DOI: 10.1111/cgf.12682
Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

Smooth Loops from Unconstrained Video

L. Sevilla-Lara[†1], J. Wulff[1], K. Sunkavalli[2] and E. Shechtman[2]

[1]Max Planck Institute for Intelligent Systems
[2]Adobe Research

Abstract

Converting unconstrained video sequences into videos that loop seamlessly is an extremely challenging problem. In this work, we take the first steps towards automating this process by focusing on an important subclass of videos containing a single dominant foreground object. Our technique makes two novel contributions over previous work: first, we propose a correspondence-based similarity metric to automatically identify a good transition point in the video where the appearance and dynamics of the foreground are most consistent. Second, we develop a technique that aligns both the foreground and background about this transition point using a combination of global camera path planning and patch-based video morphing. We demonstrate that this allows us to create natural, compelling, loopy videos from a wide range of videos collected from the internet.

1. Introduction

Loopy videos and animated GIFs have gained tremendous popularity in the last few years with the ease of video capture, and the introduction of video sharing services like Vine.co and Instagram.com. More than 100 million people watch Vine videos every month, and over one billion loops are played daily on Vine alone [Vin14]. The typical length of these videos is surprisingly short – up to six seconds on Vine and 15 on Instagram. These videos are popular in social networks, blogs, digital marketing, music clips and art, because they capture key scene dynamics and can convey a richer meaning than a single photograph, but are more concise, portable, and sharable than long videos. Most such videos are created by cutting a short clip from a longer video. This frequently leads to abrupt changes from the last to the first frame, resulting in an uncomfortable experience when watching them played as a loop. One popular "trick" to avoid this, is to play the video back-and-forth (by concatenating a copy of the video in reverse order). While this alleviates abruptness due to changes in the position of the objects in the video, the changes in motion are still abrupt, and often lead to unrealistic motions due to time-reversal.

In contrast, artists, animators, and professional photographers create strikingly hypnotizing micro-videos by perfectly "closing the loop" to seamlessly transition from the last frame back to the first frame (e.g., [Inc, Raj, Car]). Creating perfectly loopy clips from casual video footage can be highly tedious or even impossible for some videos. It typically involves manually finding the right cut locations in the video clip, and aligning the two ends with professional video editing tools. The goal of our work is to automate these two steps, and make the process of creating compelling loopy video clips significantly easier.

The seminal "Video Textures" work by Schödl et al. [SSSE00] proposed an elegant framework to automate this process for specific types of content. They showed that videos with dynamic texture-like characteristics (such as the flame of a candle) often contain multiple moments with similar appearance and dynamics that can be used as transition points for creating infinite loopy videos. The camera and background in these videos are static.

In this work, we generalize the Video Textures framework to handle videos "in the wild". These are typically captured by hand-held devices and contain arbitrary camera motion (including translation and zoom) and complex, non-rigid scene dynamics (including human motion). We focus on one popular type of content: videos of a dominant moving foreground, such as a moving person, animal or an object, in front of a roughly static background (small motion in the background is usually fine), captured by a moving camera. Motivated by research on visual attention [FS11] that shows that people have higher tolerance to inaccuracies in the pe-

† Portions of this work were performed while the first author was at University of Massachusetts, Amherst

riphery of the point of attention, our key observation is that, in many cases, finding moments where only the *foreground* is similar, is sufficient to produce pleasant looping videos. In order to handle such challenging videos we replace both the *analysis* and *synthesis* components of the Video Textures framework with new algorithms. During analysis, we find moments in the input video where the dominant foreground is similar both in appearance and dynamics. We start with a rough segmentation of the foreground in a scene. We develop a similarity metric based on this segmentation to robustly assess similarity in the motion and appearance of the foreground between two sets of video frames. In the synthesis step, we propose a patch-based method to morph between two video clips using second-order motion constraints for the foreground and automatic temporal gap estimation based on the dynamics of the scene. We show compelling results on several challenging videos downloaded from the internet, as well as comparisons to previous methods.

2. Related Work

The analysis of scene dynamics in videos is a critical component of many video editing tasks, and has been studied extensively in graphics and vision literature. We focus on the techniques that are particularly relevant to our work.

Video Transitions Combining multiple video clips is one of the most common video editing operations, and film editors have developed a taxonomy of the different kinds of transitions (or "cuts") used to achieve this (see Goldman [Gol07] for an overview). While general video editing requires a significant amount of skill and time, it can be (semi-) automated in specific instances. Zheng et al. [ZCA*09] leverage information in a light field to automatically author cinematic effects from photographs. Kemelmacher-Shlizerman et al. [KSSGS11] generate smooth animations from personal photo albums by aligning and transitioning between the faces in the photographs. Berthouzoz et al. [BLA12] focus on editing interview footage, and propose a method that uses audio-visual features to automatically find good cut locations. Most of this previous work is applicable only to specific classes of data (for e.g., faces) and cannot be trivially extended to general video sequences. In contrast, our technique does not make strong assumptions about the content of the input video sequences, and, to our knowledge, is the first general purpose approach for synthesizing realistic video transitions automatically in the presence of camera motion and complex foreground dynamics.

Video Morphing Transitioning between two shots might require morphing between the two (especially when the content is not properly aligned, or has significant differences). There is a significant amount of literature on image morphing [Gom99], and most techniques compute correspondences between images, and construct motion trajectories from these correspondences. Both of these are challenging problems that become especially harder in the presence of complex camera motion and scene dynamics. Previous work has tackled this by relying on user input to specify the region of interest [BAAR12, RWSG13], or correspondences between pairs of videos [LLN*14]. These methods cannot handle regions without correspondences, as often happens with the backgrounds in our examples. In addition, the synthesized motion trajectories need to be consistent with the motion in the original footage for the morphed result to look natural. Many morphing techniques (for e.g., Shechtman et al. [SRAIS10]) do not account for this, and produce non-realistic results for general video sequences. In our work, we compute correspondences between video frames using the technique of HaCohen et al. [HSGL11]. We morph the background and the foreground separately to account for the fact that they might move in different ways. In addition, we synthesize background motion trajectories using linear interpolation (or use linear motion constraints when background correspondences do not exist), while using parabolic constraints to synthesize foreground motion trajectories. Unlike previous work, this allows us to handle both moving cameras and fairly general scene dynamics.

Video Textures and Cinemagraphs Video Textures [SSSE00, KSE*03, DCWS03, AZP*05] create infinitely looping videos by finding similar frames in a video clip (based on image features), and transitioning between them using morphing. However, these methods were designed to work on videos that are shot by a static camera (or a smoothly moving camera), and where the dynamics are either local (e.g., a swinging candle flame or flapping flags) or stochastic in nature (e.g., flowing water, fire flames). More recent efforts use spatially varying dynamics to handle multiple motions [LJH13] and to include manual interaction [TPSK11], but they assume a static camera. Cinemagraphs are a related form of media that lie between video and photographs; the salient objects in the scene remain animated while the surrounding objects are held still. Recent work has proposed interactive tools for their creation [TPSK11, BAAR12, JMD*12]. These methods work by using one of the video frames for the background and pasting the moving foreground on top. The inputs to these methods have to be captured using a static camera, the motion is often localized or repetitive in nature, and the methods require some user interaction. Unlike this previous work, our technique can handle both camera motion and non-stochastic scene dynamics (including highly structured motions like human movement). We are able to achieve this by considering the background and foreground separately while finding good transition points, and aligning and morphing them.

3. Overview

Given an input video, V, the goal of our method is to use a subset of the original frames and produce a shorter video,

V_{out}, whose last frame seamlessly transitions to the first frame to produce a realistic and compelling video loop. In other words, any differences between the foreground and background appearance (illumination, location, pose) or scene dynamics (velocities and higher location derivatives) between these frames should change as smoothly as possible so that the transition is not noticeable. We achieve this using a two step process. In the first step – the *analysis* stage – we analyze the input video to automatically choose the short subset clip that we consider to be most likely to produce a loop with a smooth transition of the foreground. We do this by finding frames in the video sequence that have the most similar foreground regions up to a global rigid alignment. This results in a frame pair (a,b) that denotes the start and end of the subset clip that we use to synthesize the video loop. While the appearance of the foreground in these frames is similar, it needs to be aligned properly for the loop to be seamless. In the second step – the *synthesis* stage – we morph the frames around this transition point, i.e., frames $V(a+k)$ to $V(a)$ and $V(b)$ to $V(b-k)$, to handle the differences in appearance and motion, and synthesize a smooth transition of both the foreground and the background from frame b back to frame a. This results in the output loopy video V_{out}.

Pre-processing Our system is meant to work on videos "in the wild", which are often captured with handheld cameras and contain high frequency camera motion from unintended jitter. This makes it difficult to make assumptions about the motion, and so we stabilize the input to remove only the high frequency component of the camera motion using Adobe AfterEffects, set to smooth motion. Previous methods [SSSE00, LTK12, LJH13] do not handle camera motion and thus stabilize the input videos to eliminate background motion entirely.

We assume that the input video contains a single dominant foreground motion, and we identify this region using the motion segmentation algorithm of Papazoglou and Ferrari [PF13]. We choose this method because it is automatic and is designed with minimal assumptions on the input videos. It consists of an initial estimate based on motion boundaries followed by refinement based on a spatio-temporal extension of GrabCut [RKB04]. The per-frame masks computed by this technique may contain several non-connected components, and may miss some portions of the foreground. We smooth the mask with a median filter, fill in holes by dilating the mask, and choose the main connected component as the foreground. For each frame $V(i)$, we denote the binary mask that we obtain as $M(i)$. For future computation, we also construct a bounding box around it, and call this bounded portion of the image $B(i)$. This notation is demonstrated in Fig. 1.

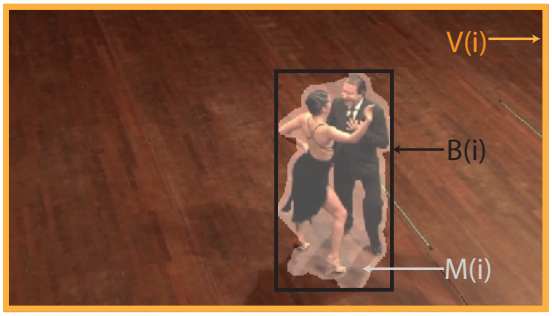

Figure 1: At each frame $V(i)$ we compute a foreground mask $M(i)$ and draw a bounding box $B(i)$ around it.

(a) (b) (c) (d)

Figure 2: **Similarity metric.** (a) and (b) Pair of bounding boxes containing the foreground mask (colored light grey). (c) Confidence mask computed using NRDC (colored light grey). (d) Foreground mask and confidence mask overlapped. Only the pixels contained in both masks (colored in the brightest grey) are used for the for computing the similarity.

4. Analysis – Choosing transition points

We analyze the input video V to choose the pair of frames (a,b) that we will use as a transition point. The choice of the transition point plays a key role in the quality of the output. To create the most natural looking video loops, we would like to find the pair of frames where the foreground is the most similar in terms of both appearance and motion. Our method for choosing this pair of frames is inspired by the work of Video Textures [SSSE00]. In particular, we choose this pair of frames by maximizing

$$(a,b) = \arg\max_{(i,j)} S_{app}(i,j) + S_{motion}(i,j), \qquad (1)$$

where the terms S_{app} and S_{motion} refer to the similarity in appearance and motion respectively, and are computed on the bounding box of the foreground in each frame, $B(i)$.

Appearance similarity There are different metrics to assess the similarity of the foreground of two frames. We can use pixel-wise metrics like the sum of squared differences (SSD) of color values, like [SSSE00], but restricted to the bounding boxes. However, these pixel-wise metrics are sensitive to transformations and deformations. A small inaccuracy of the segmentation mask might lead to a shift in the bounding

box, and thus a low similarity score, even if the foreground appearance is very similar. Even if the bounding boxes are well aligned, non-rigid deformations of the foreground (due to motion) may lead to low similarity scores.

Instead, we find correspondences between the foregrounds of two video frames and use it as a proxy for appearance similarity. This is based on the observation that the more similar the foregrounds are, the more correspondences we are likely to find between the frames. While there are many methods for finding such correspondences, in our work, we use the Non-Rigid Dense Correspondence (NRDC) algorithm [HSGL11]. We choose NRDC for several reasons: it is robust to changes in illumination, it is designed for pairs of images where only part of the content is shared and unmatched regions do not hurt the estimation of the correspondence, and it provides a confidence map that indicates which pixels have a correspondence. Given two bounding boxes $B(i)$ and $B(j)$, we compute NRDC and obtain a confidence map, that indicates whether a pixel in $B(i)$ has found a correspondence in $B(j)$. Our proposed similarity is the ratio of foreground pixels that are shared between the bounding boxes. We compute this as

$$S_{pair}(i,j) = \frac{\sum_x \gamma(B(x,i),B(x,j)) \odot M(x,i)}{\sum_x M(x,i)}, \qquad (2)$$

where the function $\gamma(\cdot)$ returns the correspondence confidence from the NRDC algorithm (0 indicates that there is no confident match, 1 if there is absolute certainty about the match), $\odot$ is the element-wise product and $M(x,i)$ is pixel x of mask $M(i)$. Bounding boxes are scaled to a resolution of 200×200 for this similarity measure. This computation is illustrated in Fig. 2.

To increase robustness and temporal coherence we seek transition points where a *sequence* of frames have high similarity. Similarly to [SSSE00], we capture similarity across a range of frames by summing it over a temporal window around a considered transition point:

$$S_{app}(i,j) = \sum_{-l \le k \le l} w(k) S_{pair}(B(i+k),B(j+k)), \qquad (3)$$

where w_k is a Gaussian weight with a standard deviation of 1, and $|l|$ is the size of the neighborhood (in our case $|l| = 5$).

Motion similarity S_{app} (Eq. 3) ensures similar appearance of the foreground across frames. This is usually sufficient to ensure temporal smoothness since often the camera follows the foreground, and therefore it stays in the same region of the image. However, S_{app} does not consider the global motion of the foreground relative to the background. This means that in a repetitive motion like a child jumping on a trampoline, a series of frames where the child is going upward could potentially be matched to a series of frames where the child is going downward. If we were to stitch the clips, this would lead to a non-realistic change in the global motion. The objective function should also capture the relative *rigid* motion of the foreground object with respect to the background.

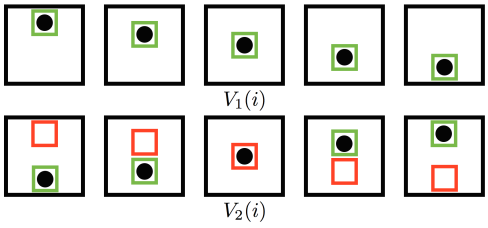

Figure 3: Given frames $V_1(i)$ and $V_2(i)$, if we used only the appearance similarity (Eq. 3) within the original bounding boxes (green boxes), we could potentially transition between two clips with different foreground motion, leading to a semantically wrong motion. Transforming the bounding boxes in the second clip according to the motion in the first (red boxes) and using Eq. 4 results in a low score in this case.

We include an additional term S_{motion}, that ensures the similarity of this global motion. We do this by transforming the bounding boxes in one of the videos to follow the relative motion of the other video, and computing the similarity using the new bounding boxes. For a pair of frames $V(i), V(j)$ we compute the transformation $\hat{T}$ that best aligns the foreground in $V(i)$ to the one in $V(j)$. We then transform the location of the bounding boxes B around frame j, i.e., $B(j \pm k)$, to compute the transformed bounding boxes, $B'(j \pm k)$. If the global motion of the foreground is similar in both videos, then the original bounding boxes $B(j \pm k)$ and the transformed ones $B'(j \pm k)$ will be similar, and therefore, $B'(j \pm k)$ will overlap largely with the foreground. However, if the motion in both clips is not similar, then $B'(j \pm k)$ will mostly contain background pixels. This process is illustrated in Figure 3. We use this transformed bounding box to compute the motion similarity between the two frames as:

$$S_{motion}(i,j) = \sum_{-l \le k \le l} S'_{app}(i+k, j+k), \qquad (4)$$

where S'_{app} computes the appearance similarity as in Equation 3 but using the transformed bounding box $B'(j)$ for the second frame. We use $l = 3$ frames to compute the motion similarity.

Optimizing the similarity function Optimizing Eq. 1 can be expensive, since it requires computing dense correspondences between every possible pair of frames. In order to make the method more efficient, we make two important approximations. First, most pairs of frames are actually bad candidates for a transition point. Since the similarity measure behaves smoothly around the good transition points (because of the averaging over multiple frames), we use a *coarse-to-fine* strategy to find the transition points. We evaluate S_{app} at every fifth pair of frames, and then choose

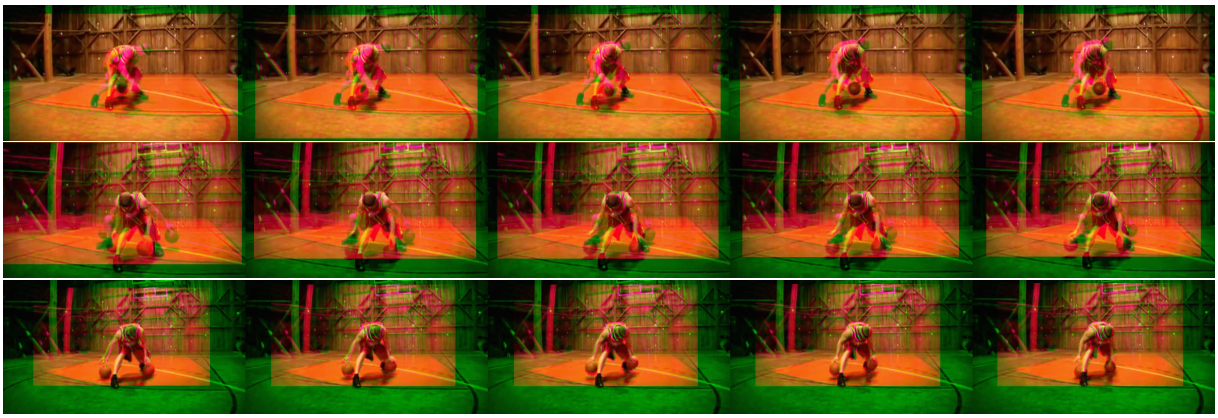

Figure 4: **Comparison of similarity metrics.** For different metrics, we show the chosen transition frames (center) with two preceding and subsequent frames, after rigid alignment. To visualize the quality of the alignment, each image is generated using the red and blue channels of one frame and the red channel of the other frame. Well aligned regions appear naturally colored, while misaligned regions appear oddly colored (green or pink). Using SSD over the whole frame (top) leads to results where the foreground is misaligned. Using SSD only in the bounding box (middle) is an improvement. The proposed method (bottom) is significantly better – the foreground is almost perfectly aligned at the cost of errors in the background.

the top 40 to explore at the finest temporal resolution. Second, computing S_{motion} is computationally expensive because it involves computing correspondences an order of $\mathcal{O}(n^3)$ number of times. To avoid such computations, we use a *two-step* process: we first compute the top 20 candidate pairs using S_{app}, and then we compute S_{motion} only on these. Finally, we avoid short loops by constraining the search to frames that are at least 20 frames apart (i.e., $|i-j+1| \geq 20$).

The total computation time varies depending on the size of the input video. In our experiments the average is around 2 hours. The average length of our videos is around 200 frames, typically of size 360×480. Most of this time is spent computing correspondences with NRDC, and thus a faster alternative would greatly improve the efficiency of our method.

We compare the performance of our foreground similarity metric with two baseline techniques on one of our examples. These techniques are: first, SSD computed on the entire frame [SSSE00], and second, SSD computed only in the masked region. These results are shown in Figure 4. Our method achieves significantly better spatio-temporal alignment of the foreground. Our technique for handling residual misalignments of the foreground, as well as differences in the background are discussed next.

5. Synthesis – Video Morphing

During the synthesis stage our method takes the transition point (a,b) $(b > a$, w.l.o.g.) resulting from optimizing Eq. 1, and produces an output video V_{out} that closes the loop seamlessly. Our goal is to ensure that the motion and appearance of the foreground change smoothly, while having a reasonable transition of the background. A good transition of both is in general not possible, and we focus on the foreground. This is based on the assumption that most of the viewer's attention is focused on the foreground, and, therefore, a smoother foreground transition will be more compelling, as long as the background does not change abruptly. In this section we describe the video morphing stage of our method. First, we align the videos using a global, coarse, rigid alignment of the foregrounds. Second, we use a local, detailed, non-rigid alignment of the foreground and background using correspondence guided patch-based synthesis.

5.1. Global rigid alignment via camera planning

Since our similarity measure is based on correspondences, it does not guarantee the alignment of the foregrounds in frames $V(a)$ and $V(b)$. We first handle gross global differences in the alignment by constructing a virtual camera path. As in Eq. 2, we compute a transformation $\hat{T}$, consisting of a translation and a scale, that best aligns these frames. To ensure that the transition is smooth, we spread this transformation over a window of frames after $V(a)$ and before $V(b)$. For this we fit a cubic spline $f(t)$, of length L and with zero first and second derivatives at the end points, for each of the three transformation parameters (2-D translation and 1-D scale), sample these splines to compute interpolated transformations, and apply them to the frames. The value of L is half the size of the loop to maximize the size of the window in which the transformation happens, and achieve a smoother transition.

5.2. Local non-rigid alignment via morphing

Applying the transformations to the start and end of the sub-clip gives us two sets of frames around the transition point that are rigidly aligned, but there might still be misalignments because of small changes in appearance or non-rigid motion in the scene. As a result, naïve approaches like concatenation or cross-fading would create artifacts like abrupt cuts or ghosting, respectively. Traditional morphing (TM) techniques use a combination of optical flow and cross-fading to alleviate this problem. However, our input videos are often noisy, and have large displacements, and this leads to poor flow results that, as illustrated in the results section, result in poor synthesis results.

Instead, we generalize the image-based Regenerative Morphing (RM) technique [SRAIS10] to video morphing. In RM, intermediate frames are synthesized using patches from two source frames while preserving local similarity to the sources and to consecutive frames for temporal coherence. RM has demonstrated good performance on different scenarios, ranging from interpolating nearby views to morphing entirely different images. However, it is designed to morph between images, where motions do not need to be realistic. We extend RM to the problem of morphing two roughly aligned video clips by adding two novel components:

Parabolic motion: RM incorporates correspondences between frames as morphing constraints. When correspondences exist, pixels in one source can be constrained to move on a linear path towards their corresponding location in the second source. In our case, the input contains complex camera motions and non-rigid scene dynamics (for e.g., moving people), where the linear motion assumption does not hold and leads to unrealistic dynamics. To tackle this problem, we generalize this method to use a *parabolic motion* for each pixel. A parabola is computed at each pixel by using correspondences between 4 frames. The motion of the pixel is generated by sampling along this parabola. As shown in Fig. 5, this leads to more realistic dynamics of the foreground in the morphed frames, compared to the linear constraints originally used in RM.

Separate foreground and background morphing: Another simplifying assumption in RM is to use a single morphing process for the entire frame. While this produces reasonable results when morphing images, in videos unnatural motions are very noticeable and unpleasant. We extend RM to contain two morphing processes, one for the foreground and one for the background. Each process models aspects specific to the foreground and background.

The *foreground morph* uses parabolic motion (since it is expected to contain non-rigid objects like people) and a short transition of fixed size (4 frames). This produces good results because our input videos are already aligned in the foreground, and the role of morphing is to improve small details. When there are not enough correspondences to compute a plausible morph, the foreground is simply concatenated. We use the NRDC confidence map to decide whether to use correspondences constraints or concatenation. More formally, we make this decision using Eq. 2 between the two reference frames used for foreground morph (frames *A* and *B* in Fig. 6), and setting a threshold of 0.6.

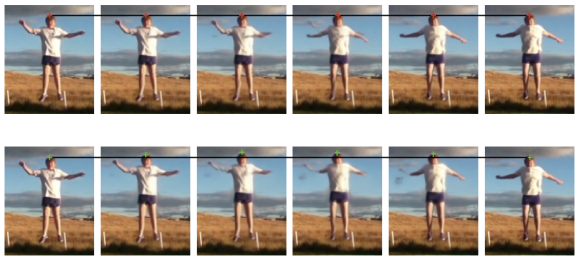

Figure 5: **Linear (top) vs. parabolic (bottom) constraints.** The transition happens around the peak point during the jump. The parabolic constraints capture the correct dynamics and lead to a natural up-down motion during the transition, while the linear constraints make the kid "freeze" in mid-air during the transition.

The *background morph* uses linear constraints for the motion. This is a better assumption for morphing rigid scenes from different viewpoints, which is often the content of the background. Also, correspondences in the background may be arbitrarily far. If the backgrounds of the two frames that we want to morph are close, a short transition is good. But if they are far, a longer transition is needed. We choose the window size of the transition automatically for the background, adapting to each video. The criterion we use is that the velocity of the camera (direction and magnitude) should change as little as possible when introducing the morph. Sometimes the camera moves fast and there are not enough correspondences in the background to compute a plausible morph. In this case, the morph uses *motion constraints* to move pixels at the same velocity as before and after the transition. We use the NRDC confidence map to decide whether to use correspondences constraints or motion constraints. As in the foreground morph, we make this decision using Eq. 2, this time between the two reference frames used for background morphing (*C* and *D* in Fig. 6). The integration of foreground and background morphing is described in Fig. 6. Once the foreground region is morphed, it is "pasted" on the background, or in other words, included as a hard constraint in the background morph.

6. Experiments

We test our method on a set of videos collected from the Internet containing a wide range of background and foreground dynamics (Figure 8). The quality of our results are easier to appreciate in the accompanying supplementary video. In order to evaluate both the similarity metric and

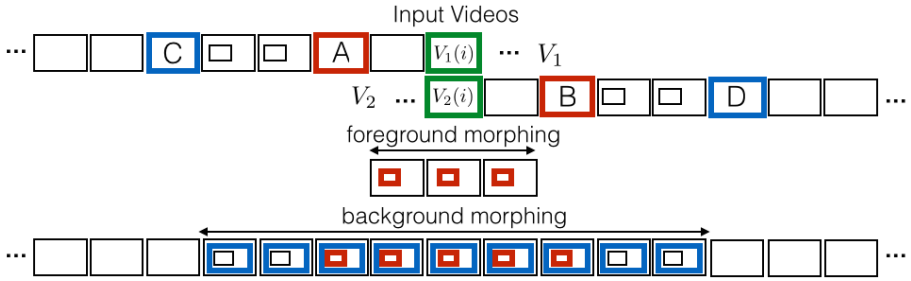

Figure 6: **Overview of video morphing**. The same video sub-clip is shown as two separate clips aligned at the transition point; the start of V_1 and the end of V_2 are the same, and we want to make the loop seamless at the transition point. Frames A through B are used to synthesize the foreground. A longer window – frames C and D – is used to synthesize the background, with the previously computed foreground as a constraint. This is signified by the small red boxes. For frames where the foreground is not synthesized (between $C-A$ and between $B-D$) the foreground constraint is simply the original foreground.

Figure 7: **Comparison of morphing techniques.** Top: Video Textures. Middle: Traditional morphing. Bottom: Proposed method. VT morphing is presented for the transition chosen by VT, while the other two show a morph for our transition.

video morphing components of our technique, we compare to three different combinations of analysis and synthesis methods. First, we compare to Video Textures (**VT**) because it is the most similar end-to-end method. To make it easier for VT, we stabilize the input to have no motion at all whenever the input video allows it. VT uses full-frame similarity to find transition points and a multi-way morphing technique to align the frames across these transition points. In unconstrained video sequences like ours, this similarity metric produces poor transition points leading to poor synthesis results. Second, to evaluate the quality of our similarity metric, we compute loops using the transition points computed using our similarity metric but align them using traditional morphing, i.e, warping using optical flow and blending using cross-fading (referred to as **TM**). The quality is better than VT because the chosen transition frames are better aligned, but the morphing itself leads to artifacts like ghosting in regions with erroneous optical flow. Our third comparison (referred to as **RM**) is to using the original RM algorithm in combination with our transition points and camera path planning. While RM does not have the same ghosting artifacts as TM, it was designed for two-frame morphing and is not able to capture the dynamics of the motion in our videos very well. In contrast, our video morphing – which extends RM with parabolic foreground motion constraints, linear motion or correspondence constraints in the background, and automatic selection of the morphing window – produces results that are visually more compelling. The differences in these techniques are much easier to appreciate in video form, and our supplementary material contains all these comparisons.

Violin This video contains an almost fixed camera, with detailed and structured foreground motions. Parabolic motion produces a more realistic result than traditional morphing.

Trampoline The small foreground causes a full-frame similarity metric to be dominated by the background. By focusing on the foreground we produce a better transition point. In addition, video morphing the background is challenging because the scene moves quickly; we use motion constraints to maintain the same motion during the morph.

Basketball The large scale changes in this video are handled by our foreground-based similarity metric and camera path. Our video morphing technique is best at handling the complex motions (hands and basketballs) because of the parabolic motion constraints.

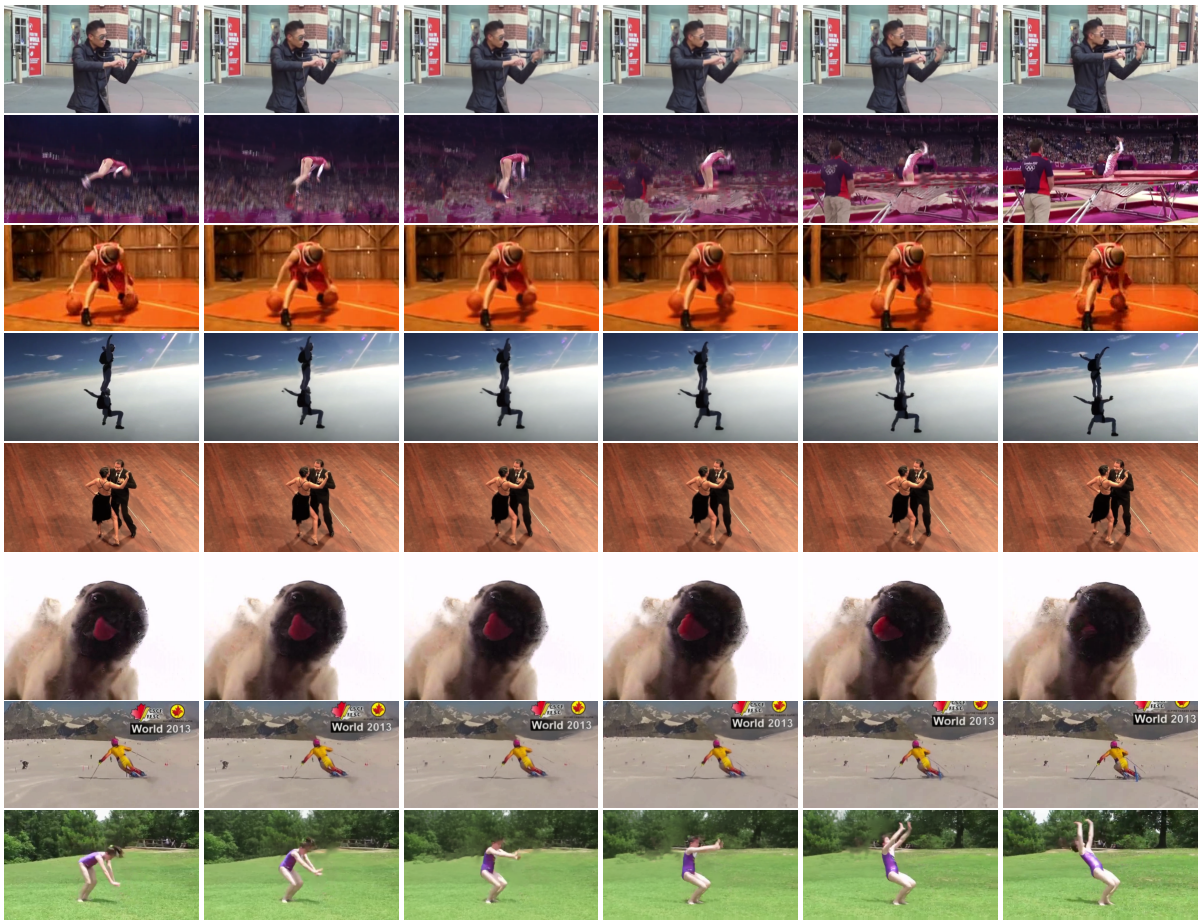

Figure 8: **Morphed results.** Results of our technique on a wide variety of examples. From top to bottom, **violin**, **trampoline**, **basketball**, **skydiving**, **tango**, **dog**, **ski-slalom**, and **back-flip**. See supplementary video for input and resulting loopy videos.

Skydiving Our foreground-based similarity metric produces better transitions. Also, traditional morphing fails because flow estimation is unreliable due to low texture regions.

Tango In this video, the background moves very slowly while the foreground moves faster. Our method automatically chooses a large window size of 17 frames to morph the background, allowing the camera motion to stay slow and smooth.

Dog In this video there is very small overlap of the foreground, in frames that are well separated, and VT picks a poor pair of transition frames. This video also illustrates the importance of the virtual camera path. Using TM, the foreground is morphed and displaced simultaneously, creating an unrealistic effect.

Ski-slalom The frame is dominated by the background, and VT does not find good transition points that align the skier well. Our method is able to produce a nice loop of the skier using parabolic motion, while interpolating the mountains in the background using linear motion constraints.

Back-flip This video depicts a gymnast moving in front of a rapidly changing background. While our method captures the natural motion of the gymnast, the synthesized background has artifacts because of the large differences between the start and end of the clip.

Trampoline-Kid The motion of the bouncing kid in this example leads to ghosting artifacts in the VT and TM results, and an unnatural freeze in mid-air in the RM result which are not present in our results.

7. Limitations

We designed our technique to be fully automatic for real-world cases by making assumptions about the scene. We assume the foreground is located in one contiguous region and has roughly consistent motion which is different from the

background. We also assume that the background has similar visual properties in the two clips, failing which morphing produces artifacts; the **back-flip** result is our worst result in that respect. Second, our technique relies on a few automatic computer vision components – i.e., foreground segmentation [PF13] and correspondences [HSGL11] – that may not always work. The most fragile component of our technique is the automatic foreground segmentation. In the supplementary material we also include results using manual segmentation to illustrate the possible improvement.

8. Conclusions and Future Work

In this paper, we have described an automatic method for creating infinitely looping clips from casual videos. Our method is capable of handling complex camera and foreground motions, and we have demonstrated this on a variety of unconstrained videos downloaded from the internet. Our method can be easily generalized to stitch two input videos instead of one, if the two clips capture the same scene under a similar viewpoint. This could be used for many applications, like video editing (a generalization of [BLA12]), view interpolation [BBPP10], or video summarization (e.g., summaries from multiple videos of the same event [APS*14]). In addition, different components of our pipeline could be changed for specific applications. For e.g., the similarity metric could be modified to evaluate only motion similarity, instead of motion and appearance. This would allow stitching videos with similar dynamics but different appearances, such as different people doing the same dance or practicing the same sport.

References

[APS*14] AREV I., PARK H. S., SHEIKH Y., HODGINS J., SHAMIR A.: Automatic editing of footage from multiple social cameras. *ACM Trans. on Graph. (Proc. SIGGRAPH) 33*, 4 (July 2014), 81:1–81:11. 9

[AZP*05] AGARWALA A., ZHENG K. C., PAL C., AGRAWALA M., COHEN M., CURLESS B., SALESIN D., SZELISKI R.: Panoramic video textures. *ACM Trans. on Graph. (Proc. SIGGRAPH) 24*, 3 (July 2005), 821–827. 2

[BAAR12] BAI J., AGARWALA A., AGRAWALA M., RAMAMOORTHI R.: Selectively de-animating video. *ACM Trans. on Graph. (Proc. SIGGRAPH) 31*, 4 (July 2012), 66:1–66:10. 2

[BBPP10] BALLAN L., BROSTOW G. J., PUWEIN J., POLLEFEYS M.: Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Trans. on Graph. (Proc. SIGGRAPH) 29*, 4 (July 2010), 87:1–87:11. 9

[BLA12] BERTHOUZOZ F., LI W., AGRAWALA M.: Tools for placing cuts and transitions in interview video. *ACM Trans. Graph. 31*, 4 (July 2012), 67:1–67:8. 2, 9

[Car] CARRÉ L.:. URL: http://lillicarre.tumblr.com/. 1

[DCWS03] DORETTO G., CHIUSO A., WU Y., SOATTO S.: Dynamic textures. *IJCV 51*, 2 (2003), 91–109. 2

[FS11] FREEMAN J., SIMONCELLI E. P.: Metamers of the ventral stream. *Nature neuroscience 14*, 9 (2011), 1195–1201. 1

[Gol07] GOLDMAN D. R.: *A framework for video annotation, visualization, and interaction*. PhD thesis, University of Washington, 2007. 2

[Gom99] GOMES J.: *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999. 2

[HSGL11] HACOHEN Y., SHECHTMAN E., GOLDMAN D. B., LISCHINSKI D.: Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. on Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 70:1–70:9. 2, 4, 8

[Inc] INCI E.:. URL: http://erdalinci.tumblr.com/. 1

[JMD*12] JOSHI N., MEHTA S., DRUCKER S., STOLLNITZ E., HOPPE H., UYTTENDAELE M., COHEN M.: Cliplets: Juxtaposing still and dynamic imagery. In *Proc. UIST* (2012), pp. 251–260. 2

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

[KSSGS11] KEMELMACHER-SHLIZERMAN I., SHECHTMAN E., GARG R., SEITZ S. M.: Exploring photobios. *ACM Trans. on Graph. (Proc. SIGGRAPH) 30*, 4 (July 2011), 61:1–61:10. 2

[LJH13] LIAO Z., JOSHI N., HOPPE H.: Automated video looping with progressive dynamism. *ACM Trans. on Graph. (Proc. SIGGRAPH) 32*, 4 (July 2013), 77:1–77:10. 2, 3

[LLN*14] LIAO J., LIMA R. S., NEHAB D., HOPPE H., SANDER P. V.: Semi-automated video morphing. *Comp. Graph. Forum (Proc. EGSR) 33*, 4 (2014), 51–60. 2

[LTK12] LEVIEUX P., TOMPKIN J., KAUTZ J.: Interactive viewpoint video textures. In *Proc. CVMP* (December 2012). 3

[PF13] PAPAZOGLOU A., FERRARI V.: Fast object segmentation in unconstrained video. *ICCV* (2013). 3, 8

[Raj] RAJKOVIC M.:. URL: http://milosrajkovic.tumblr.com/. 1

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: GrabCut: Interactive foreground extraction using iterated graph cuts. In *ACM Trans. on Graph. (Proc. SIGGRAPH)* (2004), pp. 309–314. 3

[RWSG13] RÃIJEGG J., WANG O., SMOLIC A., GROSS M.: Ducttake: Spatiotemporal video compositing. *Comp. Graph. Forum 32*, 2pt1 (2013), 51–61. 2

[SRAIS10] SHECHTMAN E., RAV-ACHA A., IRANI M., SEITZ S. M.: Regenerative morphing. In *CVPR* (2010), pp. 615–622. 2, 6

[SRDB10] SUN D., ROTH S., DARMSTADT T., BLACK M.: Secrets of Optical Flow Estimation and Their Principles. *CVPR* (2010).

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video Textures. In *ACM Trans. on Graph. (Proc. SIGGRAPH)* (2000), pp. 489–498. 1, 2, 3, 4, 5

[TPSK11] TOMPKIN J., PECE F., SUBR K., KAUTZ J.: Towards moment images: Automatic cinemagraphs. In *Proc. CVMP* (November 2011), pp. 87–93. 2

[Vin14] Vine statistics, Aug 2014. URL: http://blog.vine.co/post/95288683756/new-vine-camera-shoot-import-edit-and-share. 1

[ZCA*09] ZHENG K. C., COLBURN A., AGARWALA A., AGRAWALA M., CURLESS B., SALESIN D., COHEN M.: Parallax photography: Creating 3D cinematic effects from stills. In *Graph. Interface 2009* (May 2009). 2