Improving Retake Detection by Adding Motion Feature

Hiep Van Hoang[1], Duy-Dinh Le[2], Shin'ichi Satoh[2], and Quang Hong Nguyen[1]

[1] Hanoi University of Science and Technology, No 1 Dai Co Viet, Hanoi City, Vietnam
[2] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

Abstract. Retake detection is useful for many applications of video summarization. It is a challenging task since different takes of the same scene are usually of different lengths; or have been recorded under different environment conditions. A general approach to solve this problem is to decompose the input video sequence into sub-sequences and then group these sub-sequences into clusters. By combining with temporal information, the clustering result is used to find take and scene boundaries. One of the most difficult steps in this approach is to cluster sub-sequences. Most of previous approaches only use one keyframe for representing one sub-sequence and extract features such as color and texture from this keyframe for clustering. We propose another approach to improve the performance by combining the motion feature extracted from each sub-sequence and the features extracted from each representing keyframe. Experiments evaluated on the standard benchmark dataset of TRECVID BBC Rushes 2007 show the effectiveness of the proposed method.

1 Introduction

In film and video production, people usually have to deal with large amount of un-edited or rushes video content since one scene in rushes of video might have been recorded many times for different reasons. Each recording of the scene involves different takes, and only the best take is selected to make the final edited content while the others are removed. The takes that are removed are called redundant takes or retakes of the selected take, and finding the takes that are belong to one scene to group them into one cluster is the problem of retake detection. One motivation of retake detection is it can help editors by automatically finding the best takes to be used. Another motivation is in problem of video summarization, where editors have to make a shorter version of original video.

There are several challenges when solving the problem of retake detection:

- Different takes of the same scene usually have different lengths of time. For example, some shots are disrupted by miscues, bloopers, or various unexpected mistakes.

G. Maino and G.L. Foresti (Eds.): ICIAP 2011, Part II, LNCS 6979, pp. 150–157, 2011.

- Even if two takes whose duration are very similar are still slightly different due to differences in environmental conditions or various differences in the actions of actors. If only several keyframes are used for segment representation, it will be sensitive to the acquiring conditions such as changing light, and changing positions of objects or actors.
- The boundary of retakes and scene formed by these takes is usually unknown.

Bailer et al. [1] proposed a distance measure based on Longest Common Subsequence (LCSS) to determine the similarity of segments in the video. These similar segments were then grouped into one cluster if they belong to one scene. Dumont and Merialdo [2] divided the input video into one-second segments and then used Smith-Waterman algorithm to find all similarity segments in the video. Feng Wang and Chong-Wah Ngo [3] tried to find takes directly by using shot boundary detection combined with speech recognition.

Most of these approaches only focus on matching video sequences in some feature space. In this paper, we focus on another aspect that is how to represent video sequences (i.e. feature representation of video sequences) to improve the matching performance. We propose to use motion feature that was not considered in related work to encode spatio-temporal relation between consecutive frames. By adding the motion feature to existing features such as color and texture, the representation has more discriminative power, resulting better matching performance.

2 Framework Overview

We adopt the general framework proposed in [2] for the problem of retake detection in video (c.f Figure 1). First, the input video is divided in segments by using shot boundary detector [1,3] or sampling every 1-second length [2]. Next, keyframe images are extracted from the segments and then features are extracted from these keyframe images. A clustering method such as k-means or hierarchical clustering is used to group similar segments into clusters. After the clustering step, by using the cluster labels, the input video is represented as a list of labels.

Since one retake might contain several segments, Smith-Waterman algorithm is then applied to find all repeated sub-sequences [2]. This step is considered as a further grouping of segments to form retakes. Finally, scenes that are sets of retakes are formed. We use rand index metric proposed by W. M. Rand [4] to evaluate the results against the groundtruth.

2.1 Feature Extraction

Color Moments (CM)
Color moments have been successfully used in retrieval systems [5] and proved to be efficient and effective in representing color distributions of images [6].

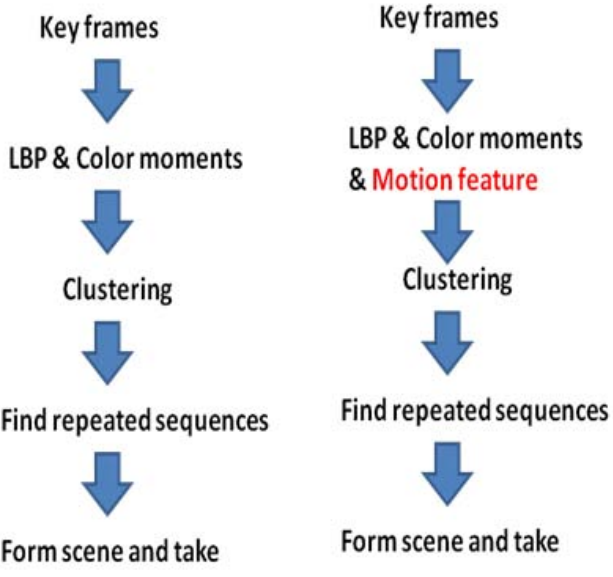

Fig. 1. An overview of a general framework for the problem of retake detection. (left) The framework used in [2]. (right) Our framework improve the performance of retake detection by adding motion feature.

The first order (mean), the second order (variance) and the third order (skewness) color moments are defined as:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij}$$

$$\sigma_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^2)^{\frac{1}{2}}$$

$$s_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^3)^{\frac{1}{3}}$$

where f_{ij} is the value of the i-th color component of the image pixel j, and N is the number of pixels in the image.

Local Binary Patterns (LBP)
The LBP operator proposed by Ojala et al. [7] is a powerful method for texture description. It is invariant with respect to monotonic grey-scale changes, hence no grey-scale normalization needs to be done prior to applying the LBP operator. This operator labels the pixels of an image by thresholding the neighborhoods of each pixel with the center value and considering the result as a binary number.

Motion Feature

Since optical flow is an approximation of image motion based on local derivatives of sequence images. It specifies how much each pixel moves between adjacent frames in the video. If I(x, y, t) is the center in $(m \times m)$ neighbourhood and moves by δx, δy in time δt to I(x + δx, y + δy, t + δt). Since I(x, y, t) and I(x + δx, y + δy, t + δt) are intensity of the same pixel, we have:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{1}$$

Assuming the movement to be small, we can perform a 1^{st} Taylor series expansion about I(x, y, t) as follows:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T.$$

Where H.O.T. are Higher Order Terms, which we assume are small and can be ignored. Following the constrain (1) above, we must have:

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

or

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0$$

which results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \tag{2}$$

Where V_x and V_y are the x and y components of the velocity or optical flow of I(x, y, t) and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x,y,t) in the corresponding direction. We normally write these partial derivatives as: $I_x = \frac{\partial I}{\partial x}$; $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$, finally the equation (2) can be rewritten as

$$(I_x, I_y).(V_x, V_y) = -I_t \tag{3}$$

We cannot compute V_x and V_y directly since (3) is an equation in two unknowns. This is known as the aperture problem of the optical flow algorithms, so we applied Lucas and Kanade method [8] to solve this problem. After the optical flow of two consecutive keyframe images was computed, an orientation histogram of optical flow is built and used as the motion feature.

To discriminate two different keyframe images more accurately, the keyframe images are divided into regions by using a 5×5 grid. The features are extracted for each region and concatenated to form the final feature vector. Then, all features including color moments, local binary patterns, and motion features are concatenated and normalized to a zero mean and unit standard deviation.

2.2 Finding All Repeated Sub-Sequences

We use k-means clustering for grouping segments. After that, the input video is presented as a list of strings $S = s_1 s_2 \dots s_n$ where s_i is the cluster label of the segment represented by keyframe image f_i in the video and n is the number of keyframes extracted from the video. Since the video contains several scenes and each scene is recorded several times (each time is called a take), several sub-sequences of S should be repeated at different positions in the video sequence. Our work is to find these repeated sub-sequences. To do so, we adopt the method described in [2] that uses Smith-Waterman algorithm [9]. This algorithm is based on dynamic programming and is designed to not only compare the global alignment of two sequences of characters (two strings) but also their local alignment. To find all aligned sub-sequence of two sequences, a scoring matrix of $(m+1) \times (n+1)$ is located with each column for each character in the first sequence and each row for each character in the second sequence. Cells of scoring matrices indicate the cost of changing a sub-sequence of the first sequence to a sub-sequence of the second sequence.

Build Scoring Matrix. Given two sequences of characters: $A = a_1 a_2 \dots a_m$, $B = b_1 b_2 \dots b_n$. A $(m+1) \times (n+1)$ scoring matrix H is built as:

$$H(i,0) = 0; 0 \le i \le m$$

$$H(0,j) = 0; 0 \le j \le n$$

$$H(i,j) = max \begin{bmatrix} 0 & \\ H(i-1,j-1) + w(a_i, b_j), & Match(Mismatch) \\ H(i-1,j) + w(a_i, -), & Deletion \\ H(i,j-1) + w(-, b_j), & Insertion \end{bmatrix}$$

$$(1 \le i \le m; 1 \le j \le n)$$

where $w(a_i, b_j)$ is the match/mismatch score, if $a_i = b_j$ then $w(a_i, b_j) = w(match)$ else $w(a_i, b_j) = w(mismatch)$. H(i,j) is the score of similarity between two sequences that end at a_i, b_j respectively. The m is the length of sequence A and the n is the length of sequence B.

Figure 2 below shows an example of the scoring matrix when two sequences $S_1 = ACACACTA$ and $S_2 = AGCACACA$ are compared. In this example, w(match) = +2; w(mismatch) = w(a,-) = w(-,b) = -1.

Trace Back to Find Optimum Local Alignment. To obtain the optimum local alignment of two sequences, we start at the highest value in scoring matrix (i, j) and then trace it back from this position to one of three previous positions $(i-1, j-1)$, $(i-1; j)$, and $(i, j-1)$ that depend on which position has a maximum value. This trace back is repeated until we meet a matrix cell with a zero value. In the example above, the optimum local alignment starts at position (8, 8) and the trace back sequence is (7,7), (7,6), (6,5), (5,4), (4,3), (3,2), (2,1),

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix}$$

Fig. 2. An example of a scoring matrix

(1,1), and (0,0); The final result of two sub-sequences A-CACACTA of sequence 1 and AGCACAC-A of sequence 2 are aligned together with a confident score of 12.

Smith-Waterman Algorithm. We use Smith-Waterman algorithm to find all repeated sub-sequences as follows:

- Input: Given video $S = s_1 s_2 \dots s_n$
- Build scoring matrix H of two sequences: S and itself
- Loop :
  - Find the position (i, j) in which H(i, j) is maximum
  - If H(i, j) $<$ threshold break;
    - Trace back to find optimum aligned sub-sequence and store it (two sub-sequences are aligned together if they are belong to trace back trajectory)
    - Update scoring matrix to find next aligned sub-sequence
- Output: a rank list of aligned sub-sequences (after this, it is called a list of take candidates)

2.3 Forming Takes and Scenes

We have a list of take candidates in this step. Our work now is grouping all takes of the same scene into one group and each group will be a scene. We do the following steps to form a scene and a take.

- Remove all take candidates whose length is too short ($<$ 4 seconds)
- Sort take candidates in time order
- Two take candidates $p_1 = [start_1, end_1]$ and $p_2 = [start_2, end_2]$ are merged into one take $p = [min(start_1\ , start_2), max(end_1, end_2)]$ if

- p_1 and p_2 overlap by time: $p_1 \cap p_2 \neq \phi$, and
- $Length(p_1 \cap p_2) > 0.5 * min(length(p_1, p_2))$

– Two take candidates are grouped into one if they are aligned together.

Figure 3 visualizes our algorithm to form scenes and takes. The first line is the ground truth of the video. There are two scenes in this example that have been visualized in blue and green. The next lines are take candidates that are sorted in time order; two takes that are aligned together have the same color. The solid colored rectangles represent take candidates; the open dotted rectangles represent takes after candidates have been merged.

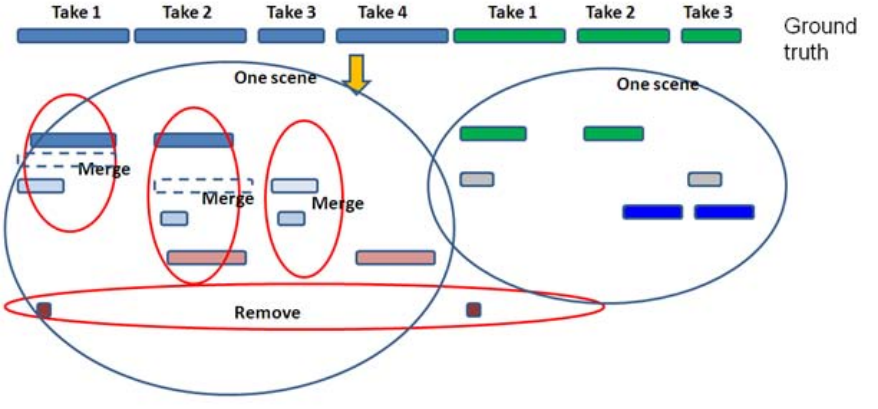

Fig. 3. Illustration on how to form takes and scenes

3 Experiments

We have tested our proposed method on five videos selected randomly from TRECVID BBC Rushes 2007. The groundtruth is provided by NIST that was used for evaluation of the summarization task in TRECVID benchmark 2007. Rand index [4] is used to evaluate the performance of the two methods without and with using motion feature. Since the performance of the methods depends on choosing the best k in k-means clustering, we tried different values $k = 180, 190, 200, 210, 220, 230, 250$ and only the best performance is reported in the comparison table.

As shown in table 1, adding motion feature boosts the detection performance both in scenes and takes. We also found that motion feature does not help improve performance in all situations. The MRS157475 video is an example. The reason is this video was recorded outdoors in high wind conditions. In this case, the motion feature is not useful since it has a lot of noises and current motion feature could not handle.

Table 1. Experimental result on 5 videos of TRECVID 2007. Performance of each method is evaluated using rand index (RI) score. The higher RI score, the better performance.

Videos	**Features**			
	CM&LBP		CM&LBP&Motion	
	Scene RI	Take RI	Scene RI	Take RI
MRS044500	0.58	0.60	0.73	0.73
MS216210	0.69	0.71	0.66	0.70
MRS157475	0.73	0.81	0.70	0.73
MRS025913	0.60	0.63	0.67	0.69
MRS144760	0.62	0.64	0.71	0.72
Average	0.64	0.68	**0.69**	**0.71**

4 Conclusions

Retake detection is a important but challenging task for applications of video summarization. Existing work [2,1] only focuses on how to group sub-sequences into takes and scenes. In order to improve further the performance, we propose to add motion feature in current features such as color and texture used in these existing frameworks. Experimental results on BBC Rushes dataset of TRECVID 2007 show effectiveness of the proposed method.

References

1. Bailer, W., Lee, F., Thallinger, G.: A distance measure for repeated takes of one scene. The Visual Computer: International Journal of Computer Graphics 25, 53–68 (2008)
2. Dumont, E., Merialdo, B.: Rushes video summarization and evaluation. Multimedia Tools and Applications 48, 51–68 (2010)
3. Wang, F., Ngo, C.W.: Rushes video summarization by object and event understanding. In: TVS 2007 Proceedings of the International Workshop on TRECVID Video Summarization, pp. 25–29 (2007)
4. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66, 846–850 (1971)
5. Flickner, M., Sawhney, H.S., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P.: Query by image and video content: The qbic system. IEEE Computer 28, 23–32 (1995)
6. Stricker, M.A., Orengo, M.: Similarity of color images. In: Proc. of SPIE, Storage and Retrieval for Image and Video Databases III, vol. 2420, pp. 381–392 (1995)
7. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. on Pattern Analysis and Machine Intelligence 24, 971–987 (2002)
8. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of Imaging Understanding Workshop, pp. 121–130 (1981)
9. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology 147, 195–197 (1981)