Content-Preserving Warps for 3D Video Stabilization

Feng Liu
Michael Gleicher
University of Wisconsin-Madison

Hailin Jin
Aseem Agarwala
Adobe Systems, Inc.

Abstract

We describe a technique that transforms a video from a hand-held video camera so that it appears as if it were taken with a directed camera motion. Our method adjusts the video to appear as if it were taken from nearby viewpoints, allowing 3D camera movements to be simulated. By aiming only for perceptual plausibility, rather than accurate reconstruction, we are able to develop algorithms that can effectively recreate dynamic scenes from a single source video. Our technique first recovers the original 3D camera motion and a sparse set of 3D, static scene points using an off-the-shelf structure-from-motion system. Then, a desired camera path is computed either automatically (e.g., by fitting a linear or quadratic path) or interactively. Finally, our technique performs a least-squares optimization that computes a spatially-varying warp from each input video frame into an output frame. The warp is computed to both follow the sparse displacements suggested by the recovered 3D structure, *and* avoid deforming the content in the video frame. Our experiments on stabilizing challenging videos of dynamic scenes demonstrate the effectiveness of our technique.

1 Introduction

While digital still photography has progressed to the point where most amateurs can easily capture high-quality images, the quality gap between professional and amateur-level video remains remarkably wide. One of the biggest components of this gap is camera motion. Most camera motions in casual video are shot hand-held, yielding videos that are difficult to watch, even if video stabilization is used to remove high-frequency jitters. In contrast, some of the most striking camera movements in professional productions are “tracking shots” [Kawin 1992], where cameras are moved along smooth, simple paths. Professionals achieve such motion with sophisticated equipment, such as cameras mounted on rails or steadicams, that are too cumbersome or expensive for amateurs.

In this paper, we describe a technique that allows a user to transform their hand-held videos to have the appearance of an idealized camera motion, such as a tracking shot, as a post-processing step. Given a video sequence from a single video camera, our algorithm can simulate any camera motion that is reasonably close to the captured one. We focus on creating canonical camera motions, such as linear or parabolic paths, because such paths have a striking effect and are difficult to create without extensive equipment. Our method can also perform stabilization using low-pass filtering of the original camera motion to give the appearance of a steadicam. Given a desired output camera path, our method then automatically warps the input sequence so that it appears to have been captured along the specified path.

While existing video stabilization algorithms are successful at removing small camera jitters, they typically cannot produce the more aggressive changes required to synthesize idealized camera motions. Most existing methods operate purely in 2D; they apply full-frame 2D warps (e.g., affine or projective) to each image that best remove jitter from the trajectory of features in the video. These 2D methods are fundamentally limited in two ways: first, a full-frame warp cannot model the parallax that is induced by a translational shift in viewpoint; second, there is no connection between the 2D warp and a 3D camera motion, making it impossible to describe desired camera paths in 3D. We therefore consider a 3D approach. Image-based rendering methods can be used to perform video stabilization in 3D by rendering what a camera would have seen along the desired camera path [Buehler et al. 2001a]. However, these techniques are currently limited to static scenes, since they render a novel viewpoint by combining content from multiple video frames, and therefore multiple moments in time.

Our work is the first technique that can perform 3D video stabilization for dynamic scenes. In our method, dynamic content and other temporal properties of video are preserved because each output frame is rendered as a warp of a single input frame. This constraint implies that we must perform accurate novel view interpolation from a single image, which is extremely challenging [Hoiem et al. 2005]. Performing this task for a non-rigid dynamic scene captured by a single camera while maintaining temporal coherence is even harder; in fact, to the best of our knowledge it has never been attempted. An accurate solution would require solving several challenging computer vision problems, such as video layer separation [Chuang et al. 2002], non-rigid 3D tracking [Torresani et al. 2008], and video hole-filling [Wexler et al. 2004]. In this paper we provide a technique for novel view interpolation that avoids these challenging vision problems by relaxing the constraint of a physically-correct reconstruction. For our application, a perceptually plausible result is sufficient: we simply want to provide the illusion that the camera moves along a new but nearby path. In practice, we find our technique is effective for video stabilization even though our novel views are not physically accurate and would not match the ground truth.

Our method takes advantage of recent advances in two areas of research: shape-preserving image deformation [Igarashi et al. 2005], which deforms images according to user-specified handles while minimizing the distortion of local shape; and content-aware image resizing [Avidan and Shamir 2007; Wolf et al. 2007], which changes the size of images while preserving salient image content. Both of these methods minimize perceivable image distortion by optimally distributing the deformation induced by user-controlled edits across the 2D domain. We apply this same principle to image warps for 3D video stabilization, though in our case we optimally distribute the distortion induced by a 3D viewpoint change rather than user-controlled deformation. Since the change in viewpoint required by video stabilization is typically small, we have found that this not-physically-correct approach to novel view interpolation is sufficient even for challenging videos of dynamic scenes.

[1] http://www.cs.wisc.edu/graphics/Gallery/WarpFor3DStabilization/

Our method consists of three stages. First, it recovers the 3D camera motion and a sparse set of 3D, static scene points using an off-the-shelf structure-from-motion (SFM) system. Second, the user interactively specifies a desired camera path, or chooses one of three camera path options: linear, parabolic, or a smoothed version of the original; our algorithm then automatically fits a camera path to the input. Finally, our technique performs a least-squares optimization that computes a spatially-varying warp from each input video frame into an output frame. The warp is computed to both follow the sparse displacements suggested by the recovered 3D structure, *and* minimize distortion of local shape and content in the video frames. The result is not accurate, in the sense that it will not reveal the disocclusions or non-Lambertian effects that an actual viewpoint shift should yield; however, for the purposes of video stabilization, we have found that these inaccuracies are difficult to notice in casual viewing. As we show in our results, our method is able to convincingly render a range of output camera paths that are reasonably close to the input path, even for highly dynamic scenes.

2 Related Work

Two-dimensional video stabilization techniques have reached a level of maturity that they are commonly implemented in on-camera hardware and run in real time [Morimoto and Chellappa 1997]. This approach can be sufficient if the user only wishes to damp undesired camera shake, if the input camera motion consists mostly of rotation with very little translation, or if the scene is planar or very distant. However, in the common case of a camera moving through a three-dimensional scene, there is typically a large gap between 2D video stabilization and professional-quality camera paths.

The idea of transforming hand-held videos to appear as if they were taken as a proper tracking shot was first realized by Gleicher and Liu [2008]. Their approach segments videos and applies idealized camera movements to each. However, this approach is based on full-frame 2D warping, and therefore suffers (as all 2D approaches) from two fundamental limitations: it cannot reason about the movement of the physical camera in 3D, and it is limited in the amount of viewpoint change for scenes with non-trivial depth complexity.

The 3D approach to video stabilization was first described by Buehler et al. [2001a]. In 3D video stabilization, the 3D camera motion is tracked using structure-from-motion [Hartley and Zisserman 2000], and a desired 3D camera path is fit to the hand-held input path. With this setup, video stabilization can be reduced to the classic image-based rendering (IBR) problem of novel view interpolation: given a collection of input video frames, synthesize the images which would have been seen from viewpoints along the desired camera path. Though the novel viewpoint interpolation problem is challenging and ill-posed, recent sophisticated techniques have demonstrated high-quality video stabilization results [Fitzgibbon et al. 2005; Bhat et al. 2007]. However, the limitation to static scenes renders these approaches impractical, since most of us shoot video of dynamic content, e.g., people.

Image warping and deformation techniques have a long history [Gomes et al. 1998]. Recent efforts have focused on deformation controlled by a user who pulls on various handles [Igarashi et al. 2005; Schaefer et al. 2006] while minimizing distortion of local shape, as measured by the local deviation from conformal or rigid transformations. These methods, which build on earlier work in as-rigid-as-possible shape interpolation [Alexa et al. 2000], are able to minimize perceivable distortion much more effectively than traditional space-warp methods [Beier and Neely 1992] or standard scattered data interpolation [Bookstein 1989]. Our method applies this principle in computing spatially-varying warps induced by the recovered 3D scene structure. A related image deformation problem is to change the size or aspect ratio of an image without distorting salient image structure. Seam Carving [Avidan and Shamir 2007] exploited the fact that less perceptually salient regions in an image can be deformed more freely than salient regions, and was later extended to video [Rubinstein et al. 2008]. However, the discrete algorithm behind Seam Carving requires removing one pixel from each image row or column, which limits its application to general image warping. Others have explored more continuous formulations [Gal et al. 2006; Wolf et al. 2007; Wang et al. 2008], which deform a quad mesh placed on the image according to the salience (or user-marked importance) found within each quad; we take this approach in designing our deformation technique.

A limitation of our approach is that it requires successful computation of video structure-from-motion. However, this step has become commonplace in the visual effects industry, and commercial 3D camera trackers like Boujou[2] and Syntheyes[3] are widely used. We use the free and publicly available Voodoo camera tracker[4], which has been used in a number of recent research systems [van den Hengel et al. 2007; Thormählen and Seidel 2008]. Finally, there are a number of orthogonal issues in video stabilization that we do not address [Matsushita et al. 2006], such as removing motion blur, and full-frame video stabilization that avoids the loss of information at the video boundaries via hole-filling (we simply crop our output). These techniques could be combined with our method to yield a complete video stabilization solution.

3 Traditional video stabilization

We begin by describing the current technical approaches to video stabilization in more detail, and showing their results on the example sequence in Video Figure 1 (since many of the issues we discuss can only be understood in an animated form, we will refer to a set of video figures that are included as supplemental materials and on the project web site[1]).

3.1 2D stabilization

Traditional 2D video stabilization proceeds in three steps. First, a 2D motion model, such as an affine or projective transformation, is estimated between consecutive frames. Second, the parameters of this motion model are low-pass filtered across time. Third, full-frame warps computed between the original and filtered motion models are applied to remove high-frequency camera shake. Video Figures 2 and 3 show two results of this approach, created using our implementation of Matsushita et al. [2006] (we do not perform the inpainting or deblurring steps, and the two videos contain different degrees of motion smoothing).

While 2D stabilization can significantly reduce camera shake, it cannot simulate an idealized camera path similar to what can be found in professional tracking shots. Since the 2D method has no knowledge of the 3D trajectory of the input camera, it cannot reason in 3D about what the output camera path should be, and what the scene would have looked like from this path. Instead, it must make do with fitting projective transformations (which are poor approximations for motion through a 3D scene) and low-pass filtering them. Strong low-pass filtering (Video Figure 3) can lead to visible distortions of the video content, while weak filtering (Video Figure 2) only damps shake; neither can simulate directed camera motions.

[2] http://www.2d3.com

[3] http://ssontech.com

[4] http://www.digilab.uni-hannover.de

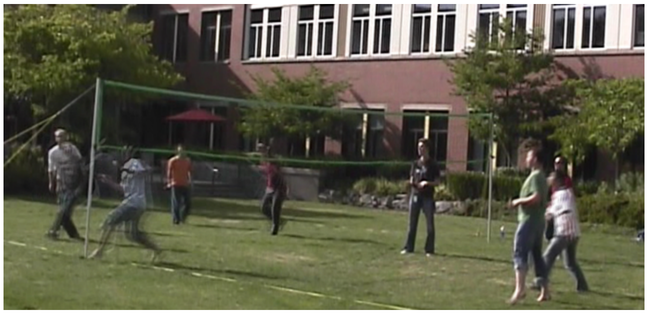

Figure 1: *A crop of a video frame created using novel view interpolation. While the static portions of the scene appear normal, the moving people suffer from ghosting.*

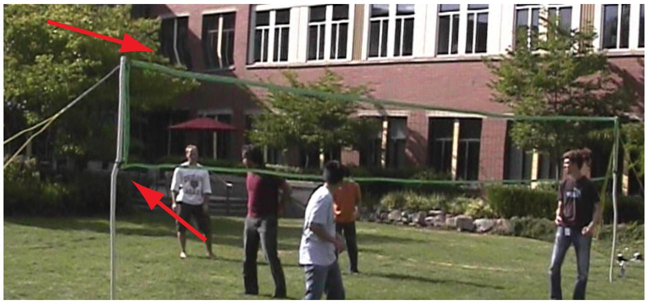

Figure 2: *A crop of a video frame created using generic sparse data interpolation. The result does not contain ghosting, but distorts structures such as the window and pole highlighted with red arrows.*

3.2 3D stabilization

The 3D approach to video stabilization is more powerful, though also more computationally complex. Here, the actual 3D trajectory of the original camera is first estimated using standard structure-from-motion [Hartley and Zisserman 2000]; this step also results in a sparse 3D point cloud describing the 3D geometry of the scene. Second, a desired camera path is fit to the original trajectory (we describe several approaches to computing such a path in Section 4.3). Finally, an output video is created by rendering the scene as it would have been seen from the new, desired camera trajectory.

There are a number of techniques for rendering novel views of a scene; in Video Figure 4 we show a video stabilization result created using the well-known unstructured lumigraph rendering algorithm [Buehler et al. 2001b]. The result is remarkably stable. However, like all novel view interpolation algorithms, each output frame is rendered as a blend of multiple input video frames. Therefore, dynamic scene content suffers from ghosting (we show a still frame example of this ghosting in Figure 1).

One approach to handling dynamic scene content would be to identify the dynamic objects, matte them out, use novel view interpolation to synthesize the background, re-composite, and fill any remaining holes. However, each of these steps is a challenging problem, and the probability that all would complete successfully is low. Therefore, in the next section we introduce the constraint that each output video frame be rendered only from the content in its corresponding input video frame.

4 Our approach

Our approach begins similarly to the 3D stabilization technique just described; we recover the original 3D camera motion and sparse 3D point cloud using structure-from-motion, and specify a desired output camera motion in 3D (in this section we assume the output path is given; our approach for computing one is described in Section 4.3). Then, rather than synthesize novel views using multiple input video frames, we use both the sparse 3D point cloud and the content of the video frames as a guide in warping each input video frame into its corresponding output video frame.

More specifically, we compute an output video sequence from the input video such that each output video frame I_t is a warp of its corresponding input frame $\hat{I}_t$ (since we treat each frame independently, we will omit the t subscript from now on). As guidance we have a sparse 3D point cloud which we can project into both the input and output cameras, yielding two sets of corresponding 2D points: P in the output, and $\hat{P}$ in the input. Each k'th pair of projected points yields a 2D displacement $P_k - \hat{P}_k$ that can guide the warp from input to output. The problem remaining is to create a dense warp guided by this sparse set of displacements. This warp, which can use the displacements as either soft or hard constraints, should maintain the illusion of a natural video by maintaining temporal coherence and not distorting scene content. We first consider two simple warping solutions, the first of which is not successful, and the second of which is moderately successful.

The first solution is to use generic sparse data interpolation to yield a dense warp from the sparse input. In Video Figure 5 we show a result computed by simply triangulating the sparse points and using barycentric coordinates to interpolate the displacements inside the triangles; the displacements are therefore treated as hard constraints. The result has a number of problems. Most significantly, important scene structures are distorted (we show a still example in Figure 2). These distortions typically occur near occlusions, which are the most challenging areas for novel view interpolation. Also, problems occur near the frame boundaries because extrapolation outside the hull of the points is challenging (for this example, we do not perform extrapolation). Finally, treating the displacements as hard constraints leads to temporal incoherence since the reconstructed 3D points are not typically visible for the entire video. Popping and jittering occur when the corresponding displacements appear and disappear over time. In this example, we use a very short segment of video and only include points that last over the entire duration of the video; however, the problem is unavoidable in longer sequences. Our approach for preserving temporal coherence, which is only applicable if displacements are used as soft constraints, is described in Section 4.1.4.

The second alternative is to fit a full-frame warp to the sparse displacements, such as a homography (thereby treating the displacements as a soft constraint). We show a somewhat successful result of this technique in Video Figure 6. This method can achieve good results if the depth variation in the scene is not large, or if the desired camera path is very close to the original. We show a less successful result of this technique in Video Figure 7. In the general case, a homography is too constrained a model to sufficiently fit the desired displacements. This deficiency can result in undesired distortion (we show an individual frame example in Figure 3), and temporal wobbling. However, this novel approach is the best of the alternatives we have considered up to now.

The first solution described above is too flexible; it exactly satisfies the sparse displacements, but does not respect structures in the image and therefore introduces local distortion near occlusions (among other problems). The second solution is too stiff, and cannot model the sparse displacements well enough to simulate the desired camera position. Can we design an in-between solution that is more flexible than a full-frame warp, but avoids visible distortion of the video frame content? As we show with our result in Video Figure 8, the answer is yes; we now describe the method.

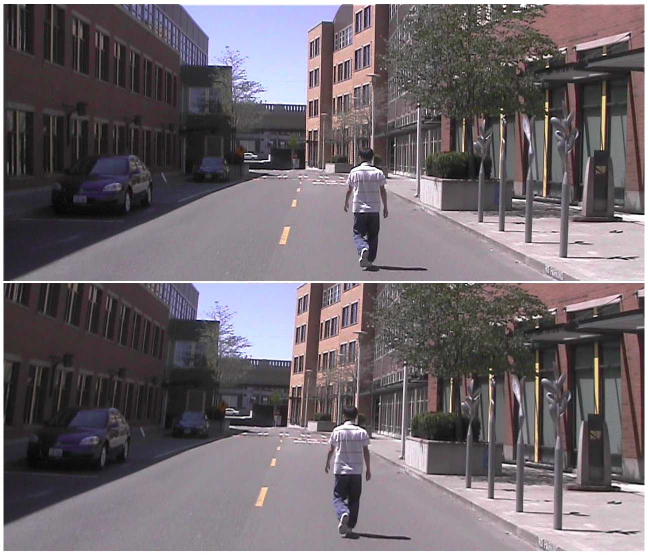

Figure 3: *A crop of an input video frame, followed by an output video frame created using a full-frame homography fit to the set of sparse displacements. Since the homography cannot model the displacements well, distortion is introduced; while the person and structures on the right side of the frame are upright, the wall and window frames on the building on the left (and, to a lesser degree, on the building in the back) shear towards the right.*

4.1 Content-preserving warps

Our content-preserving warps are designed with several principles in mind. First, the sparse displacements should be treated as soft constraints, since hard constraints will lead to distortions near occlusions and temporal incoherence. Instead, we wish to spread the error near occlusions slowly across the rest of the image and into areas where the eye will notice them less (even if this answer will not be physically accurate). Second, the warp should attempt to preserve the content of the image. This second goal is met in two ways.

First, we make the observation that since the desired camera will not be very far from the original camera, the local content in the original image should not need to be distorted significantly. So, we would like to avoid local shearing or non-uniform scaling. We therefore prefer warps that locally resemble a similarity transformation. (Recent work in image warping [Alexa et al. 2000; Igarashi et al. 2005; Schaefer et al. 2006] explore both as-similar-as-possible and as-rigid-as-possible warps, and generally reach the conclusion that rigid transformations are better. However, for our application, uniform scaling is acceptable since objects may need to move closer or farther from the camera.) Second, we observe that violations of this constraint will be less noticeable in less salient regions of the image (e.g., sky), while the shape of strong edges should be better preserved.

A solution best satisfying the above principles in a least-squares sense can be computed by discretizing the warp into a grid and minimizing an energy function of two weighted energy terms: a data term for each sparse displacement, and a similarity transformation term that measures the deviation of each grid cell from a similarity transformation, weighted by the salience of the grid cell.

We divide the original video frame $\hat{I}$ into an $n \times m$ uniform grid mesh, where $\hat{V}_{i,j}$ is the grid vertex at position (i,j). We compute a warped version of this grid for the output video frame, where each $V_{i,j}$ is a 2D unknown to be computed.

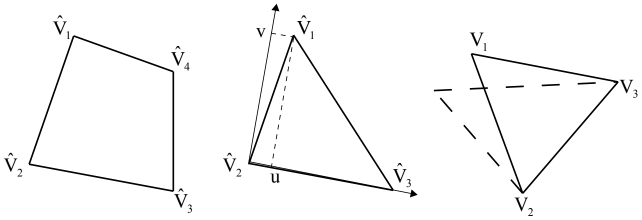

Figure 4: *A triangle vertex can be expressed in the local coordinate system (u, v) of its opposite edge. The deviation from a similarity transformation of a warp can be measured as the distance between the vertex and the location it would have had under a similarity transformation (dashed lines).*

4.1.1 Data term

Each projected point $\hat{P}_k$ in the input frame is typically not coincident with a vertex $\hat{V}_{i,j}$, so we must represent each constraint with a bilinear interpolation of the four corners of the enclosing grid cell. We define $\hat{V}_k$ as a vector of the four vertices enclosing the grid cell that $\hat{P}_k$ projects to; V_k represents the same four vertices in the output grid. The vector w_k contains the four bilinear interpolation coefficients that sum to 1, so that $\hat{P}_k = w_k^T \hat{V}_k$ represents a bilinear interpolation operation for a projected point. We compute w_k by finding the grid cell that $\hat{P}_k$ projects to and inverting its bilinear interpolation [Heckbert 1989]. Then, the data term is

$$E_d = \sum_k \| w_k^T V_k - P_k \|^2, \qquad (1)$$

where V_k contains four unknowns, and w_k and P_k are known. This term minimizes the distance between the output projected point P_k and the interpolated location in the grid cell in the output that corresponds to the input grid cell containing $\hat{P}_k$.

4.1.2 Similarity transformation term

The similarity transformation term measures the deviation of each output grid cell from a similarity transformation of its corresponding input grid cell. We split each grid cell into two triangles and then apply the method of Igarashi et al. [2005]. As shown in Figure 4, each vertex can be represented in a local coordinate system formed by the vector between the other two vertices, and the 90 degree rotation of that vector. For example, V_1 can be defined using V_2 and V_3 as

$$V_1 = V_2 + u(V_3 - V_2) + vR_{90}(V_3 - V_2), \;\; R_{90} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad (2)$$

where u, v are the known coordinates within the local coordinate system.[5] However, if the output triangle has not undergone a similarity transformation from its input, V_1 will not coincide with the location calculated from V_2 and V_3. We therefore minimize the distance between V_1 and its desired location under a similarity transformation.

$$E_s(V_1) = w_s \| V_1 - (V_2 + u(V_3 - V_2) + vR_{90}(V_3 - V_2)) \|^2, \qquad (3)$$

where w_s is a weight described in Section 4.1.3. Notice that there are two such triangles for each of the four quads cornered by V_1. The full energy term $E_s(V)$ is formed by summing Equation 3 over all eight triangles of each vertex. Using all eight triangles is redundant, but avoids special handling along the grid boundaries.

[5] Note that if the grid cell is square $u = 0$ and $v = 1$. However, in Section 4.2 we add an initial warping stage that can yield non-square grid cells.

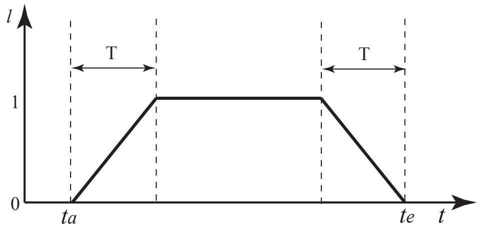

Figure 5: *The weight of a reconstructed scene point fades-in and fades-out over time in the optimization to preserve temporal coherence.*

4.1.3 Salience

The above energy term $E_s(V)$ is successful in preserving the local shape in the image. However, it is not as important to preserve the shape of quads of low perceptual salience, such as a uniform region. High salience quads, such as those containing a strong image edge, should maintain their shape more strongly. Therefore, we compute the weight w_s in Equation 3 of each triangle constraint by the salience of its enclosing grid cell.

There are a number of sophisticated techniques for measuring perceptual salience [Itti et al. 1998; Wang et al. 2008]. For our application, we found that simply setting w_s to the L_2 norm of the color variance inside a grid cell is a sufficient measure of salience. We also add an epsilon of 0.5 to w_s to make sure that each variable in the optimization is subject to a constraint, even if the variance is 0.

4.1.4 Temporal coherence

Our stabilization results are guided by the set of reconstructed 3D scene points. However, the points estimated by structure-from-motion do not typically last the entire video; each reconstructed point only projects to a subset of the video frames, so points will appear and disappear over time instantly, leading to very visible temporal artifacts. We solve this problem in two simple ways. First, we only use scene points that last for at least $N = 20$ video frames; this requirement helps to avoid incorrectly reconstructed scene points, and scene points that might correspond to moving objects. Second, and most importantly, we fade-in and fade-out the influence of each sparse displacement over time. We use the simple piecewise-linear function $l(t)$ shown in Figure 5, where t_a and t_b are the beginning and ending frames of a feature point, and T is set to 50. If $t_b - t_a < 2T$, we also scale the peak value of $l(t)$ by $(t_b - t_a)/2T$. This function $l(t)$ is then used to weight the data term in Equation 1. The result of our method with and without $l(t)$ can be seen in Video Figures 9 and 10.

4.1.5 Dynamic scene content

Surprisingly, our method requires no special handling of moving objects in the scene. Structure-from-motion should not reconstruct any 3D scene points corresponding to moving objects, since their motion will not be consistent with the motion of the static 3D scene structure (this property is reinforced by our elimination of reconstructed points that do not last more than a minimum number of frames). Therefore, our stabilization method will warp a moving scene object according to some interpolation of the displacements of the background scene points behind the object. This warp may not be accurate if there is a large depth difference between the object and its surrounding background, but this inaccuracy tends to be masked perceptually by the fact that the object is moving to begin with.

4.1.6 Optimization and result

The sum of the above weighted energy terms E_d and E_s is a linear least-squares problem in the set of unknown grid vertices $V_{i,j}$. The final energy equation is

$$E = E_d + \alpha E_s,$$

where α is the relative weight of the data and smoothness terms. This energy equation is quadratic and can be minimized by solving a sparse linear system, where the matrix is narrow-banded since each vertex only shares constraints with its eight neighbors. We solve it efficiently using a standard sparse linear system solver (specifically, the Matlab backslash operator). The target video can then be rendered using a standard texture mapping algorithm according to the warped mesh.

This approach alone can generate good results for many sequences; we show a successful result in Video Figure 8. However, its success strongly depends on the density and distribution of the sparse displacements. Figure 11 shows a result that is mostly successful. However, if one watches the buildings on the upper left, they clearly wobble. This artifact occurs because there are few or no reconstructed scene points in that area; in this case, the data term has little effect and the region remains similar to the input, which is shaky.

There is, however, a degree of freedom that we are not yet exploiting: the rotation of the input camera. The effect of a 3D rotation of the input camera can be accurately simulated by applying a projective transformation. What if the user had rotated the camera in a slightly different direction when shooting an input video frame? In areas with few reconstructed scene points, a rotation of the input camera could easily yield a different output, since the data term here is weak. This sensitivity to camera rotation implies that there are some rotations of the camera that could lead to better results than other rotations in areas with few scene points. Can we exploit this degree of freedom by pre-warping the input with a carefully chosen projective transformation? In the next section we show that we can.

4.2 Pre-warping

We now show how to improve results by applying a full-frame warp to each input video frame before our spatially-varying content-preserving warp is applied. We explored two options for pre-warping the input video with a full-frame warp; both are successful and typically yield similar results, though the second approach is slightly more robust and is therefore used in all our final results.

4.2.1 Infinite homographies

The first option is to apply a rotation to the input camera so that it points in the same direction as the desired output camera. In this way, any rotational jitter is removed, and the output and input cameras only differ by a translation. Specifically, a projective transformation that corresponds to a rigid rotation of a camera is called an infinite homography [Hartley and Zisserman 2000], and can be computed as $H = K\tilde{R}(KR)^{-1}$, where K is the shared intrinsic camera matrix, and R and $\tilde{R}$ are the camera orientation matrices of the input and output cameras. This approach is successful for the vast majority of our videos. However, we found this pre-warping technique to be less successful when the translation of the output camera from the input is large.

4.2.2 General homographies

Another option is to exploit the full warping power of a homography, rather than an infinite homography that is limited to the three degrees of freedom of rigid rotation. In this case, we simply compute the best-fit homography in a least-squares sense to the set of sparse displacements. Notice that this pre-warping approach is identical to the second alternative we considered at the beginning of Section 4, where we noted that this full-frame warp can sometimes provide good results. On the other hand, a full-frame warp often yields distortion and wobbling, since it is not sufficient to fully model the desired displacements. Nonetheless, it is useful as an initialization. It can be considered a first-order approximation of the desired warp, to which we then apply our content-preserving warp. In the regions where our 3D reconstruction has a sufficient number of reconstructed points, the data term described in Section 4.1.1 will take over and this initialization will have little influence. Where there is a paucity of reconstructed scene points, the initialization provided by the homography will have stronger influence and provide a more stable starting point than the initial, shaky video. We use this pre-warping technique to produce the final results. The final result for the example just shown is given in Video Figure 12; the wobbling of the buildings is greatly reduced.

4.3 Camera path planning

Our content-preserving warping method allows us to re-create a frame of video from a different viewpoint than it was originally taken. This capability allows us to cast the stabilization problem as one of finding a new camera trajectory (position and orientation). Our method must find a camera trajectory that satisfies two goals:

1. The trajectory should be a "good" camera movement, in the cinematographic sense. For example, it should be free of the high-frequency jitters that stabilization attempts to remove.

2. The trajectory should have maximal overlap with the original video frame. Our approach can only synthesize images for portions of the scene that were seen in the original video. We choose to crop the resulting video to the overlapping region, rather than including empty areas in the frame or hallucinating edge regions [Matsushita et al. 2006].

These goals may conflict: the original trajectory has maximal overlap with the video, but may not be a desirable camera motion (otherwise, we would not need to stabilize it). Conversely, a good camera motion may be so far from the original video that it must be cropped significantly to avoid empty regions. There is often a tradeoff between better motion (which often requires larger changes to the original path), and greater overlap (which requires smaller changes to the path).

To find camera trajectories, we apply three approaches. First, we allow a user to specify new camera trajectories interactively. Second, we use low-pass filtering to create an analog of conventional, jitter-reducing, video stabilization. Finally, we consider methods that automatically construct idealized camera paths.

While the art of cinematography provides a literature and tradition for defining "good" camera movements, these discussions tend to be subjective, content-dependent, and difficult to translate into a computational model (see [Gleicher and Liu 2008] for a recent attempt and discussion). Rather than attempting to automate aesthetic judgements, we instead provide the user with a set of tools that mimic those used by professionals. Effectively, our approach simulates the mechanical camera supports (such as steadicams, damped cranes and dollies) used by professionals as a post-process that can be applied to casually captured video.

In this section we detail the three approaches for determining the new camera trajectories using the following notation. A camera trajectory, $C(t)$ specifies the external camera parameters (position and orientation) as a function of time t. $C_p(t)$ and $C_r(t)$ refer to the position and orientation components respectively, and $\hat{C}(t)$ refers to the camera trajectory of the source video. Since we treat the camera as a rigid body, $C(t) \in SE(3)$ (rigid transformation group) and $C_r(t) \in SO(3)$ (rotation group). The need to find trajectories in these non-Euclidean spaces complicates our algorithms.

4.3.1 Interactive camera path creation

Our system provides the user with tools for specifying camera paths manually. This allows the user to use their judgement as to what constitutes a good camera motion, and to control the tradeoff between motion quality and overlap.

Video Figure 13 shows an example. The user specifies a circular arc path for the camera position, and specifies several keyframes for the camera orientations that are then interpolated. Because the source camera movement is not circular, orientation keyframing is necessary to keep the statue in view throughout the video. Note that this example is extremely challenging for structure-from-motion, as there are few rigid scene features. The lack of trackable features on the fountain yields a small amount of wobbling in our result.

4.3.2 Low-pass filtering

Traditional video stabilization seeks to remove high-frequency artifacts from the apparent camera motion. This is realized by applying a low-pass filter to the apparent camera motion, most typically in 2D. In our approach, we are able to apply filtering to the camera trajectory directly. This 3D filtering effectively simulates the smoothing effect of mechanical stabilization devices used by professionals, such as a steadicam. Filtering in the camera trajectory space also has the advantages that it can consider all parameters of the motion, and can provide distortion-free viewpoints even with aggressive filtering.

Filtering the initial camera trajectory, $\hat{C}(t)$, is complicated by the fact that it is in a non-Euclidean space. We perform the filtering by treating its components separately. $C_p(t)$ is filtered component-wise using standard signal processing operations. The rotation component $C_r(t)$ is filtered using the method of Lee and Shin [2002]. Briefly, the method linearizes the rotation space by differentiating it in time to obtain angular velocities. An adapted version of the filter kernel is applied at each time step by summing over the angular velocities in the local coordinate frame of the time step.

Filtering does not explicitly consider the second goal of maximizing coverage. However, in practice, the filtered path is usually close to the source, so that the coverage loss is sufficiently small.

Like traditional 2D stabilization, our 3D filtering approach is capable of removing jitters in the camera motion. However, it can provide more aggressive smoothing before distortion is introduced. Also, because filtering is done on the camera position and orientation, our 3D approach more closely mimics a steadicam.

4.3.3 Path fitting

We observe that some of the most dramatic cinematographic effects come from very deliberate camera movements where a complex support (like a track or crane) is used to move along a very simple path. To create such movements as a post-process, we provide a tool where the user selects a simple motion model for the camera, and the system computes a new camera motion with this model that approximates the original video and optimizes the frame coverage.

We allow the user to select a motion model for both the camera position and orientation separately. At present, we support constant, linear, and quadratic motion models, allowing the simulation of devices such as a damped tripod (constant position, linear orientation) or a rigid truck (linear position, constant orientation). We parameterize the orientation as a rotation vector (i.e., exponential coordinates [Murray et al. 1994]), so that linear trajectories correspond to (nearly) constant angular velocity motions, and the resulting equations have sufficiently simple forms for efficient computation. This use of a single linearization of the rotation space for the entire motion may be problematic if we consider ranges greater than 180 degrees. However, such movements rarely occur in practice, and can be handled by segmentation or a local linearization approach.

The system computes the new positional motion ($C_p(t)$) by using least-squares fitting to source camera path $\hat{C}_p(t)$. Selection of the rotational motion is complicated by two factors: first, we must consider the frame overlap, and second, we must work in the non-linear space of rotations. Note that we only need to consider frame overlap for rotations as small amounts of rotation can cause large differences in overlap, whereas positional movements have smaller effects. To maximize overlap, we choose to minimize the distance between the apparent (projected) positions of the feature points, rather than trying to match the initial rotations. That is, the rotational trajectory $C_r(t)$ is chosen to minimize

$$\sum_{1 \le t \le T} \sum_{P_k \in F(t)} \|P_k - \hat{P}_k\|_2^2, \tag{4}$$

the sum over each frame (t) and each feature (P_k) of the image position disparity of the feature in the result and source videos. Finding $C_r(t)$ requires solving a non-linear least-squares minimization of Equation 4 over the motion parameters. Our system uses an implementation of the Levenberg-Marquardt algorithm[6]. The supplemental video gives several examples of our fitting approach.

5 Results

We show a number of results of our technique in the supplemental video and the project web site. Each input video was shot handheld, with in-camera stabilization turned off. We also show comparisons to other methods, such as our implementation of 2D stabilization [Matsushita et al. 2006], the 2D stabilization features in iMovie '09[7] and the free Deshaker[8] plugin, and our own implementation of 3D stabilization via novel view interpolation [Buehler et al. 2001a; Buehler et al. 2001b].

As the results show, our method is able to achieve more stable and more directed camera motions than the 2D techniques while avoiding the ghosting of moving scene objects found in previous 3D techniques. Our method can also simulate a range of 3D camera motions, from simple low-pass filtering to linear camera paths with constant speed, depending on whether the user desires a more hand-held or rail-mounted camera feel. Our results are only marginally better than the 2D stabilization alternatives when the output camera path is close to the original, such as camera paths created from low-pass filtering. However, when we specify more significantly different output paths such as linear or parabolic motions, the improvements over 2D stabilization are very clear.

One artifact that can be seen in our results is that we are sometimes not able to remove all vestiges of the original camera motion. The most noticeable manifestation of this artifact is a subtle low-frequency bouncing of scene geometry very near to the camera; this bouncing is a smoothed version of the original up-and-down walking motion. We believe this bouncing arises for two reasons: first, because of small inaccuracies in the 3D reconstruction that is magnified for near-camera geometry, and second, because of a lack of reconstructed scene points in those areas. We still consider these results successful, however, since this walking motion can also be seen in the results of other methods (and usually to a larger degree).

More extreme failures of our method can also happen, such as sudden glitches. In all the examples we have seen, these errors can be traced to outliers in the 3D scene reconstruction that cause large, transient warping errors.

One interesting application of our method is to stabilize videos captured by a camera with a "rolling shutter." Many new consumer-level cameras have a CMOS sensor which cannot be read in its entirety at once, but rather in a rolling fashion from top to bottom. The result is that each video frame is not captured at an instant but rather across a short range of time. Researchers are attempting to incorporate rolling shutter effects into structure-from-motion algorithms [Meingast et al. 2005]. We found that 2D stabilization software fails dramatically on rolling shutter sequences, while our method performs reasonably well if the SFM stage completes. We include a comparison in the supplemental video.

Most of our results were computed with a default smoothness weight of $\alpha = 20$ (Section 4.1.6). However, this parameter has a natural meaning: it represents the trade-off between following the desired camera motion and avoiding distortion. Since our warping scheme is fast (about 2fps), experimenting with this slider is easy, and we expose it to the user and tweak it for a few examples. Finally, the parameter N (Section 4.1.4) is the minimum number of frames from which a 3D point must be visible to be included. The default value $N = 20$ is almost always effective; however, there were a few sequences with extremely slow-moving objects, for which longer values of N were necessary to avoid reconstructing their geometry.

Our results were computed on a 64×36 grid. The time bottleneck of our method is the computation of structure-from-motion; the Voodoo tracker can take many hours for a short sequence. However, there are commercial SFM alternatives that are much faster, and researchers have even described real-time SFM systems [Nister 2003]. Once this stage is pre-computed, our warping technique only takes about half a second per frame.

5.1 Evaluation

To informally evaluate the robustness of our method, we ran a complete testing set of 32 short videos (5-20 seconds) that we captured over two days. We found that 14 of these were completely successful, 15 were moderately successful with some noticeable bouncing, and 3 were not successful due to large 3D reconstruction errors. Note that the structure-from-motion step was completely automatic for all of our results; although the Voodoo system allows interactive editing, we did not exploit this option.

Another evaluation question for our method is: how far can the output camera diverge from the input before artifacts occur? Unfortunately, the answer to this question is hard to quantify. The 3D reconstruction is only defined up to a similarity transformation, and therefore any distances between cameras are unit-less. There are six degrees of freedom in which the camera can diverge from the input. And, finally, the acceptable variation will depend on the nature of the scene and the quality of the 3D reconstruction. In lieu of a quantitative answer, we provide several examples. In Figure 6 we show a sequence of warps of a single input video frame at increasing distances from the input camera, along with their respective grids; for small distances the warps look reasonable while for larger ones visi-

[6] http://www.ics.forth.gr/~lourakis/levmar/

[7] http://www.apple.com/ilife/imovie/

[8] http://www.guthspot.se/video/deshaker.htm

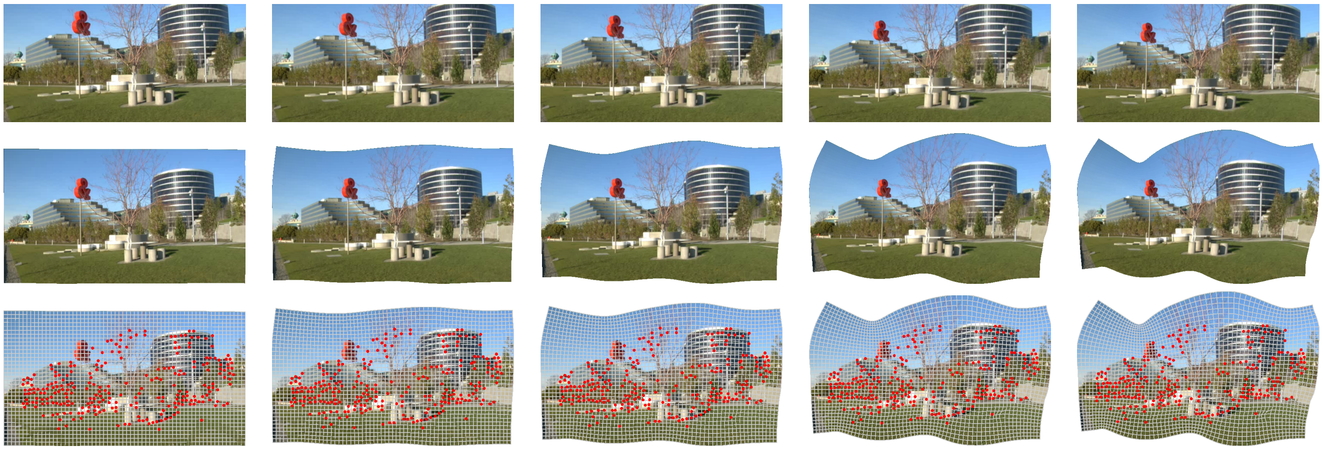

Figure 6: *Each row shows a sequence of warps of a single input video frame created by pulling the camera away from its original location. The top row shows the final cropped result; the middle shows the entire warp result; the bottom shows the corresponding grids and the points that guide the warp. For small camera motions the warps look reasonable, but they become visibly distorted at larger camera displacements.*

ble distortions appear. In the supplemental video we show an example of spiraling the output camera away from its input for a single video frame, while always looking at the centroid of the scene. Seen in this way, the result is clearly not an accurate novel view interpolation, even for small output camera displacements. However, if this video is paused, for small displacements each output frame looks entirely reasonable. This perceptual tolerance of inaccurate novel view interpolation in the context of video stabilization is, perhaps, the most surprising outcome of our technique.

Finally, it is worth noting that salience weighting (Section 4.1.3) has a more subtle effect on the results than we originally expected. We use it for all our examples, but in most cases the subtle improvement provided by salience weighting can only be noticed after watching both results several times. Since our viewpoint shifts are small, the local deformations induced by the sparse displacements are smaller than for resizing applications [Avidan and Shamir 2007; Wolf et al. 2007], and therefore a local deformation that is nearly a similarity transformation can usually be found without exploiting low-salience regions.

5.2 Limitations

A major limitation of our method compared to 2D video stabilization is that it first requires running structure-from-motion. Though SFM techniques have recently reached a high level of maturity, the process is still more brittle and heavyweight than 2D tracking. Also, SFM is limited to videos of scenes with some static regions; otherwise, the motion of the camera is ambiguous. Finally, traditional SFM requires constantly translating cameras, though there has been research on techniques that switch automatically between full and rotation-only motion models [Torr et al. 1999]; a full video stabilization solution must contain this capability, in order to handle both types of motions. Our method also depends on the quality of the 3D scene reconstruction, and the distribution of the scene points across each video frame.

Like most stabilization methods, our results exhibit a smaller field of view than the input because of cropping; however, since our stabilization is more aggressive, the loss of content might be even more severe for our technique. Also, motion blur and other temporal artifacts become more obvious in a stabilized video. Our technique can be combined with methods [Matsushita et al. 2006] that address these specific issues.

5.3 Future work

Since our method allows rendering of any 3D camera motion that is reasonably close to the original, one exciting area for future work is to better support interactive output camera manipulation. Such a system could support keyframing of camera motions, as well as interactive exploration of the range of camera positions and orientations that yield good results. Beyond keyframing, however, combining our method with through-the-lens camera control [Gleicher and Witkin 1992] could allow users to create camera paths more intuitively by controlling the motion of individual objects rather than the camera itself. A different direction would be to provide automation that selects appropriate camera motion models for various segments within each video shot, as introduced by Gleicher and Liu for their 2D system [2008].

Finally, our notion of "perceptual plausibility" is *ad hoc*. Further study may quantify the degree and types of distortions that may be added to videos before they become objectionable.

6 Conclusion

In this paper we described a technique for simulating the appearance of an idealized camera motion, such as a tracking shot, from the input of a single hand-held video sequence. We build upon existing approaches to 3D stabilization, but are able to avoid ghosting of moving scene objects by adding the constraint that each output frame be rendered as a warp of a single input frame. The key insight of our method is that for the purposes of video stabilization, small shifts in viewpoint can be faked by a carefully constructed content-preserving warp, even though the result is not physically accurate. Human vision seems to be surprisingly tolerant of the inaccuracies of content-preserving warps, and we believe this tolerance can be exploited to solve a number of problems in computer graphics.

Acknowledgements

We would like to thank the authors of the Voodoo Camera Tracker, Xue Bai and the other Adobe interns for appearing in several of our videos, and funding from Adobe and NSF grant IIS-0416284.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 157–164.

AVIDAN, S., AND SHAMIR, A. 2007. Seam carving for content-aware image resizing. *ACM Transactions on Graphics 26*, 3 (July), 10:1–10:9.

BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 35–42.

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 *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, 327–338.

BOOKSTEIN, F. L. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell. 11*, 6, 567–585.

BUEHLER, C., BOSSE, M., AND MCMILLAN, L. 2001. Non-metric image-based rendering for video stabilization. In *2001 Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 609–614.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S. J., AND COHEN, M. F. 2001. Unstructured lumigraph rendering. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 425–432.

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

FITZGIBBON, A., WEXLER, Y., AND ZISSERMAN, A. 2005. Image-based rendering using image-based priors. *International Journal of Computer Vision 63*, 2 (July), 141–151.

GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, 297–304.

GLEICHER, M. L., AND LIU, F. 2008. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimed. 5*, 1, 1–28.

GLEICHER, M., AND WITKIN, A. 1992. Through-the-lens camera control. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 331–340.

GOMES, J., DARSA, L., COSTA, B., AND VELHO, L. 1998. *Warping and morphing of graphical objects*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

HARTLEY, R. I., AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

HECKBERT, P. S. 1989. Fundamentals of texture mapping and image warping. Tech. Rep. UCB/CSD-89-516, EECS Department, University of California, Berkeley, Jun.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. *ACM Transactions on Graphics 24*, 3 (Aug.), 577–584.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics 24*, 3 (Aug.), 1134–1141.

ITTI, L., KOCH, C., AND NIEBUR, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (Nov), 1254–1259.

KAWIN, B. 1992. *How Movies Work*. Univ. of California Press.

LEE, J., AND SHIN, S. Y. 2002. General construction of time-domain filters for orientation data. *IEEE Transactions on Visualization and Computer Graphics 8*, 2 (April–June), 119–128.

MATSUSHITA, Y., OFEK, E., GE, W., TANG, X., AND SHUM, H.-Y. 2006. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 7, 1150–1163.

MEINGAST, M., GEYER, C., AND SASTRY, S. 2005. Geometric models of rolling-shutter cameras. In *6th Int. workshop on Omnidirectional vision, Camera networks, and non-classical cameras*.

MORIMOTO, C., AND CHELLAPPA, R. 1997. Evaluation of image stabilization algorithms. In *DARPA Image Understanding Workshop DARPA97*, 295–302.

MURRAY, R. M., SASTRY, S. S., AND ZEXIANG, L. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA.

NISTER, D. 2003. Preemptive RANSAC for live structure and motion estimation. *IEEE International Conference on Computer Vision 1*, 199–206.

RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2008. Improved seam carving for video retargeting. *ACM Transactions on Graphics 27*, 3 (Aug.), 16:1–16:9.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Transactions on Graphics 25*, 3 (July), 533–540.

THORMÄHLEN, T., AND SEIDEL, H.-P. 2008. 3D-modeling by ortho-image generation from image sequences. *ACM Transactions on Graphics 27*, 3 (Aug.), 86:1–86:5.

TORR, P. H. S., FITZGIBBON, A. W., AND ZISSERMAN, A. 1999. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision 32*, 1, 27–44.

TORRESANI, L., HERTZMANN, A., AND BREGLER, C. 2008. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 5, 878–892.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. Videotrace: Rapid interactive scene modelling from video. *ACM Transactions on Graphics 26*, 3 (July), 86:1–86:5.

WANG, Y.-S., TAI, C.-L., SORKINE, O., AND LEE, T.-Y. 2008. Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics 27*, 5 (Dec.), 118:1–118:8.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2004. Space-time video completion. In *2004 Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, 120–127.

WOLF, L., GUTTMANN, M., AND COHEN-OR, D. 2007. Non-homogeneous content-driven video-retargeting. In *IEEE International Conference on Computer Vision*, 1–6.