Eurographics Symposium on Rendering 2013
Nicolas Holzschuch and Szymon Rusinkiewicz
(Guest Editors)

Volume 32 (2013), Number 4

Automatic Cinemagraph Portraits

Jiamin Bai[1], Aseem Agarwala[2], Maneesh Agrawala[1] and Ravi Ramamoorthi[1]

[1]University of California, Berkeley
[2]Adobe

Abstract

Cinemagraphs are a popular new type of visual media that lie in-between photos and video; some parts of the frame are animated and loop seamlessly, while other parts of the frame remain completely still. Cinemagraphs are especially effective for portraits because they capture the nuances of our dynamic facial expressions. We present a completely automatic algorithm for generating portrait cinemagraphs from a short video captured with a hand-held camera. Our algorithm uses a combination of face tracking and point tracking to segment face motions into two classes: gross, large-scale motions that should be removed from the video, and dynamic facial expressions that should be preserved. This segmentation informs a spatially-varying warp that removes the large-scale motion, and a graph-cut segmentation of the frame into dynamic and still regions that preserves the finer-scale facial expression motions. We demonstrate the success of our method with a variety of results and a comparison to previous work.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—

1. Introduction

We all use portrait photographs to express our identities online. Portraits are often the first visuals seen by visitors to our professional webpage, social media profiles, and online dating sites. However, static portraits can fail to capture the personality we see on people's faces in real life, which includes their neutral expressions, smiles, laughs, and the transitions in-between. Short videos can capture these dynamics, but videos are awkward as online portraits because they are *too* dynamic; videos can contain camera and background motion, as well as large-scale motions of the head and body. They also have a timeline — a beginning, middle and end.

Cinemagraphs are a new medium that combines the benefits of static images and videos; most of the frame is static, but some parts animate in a seamless loop [1]. For example, a portrait cinemagraph might show dynamic facial expressions while maintaining a static overall head pose and background. Unfortunately, such portrait cinemagraphs are remarkably difficult to create, taking professional artists several hours if not a day [2].

In this paper, we completely automate the creation of cinemagraph portraits. Users simply capture a short hand-held video of a person, push a button, and our system outputs a portrait cinemagraph. There are a number of recent techniques for creating cinemagraphs, including mobile apps [3–5] and several research systems [TPSK11, JMD*12, YL12, LJH13]. However, these techniques are all challenged by portraits, because people are not good at keeping their head and body perfectly still. The existing techniques only stabilize overall camera motion, and a cinemagraph portrait requires removing the gross motion of the subject while preserving facial expression dynamics. The exception is the work of Bai et al. [BAAR12], who do remove the large-scale motions of objects. However, they require three different sets of strokes from the user, and the results can be very sensitive to stroke placement. Since compute time is also quite long, their system is impractical for casual users.

We contribute a fully automatic pipeline for portrait cinemagraphs. Our method is inspired by the system of Bai et al. [BAAR12], but does not require user strokes. We use face tracking [SLC11] as well as Kanade-Lucas-Tomasi [LK81] point tracks to automatically segment motion into two classes: large-scale head and torso motion that should be removed from the input video, and fine-scale facial expression motion that should be preserved in the cinema-

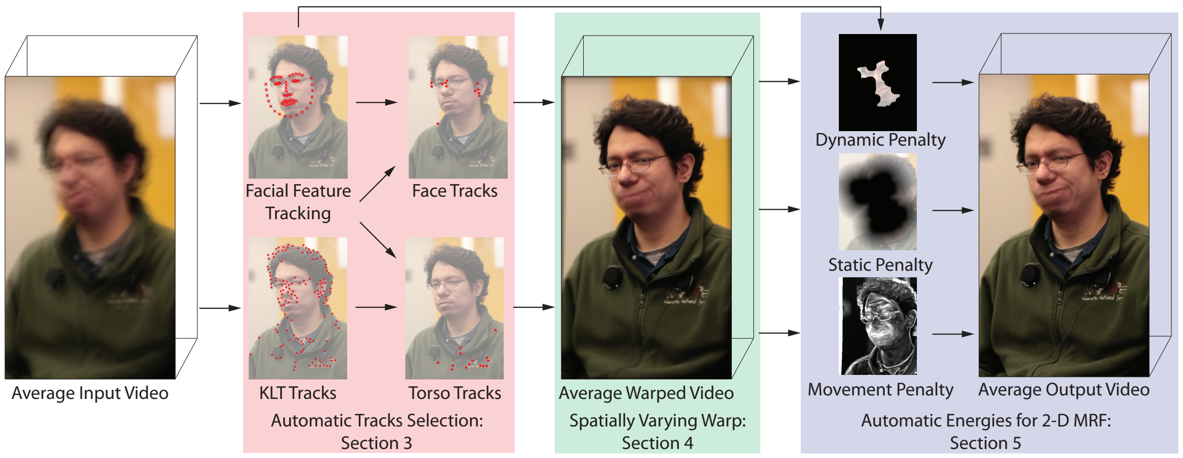

Figure 1: Our method automatically generates a portrait cinemagraph from an input video. The input, warp and output videos are visualized as averages across time. Notice that the face and torso are blurry due to motion in the input video. After warping, the face is sharp as it is stabilized, but the mouth and jaw are blurry as we preserve the internal motions of the face. The blurred border in the average warp video is due to the motion of the background. The average output video is sharp for static regions and blurry for facial parts which are animated. The key to our algorithm is a fully automatic technique to select KLT tracks which lie on static regions of the face, which allows us to immobilize the face after warping. We also compute automatic energy values for a graph-cut optimization which composites the warped video with a still image to create the final cinemagraph.

graph. We use the large-scale motion information to guide a spatially-varying warp [LGJA09, BAAR12] that removes the gross motion of the portrait subject. We then use a 2D graph-cut technique [BVZ01, KSE*03, ADA*04] to compute a mask that seamlessly combines the dynamic facial expressions with a still image for the rest of the subject and background. Our 2D approach is much faster than the 3D graph-cut of Bai et al. [BAAR12], which makes our approach amenable to implementation of portable devices such as phones and tablets. We also compare our method with a more naive approach that does not first segment motion into two types.

2. Overview

Our approach has three distinct stages as shown in Figure 1. In the first stage, we automatically select Kanade-Lucas-Tomasi (KLT) tracks which can immobilize the face and torso after a warp. Specifically, our method identifies and eliminates KLT tracks that follow expression motions in the face rather than the gross motion of the entire face. We then use RANSAC [FB81] on the remaining static KLT tracks to select a subset that can immobilize the face using a homography transform. In the second stage, we warp all frames to an automatically selected target frame using a spatially-varying warp [LGJA09, BAAR12] to immobilize the face.

In the third stage, we composite dynamic facial regions from the stabilized face video into a still background image. We use graph-cuts optimization on a Markov Random Field (MRF) to find the best compositing seam between the stabilized face video and the selected background frame. We automatically generate energy penalties corresponding to the detected moving facial parts in the output to act as constraints for the energy minimization. We also design new seam costs to encourage compositing seams to occur at regions with minimal pixel variation over time.

While our approach is similar to Bai et al. [BAAR12], there are two key technical differences that enable fast, completely automatic cinemagraph generation. First, we automatically analyze facial motion (Section 3) to differentiate between overall head motion and internal motion of facial expressions. Before this analysis the user needed to carefully place strokes in regions that only undergo overall head motion, which was challenging. Second, we improve speed by two orders of magnitude by using a 2D MRF with new cost functions based on facial motion analysis (Section 5.1), rather than a 3D MRF, along with several other optimizations.

Our approach relies on several assumptions about the input video. First, there is only one person with one facial pose in the video. We do not handle pose changes at the moment, although pose selection can be performed as a pre-processing step. Second, the face of the person should occupy a significant portion of the image; a good rule of thumb is about 10% of the image area. Third, the face can be aligned using a 2D warp, which means that there should not be large 3D rotations. However, in practice, reasonable results are obtained even when these assumptions are violated.

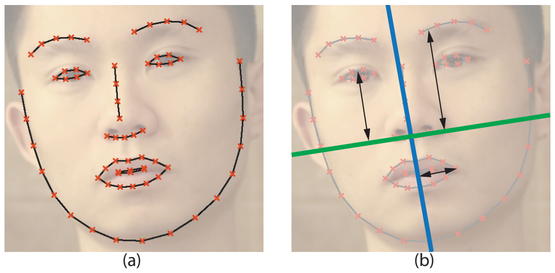

Figure 2: Facial features detected in a single video frame (a). We compute 2 axes per frame, a vertical line bisecting the face (blue) and a horizontal line across the base of the nose (green) (b). We measure facial regions by computing the perpendicular distance of their corresponding feature points to the axis perpendicular to their movement.

Facial Region	Feature #	Axis	Motion Threshold
left eye	38, 39	Horizontal	0.001
right eye	44, 45	Horizontal	0.001
left brows	20, 21, 22	Horizontal	0.003
right brows	23, 24, 25	Horizontal	0.003
left tip of lip	49	Vertical	0.0015
right tip of lip	55	Vertical	0.0015
bottom lip	57, 58, 59	Horizontal	0.0015
bottom jaw	8, 9, 10	Horizontal	0.0015

Figure 3: Each facial region is represented by the mean position of the corresponding feature number in the second column. The distance values for each region are measured from the assigned axis in the third column. The fourth column is the threshold value for motion detection after convolution with a LoG kernel and normalization by nose length l.

3. Track Selection

In the track selection stage, our goal is to select KLT tracks to guide a single spatially-varying warp to immobilize the face and torso. Our intuition is that KLT tracks that are useful for stabilizing the large-scale head motion should be located on the face, but not fall on any moving facial regions that are part of expression changes. They should also be consistent with a per-frame homography transformation, which is usually a good approximation for the overall motion of a face that is not significantly changing pose.

We first compute KLT tracks for the input video. KLT tracks which last for the entire video are denoted $\hat{K}_A$ and tracks which do not are denoted $\hat{K}_F$. This definition is similar to the anchor and floating tracks of Bai et al [BAAR12].

In order to find KLT tracks that are useful for stabilization, we use a face tracker [SLC11] to mark facial feature points throughout the video and analyze their motion to identify which facial regions are moving, as shown in Figure 1 and

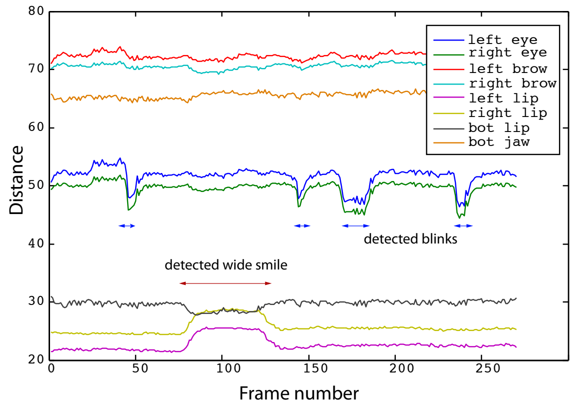

Figure 4: Typical raw distance values for facial regions in an input video. Notice the eye blinks give rise to distance drops for the eyes. These time instances are marked with a blue horizontal arrow. A smile causes the distance of the lips to change and this time instance is marked with an orange horizontal arrow.

discussed in Sec 3.1. Removing KLT tracks in moving facial regions which are not useful for stabilization significantly decreases the number of iterations for track selection, as discussed in Sec 3.2.

3.1. Facial Movement Detection

The goal of tracking facial features is to allow our algorithm to not only identify the location of the face, but also to determine the movement of the major facial regions in the face. For our application, we only consider four major facial regions; the eyes, eyebrows, mouth and lower jaw. The mouth is further sub-divided into three sub regions; the left and right-most region of the lip and the bottom lip. We use Saragih et al.'s [SLC11] face tracker to track the location of 66 facial feature points across the face as shown in Figure 2a.

The location of each facial region is the center of its corresponding facial feature points which are listed in Figure 3. We monitor the position of each facial region across time by using a coordinate transform. We measure its perpendicular distance from one of two local face orientation axes to detect finer-scale motions, as shown in Figure 2b. One axis bisects the face vertically passing through the bridge of the nose, and the other bisects the face horizontally across through the base of the nose. The vertical axis (blue) is the best fit line which passes through the facial feature points through the bridge of the nose and chin, and the horizontal axis (green) is the best fit line which passes through the facial feature points through the base of the nose. We recompute the face orientation axes for every frame. Note that the two axes are not necessarily perpendicular. We have found that this fit is stable for most of our examples as the facial feature tracker is stable for small facial motions.

Inlier ratio	0.6	0.64	0.69	0.75	0.81	0.9
# of iterations	3159	1378	556	204	65	17

Figure 5: Here we show the number of iterations needed to guarantee a probability of 0.99 to find a good homography fit for varying inlier ratios (total number of tracks = 30). As the inlier ratio improves, we need significantly fewer iterations to find a good homography.

We measure the perpendicular distance of each facial region from the axis where the motion change is dominant. This gives us fine-grained movement discrimination, so we can differentiate a slight smile where lips are closed (horizontal displacement of edges) versus a laugh (vertical displacement of lower lip). For example, the position of the eyebrow is the perpendicular distance measured from the horizontal axis (green). Figure 3 shows which axis each facial region is measured from.

We also compute the radius r and the nose length l of the face. Radius r is computed by averaging the three pixel distances from the center of the nose to the lower jaw, left ear and right ear (features # 9,1,17). Length l is simply the average length of the nose in pixels computed from the start and end feature points of the nose over all frames (features # 28, 31). We normalize all distance computations so that our algorithm is independent of face size by dividing all distances with l.

We use a median filter of kernel size 5 to smooth temporal noise after we compute the distances for each facial region per frame. Figure 4 shows a typical plot of the raw facial region responses for an input video and the detected movement. Note how the distance measurements change significantly for the facial regions when they move. We convolve the measurements with a Laplacian of a Gaussian (LoG) kernel of size 35 with sigma 4 to detect sudden change within a 1 second timeframe. The peaks of the convolved response indicate potential movement for their corresponding regions. For a facial region to be considered dynamic with respect to the face, the peaks in the convolved response have to be larger than a specified value. Figure 3 shows the threshold for movement detection we use, normalized by the nose length l. These thresholds were determined empirically from our dataset.

3.2. Track Pruning

KLT tracks in $\hat{K}_A$ that do not track large scale motion of the head should not be used for stabilization. We first select only the tracks in the facial region by removing tracks in $\hat{K}_A$ that are outside of a radius of $1.5r$ pixels from the center of the face. Second, we further prune the tracks that lie on the dynamic facial regions that form expressions by removing tracks that are inside or within a $0.2l$ distance from a moving facial region.

Once these tracks have been removed, we use RANSAC to fit a per-frame homography to the face motion; RANSAC also removes additional outliers that do not follow the overall face motion. Note that the initial removal of tracks significantly improves the ratio of inliers to outliers for homography fitting. A slight improvement in inlier ratio can significantly decrease the number of iterations needed for RANSAC to have a good probability of finding a fit which in turn decreases the computation required. Figure 5 shows the number of iterations needed for a 99% confidence of finding a good fit with varying inlier ratios. In practice, we have found that using this track removal technique decreases the number of RANSAC iterations needed by up to 2 orders of magnitude.

At each RANSAC iteration, we attempt to remove the motion of the face over the entire video by fitting homographies per frame to the remaining tracks using linear least squares. The set of tracks which gives the lowest RMS error for inliers after fitting homographies are the final set of selected tracks. We run RANSAC for 50 iterations with a 3.5 pixel inlier threshold for all of our examples. We label the final selected tracks from $\hat{K}_A$ as K_A. Figure 6a shows K_A selected for the face.

$\hat{K}_F$ tracks do not last the entire duration of the video, but can still help to guide the warp if we can be sure they are within regions which do not contain finer-scale motions. They provide consecutive frames with constraints during warping. We find useful K_F tracks by warping $\hat{K}_F$ tracks using the homographies estimated to align the frames. If a $\hat{K}_F$ track deviates no more than 3 pixels for each frame from its mean position, we consider it to be an inlier. The inlier K_F tracks are then used as input for warping along with K_A tracks.

We apply the same technique to stabilize the torso, but set the center of the torso as $3r$ below the face and only consider tracks within a radius of $4r$ of that center. No torso-specific track selection is performed, but any tracks removed during facial track selection are also not used for the torso even if they fall within the torso radius. Note that the homographies fit to torso movement are separate from the face homographies. Figure 6b shows K_A selected for the torso.

4. Warping

The goal of warping is to immobilize the face and torso in the video. Our approach finds a target frame to warp to instead of picking the first frame or forcing the user to select one. The automatically selected target frame should generate the least amount of warping after registering all other frames to it. Therefore, we find the target frame T by selecting the

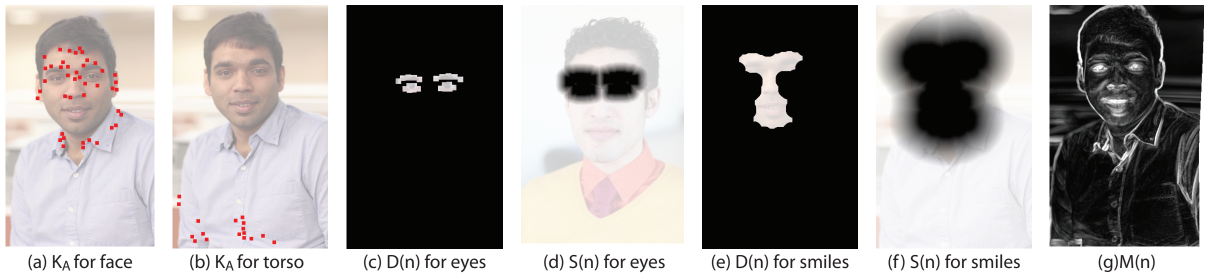

Figure 6: From left to right: (a) K_A for face, (b) K_A for torso, (c) example $D(n)$ for eyes, (d) corresponding $S(n)$ for eyes, (e) example $D(n)$ for smiles, (f) corresponding $S(n)$ for smiles, (g) example $M(n)$. The energy values are visualized overlaid with the portrait to provide spatial reference. High energy values of 1 have high luminance values, while energy values of 0 are black.

input frame t_1 that minimizes the L_2 distance between the locations of all K_A tracks in all other frames t_2.

$$T = \arg\min_{t_1} \sum_{\text{all tracks},t_2} |K_A(t_1) - K_A(t_2)|^2 \quad (1)$$

Frame T is also the target frame used in the next compositing stage.

We use the same warping algorithm as described in Bai et al. [BAAR12] (section 5.2 of that paper) with our automatically selected K_A and K_F as input. The tracks act as constraints over a grid mesh defined over each frame. Since the tracks specify locations on static regions, they should not move after the warp. Therefore, we solve for a set of meshes where the tracks K_A and K_F are stationary over time using least squares. Since we have selected the tracks K_F, we only have to use the second, refined warp described in Bai et al. (section 5.2 of that paper), thus reducing computation costs by up to 50%. The warped output is generated by texture mapping the frames onto the grid mesh.

5. Compositing

At the final stage, we use graph-cuts over a 2D Markov Random Field (MRF) to composite the warped video with the target still frame. Each node n corresponds to a spatial pixel location and is assigned one of two labels: $\lambda = \{still, warp\}$ after a cut is computed. Since our MRF is 2D, the assigned label at the node determines the pixel source for that spatial location across time; either the target frame T or the warped video. Our compositing algorithm is based on Bai et al. [BAAR12], but with three key differences. One, we use a 2D rather than 3D MRF. Two, we use a single still target frame, rather than a set. Three, since we no longer have user-provided strokes to guide the choice between still and dynamic regions, we must instead design new cost functions based on our analysis of facial motion that select only tracks indicating overall head motion.

There are two types of energies for our MRF: node potentials and seam costs. Node potentials are determined by the label assigned at the node n and seam costs are defined over two neighboring nodes, n_1 and n_2. We design our energy such that when it is minimized over the MRF using graph cuts, the labeling results in a composite cinemagraph that is natural with desired animated and static regions that transition seamlessly. We describe our cost functions (automatic energy functions) in Sec 5.1 and our overall energy functions in Sec 5.2.

We loop the output video by finding a pair of starting and ending frames that are visually similar. We minimize the total L_2 RGB distance in the warped video at the moving facial regions, using the same technique as video textures [SSSE00] to find the best loop. We then trim the input video and warped video to this duration. We also create a seamless cinemagraph by interpolating between the start and end frames to minimize visual artifacts using optical flow [Liu09]. We advect pixels from both directions based on their computed corresponding flows to generate an additional 10 intermediate frames. We cross-fade between the two constructed interpolations to create a temporal seamless loop.

5.1. Automatic Energy Functions

We compute three sets of energy functions (D(n), S(n) and M(n)) that range from 0 to 1 and are shown in Figure 6c-g. The energy terms are computed per node and thus correspond to each pixel. The computation of our energy terms is dependent on the presence of lip or jaw movements detected in the facial expressions; we use the movement detection approach described in Sec 3.1 and Figure 3. These automatic terms are designed to provide constraints to guide the MRF optimization. Specifically, D(n) is designed to encourage moving facial regions from the warped video to be composited in the final result. S(n) is designed to encourage relatively static regions to become static in the final result by compositing a still image. M(n) is designed to encourage seam transitions to happen at regions with minimal pixel variation over time.

Dynamic Region, D(n)
This energy is computed for nodes with label $\lambda = still$. We wish to assign a high energy for these nodes if they are within dynamic facial expressions, since they should be animated.

If no lip or jaw movements are detected, each feature point at a moving eye or eyebrow is assigned an energy value of 1 at its location at frame T with a radius of $0.1l$, as shown in Figure 6c. Pixels without any contributing features are assigned a 0 energy. The energy region is processed using morphological closing (image dilation followed by image erosion) with a radius of $0.25l$ to fill in holes in the energy region.

We generate this penalty differently if lip and jaw movements are present as the entire face is usually animated due to muscle structure. Each feature point at the lower jaw, eyes, eyebrows, nose and lips is assigned an energy value of 1 at its location at frame T with a radius of $0.1l$. The energy region is processed using morphological closing with a radius of $0.5l$ as shown in Figure 6e to fill in holes in the energy region.

Static Region, S(n)
This energy is associated with nodes with label $\lambda = warp$. We wish to assign a high energy for these nodes if they are not within dynamic facial expressions, since they should not be animated. We also want a ribbon of pixels between $S(n)$ and $D(n)$ where there are minimal constraints so that the algorithm is free to find seamless boundaries for compositing.

If no lip or jaw movement is detected, we generate this penalty by an image erosion of the complement of $D(n)$ by a size of $0.5l$. We then compute a linear penalty fall off with gradient 20 to encourage transitions to happen near the face, as shown in Figure 6d.

If lip and jaw movements are detected, we generate this penalty by an image erosion of the complement of $D(n)$ by a size of $3l$. We then compute a linear penalty fall off with gradient 5 to encourage transitions to happen near the face, as shown in Figure 6f. We use a larger kernel and a smoother gradient for image erosion because lips and jaw movements require a larger region for animation. The linear fall off is computed by using a Euclidean distance transform [FH04] multiplied by its corresponding gradient.

Pixel Movement, M(n)
This energy measures a component of the cost of seams between neighboring pixels. We want seam transitions between warp and still regions to happen where there are minimal pixel fluctuations across time. Hence, we take the maximum RGB L_1 distance of each pixel from the mean image of the warped video $\hat{W}(x,y)$ across all frames, as shown in Figure 6g.

$$M(x,y) = max_t(|\hat{W}(x,y) - W(x,y,t)|); \qquad (2)$$

5.2. Energy Function

There are two types of energies; node potentials Ψ and seam costs Φ. Each node n in the MRF has an energy associated with it depending on its label. A is a constant set to 1000000 to discourage any dynamic regions from being assigned a static label. Then the node potential Ψ is defined as:

$$\Psi(n,\lambda) = \begin{cases} A \times D(n), & \text{if } \lambda = still \\ S(n), & \text{if } \lambda = warp \end{cases} \qquad (3)$$

Seam costs are defined over 2 neighboring nodes n_1 and n_2 depending on their assigned labels. Let $\lambda(n)$ be the assigned label at node n and $C(n,\lambda(n),t)$ be the color of the pixel p at the node location from its respective label source at frame t. If the labels of the neighboring nodes are equal ($\lambda_1 = \lambda_2$), the seam cost is zero. Otherwise, the seam cost $\hat{\Phi}$ at time t is

$$\hat{\Phi}(n_1,n_2,\lambda_1,\lambda_2,t) = \frac{\gamma(n_1,n_2,\lambda_1,\lambda_2,t)}{Z(n_1,n_2,\lambda_1,\lambda_2,t)} \qquad (4)$$

$$\gamma(n_1,n_2,\lambda_1,\lambda_2,t) = |C(n_1,\lambda_1,t) - C(n_1,\lambda_2,t)|^2 + |C(n_2,\lambda_1,t) - C(n_2,\lambda_2,t)|^2 \qquad (5)$$

This seam cost is based on the work of Agarwala et al. [ADA*04]. Function γ measures the color similarity of the source pixels at the two nodes along the seams, while Z measures the edge strength of the warped video. Therefore, $\hat{\Phi}$ encourages seams to have similar colors or lie at edges in the warped video.

$$Z(n_1,n_2,\lambda_1,\lambda_2,t) = \begin{cases} \sigma(n_1,n_2,\lambda_1,t) & \lambda_1 \in \text{warp} \wedge \lambda_2 \in \text{still} \\ \sigma(n_1,n_2,\lambda_2,t) & \lambda_1 \in \text{still} \wedge \lambda_2 \in \text{warp} \\ \frac{1}{2}[\sigma(n_1,n_2,\lambda_1,t) + \sigma(n_1,n_2,\lambda_2,t)] & \text{Otherwise} \end{cases} \qquad (6)$$

where $\sigma(n_1,n_2,\lambda,t)$ is the edge strength within a pixel source λ and is computed with a 3×3 Sobel filter averaged across RGB.

Since we are using a 2D MRF instead of a 3D MRF, we collapse the seam cost across time to compute the best overall static compositing matte. We also include M(n) in the seam cost to encourage seams to occur in regions that have minimal pixel variation over time.

Therefore, the total seam cost Φ, across all time is

$$\Phi(n_1,n_2,\lambda_1,\lambda_2) = \sum_t \hat{\Phi}(n_1,n_2,\lambda_1,\lambda_2,t) + \alpha(M(n_1)+M(n_2)) \qquad (7)$$

where $\alpha = 200$. In total, our energy function which we seek to minimize is:

$$\frac{N}{4}\sum_n \Psi(n,\lambda) + \sum_{n1,n2} \Phi(n_1,n_2,\lambda_1,\lambda_2) \qquad (8)$$

where N is the total number of frames in the output video. Ψ provides constraints on which general region should be static

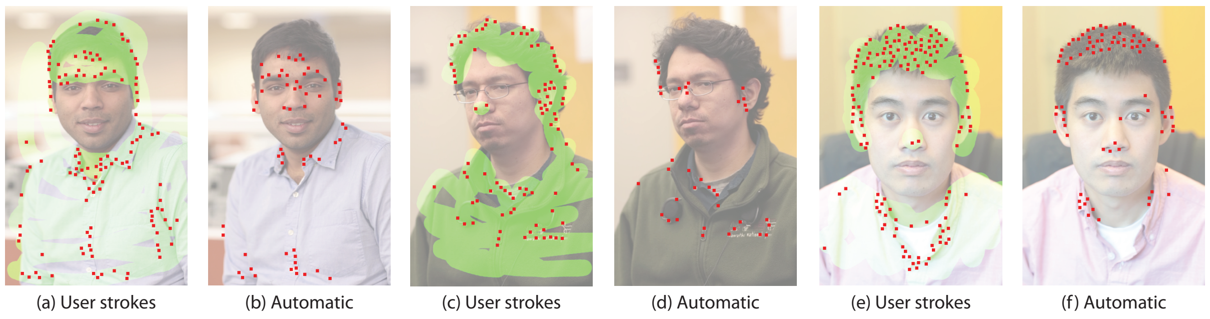

Figure 7: Comparison of the tracks selected using user-provided green strokes [BAAR12], versus our automatic method. Notice that our method selects fewer but more crucial tracks for stabilization. Our method has the advantage of selecting tracks that are very near moving regions where it is hard for the user to draw appropriate strokes.

or dynamic while Φ encourages seams to occur between pixels with similar colors and a stable appearance over time, or lie at an edge. We minimize this energy function using the alpha-expansion algorithm [BVZ01]. Once a labeling is computed, we create a final video simply by copying pixels from their respective sources.

6. Results

We captured 15 portrait videos and produced cinemagraphs to demonstrate the wide range of results our algorithm can create. Input sequences range from 5 seconds to 10 seconds. We resize the frame to 405 by 720 pixels. The videos are captured using DSLRs as well as mobile phones. Please refer to our main video and portrait cinemagraphs for each model for the final results, as well as the supplementary material for comparisons.

Comparisons: To evaluate the quality of our result, we show some comparisons of our automated cinemagraphs against a user-directed cinemagraph [BAAR12] in Figure 7. Notice that the stabilization in the result video is equally good even though our method selects fewer (but the most critical) tracks. Our method is capable of selecting tracks within the face that might be too tedious for the user to select with fine strokes, such as tracks that are between the eyes (Figure 7b,d,f). The compositing seams are also comparable even though our method uses only a 2D matte (see videos). In some cases, our method includes a small portion of the background in the final output, which can be distracting as the background will be animated. This could be due to the use of a static 2D matte or the automatic approach failing to find a compositing seam within the face.

In some examples, the motion of the camera and face is simple and a regular video stabilization algorithm will be able to stabilize the input video well enough for creating cinemagraphs. However, input videos with moving backgrounds or erroneous tracks in the background region can cause regular video stabilization to fail. Our method does not suffer from this issue because we only use tracks which lie on stationary regions on the face and torso. Please refer to the supplementary material for comparisons.

Another comparison we perform is to a naive algorithm that does not first prune tracks on dynamic facial expressions. Given the inlier/outlier ratios for each of our 15 videos, we simulate the number of RANSAC iterations needed to give a 99% probability of finding the correct homography. We find an average reduction in the number of iterations to be a factor of 27, with the largest reduction a factor of 185 across our examples.

Timings: The timings for each stage of our pipeline are shown in Figure 8. Our current implementation is single threaded and timings are measured on a 2 Ghz Intel i7 laptop. Our method is faster than previous work in all stages of the pipeline. For track selection, our method averages 0.70 seconds while a user will take 90 seconds [BAAR12], on average. Since our warp is simpler, our timings are also much faster. Our 2D graph cut is significantly faster than previous work due to the smaller number of nodes as well as the smaller set of labels. Our average compositing times are 14.41 seconds while the average compositing time in previous work is about 600 seconds for shorter videos [BAAR12]. Also, previous work often required iterations from the user to improve the input strokes.

Eyes and Eyebrow: All the input videos have eye movement from blinking. Our algorithm successfully composites the blinks as well as any eyebrow movements into the final cinemagraph. The large movements of the eyebrow in Model A are successfully retained and animated in the output cinemagraph. However, for some examples such as Model R, slight movements beyond the eye are also included in the output cinemagraph as seen in the video.

Lips and Jaw: Most of our examples have mouth movements such as smiles that are successfully animated in their output cinemagraphs. Jaw movements are the hardest to composite as the face usually changes its appearance significantly. Track selection becomes harder as there are fewer

Eg.	# frames	Track (Sec. 3)	Warp (Sec. 4)	Comp (Sec. 5)	Total
A	131	0.43 s	3.86 s	5.53 s	9.82 s
B	235	0.69 s	6.78 s	12.97 s	20.44 s
C	383	0.69 s	11.19 s	43.62 s	55.5 s
D	243	0.93 s	7.38 s	14.51 s	22.82 s
E	157	0.76 s	5.10 s	7.77 s	13.63 s
F	157	0.44 s	4.49 s	8.33 s	13.26 s
G	157	0.56 s	5.39 s	9.28 s	15.23 s
J	279	0.60 s	8.53 s	19.91 s	29.04 s
M	181	0.76 s	6.33 s	12.12 s	19.21 s
N	281	1.70 s	9.64 s	21.88 s	33.22 s
R	292	0.83 s	9.02 s	22.52 s	32.37 s
S	153	0.57 s	4.02 s	6.57 s	11.16 s
T	180	0.43 s	5.48 s	10.18 s	16.09 s
U	180	0.53 s	5.70 s	9.61 s	15.84 s
Y	180	0.72 s	5.93 s	11.37 s	18.02 s

Figure 8: The timings for track selection, warping, compositing and total time taken of our automatic algorithm for all our examples. Please look at our accompanying video for the results. Models E,F,T are shot with a mobile phone.

static tracks. The margin of error for compositing is also greatly reduced as most of the face is animated. Therefore, no previous work demonstrates animated smiles. None the less, our algorithm successfully composites large jaw movements and smiles such as Model B and Model R, as shown in our video. Notice that while there are large motions for the entire face for Model R, we are still able to generate a good portrait cinemagraph.

Failures: Our method fails at times when the facial deformation is too large and too fast. Our seam cost is an average of the seam costs across time. Therefore, if a large deformation lasts for only a short period of time, the aggregate seam cost will not change significantly and the composite will be poor as shown in Figure 9a as well as our video result.

User Study: The aesthetics of portrait cinemagraphs can be rather subjective as some people find the motion of only parts of the face surreal or uncanny. Therefore, we conducted a user study on Amazon Mechanical Turk to better understand how the general population would respond to portrait cinemagraphs. We showed 30 users 7 sets of results (30 HITs, each HIT has 7 sets). For each set, we showed the still image, the input video and our cinemagraph result side-by-side and asked them to pick their favorite medium to represent each person. We recorded 218 total votes and 80.3% of them were in favor of dynamic portraits (either input video or cinemagraph). 53.0% of the total votes were for cinemagraphs and only 19.7% were for a still image. Our collected votes yield $\chi^2 = 40.5229$ when computed against an expected uniform distribution and exceeds the χ^2 value of 13.82 for $p = 0.001$ with 2 degrees of freedom. Therefore,

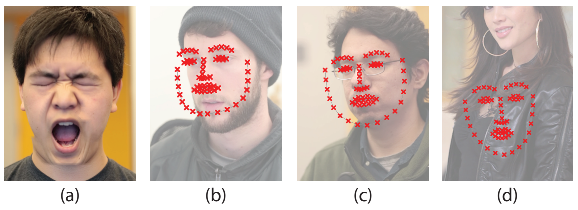

Figure 9: The example in (a) has a poor compositing result at the mouth due to the sudden large movement. The average seam cost is dominated by the rest of the video. The facial features detection is bad in examples (b),(c) and (d). While example (c) still produces a good result due to the correct estimates for the mouth and jaw, examples (b) and (d) are failures.

the survey supports our hypothesis that people do in general prefer cinemagraphs for portraits. We also asked the users if they would use an automated technique to generate their own cinemagraphs and 73.3% of them indicated that they are willing to try such a technique.

Limitations: Since our method is dependent on accurate facial feature tracking, we will not always produce good results if the facial feature tracker fails. In the example shown in Figure 9b, the jaw is moving but the face tracker fails to accurately detect its location. As a result, the video is stabilized with respect to the jaw causing the head to move up and down in the result video. Our method works in some cases when the facial feature tracker is partially correct as shown in Figure 9c but will certainly not work when it misses the face entirely as in Figure 9d.

Since we are using a spatially-varying warp to align faces, we cannot handle large rotations of the face, where occlusions as well as depth discontinuities become prominent.

7. Conclusions and Future Work

We have presented a fully automated technique for creating portrait cinemagraphs that requires no user input; the user simply captures a hand-held video and receives a portrait cinemagraph. Our technique also runs faster than previous work due to a more efficient MRF optimization and a simpler one-stage warp. The key contribution is a technique for segmenting facial feature tracks into those that are part of facial expressions, and those that track the overall face pose. Also, by defining energy functions unique to our problem, we can automatically composite the warped video to create a portrait cinemagraph.

One avenue for future work would be to automatically detect and composite additional moving regions such as cloth or hair onto the final cinemagraph. These regions are significantly more difficult to handle, especially when the person is moving in the video, because of motion ambiguity between the background, cloth, hair, and rest of the person.

In summary, we believe our automatic technique will empower novice users with the ability to create their own cinemagraph portraits without effort and expertise. As cinemagraph portraits are more expressive than static photographs, we think that they will eventually become a popular alternative to conventional portraits in webpages and social media.

Acknowledgements: We thank the reviewers for their detailed comments. We would also like to thank the models, Armin Samii, Brandon Wang, Dikpal Reddy, James Andrews, Michael Tao, Rahul Narain, Sean Arietta, Siang Yun Ang and Soham Mehta. This work was supported in part by ONR PECASE grant N00014-09-1-0741, NSF grants CCF-0643552, IIS-1016920, an A*STAR NSS PhD fellowship, gifts and software from Adobe, and the Intel Science and Technology Center for Visual Computing.

References

[1] URL: http://cinemagraphs.com. 1

[2] URL: http://www.filmindustrynetwork.biz/nyc-photographer-jamie-beck-cinemagraph/12173. 1

[3] URL: http://kinotopic.com. 1

[4] URL: http://cinemagr.am. 1

[5] URL: http://www.icinegraph.com. 1

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 294–302. 2, 6

[BAAR12] BAI J., AGARWALA A., AGRAWALA M., RAMAMOORTHI R.: Selectively de-animating video. *ACM Transactions on Graphics* (2012). 1, 2, 3, 5, 7

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (Nov. 2001), 1222–1239. 2, 7

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June 1981), 381–395. 2

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: *Distance transforms of sampled functions*. Tech. rep., Cornell Computing and Information Science, 2004. 6

[JMD*12] JOSHI N., MEHTA S., DRUCKER S., STOLLNITZ E., HOPPE H., UYTTENDAELE M., COHEN M.: Cliplets: juxtaposing still and dynamic imagery. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (2012), UIST '12, pp. 251–260. 1

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3 (July 2003), 277–286. 2

[LGJA09] LIU F., GLEICHER M., JIN H., AGARWALA A.: Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics 28*, 3 (July 2009), 44:1–44:9. 2

[Liu09] LIU C.: *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, May 2009. 5

[LJH13] LIAO Z., JOSHI N., HOPPE H.: Automated video looping with progressive dynamism. *ACM Transactions on Graphics* (2013). 1

[LK81] LUCAS B. D., KANADE T.: An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence* (1981). 1

[SLC11] SARAGIH J. M., LUCEY S., COHN J. F.: Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vision 91*, 2 (Jan. 2011), 200–215. 1, 3

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Computer Graphics Proceedings, Annual Conference Series, pp. 489–498. 5

[TPSK11] TOMPKIN J., PECE F., SUBR K., KAUTZ J.: Towards moment imagery: Automatic cinemagraphs. *Visual Media Production, Conference for 0* (2011), 87–93. 1

[YL12] YEH M.-C., LI P.-Y.: A tool for automatic cinemagraphs. In *Proceedings of the 20th ACM international conference on Multimedia* (2012), MM '12, pp. 1259–1260. 1