Time Slice Video Synthesis by Robust Video Alignment

ZHAOPENG CUI, Simon Fraser University
OLIVER WANG, Adobe Research
PING TAN, Simon Fraser University
JUE WANG*, Adobe Research

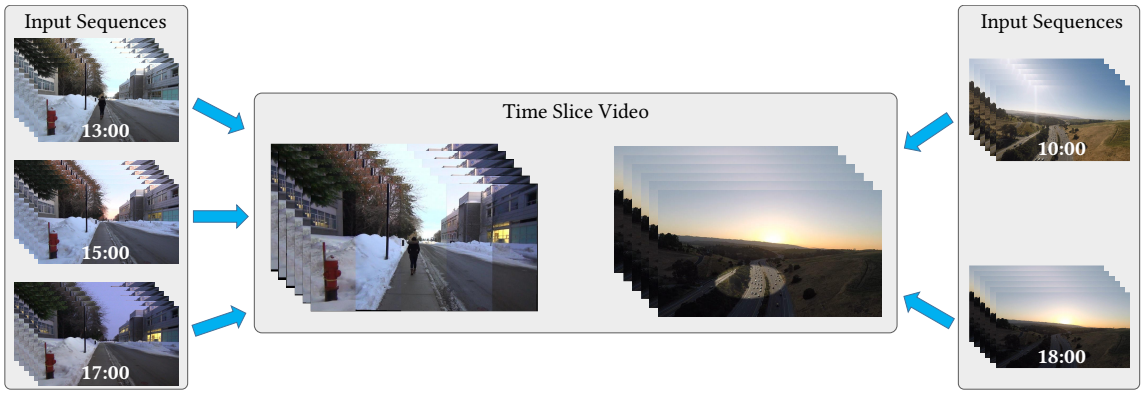


Fig. 1. Time slice video makes use of a robust spatio-temporal alignment to enable the blending of multiple videos recorded with different appearances to be blended together in a number of configurations. Here we show traditional time slice vertical bars (left), as well as a world-space shape that drives the compositing, such as the 3D spotlight (right).

Time slice photography is a popular effect that visualizes the passing of time by aligning and stitching multiple images capturing the same scene at different times together into a single image. Extending this effect to video is a difficult problem, and one where existing solutions have only had limited success. In this paper, we propose an easy-to-use and robust system for creating time slice videos from a wide variety of consumer videos. The main technical challenge we address is how to align videos taken at different times with substantially different appearances, in the presence of moving objects and moving cameras with slightly different trajectories. To achieve a temporally stable alignment, we perform a mixed 2D-3D alignment, where a rough 3D reconstruction is used to generate sparse constraints that are integrated into a pixelwise 2D registration. We apply our method to a number of challenging scenarios, and show that we can achieve a higher quality registration than prior work. We propose a 3D user interface that allows the user to easily specify how multiple videos should be composited in space and time. Finally, we show that our alignment method can be applied in more general video editing and compositing tasks, such as object removal.

CCS Concepts: • **Computing methodologies** → **Image manipulation**; *Computational photography*;

Additional Key Words and Phrases: time slice, video alignment, 3D reconstruction, SIFT flow, video compositing

ACM Reference format:
Zhaopeng Cui, Oliver Wang, Ping Tan, and Jue Wang. 2017. Time Slice Video Synthesis by Robust Video Alignment. *ACM Trans. Graph.* 36, 4, Article 131 (July 2017), 10 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073612

*Jue Wang is now with Megvii Inc..

1 INTRODUCTION

Time slice photography refers to the artistic effect of combining multiple images of a scene captured at different times together into a single composite, where each slice or stripe of the final image shows part of the scene at a specific time. Putting all slices together, the image conveys a passage of time and how it changes the appearance of a place, as shown in the example in Figure 2. Our goal in this paper is to extend this effect from still images to video, to create time slice videos from multiple video sequences captured at different times. In contrast to traditional time slice photography where all images are captured by a single static camera, our system allows videos to be captured by handheld cameras in order to cover a large scale scene, although we do require the input videos roughly follow the same camera motion trajectory to ensure scene continuity and with minimal parallax. We additionally require that the majority of the scene content is static, so that correspondences can be found across videos. The final video is a spatio-temporal composite of the input sequences, where different parts of the scene are from videos captured at different times, while the whole scene structure is well preserved as if it was shot from a single camera.

It is not surprising that such an extension from still images to video is nontrivial and brings many new technical challenges. As we will discuss in more detail later, previous video alignment approaches have attempted to solve this problem, but only with limited success under restricted settings. The fundamental problem is how to compute a spatially accurate and *temporally stable* alignment of multiple video sequences, given that they can have different camera trajectories and motion. While this looks like a standard video

Fig. 2. Time slice photograph. Photo credits: Miguel Mendez.

alignment problem, it becomes especially challenging in our application since different videos are captured at different times, leading to significant appearance difference. Furthermore, our application requires high quality alignment throughout the video, as even small pixel misalignments become easily noticeable and break the scene integrity.

To achieve this, we present a mixed 2D-3D multi-video alignment method that is robust to appearance difference of video sequences, as well as to content that may appear only in one video. Once aligned, sequences recorded at different times can be blended in various ways to create time slice videos. We first compute a joint sparse 3D reconstruction using structure-from-motion (SfM), giving us a single global reconstruction across all videos. This can be used to derive reliable drift-free, albeit sparse, static scene points in correspondence, and also allows us to achieve frame-level (temporal) alignment across videos based on camera position and orientation. We then compute a dense 2D matching using a modified version of SIFT flow [Liu et al. 2011] that incorporates these sparse 3D points as constraints and an extra intra/inter-video loop consistency check to remove outliers. This method is applied on a keyframe basis, and alignments are propagated to in-between frames using optical flow. Finally, the remaining reliable SIFT flow points are used to compute a seam-aware mesh warping. These 2D alignment steps allows us to create a dense alignment result in cases where 3D content cannot be reconstructed perfectly, or where mis-calibration may make it hard to reconstruct the scene in 3D.

Another key advantage of operating in a mixed 2D-3D space, is that we are able to introduce an intuitive way to specify time slices that exist not just in *image-space*, but in *world-space*, corresponding to real physical 3D regions.

In summary, we present the following contributions. We describe a solution to create time slice videos. To achieve this, we describe a number of modifications to existing registration pipelines, which we found to be essential to get the alignment quality necessary. Additionally, we present a 3D selection interface, which allows us to create *world-space* aligned time slices in addition to image-space ones.

2 RELATED WORK

Time slice. Time slice photography is usually created using a fixed camera position and image-space blending shapes added in post production. One option to reduce the capture requirements, is to hallucinate the appearance of a photograph at different times. This has been accomplished using a database of webcam or timelapse videos to learn a locally affine color mapping [Shih et al. 2013], or or a set of attributes that allow the user to make the scene look e.g., "more like winter" [Laffont et al. 2014]. These approaches generate realistic looking results for images, but do not extend trivially to video sequences with moving cameras. Other work on video has proposed combining multiple parts of different videos [Pritch et al. 2008] into a video synopsis, however this method works only on videos recorded with fixed camera, and performs all operations in image space.

Image alignment. Non-static cameras require aligning frames across videos, which is a classic problem of computer vision and computer graphics. Traditional approaches rely on matching feature points between images [Lowe 1999] and applying a global or smoothly interpolated warping to register them, or by computing per-pixel correspondences, e.g., using optical flow [Horn and Schunck 1981]. In general, global alignment methods are robust to outlier estimates, but cannot handle parallax effects, while optical flow based methods are able to handle arbitrary scenes, but are more prone to warping artifacts. Additionally, optical flow relies on an assumption of brightness constancy, which restricts it to matching image pairs with similar appearance. SIFT Flow [Liu et al. 2011], replaces optical flow pixel matching with dense SIFT descriptors used in feature matching, which adds robustness to appearance difference. Each of these approaches has advantages and disadvantages, but in particular they are all designed to match a single pair of images, and do not trivially extend to pairs of videos, due to temporal coherency issues.

2D video alignment. Rüegg et al. [2013] introduce a block-based local search between views, using intra-view (temporal) homographies to initialize a inter-view (spatial) homography used for alignment. This approach is restricted to scenes where geometry can be well approximated by a single plane. Sand and Teller [2004] propose a method for registering video clips that consists of robust feature matching and dense interpolation. They also propose an image preprocessing step to reduce the effect of lighting for sequences recorded at different intensities. This method however, relies on a good initial guess for frame-level registration due to the local regression method used for this step. It can handle certain amount of lighting change, but the ability is inherently limited by the image features it uses (Haris corners). Thanks to the 3D reconstruction and SIFT flow methods used in our system, we can achieve better frame-level alignment and handle more dramatic appearance change, in a more robust way. Beyond registration, we demonstrate how 3D reconstruction can help users quickly specify semantically-meaningful spatial-temporal seams for compositing, which has not been explored in previous methods.

Zhong et al. [2014] present a different solution to compositing two videos. It takes as input a pre-segmented foreground object, and

computes an optimal spacetime warping, focusing on the contact points of the object and the background video to prevent slippage.

Aligning views from multiple cameras is also a fundamental step in stitching together wide angle (incl. 360°) video. These approaches have used methods similar to single-image registration techniques such as feature matching and mesh warping [Guo et al. 2016; Lee et al. 2016], optical flow [Perazzi et al. 2015], or joint 3D reconstruction [Lin et al. 2016]. Temporal consistency is enforced by restricting the mesh to undergo temporally smooth warping [Lee et al. 2016; Lin et al. 2016], or by directly regularizing flow estimates [Anderson et al. 2016]. These methods however cannot be directly adapted for our application. In particular, creating panoramic video requires aligning frames that are captured at the same time, which means the only major source of differences between images is parallax, which can be kept to minimum by customized hardware configurations. In our case, we align videos recorded by handheld cameras at different times, days, or even months, thus our videos have substantially larger differences in both appearance and content, which cannot be handled well by these previous methods.

3D video alignment. Several methods have leveraged 3D reconstruction to help with computing image space alignment. For example, Liu et al [2009] reconstruct sparse 3D points and use them for computing an image warp to render the video in a stable path. A similar idea was used by Kopf et al [2014] to create smooth, watchable high speed videos. Zhang et al. [2009] propose an approach where an accurate depth map is computed, which allows for depth-specific video effects such as refocusing. Similarly, Klose et al. [2015] compute per-frame depth maps and project all pixels in a video into a 3D space. These pixels are then gathered to render a modified output video. All of these approaches use a 3D reconstruction (sparse or dense) from *single* video. In our work, we use 3D reconstructions across multiple videos, which provide us with additional information to help with alignment, as registering videos in 3D space can be easier than trying to compute sparse 2D matches when the appearance is significantly different across views.

More recently, Lin et al. [2016] utilize 3D reconstruction for multi-video stitching. Their approach uses CoSLAM [Zou and Tan 2013] to compute camera poses, and then computes a dense stereo map, which is fed into a warp procedure for alignment. While this works for aligning videos captured *simultaneously* by multiple devices, both CoSLAM and dense stereo matching are not applicable for videos recorded at different times, or with substantially different appearances. Our approach is quite different from this method, as we apply global SfM using all videos and use the sparse (instead of dense) 3D points as constraints in our alignment method. Unlike [Lin et al. 2016], our approach can also handle videos of different frame rates and speeds.

Sequence Alignment. Unlike prior work that assumes that the clips are already in temporal alignment [Rüegg et al. 2013], we derive a frame-to-frame temporal alignment to compensate for differences in speed of each video. Previous methods have used histograms of image-based feature matching [Wang et al. 2014] to align two video clips, or looked for nearby frames that result in the best image matching [Sand and Teller 2004]. Recently, Freeman et al. [2016] propose a deep learning approach, that trains CNNs to compute pairwise frame similarity for driving videos recorded under different weather conditions. The main focus of this work is to find temporal correspondences by searching for a shortest path in a frame-to-frame cost matrix, after which the method aligns frames using optical flow. In our case we use the 3D reconstruction to compute temporal frame correspondences, which gives added robustness.

3 METHOD

Our pipeline is visualized in Figure 3. Although our input videos have roughly the same camera trajectories, they are shot by handheld cameras, thus the pace and camera motion are slightly different. This requires us to first apply *frame-level registration* to find most similar frames across multiple videos, before applying *pixel-level registration* among these frames.

We first apply 3D reconstruction jointly on all input video sequences, and compute a frame-level registration between two different videos based on camera configuration. Secondly, we compute a pixel-level registration between two keyframes that are paired in the previous step, using sparse 3D scene points that exist in both videos as constraints. Finally, based on user-specified 3D seams, we perform *video synthesis* to generate the combined result.

We describe each of these steps in the following subsections. For simplicity we first assume two input sequences only, and then extend our method to to handle more than two videos.

3.1 Frame-level 3D registration

For 3D reconstruction, we first extract keyframes from each input video with uniform subsampling (one every ten frames). We feed all keyframes from different videos into a global SfM system [Cui and Tan 2015] and obtain the sparse 3D reconstruction as shown in Figure 3. We then interpolate the camera poses from extracted frames to in-between frames using linear interpolation for camera positions, and quaternions to interpolate camera rotation.

With the estimated camera poses of all frames, we construct a cost matrix of frame correspondences, and use dynamic programming to find the optimal frame-to-frame alignment. This is similar to the approach in [Wang et al. 2014], but instead of using image-based features, we can directly compare the camera poses. This is especially good for pixel-wise alignment as we can pair frames with closest camera poses to minimize parallax. Specifically, denoting two videos as A and B, and two series of camera poses as $Q^A = (\mathbf{q}_1^A, \mathbf{q}_2^A, ..., \mathbf{q}_n^A)$ and $Q^B = (\mathbf{q}_1^B, \mathbf{q}_2^B, ..., \mathbf{q}_m^B)$, $\mathbf{q}_i$ encodes the camera rotation angle θ_i and translation $\mathbf{c}_i$ of the ith camera in each sequence. We then construct an n-by-m matrix C where its (i, j) element corresponds to the cost $d(\mathbf{q}_i^A, \mathbf{q}_j^B) = \|\mathbf{c}_i^A - \mathbf{c}_j^B\| + \beta\|\theta_i^A - \theta_j^B\|$ of aligning $\mathbf{q}_i^A$ with $\mathbf{q}_j^B$. To find the best match between these two sequences, we compute the optimal warping path W^* by solving:

$$\begin{aligned} DTW(Q^A, Q^B) &= C_{W^*}(Q^A, Q^B) \\ &= min\ C_W(Q^A, Q^B), \end{aligned} \tag{1}$$

where W is a (n, m)-warping path [Müller 2007],

$$C_W(Q^A, Q^B) = \sum_{k=1}^{K} w_k \tag{2}$$

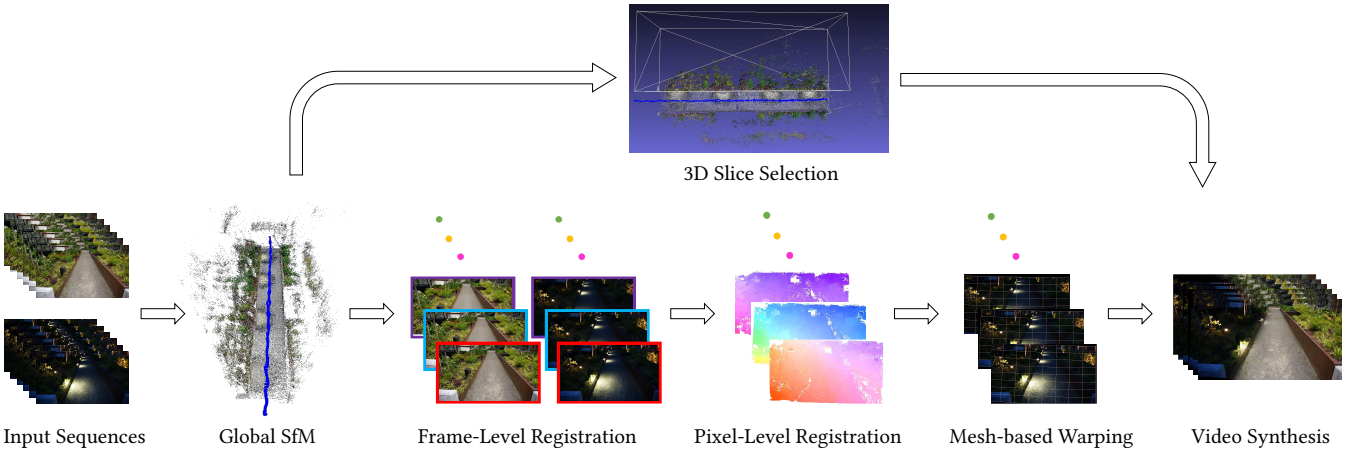


Fig. 3. Pipeline of the proposed method highlighting the main steps: joint 3D reconstruction, temporal registration, semi-dense 2D registration, mesh-based warping, and finally video synthesis. By working in 3D, we can optionally select regions to blend directly in world-space.

is the total cost for W, and w_k corresponds to the matrix element (i_k, j_k) of C. This ensures the final matching results are monotonic and achieve a good balance between matching accuracy of individual frames and temporal matching coherence. In our system, β is set based on the scale of the 3D reconstruction. We first compute the shortest distance to the frames of B for each keyframe of A and compute their median value λ, and then set β as 0.1λ. As the output of this stage, we have a series of known camera poses for each camera, and a set of frames in correspondence.

3.2 Pixel-level 2D registration

Given a pair of matched frames from the previous step which we denote A_i, B_j, the next step is to compute a dense pixel-wise registration for each image pair. For efficiency, we use a hierarchical strategy, first computing a dense pixel-wise registration between the keyframe pairs to get reliable feature correspondences. We then propagate the correspondences to the remaining frame pairs using optical flow. The keyframe pairs are chosen by using all keyframes in the first sequence A, which acts as a reference, and the corresponding matched frames in the second sequence B.

Guided SIFT flow matching for keyframe pairs. Given that our input videos can be taken at different times, there may be large appearance differences between matched frames. SIFT flow [Liu et al. 2011] was designed to handle illumination changes well, but it may fail when there is large parallax between the two frames. In order to deal with these problems, we propose a *guided SIFT flow* method that leverages the sparse 3D points computed by the global SfM and as constraints, and includes a subsequent intra/inter-frame loop consistency check.

Specifically, we first project the sparse 3D points onto the ith frame as:

$$[x_i^j, y_i^j, 1]^\top = \gamma \mathbf{K}_i[\mathbf{R}_i | -\mathbf{R}_i\mathbf{c}_i]\mathbf{P}^j, \tag{3}$$

where $\mathbf{K}_i$ is the camera intrinsic matrix , $\mathbf{R}_i$ and $\mathbf{c}_i$ are the camera rotation and position, γ is the scale factor, and $\mathbf{P}^j$ is the homogeneous coordinate of a 3D point. By projecting a 3D point to the two frames, we obtain an image space (2D) correspondence between they frames as shown in Figure 4. Using these *reliable* 2D correspondences, we compute a global homography transformation between them, and use it to pre-warp B to B' to roughly align it to A through a global homography transformation H. The global warping is used to compensate for large camera pose and orientation differences between the two images.

After global warping, we compute dense SIFT images S_A and $S_{B'}$ for A and B'. We then compute pixelwise matching by minimizing the following matching energy function:

$$E = E_d + E_s + E_g. \tag{4}$$

As in [Liu et al. 2011], the data term E_d is defined as:

$$E_d = \sum_{\mathbf{p}} min\left(\|S_A(\mathbf{p}) - S_{B'}(\mathbf{p} + \mathbf{w}(\mathbf{p}))\|_1, t\right), \tag{5}$$

where $\mathbf{p} = (x, y)^\top$ is pixel location on the image, $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ is the flow vector that matches $S_A(\mathbf{p})$ with $S_{B'}(\mathbf{p} + \mathbf{w}(\mathbf{p}))$, t is a threshold which is set according to the histogram of the SIFT feature matching [Liu et al. 2011]. This term encourages the matched points to have similar SIFT descriptors.

Also as in [Liu et al. 2011], the smoothness term E_s is defined as:

$$E_s = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{N}} min\left(w_s|u(\mathbf{p}) - u(\mathbf{q})|, d\right) + min\left(w_s|v(\mathbf{p}) - v(\mathbf{q})|, d\right), \tag{6}$$

where $\mathcal{N}$ is the set that contains all the spatial neighborhoods (we use a four-neighbor system). This term encourages the flow vectors of adjacent pixels to be similar.

Finally, our novel guidance term E_g is defined as:

$$E_g = \sum_{\mathbf{x}\in\mathcal{M}} F\left(\mathbf{p}_\mathbf{x} + \mathbf{w}(\mathbf{p}_\mathbf{x}) - H(\mathbf{q}_\mathbf{x})\right) = \sum_{\mathbf{x}\in\mathcal{M}} F\left(\mathbf{z}\right), \tag{7}$$

where $\mathbf{z} = \mathbf{p}_\mathbf{x} + \mathbf{w}(\mathbf{p}_\mathbf{x}) - H(\mathbf{q}_\mathbf{x}) = (x_\mathbf{z}, y_\mathbf{z})^\top$ is the vector difference,

$$F(\mathbf{z}) = f\left(|x_\mathbf{z}|\right) + f\left(|y_\mathbf{z}|\right), \tag{8}$$

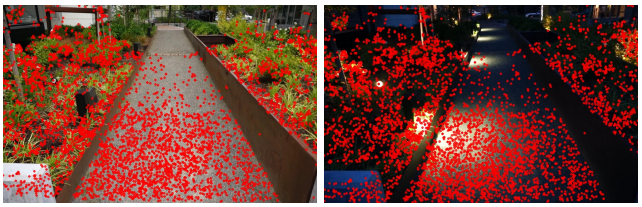


Fig. 4. This figure shows the projections of 3D points (red dots). We can see that they are not well distributed where the upper part of images has few points.

$$f(x) = \begin{cases} 0 & x \le d_2 \\ \psi & \text{otherwise} \end{cases}, \tag{9}$$

$\mathcal{M}$ is a set of 3D points that are visible in either A or B, $\mathbf{p_x}$ and $\mathbf{q_x}$ are the projected 3D scene points on A and B, respectively. $H(\mathbf{q_x})$ is the position of $\mathbf{q_x}$ in the warped image B'. This term enforces that the two projections of the same 3D scene point should match. In our system, we set w_s, d, d_2 and ψ to be 2, 40, 8 and 25 respectively. To enhance the robustness, we compute the bidirectional flows and remove unreliable matches through bidirectional checking. Note that once the flow between A and B' is computed, we can easily apply an inverse global homography warping to the flow filed to produce the flow between original video frames of A and B. We solve Equation 4 using belief propagation, similar to [Liu et al. 2011].

Loop consistency check. Dense matching often employs a forward-backwards consistency check, where a point is warped from A to B and then from B to A. If the point ends in the same place, it is likely that the motion estimation is accurate. However, this approach ignores any temporal relationships between keyframes. So we further use an inter/intra video loop consistency check. To do this, we compute optical flow between keyframes within the same sequence (where appearance similarity assumption holds) using a recent fast optical flow method [Kroeger et al. 2016]. Up until now, we have computed dense flows between sparse key frames. As illustrated in Figure 5, suppose we have two adjacent keyframes A_1 and A_2 in one sequence, and their matching counterparts B_1 and B_2 in the other. We have computed SIFT flow $SF(A_1, B_1)$ and $SF(A_2, B_2)$ between two pairs of keyframes, and also computed optical flow within each sequence as $OF(A_1, A_2)$ and $OF(B_1, B_2)$. For every pixel p in A_1, there are two paths that lead it to its destination in B_2: $OF(A_1, A_2) + SF(A_2, B_2)$, or $SF(A_1, B_1) + OF(B_1, B_2)$, resulting in two candidate matching points. If all correspondences are computed accurately, these two points should collide in the same location in B_2. In other words, if these two points have a large spatial distance, then it means the matching in this loop are not reliable. In our system we set a distance threshold of 2 pixels as the loop consistency criterion. If it is violated, we then conservatively label all pixels in the loop as unreliable.

Correspondence propagation. In theory one can apply above guided SIFT flow between every pair of matched frames. However this requires a significant amount of computation. In practice we have found that we can achieve very similar results by building dense correspondence between sparse keyframe pairs first, then propagate the correspondence to in-between frames using optical flow, which typically works well within the same video sequence, and is much faster to compute compared with SIFT flow.

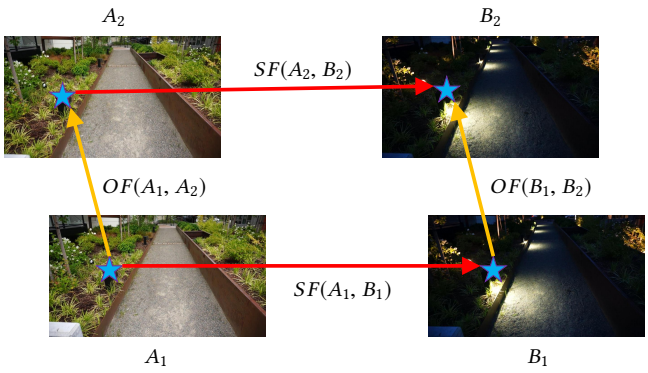


Fig. 5. Illustration of loop consistency check.

3.3 Video synthesis

Slice selection. Once the two sequences are aligned, we now have to specify which part of each sequence we want to use in the final composite. There are a number of ways to do this, we can use simple image-space polygons to specify blending regions (see Figure 9) as is used in traditional time-slice images. Alternately, we propose a 3D interface for scene selection, which creates a new type of time slice videos, exhibiting interesting *world-space* seams.

In general, object selection in video is challenging, frame-by-frame labeling is often impractical, and lacks temporal coherence. Although more intelligent video object segmentation systems can be employed [Li et al. 2016; Rother et al. 2004], the amount of required user interaction is still very large for general videos. Prior seam-based work [Rüegg et al. 2013], finds an optimal mask where there is minimal color difference using graph cuts, however this method won't work in our case as the appearance is different between clips. Furthermore, the user may sometimes choose to create a seam inside a textureless region, which is hard for any segmentation or tracking methods to follow.

As we are working in a mixed 2D-3D environment, we propose a 3D object selection interface as an efficient way to identify the spatio-temporal stitching seam between two videos. Working in a 3D interface has a number of advantages in our application. For instance, object selection in 3D is often easier than in 2D. Thanks to the depth information, a single bounding box is often sufficient to select an object, which may have complex image-space boundaries that are hard to segment. Furthermore, as the 3D reconstruction is done for the entire sequence, selecting an object in 3D simultaneously provides temporally consistent constraints on all frames in which that object is visible. This eliminates the need to propagate selection regions using conventional visual tracking, which is prone to drifting or occlusions. Figure 8 shows some examples of the user-specified 3D scenes. In Figure 8(a), we evenly divide the scene using 3D cubes, so the scene changes between day and light as the camera moves along the path. In Figure 8(b), we use a single 3D cube to

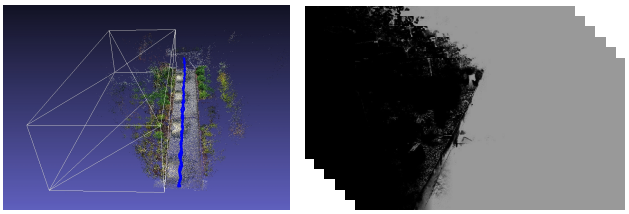
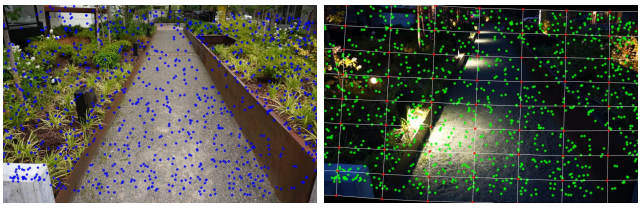


Fig. 6. This figure shows the computed 2D masks (right) guided by the 3D mask (left).



Reference Source with mesh warping

Fig. 7. This figure shows the selected reliable points after our feature refinement (blue and green dots), and the result of the mesh-warping in the source image.

select the entire left garden. In Figure 8(c) we use a spot-light effect by selecting a circular cylinder in 3D.

After the selection of the slice in 3D space, we need further obtain the mask in 2D space. We first project both the 3D mask and the 3D points onto key frame images. The projections of 3D mask provides soft mask constraints for 2D segmentation. What's more, we can easily determine whether a 3D point is in or outside the 3D cube, so we can also get the some hard constraints from the 3D points' projections. With these prior information, we can take 2D segmentation on the key frame images. Then we can propagate the 2D mask on key frames to non-key frame by taking the local video segmentation in a small window with optical flows or using advanced video segmentation algorithms like [Märki et al. 2016]. The 2D mask examples are also shown in Figure 6.

Seam-aware mesh warping. We now have a reliable piecewise-dense correspondence field, however we cannot directly use this for warping. This is because after the loop consistency check, large parts of the correspondence field may be removed, especially when there are different objects present in the videos. In addition, smoother warping fields are much less likely to generate warping artifacts, at the expense of being able to handle parallax. We found that we could obtain the best results by using our reliable 2D correspondence field to drive a mesh-warping.

For mesh warping, we select only the 2D correspondences with the highest confidence by eliminating unreliable correspondences in three steps. First, we remove the weaker points in low contrast regions (e.g. some point in the blue sky), which are often incorrect even if they pass the loop consistency check. We compute the Difference-of-Gaussian (DoG) images, and compute the median DoG values of all candidates. We then examine each candidate point, and remove those whose DoG values are smaller than the median. For computing DoG we set the kernel size as 3×3, and δ_1 and δ_2 are set to be 0.5 and 5 respectively. Second, we make sure that the selected matched points are distributed in each grid cell (8×8) as evenly as possible. If a cell contains too many points, we only sample a portion of them. Finally, we require that the final selected points to be temporally as consistent as possible. Specifically, we make sure that at least 50% of selected points on a keyframe has correspondences with the selected points on the previous keyframe. If more points have no temporal correspondence, we subsample from them.

We have observed that when creating time slice video, human perception is very sensitive to the alignment errors around the stitching seams. This implies that we need to give the regions around the seams higher importance when aligning videos. We achieve this by sampling additional points within a certain distance from the seams. These new added points will naturally guide the mesh warping towards more accurate alignment around the seams.

Given a reliable set of correspondences (Figure 7), we divide the original frames into uniform grid meshes, and use the energy minimization technique proposed in [Liu et al. 2013] to derive the final warped mesh. Please refer to [Liu et al. 2013] for well-documented technical details.

	Time (s)		Time (s)
Global SfM	0.7166	Frame-level registration	0.0025
Pixel-level registration	3.5446	Mesh-based warping	1.7508
3D slice selection	1.9469	Blending	0.2188
Total	8.1802		

Table 1. Runtime of different components of our system in seconds per frame.

3.4 Blending

Given the warped videos and user-specified scene, various blending methods can be used to create the final composite. A simple solution is just to feather the seams and apply linear blending. Alternatively, one could use more advanced blending methods such as multi-band blending [Burt and Adelson 1983]. Most of the results shown in the paper are created using feathering; only the example in Figure 12 uses multi-band blending for a smoother transition.

3.5 Multiple videos

Our method can be naturally extended to handle more than two input videos. While it would be theoretically possible to optimize the alignment among all videos simultaneously, the computational cost of such an approach is high. We instead adopt a strategy similar to Videosnapping [Wang et al. 2014], which sequentially matches videos to a reference. We have found that such a simple method works quite well in practice (see Figure 9 for an example with three input videos).

4 RESULTS

Runtime. We evaluated the system on a desktop PC with two 2.3GHz Intel Xeon E5-2650 CPUs and one Nvidia Quadro K5200 GPU. The video resolution is 960×540, and computational costs of individual steps are listed in Table 1.

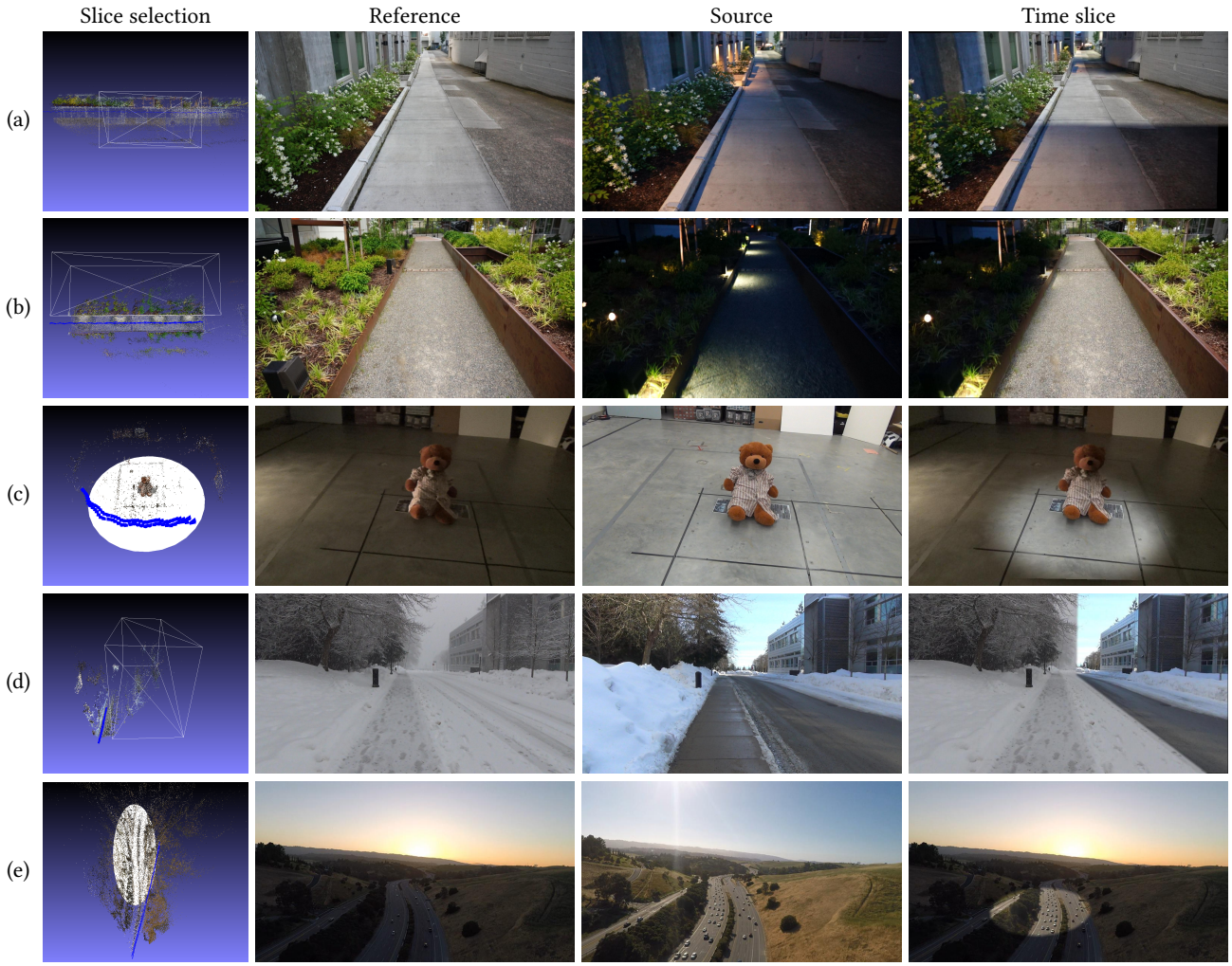


Fig. 8. Example time slice configurations showing the world-space (3D) slices. Please see supplemental result videos. From top to bottom: (a) Alley, (b) Garden, (c) Bear, (d) Snow and (e) Drone.

	SIFT flow	Guided flow	Guided flow + checking
w/ moving obj	7.44	4.00	2.06
w/o moving obj	7.34	3.18	2.66

Table 2. Quantitative evaluation on alignment error in pixels. See text for details.

Quantitative evaluation. Our pixel-level 2D alignment has two novel contributions: (1) using 3D scene points as guidance for SIFT flow computation; and (2) a robust check to remove unreliable matches. To evaluate how much they contribute to the alignment quality, we conduct a quantitative evaluation. We project 3D points to two input videos, and compare the average distance (or error) between pairs of projected 2D points after the same image warping driven by: 1) SIFT flow, 2) Guided SIFT flow, and 3) Guided SIFT flow with robust checking. The mean error (in pixels) over datasets with (e.g. Drone and Girl) and without (e.g. Garden and Bear) moving objects are listed in Table 2. The results suggest that both guided SIFT flow and robust checking play significant roles in improving the alignment accuracy. Note that we can only measure accuracy at known 3D points, in practice we have found that the visual quality improvement of the results after robust checking is far greater than the numbers indicate.

Comparisons. We compare our alignment method to a number of alternative solutions including pure image-based baseline methods, warping methods utilizing our 3D points, and recent mesh-based video stitching systems. Two datasets Garden and Bear are used for testing. Please see the supplementary video for a temporal alignment stability comparison.

For pure image-based baseline methods, we compare to optical flow [Kroeger et al. 2016] and regular SIFT Flow [Liu et al. 2011], computed on each pair of the matched frames. The video results show that optical flow alignment has very poor performance, which

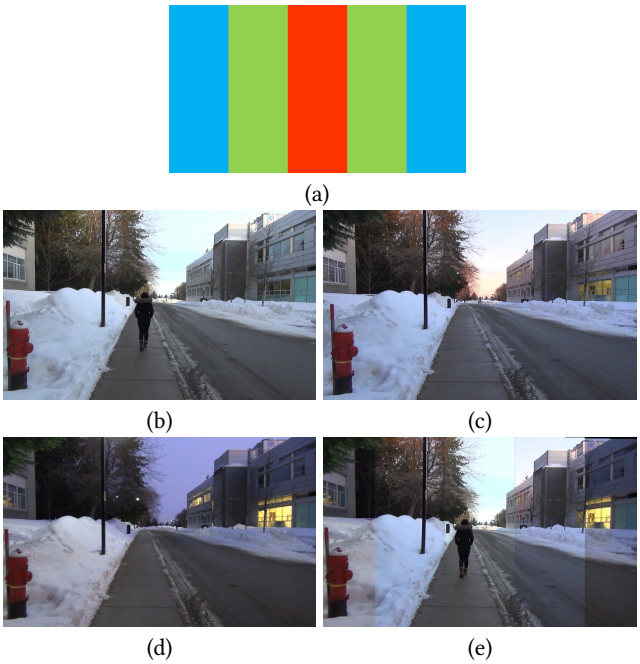


Fig. 9. Result of Walking with 2D slices. (a) is the image-space (2D) slice selection. (b), (c) and (d) are sample frames from the three input videos, and (e) is a frame from the synthesized video. Please see the supplemental video.

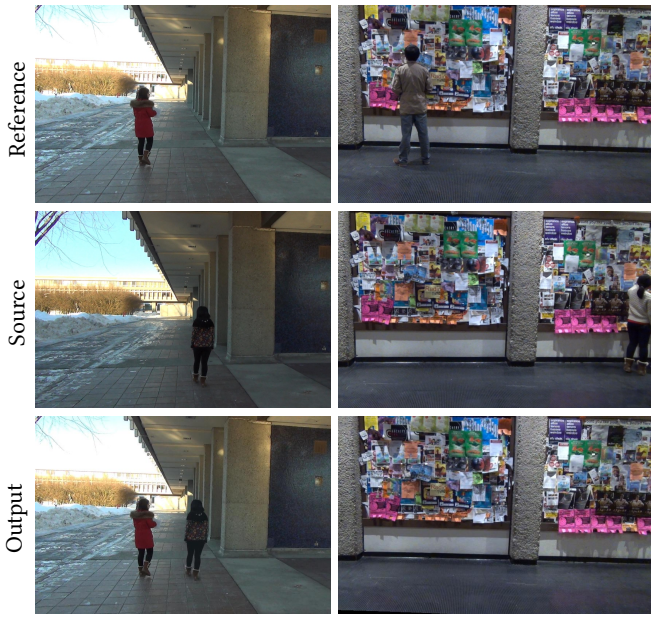


Fig. 10. Example of other applications, including compositing (left), and clean-plate extraction (right). Top two rows show the sample frames from two input videos, and the bottom row is the frame from the synthesized video. Please see the supplemental video.

is expected due to the limitations of the brightness constancy assumption. SIFT Flow performs slightly better, but without the guidance from 3D points in our approach, the correspondence fields are not stable and we see some obvious distortions in some image regions. We additionally compare to a commercial tool Nuke, which contains a video alignment node, likely based on feature matching and a global homography warp. This approach similarly cannot handle the large appearance difference between sequences.

We further compare to methods that use 3D points computed in our system. The most straightforward baseline would be to apply a single homography or mesh-based warp using the projection of the 3D points as constraints. Video results show that warping based on these 2D projections are sensitive to the accuracy, stability and distribution of 2D projections, causing obvious drifting in the video results.

Finally, we compare against two recent mesh-based video stitching methods [Guo et al. 2016; Lin et al. 2016]. Guo et al. [2016] integrate inter-sequence feature tracking with intra-sequence feature matching, while Lin et al. [2016] utilize 3D information from a stereo reconstruction. Given that direct matches across videos may be quite sparse when large appearance difference exists, mesh warping based on these sparse matches [Guo et al. 2016] will be inaccurate and unstable, leading to severe distortion and jitters in the video results. [Lin et al. 2016] only succeeds on the Bear example, and has obvious distortions on the ground due to noisy stereo reconstruction in that area. It totally failed on the Garden example as the stereo reconstruction could not work.

Time slice video. We have experimented with a variety of datasets, including both indoor and outdoor videos, with different kinds of motions (circling, forward, sideways), including hand held and drone-filmed footage. We demonstrate both image-space (2D) and world-space (3D) slices generated using our 3D editing interface. Sample frames of input videos and final synthesized videos are shown in Figure 8. We also show multi-video alignments in the Walking dataset, with three videos captured at noon, dusk and night (see Figure 9).

Other applications. While our method is motivated by time slice videos, we can use a temporally stable alignment for a number of other applications. One example is video compositing, e.g. transferring a segmented object from one video to another. In the Girl example shown in Figure 10 (left), we capture the same person walking twice, and then transfer the girl from the second sequence into the first sequence where she looks like talking to herself. The segmentation masks in this example are automatically computed based on thresholding the color difference after video alignment.

Another application is constructing clean plates from multiple takes. As long as there is no occluded region in *all* the videos, we can create a clean plate by combining the background parts. In this application, no precise segmentation is needed, and we can find any open place to blend the sequences. In Figure 10 (right), instead of masking out person in image space through the video ,which requires careful rotoscoping, we use our partial 3D reconstruction to specify rough areas e.g. the ground floor and the gray pillar as it is shown in Figure 11.

We can also generate time-lapse video, e.g. a person walking from day to night. We use Walking dataset and mask out the people in the first sequence with a video segmentation method [Märki et al.

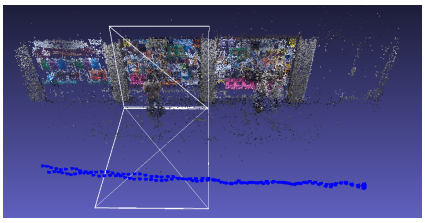


Fig. 11. 3D selection for the clean-plate example.

Fig. 12. Time-lapse video. We transfer the person in the first sequence (top row) to other sequences (bottom row). Please see the supplemental video.

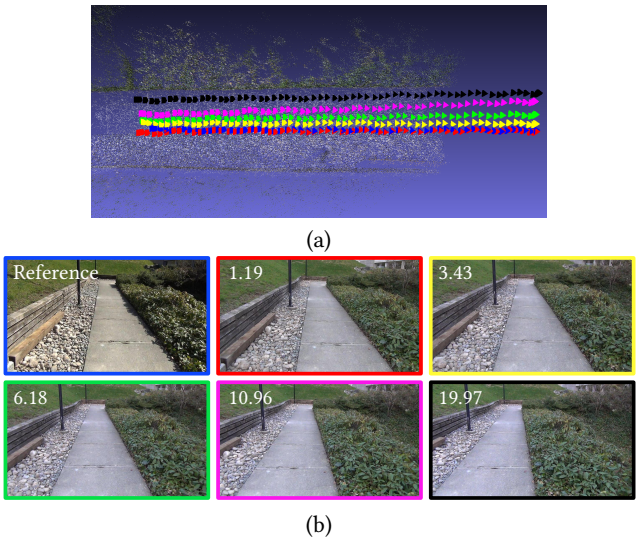


Fig. 13. Alignment error as related to parallax. (a) The camera paths of six videos in 3D space. The blue one is the path for the reference video. (b) The first frame of all input videos with increasing parallax. The color of the frame boundary corresponds to the color of its camera path. The number on the frame is the average alignment error in pixels from the current video to the reference video.

2016]. With all the other sequences registered to the first sequence, we can transfer the girl into other sequences easily, and generate a time-lapse video.

Effect of increasing parallax. Our system requires the input videos to have roughly the same camera path. To explore how robust our system is to the camera path difference of input videos, we conducted a stress test. As shown in Figure 13, we captured six videos of the same scene, with camera paths progressively farther from the reference one, resulting in increasing parallax shown in Figure 13(b). We then compute the average alignment errors from five source videos to the reference, shown at the top-left corner of each video frame. We note that for the source video (marked as red) which has the smallest parallax, its alignment error is the smallest and its visual quality is also the best. For the source video marked as yellow, the alignment error is still relative small (3.43) although it has obvious parallax. When the camera path difference becomes larger, the alignment error increases. However, our method does not fail miserably even when the parallax is quite significant. The video results for this test can be found in the supplementary material.

Limitations. Video alignment is a challenging problem, and while we found our method to be more robust than existing solutions, there are still cases where it can fail, and some artifacts remain. In our examples, we have tried to create as similar as possible camera trajectories. This is because the mesh warping step restricts our ability to correct for strong parallax effects. The effect of this can be seen in our parallax experiment, where warping artifacts become visible in regions that are close to the camera. However, there is always a balance between continuity and fidelity of the warp, and we found our approach provided the best compromise between them.

In addition, we require that we are able to obtain a reasonable 3D reconstruction of the scene using SfM. If the camera poses are badly recovered, it will influence both frame-level and pixel-level registration. As shown in Figure 14, as the night video has very bad quality, including heavy motion blur, the reconstruction is quite bad for this frame, which in turn influences final pixel registration.

We can also see occasional wobbling artifacts especially around the borders of the videos. This is because these regions often have very few constraints (sometimes there is no overlap with the other video at all), and so the warp has to extrapolate from the limited constraints that exist further inside the mesh. One solution, which is already employed in many productions, would be to always record a wider angle field of view than required at the end, so that observed points outside the view can constrain the warp.

5 DISCUSSION

In conclusion, we have presented the first robust solution to time slice video. Our approach is based off of a joint 2D-3D robust alignment system that outperforms other similar approaches. Additionally, we have demonstrated that *world-space* slices are possible, which gives rise to a new category of possible visual effects. One of the main challenges of creating time slice video is capturing the data, as it requires repeatedly following the same trajectory. We have included one example recorded on a drone, but drone cameras are well suited to capture this type of recurring camera trajectories.

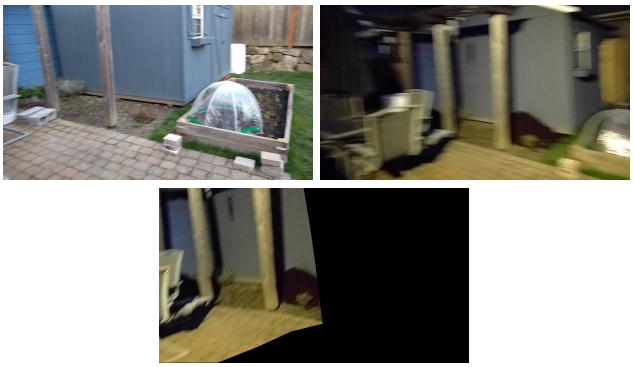


Fig. 14. Example of failure cases. As 3D reconstruction is not successful due to severe motion blur (top row), our final alignment is not accurate (bottom row).

One area for future work could be to more seamlessly integrate the mesh warping step with the dense pixel correspondences, for example by adaptive subdivision.

While our method makes use of SIFT descriptors, augmenting them with 3D registration, we believe that descriptors that are learned specifically for the dataset that we are trying to match are a promising way to improve the registration quality. Possibly, using the sparse 3D information to train a video-pair specific feature descriptor could improve the results.

ACKNOWLEDGEMENTS

We thank the Flickr user Miguel Mendez whose photograph we use under Creative Commons license[1]. We are grateful to Shuaicheng Liu and Kaimo Lin for providing the results of their methods in our comparisons, and to Renjiao Yi for her help in capturing the data. We would also like to thank all the reviewers for their constructive comments. This study is partially supported by Canada NSERC Discovery Grant 31-611664, Discovery Accelerator Supplement 31-611663, and a gift grant from Adobe.

REFERENCES

Robert Anderson, David Gallup, Jonathan T Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M Seitz. 2016. Jump: virtual reality video. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 198.

Peter J Burt and Edward H Adelson. 1983. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)* 2, 4 (1983), 217–236.

Zhaopeng Cui and Ping Tan. 2015. Global Structure-from-Motion by Similarity Averaging. In *Proceedings of the IEEE International Conference on Computer Vision*. 864–872.

Ido Freeman, Patrick Wieschollek, and Hendrik Lensch. 2016. Robust Video Synchronization using Unsupervised Deep Learning. *arXiv preprint arXiv:1610.05985* (2016).

Heng Guo, Shuaicheng Liu, Tong He, Shuyuan Zhu, Bing Zeng, and Moncef Gabbouj. 2016. Joint Video Stitching and Stabilization From Moving Cameras. *IEEE Transactions on Image Processing* 25, 11 (2016), 5491.

Berthold KP Horn and Brian G Schunck. 1981. Determining optical flow. *Artificial intelligence* 17, 1-3 (1981), 185–203.

Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. 2015. Sampling based scene-space video processing. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 67.

Johannes Kopf, Michael F Cohen, and Richard Szeliski. 2014. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 78.

[1]https://creativecommons.org/licenses/by/2.0/

Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. 2016. Fast Optical Flow using Dense Inverse Search. In *European Conference on Computer Vision*. Springer.

Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. 2014. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 149.

Jungjin Lee, Bumki Kim, Kyehyun Kim, Younghui Kim, and Junyong Noh. 2016. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 63.

Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J. Brostow, and Neill D.F. Campbell. 2016. Roto++: Accelerating Professional Rotoscoping using Shape Manifolds. *ACM Transactions on Graphics (In proceeding of ACM SIGGRAPH'16)* 35, 4 (2016).

Kaimo Lin, Shuaicheng Liu, Loong-Fah Cheong, and Bing Zeng. 2016. Seamless Video Stitching from Hand-held Camera Inputs. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 479–487.

Ce Liu, Jenny Yuen, and Antonio Torralba. 2011. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence* 33, 5 (2011), 978–994.

Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. 2009. Content-preserving warps for 3D video stabilization. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 44.

Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. 2013. Bundled camera paths for video stabilization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 78.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Vol. 2. Ieee, 1150–1157.

Nicolas Märki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. 2016. Bilateral space video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 743–751.

Meinard Müller. 2007. *Information retrieval for music and motion*. Vol. 2. Springer.

Federico Perazzi, Alexander Sorkine-Hornung, Henning Zimmer, Peter Kaufmann, Oliver Wang, S. Watson, and Markus H. Gross. 2015. Panoramic Video from Unstructured Camera Arrays. *Comput. Graph. Forum* 34, 2 (2015), 57–68.

Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. 2008. Nonchronological video synopsis and indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 11 (2008), 1971–1984.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 309–314.

Jan Rüegg, Oliver Wang, Aljoscha Smolic, and Markus Gross. 2013. Ducttake: Spatiotemporal video compositing. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 51–61.

Peter Sand and Seth Teller. 2004. Video matching. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 592–599.

Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. 2013. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 200.

Oliver Wang, Christopher Schroers, Henning Zimmer, Markus Gross, and Alexander Sorkine-Hornung. 2014. Videosnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 77.

Guofeng Zhang, Zilong Dong, Jiaya Jia, Liang Wan, Tien-Tsin Wong, and Hujun Bao. 2009. Refilming with depth-inferred videos. *IEEE Transactions on Visualization and Computer Graphics* 15, 5 (2009), 828–840.

Fan Zhong, Song Yang, Xueying Qin, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2014. Slippage-free background replacement for hand-held video. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 199.

Danping Zou and Ping Tan. 2013. Coslam: Collaborative visual slam in dynamic environments. *IEEE transactions on pattern analysis and machine intelligence* 35, 2 (2013), 354–366.