Keyframe-Guided Automatic Non-linear Video Editing

Vaishnavi Rajgopalan
Concordia University, Montreal
vaishnavi.rajgopalan@gmail.com

Ananth Ranganathan
Honda Research Institute USA
aranganathan@honda-ri.com

Ramgopal Rajagopalan
Research in Motion, Canada
rrajagopalan@rim.com

Sudhir P. Mudur
Concordia University, Montreal
mudur@cs.concordia.ca

Abstract

We describe a system for generating coherent movies from a collection of unedited videos. The generation process is guided by one or more input keyframes, which determine the content of the generated video. The basic mechanism involves similarity analysis using the histogram intersection function. The function is applied to spatial pyramid histograms computed on the video frames in the collection using Dense SIFT features. A two-directional greedy path finding algorithm is used to select and arrange frames from the collection while maintaining visual similarity, coherence, and continuity. Our system demonstrates promising results on large video collections and is a first step towards increased automation in non-linear video editing.

1. Introduction

Following the ease of recording and sharing digital videos, a growing need for applications to organize and present users' video collections has emerged. Personal video collections generally contain video clips of varying lengths and subjects. Each video clip also frequently contains unusable portions due to improper lighting, blur, occlusions, and other technical shortcomings. A system that edits a video collection to create coherent movies with specified content while omitting the unusable portions, would go a long way towards satisfying this need.

In this paper, we present a system that automatically generates visually coherent videos from a video collection. The input to our system consists of one or more keyframes that determine the content of the generated video. For instance, keyframes depicting a birthday party, a beach trip, and a class lecture would generate a video containing these three scenarios with appropriate transition sequences, if available. Importantly, annotation of video clips is not required.

To start with, we compute visual similarity across the frames in the collection. Similarity is measured using the spatial pyramid kernel[1], which computes a distance between histograms of SIFT features[2]. Based on this distance, the output video is produced by computing a greedy shortest ”path” through the frames of the video collection where the path is constrained to pass through the keyframes.

We demonstrate that our approach scales to large video collections and can generate videos in a matter of minutes once the video collection has been pre-processed. With minimal changes, it can be made to generate an infinite ”screen-saver” video or a video collage containing segments corresponding to each of the keyframes. These options make it a versatile tool for presenting large video collections in an intuitive manner. In the following sections, we first describe the pre-processing step wherein features for the frames in the video collection are computed. Subsequently, the video generation process is discussed, followed by results from experiments.

2. Pre-processing and Feature Extraction

Pre-processing involves extracting the frames from the videos in the collection, normalizing them, and computing features on them. To begin with, the extracted frames are histogram equalized and converted to grayscale. Subsequently, SIFT features are computed for each frame on an equally spaced grid. The grid spacing and the patch size on which the SIFT features[2] are computed are tunable parameters.

As mentioned above, we use the spatial pyramid kernel (SPK) [1] for computing similarity between frames.

The SPK works by computing histograms of clustered SIFT features at various spatial scales across the video frame. These histograms are appropriately normalized and then concatenated to yield the final representation of the frame. The SPK has a number of advantages over more commonly used features such as color histograms and texture[5]. First, SIFT itself is invariant to rotation, color variations, and some viewpoint variation. Second, by combining histograms from various scales, the SPK can effectively encode both global and local characteristics of the image. Finally, when compared to other features, the SPK is not very expensive to compute.

The SIFT features computed on the frames are clustered using K-means, with the number of clusters K usually set in the range of 200-400. For increased efficiency, only a small subset of features are used during the clustering, and the cluster centers, also called the dictionary, are stored. SIFT features from each frame are mapped to their corresponding clusters.

The SPK divides a frame into regions over multiple spatial scales. The number of spatial levels is denoted by L, and the frame is divided into equal sized regions such that each successive level has four times the number of regions as the previous level. A histogram of the number of features in each feature cluster is computed for each region. An illustration of this process is given in Figure 1. This can be done efficiently by noting that the histogram of a region at a higher spatial level is simply the sum of four histograms from the lower level. Hence, we only compute histograms at the finest spatial resolution and simply sum them up to obtain the histograms at coarser resolutions.

The final SPK histogram H, which is the feature used by our system to compute image similarity, is simply a weighted concatenation of all the histograms from regions at different resolutions.

$$H = \frac{1}{2^L} h^0 + \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} h^l \qquad (1)$$

where h^l is obtained by concatenating all the region histograms at level l, and h^0 is the histogram of the whole image. The SPK histogram given by (1) is computed and stored for each frame in the video collection.

For large video collections, we also compute representative frames, denoted henceforth as *repframes*, $\mathcal{F}_v$ for each of the videos v through hierarchical clustering of the SPK histograms. The clustering is performed using kernel k-means [4] using the histogram intersection function as the metric. More details about the clustering procedure can be found in [8]. Upto five repframes are extracted per video, although usually one suffices for small video clips. The repframe is taken to be the frame closest to a cluster center.

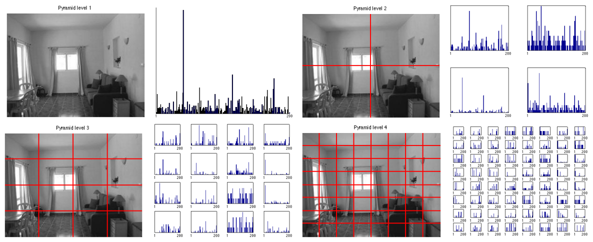

Figure 1. The Spatial Pyramid Kernel (SPK) representation: Histograms of clustered SIFT features computed on image regions at different spatial resolutions are concatenated to yield the SPK representation of the image. The image regions are obtained by dividing it into successively finer grids.

3. Video Generation

The input to the system for generating a video is a set of keyframes that determine the video content. Features are extracted from the keyframes in exactly the same manner as described in Section 2 above. Once the SPK histograms for the keyframes have been obtained, video generation proceeds by successively picking visually similar frames.

The histogram intersection function, used for computing visual similarity, is defined as

$$\mathcal{I}(x, y) = \sum_i \min\left(H_x(i), H_y(i)\right) \qquad (2)$$

where x, y are images, H_x, H_y are their SPK histograms, and the sum is over the bins of the histograms. It can be shown that (2) is a proper distance function.

The input keyframes are assumed to be arranged sequentially and the first and last keyframes are taken to be the start and end frames of the output video. Assuming N keyframes and representing these by $\{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_N\}$, we will now present the algorithm for generating the output video segment between keyframes $\mathcal{K}_i$ and $\mathcal{K}_{i+1}$.

The next frame in the video is obtained by computing the distance from the current frame to all the other frames, and selecting the one which is closest. However, for efficiency reasons, this step is implemented by first selecting the closest repframe $\mathcal{F}_c$, and subsequently, by selecting the closest frame in the corresponding video c as shown in Figure 2. This frame selection procedure is carried out from both keyframes $\mathcal{K}_i$ and $\mathcal{K}_{i+1}$ until the "paths" meet up. This yields us the output video segment between these two keyframes. This procedure is repeated for each pair of keyframes from $\mathcal{K}_1$ to $\mathcal{K}_N$ to yield the output video. The two-directional greedy path-finding algorithm provides an

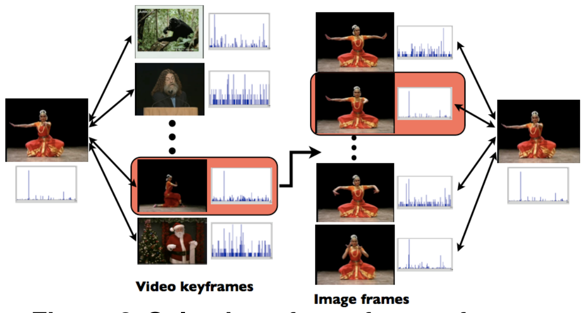

Figure 2. Selection of next frame of output video. (left) The current frame is matched with all repframes to select the closest video (shaded). (right) All the frames in the selected video are again matched with the current frame to obtain the best one (shaded).

effective and efficient solution since optimal algorithm such as Dijkstra's algorithm cannot be used in this case. Note that the runtime of the frame selection algorithm is $O(N_f V)$ where N_f is the number of frames per video and V is the number of videos in the collection.

Smoothing Factor: The above algorithm while producing visually similar frames in sequence, does not maintain dynamics nor does it prevent short loops which make the output video jerky. To maintain video dynamics, we modify the distance function to take into account the distance between future frames, i.e. the distance between the ith frame of video v, and the jth frame of video w is

$$D(v_i, w_j) = \sum_{k=0}^{K} c_k \mathcal{I}\left(v_{i+k}, w_{j+k}\right) \quad (3)$$

where c_k are manually-defined weights and the number of frames K to sum over is a parameter. A higher K leads to the output video following the videos in the collection more closely. The weighted distance (3) is the same as the one used in [3]. In our implementation of the weighted distance, we compute the two most similar video repframes to the current frame. Successor frames to w_j are assumed to come from the second-most similar video if w_j is at the end of video w.

Time-Weighted Distance Factor: The algorithm may get trapped into producing short, repetitive loops despite the weighted distance function if a small set of frames or a particular video is sufficiently dissimilar from all others in the video collection. This is avoided in our system through the introduction of a time-weighting factor into the distance function. A counter is maintained for each frame which is set to a constant T_{max} if the frame is selected. At each time-step, all non-zero counters are decremented by one. The distance function is modified to reflect the counter as $D'(v_i, w_j) = t_{w_j} \prod D'(v_i, w_j)$ where t_{w_j} is the counter corresponding to frame w_j. A large T_{max} makes the algorithm return to a particular frame after a longer time. The counters are maintained as a sparse vector since most of the values at any given time are zero ($T_{max} \ll$ number of frames in the collection), and hence this modification is efficient both in terms of memory and runtime.

Generation of Video Textures

Our system can also generate video textures [3], which are essentially videos of infinite length. This is done by converting the distance function D' into a probability distribution on the selection of the next frame $p(f) \propto \exp\left(-D'(f_c, f)\right)$, where f_c is the current frame. The next frame is sampled from its probability distribution but in a hierarchical manner, where first the video is selected by sampling from the distribution over repframes and a frame from the video is sampled subsequently. Since the distance function already incorporates dynamics and avoids loops, the generated video is relatively smooth. However, occasional unrelated jumps happen due to sampling effects.

4. Results

Our system was implemented in Matlab and we performed experiments using a collection of 45 videos ranging from 1 minute to 5 minutes in length. The videos contain diverse material such as lectures, dancing, skiing, wildlife, and parties. All videos were reduced to 320x240 pixels in size. SIFT features were detected on 16x16 image patches over a grid with spacing of 8 pixels. We computed 200 SIFT clusters during the SPK computation with a pyramid level of 3. The number of smoothing frames K in (3) was also set to 3 while the maximum time weight T_{max} was 100. One repframe was computed per video. Preprocessing the video collection took about 70 minutes. Output video generation occurs at 2-3 frames/sec.

The use of other features, in place of dense SIFT, was explored. We implemented our system with the Census Transform [9] and color histograms respectively. The Census Transform is a local feature that encodes the value of a pixel's intensity relative to that of its neighbors and was originally introduced for identifying correspondence between local patches. It can be computed extremely fast and also does not require the use of k-means for creating clusters since the number of possible transform values is finite and small. We used the Census transform with window size 3 yielding a histogram with 256 bins, more details of which can be obtained from

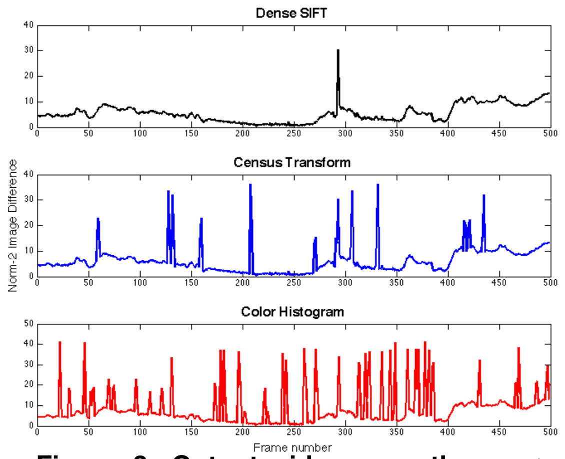

Figure 3. Output video smoothness (as normed frame difference between successive frames) for three types of basic features. Dense SIFT performs best.

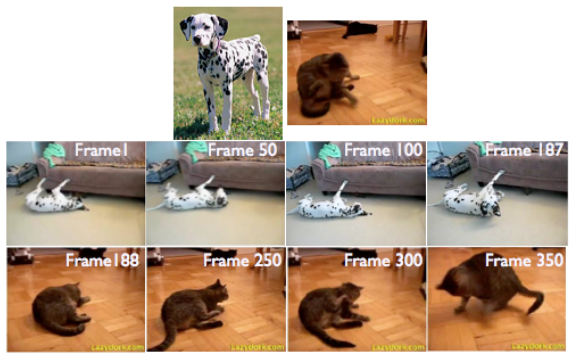

Figure 4. Output video progression for an input of two keyframes (top row)

[7]. These features are quantitatively compared in the context of our system using the pixel-wise Euclidean distance between successive frames of the output video generated when using each of the features. This is given in Figure 3 for an output video of length 500 frames. A sudden jump in the graph indicates an abrupt transition in the output, and the ideal graph would be steady and have low values. As can be seen, the Census transform, and especially the color histogram, produce output with many more abrupt transitions than Dense SIFT.

The progression of an output video sequence is shown in Figure 4. Note that even though the first input keyframe does not appear in the video collection, video frames containing a Dalmatian are chosen correctly as the subject of the output. In this case, the output transitions smoothly between four Dalmatian videos in the collection. Our system produces coherent videos and is able to select frames resulting in smooth transitions except when such connective frames are not available. It is hoped that larger video collections will give rise to more coherent output videos.

5. Discussion

We have presented a system for generating coherent videos out of unedited video collections. The content of the output videos is guided by a few input keyframes from the user. Dense SIFT features are used as the basis for computing similarity between frames, and this choice of feature is borne out by experimental results.

Our system is a first step towards providing increased automation support in non-linear video editing. In future work, other associated information, say, video or frames annotated with category labels could be used to enhance the video generation process. In particular, category labels have been shown to be amenable to spatial pyramid kernels [1]. However, visual similarity would still be required to produce a smooth video since intra-class appearance variation within category labels is very high. Video generation based on objects present in the frame [6] is also future work.

References

[1] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[2] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision*, 60(2):91–110, 2004.

[3] A. Schödl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *SIGGRAPH*, 2000.

[4] B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[5] F. Schroff, C. L. Zitnick, and S. Baker. Clustering videos by location. In *British Machine Vision Conf. (BMVC)*, 2009.

[6] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1470–1477, 2003.

[7] J. Wu and J. M. Rehg. Where am i: Place instance and category recognition using spatial pact. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[8] J. Wu and J. M. Rehg. Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In *Intl. Conf. on Computer Vision (ICCV)*, 2009.

[9] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Eur. Conf. on Computer Vision (ECCV)*, volume 2, pages 151–158, 1994.