Computational Visual Media
https://doi.org/10.1007/s41095-020-0187-3 Vol. 6, No. 3, September 2020, 291–306
Research Article

Coherent video generation for multiple hand-held cameras with dynamic foreground

Fang-Lue Zhang[1] (✉), **Connelly Barnes**[2], **Hao-Tian Zhang**[3], **Junhong Zhao**[1], **and Gabriel Salas**[1]

Abstract For many social events such as public performances, multiple hand-held cameras may capture the same event. This footage is often collected by amateur cinematographers who typically have little control over the scene and may not pay close attention to the camera. For these reasons, each individually captured video may fail to cover the whole time of the event, or may lose track of interesting foreground content such as a performer. We introduce a new algorithm that can synthesize a single smooth video sequence of moving foreground objects captured by multiple hand-held cameras. This allows later viewers to gain a cohesive narrative experience that can transition between different cameras, even though the input footage may be less than ideal. We first introduce a graph-based method for selecting a good transition route. This allows us to automatically select good cut points for the hand-held videos, so that smooth transitions can be created between the resulting video shots. We also propose a method to synthesize a smooth photorealistic transition video between each pair of hand-held cameras, which preserves dynamic foreground content during this transition. Our experiments demonstrate that our method outperforms previous state-of-the-art methods, which struggle to preserve dynamic foreground content.

Keywords video editing; smooth temporal transitions; dynamic foreground; multiple cameras; hand-held cameras

1 Victoria University of Wellington, Wellington 6012, New Zealand. E-mail: F.-L. Zhang, fanglue.zhang@vuw.ac.nz (✉); J. Zhao, junhong.zhao@vuw.ac.nz; G. Salas, gabrielsalas@vuw.ac.nz.
2 Adobe Research, Seattle, USA. E-mail: cbarnes@adobe.com.
3 Stanford University, San Francisco, USA. E-mail: haotianz@cs.stanford.edu.

Manuscript received: 2020-06-30; accepted: 2020-07-16

1 Introduction

With modern camera technology, people can easily capture high-quality videos with hand-held cameras and smartphones. The process of capturing daily events and sharing them on social networks or storing them privately has become an important part of the everyday life of many people. However, the photographers making such casual captures often have little control of the scene. Additionally, the photographer is often an amateur, who may lack photographic skills, attention, or time, and thus may fail to capture the desired object or the full time range of the event. Thus, if a viewer is watching a video from any single camera, he/she may find that the foreground object he/she is interested in simply moves out of the camera, or the camera may stop capturing entirely at a time that is not ideal. These issues can be quite frustrating for the viewer. However, for many cases such as in public events or performances, multiple cameras may capture the same event. If a foreground object of interest moves out of the frame of one camera, then the object may still be captured by another capture. In this paper, we investigate the problem of producing a single coherent video with smooth transitions by taking as input such a casually captured multi-camera feed. Specifically, we assume that the input consists of different videos of the same dynamic foreground objects that were captured at different but overlapping ranges of time.

Researchers in computer vision and computer graphics have proposed powerful video processing and editing methods that are related to this problem. To spatially combine videos from different cameras, video stitching methods have been explored for years. Recently, stitching methods based on sparse 3D information reconstruction [1, 2] and dynamic

foreground separation [3] have achieved good results even when the hand-held cameras are obviously shaking. For scenes captured by multiple social cameras, Arev et al. [4] proposed a temporal domain video editing algorithm, which can produce a video consisting of several shots from multiple social cameras. However, existing tools cannot achieve our goal of generating a smooth photorealistic video of a moving object captured by different cameras at different time. For example, the method of Ref. [4] simply provides an optimal cut between different shots rather than a smooth transition.

Achieving our goal is challenging because some 3D information about camera and object positions is required to synthesize a plausible output video, but complete 3D information may be missing, especially when the dynamic foreground objects are not simultaneously captured in all of the input videos. Thus, one cannot always use triangulation to reconstruct 3D information for the foreground objects seen at each frame. This challenge makes previous free-view video generation methods relying on a structured array of well-calibrated cameras [4–6] all fail to synthesize satisfactory results for our problem.

In this paper, we focus on how to combine videos containing dynamic foreground objects, where the captures are not only spatially but also temporally complementary to each other. Due to the moving foreground content, this is a challenging problem that requires us to use both 3D reconstruction and foreground–background separation. As shown in Fig. 1, our method does not require the target objects be captured by all the cameras for the whole sequence. Given the target objects, we first use a graph-based method to find the optimal transition route among all videos. In the edges of our graph, we encode the information of which hand-held camera videos we should transit between and the starting and ending frames of each transition. Next, we propose a method to synthesize a smooth transition video between each pair of hand-held camera videos. In our synthesized transition video, the starting frame matches the same frame of the first camera, and the ending frame matches that of the second camera. In this way, our method finds suitable video segments from all different cameras, and smoothly transits from one camera which contains the target object to another. We can then synthesize a final continuous video sequence for the foreground object of interest.

2 Related work

We discuss related work within three areas: video stitching, free-viewpoint video, and algorithms that allow video editing and navigation in the temporal domain.

Video stitching. Our approach is highly related to video stitching algorithms. Early research in this area did not focus on the temporal continuity of video content. To combine content from different cameras, these papers instead focused on how to stitch images. One representative method by Szeliski and Shum [7] worked by estimating the global transformation between image pairs and then used warping to create panoramas or mosaics. El-Saban et al. [8] also demonstrated real-time stitching between mobile phone videos using a global transformation. In order to handle displacements due to parallax between the

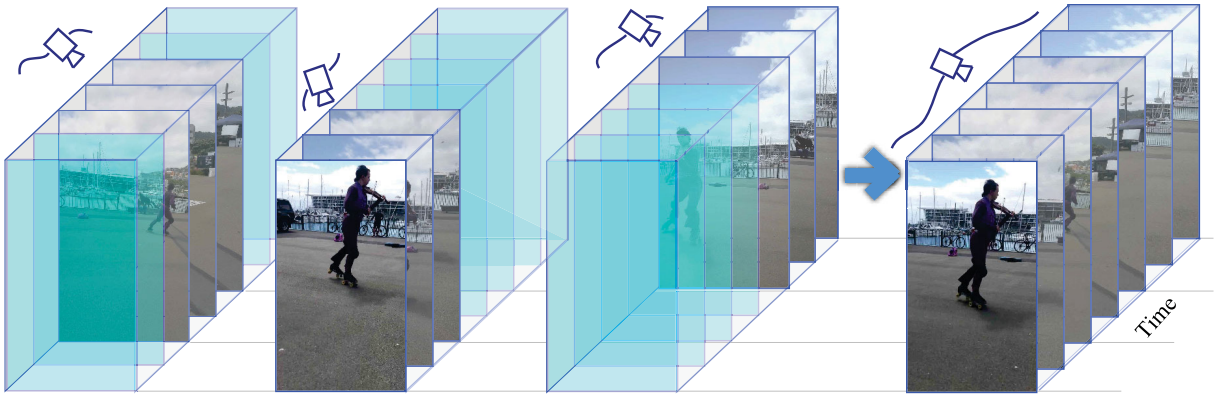


Fig. 1 We propose a method to synthesize a smooth video sequence from multiple videos captured by hand-held cameras. The captured footage may contain dynamic foreground objects such as the human performer shown above.

images, locally varying warping methods have also proved to be effective [9–11]. Such local warping approaches are incorporated in state-of-the-art video stitching and transition methods, including ours. Based on 3D content-preserving warping [12], Lin et al. [2] generated the positions of control points in a stitched video, which were used to warp each input video according to new camera paths. Guo et al. [1] also used both stitching and stabilization objectives in an optimization process to obtain a more aesthetically pleasing stitched video. Unlike our method, Guo et al. did not consider inputs which have moving foreground objects at the stitching boundary. To avoid ghosting, both Guo et al. [1] and Lin et al. [2] prefered to place on top or "overlay" a region from one input video that contains a complete moving object. For our problem, however, such an overlay does not work, because we would like to create a smooth transition between two videos rather than an abrupt change. In the work of Nie et al. [3], the dynamic foreground is processed by the method of Zhang et al. [13] to estimate the transformations between frames. We show in our experiments that recent methods such as Lin et al. [2] and Nie et al. [3] struggle to combine dynamic foreground objects from different videos.

Instead of using global or local transformations to align image pairs, Kwatra et al. [14] used graph cuts to obtain seamless boundaries between stitched video pairs. This idea was extended to 3D to obtain spatio-temporally stitched panoramic video [15]. But these works simply treat the video content as textures, which is not suitable for structured moving objects. There are also recent works focusing on how to improve the performance and quality of stitching for 360° videos, such as for virtual reality [16, 17]. These methods are not designed to deal with independently moving, hand-held cameras.

Free-viewpoint video. Researchers have explored using videos captured from camera arrays to generate free-viewpoint videos. Some of these papers rely on a group of well-calibrated cameras to generate synthetic views [4–6], or focus on view-consistent feature analysis for binocular videos [18]. For human actors, Wei and Chai [19] obtained very high quality 3D motion sequences. But because the cameras are static for Wei and Chai, that method cannot be used for our setting of handheld video. Ballan et al. [20] proposed a method to interpolate between monocular videos of a dynamic scene, with unstructured cameras. However, Ballan et al. assumed that the dynamic objects exist simultaneously in all the input videos. Our method can handle inputs where the foreground content of interest is not present in all cameras at any given time. Tompkin et al. [21] created a system to organize unstructured videos of related content, and apply 2D and 3D transitions between them. 360° panoramic video is capable of providing omni-directional view directions for users [22, 23]. Researchers have focused on how to produce high-quality 360° videos by stitching the videos captured by camera arrays [24, 25] or an unstructured group of cameras [26]. However, their methods all require that the videos cover the full temporal and spatial range of the scene.

Video editing in the temporal domain. Researchers have also focused on how to edit video in the temporal domain to help users navigate content and obtain more pleasing output. Arev et al. [4] automatically produced a single video by finding the best cut points among several social videos taken of the same event. Like we mentioned in our paper, Arev et al. used a graph-based method to select the cut points that produce pleasing transitions. We have a different construction for our graph, because our transitions are designed to be smooth and unnoticeable. Wang et al. [27] proposed a method of interactive synchronization of multiple videos. More recently, Cui et al. [28] utilized the video captured in different temporal ranges during one day to generate a time slice video. Navigation of video content along the temporal axis has also been investigated by a "video tapestries" paper [29]. More recently, Zhang et al. [30] proposed a method for integrating multi-video object synopsis with the display method of optimal multi-view switching. Wang et al. [31] developed a method to generate a wider field of view (FOV) for selfie videos by combining a normal selfie video and a background video with much wider FOV. These methods do not consider the problem of synthesizing a continuous output from input videos with moving foreground objects.

3 Overview

Our system has three stages as shown in Fig. 2. In the first stage, we preprocess the input videos, which may contain multiple temporal regions where the

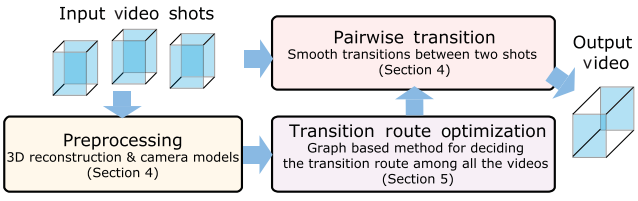


Fig. 2 Our system has three stages. The input videos are preprocessed and then fed to the graph based transition route optimization. On the optimal graph route, a smooth transition is computed on each edge between two video shots. Finally a coherent video of the foreground object is generated with seamless transitions between videos from different hand-held cameras.

foreground objects are visible. The preprocessing involves performing temporal synchronization, then automatic foreground–background separation [13], followed by running 3D reconstruction [32, 33] on only the background features.

In the second stage of our system (Section 5) we construct a graph that allows us to automatically select a pleasing transition route between the individual shots in our video collection. We obtain this good transition route by finding an optimal route through the graph. This stage can be useful for automatically producing a single video from a set of socially captured videos of interesting foreground content. However, this stage can also be performed manually by a human carefully selecting transition points in a video editing software, so the automatic transitions selected by the second stage are optional.

In the third stage of our system (Section 6), we create a smooth transition between a given pair of input video shots V_i and V_j, by applying separate local warps to the background and foreground content. The preprocessing has already given us a foreground–background separation, and a sparse 3D point cloud as well as the camera parameter sequences for the two videos. Using the recovered camera models in each frame, we find a temporal transition for each foreground object. We do this by first reconstructing the 3D positions of the sparse feature points on the foreground objects using the recovered camera models and the matching feature points in the pairs of input frames. We generate a new camera path for the transition frames, apply local warpings to the foreground and background separately, and then apply a spatiotemporal graph-cut to generate a smooth transition for the foreground region. Finally, the foreground and background can be combined together to generate the final transition. By using the transition videos to connect the original video shots, we can obtain a final smooth synthesized video that contains the foreground objects.

4 Preprocessing

Our preprocessing is visualized in the left of Fig. 4. We use the same preprocessing for both the transition route optimization (Section 5) and the pairwise smooth transition (Section 6). Preprocessing involves first temporally synchronizing the input videos, then performing automatic foreground–background separation, and then running 3D reconstruction on only the background features. We now discuss them in order.

Temporal synchronization. In order to run our method, we need the input videos to be synchronized temporally. For the videos captured by smartphones, we wrote our own video capture APP to directly record the time stamps of the videos for synchronization. In our APP, we obtain the Unix timestamp in milliseconds when we start and end the video recording function. Due to the latency of 4G networks, the errors are usually 15–75 ms, which are visually acceptable for objects moving in a normal speed. From the earliest frame to the last frame among all the videos, we denote the time stamps by $t_0, t_1, ..., t_n$. If there are multiple frames at time t_k from different videos, we denote all these frames by F_k^j, where j is the identifier of the input camera.

Foreground–background separation. We require users to specify the foreground objects they are interested in by applying a few strokes using the Roto Brush tool provided in the commercial video processing software Adobe After Effects. Based on the occurrence of the specified foreground objects, we trim the captured videos to the specified time ranges, and collect a series of video shots that contain the objects of interest. We denote the i-th video shot that contains the object of interest in our collection as V_i. These are visualized as orange bars in Fig. 3. With the initial masks provided by Roto Brush, we use the method of Zhang et al. [13] to segment the videos into foreground video F_i and background video B_i.

Sparse 3D reconstruction. Due to the moving foreground objects, if we use all feature points in 3D reconstruction, this will not give a reliable result. Thus, we use only the background feature points to run a Structure from Motion (SFM) algorithm [33]

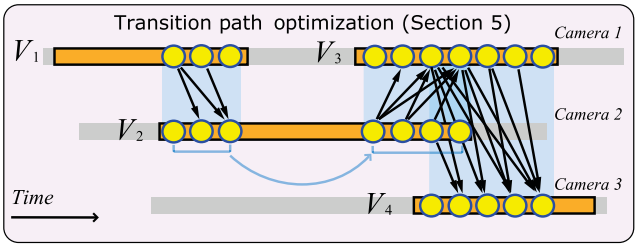


Fig. 3 In our system, a graph model can be used to automatically select a pleasing transition route such that the resulting synthesized video shows the desired foreground objects. Here grey regions denote captures from each camera, orange regions denote shots that contain foreground objects of interest, and black arrows represent possible transitions.

so we can obtain better camera models. This gives us 3D point clouds for the backgrounds of the videos input to SFM.

We also need to specify the videos that are input to SFM. For the transition route optimization (Section 5), we first identify all time ranges where two or more videos see the foreground content of interest, and then for each such time range, trim all such videos to the time range and feed them through SFM. For the pairwise smooth transition (Section 6), we simply apply SFM to the pair of videos.

5 Transition route optimization

In this section, we explain the optional second stage in our system. This stage automatically selects a pleasing transition route between the different captured shots of the foreground content. Our aim is to find which video or which pair of videos we should use for each frame. It is difficult to fully reconstruct the 3D video content of a target foreground object from only hand-held camera input. Thus, to obtain a continuous synthesized video, we simply smoothly transit from one captured video to another. In that way, we just need to synthesize a transition video sequence to connect any given pair of videos. We automatically find a reasonable transition route by modeling this as a graph problem. Arev et al. [4] also utilized a graph-based approach to handle their videos from social cameras. However, their method is not able to directly performed to our problem, because their aim is just finding optimal cutting points, while we need a series of continuous frames for seamlessly transition. We find the optimal route from the beginning video shot to the ending shot. This tells us for each frame of the final synthesized video, which input video shot we should draw from, and for which time intervals we should create a smooth transition.

5.1 Graph nodes and edges

We show our graph in Fig. 3. As previously described, we firstly run the preprocessing steps, which result in a temporal synchronization between the video frames as well as a 3D reconstruction of background features. In our graph, for every video frame where the foreground objects of interest can be seen by two or more cameras simultaneously, we assign a node for each frame of the corresponding input videos. These nodes represent potential transition points between different videos: these are depicted as yellow circles in Fig. 3.

Next, we add directed edges, which represent either potential transitions or direct playback of the video. To create potential transitions, we connect nodes from earlier video shots V_i to nodes in video shots V_j with later starting time, if three criteria are met: (1) the nodes belong to different videos; (2) the foreground objects of interest are visible in both videos at all time during the transition; and (3) the transition lasts a minimum time of τ. We use $\tau = 1/3$ s as a default setting. The edge weights indicate the smoothness of the given candidate transition. For nodes in the same video, we also add direct edges with no cost: these are depicted as blue arrows in Fig. 3. These zero-cost edges impose no penalty for directly playing back unaltered input camera footage, and allow subsequent transitions to be linked together.

By finding a route from the first frame of the earliest video to the last frame of the last video, we can obtain a candidate sequence that always contains the foreground objects of interest. The summed weights along this route serve as a measurement of the smoothness of all transitions along it. We now give more details about the edge weights and the transition route selection for our graph.

5.2 Graph edge weights

Each weighted edge E in our graph represents a possible transition from a source to a target video shot, which connects from the frame F_{s}^{i} associated with the source video shot i to the ending frame F_{e}^{j} of the target video shot j. Thus, in each candidate transition, we analyze the similarity of all corresponding frame pairs between the starting

and ending times t_s and t_e of the transition to obtain weights, which we use to measure the smoothness of the potential transition. We first apply the preprocessing steps of Section 4, including the foreground–background separation and SFM. We then align the background regions of the frames involved in each candidate transition by using 3D content-preserving warping [12] to align the later video shot by its background features to the earlier video shot. During our experiments, we find that transition smoothness can be measured based on three criteria: similarity of color distributions, similarity of cameras, and large overlapping region size. We now discuss these.

- *Color distribution.* Because of the different settings of white-balance, exposure, and other sensor properties among different hand-held cameras, the color distribution for the same content in different videos can have large variation. Even within the same video, color distributions for the same foreground and background content may change due to effects such as auto-exposure. To compensate for this, we use the 3-channel histogram equalization as a preprocessing step to attempt to align the colors of shots. We then measure the color distribution similarity using the histogram intersection method [34] for all aligned frame pairs throughout the candidate transition, in the spatial regions where the two videos overlap. We denote the color similarity for corresponding frame pairs at time t_k by $C_k^{i,j}$, where i and j are the video shot IDs.
- *Camera pose.* Multiple hand-held cameras only provide limited numbers of different camera poses for the same dynamic scene. Thus, we cannot assume that we can recover 3D videos associated with arbitrary positions or orientations of a virtual camera. When warping to generate a new frame, it would be better if the camera parameters of the two frames are similar. We measure this by computing the distance of the extrinsic camera parameters recovered from SFM. We only consider the difference of the camera orientation $\overrightarrow{o} = (\theta, \phi, \psi)$ and position $\overrightarrow{p} = (x, y, z)$. We calculate a camera pose similarity by negating the Euclidean distances with different weights:
$$P_k^{i,j} = -\omega_1 \widetilde{D}\left(\overrightarrow{o}_k^i, \overrightarrow{o}_k^j\right) - \omega_2 D\left(\overrightarrow{p}_k^i, \overrightarrow{p}_k^j\right)$$
We set ω_1 as 0.7 and ω_2 as 0.3 by default, because angular differences can make warping-based frame synthesis more difficult. Here $\widetilde{D}$ indicates that the pairwise sum-of-squared distances between each of the three Euler angles is computed by taking the minimum angular distance either clockwise or counterclockwise. Also, D indicates squared Euclidean distance (L_2).
- *Overlapping region size.* When we perform content-preserving warping to align one video to another, it is beneficial to have a larger overlapping region. This will create fewer visual artifacts when transiting from the content of one video to another. Thus, we use the ratio of the overlapping region to the original frame size as a term to measure the smoothness for the transition. We denote this as $R_k^{i,j}$.

The above three terms are calculated separately for each frame pair. The final edge weight for edge E is defined as
$$W^E = \delta^E + \frac{1}{e-s}\sum_{k\in[s,e]}(\alpha\widehat{C}_k^{i,j} + \beta\widehat{P}_k^{i,j} + \gamma\widehat{R}_k^{i,j}) \quad (1)$$
For the entire collection of input videos, the terms $C_k^{i,j}$, $P_k^{i,j}$, and $R_k^{i,j}$ are each statistically normalized over the collection to have a range of [0, 1]; $\widehat{C}_k^{i,j}$, $\widehat{P}_k^{i,j}$, and $\widehat{R}_k^{i,j}$ are the corresponding normalized variables. Here α, β, and γ are set to $0.4, 0.4, 0.2$, respectively.

In our experiments, we also find that if there are too large differences in camera poses or too small overlapped regions, the generated results will have obvious artifacts. Thus, for every edge, we compute the term δ^E, which can prevent choosing such edges E in the final generated route. Here δ^E is set to $-\infty$ if the average angle between the two cameras is larger than 20 degrees, or the overlapped region is less than 30% of the original frames for any frame. Otherwise, δ^E is set to zero.

5.3 Graph route selection

Given our graph, we now need to find the optimal route that represents the most feasible transitions for final synthesized video. Because the weights represent the smoothness of transitions, we prefer routes with high weight sums. Starting with the earliest video shot, we run depth-first search over all edges with finite weight, to locate the last shot connected to it by a route. We then select the route with the largest sum of weights that connects these first and last shots. This can be computed by solving the shortest route problem on the negated weights.

6 Pairwise smooth transition

In this section, we propose a method to create a smooth transition between two temporally-aligned videos with dynamic foreground content. As a reminder, the whole pipeline is shown in the right of Fig. 4. We are given a source video shot to transition from and a target video shot to transition to. For simplicity of notation, in this section, we assume that the source video is V_1 and the target video is V_2. We are also given a starting time t_s and an ending time t_e for the transition. During this time, the foreground objects of interest are assumed to be visible in both cameras. From the preprocessing stage of Section 4, we obtain the foreground segmentation and camera models for each pair of frames during the transition. Based on this information, we generate a frame sequence that smoothly transitions from video V_1 to V_2. We create this transition by first generating a new camera path for the transition frames (Section 6.1), then applying local warpings separately to the background (Section 6.2) and foreground (Section 6.3.1), then using a spatiotemporal graph-cut to make a smooth transition for the foreground (Section 6.3.2), and finally using additional processing to finalize the result (Section 6.4).

6.1 Transition camera path optimization

We assume the camera paths for video V_1 and V_2 are denoted as C_1 and C_2, respectively. As shown in Fig. 5, we would like to find a camera path that smoothly transitions from C_1 to C_2. This transition should happen within the time range that the foreground object can be seen in both videos. We call the resulting new camera path C', which is shown in green in Fig. 5. We temporally interpolate camera parameters between the start and end of the transition. The 3D camera pose is represented by the projection matrix P, which consists of an intrinsic matrix K, a rotation matrix R, and the camera center c. The projection matrices for cameras C_1 and C_2 at time t are

$$P_1^t = K_1[R_1^t, -R_1^t c_1^t],\ P_2^t = K_2[R_2^t, -R_2^t c_2^t]$$

Following previous work [2], we interpolate the camera projection matrices by linear interpolation as

$$P^t = \frac{t - t_s}{t_e - t_s} P_2^t + \left(1 - \frac{t - t_s}{t_e - t_s}\right) P_1^t \qquad (2)$$

To avoid shaking in the final camera path caused by errors in camera model estimation, we also use a temporal median filter to smooth every parameter of the resulting projection matrices.

6.2 Background reconstruction and warping

We generate our final video by compositing foreground objects on the background. For this reason, we would like the background to be completed separately from the foreground. We can identify the regions in the background that need to be completed by using the foreground masks recovered in our preprocessing stage. Then we can complete the background region by Newson et al. [35]. With the generated camera models for each frame in the transition, we project all the background 3D feature points to the new camera's plane. Then we use 3D content-preserving warping [12] to align the recovered background frames to obtain the novel background

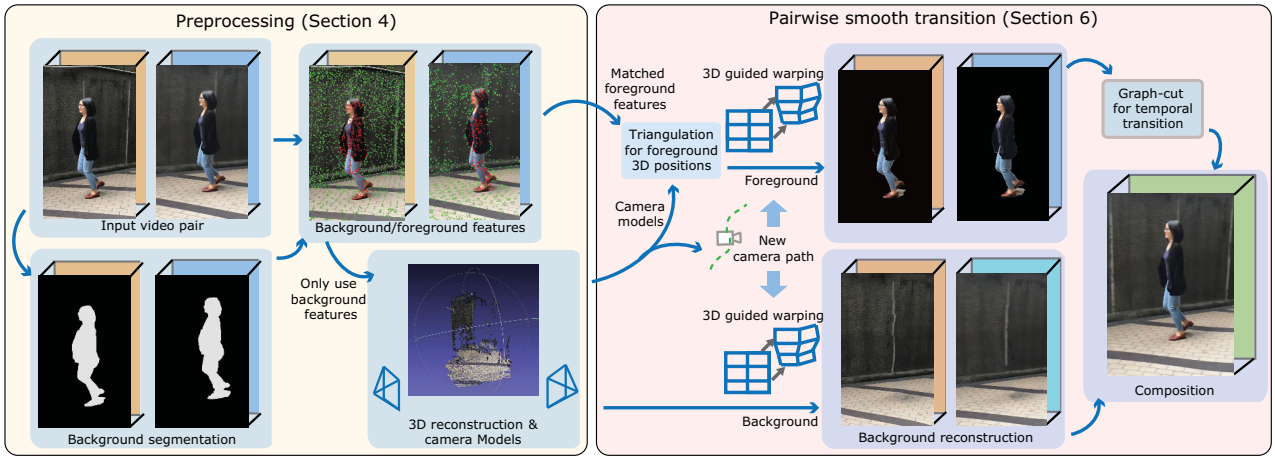


Fig. 4 Here we give an overview of both our preprocessing stage (Section 4) and the latter step of synthesizing a continuous transition (Section 6) between a given pair of input videos.

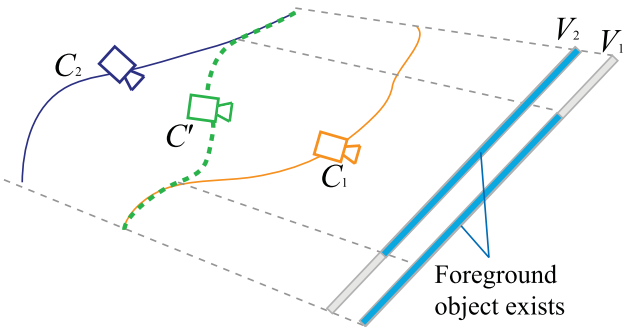


Fig. 5 We would like to find a new camera path C' (shown in green) which can transition from camera C_1 (shown in orange) to camera C_2 (shown in blue). For the video generated along camera C', the content should be the same as video shot V_1 at the start of when the foreground object is visible in both cameras and match video V_2 at the end of this period.

content with the new camera parameters. One example is shown in Fig. 6.

6.3 Foreground temporal transition

We also need to separately synthesize a plausible foreground region that contains dynamic content. We do this in two steps: we first warp the foreground regions in the input videos V_1 and V_2 to align them, and then we use a spatiotemporal graph cut to find good transition points for the foreground region.

6.3.1 Foreground warping

To help improve the reliability of the 3D positions of the foreground feature points, we already constrained the transition to happen when the foreground region is visible in both videos. This allows us to find correspondences between the same foreground object captured at the same time from two different views. For the foreground region, we detect SIFT feature points [36] and match them between each frame pair, and recover the 3D positions using triangulation. We also align the foreground objects from the first video V_1 and the second video V_2 to the new camera path C' by using 3D content-preserving warping [12]. To improve the visual quality, we use the guided image filter [37] to further refine the foreground masks after warping. Comparing with deep learning based matting method [38], guided image filtering provides visually equivalent result and is more handy to be integrated into our system. The resulting foreground masks are used when compositing the foreground and background to generate the final frames. One example of the foreground warping is shown in Fig. 6.

6.3.2 Foreground transition optimization

As we discussed in Section 6.1, we find a new camera path that smoothly transitions from the camera parameters for the starting frame F_s^1 in V_1 to the camera parameters for the ending frame F_e^2 in V_2. This means that the synthesized frame F_s' in the final result will be the same as F_s^1 in V_1, and the synthesized frame F_e' will be the same as F_e^2 from V_2. Given the aligned foreground frame pairs from V_1 and V_2, we would like to have a video sequence of the foreground object where the appearance gradually changes from V_1 to V_2. However, we already have two aligned foreground sequences from Section 6.3.1.

Fig. 6 Inputs and outputs of the foreground and background warping. The upper row and lower row are frames at the same time t from two videos.

Therefore, we can transition between these two foreground sequences. In our method, we perform this foreground transition by a 3D graph labelling problem. We construct a 3D graph where nodes represent all space-time samples in the video cube of the aligned foreground sequences. For each node, there are two possible color samples that could be drawn: one from the aligned video V_1 and the other from the aligned video V_2. We use edges to connect spatially and temporally neighbouring nodes. The label of a node represents which of the two videos it should come from. We make constraints that all the pixels at F'_{s} should come from V_1, and all pixels at F'_{e} should come from V_2. Then we use the graph-cut algorithm [39] to find the 3D seam for stitching the foreground objects by optimizing the following objective:

$$E(L) = \sum_{p_i \in \Omega} D_{p_i}(L_i) + \sum_{(p_i,p_j)\in\mathcal{N}} T_{(i,j)}(L_i, L_j) \quad (3)$$

Here Ω is the set of all samples in the video cube for the transition and L_i is the label of p_i. The smoothness term $T_{(i,j)}$ is defined as zero where $i = j$. Where $i \neq j$, $T_{(i,j)}$ is defined as $|E_i - E_j|$. Here E is the 3D energy map defined as the difference between the warped pixel color of V_1 and V_2 at the given sample location. The data term D_{p_i} is derived from the temporal distance from the current pixel to the end of the video of that sample. If the time of sample D_{p_i} is t_i then we define $D_{p_i}(L_i) = (t_i - t_{\mathrm{s}})/(t_{\mathrm{e}} - t_{\mathrm{s}})$ if $L_i = 1$, and $D_{p_i}(L_i) = 1 - (t_i - t_{\mathrm{s}})/(t_{\mathrm{e}} - t_{\mathrm{s}})$ if $L_i = 2$. As shown in Fig. 7, the graph cut gives us the best seam within every frame so that the appearance smoothly changes from the first video to the second. We blend these images together according to the labels to obtain the final foreground sequence.

6.4 Finalizing results

To give a high-quality final result, we also need to perform some additional video processing: we smooth the warps, apply cross-fading, and crop the created viewpoints appropriately. For the content-preserving warping, we smooth the warps by first estimating the positions of each grid vertex, and then using a median filter to temporally filter the trajectories of the vertex positions to guide the final warping. To ensure that the start and end frames of the transition are consistent with the original input videos, we perform a cross-fade on the background. In some cases, the warped frames will have irregular shapes, so cropping may be necessary to preserve a rectangular viewport. We do this by at each frame calculating the largest rectangular window which only contains valid pixels, and then we take the smallest window among all such windows as the target cropping window for the transition. To maintain continuity with the original videos, we normally begin cropping 2/3 s before the transition start time t_{s} and stop cropping 2/3 s after the transition end time t_{e}. The four corners of the cropping window are linearly interpolated over time, from the full window to the target window.

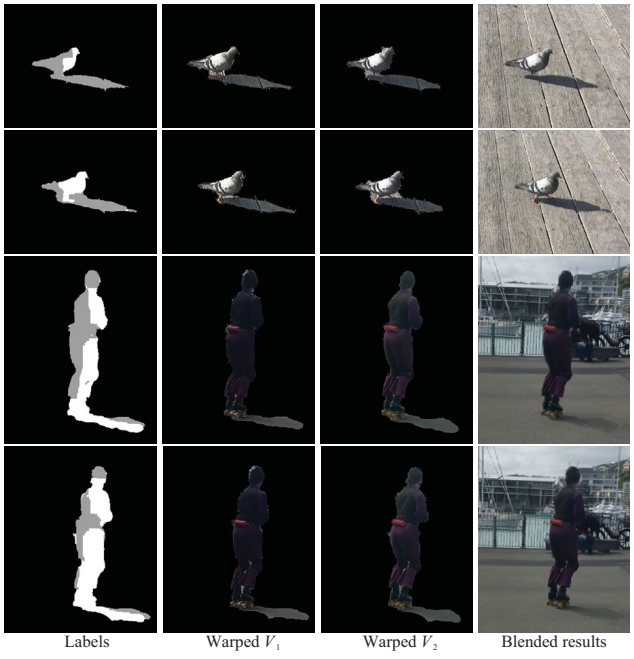


Fig. 7 A 3D spatiotemporal graph cut optimization is used to find the best seam at which we can transition between the foreground contents from the two different videos. Here the grey label indicates that the pixel should use the color from the warped video V_1, and white indicates V_2.

7 Discussions and results

A full result of our algorithm is shown in Fig. 8. Our method can determine good time to transit between different videos, so as to obtain a single smooth video of the target foreground object.

7.1 Comparisons with video stitching

One possible alternative to apply our transitions to two input videos is video stitching. Thus, we compare our pairwise smooth transition method in Section 6 with state-of-the-art video stitching algorithms. We choose two representative methods. Lin et al. [2] showed that using 3D information

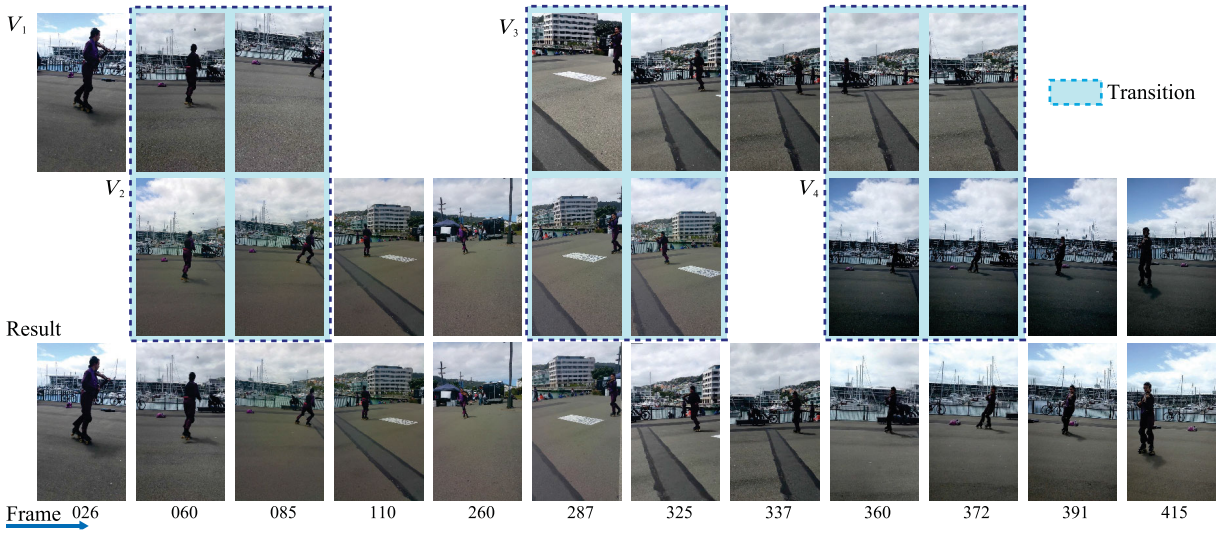


Fig. 8 Inputs and outputs of our full algorithm. Blue rectangles indicate when the transition occurs.

from SFM can significantly improve stitching results, especially in the cases where depth changes are large. However, that method does not consider the reliability of the reconstructed 3D information in dynamic foreground object regions. When SFM systems filter out such unreliable tracked feature points, the foreground region ends up being warped according to the surrounding background region. However, the foreground and background contents often lie in different depth layers, which cause the ghosting effects shown in Fig. 9. Alternatively, if the moving objects do not have large deformation, there may be foreground feature points that are erroneously reconstructed as background content by the SFM system, with incorrect 3D positions. Both cases will generate obvious artifacts for videos with dynamic foregrounds. Note that Lin et al. [2] only took content from one video to avoid ghosting. But in our task, we need to draw color information from both videos to enable a smooth transition from one to another. Therefore, in our comparison, we blend the two aligned videos for the method of Lin et al., in order to enable a smooth transition for that method.

Fig. 9 Comparison with state-of-the-art video stitching methods.

Nie et al. [3] proposed a video stitching method based on the robust background segmentation of Zhang et al. [13]. Nie et al. [3] used only the extracted background feature points to estimate a homography transformation as the camera model for each neighboring video pair. That method is fast and applicable in the situation where the appearance changes caused by depth differences are far less obvious than those caused by camera motion. However, a global transformation applied to the whole frame can create mis-alignment artifacts on the foreground regions, as shown in Fig. 9. Our method deals with the moving foreground and background separately. We also only use the camera model reconstructed from background feature points. These allow us to generate a better result for the transition sequences.

7.1.1 Quantitative evaluation

We conduct quantitative comparisons with the state-of-the-art video stitching algorithms in terms of the

alignment quality of the content at the transition frame pairs. The evaluation is conducted on 5 groups of input videos, where there are 817 transition frame pairs in their optimal transition route. As shown in Fig. 9, because of the saliency of the moving foreground objects, any alignment errors in the foreground are highly visually objectionable. Thus, we measure the visual quality in the final stitched result on the foreground and background regions separately. We use the per-frame root-mean-square error (RMSE) of the positions of matching feature points from a frame pair in the final stitched result as our measurement, as in Ref. [40]. In addition to the mean RMSE of all the transition frame pairs, we also calculate the standard deviations of the RMSE in each transition video sequence to determine the stability of the tested methods. The results are shown in Table 1. From the mean RMSE, we can see that our method has by far the best foreground alignment results, due to the separate processing for foreground and background regions. Due to a homography-based approximation, the method of Nie et al. [3] has the highest errors in both foreground and background: this typically occurs when the depth changes significantly. The method of Lin et al. [2] has a similar mean RMSE for the background content alignment, because the 3D camera model can be reconstructed with the largest inlier group in the matched feature point set, which is highly likely to be the background points. However, the standard deviation value for Lin et al. [2] is larger, because when the foreground region is relatively large, the SFM method will not generate robust camera models due to outlier moving foreground objects. From the quantitative evaluation, it can be seen that our method is the most suitable way to combine two videos for a smooth transition, especially when there is a moving foreground object.

Table 1 A quantitative evaluation of the transition quality. Here RMSE is the root-mean-square error of feature positions within a single frame pair. Within each transition sequence, based on all the frame pairs, we collect two statistics: mean and standard deviation ("Std.") of RMSE. Since there are multiple videos, we then report the mean of these statistics across all videos

RMSE of method	Foreground		Background		Full frame	
	Mean	Std.	Mean	Std.	Mean	Std.
Ref. [2]	4.310	2.926	0.491	0.211	1.09	0.639
Ref. [3]	5.693	3.862	3.214	2.475	3.605	2.681
Ours	0.778	0.329	0.457	0.132	0.507	0.163

7.2 Foreground warping methods

When warping the foreground objects, we use the new positions of extracted sparse feature points to guide the grid-based warping. A possible alternative is to use matched shape contours as the control points when warping the two images. To compare with this approach, we use the Shape Context method [41] to match the contour points of the foreground masks. Using these correspondences, we perform triangulation to obtain the 3D feature point positions with the given camera models. Next, we project the 3D points into the new camera as we have previously done. As in RepFinder [42], we use thin-plate spline interpolation to obtain the aligned foreground region. One warping result is shown in Fig. 10. We can see that the errors in the contour matching by shape-context will cause an unnatural deformation for the warping. Thus, we rely on the robust feature point matching results for our foreground warping.

7.3 User study assessing transitions

Our transition algorithm relies on finding good foreground correspondences and a seam that enables a pairwise smooth transition for the foreground. However, if the viewing angles in the frame pairs are significantly different, we may introduce noticeable visual artifacts during the transition. To test the tolerance for viewing angles, we perform the following experiment. We chose the toy train foreground object shown in Fig. 11, and capture it from a variety of different viewpoints. Two hand-held cameras were used to track and capture the motion of the foreground object. We record the initial physical distance between the foreground object and the two cameras, and the angle between the two cameras. We repeated a similar motion of the foreground

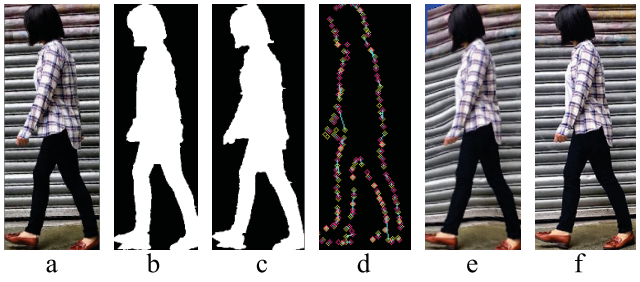


Fig. 10 Comparison with warping with matched shape contour. (a) Input frame from video V_1. (b) Mask for video V_2. (c) Input Mask for video V_1. (d) Shape-Context matching result. Blue lines show the correspondences. (e) Warped result using contour matching. (f) Our result.

Fig. 11 Large angles cause difficulties in finding foreground correspondences and finding good seams to blend aligned foreground regions.

object and captured it for different distance and angle settings. Our pairwise smooth transition method (Section 6) was applied to each video pair.

We invited 6 user study participants to evaluate the artifacts of the generated results. These participants were of age 18 to 30 years old, with 3 participants of each gender, and no participant had expertise in photography. The participants were asked whether they noticed unpleasant artifacts on the foreground objects in the generated videos. We record the minimum, maximum, and average angle and distance for each capture scenario in Table 2, as well as the user assessment. We can see that for cameras around 15 degrees apart, our method still gives an acceptable result, with only one person reporting artifacts. However, at around 20 degrees, there are obvious visual artifacts as shown in Fig. 11. These obvious misalignment artifacts occur particularly at corner and edge features, and are noticed by the users. Interestingly, if for a given transition, only a few frame pairs contain large angles between cameras, the result can still be acceptable. This is because our graph-cut optimization method will find seams on frame pairs that have smaller angular difference. Thus, based on our user study results, in Section 5, when we are finding an optimal transition route, we chose an average angle of 20 degrees between cameras as the threshold for excluding transition edges.

Table 2 The number of participants who report noticeably unpleasant artifacts is shown in the rightmost column. We only report one different distance setting (case 3) here

Case	Data	Min	Max	Ave	Num
1	Angle (°)	2.5	7.4	5.31	1 of 6
	Distance (m)	0.7	1.5	0.96	
2	Angle (°)	7.6	12.5	10.47	1 of 6
	Distance (m)	0.8	1.4	1.09	
3	Angle (°)	7.5	13.1	10.18	1 of 6
	Distance (m)	1.7	2.6	2.12	
4	Angle (°)	8.9	20.1	15.13	2 of 6
	Distance (m)	0.6	1.6	1.03	
5	Angle (°)	14.3	24.5	19.61	6 of 6
	Distance (m)	0.7	1.4	0.97	

7.4 Results and performance

Our method can handle multiple non-overlapping foreground objects for the smooth transition. In this case, we require the user to specify the different foreground objects of interest. Then we perform the foreground temporal transition separately on each different object.

For some input video collections there may be time at which no camera contains the foreground object. Our method in this case would ordinary only produce a result up to the first such time. However, optionally, we can first remove such frames, and then generate a final sequence containing the foreground region for a longer time. We provide examples for this case and the previous case in Fig. 12 and our supplementary video.

We tested our algorithm's performance on a Windows machine with a Core i7 CPU at 2.6 GHz and 8 GB of RAM. Considering that the videos captured by different devices may have different frame rates and resolutions, we encode all the input videos at a resolution of 1080×1920 and a frame rate of 24 fps before alignment. For a group of aligned videos with $\sim$100 overlapped frames, it takes 5–10 min to perform SFM using VisualSFM [33] or COLMAP [43]. For each frame, it takes $\sim$25 s to perform background/foreground separation and matting on a single core. For the temporal transition, it takes 6–7 s to perform background warping and 3–4 s for foreground warping. For a transition over 40 frames, the 3D graph-cut takes 1.1 s to optimize the transition seams.

7.5 Limitations and future work

Our method is based on SFM, so limitations in existing SFM methods may affect our pipeline. These SFM limitations can include: reconstruction issues for scenes that have too few background features, too few parallaxes in the camera motion, too large foreground objects which can cause background model fitting to fail, or highly specular materials that violate common Lambertian scene assumptions. Camera

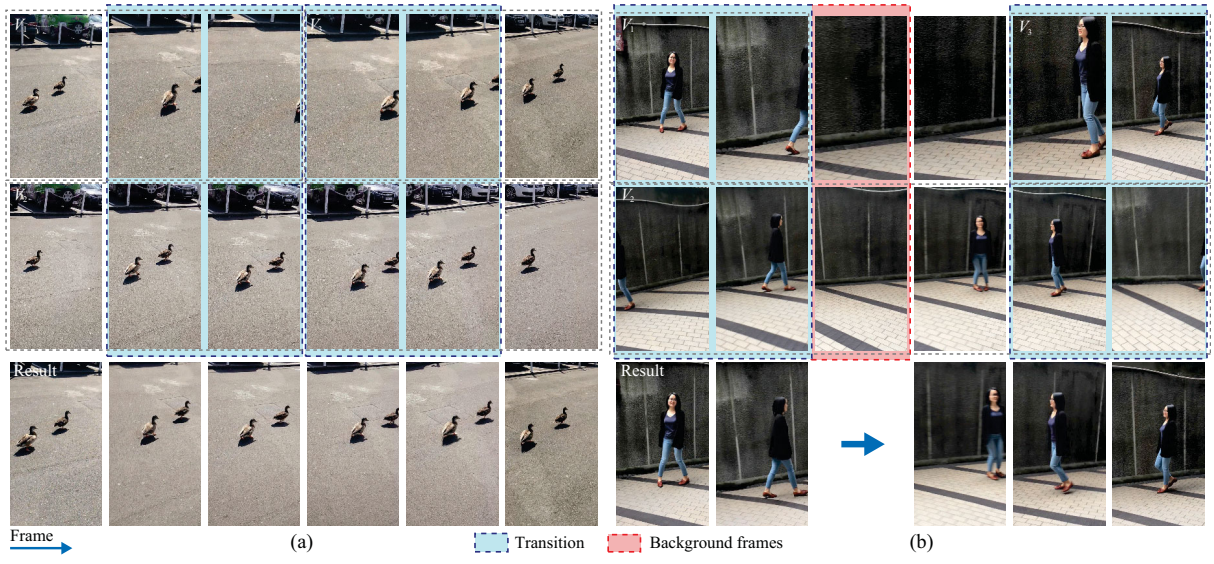


Fig. 12 Additional results. At left we show an example containing two foreground objects. At right we show an example where at one time, all input videos contain only background objects. Our system can find a transition to skip these frames. Please see Section 7.4 for more details.

models reconstructed from too few feature points will also cause warping artifacts for background on those regions without extracted features. More information about the applicable range of SFM can be referred to the work of Wu et al. [33]. For our inputs, we used hand-held cameras and intentionally allowed the camera to have some motion so that SFM will succeed. Future work might address these by either improving the SFM algorithms or reducing the reliance on SFM, such as using simpler models like homographies that use less 3D information.

If the foreground and background are similar in color, then the foreground–background separation algorithm [13] may be challenged. However, in this ambiguous case the results still tend to be good because it is also hard for a human to see the color differences. Although our method is not limited to certain motion types, the motion does potentially influence the coherent video generation if it causes motion blur that could make our method fail to find foreground correspondences between different views.

There are also a few limitations in our method that are due to difficulties of finding foreground correspondences. As previously mentioned, viewing angle differences greater than about 20 degrees cause challenges for our system. This is because the foreground SIFT [44] correspondences are less reliable and the seam finding problem is much harder. Also, if there are too few features in the foreground object, then results may have foreground object ghosting. This could happen if there is heavy motion blur or if the object is simply untextured, such as a balloon. If there are multiple overlapping foreground objects, our current pipeline is not able to produce satisfactory results, because the missing parts of moving foreground in the novel view are difficult to be inpainted properly. These limitations could be addressed in future work by exploring a variety of dense correspondence and new view synthesis algorithms for the foreground region.

8 Conclusions

In this paper, we introduced a novel method to obtain smooth transitions between dynamic foreground video content from multiple hand-held cameras. A continuous foreground object video can be created by finding optimal transitions between multiple camera views. We use a graph-based method to determine which transitions to use. Given optimal frames for pairwise video transition, our algorithm synthesizes a smooth transitional video between two temporally-aligned videos. Comparisons show that state-of-the-art video stitching techniques do not provide transition results as good as those from our algorithm.

Acknowledgements

This work was supported by a Research Establishment Grant of Victoria University of Wellington (Project

No. 8-1620-216786-3744) and a Victoria Research Excellence Award.

Electronic Supplementary Material Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-020-0187-3`.

References

[1] Guo, H.; Liu, S. C.; He, T.; Zhu, S. Y.; Zeng, B.; Gabbouj, M. Joint video stitching and stabilization from moving cameras. *IEEE Transactions on Image Processing* Vol. 25, No. 11, 5491–5503, 2016.

[2] Lin, K. M.; Liu, S. C.; Cheong, L. F.; Zeng, B. Seamless video stitching from hand-held camera inputs. *Computer Graphics Forum* Vol. 35, No. 2, 479–487, 2016.

[3] Nie, Y. W.; Su, T.; Zhang, Z. S.; Sun, H. Q.; Li, G. Q. Dynamic video stitching via shakiness removing. *IEEE Transactions on Image Processing* Vol. 27, No. 1, 164–178, 2018.

[4] Arev, I.; Park, H. S.; Sheikh, Y.; Hodgins, J.; Shamir, A. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 81, 2014.

[5] Carranza, J.; Theobalt, C.; Magnor, M. A.; Seidel, H.-P. Free-viewpoint video of human actors. *ACM Transactions on Graphics* Vol. 22, No. 3, 569–577, 2003.

[6] Collet, A.; Chuang, M.; Sweeney, P.; Gillett, D.; Evseev, D.; Calabrese, D.; Hoppe, H.; Kirk, A.; Sullivan, S. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics* Vol. 34, No. 4, Article No. 69, 2015.

[7] Szeliski, R.; Shum, H.-Y. Creating full view panoramic image mosaics and environment maps. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 251–258, 1997.

[8] El-Saban, M. A.; Refaat, M.; Kaheel, A.; Abdul-Hamid, A. Stitching videos streamed by mobile phones in real-time. In: Proceedings of the 17th ACM International Conference on Multimedia, 1009–1010, 2009.

[9] Lin, W.-Y.; Liu, S.; Matsushita, Y.; Ng, T.-T.; Cheong, F. L. Smoothly varying affine stitching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 345–352, 2011.

[10] Zaragoza, J.; Chin, T. J.; Tran, Q. H.; Brown, M. S.; Suter, D. As-projective-as-possible image stitching with moving DLT. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 36, No. 7, 1285–1298, 2014.

[11] Ma, T. Z.; Nie, Y. W.; Zhang, Q.; Zhang, Z. S.; Sun, H. Q.; Li, G. Q. Effective video stabilization via joint trajectory smoothing and frame warping. *IEEE Transactions on Visualization and Computer Graphics* doi: 10.1109/TVCG.2019.2923196, 2019.

[12] Liu, F.; Gleicher, M.; Jin, H. L.; Agarwala, A. Content-preserving warps for 3D video stabilization. In: Proceedings of the ACM SIGGRAPH 2009 papers, Article No. 44, 2009.

[13] Zhang, F.-L.; Wu, X; Zhang, H.-T.; Wang, J.; Hu, S.-M. Robust background identification for dynamic video editing. *ACM Transactions on Graphics* Vol. 35, No. 6, Article No. 197, 2016.

[14] Kwatra, V.; Schedl, A.; Essa, I.; Turk, G.; Bobick, A. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics* Vol. 22, No. 3, 277–286, 2003.

[15] Agarwala, A.; Zheng, K. C.; Pal, C.; Agrawala, M.; Cohen, M.; Curless, B.; Salesin, D.; Szeliski, R. Panoramic video textures. *ACM Transactions on Graphics* Vol. 24, No. 3, 821–827, 2005.

[16] Anderson, R.; Gallup, D.; Barron, J. T.; Kontkanen, J.; Snavely, N.; Hernández, C.; Agarwal, S.; Seitz, S. M. Jump: virtual reality video. *ACM Transactions on Graphics* Vol. 35, No. 6, Article No. 198, 2016.

[17] Silva, R. M. A.; Feijó, B.; Gomes, P. B.; Frensh, T.; Monteiro, D. Real time 360° video stitching and streaming. In: Proceedings of the ACM SIGGRAPH 2016 Posters, Article No. 70, 2016.

[18] Guo, H.; Liu, S. C.; Zhu, S. Y.; Shen, H. T.; Zeng, B. View-consistent MeshFlow for stereoscopic video stabilization. *IEEE Transactions on Computational Imaging* Vol. 4, No. 4, 573–584, 2018.

[19] Wei, X.; Chai, J. Videomocap: Modeling physically realistic human motion from monocular video sequences. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 42, 2010.

[20] Ballan, L.; Brostow, G. J.; Puwein, J.; Pollefeys, M. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 87, 2010.

[21] Tompkin, J.; Kim, K. I.; Kautz, J.; Theobalt, C Videoscapes: exploring sparse, unstructured video collections. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 68, 2012.

[22] Wang, M.; Lyu, X. Q.; Li, Y. J.; Zhang, F. L. VR content creation and exploration with deep learning: A survey. *Computational Visual Media* Vol. 6, No. 1, 3–28, 2020.

[23] Zhu, Z.; Lu, J. M.; Wang, M. X.; Zhang, S. H.; Martin, R. R.; Liu, H. T.; Hu, S.-M. A comparative study of algorithms for realtime panoramic video blending. *IEEE Transactions on Image Processing* Vol. 27, No. 6, 2952–2965, 2018.

[24] Lee, W.; Chen, H.; Chen, M.; Shen, I.; Chen, B. Y. High-resolution 360 video foveated stitching for real-time VR. *Computer Graphics Forum* Vol. 36, No. 7, 115–123, 2017.

[25] Liu, Q. X.; Su, X. Y.; Zhang, L.; Huang, H. Panoramic video stitching of dual cameras based on spatio-temporal seam optimization. *Multimedia Tools and Applications* Vol. 79, 3107–3124, 2020.

[26] Perazzi, F.; Sorkine-Hornung, A.; Zimmer, H.; Kaufmann, P.; Wang, O.; Watson, S.; Gross. M. Panoramic video from unstructured camera arrays. *Computer Graphics Forum* Vol. 34, No. 2, 57–68, 2015.

[27] Wang, O.; Schroers, C.; Zimmer, H.; Gross, M.; Sorkine-Hornung, A. VideoSnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 77, 2014.

[28] Cui, Z. P.; Wang, O.; Tan, P.; Wang, J. Time slice video synthesis by robust video alignment. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 131, 2017.

[29] Barnes, C.; Goldman, D. B.; Shechtman, E.; Finkelstein, A. Video tapestries with continuous temporal zoom. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 89, 2010.

[30] Zhang, Z. S.; Nie, Y. W.; Sun, H. Q.; Lai, Q. X.; Li, G. Q. Multi-video object synopsis integrating optimal view switching. In: Proceedings of the SIGGRAPH Asia 2017 Technical Briefs, Article No. 17, 2017.

[31] Wang, M.; Shamir, A.; Yang, G. Y.; Lin, J. K.; Yang, G. W.; Lu, S. P.; Hu, S.-M. BiggerSelfie: Selfie video expansion with hand-held camera. *IEEE Transactions on Image Processing* Vol. 27, No. 12, 5854–5865, 2018.

[32] Wu, C. VisualSFM: A visual structure from motion system. 2011. Available at http://ccwu.me/vsfm.

[33] Wu, C.; Agarwal, S.; Curless, B.; Seitz, S. M. Multicore bundle adjustment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3057–3064, 2011.

[34] Lee, S. M.; Xin, J. H.; Westland, S. Evaluation of image similarity by histogram intersection. *Color Research & Application* Vol. 30, No. 4, 265–274, 2005.

[35] Newson, A.; Almansa, A.; Fradet, M.; Gousseau, Y.; Pérez, P. Video inpainting of complex scenes. *SIAM Journal on Imaging Sciences* Vol. 7, No. 4, 1993–2019, 2014.

[36] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* Vol. 60, No. 2, 91–110, 2004.

[37] He, K. M.; Sun, J.; Tang, X. O. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 35, No. 6, 1397–1409, 2013.

[38] Wu, X.; Fang, X. N.; Chen, T.; Zhang, F. L. JMNet: A joint matting network for automatic human matting. *Computational Visual Media* Vol. 6, No. 2, 215–224, 2020.

[39] Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 23, No. 11, 1222–1239, 2001.

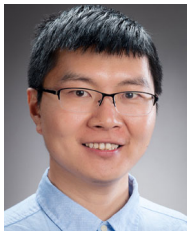
[40] Zhang, Y.; Lai, Y.-K.; Zhang, F.-L. Content-preserving image stitching with regular boundary constraints. *arXiv preprint* arXiv:1810.11220, 2018.

[41] Belongie, S.; Malik, J.; Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 24, No. 4, 509–522, 2002.

[42] Cheng, M. M.; Zhang, F. L.; Mitra, N. J.; Huang, X. L.; Hu, S. M. RepFinder: Finding approximately repeated scene elements for image editing. In: Proceedings of the ACM SIGGRAPH 2010 Papers, Article No. 83, 2010.

[43] Schönberger, J. L.; Frahm, J.-M. Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4104–4113, 2016.

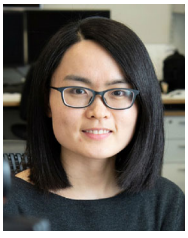
[44] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* Vol. 60, No. 2, 91–110, 2004.



Fang-Lue Zhang is currently a lecturer with Victoria University of Wellington, New Zealand. He received his bachelor degree from Zhejiang University, Hangzhou, China, in 2009, and his doctoral degree from Tsinghua University, Beijing, China, in 2015. His research interests include image and video editing, computer vision, and computer graphics. He is a member of IEEE and ACM. He received Victoria Early-Career Research Excellence Award in 2019.

Connelly Barnes is a senior researcher at Adobe Research. Previously, he was an assistant professor at the University of Virginia. He received his Ph.D. degree from Princeton University in 2011. He develops techniques for efficiently manipulating visual data in computer graphics by using semantic information from computer vision, with applications in computational photography, image editing, art, and hiding visual information. Many computer graphics algorithms are more useful if they are interactive; therefore, he also focuses on efficiency and optimization, including some compiler technologies.

Hao-Tian Zhang is currently a Ph.D. student at Stanford University. He received his B.S. degree from Tsinghua University in 2017. His research interests include image and video editing, and physically-based simulation.



Junhong Zhao is a postdoctoral research fellow of the Computational Media Innovation Centre (CMIC) at Victoria University of Wellington. She completed her doctoral degree in 2015 at the Institute of Electronics, Chinese Academy of Sciences. She worked in the Human–Computer Speech Interaction Lab of Tsinghua University, on computer-assisted language learning using speech recognition and computer graphics techniques (2011–2015). She then moved to CAS Institution of Information Engineering, where her research focused on machine learning and its applications on image understanding and audio signal processing (2015–2017).

Gabriel Salas is a research assistant and undergraduate student at the School of Engineering and Computer Science at Victoria University of Wellington. His research focuses on computer graphics and image processing.