TrackCam: 3D-aware Tracking Shots from Consumer Video*

Shuaicheng Liu[1] Jue Wang[2] Sunghyun Cho[2†] Ping Tan[1,3]
[1]National University of Singapore [2]Adobe Research [3] Simon Fraser University

Figure 1: *Our system enables creating realistic tracking shots from shot video sequences, with little user input. The resulting tracking shot depicts object motion in a still image by adaptively blurring the background according to both foreground motion and scene structure in 3D.*

Abstract

Panning and tracking shots are popular photography techniques in which the camera tracks a moving object and keeps it at the same position, resulting in an image where the moving foreground is sharp but the background is blurred accordingly, creating an artistic illustration of the foreground motion. Such shots however are hard to capture even for professionals, especially when the foreground motion is complex (e.g., non-linear motion trajectories).

In this work we propose a system to generate realistic, 3D-aware tracking shots from consumer videos. We show how computer vision techniques such as segmentation and structure-from-motion can be used to lower the barrier and help novice users create high quality tracking shots that are physically plausible. We also introduce a pseudo 3D approach for relative depth estimation to avoid expensive 3D reconstruction for improved robustness and a wider application range. We validate our system through extensive quantitative and qualitative evaluations.

CR Categories: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Video analysis

Keywords: tracking shot, panning shot, 3D reconstruction, spatially-varying blur, motion depiction

Links: DL PDF

1 Introduction

Using intentional camera movement to introduce artistic blur in a still image for motion depiction is a classic photography technique. When done properly, the foreground object remains to be sharp in the final image while the background is blurred according to the foreground motion and scene structure, creating a vivid illustration of the intensive object motion in a single photograph.

This effect is traditionally called a ***panning shot***, if the camera moves horizontally and sweeps around the scene to follow the subject. However this is only limited to object motion that is parallel to the image plane. For capturing more complex object motion that is not parallel to the image plane, such as a running car shown in Figure 2, the camera has to move in the 3D space to track the object, which we call a ***tracking shot***[1]. Such techniques are known to be hard to master, as the photographer has to carefully plan the shutter speed, focal point and other camera settings beforehand, and more importantly, follow the subject in a steady and smooth way in the duration of the exposure, all requiring an accurate prediction of the subject motion and good photography skill sets. In practice, it often requires physically mounting the camera on the object itself (see Figure 2). As a result, high quality tracking shots usually can only be produced by professionals with dedicated equipment, and are difficult to capture for consumers with low-end devices such as cellphone cameras.

In this work, we aim to significantly lower the barrier of creating tracking shots, and make it accessible for consumer-level photography enthusiasts for the first time. Instead of directly capturing a single image, our system allows the user to capture a short video of the target object in motion, and employs modern computer vision techniques to assist the user to create a desired image through simple interactions.

Specifically, given an input video, we first ask the user to select a base frame as the targeted output, and mask out the moving object that the camera is supposed to track. The system then segments the object in the video, tracks feature points in the background and estimates the 3D scene. The system also estimates a dominant 3D motion trajectory of the foreground object. Using the 3D information from both the foreground and the background, we generate dense spatially-varying blur kernels, which are then applied to the base frame to render the final image.

Our system is built upon a combination of advanced computer vision techniques such as 3D reconstruction and video stabilization. We first show that "physically correct" tracking shots can be generated by reconstructing the whole scene in 3D. However, 3D reconstruction is known to be fragile and computational expensive, thus

*This work started when the first author was an intern at Adobe.
†Sunghyun Cho is now with Samsung Electronics.

[1]Tracking shot is also commonly used in videography for describing a video sequence captured in this way, in this paper we use it to describe a single photograph.

Figure 2: *Capturing a high quality tracking shot (left) usually requires using dedicated professional equipment such as car camera rigs (right). Note that the spatially-varying blur on the background reflects both the foreground motion and the scene depth. Image credits: Peter Adermark at Flickr.*

has a limited application range. To overcome this limitation, we further propose a pseudo 3D method to eliminate the requirement of 3D reconstruction for improved robustness. We show that the pseudo 3D approximation can lead to similar result quality, but has a wider application range.

2 Related work

We briefly review the most related work in a few areas.

Artistic blur. Blur is a natural phenomenon in photographs caused by object motion, camera shake or limited depth of field. Although it is an active research area in computer vision for removing blur from images [Fergus et al. 2006; Cho and Lee 2009] and videos [Cho et al. 2012], adding synthetic blur to an image is a popular artistic choice for either motion illustration or highlighting the main subject, and is supported by commercial software such as the Blur Gallery in Photoshop.[2] However, these tools only generate blur kernels with simple profiles such as disk, Gaussian, or directional lines, which cannot serve our purpose of simulating tracking shots.

For realistic motion blur simulation, Sung et al. [2002] introduced visibility and shading functions in a spatio-temporal domain, while Lin and Chang [2006] considered nonlinear intensity response of cameras in their image formation model. Brostow and Essa [2001] proposed to synthesize motion blur for stop motion animation from image sequences, by integrating scene appearance to imitate long camera exposure. This method however is limited to stop motion data with a fixed camera position, and is hard to be extended to videos captured by handheld cameras. Besides, all above systems are designed for simulating motion blur of moving objects rather than tracking shots.

3D reconstruction. Reconstructing 3D geometry of a rigid scene from 2D images was first studied by Longuet-Higgins [1981]. Research results of the following three decades in this area are well documented in [Faugeras et al. 2001; Hartley and Zisserman 2003]. Modern structure-from-motion (SfM) algorithms can recover a 3D rigid scene and camera motion simultaneously from tracked feature points in an image sequence, which we use for background 3D reconstruction in our 3D approach.

Recently, non-rigid SfM techniques have been developed to reconstruct time-varying shape and motion of dynamic objects from a monocular video. The majority of these techniques represent a non-rigid shape as a linear combination of some basis shapes [Bregler et al. 2000] and adopt the factorization-based approach introduced by Tomasi and Kanade [1992]. This approach is further extended to deal with articulated objects [Yan and Pollefeys 2005; Tresadern and Reid 2005]. However, all these methods require a large number of tracked feature trajectories, which are often not available in consumer video. While the method presented by Park et al. [2010] can reconstruct a single trajectory, it requires randomized camera motion and cannot be applied to a video clip. Ozden et al. [2004] addressed the problem of reconstructing 3D trajectories of moving objects from a monocular video using generic constraints for the analysis of dynamic scenes, with an emphasis on determining the relative motion scales among multiple independently-moving objects. We address this problem by proposing a novel method that combines trajectory triangulation [Avidan and Shashua 2000] [Kaminski and Teicher 2004] and motion priors such as constant velocity and/or acceleration.

It is well known that there are some fundamental issues that make SfM challenging in certain cases, such as lack of parallax, camera zooming, in-camera stabilization and rolling shutter [Liu et al. 2011]. To avoid these limitations, we further propose a pseudo 3D approach for creating tracking shots without explicitly relying on SfM.

Video stabilization. Videos captured by handheld cameras are often shaky and undirected. Video stabilization techniques can estimate the original camera motion with predefined motion models, and remove undesirable jitter to generate smooth camera paths that are more pleasant to watch. Depending on the specific types of camera motion models employed, existing video stabilization techniques can be classified as 2D [Matsushita et al. 2005; Liu et al. 2014], 3D [Liu et al. 2009; Liu et al. 2012] and 2.5D [Liu et al. 2011; Liu et al. 2013] methods. 2D approaches are generally robust but cannot handle scene parallax well. 3D methods are more accurate, but are less robust and computationally expensive. 2.5D approaches combine the advantages from both sides to be more practical. Our pseudo 3D approach relies on video stabilization for removing unwanted camera motion, as detailed in Section 4.

Image and video fusion. Our system is related to image and video fusion, i.e., combining information from multiple images/videos to create a single output. Agarwala et al. [2004] proposed an interactive Photomontage system for combining regions from multiple images together using Graph Cuts optimization. Bhat et al. [2007] proposed an approach to enhance videos using high resolution photographs, but only for static scenes. Chaurasia et al. [2013] introduced a depth-synthesis method for view interpolation and image-based navigation. Sunkavalli et al. [2012] proposed an approach to generate a single high-quality still image called a snapshot from a short video clip. This system employs multi-image super-resolution, and noise and blur reduction to enhance the visual quality of a snapshot, and also provides a visual summarization of activities in an input video. The DuctTake system [Rüegg et al. 2013] enables users to composite multiple takes of a scene into a single video. However, none of the above systems is designed for simulating tracking shots, the main goal of our system.

3 3D approach

As shown in the example in Figure 2, a visually convincing tracking shot should meet two criteria:

- the background blur should be spatially-varying according to the 3D scene geometry;
- the spatially-varying blur directions should be consistent with the dominant foreground motion in 3D.

[2] http://helpx.adobe.com/photoshop/using/blur-gallery.html

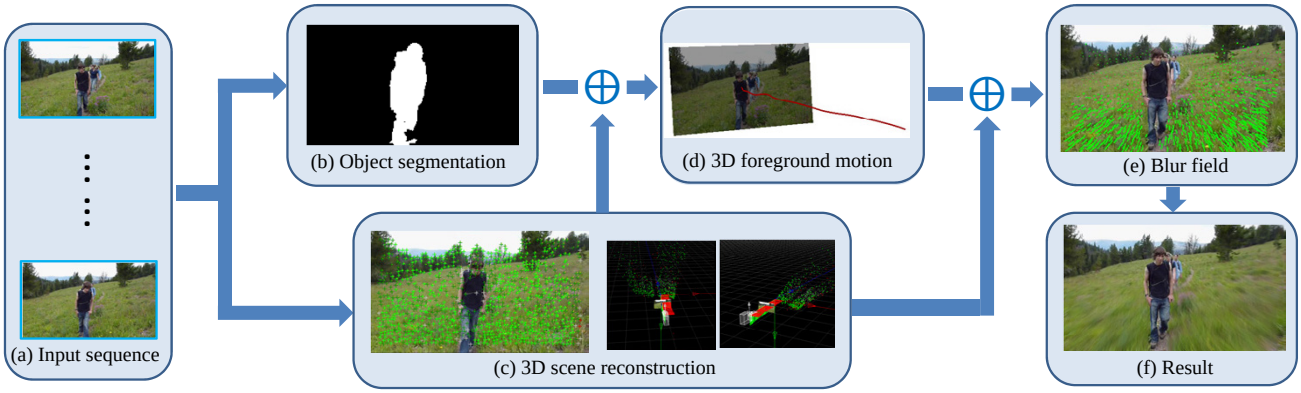


Figure 3: *The flowchart of the proposed 3D method. Given the input video sequence (a), we first obtain a rough foreground mask (b), then apply 3D scene reconstruction (c) and 3D foreground motion extraction (d). A spatially-vary blur field (e) is generated from (c) and (d). Final tracking shot (f) is obtained by applying the blur field to the input base frame.*

Both criteria need to be 3D-aware, which motivates us to first design an approach with explicit 3D reconstruction. The flowchart of this approach is shown in Figure 3. Given an input video, we first segment the foreground moving object using existing interactive solutions (Figure 3b). For our application, we do not need accurate segmentation masks, and a rough segmentation is often sufficient for the following steps. We found the RotoBrush tool in Adobe After Effects which is based on video SnapCut [Bai et al. 2009] fits our need well, as it provides both quick scribble-based selection on the keyframe and efficient automatic segmentation propagation.

Next, we extract a dominant, smooth 3D motion trajectory (Figure 3d) for the segmented foreground, along which the simulated virtual camera should move to track the object (Section 3.1). We also use the feature trajectories in the background to reconstruct the 3D scene using the publicly available Voodoo SfM system [3], as shown in Figure 3(c).

Finally, as detailed in Section 3.2, we simulate a virtual camera that moves along the extracted 3D foreground motion trajectory. By projecting the 3D background scene points into this camera, we obtain sparse and spatially-varying blur kernels, as shown in Figure 3(e). We then use edge-aware interpolation to create a dense blur field from the sparse constraints, and apply it to the selected base frame to render the final output, as shown in Figure 3(f). The amount of blur is controlled by the duration of the exposure of the virtual camera, which is adjustable as a user parameter.

3.1 Recovering 3D foreground motion trajectory

An important component of the proposed 3D approach is to recover the dominant 3D foreground motion trajectory, which defines the path of the virtual camera for object tracking. Since the moving object has already been segmented in all frames, we simply concatenate the center of the object mask on each frame to obtain an approximate 2D trajectory of the object. This trajectory is further smoothed by a temporal Gaussian filter with a window size of 10 to reduce jitters. We then recover the 3D trajectory from its 2D projection, which is related to the problem of non-rigid SfM.

The majority of the non-rigid SfM techniques, e.g. [Bregler et al. 2000; Yan and Pollefeys 2005], require a large number of long feature trajectories of related but different motions, e.g. trajectories of different joints of an articulated object. Thus, they are not directly applicable in our application, since there is only one approximate 2D trajectory of the moving foreground object.

We thus develop a new algorithm to address this issue. Specifically, we first move a sliding window in the temporal domain, which cuts a 2D trajectory into short 2D sub-trajectories. We then recover a 3D sub-trajectory from each 2D sub-trajectory, based on the assumption that the 3D sub-trajectory is close to a linear motion as the sub-trajectory is short, i.e., it roughly forms a line. The final 3D trajectory is computed as the mean of all the overlapping local sub-trajectories, which can be far from linear.

We denote a 3D sub-trajectory by $\{X_i; 1 \leq i \leq K\}$, where i is the frame index and X_i is the 3D Cartesian coordinates at the i-th frame. K is the number of frames in the sub-trajectory. Each 3D sub-trajectory is then recovered by minimizing the energy function:

$$\begin{aligned} E_l(\{X_i\}, L) &= \lambda_1 E_{algebr}(\{X_i\}) + \lambda_2 E_{line}(\{X_i\}, L) \\ &+ \lambda_3 E_{motion}(\{X_i\}) + \lambda_4 E_{pers}(\{X_i\}). \quad (1) \end{aligned}$$

Here, L is an auxiliary variable representing a 3D line, which is fit to the 3D sub-trajectory $\{X_i\}$. E_{algebr}, E_{line}, E_{motion}, and E_{pers} accounts for constraints derived from algebraic error, linear motion, const motion velocity (or acceleration), and perspective effects, respectively. λ_1, λ_2, λ_3, and λ_4 are weights for these constraints. We alternatingly minimize $E_l(\{X_i\}, L)$ with respect to $\{X_i\}$ and L. In the following, we describe each constraint in detail.

Algebraic error constraint Suppose the corresponding 2D sub-trajectory is $\{p_i; 1 \leq i \leq K\}$. For convenience, p_i are homogeneous coordinates. Also suppose that the camera matrix at the i-th frame is P_i, which is estimated beforehand from static scene points by standard SfM techniques. Then, X_i should be projected onto p_i by P_i, leading to the projection relation which is defined up to a scale: $[p_i]_\times P_i \hat{X}_i = 0$, where $\hat{X}_i = [X_i^\top, 1]^\top$ is a homogeneous coordinate representation of X_i. $[\cdot]_\times$ is the skew-symmetric matrix form of the cross product [Hartley and Zisserman 2003], i.e. $a \times b = [a]_\times b$. After some derivation, we get the following relation: $[p_i]_\times P_{i,1:3} X_i \simeq [p_i]_\times P_{i,4}$, where $P_{i,1:3}$ is the matrix formed by the first three columns of P_i, and $P_{i,4}$ is the fourth column of P_i. Based on this relation, we define E_{algebr} as:

$$E_{algebr}(\{X_i\}) = \sum_i \| [p_i]_\times P_{i,1:3} X_i - [p_i]_\times P_{i,4} \|^2. \quad (2)$$

[3] www.digilab.uni-hannover.de

Linear motion constraint As mentioned earlier, we assume that a 3D sub-trajectory is close to a linear motion, i.e., $\{X_i\}$ roughly forms a line L. We take the Plücker representation, where a line passing through two 3D points with homogenous coordinates A and B is represented by a 4×4 skew-symmetric matrix $L = AB^\top - BA^\top$. This line is uniquely determined by a 6D vector $\mathcal{L} = [l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}]^\top$, where l_{ij} is the (i,j)-th element of the matrix L. We use $\mathcal{P}_i$ to denote the line projection matrix [4] of the i-th video frame. It is proved in [Avidan and Shashua 2000] that

$$p_i^\top \mathcal{P}_i \mathcal{L} = 0. \tag{3}$$

Avidan and Shashua [2000] solved $\mathcal{L}$ from at least five different camera positions with sufficient viewpoint changes, i.e. the row vectors $p_i^\top \mathcal{P}_i$ collected from all images are rank 5. However, in our application, the camera moves smoothly and often provides insufficient constraints to solve the linear trajectory. In practice, we find these vectors typically have rank 2 or 3.

Additionally, since X_i lies on the line L, it should satisfy the line constraint:

$$L^* \hat{X}_i = 0, \tag{4}$$

where L^* is the dual skew-symmetric matrix of L, which can be obtained from L directly by a simple rewriting:

$$l_{12} : l_{13} : l_{14} : l_{23} : l_{42} : l_{34} = l^*_{34} : l^*_{42} : l^*_{23} : l^*_{14} : l^*_{13} : l^*_{12}.$$

By combining Equations 3 and 4, we define E_{line} as:

$$E_{line}(\{X_i\}, L) = \sum_i \|p_i^\top \mathcal{P}_i \mathcal{L}\|^2 + \sum_i \|L^* \hat{X}_i\|^2. \tag{5}$$

Constant velocity (or acceleration) constraint We assume the foreground object to have near constant velocity or acceleration in the duration of a 3D sub-trajectory. Since the video frames are captured with a constant time interval, these two constraints are simply formulated as:

$$X_i - X_{i-1} = X_{i+1} - X_i, \tag{6}$$

$$X_{i+1} + X_{i-1} - 2X_i = X_{i+2} + X_i - 2X_{i+1}. \tag{7}$$

Again, by combining Equations 6 and 7, we define E_{motion} as:

$$E_{motion}(\{X_i\}) = \sum_i \|X_{i+1} + X_{i-1} - 2X_i\|^2 + \sum_i \|X_{i+2} + 3X_i - 3X_{i+1} - X_{i-1}\|^2. \tag{8}$$

Perspective constraint Under a perspective camera, the apparent size of an object is inversely proportional to its depth. We use $D(X, P)$ to denote the depth of a point X in the camera P. Specifically, the principal axis direction m of a camera can be determined from its projection matrix P as $m = [P_{31}, P_{32}, P_{33}]^\top$. Here P_{ij} is the (i,j)-th element of the 3×4 camera projection matrix P. After normalizing m to a unit vector, the depth can be computed as $D(X, P) = m^\top X$, which is a linear function of X. This leads to the following relation between any two frames i and j:

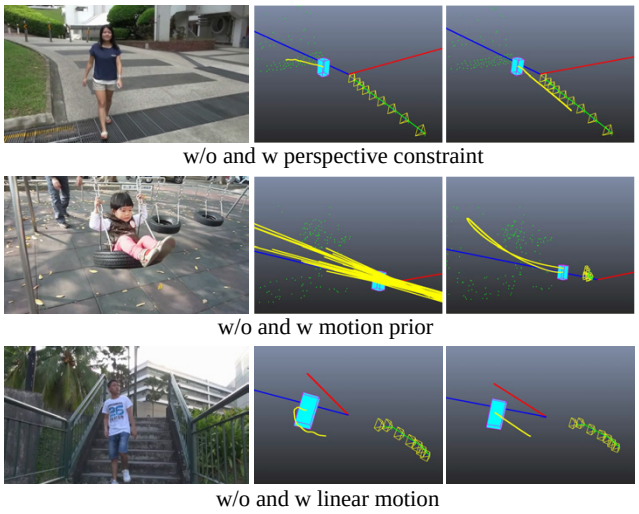
$$D(X_i, P_i) : D(X_j, P_j) = 1/S_i : 1/S_j, \tag{9}$$

where S_i is the foreground segmentation size in the i-th frame. Based on Equation 9, we define E_{pers} as:

$$E_{pers}(\{X_i\}) = \sum_{|i-j|>\delta} \left\| \frac{D(X_i, P_i)}{S_j} - \frac{D(X_j, P_j)}{S_i} \right\|^2, \tag{10}$$

where we set $\delta = 10$ for robustness.

[4] The line projection matrix $\mathcal{P}_i$ can be easily obtained from its ordinary 3×4 projection matrix P_i. Please see page 198 of the textbook [Hartley and Zisserman 2003].



w/o and w perspective constraint

w/o and w motion prior

w/o and w linear motion

Figure 4: *Examples of reconstructed 3D foreground motion trajectories (yellow curves). Green dots are background points. Each row shows a comparison of without and with one of the constraints in Equation (1) in solving the sub-trajectories.*

Implementation In our implementation, we set $\lambda_1 = \lambda_2 = \lambda_4 = 1$ and $\lambda_3 = 10$. Except the bi-quadric term $\|L^* \hat{X}_i\|^2$ in Equation (5), all the other terms are quadratic functions of either L or $\{X_i\}$. Thus, we iteratively estimate L and $\{X_i\}$ by keeping one fixed at a time. For initialization, we first estimate $\{X_i\}$ by minimizing Equation (1) without including E_{line}.

In all our experiments, we fix the length K of each sub-sequence to 20, and let neighboring sub-sequences overlap for 10 frames. Some examples of the estimated foreground motion trajectory are provided in Figure 4. In each row, the left image is a sample frame of the input video. The right image shows the 3D trajectory of the foreground object (the yellow curve), the cameras, and some static background points (the green dots). In each row, we evaluate our method by turning off each constraint one by one. The results suggest that every component is important to ensure reliable reconstruction of the foreground motion.

3.2 Spatially-varying blur map

Once the foreground 3D motion trajectory is determined, we combine it with the background structure to generate a spatially-varying blur field. The final tracking shot is rendered by convolving the sharp base video frame with this blur field.

To simulate a physically-correct tracking shot, we place a series of virtual cameras along a 3D trajectory according to the foreground motion and virtual exposure duration. Figure 5 shows an example. Specifically, a virtual camera consists of an intrinsic matrix K_v, a rotation matrix R_v and a camera center C_v. A projection matrix P_v can be derived as $P_v = K_v[R_v \ -C_v]$. From the 3D reconstruction of the static background, we obtain the projection matrices of the real camera, which also consist of intrinsics K_r, rotations R_r and camera centers C_r. We borrow the real intrinsics for the virtual cameras, i.e. $K_v = K_r$. The rotation matrices of the virtual cameras are fixed, and set to be the same as the base frame at time t as $R_v = R_r^t$. The camera centers C_v are equally sampled along a 3D trajectory which passes through the camera center C_r^t at time t and

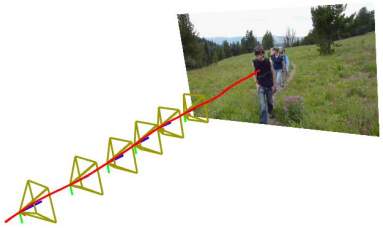


Figure 5: *The virtual cameras sampled along a trajectory (red curve) that passes through the camera center of the base frame and has the same shape as the foreground 3D motion trajectory.*

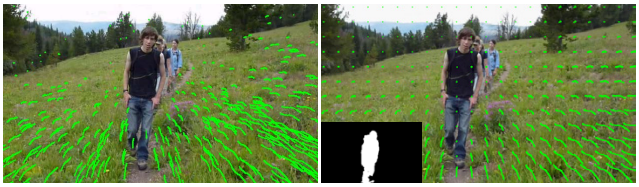


Figure 6: *Generating spatially-varying blur kernels. Left: blur kernels generated by projecting 3D scene points onto virtual cameras. Right: final interpolated dense blur kernel map (sampled on a regular grid for visualization).*

has the same shape as the foreground 3D motion trajectory. The spatial extent of these sampled camera centers is determined by the virtual exposure duration, which the user controls with a sliderbar.

Next, we generate blur kernels by projecting 3D background points using the virtual cameras. Specifically, for each reconstructed 3D point in the background, we project it using the virtual cameras and obtain a set of 2D projections. This set represents a blur kernel, because it describes how each 3D point moves on the image plane while the virtual camera moves during the exposure time. Each 3D point yields a different set of 2D projections, so the resulting blur kernels are spatially-varying as shown in Figure 6. In our system, we use 20 virtual cameras to generate blur kernels.

Since feature points are sparse, we have only obtained sparsely-distributed blur kernels and need to obtain a dense blur kernel map from them. This is a very similar problem to interactive tonal adjustments from sparse user controls [Lischinski et al. 2006], and can be achieved using edge-aware interpolation approaches which respect strong appearance changes that are usually caused by depth discontinuity. In our system we adopt the edge-aware optimization-based propagation method [Levin et al. 2004] given its robustness and high quality results. Finally, to keep the foreground object sharp in the final result, we force the blur kernels inside the foreground segmentation mask to be a delta function. Figure 6 shows an example of a final dense blur kernel map.

4 Pseudo 3D approach

The 3D method described in Section 3 relies on both rigid and non-rigid SfM, which are both computationally expensive and less reliable on consumer videos. In this section, we derive an approximation method that does not require recovering physically-correct 3D information. We refer this method as a pseudo 3D method, given that we only hallucinate pseudo 3D information from an input video. This method follows the same pipeline shown in Figure 3, the only difference is that both foreground and background 3D reconstruction steps (Figure 3 (d) and (c)) are replaced with new hallucination methods.

4.1 3D background hallucination

It is well known that the disparities of matched image features are inversely proportional to their distances to the image plane [Kanade and Okutomi 1994]. This gives us strong cues to hallucinate pseudo depths for feature tracks.While conventional 3D reconstruction methods require calibrated cameras to compute depth from disparities, we use the relative disparities at different points to approximate their relative depths without camera calibration, which is sufficient for our application. This can be done by two simple steps presented in the following.

Stabilization and relative disparities When the camera undergoes only translational motion, the disparity of a feature point can be simply measured by the length of its motion vector. However, when the camera is also rotating, the motion consists of two components caused by rotation and translation respectively. Ideally, we need to minimize the rotational camera motion to measure the disparities, which is possible through the 3D video stabilization [Liu et al. 2009] or the gyroscope based stabilization [Karpenko et al. 2011]. Here, we want to achieve this goal without resorting to 3D reconstruction or special sensors.

We apply a conventional 2D video stabilization method [Matsushita et al. 2005], which models the motion between consecutive video frames using a 2D homography. A homography only captures the global camera motion to roughly align consecutive frames. While this stabilization changes the absolute disparities, the relative disparities at points of different depths remain alive after stabilization. In other words, faraway points still have smaller disparities than closer points, which allows us to hallucinate a pseudo 3D reconstruction. Therefore, we compute disparities as the length of motion vectors on the stabilized video. We empirically find this simple method works well for our application.

Depth hallucination After stabilizing the input video, we compute a pseudo depth for each feature point based on its disparities between frames. Suppose q_i^t is the coordinates of the i-th feature point on the t-th frame. Its overall disparity across different frames is computed as $T_i = \sum_t \|q_i^t - q_i^{t-1}\|$. Typically, this disparity is quite noisy. We thus develop a method to spatially smooth the disparity values of nearby tracks. We compute the average of all T_i as $\bar{T}$, and estimate the relative disparity α_i at q_i by minimizing,

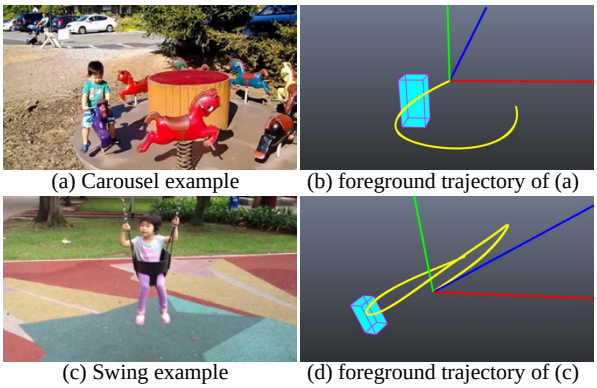
$$E_{\bar{T}}(\{\alpha_i\}) = \sum_i (\|\alpha_i \bar{T} - T_i\|^2 + w \sum_{j \in N_i} \|\alpha_i - \alpha_j\|^2). \quad (11)$$

The first term is a data fitting term that encourages the new disparity $\alpha_i \bar{T}$ to be close to the original measure T_i. The second term is a spatial smoothness term encouraging nearby points to have similar disparity values, under the assumption that they are likely to be on the same object with similar depth. N_i refers to the neighboring tracks of q_i, defined in an edge-aware way by thresholding both the spatial and the color distances. Two tracks are neighbors if their spatial distance is always within 40 pixels and color difference is always smaller than 0.1 in the RGB space (for pixel values within $[0, 1]$). w is a balancing weight we fix at 0.5.

The quadratic energy function in Equation (11) is simply minimized by a linear least square solver. The obtained α_i contains less noise and is spatially smoother than T_i. The depth of q_i is computed directly from α_i as:

$$D_i^B = 2^{(1/\alpha_i)} * \beta, \quad (12)$$

where β is a scale value which we will discuss in Section 4.3.



(a) Carousel example (b) foreground trajectory of (a)

(c) Swing example (d) foreground trajectory of (c)

Figure 7: *Hallucinated 3D foreground motion trajectories.*

With the estimated depth, we can compute the pseudo 3D coordinates of all the tracked scene points in the camera coordinate system. Specifically, the 3D coordinates (X_i, Y_i, Z_i) of q_i are:

$$\begin{cases} X_i &= D_i * (x_i - c_x)/f \\ Y_i &= D_i * (y_i - c_y)/f \\ Z_i &= D_i \end{cases} . \tag{13}$$

Here, D_i is the depth of q_i, f is the camera focal length, (c_x, c_y) is the principal point of camera intrinsics (fixed at the image center in our system), and (x_i, y_i) is the image coordinates of q_i in the selected base frame.

4.2 3D motion trajectory hallucination

We rely only on the 'Perspective Constraint' proposed in the 3D method to hallucinate the 3D motion trajectory of the foreground object. Basically, we first seek to remove all the camera motions from the input video. After that, we use the size of the segmented foreground region to hallucinate the depth of the moving foreground object.

Specifically, we align all the frames to the selected base frame by homography transformations. The segmentation masks are transformed by the same homographies. The depth of the foreground object D_t^F at frame t is simply set to be inversely proportional to its segmentation size at that frame, i.e.

$$D_t^F = \frac{\gamma}{S_t}, \tag{14}$$

where S_t is the size of foreground segmentation at frame t, γ is a scale parameter, which will be discussed in Section 4.3. The 3D position of the foreground object is computed according to Equation (13) from its depth. Figure 7 shows two examples of the hallucinated trajectory.

4.3 Combining pseudo 3D

The foreground and background 3D information is computed using two different methods and is often inconsistent. We thus need to adjust the two scalars β in Equation (12) and γ in Equation (14) to minimize this inconsistency. In practice, we fix γ to 1 and only adjust β. Effectively, β controls how far the scene points are from the camera, which determines the size of the generated blur kernels. So effectively, this parameter acts similar to the virtual exposure time in our 3D method. For example, Figure 8 shows the sparse blur kernels of one example with different settings of β. From left to right, these images correspond to $\beta = 0.1, 1$, and 10 respectively.

Determining exactly how much blur should be synthesized on each example could be subjective. In our user interface we allow the user to adjust β within $[0.1, 10]$ using a sliderbar. This provides the flexibility to synthesize different amount of blur, mimicing different exposure duration in real tracking shots. Once β is determined, we use the method described in Section 3.2 to synthesize the final result.

(a) $\beta = 0.1$ (b) $\beta = 1$ (c) $\beta = 10$

Figure 8: *The scale value β in Equation (12) controls the size of blur kernels. A smaller β leads to a larger amount of blur.*

5 Experimental results

Implementation details We implemented our system using C++. We run our method on an Intel i7 3.2GHZ Quad-Core machine with 8G RAM. Input videos typically contain 80-120 frames. For 3D background reconstruction, the Voodoo SfM system normally takes about 3-5 minutes for a 10 second video. Foreground segmentation using the RotoBrush tool takes 2-3 minutes. Foreground path recovery takes less than a second. Generating sparse blur kernels is also very fast, and dense kernel interpolation takes around 3 seconds. For the pseudo 3D method, background reconstruction takes about 10 seconds, mainly on the OpenCV KLT tracker. The foreground trajectory recovery takes about 2 seconds.

Note that although our systems relies on two third-party packages for foreground segmentation and background SfM, both the RotoBrush and Voodoo system are publicly available and have been used in various recent research systems as building blocks. We used both systems with their default parameters without any special tweaking. The RotoBrush system requires user interaction, however since our application does not require highly accurate foreground masks, we found that in general the user only needs to apply occasional corrections on one or two frames other than segmenting the base frame for most examples shown in the paper.

Representative results Figure 9 shows a few results generated from the 3D method and the pseudo 3D method. For each example, the base frame is shown on the left, and the simulated tracking shot is shown on the right. The input videos include both slow and fast, linear and non-linear foreground motions. The results show that the spatially-varying blur generated by both methods reflects the correct scene geometry and the foreground object motion. Please refer to the project page[5] for more results.

6 Comparison with Alternative Approaches

Given that the final effect our system produces is the blurred background, one may wonder if some alternative simple solutions can be used to generate good results. We first introduce a naïve 2D method and a manual tool as two alternative approaches. Later, we compare them with our method by a quantitative evaluation on synthetic data and a user study with real images.

[5] http://www.liushuaicheng.org/trackcam/index.html

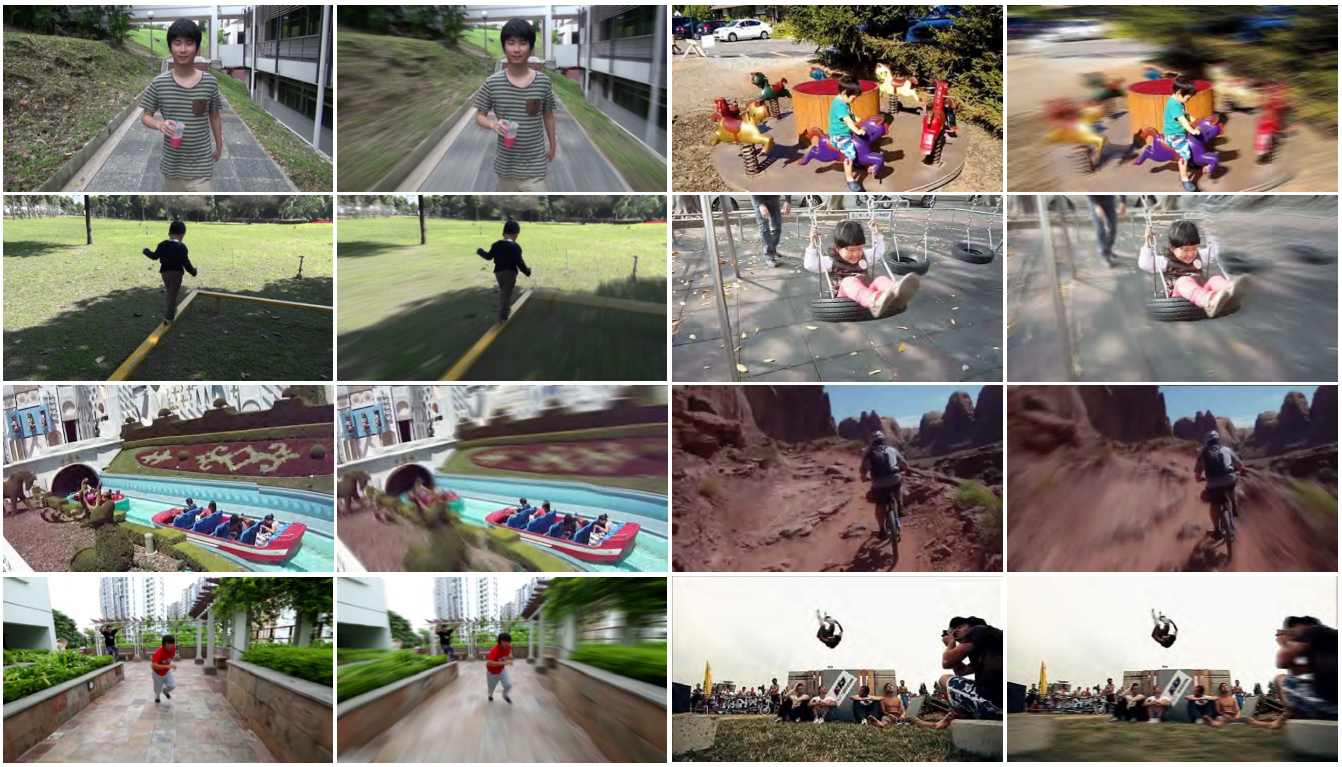


Figure 9: *Example tracking shots generated by our system. The first and third columns are base frames. The second and fourth columns are results. The first two rows are generated by the 3D method. The last two rows are produced by the pseudo 3D method.*

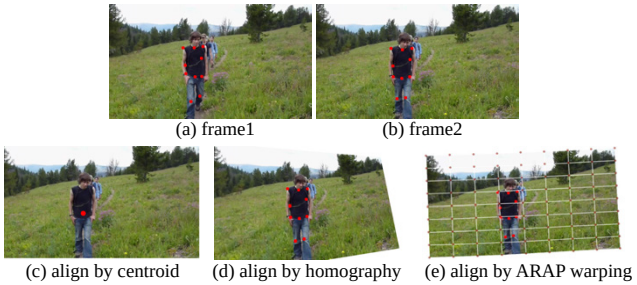


Figure 10: *Three alignment methods of the naive 2D approach.*

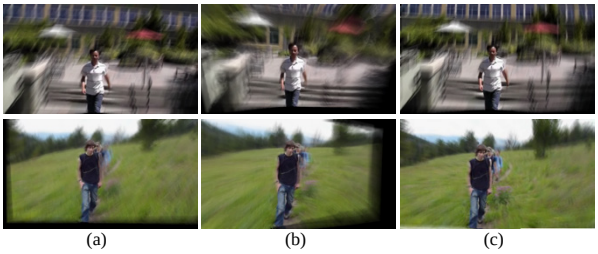


Figure 11: *Tracking shot results by the naïve 2D method. (a) align by a translation. (b) align by a single homography. (c) align by an ARAP warping.*

6.1 A naïve 2D method

A straightforward idea is to align the foreground object first, then compute the optical flow in the background region, and generate the motion blur kernels according to the smoothed flow field. Here we explore the effectiveness of this naïve 2D approach.

We manually mark control points on the foreground throughout the video, since automatical feature tracking (e.g. KLT) is brittle for non-rigid and fast-moving objects. The red dots in Figure 10(a) and (b) show examples of the control points. We use these control points to align the foreground through three motion models: (1) a translation computed from the centroid only; (2) a single homography computed from all control points; and (3) an as-rigid-as-possible(ARAP) warping [Igarashi et al. 2005]. Effectively, we stabilize the foreground by adopting single homography-based video stabilization [Matsushita et al. 2005] for translation- and homography-based alignment, and bundled paths video stabilization [Liu et al. 2013] for ARAP alignment. The results of the three alignment methods are shown in Figure 10(c),(d),(e). Note that since the foreground and background do not reside in a same plane, a single homography would introduce large displacement on the background, which may cause difficulty in optical flow computation. We calculate optical flow [Liu 2009] between neighboring frames on the 'foreground-stabilized' video. We concatenate flow vectors around the base frame to form blur kernels. Figure 11 shows the results. Please compare the corresponding results generated by our method in Figure 1 and in the supplementary video, which suggest that our method generates more visually appealing results. This is not surprising because stabilizing dynamic foreground is difficult, which makes the optical flow calculation on the background unreliable.

Figure 12: *Our manual tool to create tracking-shots. The yellow curves are blur traces manually drawn by the user.*

6.2 The manual tool

There are commercial software packages for manually generating artistic blur such as Photoshop Blur Gallery and Google Analog Efex Pro 2. They are however limited either to spatially invariant blur, or a small number of user-specified blur kernels. Thus, they do not provide full functionality for creating tracking shots where the blur is spatially-varying and needs to be carefully and densely specified.

We thus developed an interactive tool that allows the user to create more sophisticated blur by manually specifying curve-based spatially-varying blur traces. Our tool also provides live preview whenever the user adds, modifies, or removes a blur trace. We interpolate these traces using the method proposed in Section 3.2 to render a tracking shot. For a fair comparison, we do not allow the users to specify blur kernels inside the foreground mask that is used in our methods, to ensure that the foreground object has the same degree of sharpness in all results. Figure 12 shows some examples with manually specified blur traces using this UI. On average it takes about $3 \sim 5$ minutes for users to create a tracking shot using this tool. Please refer to the supplementary video for a live demonstration of this tool.

6.3 Quantitative evaluation on synthetic data

Synthetic data We create several synthetic examples in Maya and use them to conduct a quantitative evaluation. Figure 13 shows one such example. We put a moving foreground object in a virtual 3D environment, and render the scene twice by two different cameras: a hand-held camera (green in Figure 13 (a)) and a tracking camera (white in Figure 13 (a)). The hand-held camera follows a shaky and irregular motion, which yields a video sequence as the input to our system. The tracking camera has the same motion trajectory as the moving foreground, and is used to render the ground-truth blur kernels and tracking shot. Both cameras share a common base frame, as shown in Figure 13 (b). Figure 13 (c) shows the ground truth depth map of the base frame. We sample dense 3D points from this depth map, and project them to the tracking camera to generate the dense blur kernels as shown in Figure 13 (d). Finally, the ground-truth blur kernels are used to blur the background of the base frame, which is combined with the segmented foreground in the base frame to generate the ground-truth tracking shot. We synthesize six examples as shown in Figure14, please refer to the supplementary video for these examples.

Evaluation For each example, we compute the results of our 3D method, pseudo 3D method, the naïve 2D approach, and the manual approach (with two different users). Some of the results are shown in Figure 15.We then compare these results against the ground-truth in terms of (1) blur kernel similarity, measured by Normalized Cross-Correlation (NCC); and (2) final image similarity, measured by both PSNR and SSIM [Wang et al. 2004]. Table 1 to 2 show the quantitative results of different methods, which suggest that our 3D and pseudo-3D methods generate consistently better results than the alternative choices. Although the manual tool generates better results than the naïve 2D approach, it is still significantly inferior to our approaches. Note that our methods require much less user effort; and furthermore, the required user input is more intuitive (i.e. foreground segmentation), compared with drawing depth- and motion-dependent blur traces.

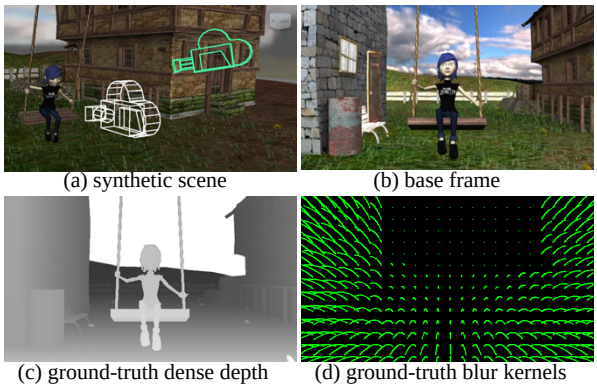


Figure 13: *Synthetic examples for quantitative evaluations.*

Table 1: *Blur kernels similarity (NCC) of different methods (columns) on difference synthetic examples (rows) against the ground-truth.*

NCC	3D	pseudo	naive	subject A	subject B
swing	**0.63**	0.59	0.27	0.47	0.44
carousel	**0.73**	0.68	0.30	0.59	0.35
run1	0.54	**0.56**	0.31	0.41	0.32
run2	0.58	**0.59**	0.26	0.42	0.34
car1	0.66	**0.67**	0.29	0.51	0.43
car2	**0.70**	0.68	0.22	0.44	0.39

3D vs. pseudo 3D The quantitative results show that our 3D method and pseudo 3D method generate very similar results in terms of both the blur kernel similarity and final image similarity. In these synthetic examples, the 3D method generates better results for the 'swing' and 'carousel' example, which contain significantly non-linear foreground motion. On the other hand, the pseudo 3D approach produces slightly better numerical results on other examples where the foreground motion is closer to straight lines. However, the differences are too small to be visually noticeable, as shown in Figure 15. More examples are in the supplementary material.

Table 2: *Image similarity (top: PNSR, bottom: SSIM) of different methods (columns) on difference synthetic examples (rows) against the ground-truth.*

	3D	pseudo	naive	subject A	subject B
swing	**33.36**	32.28	22.54	26.40	25.07
	0.964	0.959	0.807	0.858	0.877
carousel	**34.12**	31.29	25.26	27.80	26.48
	0.985	0.969	0.894	0.928	0.882
run1	32.01	**34.32**	28.49	27.41	26.57
	0.975	**0.976**	0.929	0.919	0.899
run2	34.74	**35.64**	27.46	28.27	26.96
	0.986	0.978	0.927	0.933	0.913
car1	30.35	**32.53**	23.37	28.14	25.22
	0.948	0.945	0.730	0.884	0.818
car2	**31.55**	29.48	21.41	22.07	21.52
	0.939	0.923	0.732	0.776	0.740

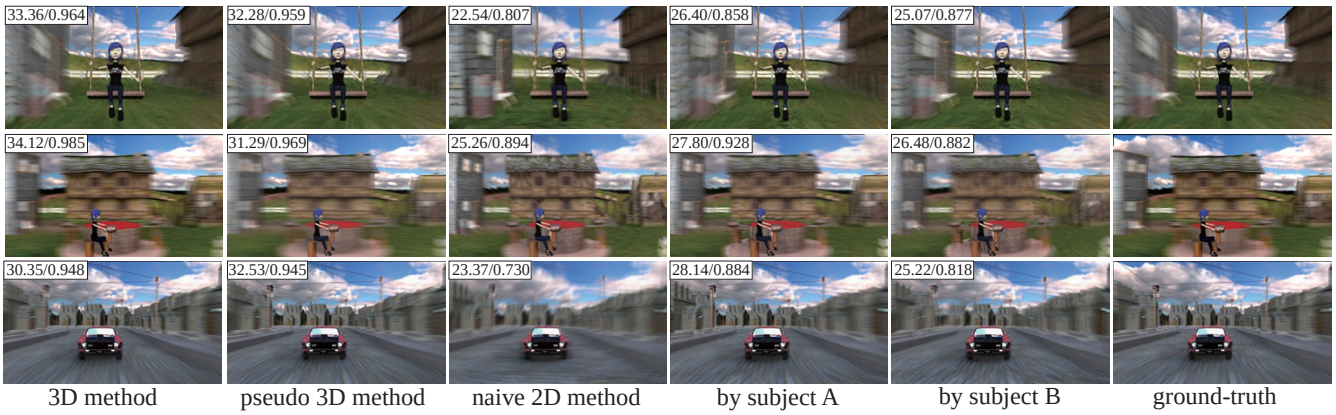


Figure 15: *Comparisons on the synthetic data set. The PSNR/SSIM values are labeled on the top-left corner of images.*

Figure 14: *Thumbnails of the synthetic examples: swing, carousel, run1, run2, car1 and car2.*

6.4 A user study

Since the quantitative evaluation does not necessarily reflects the visual quality of the result, we further design a user study to evaluate our methods. We only compare our methods with the manual interactive tool, because the naïve 2D method is clearly inferior to the other methods as shown in Section 6.3.

Data preparation We invited two subjects to manually create tracking shots using our manual tool for 20 real images. The first subject A is a photographer with rich experience on image editing, sketching and drawing in Photoshop. He is also familiar with depth and blur. The second subject B is a photography hobbyist with no background knowledge on 3D vision and no experience on painting and sketching. Before starting the user study, we gave each subject a training session of this interactive tool, and let them practice on multiple images until they felt comfortable. After training, the subjects were able to generate one tracking shot in the average time of 7 minutes, after watching the input video several times.

We also taught a third subject (a graduate student in the CS department) on how to use our system, and asked him to generate results of our methods. Besides object segmentation which is done interactively in After Effects, the only user input that was allowed in our approach is to adjust the amount of blur using a slide bar. Among the 20 results, 10 of them were created using the 3D method, the other 10 using the Pseudo 3D approach. Figure 16 shows two examples in the user study data.

Perceptual comparisons We invited another 30 viewers to evaluate the quality of the results generated by our system and by the manual tool. Before starting the evaluation, we gave each viewer a short video tutorial about tracking/panning shots, which also includes various successful and failure examples we found online. For each viewer and for each example, we randomly sampled one manually created shot, and put it side-by-side with our result for a pairwise comparison. We also show the original frame at the top as a reference. The viewer was asked to pick one result that looks more natural and visually appealing.

Table 3: *User study results. The numbers are the percentages of viewers who favored our results over those created by subjects.*

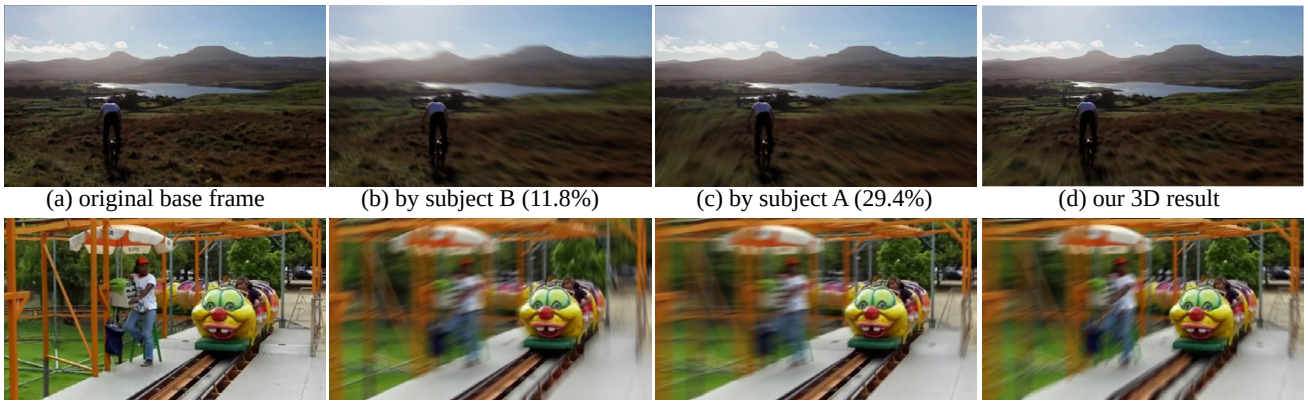
	subject A	subject B
3D method	61.8%	90.6%
Pseudo 3D method	67.7%	91.2%

The evaluation is summarized in Table 3. The majority of viewers favored our results over manually-created ones, especially those created by subject B ($\sim$ 90%). The ratio drops down to about $\sim$ 65% for subject A, which is as expected given he is more experienced on image editing. These results suggest that our system could be more helpful for people with limited 3D knowledge and drawing skills. Note that with more advanced tools, more practicing and more user time, the manually-created results could be potentially improved. Nevertheless the user study suggests that our system is already helpful for the vast majority of consumer-level users who cannot afford the effort and time for mastering the manual tools.

6.5 Discussion

Overall, none of these alternative approaches produce the same quality tracking shots as our system does, the visual artifacts in their results are obvious. The naïve 2D method does not work, because stabilizing the foreground is challenging, especially for highly dynamic objects. Furthermore, the optical flow on background is often inaccurate due to large displacement. For the manual tool, the two subjects provided important feedback to us. They uniformly felt that drawing 2D blur kernels to convey 3D information is non-intuitive and requires constant thinking. This task became harder when the foreground object is moving on a non-linear path. They also felt uncertain about how large the blur traces should be and how much variation they should have from region to region. All these comments suggest that simulating tracking shots without using dedicated tools like our system is in general hard for users at any skill level.

It is worth emphasizing that our system is capable of rendering tracking shots for deforming objects that are impossible to capture by a physical camera. If the target object is highly dynamic, such



(a) original base frame (b) by subject B (11.8%) (c) by subject A (29.4%) (d) our 3D result

(e) original base frame (f) by subject B (23.5%) (g) by subject A (47.1%) (h) our pseudo 3D result

Figure 16: *Two examples in the user study. The numbers in (b, c, f, g) are the percentages of viewers that favored these results over ours.*

Figure 17: *Two failure examples. Top: our method generates excessively large blur at the faraway sea region, due to the lack of features in this region. Bottom: the camera quickly rotates around the object, causing a failure of both 3D reconstruction and non-rigid SfM. Note that the tracking direction is inconsistant with object motion in this example. Please refer to the video for these two examples.*

as a running person (see Figure 9), then a long exposure time of a physical camera will lead to severe motion blur on the person's leg region no matter how well it tracks the person's body, which may not be desirable. In our system, since we process foreground and background regions separately, we can simulate a much shorter exposure time for the foreground alone, producing sharp foreground regions, which is similar to the bullet time effect as shown in many of our examples. This is clearly preferable in many cases. However, sometimes the user may also want to introduce some foreground blur, and not allowing it could be a disadvantage. Our system could potentially be extended to simulate additional foreground motion blur, by aligning and averaging the segmented foreground object in multiple frames around the reference one. We leave this additional step as future work.

7 Limitations

We find that a uniform spatial distribution of static feature points is crucial to generating successful results, given that the final dense blur kernels are interpolated from the sparse ones located at the feature points. A typical failure case is shown on the top of Figure 17. In this example, all the static feature points are on the nearby corals, and the faraway sea region on the top-right corner are not covered. Thus, the depth of the sea region cannot be estimated and the blur kernel interpolation will extend excessively large blur kernels from the nearby coral region to the faraway sea region. In general, any textureless region with large depth differences from its surrounding regions will suffer from this problem. As future work we plan to provide additional UI controls to allow the user to specify the amount of blur in a specific region, and use it as a hard constraint to produce correct results in this case.

Figure 18: *Failure cases. Frames contain large moving objects.*

Our system also has difficulty to deal with videos that contain large, quick camera rotation that centered at the target object, such as the bottom example in Figure 17. The quick camera rotation around the object directly leads to the failure of 3D scene reconstruction, also the failure of our non-rigid SfM. While the actress walks from right to left, the reconstructed 3D foreground motion trajectory is nearly a line that is perpendicular to the image plane, resulting in a final result that is inconsistent with the object motion. Both 3D and pseudo 3D methods fail on this example.

Our system may also fail in the case where the scene is crowded and the frames are occupied by large moving objects. Three such examples are shown in Figure 18. This is a challenging case for all previous video stabilization and 3D reconstruction methods.

8 Conclusion

We develop a system for generating tracking shots from a shot video clip captured by a handheld camera. We show that by combining advanced computer vision techniques such as segmentation and structure-from-motion, our system can obtain realistic, 3D-aware tracking shots. We further studied how to relax 3D reconstruction to 3D hallucination for improved robustness. Quantitative evaluation and user study results show that our system is capable of generating high quality tracking shots that are hard to achieve otherwise.

Acknowledgements

We thank the reviewers for their constructive comments and the user study participants for their time.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., S.DRUCKER, AND A.COLBURN. 2004. Interactive digital photomontage. *ACM Transactions on Graphics (TOG) 23*.

AVIDAN, S., AND SHASHUA, A. 2000. Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 4, 348–357.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video snapcut: Robust video object cutout using localized classifiers. *ACM Trans. Graph. 28*, 3.

BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., COHEN, M., CURLESS, B., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *Comput. Graph. Forum, Proc. EGSR*, 327–338.

BREGLER, C., HERTZMAN, A., AND BIERMANN, H. 2000. Recovering non-rigid 3d shape from image streams. In *CVPR*.

BROSTOW, G., AND ESSA, I. 2001. Image-based motion blur for stop motion animation. *ACM Transactions on Graphics (TOG)*.

CHAURASIA, G., DUCHENE, S., SORKINE-HORNUNG, O., AND DRETTAKIS, G. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG) 32*, 3, 30.

CHO, S., AND LEE, S. 2009. Fast motion deblurring. *ACM Trans. Graph. 28*, 5.

CHO, S., WANG, J., AND LEE, S. 2012. Vdeo deblurring for handheld cameras using patch-based synthesis. *ACM Trans. Graph.*.

FAUGERAS, O., LUONG, Q.-T., AND PAPADOPOULOU, T. 2001. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA.

FERGUS, R., SINGH, B., HERTZMANN, A., ROWEIS, S. T., AND FREEMAN, W. T. 2006. Removing camera shake from a single photograph. *ACM Trans. Graph. 25*, 3.

HARTLEY, R., AND ZISSERMAN, A. 2003. *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (TOG) 24*.

KAMINSKI, J. Y., AND TEICHER, M. 2004. A general framework for trajectory triangulation. *Journal of Mathematical Imaging and Vision 21*, 1-2, 27–41.

KANADE, T., AND OKUTOMI, M. 1994. A stereo matching algorithm with an adaptive window: Theory and experiment. *Pattern Analysis and Machine Intelligence 16*, 920–932.

KARPENKO, A., JACOBS, D., BAEK, J., AND LEVOY, M. 2011. Digital video stabilization and rolling shutter correction using gyroscopes. *Stanford University Computer Science Tech Report CSTR 2011-03*.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM Trans. Graph. 23*, 3, 689–694.

LIN, H.-Y., AND CHANG, C.-H. 2006. Photo-consistent motion blur modeling for realistic image synthesis. In *Proc. PSIVT*.

LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. In *ACM Transactions on Graphics (TOG)*, vol. 25, 646–653.

LIU, F., GLEICHER, M., JIN, H., AND AGARWALA, A. 2009. Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics (TOG) 28*.

LIU, F., GLEICHER, M., WANG, J., JIN, H., AND AGARWALA, A. 2011. Subspace video stabilization. *ACM Transactions on Graphics (TOG) 30*.

LIU, S., WANG, Y., YUAN, L., BU, J., TAN, P., AND SUN, J. 2012. Video stabilization with a depth camera. In *CVPR*.

LIU, S., YUAN, L., TAN, P., AND SUN, J. 2013. Bundled camera paths for video stabilization. *ACM Transactions on Graphics (TOG) 32*.

LIU, S., YUAN, L., TAN, P., AND SUN, J. 2014. Steadyflow: Spatially smooth optical flow for video stabilization. In *CVPR*.

LIU, C. 2009. Beyond pixels: Exploring new representations and applications for motion analysis. *Doctoral Thesis. Massachusetts Institute of Technology*.

LONGUET-HIGGINS, H. 1981. A computer algorithm for reconstructing a scene from two projections. *Nature 293*, 133–135.

MATSUSHITA, Y., OFEK, E., TANG, X., AND SHUM, H.-Y. 2005. Full-frame video stabilization. In *CVPR*.

OZDEN, K. E., CORNELIS, K., VAN EYCKEN, L., AND VAN GOOL, L. 2004. Reconstructing 3d trajectories of independently moving objects using generic constraints. *Computer Vision and Image Understanding (CVIU) 96*, 3, 453–471.

PARK, H., T. SHIRATORI, I. M., AND SHEIKH, Y. 2010. 3d reconstruction of a moving point from a series of 2d projections. In *ECCV*.

RÜEGG, J., WANG, O., SMOLIC, A., AND GROSS, M. H. 2013. DuctTake: Spatiotemporal video compositing. *Comput. Graph. Forum 32*, 2, 51–61.

SUNG, K., PEARCE, A., AND WANG, C. 2002. Spatial-temporal antialiasing. *IEEE Trans. Visualization and Computer Graphics 8*, 2, 144–153.

SUNKAVALLI, K., JOSHI, N., KANG, S. B., COHEN, M. F., AND PFISTER, H. 2012. Video snapshots: Creating high-quality images from video clips. *IEEE Transactions on Visualization and Computer Graphics 18*, 1868–1879.

TOMASI, C., AND KANADE, T. 1992. Shape and motion from image streams under orthography: A factorization method. *Int. J. Comput. Vision 9*, 2, 137–154.

TRESADERN, P., AND REID, I. 2005. Articulated structure from motion by factorization. In *CVPR*.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc. 13*, 4, 600–612.

YAN, J., AND POLLEFEYS, M. 2005. A factorization-based approach to articulated motion recovery. In *CVPR*, 815–821.