontinuity happens at shot transitions where content changes. There are basically two types of shot transitions, abrupt and gradual [2]. Abrupt change is often easier to detect. Hence, most shot boundary detection algorithms have been proposed to handle gradual transitions [3, 2]. However, since rushes videos are unedited, they often do not have gradual transitions. Thus, simple approaches can obtain pretty good results for our application. Here, we adapt a histogram-based approach. Our system calculates a local color histogram which divides a frame into 4×4 blocks using 6-bit RGB color code (2 bits for each component). The block layout is shown in Figure 2(a). We use H_t^k to denote the local color histogram for the k-th block of frame t, where $k = 0..15$. Thus, $H_t^k[i]$ represent the value of the i-th bin of the histogram, where $i = 0..63$. We have tried other color spaces such as HSV but their performances are not better than RGB.

To compute the distance between two histograms, we use the χ^2 test because, as suggested by Sethi and Patel [5], it generally has better performance than other measures. The χ^2 difference $D_{\chi^2}(H, G)$ between two histograms H and G is defined as

$$D_{\chi^2}(H,G) = \sum_{i=0}^{63} \begin{cases} \frac{(H[i]-G[i])^2}{\max(H[i],G[i])} & \text{if } \max(H[i],G[i]) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the difference χ_k^2 between the k-th blocks of frames t and $t+1$ can be naturally defined as [3]

$$\chi_k^2 = D_{\chi^2}(H_t^k, H_{t+1}^k).$$

Next, we sort these 16 χ_k^2 values into an ascending order. Let k' denote the block index after sorting so that $\chi_{k'}^2 \leq \chi_{k'+1}^2$. For each pair of consecutive frames t and $t+1$, we obtain their sorted block differences, $\chi_{k'}^2$. Whenever the sum of the middle eight of these 16 values exceeds a pre-defined threshold ϵ_{cut}, i.e. $\sum_{k'=4}^{11} \chi_{k'}^2 > \epsilon_{\text{cut}}$, we claim that there is a cut between frames t and $t+1$.

One thing to note is that Lupatini *et al.* suggested to use the lowest eight χ^2 values to avoid false alarms due to large object motion between frames. Large object motions could result in very different color distributions for the blocks affected by the object motion. However, as long as the object occupy less than half of the frame and the camera does not change dramatically, the lower part of those χ^2 values essentially captures the color distributions of the background, which should be still similar within a shot. It is why lower portion was recommended. However, in rushes, we have observed that using lower portion results in a lower recall for cut detection. It is because rushes have many retaken shots with similar scene content. These shots can not be detected if the lower portion is used. We tried three options of using lower, middle and higher portion of these 16 values. Table 1 summarizes the performance of using these options. Hence, instead of using the lower part, we used the middle eight values for better compromise between precision and recall.

Because we use the middle part instead of the lower part, it increases the chance of false alarms when object motion is large. Thus, it has lower precision than using lower-portion option. To overcome this problem, we add an additional stage to test whether the detected cut is due to large object motion. To measure object motion for a frame t, we extract motion vectors directly from the compressed MPEG stream

	precision	recall	F_2-measure
lower 8	0.702	0.751	0.734
middle 8	0.560	0.893	0.745
higher 8	0.267	0.957	0.514
middle 8+motion	0.802	0.813	0.809

Table 1: Comparisons for the performance of shot boundary detection by summing the lower, middle and higher portions of the 16 χ^2 differences.

and add up the magnitude of all motion vectors to form a single motion measure, M_t. Although motion vectors are optimized for compression, not directly for depicting motion, we have found that they give a reliable indication for the degree of motion*. Thus, if the detected cut frame t whose motion measure M_t is larger than a threshold ϵ_{motion}, we treat the cut as a false alarm and throw it away because it was detected as a cut likely for large object motion. By doing so, we have obtained a better compromise between precision and recall as shown in the last row in Table 1.

The above scheme can only detect abrupt transitions. However, there are some shots with gradual transitions to gradually become dark or bright. To handle these shots, we simply add a cut right before single-color frames that are detected by the method in Section 3.1.

2.2 Sub-shot segmentation

In rushes, a shot represents one take to a scene. Later in Section 3.2, we have to compare whether two shots are taken for the same scene. This can be tricky because shots of the same scene do not necessarily have similar durations. Some shots could be cut very short because of early shooting errors. Thus, we divide shots further into smaller units, sub-shots, for more reliable shot comparison.

For this purpose, we want that all frames of a sub-shot have similar visual content. To reach this goal, we start by choosing the first frame of the shot as the base frame, say frame b. We compare frames sequentially until some frame, say frame c, is different enough from the base frame. The frames from b to $c-1$ then form a sub-shot and frame c are used as the next base frame. We repeat this process until all frames of the shot have been processed. Specifically, assume that frame b is the base frame and the current frame is frame c. We conclude the current sub-shot and use frame c as the new base frame if

$$\sum_{k'=0}^{7} D_{\chi^2}(H_b^{k'}, H_c^{k'}) > \epsilon_{\text{subshot}}.$$

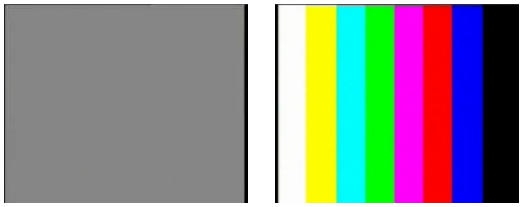
Notice that we use the lower portion for sub-shot segmentation because we want to concentrate on the scene itself and minimize the impact of object motion.

Since all frames of a sub-shot s have similar visual content, we use the average local histogram H_s^k as a representative feature for the sub-shot s,

$$H_s^k[i] = \frac{1}{|s|} \sum_{t \in s} H_t^k[i].$$

Sub-shots are used as the units of comparison for clustering to avoid the issue of various durations of retaken shots.

*An I-frame's motion measure is averaged from neighboring P-frames.



(a) single color frame (b) color bar frame

Figure 3: Single-color and color-bar frames.

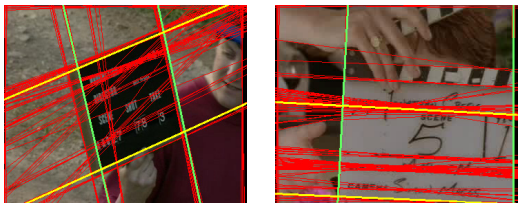


Figure 4: Examples of clapper shots and clapper board detection. Red lines are detected by Hough transform. Yellow and green lines are the hypothesis that validates existence of clapper boards.

3. REDUNDANT SHOT REMOVAL

There are essentially two types of redundant shots, junk shots and duplicate shots. Junk shots include short shots, single color shots, color bar shots and clapper shots. Duplicate shots are the repetitive shots in rushes. We will discuss junk shot detection in Section 3.1 and duplicate shot detection in Section 3.2. In Figure 1, gray blocks represent detected junk shots. For that example, one shot is completely removed from the six shots detected from Section 2. The remaining five shots are then clustered into two groups shown in red and green individually in Figure 1. A representative shot (shaded in Figure 1) is then selected for each group to remove duplicate shots.

3.1 Junk shot detection

Detecting short shots is simple. We simply remove shots whose durations are less than 25 frames. In the following, we will discuss methods for detecting single color shots (e.g. Figure 3(a)), color bar shots (e.g. Figure 3(b)) and clapper shots (e.g. Figure 4). For the first two categories, we use the average local color histogram H_s^k for each sub-shot s, extracted in Section 2.2, to judge whether a sub-shot is a single color shot or a color bar shot.

Single color shots. A sub-shot s is a single color shot if there is a dominant color in its global color histogram H_s, where $H_s[i] = \frac{1}{16}\sum_{k=0}^{15} H_s^k[i]$. A sub-shot s is a single-color shot if following is true,

$$\max_i H_s[i] > \epsilon_{\text{single}}.$$

Color bar shots. Since all color bars are vertical, the color histograms for the blocks of the same column should all be similar†. Thus, we calculate twelve χ^2 histogram differences between any two neighboring blocks on each column, $D_{\chi^2}(H_s^k, H_s^{k+4})$, where $k = 0..11$. If at least ten of these

†Horizontal color bars can be detected in a similar way.

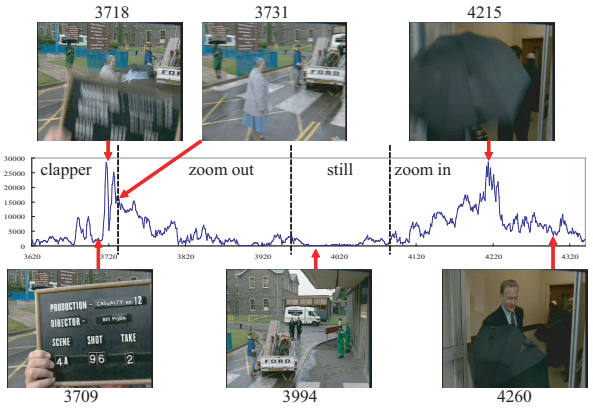


Figure 5: Motion graph corresponding to a portion of MRS044500.mpg. This example shows the motion of a clapper board in action. It also demonstrates that motion measures directly derived from motion vectors are good indicators for motion within frames.

twelve values are smaller than $\epsilon_{\text{colorbar}}$, then we claim that sub-shot s is a color bar shot.

Clapper shots. The detection of clapper shots is more complex because of the great variety of visual appearances of clapper boards. The clapper boards differ mainly in colors and sometimes in shapes as well. At times, even gestures or clothes are used as clapper boards. However, rectangular clapper boards still represent the majority of clapper shots. Hence, we only focus on the detection of regular rectangular clapper boards here.

We have observed two common scenarios of using clapper boards. In the first scenario, the clapper board is moved into the scene at the very beginning of a shot, held for a while, and then moved out of the scene. In the second one, the clapper board is already in the scene at the beginning of a shot. It is kept still for a while and then moved out of the scene. In either scenario, the clapper board has to be pulled out. This often causes very high motion measure especially because the board is very close to the camera. Hence, we first find top three motion peaks of the first 10 seconds of each shot. It is not a good idea to detect clapper board from frames with high motion measure because the board is often blurred due to its high motion. Instead, we look at the low-motion bottoms in front of the high-motion peaks in the motion graph. These low-motion bottoms often represent the moments when the board is held still. It is easier to detect clapper boards at these frames.

Figure 5 demonstrates a typical scenario of using a clapper board. The board is there at the beginning of the shot. It was adjusted slightly around frame 3680, causing a couple of motion peaks. Finally, at frame 3718, the board was pulled out of the scene, causing a very high motion peak in the motion graph. Instead of checking frames with high motions, we find a frame at low-motion bottom in front of the peak, frame 3709, and detect the clapper board at this frame. Figure 5 also shows that the motion measure is a good indicator for the degree of motion for a frame.

The basic idea for clapper board detection is to detect lines for a frame and test all hypothesis of quadrangles formed by detected lines to see whether the quadrangle is large enough and there is a dominant color within the hypothesized area. If there is at least one hypothesis satisfying these constraints, we claim that this frame contains a clapper board. As mentioned earlier, we only perform this expensive operation for at most three frames per shot.

For each candidate frame to check, we apply the Hough Transform to detect lines. We classify detected lines into 4 groups according to their slopes, H (horizontal, $0° \sim 30°$ and $150° \sim 180°$), V (vertical, $60° \sim 120°$), R (right, $30° \sim 60°$) and L (left, $120° \sim 150°$), as shown in Figure 2(b). To form a quadrangle hypothesis, we pick two pairs of lines using one of two possibilities: (1) one pair from group H and another from V or (2) one pair from group L and another from R. We test all possible hypothesis. If any of them satisfies the following two constraints, we claim that there is a clapper board in the frame. First, the area of the hypothesized quadrangle should be larger than ϵ_{area} (5,000 for current implementation) pixels. Second, there is a dominant color (usually black or white) within the quadrangle. To detect the (clapper) dominant color in an unsupervised manner, we adopted RANSAC algorithm. We randomly sample several colors within the quadrangle and calculate their average color. We then check how many pixels within the quadrangle agree with the average color. If there are over 70% of pixels which agree, there is a dominant color.

If we find that the frame t contains a clapper board, we remove the segment from the first frame of the shot to the frame $t + 50$[‡]. Here, we use 50 because we have observed the time that the board disappears after being held still usually doesn't exceed two seconds. Figure 4 shows examples of clapper board detection. The red lines are detected lines. The lines which form the hypothesis indicating the existence of a clapper board are shown in yellow and green. Notice that the hypothesis does not necessarily find the true boundary of the clapper board since the procedure returns immediately once we have found a valid hypothesis. The precision and recall for clapper shot detection are 0.68 and 0.61 respectively. The main culprit for failures is the line detection procedure we adapted from OpenCV.

3.2 Duplicate shot detection

At this point, each shot consists of several sub-shots and each sub-shot s is represented by its average local histogram H_s^k. We then group shots using a hierarchical agglomerative clustering algorithm. Initially, each shot forms a cluster itself and includes all its sub-shots into its cluster. Clusters with a minimal distance are then merged until all clusters are already well separated. We use the single-linkage algorithm, meaning the distance between two clusters is defined as the minimal distance from any member of one cluster to any member of the other cluster. Here, members involved in clustering are sub-shots. Thus, we only have to define the distance between two sub-shots for clustering.

The distance, $D_{vis}(s, s')$, between two sub-shots s and s' based on visual content can be computed by averaging χ^2 distances between local histograms,

$$D_{vis}(s, s') = \frac{1}{16} \sum_{k=0}^{15} D_{\chi^2}(H_s^k, H_{s'}^k).$$

In addition to the visual distance, because duplicate shots are often taken sequentially, the temporal distance $D_{tmp}(s, s')$

[‡]The frame rate of the test videos is 25fps.

detection \ truth	same	different	precision
same	142	0	1.000
different	34	1568	0.979
recall	0.807	1.000	

Table 2: Confusion matrix on clustering performance.

between s and s' should also play a role on the distance between sub-shots. Thus, we define the overall distance $D(s, s')$ between two sub-shots s and s' for clustering as

$$D(s, s') = 1 - (1 - D_{vis}(s, s')) \times \exp\left(-\frac{D_{tmp}(s, s')^2}{2\sigma^2}\right),$$

where σ is determined empirically from training videos. The clustering process stops when the distance of any two clusters is larger than $\epsilon_{\text{cluster}}$. Table 2 summarizes the performance of the shot clustering algorithm. Note that we intentionally set a more rigorous $\epsilon_{\text{cluster}}$ to avoid clustering shots of different scenes into the same group. This is good for increasing the inclusion rate but it could lead duplication in the video summary.

After clustering, for each cluster, we pick one shot as its representative shot based on motion measures and durations, and treat others as duplicate shots. The importance Φ_S of a shot S is calculated by summing motion measures of all frames in S, i.e. $\Phi_S = \sum_{t \in S} M_t$. By doing so, a shot with larger motion or a longer duration will have a larger importance value and more likely be selected as the representative shot for the group which it belongs to.

4. SUMMARY CREATION

Segment selection. As the first step for summary creation, we select important segments for each representative shot. Again, we rely on motion measures to make the selection. We classify a frame t as a high-motion frame if its motion measure M_t is higher than ϵ_{motion}. A sliding window of size w moves from the first frame to the last frame of the shot. If the number of high-motion frames within the current window exceeds ϵ_{count}, all frames of the current window are tagged as important frames. Finally, consequent important frames are grouped together to form segments.

Next, we cluster segments of a shot using the same hierarchical agglomerative clustering algorithm described in Section 3.2 and only keep one segment for the segments in the same cluster. This is a remedy for the imperfect shot segmentation results in which around 20% of shot boundaries are not correctly detected. These are mostly caused by repetitive shots which were taken without moving and turning off the camera. By grouping visually similar segments within the same shot, we could avoid duplication even with imperfect shot detection.

Summary creation. Since there is an upper bound T_{limit} for video summary (4% of the original video duration), we have to squeeze the N selected high-motion segments into the summary. For not missing any segment, we assign a minimal base time T_{base} to each segment. The remaining time is allocated to segments based on their importance values. We define the importance Ψ_j for segment j as

$$\Psi_j = \alpha \frac{L_j}{L_{\text{all}}} + (1 - \alpha) \frac{M_j}{M_{\text{all}}},$$

where L_j is the duration of the segment j, $L_{\text{all}} = \sum_j T_j$, $M_j = \frac{1}{|j|} \sum_{\text{frame } t \in j} M_t$, $M_{\text{all}} = \sum_j M_j$. In the current implementation, we set α as 0.5. Finally, we allocate time T_j to the segment j, where

$$T_j = T_{\text{base}} + (T_{\text{limit}} - N \times T_{\text{base}}) \times \Psi_j.$$

If $T_j > L_j$, we just put the whole segment into the summary. Otherwise, we use a sliding window of width T_j to select a portion of segment j, which has the maximal sum of motion measures, and put this portion of segments into the video summary.

5. RESULTS AND DISCUSSIONS

Among all 24 submissions (including two CMU baselines), our system ranks 9th in terms of inclusion rate, 13th in easiness to understand, 9th in content duplication and 7th in summary creation time. Our system always selects high-motion segments into the summary. As a result, our summary is full of short and high-motion segments, which might be harder to understand to the viewers. It is probably why we had a lower rank for easiness to understand. The evaluation of content duplication is mainly about the performance of clustering. Our algorithm seems to work reasonably well. However, it could be further improved by parameter tuning and employing more sophisticated similarity measures.

Overall, our system achieves a reasonable balance between creation time and effectiveness. Our system is fast because it only uses two low-level features and uses them for different purposes throughout the system. Though simple, these features seem quite useful for making effective summaries.

One thing to note is that, in our submission to NIST, we have not included clapper shot detection into our system because it was finished after the submission deadline. Because of clapper board's large motion, clapper shots are likely to be selected into the summary. Thus, our submission summaries contain quite a few clapper shots. It degraded our system's performance. The summary looks better after we have added clapper shot detection into our system.

To advance the robustness and exploit system capabilities, we are devising a more systematic approach to determine parameters used in the proposed system. Because rushes only include ambient sounds, sound might not help for analysis. However, audio might still be helpful for detection. For example, clapper shots can probably be detected more accurately by incorporating sound analysis. Another promising direction to boost (near) duplicate shot detection is to analyze the take number on the clapper board.

6. REFERENCES

[1] `http://www-nlpir.nist.gov/projects/tv2007/`.

[2] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. 16:477–500, 2001.

[3] G. Lupatini, C. Saraceno, and R. Leonardi. Scene break detection: a comparison. In *Proceedings of International Workshop on Research Issues In Data Engineering, Continuous-Media Databases and Applications*, 1998.

[4] P. Over, A. F. Smeaton, and P. Kelly. The TRECVID 2007 BBC rushes summarization evaluation pilot. In *Proceedings of the TRECVID Workshop on Video Summarization (TVS'07)*, pages 1–15, September 2007.

[5] I. K. Sethi and N. V. Patel. Statistical approach to scene change detection. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 329–338, 1995.