Video Montage

Abhishek Gupta, Shakti Sinha

December 12, 2008

1 Introduction

Unedited video often has significant content that is of low interest to the viewer. One way to present such videos is by creating a video montage preserving interesting content. Manually doing this is difficult for a non-expert without specialized video editing software. We propose and demonstrate a technique for creating a video montage automatically from unedited video with minimal need for user intervention.

Interestingness in a video is inherently subjective, therefore to incorporate the user's notion of *interestingness* we ask for user input. This constrains us to using a small amount of data for training. The scenario that we have in mind is: the user browses through a few segments of the original video, marking them as interesting or uninteresting. We use this labeled data to train a classifier about what is interesting to the user, and then classify the rest of the video. The video classified into interesting and uninteresting segments is then processed to join the interesting segments and to shorten the uninteresting segments.

2 Building the feature set

After the user input, we partition the video into shots (a small portion of the original video) based on scene changes. The label that the user selected for a segment is considered for the whole shot. All the labelled shots are treated as part of the training set and each unlabelled shot as a test video. For every shot in the test set, we compute the percentage of its frames classified as interesting with the aim of classifying the whole shot into one category.

We Compute Pyramidal Lucas-Kanade Optical flow for every pair of adjacent frames. We get k (Optical Flow parameter) 2-D positions and 2-D velocities as a combined vectror(x,y,Vx,Vy) of the k fastest moving points with respect to the next frame. We use these vectors and compute the following features:

1. Optimal Number of Clusters (using a modified K-means algorithm).
2. Mean speed of the cluster
$$(1/k)(\sum_{i=1}^{k}\sqrt{Vx_i^2+Vy_i^2})$$
3. Maximum speed of the cluster
$$\max_{i=1..k}\sqrt{Vx_i^2+Vy_i^2})$$
4. Minimum speed of the cluster
$$\min_{i=1..k}\sqrt{Vx_i^2+Vy_i^2})$$

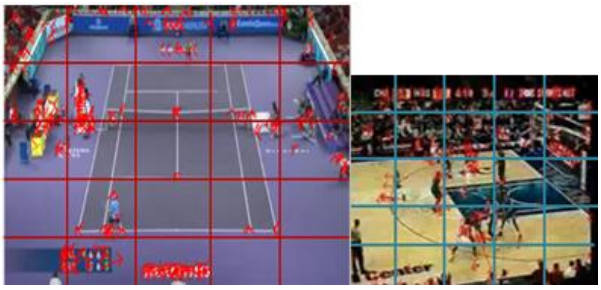

Figure 1: Illustration of the division of frames into 25 equal sized rectangles. The points indicate the fastest moving points in this frame relative to its next frame. The arrow originating from these points indicate the velocity vectors of these points.

As shown in **Figure 1**, we divide the whole video into 5×5 (25) equal rectangular frames and compute the following features:

1. Mean of the YCbCr values of the points that lie inside each of these squares. (25 each for Y, Cb and Cr)

2. Number of points that lie inside each of these squares when we run optical flow algorithm for every pair of successive frames. (25 features)

3. Sum of speed of all points that lie inside each such square. (25 features)

4. Histogram of L2-norm of (Vx, Vy) of the "k" vectors as computed from the optical flow. (giving us 21 features)

3 Forward Backward Smoothing

We also want to capture the time co-relation between the frames in the shots. Consider that our classifier predicts a shot to contain many interesting frames together and one or two non-interesting frames in between. We know that there is some time correlation between frames and therefore such a scenario is unlikely to occur. Hence we try to smooth the predictions for every shot. Furthermore, we divide the frames in the shot into the following categories:

1. BOUNDARY_FRAMES: frames at the start of the shot and the ones at the end of the shot

2. STATIC_FRAMES: frames that don't change much w.r.t the neighboring frames

3. NORMAL_FRAMES: the rest of the frames in the shot

For the different type of frames we have varying degree of confidence in the prediction made by the learning algorithms and coupled with aim of time correlation between the frames we have devised the following equations:

FRAME_TYPE	λ	μ
BOUNDARY_FRAMES	0.1	0.03
STATIC_FRAMES	0.1	0.9
NORMAL_FRAMES	0.85	0.15

Finally, if 75% of the frames in a shot are classified as interesting then we mark the whole shot as interesting else we mark it as non-interesting. We apply Forward-Backward Smoothing on the output of the classifier for every shot (y vector). Our smoothing algoithm is as follows:

Algorithm 3.1: FBSMOOTHING(y)

for $i \leftarrow 1$ **to** NUM_FRAMES
  do $\{w[i] = 0$
for $iter \leftarrow 1$ **to** NUM_ITER
  do $\begin{cases} \textbf{for } i \leftarrow 1 \textbf{ to } NUM\_FRAMES \\ \quad \textbf{do} \begin{cases} confidence = w[i] * y[i] * \lambda[frame_{type}] \\ smooth = y[i+1] * w[i-1] * \mu[frame_{type}] \\ w[i] = \arg\max_{w[i] \in +1,-1}(confidence + smooth) \end{cases} \\ \textbf{for } i \leftarrow NUM\_FRAMES \textbf{ to } 1 \\ \quad \textbf{do} \begin{cases} confidence = w[i] * y[i] * \lambda[frame_{type}] \\ smooth = y[i-1] * w[i+1] * \mu[frame_{type}] \\ w[i] = \arg\max_{w[i] \in +1,-1}(confidence + smooth) \end{cases} \end{cases}$
return (w)

where y[i] - the labels computed by running the shot as the test set and using the parameters learned by the training set using logistic regression

1. For STATIC FRAMES since there is not much motion in the frame relative to its previous or next frame, our classifier may not be able to predict its label correctly. But we are sure of the fact that if the frame preceding it and the frame succeeding it are labeled as interesting then probably this frame is also equally interesting.

2. For NORMAL FRAMES we give higher weight to the prediction made by the classifier.

3. For BOUNDARY FRAMES, typically there might be some errors in prediction near the start of the shot because it takes time to build some continuity within the shots. Similarly, typically near the end of the shot there might be a sharp break in continuity (as the shot is about to finish). So we don't trust the prediction made by the classifier and neither do we trust the continuity of the adjacent frame.

4 Seams for transitions

We claim that seams (defined below) can be used effectively for creating smooth and visually appealing transitions in the videos. We get information about the interesting and non-interesting video segments using our classifier as described in the previous sections. The techniques outlined in this section are then used to remove the non-interesting portions of the video and to join the interesting portions together.

A seam is a monotonic and connected manifold of pixels cutting across the time axis of a video represented as a video cube. Seams are associated with regions of low energy. Depending on the energy function, seams can represent regions of low activity and low importance.

5 Computing the seam

We represent the video cube as a graph where each voxel is a node. The nodes are connected to their adjacent nodes using the energy value of the voxel. With the graph construction shown in Figure 2, a minimum cut on the graph gives us the optimal seam in the video [1]. For finding the minimum graph cut, we use an implementation of [2]. We resize the videos to a lower resolution for efficiency.

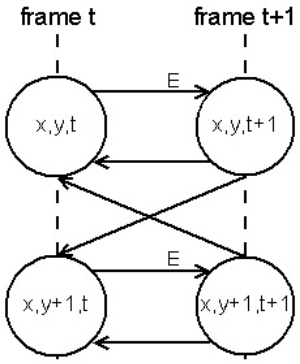

Figure 2: Graph construction for finding seam. This figure shows a longitudinal section of the video cube along the Y×T plane. Similar construction is present in the X×T plane. The unlabelled edges have infinite weights. These edges maintain the seam monotonicity and connectivity [1]. The nodes in the first frame are connected to the source and the nodes in the last frame are connected to the sink.

6 Shortening videos

We reduce the length of the video by removing low importance regions from the video cube. To represent the importance of a point, we use the difference in intensity of the point with the points surrounding it in space and time. We use the energy equation

$$\begin{aligned} E(x,y,t) = &|I(x+1,y,t) - I(x,y,t)| \\ &+|I(x,y+1,t) - I(x,y,t)| \\ &+|I(x,y,t+1) - I(x,y,t)| \end{aligned}$$

The seam computed on a graph using the above energy equation is the minimum energy manifold in the video cube. Removing this manifold reduces the length of the video by one frame. We repeat this process till we have removed the desired number of frames. Shot boundaries have very high energies which prevent seams from cutting across them. We use median filtering at places where we find energy spikes to allow the seam to cut across shots.

7 Joining videos

Videos should be joined along a seam such that the transition is smooth and preferably occurs in low importance regions. With this aim, we define the energy function as a weighted sum of the individual voxel energies and the difference in intensities of voxels in the two videos. We consider the combined video to be an overlap of the source videos, such that the output contains one of the videos before the seam and the other video after the seam. Consider the video cube configurations shown in Figure 3. We start with a configuration where a small number of frames from the beginning of the first video overlap with frames from the end of the second video (configuration A), and compute the seam for the overlapping region. We then shift the videos one frame at a time and find seams for each position till we reach configuration C. The seam having the minimum cost among all the seams found is the optimal transition manifold at which we join the videos.

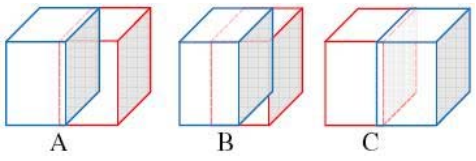

Figure 3: Configurations for finding seams. The two cubes represent the two videos being joined.

If in some configuration, t_1 and t_2 are the overlapping frames, the energy function we use while constructing the graph for computing the seam is given

by

$$E(x,y,t) = \alpha(E_1 + E_2) + (I(x,y,t_2+1) - I(x,y,t_1))$$

where E_1 and E_2 are the individual voxel energies in both the videos as defined in the previous section, $I(x,y,t_n)$ is the intensity of voxel in video n at point (x,y) in frame t. α is a parameter that controls the relative importance of the individual voxel energies and the difference in energies for finding the seam. A small value of α makes the seam join similar regions of videos, but it might cut through important areas. A higher value of α causes the seam to go through low activity areas of the video, but the transition might be less smooth. We used values of α between 0.1 and 0.01 for our results.

8 Experimental setup and results

First the user is asked to mark some of the segments as interesting and some other ones as non-interesting. We then split the video into shots based on scene changes. For all the frames that were marked as interesting, we treat the shots containing as interesting. We do the same for non-interesting parts. With this we have a collection of shots labeled as interesting and another collection of shots labeled as non-interesting. We use these shots as training sets. The rest of the shots are treated as the test set. We then compute the above described features for every shot. We use logistic regression to learn the parameters. For each shot in the testing set, we use the learned parameters, run logistic regression and get a vector which contains a label (interesting or non-interesting) for every frame in the test video. We have heuristically decided that if greater than or equal to 75% of the frames within a shot are predicted as interesting then we consider the shot to be interesting else the shot is marked as non-interesting.

We present here the results from one of the three experiements that we carried out. The experiemnt was performed on 2718 frames (1348 Interesting and 1370 Non-interesting frames) as training set i.e. approximately 90 seconds of video. The test set comprised of 18924 frames (around 4629 interesting frames and about 14295 non-interesting frames) i.e. approximately 630 seconds of video. This experiment was done using tennis videos. Similar experiment was also performed and similar results were obtained for basketball and snowboarding videos of approximately the same duration.

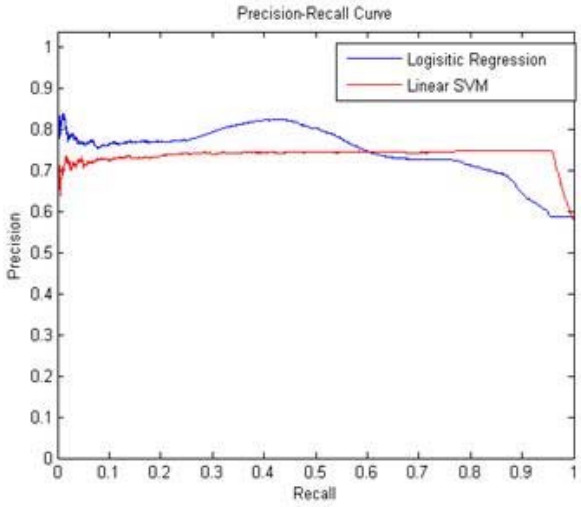

Figure 6: PR curves for logistic regression and Linear SVM

Classifier	Accuracy
Logistic Regression	70.68
Linear SVM	78.96

Class	Precision	Recall	F-Measure
Interesting	0.831	0.622	0.711
Non-Interesting	0.612	0.824	0.702

From the results we see that SVM performs slightly better than logistic regression for our data set. In general, we want our classifier to identify interesting segments as much as possible so that user's notion of interestingness is preserved. We can see from the results that the precision for the interesting class is quite high i.e. around 83.1%. On the downside the precision for the non-interesting class is low, equalling only 61.2%. This means we are performing poorly at identifying the non-interesting segments but are performing reasonably well in identifying interesting segments. This also means that we are able to shorten only 61.2% of the actual non-interesting segments.

We tried the seam techniques on a number of videos. We observed that joining and shortening videos preserved areas with high variation content and motion. Since it is easier to evaluate splicing of images compared to videos, we also tried the techniques on images, and found that images were joined with minimal artifacts. To ensure that joining videos worked, we took two overlapping segments of the same video, and the

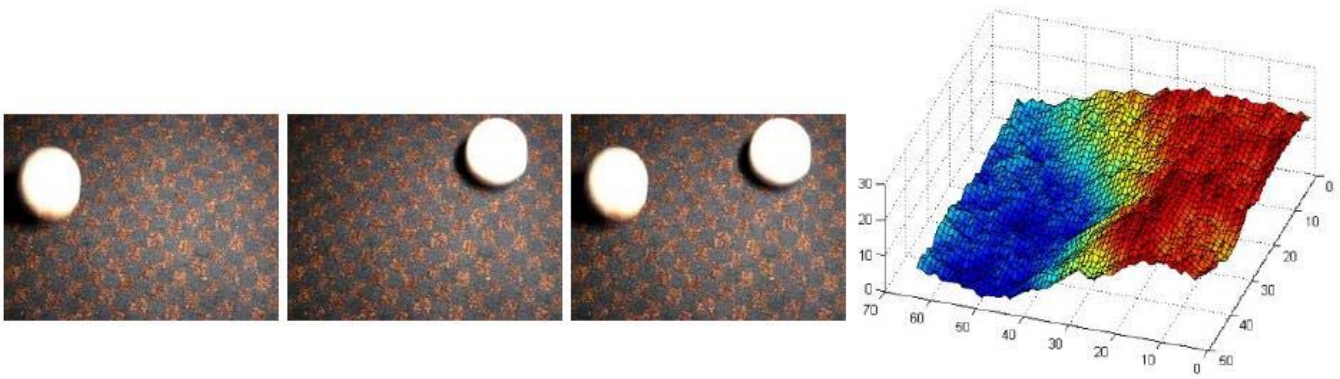

Figure 4: Shortening videos: Frames in the original video (first two images) and the shortened video (third image) show that static frame regions have been replaced by objects from other frames. (Right): One of the manifolds that were removed demonstrates how it cuts through the static regions of the video.

Figure 5: Joining videos: Some frames from the transition of two videos. We can observe that the videos have been joined at a position of similar content, and the snowboarders from the two videos have been matched. Figure also shows the seam along which the videos are joined.

output video was a perfect join as in the original video. The results are available online at http://www.stanford.edu/˜shakti/VideoMontage/.

9 Acknowledgements

We thank Ashutosh Saxena and the Stanford AI Lab for guidance and resources for this project.

10 References

1. Rubinstein, M., Shamir, A., and Avidan, S. 2008. Improved seam carving for video retargeting. In ACM Trans. Graph. 27, 3(Aug. 2008), 1-9.

2. Boykov, Y., Kolmogorov, V., 2004. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision. InIEEE Transactions on Pattern Analysis and Machine Intelligence, September 2004.