CS231A Course Project Milestone

Walter Li
Stanford University
walterli@stanford.edu

Bo Xian See
Stanford University
bsee@stanford.edu

Abstract

We implement a method of automating the creation of cinemagraphs for an input video using stabilization, object tracking, image blending and video texturing. Currently, artists must carefully setup and edit the video to create a cinemagraph. This process is both tedious and time-consuming. Our method streamlines the pipeline to reduce the amount of work the user have to perform to create a cinemagraph. The only user input our method requires is a selection of four points for stabilization and tracking.

Using the fact that the frames are highly similar, we adopted an affine model for stabilization for computation efficiency. Next, we use SIFT to track the object's bounding box between frames. Then, an efficient two-band blending is used to blend across frames. Lastly, we adopted video texturing to find a good seamless loop for the GIF images. Our method will enable users to create cinemagraphs easily using videos captured from any devices, including their smart phones.

1. Introduction

Cinemagraphs are short, repeating animated images of a mainly static scene with slight motions, usually in GIF format. It was first introduced by a professional photographer, on his blog in Tumblr [1]. As mentioned earlier, it is extremely difficult for a common user to create cinemagraph as it takes time, skills and equipment to create one. An artist must carefully set up a scene to create a cinemagraph, as any motion shake increases the complexity of creating cinemagraphs. Moreover, artists must manually select masks for the area in motion, and blend frames onto one another.

Our novel system enables users to automate the creation process. The user can feed in a video recorded casually, removing the need to carefully set up the scene. We can thus assume that the frames will be highly similar, since the video is assumed to be taken from the same scene with roughly the same field of view.

1.1. Method

An overview of our system is as follows. Firstly, based on the review of stabilization techniques by Rawat and Singhi[2], we found out that the software stabilization technique proposed by Farid and Woodward[3] was best suited for our application. The algorithm is computationally efficient and its affine model captures sufficiently rich range of motion such as translation, rotation and scaling. The user's input will serve as the region of interest for video stabilization. Since as mentioned before, the video is assumed to be taken from the same scene, this method is well-suited for our needs.

We then use a SIFT descriptor to find the bounding box of the tracked object through the scene. In this step, we require the user to select the bounding box for tracking. The user can choose to skip this step if the change in the object's position is lower than a threshold.

After tracking the bounding box, we blend the bounding boxes of subsequent frames onto the first frame using the two-band blending algorithm proposed by Lowe[4]. This method fast and provides a good result for highly similar frames.

Lastly, we implemented techniques for transforming normal videos into video textures [5] to create a seamless endless loop for our video. These techniques include methods of calculating transition cost and probability, optimizing loops for minimum cost, and sequencing a list of transitions into a loop for GIF format. Due to a lack of a video texture format, we settled on producing an animated GIF format image file. Although this loses the randomness of true video textures, this is coherent with how cinemagraphs are generated.

1.2. Data Set

We intend to record video using a digital camcorder. Our sample size should be at least five videos of different scenes. Sample data will include approximately five-second clips of mostly static scenery. For each video, we manually select an object moving in the scene. Due to the expected

limitations of image segmentation and tracking, scenes must be generally static with only a few dynamic objects. To exclude video stabilization, at least one of the videos should be shot from a stable orientation, such as a tripod, to eliminate shaking in the image.

1.3. Evaluation

Evaluation of a piece of art is usually taken qualitatively. We will visually evaluate the final animated image and grade it based on smoothness of frame transitions, and the smoothness of blending with background image. However, some quantitative measurements can be employed to evaluate the results.

Intermediate results are produced by each subsystem, and evaluated individually. First, the video stabilization system can be evaluated by considering the complete video excluding the region of interest of the subject. The average standard deviation in pixel intensities can be used to compare the difference between the original video and the stabilized video.

Second, the object tracking and pyramid blending systems can be hard to quantify. But we can overlay a highlighting color over the tracked region of interest, and generate an animation over all the frames. A human user can evaluate all frames and count the number of frames with a correct and complete highlighting overlay.

Third, the two-band image blending step is also difficult to quantify. On the one hand, sharp edges due to image boundaries are unwanted. On the other hand, we do not want to blend away important subject features. Image overlay boundaries can be detected using edge detection. The presence of edges as an outline around the object signifies a non-optimal blending scheme. Faded features sometimes may be necessary to reduce sharp jumps in loop transitions, and thus unavoidable.

Finally, video textures can be evaluated based on total loop length, total cost of transitions in final animation (in comparison with other possible transitions not used), and the *loop performance*, which is an aggregate metric defined as length divided by cost.

2. Related Work

Previous related works to automate the process use video texturing to continuously loop through a video at selected regions [6]. However, their method does not track objects in motion, but instead consider regions containing dynamic objects. This loses the context of the scene as the regions are considered independent of each other.

In this paper, we aim to overcome this by using object tracking. Instead of creating the blending area based just on using changes between frames, we use tracking to determine the blending area across frames. This accounts for translational movement, and thus is more robust than the pipeline proposed by James et al.

We also referred closely to the works and tutorials by artists [7-9], to understand how cinemagraphs are traditionally created.

3. Technical Approach

3.1. Video Stabilization

To stabilize the video, we modelled the camera movement between two frames using a 2x2 affine matrix, and a 2x1 translational vector, as described by Farid and Woodward [3] via the equation:

$$f(x, y, t) = f(m_1 x + m_2 y + m_5, m_3 x + m_4 y + m_6, t - 1)$$

The variables m_1, m_2, m_3, m_4 form the 2 × 2 affine matrix A and m_5 and m_6 the translation vector T.

We solve for these 6 unknown variables by minimizing the quadratic error between the two frames.

$$E(\vec{m}) = \sum_{x,y \in ROI} [f(x, y, t) - f(m_1 x + m_2 y + m_5, m_3 x + m_4 y + m_6, t - 1)]^2$$

The ROI is the region of interest. Using the Taylor expansion, and discarding high order terms, we can solve the minimization function efficiently.

$$\vec{m} = \left[\sum_{ROI} \vec{c}\vec{c}^T\right]^{-1} \left[\sum_{ROI} \vec{c}k\right]$$

$$\vec{m} = [m_1 \quad \dots \quad m_6]^T$$

$$\vec{c}^T = (xf_x \; yf_x \; xf_y \; yf_y \; f_x \; f_y)$$

$$k = f_t + xf_x + yf_y$$

The terms f_x, f_y, and f_t are partial differentials of $f(x, y, t)$.

As long as the user selects a reasonably big region of interest, this minimization function is solvable.

A coarse-to-fine scheme is adopted in order to compute coarse movements efficiently. We use a 3-level Gaussian Pyramid to give us better estimation for larger movements. We chose to use 3-levels because we know that the frames will be from the same scene, and a 3-level Gaussian Pyramid should be sufficient to capture the camera shakes.

3.2. Object Tracking

In setting up our project, we have worked on all the other parts of the code, but haven't started on object tracking yet, so our information is not yet complete. The goal of object tracking is to produce a tight sub-image containing only the dynamic subject of interest. While the subject might move throughout all the frames, the selected region can grow to incorporate the moving subject. Also, the movement of the selected region can be used to augment detection of loop transition points in the video texture step.

The user has control over what the animated subject should be in the final cinemagraph, so some fashion of user input is allowed. The user first draws a bounding box around, or generously highlights a certain portion of the

image in a single frame, typically the first frame. This frame will be the base background frame. Sub-images in subsequent frames will be overlaid on top of this image.

The system samples the selected sub-image, and select keypoints of interesting features (i.e. corners). If not enough keypoints are found, we could supplement with uniformly sampled keypoints. Using SIFT [Lowe], we generate a set of descriptors. Then on each successive frame (or every N frames), a set of keypoints is generated and their descriptors are calculated. The area with the most keypoint matches is our dynamic region of interest. And we can even calculate how the object is being rotated or shifted with respect to the previous frame.

3.3. Two-Band Blending

We implemented a modified version of the two-band blending algorithm proposed by Lowe et al. [4]. In the paper, blending was only done for overlapping regions between two panorama images. Our modification enables the user to blend bounding boxes onto a common frame.

To blend the two frames, we first divide the frames into its high frequency component and its low frequency component. This is achieved using a simple two level Gaussian Pyramid. We combined the low frequency components using linear interpolation over a size that is specified by the user. The low frequency component of the blended image is thus the two dimensional liner interpolation of the overlapping region. As described in the paper, the high frequency component is not interpolated because we want to keep the details of the objects. We simply used the high frequency of bounding box because it contains the details we want to preserve.

The overall blended frame will then be the combination of the low frequency component and high frequency component.

3.4. Video Texturing

Given a series of frames, the objective now is to generate an infinite loop of frames with minimal discontinuities between frames. The video textures work by Arno Schodl, et al. can be applied to find the optimal transitions between non-continuous frames [5]. Since we are only concerned with a subsample of each frame, video texturing is given only a subsample of the whole video.

First, we compute the cost of a transition as L2-norm distances between the *i*th and *j*th frame for all the frames. This is saved as an *n*-by-*n* square matrix (D_{ij}). Dynamics preservation and future costs are also incorporated by, respectively, iteratively computing and solving the following set of equations.

$$D_{i,j} = \left\| I_i - I_j \right\|_2$$

$$w_k = \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{bmatrix}$$

$$D'_{ij} = \sum_k w_k D_{i+k,j+k}$$

$$D''_{i,j} = \left(D'_{i,j}\right)^p + \alpha \min_k D''_{j,k}$$

$$P''_{i,j} = K \cdot e^{-\frac{D''_{i,j}}{\sigma}}$$

The matrices I_i and I_j are the images at frame *i* and *j*, which can be used to calculate the cost matrix (*D*). The vector *w* is a set of weights for computing the cost matrix after considering preserving dynamics (*D'*). Finally, the cost matrix that also considers future costs is calculated as *D"*. The probability of each transition is modelled exponentially, and normalized such that each row sums to one.

This results in the 3D plots for probability of each transition shown in figure. This matrix is then filtered for local maxima, which will become our list of possible transitions. To make things interesting, we would like to weight the probabilities by distance of each transition (in indices from source to destination). This allows us to prefer multiple large jumps to many small transitions which could produce jittery motions in the animation. This is an issue because we must make deterministic rather than probabilistic decisions on transitions.

Following analysis of transitions, we consider only the top 20 (or however many) transitions. These are then used to find the optimal loop for up to a certain number of frames, with the lowest cost. This is optimized by dynamic programming. With the optimal set of transitions, we then sequence them using the steps outlined in section 4.3 of Schodl's paper [5] to generate the final sequence of frames, which is written to an animated GIF file.

4. Results

Because our results are in animated GIF format, please go to the following links to view the intermediate results. The original input GIF, taken by our smart phone and imported into Matlab, can be found here: http://imgur.com/8dgzV

4.1. Video Stabilization

The stabilized video can be found here: http://imgur.com/QivLt

There are still some movements in the non-subject areas of the image. However, this is a limitation of affine transformations. A shaky camera still experiences slight perspective changes, which cannot be fixed with an affine transform.

4.2. Two – Band Blending

Figure 1: Unblended image (left) and blended image (right)

The images shown in figure 1 are the results of image blending an orange with an apple. The original image samples are taken from Lowe's paper.

4.3. Video Texturing

The following charts depict some of the intermediate variables and data used by video textures to determine the optimal transitions. Each local maximum is considered good transitions to take. This is necessary because many local maxima are far lower in probability than the peaks in the center, but we still want to take distant transitions, and not a lot of small steps.

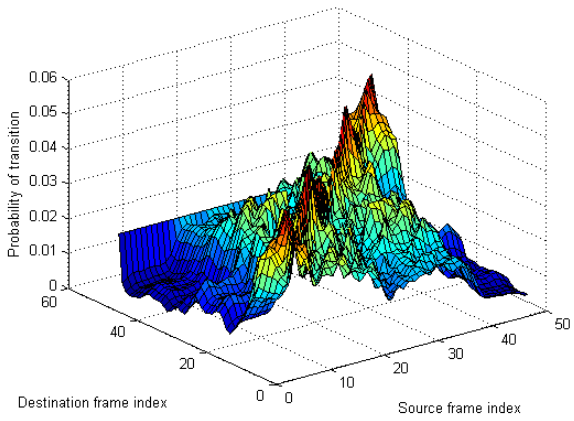

Figure 2: Probability of transition, modeled exponentially on the cost matrix, after considering dynamics preservation and future costs.

We define a new metric in order to measure the "performance" of a transition. A high performing transition would take a longer jump, but still have very low cost. The transition performance (TP) is defined as:

$$TP = \frac{Distance\ (Length)\ of\ Transition}{Cost\ of\ Transition}$$

This effectively weights the farther transition as "better" than very tiny transitions that only move back by a few frames.

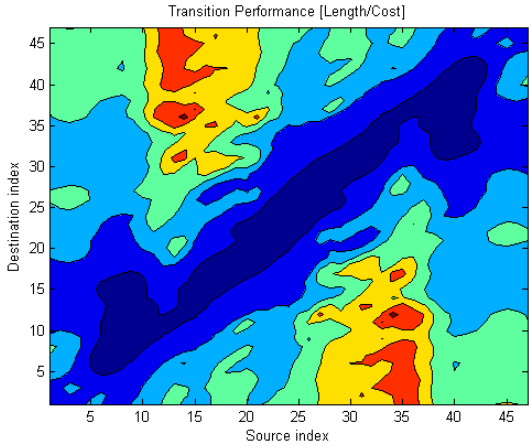

Figure 3: Transition performance is defined as the length of the loop created by the transition divided by the cost of that transition.

The result after creating a seamlessly looping animated GIF image using video textures can be found here: http://imgur.com/DcJ1r

5. Conclusion

In conclusion, we have made progress in creating a program for automating the production of cinemagraphs, partially animated photographs depicting subtle motions. We have made progress in integrating a video stabilization system based on affine image transformations, a function for blending the subject in each frame onto a static background image, and a method of creating a loop through a series of frames via video textures.

Further work is still necessary for a system to track the subject in every frame of the video. We also need to tweak and optimize each step of the process to be usable in a variety of scenarios through extensive testing.

6. References

[1] J. Beck, K. Burg. (2011, Feb.). *From Me To You – les tendrils.* [Online]. Available: http://fromme-toyou.tumblr.com/post/32635977796/les-tendrils-kaelen

[2] P. Rawat and J. Singhai, "Review of Motion Estimation and Video Stabilization techniques For hand held mobile video," *Signal & Image Processing: Int. J.*, vol. 2, no. 2, pp. 159–168, Jun. 2011.

[3] H. Farid, and J. B. Woodward. "Video Stabilization and Enhancement." *Science* (1997)

[4] M. Brown, and D. G. Lowe, "Recognizing Panoramas." *International Conference on Computer Vision*. 2003. 1218-1225.

[5] A. Schödl, R. Szeliski, D. H. Salesin, I. Essa, "Video Textures." *Proceedings of the 27th annual conference on Computer graphics and interactive techniques SIGGRAPH 00* (2000) : 489-498.

[6] J. Tompkin, F. Pece, K. Subr, J. Kautz, “Towards Moment Images: Automatic Cinemagraphs.” *Proceedings of the 8th European Conference on Visual Media Production (CVMP 2011)*. Nov. 2011.

[7] F. J. Baez. (2011, Apr.). *Cinemagraph Tutorial*. [Online]. Available: http://fernandojbaez.com/cinemagraph-tutorial/

[8] L. Banks. (2011, May). *How to Make a Cinemagraph with Photoshop and After Effects*. [Online]. Available: http://lesterbanks.com/2011/05/how-to-make-a-cinemagraph-with-photoshop-and-after-effects/

[9] P. Edenberg, et al. (2011, Jun.). *How to Make a Cinemagraph*. [Online]. Available: http://www.adorama.com/alc/article/How-To-Make-A-Cinemagraph