Video Montage - Video Abstraction

I-Ting Fang

1 Introduction

The amount of video has rapidly increased in recent years, and therefore how to extract useful video content and retrieve video from huge database is a crucial issue. Video sharing websites now use metadata to describe videos, while there might be so many scenes contained in single video, the information that tags reveal could be limited. Furthermore, there might be chances that only a small portion in a video that users are interested in, while some useful and relevant information across videos. Considering those situations, it's then important to describe the content directly from videos and be able to collect those users' interested shots. Before the shots collection we need to first know users' preference, and thus an overview of video contents becomes necessary which is the main topic of this project – Video Abstraction, see Figure 1 (a). In fact, above description is just one of applications of video abstraction. The technique can be applied in facilitating browsing video database back and forth between joined interesting videos and original videos, and to save the bandwidth as well.

The relevant researches [1] in video abstraction are mostly done for single video, how to shorten the video while containing as much information as possible and how to extract the highlight of a video. In this project, in the contrast, we try to give a content overview of several videos aiming for providing a novel way to browse, search and create personal videos.

Our system, Figure 1 (b), first describes frames by their color and spatial structure features, and extract shots from videos if they remain consistent for certain amount of time. After representing each shot by three of its frames and rescaling feature coordinates using trained weighting vector, we then separate them into clusters where similar shots are supposed to fall in the same cluster. Since we have no idea what scenes would look like in feature space, and how they would differ from each other, we apply clustering algorithm K-means to data trying to find appropriate boundaries. Also, relying on the strong relation among frames from the same shot, we can automatically decide the best number of clusters. Finally, users will be able to browse clustered shots and select interesting ones. The resulting video is joined by user's preference and along with our smooth shot transition scheme.

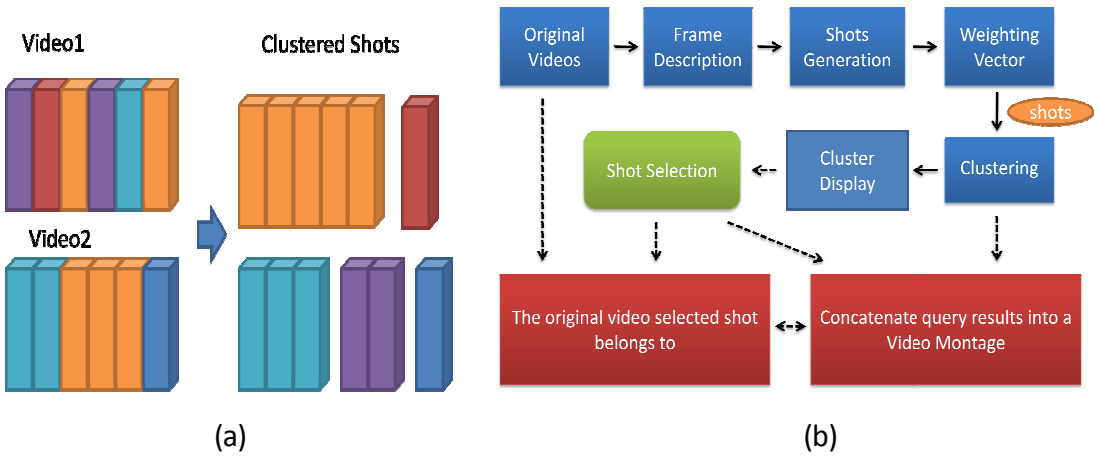


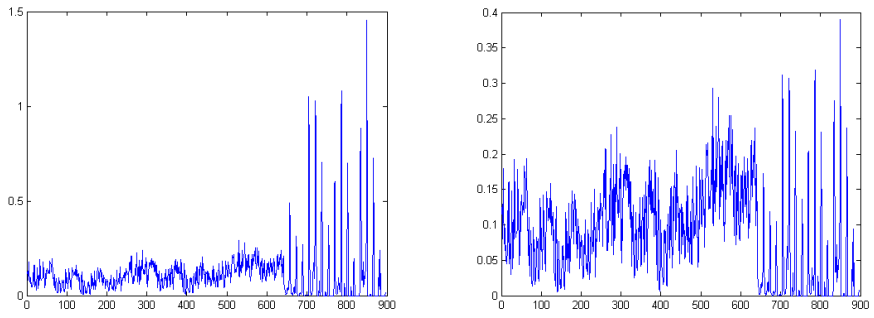
Figure 1. Figure (a) illustrates the concept of the purpose of video abstraction. By showing clustered shots, users can have a big picture of video contents. Figure (b) shows the flow chart of our system. Dash arrows are our future work

2 Feature Selection and Distance Computation

To automatically describe the video content, we need to employ low level features. Different types of features are reported to have good performance on the video abstraction task, such as text, visual, audio and motion, these features are chosen according to the application. In this project, we make our effort on global visual information, HSV color histogram and spatial envelope.

1. *HSV Color Histogram*

   HSV color histogram and other modified versions have been widely used in this area, and could be discriminated in different scenes. To compute the histogram, the HSV space is uniformly quantized into a total of 256 bins which includes 16 levels in H, four levels in S, and four levels in V. In order to make them work with the other feature and be more efficient, the values are rescaled and quantized nonlinearly, giving higher significance to the small values with higher probability. Figure 2 shows that if we only scale the histogram values, the distance of two frames is still dominated mostly by color difference. Thus, we further apply nonlinear quantization to eliminate high variances of color histogram.



(a) (b)

Figure 2. Combined feature vector. First part is spatial feature and second part is color feature. (a) Scaled color histogram. (b) Scaled and nonlinear quantized color histogram.

2. *Spatial Envelope*

   Spatial Envelope was originally developed for scene recognition. In [2], the authors use it to distinguish different secne categories for both natural scenes and man-made scenes. It reveals information about openness, expansion, naturalness and roughness of space which are very suitable for shots clustering. By applying different scale and oriented Gabor-like filters onto images, the spatial envelope is represented by the relationship between the outlines of the surfaces and their properties including the inner textured pattern generated by windows, trees, cars, people etc. Here we use three scales, 10 filters in total and partition filtered results into 8x8 blocks. The dimension of spatial envelope is then 640, along with color histogram, each frame is represented by a 896 dimensions vector.

Distance Computation

There exist several metrics to compute distance. In order to find the best solution for our compound feature vectors, spatial envelope and color histogram, we try to measure the correlation between the ground truth and distances computed by different methods, mainly Euclidean distance and Cityblock distance. The ground truth here is generated by manually labeling 0 or 1 indicating similarity. For example, A= [2.1, 3, 2], B= [0.5, 2.3, 1.7], GroundTruth= [0, 1, 1], we would conclude that sequence B has higher correlation with the ground truth than A. Based on this scenario, we found correlations are 0.3 and 0.5 respectively,

where Cityblock metrics has stronger correlation with ground truth. This could be understandable since the distance of histograms computed in this way makes more sense than in Euclidean distance while two metrics don't differ so much for spatial envelope.

3 Shots Generation and Representation

Videos are often composed of several chunks, and viewers may distinguish chunks from chunks by detecting scene changes. Based on this observation, we first split videos into shots before clustering. Intuitively, we record those frames with significant scene changes as salient frames, and examine the duration between two salient frames to see how long the shot remains consistent. If the duration passes the minimum length we set, the shot is kept, otherwise it's ignored.

The frames might be slightly different within one shot, but we still expect all frames should fall in the same cluster. Since using all frames to represent one shot will cost too much computation, and using average of one shot will, on the other hand, lose information, we therefore extract three frames from a shot. And because of the strong relation among those three frames, it can later help in clustering.

4 Clustering

The original distance d is computed as the sum of absolute differences, d=sum($|x_1-x_2|$), and its correlation with ground truth is around 0.5 which is not high enough, and our experiments also show that it can only achieve average 65% accuracy when clustering. Thus we try to use some labeled data and train the weighting vector which gives weights to features to improve accuracy.

Weighting Vector Learning

Instead of directly using ground truth (0, 1), where 0 indicates similar shots and thus distance is zero, and dissimilar for 1, as our objective distance, we modified the distance by setting d'=0.1*d if x_1 and x_2 are similar, and d'=d otherwise. This will preserve original distance in some sense while increase the correlation with ground truth to 0.9. Using this modified distance, we try to learn w such that $|x_1-x_2|*w$ is close to d' for m frame pairs, with restriction that w_i must be greater or equal to zero. That is to solve Aw=d' s.t. w>0, or minimize $1/2*w^TA^TAw-(A^Td')^Tw$, s.t. w>0, where

$$A=\begin{bmatrix} -|\vec{x}_1-\vec{x}_2|- \\ -|\vec{x}_3-\vec{x}_4|- \\ \vdots \end{bmatrix},\quad A\in R^{m\times n},\quad w\in R^{n\times 1},\quad d'\in R^{m\times 1}$$

Applying weighting vector is just like rescaling the coordinates of the feature space such that data can be clustered better. Given learned weighting vector, and since $w_i>0$, $|x_1\ w_i\ -x_2\ w_i\ |=|x_1-x_2|\ w_i$, we can directly multiply it into original feature space, and do clustering.

Deciding the number of clusters K

In most cases of application, users don't know how many types of shots in the targeting videos, so we develop a way trying to automatically choose the best K. At the beginning of clustering, we set K to be a large number and separate shots into K clusters. And then using the information mentioned in shots representation that three frames from one shot should fall in the same cluster, we record the ratio of the number of shots that frames fall in cluster i also fall in cluster j and the number of shots in cluster i in Clusters Relation Table R(i, j). If R(i, j) is high, we know that there is strong relation between them, and thus combine them into one and set K=K-1. Same procedures continue until no more high R(i, j) or reaching the minimal K we set.

After finishing iterative clustering, for those shots that three frames still fall in different clusters, we can keep the portion that most frames fall in the same cluster and ignore the other or generally ignore entire shot.

5 Experiment and Results Discussion

Database

In order to have videos that are close to reality, we collect our videos from youtube by keying the key words and randomly selecting. There are five categories of video in our database, tennis, basketball, surfing, snowboarding and news. Each of them contains ten or more videos and is totally around 30 minutes long. To train the weighting vector, we need to label frames. However the videos we downloaded are polluted, meaning that although some frames appear in a certain type of video, they might not visually belong to that category. For example, there are people talking in snowboarding videos, tennis players walking toward their seat in tennis video and so on. Therefore, we manually join around 10 minutes clean shots and use first 500 frames of them for training and remaining part for testing which contains around 1600 frames, as shown in Figure 3.

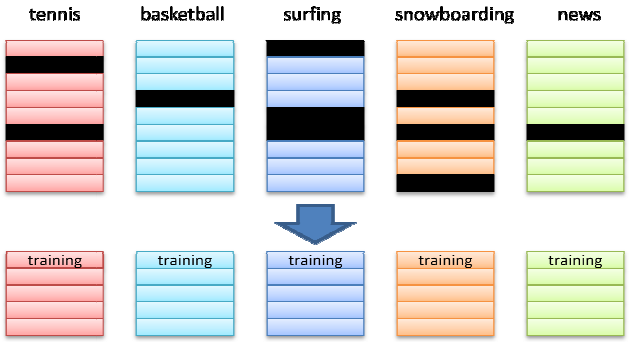


Figure 3. Black bars represent polluted shots. The bottom 5 manually extracted videos are used for training and testing, while eventually it is videos with polluted shots that need to be clustered.

Results and Discussion

Within 2500 frames, 500 from each category, we randomly select two frames and record their difference, modified distance and ground truth for 1000 to 20000 times, and learn weighting vector from those training data. The training and test error rates are shown in Figure 4, where dash lines represent clustering error for training set and test set without using weighting vector. The error rate of clustering 2500 frames used for training slowly decrease to nearly 0.04 while the test error doesn't change too much as we increase training data. This might be explained as the fact that there are obvious differences even within one clean video. For instance, people ski both in mountain and artificial skiing resort, and tennis courts look very different in U.S. Open and Wimbledon. If they are not contained in training set, we can hardly put them into the same cluster even we increase training data.

Using the scenario described in section 4 and applying the learned weighting vector, we classify original polluted videos and obtain 6 final clustered videos. For each cluster, although misclassified shots exist, we can easily tell what the theme is as in Table 1. For news cluster, most of the 35% that don't belong to news category are essentially close-ups of people, see Figure 5(a). So, even if there are only 65% that are exact news shots, the whole clustered video is visually consistent. For surfing cluster, most of misclassified shots are snowboarding which share common low level features with surfing as we can see in Figure 5(b). And cluster 2 is what we didn't train with, but it automatically extracts all the shots displayed in widescreen.

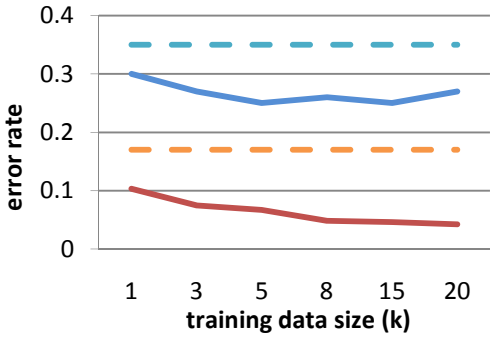


Figure 4. Red for training and blue for test error rate. The dash lines are error rates of training and test set without using weighting vector.

#	Theme	Correct/ total	Accuracy
1	Snowboarding	8:25/ 8:45	96%
2	Widescreen (black bars at top and bottom)		
3	Basketball	10:45/ 11:58	90%
4	News	14:22/ 22:18	65%
5	Tennis	15:48/ 18:20	86%
6	Surfing	19:18/ 28:15	68%

Table 1. Clustering results.

(a) (b)

Figure 5. (a) The left shot, close-up of people from snowboarding video, falls in the news cluster, the right one. (b) Snowboarding and surfing images share common low level features, thus in the surfing cluster, up to 20% are snowboarding shots.

6 Conclusion and Future Work

So far, two kinds of features work well in current database, especially spatial envelope which can distinguish major scene prospect. But there are still problems that current features can't solve. To apply to more general videos, we will need more features like motion, audio information and even object detectors. Video abstraction is only the first step of many applications; we'll use clustered results to further develop tools for video searching, browsing and creating, Figure 1(b) shows one of them.

7 Acknowledgements

Special thanks to media group leading by Ashutosh Saxena for providing precious opinion. This project is cooperated with Video Montage by Abhishek Gupta and Shakti Sinha.

Reference

[1] OLIVA, A., & TORRALBA, A. (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. International Journal of Computer Vision 42(3) , pp. 145-175.

[2] TRUONGTUBA, & VENKATESHSVETHA. (2007). Video Abstraction: A Systematic Review and Classification. ACM Transactions on Multimedia Computing, Communications and Applications, (Vol. 3, No. 1, Article 3).