Slippage-free Background Replacement for Hand-held Video

Fan Zhong[1] Song Yang[1] Xueying Qin[1*] Dani Lischinski[2] Daniel Cohen-Or[3] Baoquan Chen[1]

[1]*Shandong University* [2]*The Hebrew University of Jerusalem* [3]*Tel Aviv University*

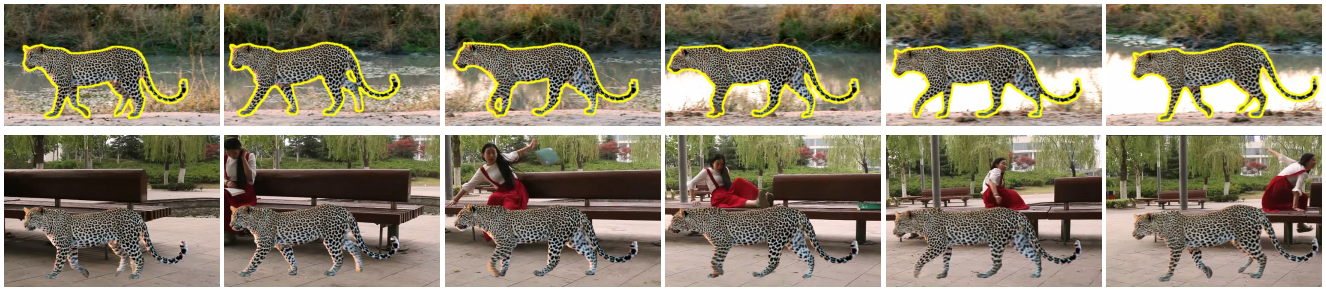


Figure 1: *Given a video with a* pre-segmented *foreground subject, our method can replace the original background with a background from another video, even when both videos are captured using a hand-held camera with different motion. One of our main goals is to avoid the* slippage *artifact, which cannot be appreciated with the still frames shown in this and other figures. Please view the companion video!*

Abstract

We introduce a method for replacing the background in a video of a moving foreground subject, when both the *source video* capturing the subject, and the *target video* capturing the new background scene, are natural videos, casually captured using a freely moving hand-held camera. We assume that the foreground subject has already been extracted, and focus on the challenging task of generating a video with a new background, such that the new background motion appears compatible with the original one. Failure to match the motion results in disturbing *slippage* or *moonwalk* artifacts, where the subject's feet appear to slide or slip over the ground. While matching the motion across the entire frame is impossible for scenes with differing geometry, we aim to match the *local motion* of the ground in the vicinity of the subject. This is achieved by reordering and warping the available target background frames in a manner that optimizes a suitably designed objective function.

CR Categories: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion,Tracking;

Keywords: background replacement, compositing, homography, motion analysis, slippage

Links: DL PDF WEB

1 Introduction

Replacing the background in a video of a performing subject is frequently used as a real-time effect in television news and weather broadcasts, and for special effects in the motion picture industry. In these settings, the performance is usually captured in a studio, typically with controlled camera motion and in front of a green screen. With the increasing abundance of digital video content in personal collections and on the world-wide-web, the goal of this work is to provide a simple, yet effective tool for background replacement for the challenging case of casually captured hand-held videos.

Convincing background replacement involves overcoming several non-trivial challenges. First, the performance should be extracted from its original background, along with an alpha matte for each frame. Second, the camera motion used to capture the performance must match the apparent camera motion after compositing the subject with its new background. Finally, the lighting of the subject and the new background must be made consistent.

While there has been extensive research on foreground extraction and matting (e.g, [Chuang et al. 2002; Agarwala et al. 2004; Bai et al. 2009]), the issue of matching the camera motion between the original and the new background has not received nearly as much attention. Failing to match the camera motion may yield various artifacts that severely undermine the realism of the composited result. Perhaps the most disturbing artifact, for performances involving locomotion, is the impression that the subject is sliding or slipping across the ground. Henceforth, we refer to this artifact as *slippage*. Significantly incompatible motion of the new background, e.g., due to strong changes in zoom, may yield unnatural results, even if the point of contact between the subject and the ground is not visible.

In this work, we introduce a method for *motion-compatible* background replacement, where both the *source video* of the moving subject, and the *target video* of the new background scene, are natural videos, captured using a freely moving hand-held camera. In the absence of 3D models, and due to differences between the scene geometries and the camera motions, it is impossible to perfectly match the source background motion by re-projecting the target frames. In fact, no simple model is able to adequately describe the overall motion of either background. However, by assuming that the ground around the foreground subject is *locally* planar, we are able to use a homography to model the *local motion* in that portion of the frame. Thus, the gist of our approach is to temporally reorder and spatially warp the target background frames, so as to match as closely as possible the local motion in the vicinity of the contact between the foreground subject and the ground. We refer to this process as *dynamic space-time warping*.

We begin by tracking the estimated contact point between the subject and the source background. As stated earlier, the motion of the background near the contact point is modeled using a homography between each pair of successive frames. Next, we precompute the homographies between pairs of sufficiently similar (but not necessarily successive) frames in the target video, and produce a similarity graph that represents the geometric relationships among target frames. We show that using dynamic programming, we can find a path in this graph, along with a suitable warp function for each node (frame) along the path, such that the resulting sequence of warped background frames appears to match the local motion recovered from the source video.

Note that we do not describe an end-to-end solution for the entire background replacement process. We address neither the foreground extraction task, nor the relighting of the subject, and focus only on motion compatibility, a challenging aspect of background replacement, which has not been addressed by previous work.

Our approach has some limitations: (i) the source video should show the subject on a roughly planar ground region without other dynamic objects in the vicinity of the ground contact point; (ii) dynamic objects in the target background are allowed, but may limit the ability to match the camera motion; (iii) the local geometry of the ground in the target background should be similar to that in the source video (i.e., we cannot composite a person climbing stairs onto a planar background). However, we do not require the overall geometry of the scene or the camera motion to be well coordinated with that of the source video. To our knowledge, ours is the first method capable of handling such a scenario.

2 Related Work

Video Compositing is routinely used in movie production for combining visual elements from a variety of different sources into a single video stream. The main challenge in this setting is the extraction of the foreground along with an alpha matte, while the alignment of the camera motion is achieved using a motion controller, which is expensive and not widely available. In a studio, foreground extraction may be accomplished using a constant color screen [Smith and Blinn 1996]. Although there has been much work on natural video matting and segmentation [Chuang et al. 2002; Agarwala et al. 2004; Wang et al. 2005; Bai et al. 2009], it remains a challenging problem. In this work we use existing user-assisted tools to extract the foreground from a natural source video, and focus only on the alignment of camera motion between the source and the target videos.

Video Textures: Several researchers describe methods for synthesizing larger or longer videos from an input video. Such methods have been used for synthesis of video sprites [Schödl et al. 2000], fire [Zhang et al. 2011], fluid [Okabe et al. 2011], and panoramic videos [Agarwala et al. 2005]. Human performances can also be synthesized from a pre-captured human video database [Flagg et al. 2009], with user-specified character and viewpoint motions [Xu et al. 2011]. While the latter works focus on controlling the foreground subject, while keeping the background unchanged, our goal is to keep the performance unchanged, while completely replacing the background. The central task in video texture synthesis is to search for smooth transitions in the foreground sequence, while in this work the challenge is to ensure that the new background's motion matches that of the original background.

Video Stabilization methods effectively impose a new, smoother, camera motion on a video sequence, and are therefore somewhat related. 2D video stabilization methods operate by estimating and smoothing the 2D transformations between consecutive frames in the sequence [Grundmann et al. 2011; Liu et al. 2013]. 3D methods use structure from motion to recover the 3D structure of the scene and the camera parameters [Liu et al. 2009], which is difficult. More recent methods achieve 3D path smoothing implicitly using 2D feature trajectories [Liu et al. 2011; Goldstein and Fattal 2012], requiring moderately long trajectories for good results.

Our method is closer in spirit to the 2D methods, since we also only estimate 2D transformations between consecutive frames. However, in contrast to the video stabilization setting, our source and target videos may differ significantly in their camera motions, frame rates, and scene geometry, introducing additional challenges.

Video-Based Rendering methods typically represent a dynamic scene as a collection of video streams with correspondences among them, allowing to re-render the performance from novel views (e.g., [Stich et al. 2008; Germann et al. 2012]). In our case, the source and the target videos are captured in different scenes, and our goal is to replay the performance from the same (moving) point of view, but using a different background. Thus, our task is to re-render the target sequence approximating the camera motion of the source sequence without the benefit (or the cost) of 3D reconstruction.

Video Matching: Sand and Teller [2004] explore spatio-temporal alignment of two videos of the same scene, captured using spatially similar, though not identical, motions. In contrast, in this work the scenes and camera motions may differ significantly.

3 Overview

The input to our method consists of a *source* video I_t of a performing subject, and a replacement (*target*) background, captured by a collection of frames B'_i. We assume that each source frame I_t has already been segmented into a foreground region F_t, capturing the performance of interest, and the source background B_t, which we would like to replace. The target background frames B'_i may belong to one or more video sequences, whose lengths and camera motions typically differ from those of the source video. Given the above inputs, our goal is to generate a convincing composite of the performance F_t over the target background. This entails synthesizing a new background video $\hat{B}_t$, whose motion with respect to the camera closely approximates the apparent motion of the source background B_t. In other words, denoting by M_t the motion from B_t to B_{t+1}, and by $\hat{M}_t$ the motion from $\hat{B}_t$ to $\hat{B}_{t+1}$, our goal is to ensure that M_t and $\hat{M}_t$ are as close as possible, for every t. We refer to this process as motion-compatible video synthesis.

3.1 Challenges and Requirements

Motion-compatible synthesis is a highly challenging task. In our scenario, the source and target sequences come from a hand-held camera, with different camera motions and background scene geometries. Thus, in general, it is impossible to reproduce the source motion M_t by using the target frames B'_i as they are, or by merely reshuffling their order. Warping the target frames may be used in order to better match the source motion, but any such warps must be applied sparingly in order to avoid visible artifacts. Specifically, we seek a solution that satisfies the following properties:

Slippage-free: For foreground subjects in contact with the background, the relative motion between the subject and the background must be matched by the new background $\hat{B}_t$. Violation of this requirement might result in the impression that the subject is slipping across the ground.

Minimal distortion: Transformations applied to the target frames B'_i should introduce minimal geometric distortion.

Maximal coverage: Warping the target frames to meet the above two requirements might result in only a partial coverage of the frame area. The missing portions can be filled using image completion or by stitching with nearby frames, but if the missing areas are large, this can result in visible artifacts.

3.2 Straighforward Approaches

Before describing our approach, let us first briefly discuss two more straightforward alternatives for motion-compatible video synthesis.

Using panoramas: One might consider stitching the target background video frames into a large panoramic mosaic (cf. [Steedly et al. 2005]). Each of the new frames $\hat{B}_t$ may then be obtained by applying the appropriate transformation T_t to the panorama and clipping it to the frame window. Assuming that the initial transformation T_0 is provided by the user, T_t can be obtained by concatenating the motion transformations between the source frames: $T_t = M_t M_{t-1} \cdots M_1 T_0$. A major limitation of this approach stems from the panoramic stitching, which is known to be difficult in many cases, e.g., large parallax, zooming, etc.

Background propagation: Instead of precomputing a panorama, one might attempt to transform and stitch the background on demand. Each frame $\hat{B}_{t+1}$ can be obtained by transforming $\hat{B}_t$ with M_t, and then filling any uncovered regions by stitching neighboring target frames. However, note that a large portion of each resulting frame consists of transformed parts of previous frames. This introduces two problems: First, $\hat{B}_t$ becomes progressively less similar to the original source frames B'_i, making feature matching and stitching more difficult; Second, frames become increasingly blurry due to repeated resampling, which decreases the quality and further increases the difficulty of matching and stitching.

The above approaches address only slippage and coverage. Distortion could be reduced by making $\hat{M}_t$ consist mainly of 2D translation and limited uniform scaling. However, restricting the motion model in this manner might not approximate well the original background motion, thereby re-introducing slippage artifacts. Also, the local nature of the background propagation approach may quickly lead to a dead-end after running out of target background images. In addition, because the resulting background images are essentially clipped from a panorama, no parallax effects can be produced, and any dynamics in the target background video are lost.

3.3 Our Approach

The basic idea in our approach is to obtain the new background frames $\hat{B}_t$ by selecting a subset of the target video frames, $\{\hat{B}'_t\} \subset \{B'_i\}$, and applying a warp W_t to each frame (see Figure 2). In order to determine the frames $\hat{B}'_t$ and their corresponding warps W_t, we use global optimization that *simultaneously* considers slippage, distortion, and coverage. We refer to our method as Dynamic Space-Time Warping (DSTW), since we effectively warp the original target background video in both space and time, using dynamic programming.

We begin by tracking the approximate contact point between the subject in F_t and the original background B_t. The motion of the background M_t in the vicinity of the contact point is modeled by a homography.

Next, we organize the target background frames B'_i as a similarity graph, where two frames are connected if a smooth transition between them can be produced. For static scenes, this only requires that the transformation between neighboring frames is well approximated by a low-distortion homography (see Section 5.2). This

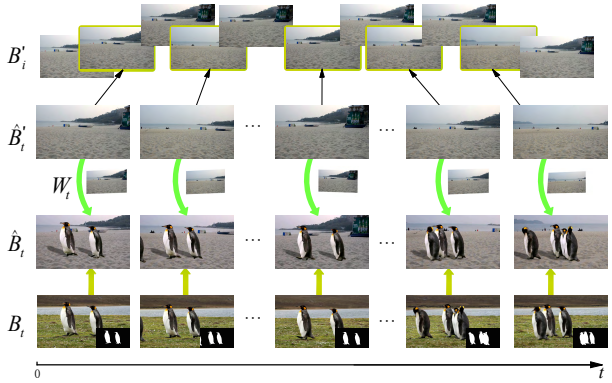


Figure 2: *Dynamic Space-Time Warping. For each source foreground frame B_t, a corresponding target background frame $\hat{B}'_t$ is chosen, along with a spatial warping function W_t. The new background frame $\hat{B}_t$ is produced by warping $\hat{B}'_t$ with W_t.*

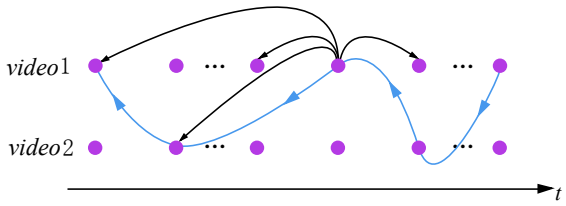


Figure 3: *The neighborhoods of source background frames, and the path of selected frames (blue line). Note that the path direction may be different from time axis, and may use multiple videos.*

means that the graph neighbors of each frame are not necessarily its temporal neighbors in the sequence; in fact, they could even belong to different target video sequences (of the same scene), as shown in Figure 3. Such remote connections are useful for overcoming the significant differences that might exist between the camera motions of the source and target sequences. For dynamic scenes, the original temporal order of target background frames must be preserved, and remote connections also should be disabled. This may be achieved if the target background video is captured a bit more carefully. Below we mainly focus on static scenes, and discuss the handling of dynamic scenes in Section 5.3.

Finally, we use dynamic programming to find an optimal path in this graph, along with a suitable warp function W_t for each node (frame $\hat{B}'_t$) along the path, such that the resulting sequence appears to match the camera motion in the source video. The objective function optimized in this process consists of terms designed to achieve the desired motion, while minimizing distortion and maximizing frame coverage. The next section describes these terms and presents the dynamic programming algorithm that we use.

4 Dynamic Space-Time Warping

Our goal is to find an optimal path $\hat{B}'_t$ in the similarity graph defined over the collection of target background frames B'_i, along with a suitable warp W_t for each frame in the path. Formally, we seek a sequence of configurations $\mathbf{\Phi} = \{\Phi_0, \Phi_1, \cdots, \Phi_{L-1}\}$, where $\Phi_t = (\hat{B}'_t, W_t)$ is the configuration of the t-th frame, and L is the length of the source foreground video. The sequence $\mathbf{\Phi}$ should

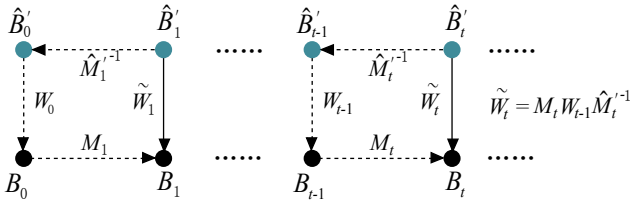


Figure 4: *Relationship of transformations between successive video frames.*

source frame 0 result frame 100 result frame 120

Figure 5: *Distortion caused by using homographies for warping.*

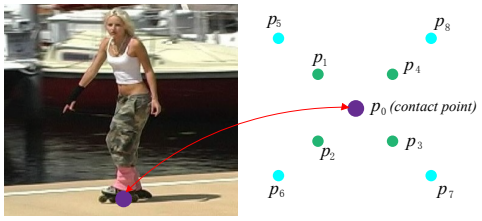


Figure 6: *The representative points around the contact point.*

minimize the objective function

$$E(\mathbf{\Phi}) = E^m(\mathbf{\Phi}) + E^d(\mathbf{\Phi}) + E^c(\mathbf{\Phi}), \quad \text{s.t. } W_0 = \dot{W}_0 \qquad (1)$$

where E^m, E^d, E^c are *motion*, *distortion* and *coverage* terms, respectively. E^m measures the motion consistency between $\hat{B}$ and B, which is crucial for avoiding slippage, as well as the smoothness of the resulting motion. E^d penalizes undesirable distortion of the target background, and E^c measures the coverage of the resulting video frames by the new background. $\dot{W}_0$ is a user specified initial warp that determines the initial relative position and scale of the subject in the first replacement background frame $\hat{B}'_0$. The first frame itself can be automatically determined by the optimization (as demonstrated in Figure 9), but it can also be specified by the user, if necessary, as described in Section 4.4.

The goal of the optimization is to ensure that the apparent motion of the new background is compatible with that in the source video. In general scenes, the motion of the background is too complex to capture with a simple motion model. Furthermore, since the geometries of the source and target background scenes can differ significantly, even if we had an exact motion model, it would not be reasonable to simply force the source motion onto the target background everywhere in the frame.

Therefore, our idea is to focus on matching the motion mostly in the vicinity of the contact point between the moving foreground elements and the background. Assuming that the ground is locally planar, we use a homography M_t to describe the *local* motion between source background frames B_{t-1} and B_t. The eight parameters of M_t are estimated using tracked feature points in the vicinity of the contact between F_t and B_t, as described in Section 5.1. We also use a homography M'_t to describe *local* motion between successive replacement candidates $\hat{B}'_{t-1}$ and $\hat{B}'_t$, estimated as described in Section 5.2. Note that $\hat{B}'_{t-1}$ and $\hat{B}'_t$ must be neighbors in the similarity graph.

The main constraint that provides the basis for our motion term and for the entire dynamic programming process, is the following relationship between a pair of successive warp functions W_{t-1} and W_t (also see Figure 4):

$$\tilde{W}_t = M_t W_{t-1} M'^{-1}_t. \qquad (2)$$

Thus, given the configuration $(\hat{B}'_{t-1}, W_{t-1})$ and a candidate for the next frame $\hat{B}'_t$, the corresponding warp W_t is given by eq. (2).

Removing distortion: The warp $\tilde{W}_t$ that is directly given by (2) is also a homography, and therefore may introduce perspective distortion. Since each warp depends on the previous one, these distortions might accumulate quickly. Figure 5 shows such an example. In order to reduce such distortions, we restrict our warps W_t to consist of only 2D translation, uniform scaling, and rotation.

Obviously, these more restricted warps can only approximate the homographies $\tilde{W}_t$. In order to reduce slippage artifacts due to this approximation, we first compute $\tilde{W}_t$ as a full homography using eq. (2), and then approximate it with a more restricted warping function by minimizing the difference between W_t and $\tilde{W}_t$ over a few points p_i positioned around the contact point in the source frame B_t (see Figure 6):

$$W_t = \arg\min \sum_i \omega_i \parallel p_i - W_t \tilde{W}_t^{-1} p_i \parallel^2 \qquad (3)$$

As shown in Figure 6, for each contact point, we select 9 points around it. p_0 is positioned at the contact point itself, in order to capture the translation of the contact point, which should be well fit in order to avoid slippage. p_1, p_2, p_3, p_4 are close to the contact point, and capture the local effect of $\tilde{W}_t$. p_5, p_6, p_7, p_8 are intended to capture the more global effect of $\tilde{W}_t$, and positioned farther from the contact point. The relative position of p_i with respect to the contact point is the same for all frames B_t. The weights ω_i balance the fitting of local and global effects, and are defined as $\omega_i = \frac{1}{1+\|p_i - p_0\|^\gamma}$, where γ controls the attenuation away from the contact point. We use $\gamma = 2$, while larger γ would decrease the weight of global effect fitting. Note that this particular layout of p_i is fairly arbitrary, and was simply chosen to ensure that local and global motions are assigned different weights.

The above spatially-variant fitting method effectively removes distortion, without re-introducing slippage, while at the same time accounting for overall image motion. Thus, it produces much better results than naive approaches that attempt to represent global motion using a restricted model (please see the supplementary video for a comparison with a 2D similarity transform).

Hybrid warp models: W_t can be chosen as any combination of 2D translation, uniform scaling and rotation. The choice might have a significant effect on the motion of the new background. Allowing translation, scaling, and rotation can better fit the original motion, but might lead to easily noticeable accumulation of rotation error. Using only translation or a combination of translation and scaling may produce more visually pleasing results in most cases, but the resulting overall background motion may differ more. To address this issue, we use a mixture of warp models. For each frame, the proper warping model that minimizes eq. (1) can be automatically determined. This is described in more detail in Section 4.4.

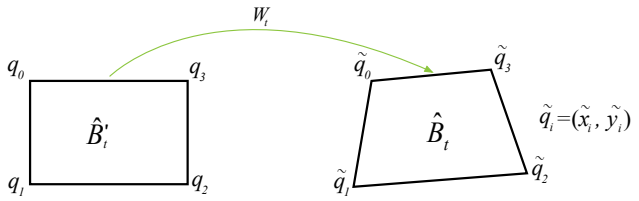


Figure 7: *Warping a selected target background image.*

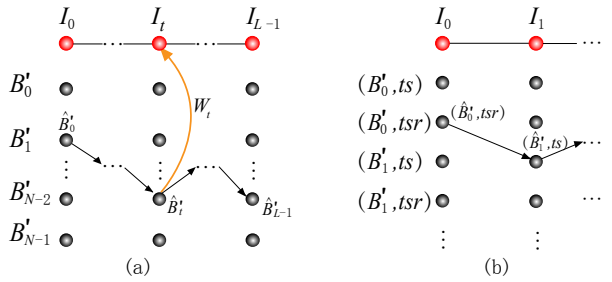


Figure 8: *(a) Solving for the optimal sequence by dynamic programming. (b) Incorporating a hybrid motion model into the optimization (ts or tsr in this figure).*

4.1 The Motion Term

The motion term measures how well W_t satisfies the constraint given by eq. (2) by computing the residual error of eq. (3):

$$err(W_t) = \left\langle \omega_i \parallel p_i - W_t \tilde{W}_t^{-1} p_i \parallel^2 \right\rangle, \quad (4)$$

where $\langle . \rangle$ is the expectation operator. Besides minimizing the above error, we also require the resulting background motion to be smooth, so the entire motion term is defined as follows:

$$E^m(\mathbf{\Phi}) = \omega^m \sum_t err(W_t) + \omega^s \sum_{t>0} \left\langle \parallel W_{t-1} q_i - W_t q_i \parallel^2 \right\rangle, \quad (5)$$

where q_i are the four image corners of the source background images, which are assumed to have a fixed size (see Figure 7).

The added smoothness constraint is important for a more stable background motion. Note that the motion term considers mainly the local motion around the contact point, and there is no special constraint for the motion of the more distant image regions, because the geometry of source and target scenes might be quite different. Thus, the smoothness constraint serves as a regularization term for these regions.

4.2 The Distortion Term

Undesirable distortion of the warped background usually arises due to perspective effects and large rotations, while the mismatch of translation and scale is less noticeable. Therefore, our distortion term is designed mainly to penalize perspective and rotation:

$$E^d(\mathbf{\Phi}) = \omega^d \sum_t |\tilde{x}_0 - \tilde{x}_1| + |\tilde{x}_2 - \tilde{x}_3| + |\tilde{y}_0 - \tilde{y}_3| + |\tilde{y}_1 - \tilde{y}_2| \quad (6)$$

Here $\tilde{x}_i, \tilde{y}_i, i = 0, 1, 2, 3$ are the $\tilde{x}, \tilde{y}$ coordinates of the four warped background image corners $\tilde{q}_i$ (see Figure 7), so the above equation penalizes any transformation that results in a non-rectangular or rotated image region. Although there exist more sophisticated methods to estimate the anisotropic scaling of a homography warping [Hartley and Zisserman 2004], the above equation is simple and was found to work well. Note that the coordinate differences above are not normalized, so the strength of the distortion term is dependent on the region's size. In theory, this could result in a tendency to reduce the scale of the warped image; in practice, however, excessive down-scaling is prevented by our use of the coverage term. On the other hand, this term does penalize increase in scale, which is desirable in order to maintain a reasonable relative scale between the foreground and the background.

4.3 The Coverage Term

Our final requirement is that the resulting new background frames cover as much of the area of the original frames as possible, minimizing any remaining blank regions. This issue has also been discussed in previous video stabilization works. Gleicher and Liu [2008] avoid blank regions by applying a proper scaling to the warped frame. Liu *et al.* [2013] consider the area of blank regions in their optimization process, and remove remaining blank regions by cropping.

To penalize large blank regions we define the coverage term as the total area of blank regions:

$$E^c(\mathbf{\Phi}) = \omega^c \sum_t \frac{area(I_t \ominus W_t * \hat{B}'_t)}{area(I_t)} \quad (7)$$

where $\ominus$ denotes the set difference, and $*$ is the image warping operator. Note that the coverage term could encourage increase in the scale of the target frame. However, in practice, we find that our method does not result in unnecessary upscaling. This is because, first, the coverage term becomes zero when there is no blank region, making upscaling unnecessary; second, W_t is restricted by the motion term (2), and thus is not arbitrary; and third, the distortion term penalizes large scales, as explained earlier.

Local stitching: in order to facilitate reduction of blank regions, we can also expand each target background frame B'_i by stitching it with its neighboring frames in the similarity graph. Because image stitching is difficult in some cases, it is performed only when two images can be well aligned by a homography. Other more sophisticated stitching methods, such as local warping or seam finding, may align images better, but may also cause temporal inconsistencies in the resulting video.

Note that the target background frames may no longer be quadrilateral after stitching. We represent the region of each expanded frame as a general polygon. Since it is not known in advance which frames will be used in the final result, we compute only the polygonal outline of each stitched frame during pre-processing. The actual stitching is performed only when necessary for the final output.

4.4 Dynamic Programming Optimization

We solve for the optimal sequence of configurations

$$\mathbf{\Phi} = \{\hat{B}'_0, (\hat{B}'_1, W_1), \cdots, (\hat{B}'_{L-1}, W_{L-1})\}$$

by minimizing eq. (1), under the constraints imposed by eq. (2) and eq. (3). This may be done efficiently using dynamic programming (DP), as summarized in the pseudocode in Table 1.

As illustrated in Figure 8(a), for each source video frame I_t, there are N candidates for $\hat{B}'_t$. For each possible choice of $\hat{B}'_t$, the associated warp function W_t is determined from a previous configuration $(\hat{B}'_{t-1}, W_{t-1})$ using equations (2) and (3). Rather than considering all N possible previous frames for every choice of $\hat{B}'_t$, we only

Define $\mathbf{A}[L][N]$, with $\mathbf{A}[\,t\,][\,j\,] = \{cost, parent, warp\}$

Set $\mathbf{A}[\,0\,][\,j\,] = \{0,\ -1,\ W_0\}$ for $0 \le j \le N-1$
for every source frame $B_t,\ t = 1 : L-1$
  for every target frame $B'_j,\ j = 0 : N-1$
    Set $\hat{B}'_t = B'_j$ and $\mathbf{A}[\,t\,][\,j\,] = \{max, -1,\ identity\}$;
    for each $B'_k \in$ neighbors(B'_j)
      Set $\hat{B}'_{t-1} = B'_k$ and $W_{t-1} = \mathbf{A}[t-1][k].warp$;
      Compute W_t using equations (2) and (3);
      Compute $E(\Phi_t)$ as the increment of $E(\mathbf{\Phi})$ by the appending of $\hat{B}'_t$;
      Set $cost = \mathbf{A}[t-1][k].cost + E(\Phi_t)$;
      If $cost < \mathbf{A}[t][j].cost$, then set $\mathbf{A}[t][j] = \{cost, k, W_t\}$;
    end for
  end for
end for

Set $k = \arg\min_k \{\mathbf{A}[L-1][k].cost \mid 0 \le k \le N-1\}$
for $t = L-1 : 0$
  Output $\hat{B}'_t = B'_k,\ W_t = \mathbf{A}[t][k].warp$
  Set $k = \mathbf{A}[t][k].parent$;
end for

Table 1: *Pseudo-code for our DP-based optimization.*

consider neighboring frames in the similarity graph. Thus, for each candidate B'_j for $\hat{B}'_t$ we can efficiently compute and store the lowest cost sequence of $t+1$ configurations ending with B'_j. Having reached $t = L-1$ we can trace back and retrieve the lowest cost sequence of length L.

In order to initialize this process we only need the initial warp function $\dot{W}_0$. This warp function is determined from the user-provided initial scale and position of the target background with respect to the first foreground frame F_0, so $\dot{W}_0$ is represented as three parameters $[scale, d_x, d_y]$. In our implementation we use the default setting of $[1.2, 0, 0]$. This setting was used in most of our results (see Section 6 for details). In order to deal with different size of source and target frames, before applying $\dot{W}_0$ we first scale the target frame to be the same size as the source and align the frame centers. Note that changing W_0 affects both W_t and $\hat{B}'_t$; the effect of W_0 on the first frame $\hat{B}'_0$ is demonstrated in Figure 9. Thus, the ability to specify $\dot{W}_0$ is an important means of user control, because the proper initial scale and position cannot be inferred from the video data without geometry and semantics.

Incorporating hybrid warping: Our DP-based approach is easily extended to support hybrid warping, allowing warp functions W_t of different types: we support translation+scale (ts), and translation+scale+rotation (tsr) warps. Figure 8(b) illustrates this extension: the candidate set of frames is simply multiplied by the number of supported models, and thus choosing a specific candidate determines both frame and its warp model. Generally, increasing ω_d and ω_s will result in more frequent usage of ts, while increasing ω_m results in more usage of tsr. Figure 10 shows the effect of using a hybrid warp model. Note that although excessive rotation can also be removed by increasing ω_d, in practice, the proper value of ω_d is difficult to determine without using the hybrid model.

Adding user constraints: The algorithm in Table 1 automatically determines the initial selected target frame $\hat{B}'_0$, which provides the most flexibility for the optimization. However, sometimes the user might want to explicitly specify $\hat{B}'_0$ in order to impose a specific starting frame, or more generally, indicate a range of frames from which $\hat{B}'_0$ should be chosen. Such a constraint may also be

source frame | $W_0 = [1.2, 0, 0]$, $\hat{B}'_0 = 195$ | $W_0 = [1.2, 80, -80]$, $\hat{B}'_0 = 169$

Figure 9: *Effect of W_0 ($= [scale, d_x, d_y]$) on the selected target correspondence $\hat{B}'_0$.*

$tsr, \omega_d = 0.1$ | $tsr, \omega_d = 10$ | $ts + tsr, \omega_d = 0.1$

Figure 10: *Effect of using the hybrid warp model. The shown image is frame 260 of BASKETBALL. Using only tsr warping results in excessive rotation, even when ω^d is set to be very large.*

used to avoid improper initial position of the subject (e.g., Figure 18 right). To incorporate such constraints, we set the initial cost ($\mathbf{A}[0][j].cost$) to a large number for any frame B'_j outside the desired range. A similar range constraint may be also specified at any other time t, but we have not used such constraints in our results.

5 Pre-processing and Implementation

Our system processes input videos by first performing a fully automatic pre-processing, followed by the optimization described in Section 4. The pre-processing is time-consuming, but it needs only to be executed once. Once the pre-processing is complete, the optimization takes only a few seconds (see Table 3).

5.1 Pre-processing of Source Video

We pre-process the source sequence I_t in order to locate the contact points between the moving subjects and the ground, and to compute the motion M_t between successive frames. As stated earlier, M_t is a homography that captures the local motion of the background around the contact point(s). If there are multiple subjects, we assume that they are roughly on the same plane, so that their local motions can be represented by the same homography.

To estimate M_t, we first detect a set of KLT features in B_{t-1} (foreground regions are excluded), and then filter these features to find out those that are most likely to be on the ground. This is achieved by first selecting K_0 features nearest to the contact points, where K_0 is set to be small, to ensure that the selected features are on the ground. Other features are sorted according to their appearance difference (SSD over a 7×7 window centered at each feature) to the K_0 seed features, and the most similar K_1 features are then selected. The $K_0 + K_1$ features form our final selected feature set. We then run a KLT tracker to obtain their corresponding locations in the next frame, and compute a homography transformation M_t using RANSAC. In our implementation we set $K_0 = 10, K_1 = 90$, and for the RANSAC procedure we use the OpenCV function `findHomography` with default parameters.

Given the alpha matte of each foreground subject, the contact points are easy to estimate. Regardless of whether there is a single contact point, or an entire contact area, we simply compute and use the

average coordinates of the bottom 10 rows of foreground pixels. This is applicable also for cases where the actual contact point is not visible in the frame, such as BOY in Figure 11. The resulting positions are then smoothed temporally using a time window size of 15. Note that these positions only serve as the centers of the areas around which we estimate the local motion. Thus, small deviations between these positions and the actual contact points do not cause any problems in practice.

5.2 Pre-processing of Target Background Frames

The collection of target background frames is pre-processed to determine the neighbors of each frame, as well as to compute the transformations between neighboring frames. Note that for static scenes, the neighbor of a frame B'_i is not necessarily temporally adjacent to it, but may be any other frame B'_j, provided that a smooth transition between the two frames may be generated.

To determine the neighbors of each frame, firstly the temporal neighbors in the range $t-10$ and $t+10$ are selected as neighbor candidates; and then remote neighbor candidates are searched by SIFT feature matching. For each pair of frames B'_i and B'_j, SIFT detection and matching is first performed, and then the homography between them is estimated with RANSAC. We then compute the residual error r_{ij} and the distortion d_{ij} of the homography. d_{ij} is computed using eq. (6), without scaling by ω^d, and normalized by the length of the image diagonal. B'_i and B'_j are defined as candidates to become neighbors in a similarity graph if both r_{ij} and d_{ij} are less than specified thresholds ($r_{ij} < 5$, $d_{ij} < 0.3$). Hierarchical search is used for acceleration by first considering every 16th frame, and then checking every frame only in the ranges that were identified during the first pass.

The next step is to further refine the candidates and compute local transformations. For each candidate pair of neighbors B'_i and B'_j, we first align them with the homography computed in the previous step, and then invoke a KLT tracker to find a set of matching features. KLT usually produces more matched features than SIFT, especially in regions without strong features. Using the matched KLT features, we re-compute the homography between B'_i and B'_j, as well as the residual error and the distortion score. A candidate is eliminated if it does not pass the threshold test ($r_{ij} < 3$, $d_{ij} < 0.2$) — in this step we use smaller thresholds in order to meet the requirement of smooth transition. The final similarity graph usually contains 15-25 neighbors for each frame.

Unlike in the case of the foreground video, here we do not yet have the position of the contact point for computing local motion. One solution is to store the features computed and matched in the preprocessing stage, and then estimate the motion M'_t in the area where it is needed during the optimization process. However, this makes the dynamic programming considerably slower. In our implementation, we pre-compute a grid of local homography transformations with the method of Liu *et al.* [2013], and then retrieve the appropriate local homography from the grid cell containing the contact points. In our experiments we found that a coarse 5×5 grid suffices for all of our examples.

5.3 Handling Dynamic Scenes

Our technique, as described so far, assumes that the target background is static. The optimization then has more flexibility in matching the source camera motion by considering not only successive target video frames as neighbors in the graph, but also allowing backward and remote jumps. However, if coherently moving objects are present in the target background, backward or remote jumps might introduce visible discontinuities into their motion. Also, selecting the same target frames several times in a row will cause dynamic background object to become still for a few frames.

Thus, for dynamic target backgrounds several temporal constraints are enforced. First, any backward or remote neighbors are removed from the graph. Specifically, for each target frame B'_t we restrict its neighbor candidates to the frames from B'_t to B'_{t+3}. Second, we slightly modify the algorithm in Table 1 to prevent it from selecting the same frame more than twice in a row. Note that the above constraints are only necessary for large coherent motions. For example, for periodic motions like sea waves, backward neighbors can also be used; and for small stochastic motions like tree leaves, both backward and remote neighbors can be allowed.

5.4 Additional Implementation Details

Pre-segmentation: In this paper we do not address the foreground extraction task. This task is outside the scope of this work, and may be carried out using a number of existing interactive tools, such as Rotobrush in Adobe After Effects [Bai et al. 2009]. The required interaction time to extract the foreground ranges from half an hour to several hours, depending on the user's skill and on the nature and length of the source video.

Shadow compositing: Shadows are important for a realistic composite. Since the direction of illumination in the target background is generally different from the source, there is no point in extracting the shadow along with the source foreground. Instead, in our implementation we use a simple method to generate a fake shadow. The shape of the shadow is approximated by applying a homography transformation to the foreground mask, which simulates the projection of the foreground mask onto the target ground plane. This trick is commonly used by Photoshop artists. The transformation is easy to specify interactively by dragging the corners of a rectangle around the foreground mask.

Illumination and auto-gain smoothing: Our method may use remote jumps between target background frames, which may introduce sudden illumination changes in the resulting video. One possible solution is to apply the method of Farbman *et al.* [2011] in order to stabilize tonal fluctuations in the target background video before using it. Our current implementation employs a simpler solution, which smoothes the mean intensity (L channel of Lab color space) of the frames in the resulting new background sequence using a temporal window of size 51. Let G_t denote the mean intensity of a frame, and G'_t denote the intensity after smoothing, then the intensity of each background pixel is shifted by $G'_t - G_t$. We found this simple solution to suppress illumination and auto-gain discontinuities in a satisfactory manner, as demonstrated in the companion video.

6 Results

In this section we present, discuss, and evaluate the results produced by our method on several test examples. Representative frames from six examples are shown in Figure 11. Each row shows the frames before and after background replacement. All of the videos include typical free camera motions, including panning, rotation, zooming, and combinations thereof. Most of the captured scenes have planar ground, but differ from each other in the surrounding geometry. The cumulative motion difference between the original and new backgrounds is visualized using superimposed triangle pairs, as explained in Section 6.1. Obviously, the static nature of the figures in this paper makes it impossible to visually demonstrate the effectiveness of our results, and we refer the reader to the companion video, which includes all of them.

Figure 11: *Results, from top to bottom: BOY, GIRL, WALK, SKATER, GOAT, COUPLE. Left: source frames (frame 10, 100, and 200). Right: The corresponding frames with new background. The green and blue triangles are the transforming triangles of the source and the result, respectively. The source videos of BOY, GIRL, SKATER, GOAT were obtained from* pond5.

Table 2: *Resulting matching costs ($\times 10^3$) for our cross test. Each source video (row) is tested with all 6 target videos (columns) in Figure 11 to obtain the optimal matching cost for each. All tests are performed with the default setting of initial warp W_0 and weighting parameters, without using dynamic scene constraints.*

	BOY	GIRL	WALK	SKATER	GOAT	COUPLE
BOY*	**3.76**	24.5	105	109	112	4.00
GIRL*	36.3	22.6	8586	7187	15472	**21.9**
WALK*	223	6759	75.5	**41.5**	155.4	213
SKATER*	78.2	207	117	**8.31**	1130	181
GOAT*	60.9	410	7226	1286	**18.9**	1405
COUPLE*	14.1	19.2	21.2	20.5	21.9	**9.97**

Several source videos were obtained from *pond5*[1], while others were captured by the authors using a hand-held camera. Videos of the target backgrounds were mainly captured by a non-author volunteer using a hand-held camera, after viewing the source videos several times; others were either downloaded from *pond5* (BOY and GIRL), or captured by a volunteer based only on a verbal description of the required motion (WALK and PENGUIN). For common camera motions like panning and zooming, motion-based video retrieval can be developed to facilitate the process of finding a suitable target video from a video stock website. Our method can support such retrieval by ranking according to the optimal cost computed by our method. To demonstrate this we ran a cross test using the examples shown in Figure 11. Each of our 6 source videos was tested against each of the 6 target videos, with the resulting matching costs are listed in Table 2. For GIRL and WALK, our selected target videos are not the ones that received the best score; however, they received the second best score, and are very similar in motion to the best matching ones (please see the video).

In both the BOY and GIRL examples, the source camera moves backward, while the target camera moves forward. Since our method reorders the target frames as needed to best fit the source motion, this poses no problem. In the BOY example, the source background has the ground plane extending to the horizon, while in the target scene the background has mostly vertical geometry (trees) surrounding the narrow road. Since our method extracts and matches only local motion, the result is not affected by these differences in geometry, and maintains realism.

In the SKATER example, the source camera is tracking the skating girl, while zooming in and out. This complex camera motion is difficult to match using only a single target video with a simpler motion. In such cases, the ability to capture multiple videos of the target background becomes helpful. Three target video clips were captured, and provided as input to our method, which selected target frames from all three clips, and the selected target frames are dramatically re-ordered so as to best match the source motion (as seen in Figure 12).

Although our method assumes that the geometry around the contact points is locally planar, it can still handle some mild violations of this assumption. This is demonstrated in the GOAT example, where the ground is not truly planar.

In the COUPLE example, both the source and the target background scenes contain significant dynamic activities. Because of the feature filtering introduced in Section 5.1 and the use of RANSAC,

[1] A stock footage website: http://www.pond5.com

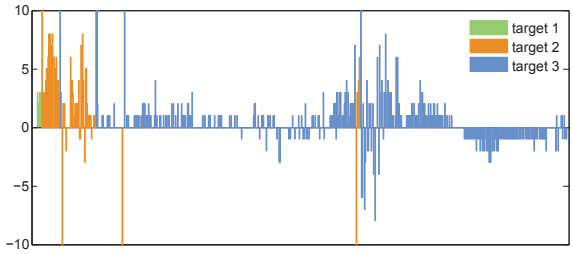
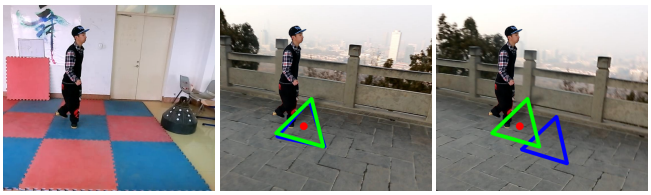


Figure 12: *Visualization of selected target videos and frame jumps for frames of SKATER. The line height denotes target frame jumps between adjacent source frames (negative for backward).*



source frame 90 local motion global motion

Figure 13: *Comparison of motion inconsistency using local and global motion matching.*

our method is not sensitive to dynamic objects in the source background. Dynamics in the target background are preserved using the method described in Section 5.3, so long as the source and target camera motions do not differ too much in their speed. Please see the video for additional examples of dynamic backgrounds.

We found that in some cases applying video stabilization as a post-processing step (using Adobe AfterEffects Warp Stabilizer) improved the quality of the result. One such example is WALK-SEASIDE in the companion video. However, we found that stabilization can also sometimes produce overly cropped results. The results shown in the paper and video are all without stabilization.

6.1 Evaluation

Motion inconsistencies between source and result videos can be visualized and quantified by the differences between point trajectories, as they are transformed with the motion. Starting with an equilateral triangle centered around the contact point in the first frame, at each subsequent frame we transform the triangle by applying the local motion model, estimated near the contact point. By doing this for both the source video and the result, we can assess how well the local source motion is matched. Ideal matching would keep the source and result triangles aligned, while accumulation of matching error causes the two triangles to drift apart and/or deform or rotate differently from each other.

Figure 11 shows a pair of triangles transformed in this manner over each result frame. The green triangle corresponds to the source motion, while the blue to the result motion, with the contact point shown as a red dot. Note that in some examples the triangles eventually become small due to the retreating motion of the camera in these examples. It may be observed that some differences between the source and result triangles arise with time, but overall the triangles remain similar to each other.

Figure 13 uses the same visualization to compare the results produced by our method using local vs. global motion models. For a global motion model we use an affine transformation, estimated using the OpenCV function `estimateGlobalMotionRobust`. It may be seen that with global motion after 90 frames there is a much bigger discrepancy between the motion of the result and that of the source.

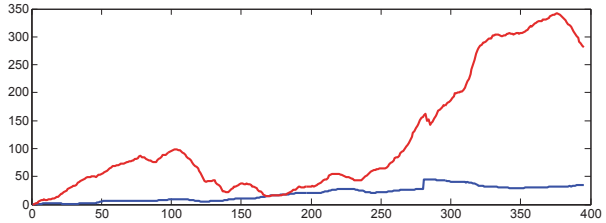


Figure 14: *Motion inconsistency (shape difference of transforming triangles) when using our spatially-variant fitting (blue) vs. naive fitting (red, see text).*

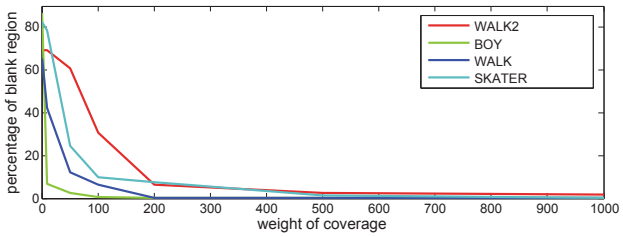


Figure 15: *Effect of the coverage term weight.*

The approximation of the homography in eq. (3) is designed to balance between eliminating slippage and avoiding distortion. Replacing our spatially-variant fitting scheme with a more naive scheme, which does not consider the contact point, is not able to achieve our goals. Figure 14 shows this by plotting the *shape difference* (sum of the squared distances between corresponding vertices) of the transforming triangles resulting from our fitting scheme vs. a naive similarity fitting that uses the four image corners.

Our coverage term is evaluated in Figure 15, which shows the relative amount of blank regions using different weights of coverage term ω_c for serval examples. By increasing ω_c, the selected source background frames $\hat{B}'_t$ will be adjusted to reduce the blank region; on the other hand it also may result in a larger scale factor to the source background frames, which may result in incorrect relative size of foreground and background. The use of local background stitching is helpful in this case. Note that the stitched background region is computed during pre-processing, and is accounted for in the optimization process, which is more effective than only attempting stitching during post-processing.

Our hybrid warp model is effective for adaptive handling of different camera motions. By using both ts and tsr, small rotations are filtered, while significant rotations are still preserved under the constraint of the motion term. Figure 16 plots the distortion cost with and without the use of a hybrid warp model, showing that the hybrid model produces much less distortion with the same parameters.

The default values for the weights of the four energy terms are $\omega^m = 10, \omega^s = 0.1, \omega^d = 0.1, \omega^c = 1000$. ω^m and ω^c use these default values in all our examples. ω^d is increased in the case of excessive rotation (we set it to 0.5 for the DANCE and HIP-POP examples). ω^s is increased when the new background seems not stable enough (increased to 0.5 for BOY, DANCE, and WALK). Changing the weights only requires recomputing the dynamic programming optimization, which is relatively fast (see below).

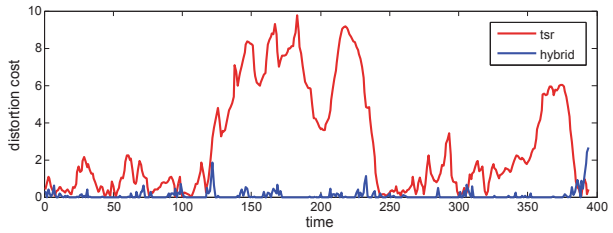


Figure 16: *Comparison of distortion cost resulting from using tsr only vs. the hybrid warp model for the DANCE example, with $\omega^m = 10$, $\omega^s = 0.1$, $\omega^d = 0.1$, $\omega^c = 1000$.*

Table 3: *Processing time for selected examples.*

	Fg Frames	Bg Frames	Pre-Pro. (s)	DP (s)
GOAT	187	383	293	4
BOY	494	2022	2154	43
PENGUIN	250	653	562	18
GIRL	599	795	756	31

6.2 Performance

Table 3 reports the computation times for several examples. Most of the computation time is spent in pre-processing, which is slow because of searching for similar neighbors among the target frames, and computing the transformation between each pair of neighbors. Fortunately, pre-processing is not sensitive to parameters, and needs to be performed only once for each example. We did not perform special optimizations to the pre-processing code.

After pre-processing, the dynamic programming optimization stage is rather fast with our optimized C++ implementation. The most costly operation in optimization is to compute the fitting given by eq. (3), which may be written as a least squares problem $Ax = B$. Note that there are only 4 parameters for tsr and 3 parameters for ts, and for better efficiency we use a direct method to solve the least square problem, $x = (A^T A)^{-1} A^T B$, where $A^T A$ is a 3×3 or a 4×4 matrix, whose inverse can be computed analytically.

Further speedup can be achieved by using the GPU. The algorithm in Table 1 is highly parallel because for each time t, the processing of each candidate B'_j is independent.

6.3 Limitations

Our method is fully 2D-based, and operates by searching for a set of optimal correspondences and warping functions, so its ability to produce a good result is greatly dependent on the inputs, and artifacts will appear whenever the required background motion cannot be reproduced via this mechanism. A few examples are described below.

Figure 17 shows a case where the background is scaled up too much. This typically happens when the source camera zooms in, but not the target camera. Our method then attempts to compensate for the lack of suitable target frames by introducing more scaling. On the other hand, when a zoom-out is present in the source, but not in the target, it may result in introduction of blank regions, as shown in Figure 18(left). Generally, blank regions can appear if the target scene is not covered well enough to "contain" the source camera movement. Such a case is shown in Figure 18(middle).

Significant differences in view direction and perspective can also introduce artifacts. To test the tolerance of our method to the difference in view direction, we use the WALK example for a stress test. The original camera simply pans from right to left, with the view direction almost perpendicular to the moving direction. We captured several target videos where the camera translates almost in the same way as the source camera, but with different view directions (the horizontal angular difference ranges as 0°, 15°, 30°, 45°). The results are shown in Figure 19. For this example we find that our method can tolerate an angular difference up to 15 degrees, while larger differences introduce undesirable rotation and blank regions. Note that although video-based novel view synthesis has been extensively studied [Germann et al. 2012], this remains a difficult problem in general scenes.

Figure 17: *Excessive zoom in of the background (left), the green rectangle is the clipping region in the original background.*

Figure 18: *Blank regions due to camera zoom out (left) and uncaptured regions in the target scene (middle). The rightmost figure shows the case of unnatural subject placement (feet in the air).*

Since our method does not consider geometry and semantic meaning, sometimes it may place the subject in unnatural positions, as demonstrated in Figure 18(right). Generally, if the ground area is large enough, the user may avoid this case by adjusting W_0. In addition, if the geometry of the ground supporting the subject is significantly different between source and target, unnatural results would also be produced.

The above limitations may be aggravated in dynamic scenes, as has been pointed out in Section 5.3. Although we show some examples of dynamic scenes, as the camera motion becomes more complex, and the differences between the source and target camera motions increase, our solution will reveal its limitations sooner with dynamic scenes than with static ones. Dynamic scenes usually prove more challenging in many video editing tasks, such as video panoramas, video retargeting, stabilization, etc. Our method is no exception in this regard, and future work is needed on motion compatible dynamic background replacement.

7 Conclusion

This paper proposes an effective method for background replacement in hand-held video, making it possible to produce challenging video composites without using expensive professional equipment. To our knowledge, we present the first effective solution for this purpose.

Our method is elegantly formulated as the problem of Dynamic Space-Time Warping, which produces slippage-free results by searching for the optimal set of target correspondences and warping functions for the entire source sequence. Slippage is eliminated by

Figure 19: *Limitation in handling difference of view direction. The shown image is frame 150 of WALK.*

matching the local motion around contact points, and distortion is reduced by our spatially-varying fitting of a restricted hybrid warp model. No 3D reconstruction or view interpolation is performed in the process, making our method more robust in general cases. The globally optimal solution can be found efficiently using dynamic programming.

Future work is needed to alleviate existing limitations in handling differences in geometry and camera motion, as well as improved handling of dynamic target scenes. Our current method does not make any changes to the foreground; however, in some situations it could be helpful to alter the foreground in order to better fit the target camera motion and scene geometry. In addition, in this paper we focus on motion consistency, while for a more comprehensive compositing solution, other related issues, such as more sophisticated shadow generation, as well as illumination and appearance harmonization [Sunkavalli et al. 2010] should also be addressed.

Acknowledgements

We thank the anonymous reviewers for their valuable comments, Maggie Chen for video narration, and Xibin Song and Xueyu Bai *et al.* for their hard work preparing the dataset. This work was supported by NSFC (No. 61202149, 61173070, 61232011, 61025012), RGC-NSFC (No. 61161160567), Special Funds of Taishan Scholar Construction Project, Foundation for Outstanding Young Scientist in Shandong Province (No.BS2013DX011), the Israel Science Foundation and the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

AGARWALA, A., HERTZMANN, A., SALESIN, D. H., AND SEITZ, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph. 23*, 3 (Aug.), 584–591.

AGARWALA, A., ZHENG, K. C., PAL, C., AGRAWALA, M., COHEN, M., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2005. Panoramic video textures. *ACM Trans. Graph. 24*, 3 (July), 821–827.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video SnapCut: Robust video object cutout using localized classifiers. *ACM Trans. Graph. 28*, 3 (July), 70:1–70:11.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Trans. Graph. 21*, 3 (July), 243–248.

FARBMAN, Z., AND LISCHINSKI, D. 2011. Tonal stabilization of video. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011) 30*, 4, 89:1 – 89:9.

FLAGG, M., NAKAZAWA, A., ZHANG, Q., KANG, S. B., RYU, Y. K., ESSA, I., AND REHG, J. M. 2009. Human video textures. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, 199–206.

GERMANN, M., POPA, T., KEISER, R., ZIEGLER, R., AND GROSS, M. 2012. Novel-view synthesis of outdoor sport events using an adaptive view-dependent geometry. *Comp. Graph. Forum 31*, 2pt1 (May), 325–333.

GLEICHER, M. L., AND LIU, F. 2008. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 5*, 1, 2.

GOLDSTEIN, A., AND FATTAL, R. 2012. Video stabilization using epipolar geometry. *ACM Trans. Graph. 31*, 5, 126.

GRUNDMANN, M., KWATRA, V., AND ESSA, I. 2011. Auto-directed video stabilization with robust l1 optimal camera paths. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 225–232.

HARTLEY, R., AND ZISSERMAN, A. 2004. *Multiple view geometry in computer vision*, 2nd ed. Cambridge Univ Press.

LIU, F., GLEICHER, M., JIN, H., AND AGARWALA, A. 2009. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph. 28*, 3 (July), 44:1–44:9.

LIU, F., GLEICHER, M., WANG, J., JIN, H., AND AGARWALA, A. 2011. Subspace video stabilization. *ACM Trans. Graph. 30*, 1, 4.

LIU, S., YUAN, L., TAN, P., AND SUN, J. 2013. Bundled camera paths for video stabilization. *ACM Trans. Graph. 32*, 4, 78.

OKABE, M., ANJYOR, K., AND ONAI, R. 2011. Creating fluid animation from a single image using video database. *Computer Graphics Forum 30*, 7, 1973–1982.

SAND, P., AND TELLER, S. 2004. Video matching. *ACM Trans. Graph. 23*, 3, 592–599.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proc. 27th annual conference on Computer Graphics and interactive techniques*, ACM Press/Addison-Wesley, 489–498.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *Proc. 23rd annual conference on Computer Graphics and interactive techniques*, ACM, 259–268.

STEEDLY, D., PAL, C., AND SZELISKI, R. 2005. Efficiently registering video into panoramic mosaics. In *Proc. ICCV '05*, vol. 2.

STICH, T., LINZ, C., ALBUQUERQUE, G., AND MAGNOR, M. 2008. View and time interpolation in image space. *Comp. Graph. Forum 27*, 7, 1781–1787.

SUNKAVALLI, K., JOHNSON, M. K., MATUSIK, W., AND PFISTER, H. 2010. Multi-scale image harmonization. *ACM Trans. Graphics 29*, 4, 125:1–125:10.

WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cutout. *ACM Trans. Graph. 24*, 3 (July), 585–594.

XU, F., LIU, Y., STOLL, C., TOMPKIN, J., BHARAJ, G., DAI, Q., SEIDEL, H.-P., KAUTZ, J., AND THEOBALT, C. 2011. Video-based characters: Creating new human performances from a multi-view video database. *ACM Trans. Graph. 30*, 4 (July), 32:1–32:10.

ZHANG, Y., CORREA, C. D., AND MA, K.-L. 2011. Graph-based fire synthesis. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 187–194.