Light Field Video Stabilization

Brandon M. Smith Li Zhang Hailin Jin Aseem Agarwala
University of Wisconsin–Madison Adobe Systems Incorporated
http://pages.cs.wisc.edu/~lizhang/projects/lfstable/

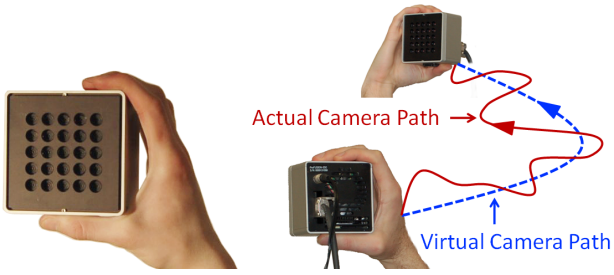
Abstract

We describe a method for producing a smooth, stabilized video from the shaky input of a hand-held light field video camera — specifically, a small camera array. Traditional stabilization techniques dampen shake with 2D warps, and thus have limited ability to stabilize a significantly shaky camera motion through a 3D scene. Other recent stabilization techniques synthesize novel views as they would have been seen along a virtual, smooth 3D camera path, but are limited to static scenes. We show that video camera arrays enable much more powerful video stabilization, since they allow changes in viewpoint for a single time instant. Furthermore, we point out that the straightforward approach to light field video stabilization requires computing structure-from-motion, which can be brittle for typical consumer-level video of general dynamic scenes. We present a more robust approach that avoids input camera path reconstruction. Instead, we employ a spacetime optimization that directly computes a sequence of relative poses between the virtual camera and the camera array, while minimizing acceleration of salient visual features in the virtual image plane. We validate our novel method by comparing it to state-of-the-art stabilization software, such as Apple iMovie and 2d3 SteadyMove Pro, on a number of challenging scenes.

1. Introduction

Videos shot with a hand-held conventional video camera often appear remarkably shaky. Camera shake is one of the biggest components of the large quality difference between home movies and professional videos. Recently, novel camera designs such as light field cameras [5, 31, 26, 14, 30] have been used to significantly improve the solution of a number of computer vision problems, such as reducing image blur and increasing the signal-to-noise ratio, at the cost of additional hardware complexities. In this paper, we show that stabilization of hand-held videos can be greatly improved by leveraging the information in a multi-view video stream, such as the output of a small hand-held video camera array shown in Figure 1(a).

The potential benefit of multiple viewpoints for video stabilization is easy to see. Video stabilization can be viewed as an image-based rendering problem [9]: given a sequence of input images captured along an unsteady camera path, generate a new image sequence that appears as if it were captured along a new and smooth *virtual* camera path, as illustrated in Figure 1(b). This problem can be challenging to solve if only a single video captured by a conventional camcorder is available, due to parallax and occlusion effects between the actual and desired viewpoints. In contrast, a light field video camera provides multiple viewpoints for each time instant that can be used to interpolate or even extrapolate novel viewpoints using view synthesis methods. Therefore, light field cameras allow us to significantly widen the operating range of video stabilization and create superior results.



(a) ProFUSION-25C (b) Stabilization goal

Figure 1. The camera array shown in (a) is capable of capturing 25 synchronized VGA videos at 25 frames per second [4]. We demonstrate how to exploit such a light field camera for high-quality video stabilization using image-based rendering techniques. For efficiency reasons, we used only the four corner cameras and the center camera in this paper.

While this idea is intuitive, we found that the obvious approach to implementing such a method is brittle. A straightforward implementation consists of three major steps: (1) extract image features (*e.g.*, SIFT features [20]) in each frame for each video and match the features across views and over time; (2) estimate the input 3D camera path and a sparse 3D scene geometry by running a structure-from-motion algorithm on the matched features [17]; and (3) generate a desired camera path and synthesize images along the path. This pipeline can work for videos that are friendly to structure-from-motion algorithms. Conditions for success include the presence of enough distinct image features, minimal motion blur and noise, and scenes composed mostly of rigid background with only small dynamic scene elements whose motion can be filtered as outliers. However, many home videos that we would like to sta-

bilize are not captured under such conditions, and thus this pipeline often fails at the structure-from-motion stage.

In this paper, we propose a novel method to exploit multi-view video streams for stabilization that does not require reconstructing the actual 3D camera motion over a long video sequence. Instead, we directly compute the desired virtual camera path relative to the input camera array using a spacetime optimization. The objective of the optimization encourages the salient image features in the input video to move smoothly in the output, stabilized video. The image features are projected into the output virtual viewpoint using their depth information, which is computed from the multi-view images captured at the same time instant; the smoothness of the features in the virtual viewpoint is measured using their acceleration in the image plane. By avoiding structure-from-motion, our method can handle videos shot under conditions that are challenging for the straightforward approach.

We have tested our method on a number of challenging video sequences, and compared our results with state-of-the-art stabilization software such as Apple iMovie '09 [2] and 2d3 SteadyMove Pro [1] that operate on single-view videos, as shown in Section 4.These experiments show that our solution can effectively remove severe camera shake for complex dynamic scenes that include nearby moving targets and large depth variations—issues that challenge existing video stabilization techniques. We present the details of our method in Section 3 after briefly reviewing related work in the next section.

2. Related Work

Video stabilization appeared in the early nineties when it was jointly studied with image mosaicing and registration [11, 16, 23, 25]. At that time, image motions were commonly represented using 2D models (translational, affine, and projective) due to their simplicity and ease of estimation. The latest effort along this line is the work of [15], in which a video is temporally segmented into clips and each clip is processed separately. However, 2D models may lead to gross motion estimation errors and rendering artifacts if the camera translates and the scene is not planar or distant.

To overcome limitations of 2D motion models, researchers proposed techniques based on 3D camera motion estimation [18, 24]. However, these approaches only compensate for camera shake introduced by camera rotations. As a result, videos with jitter in the translational component of the camera trajectory would not be fully stabilized.

Removing translational jitter requires the ability to render an image from a different vantage point. For videos of a static scene obtained from a moving camera, the images at different time instants can be viewed as images taken from different viewpoints. Therefore, one can use image-based rendering techniques (a.k.a. view synthesis) [19] to generate an image at any location with any orientation, as pointed out in [9]; research following this idea includes [13, 7]. The quality of view synthesis depends on the quality of depth estimation (either sparse or dense). For dynamic scenes, accurate depth is difficult to recover from images at different time instants, however. Finally, these view synthesis techniques require reconstructing the 3D trajectory of the input camera path using structure-from-motion. Along with the problems already mentioned, structure-from-motion can also be confounded by degenerate camera motions (e.g., pure camera rotations) and videos whose static content is planar (e.g., a video of a person in front of a wall). Both cases are common in consumer-level video.

Light field cameras, also known as plenoptic cameras [5, 31, 26, 14, 30], capture multiple images of the scene from different viewpoints at the same time instant and thus enable the possibility to apply image-based rendering to dynamic scenes. While light field cameras have different designs, the main difference among them for our purposes is their effective baseline. A hand-held array has more appreciable baseline than other models and therefore better serves the purpose of new view synthesis for light field video stabilization.

Other hardware solutions exist for stabilization: Optical stabilization uses a floating lens element to compensate for pitch and yaw (but no in-plane rotation), and mechanical stabilization uses a moving CCD to compensate for in-plane translation. Our approach handles general camera motion and can achieve more significant stabilization because our baseline is larger than the amount of jitter optical/mechanical devices can compensate for, and we can even extrapolate beyond our baseline.

3. Light Field Video Stabilization

In this section, we define the problem of light field video stabilization and present our solution. Light field video stabilization takes as input a set of shaky[1] videos from multiple viewpoints captured by a handheld camera array, such as the one shown in Figure 1, and produces a stable single-view video as output.

To simplify this problem, we assume the camera array is synchronized, and the internal camera parameters and their relative poses are pre-calibrated and remain fixed for the duration of video capture. This assumption is valid for the ProFUSION-25C camera array used in our experiments. For flexible camera arrays [28, 32], online calibration procedures will need to be developed.

We have explored two approaches to the light field video stabilization problem. The first is a straightforward extension of Buehler *et al*. [9] to multi-view videos and to metric image-based rendering. We have found this extension to be less robust than desired in practice because it requires solving a structure-from-motion problem, which is very difficult for general dy-

[1]Here the word "shake" is used to refer to camera jitter, not motion blur; removing blur is another research problem beyond the scope of this paper.

namic scenes that may include an arbitrary number of moving targets. Our second approach exploits multi-view video streams in a novel way to eliminate the need of recovering the 3D camera path from the input videos. We briefly describe our first approach as a baseline method, which motivates our second approach.

3.1. A Straightforward Approach

The following basic steps outline our straightforward approach to generate a stabilized output video given a set of shaky, multi-view input video streams.

1. **Matching Features** We extract SIFT features [20] for each video in each frame and match the features across views and over time.
2. **Estimating Input Camera Motion** At each time instant, given the matched features across different views we compute their 3D positions using triangulation [17]. We only keep features whose reprojection errors are less than 2 pixels for at least 3 cameras. Using the 3D feature points in each frame, we then estimate the rigid camera motion between each pair of consecutive frames. RANSAC [17] is used to filter out moving targets in the scene. Finally, given the initial estimation of 3D feature positions and the camera motion, we use bundle adjustment [17] to refine the estimation. In the bundle adjustment, we force each set of matched features to share the same unique 3D position.
3. **Generating Virtual Camera Path** From the input camera path, we generate a desired virtual camera path. As stated in [9], this step can be done in several different ways, e.g., by filtering the input path to remove jitter, or by computing a piecewise linear/smooth path that approximates the input path.
4. **New View Synthesis** We use multi-view stereo [29] to compute dense depth maps for each camera at each time instant. Using these depth maps, we synthesize an output image sequence from the desired virtual camera viewpoint.

All four steps above are well studied in computer vision and graphics. However, we have found that building such a system that works robustly in general cases is very difficult. Specifically, the weakest links are *feature matching* and *3D camera motion estimation*. These two steps work best if a scene has many matchable features. This assumption is valid for friendly scenes such as the one shown in Figure 2; however, in many scenes, we often do not have an adequate number of good features to track.

Furthermore, for dynamic scenes, features must be automatically removed from moving targets in order to recover the 3D camera motion relative to rigid portions of the scene. This removal can be done using RANSAC; however, it is only robust if the moving targets occupy a small portion of the scene. It is very challenging to recover the 3D camera motion in cases involving moving targets that are close to the camera. Figure 4(c) shows such a case.

Finally, to perform view synthesis, we need to compute the relative pose between the camera array and the virtual camera at each time instant from their absolute paths in 3D space. As a result, any noise in the estimation of the input 3D camera path will lead to errors in the relative poses between the camera array and the virtual camera. This happens regardless of the smoothness of the virtual camera path itself. With erroneous relative poses, new view synthesis may generate an unstable output video.

3.2. A Novel Robust Approach

We now present a novel approach to light field video stabilization. Our key observation is that, given dense depth maps, we only need to know the relative poses between the virtual camera and the camera array in order to synthesize the output video; the motion of the camera array itself in 3D space is not explicitly required. Based on this observation, our basic idea is to search for a sequence of virtual camera poses with respect to the camera array that will yield a smooth synthesized output video.

To achieve this goal, we cast light field video stabilization as a space-time optimization problem, where the unknown variables are the virtual camera poses at each time instant (with respect to the camera array), and the objective function is the smoothness of the output video. Let F be the number of frames in the input videos over time and $\{\mathbf{R}_f, \mathbf{t}_f\}_{f=1}^{F}$ be the set of rotations and translations of the virtual camera we are searching for. We next describe how to relate these variables to the smoothness of the output video.

3.2.1 Salient Features in Output Videos

We measure the smoothness of the output video based on the motion of salient visual features in the virtual viewpoint. We choose to use intensity edges as salient features because the human visual system is known to be sensitive to edge motion. In addition, an image usually has many more edge features than corner features, therefore addressing the issue of inadequate number of features in the straightforward approach.

Specifically, we first use Canny edge detection [12] to obtain a set of edge pixels as feature points in each input video at each time instant. Since edge points often cannot be matched as conveniently as SIFT features, we compute the 3D locations of these edge feature points by computing dense depth maps for each entire image using multi-view stereo [29]. (The computed depth maps will also be used later in the view synthesis stage.) As a result, for each frame f, we have a set of 3D feature points $\mathcal{X}_f = \{X_{f,j}\}_{j=1}^{N_f}$, where N_f is the number of feature points at frame f. N_f in general varies among different frames.

We remark that $\mathcal{X}_f$ is computed with respect to a reference camera in the array. As a result, given the relative pose sequence $\{\mathbf{R}_f, \mathbf{t}_f\}_{f=1}^F$ for the virtual camera with respect to the same reference camera, we can compute the location of the feature point j in the virtual camera at frame f, $\mathbf{u}_{f,j}$, as

$$\mathbf{u}_{f,j} = \mathbf{K}\left(\mathbf{R}_f X_{f,j} + \mathbf{t}_f\right) \tag{1}$$

where $\mathbf{K}$ is the intrinsic matrix for the virtual camera and "=" holds in a homogeneous coordinate sense. Since all the cameras in the array have roughly the same internal parameters, we use those of the reference camera for the virtual camera.

3.2.2 Minimizing Feature Motion Acceleration

We evaluate the smoothness of feature motion using acceleration. To compute acceleration, we match features in each frame to both the previous frame and the next frame. Again, since edge points often cannot be matched conveniently, we match edge points between neighboring frames by computing optical flows between them. Specifically, if feature point j at frame f is warped to the next frame using optical flow and has a nearest feature point $j_{\texttt{next}}$ that is within a distance threshold Δ, we treat j and $j_{\texttt{next}}$ as a match. The threshold Δ is set to 1 pixel in our experiments. We used Bruhn *et al*. [8] for the optical flow computation, although other state-of-the-art optical flow methods reviewed in [6] may also be used.

For the feature j at frame f, we use $j_{\texttt{prev}}$ and $j_{\texttt{next}}$ to represent its corresponding feature indices in frame $f-1$ and frame $f+1$, respectively. Note that not all features have matches in both previous and next frames. Let ϕ_f be the set of feature points at frame f that have matching features both in frame $f-1$ and in frame $f+1$. We also note that while Bruhn *et al*. [8] is one of the start-of-the-art optical flow methods, outlier matches occasionally still exist in ϕ_f. Another major source of outlier matches we have found occurs near object boundaries where depth changes rapidly—whether an edge pixel is assigned to the foreground or the background is not always consistent in the edge detection process. To address this issue, we do not use edge pixels that have significantly different depth values compared with their matches in adjacent frames. In our experiments, matched edge pixels differing by more than two depth levels were ignored.

Based on the matched features, we use the sum of squared acceleration for each feature point over all frames to measure the smoothness of the output video. We write this as

$$E_{\texttt{acc}}\left(\{\mathbf{R}_f, \mathbf{t}_f\}_{f=1}^F\right) = \sum_{f=2}^{F-1} \sum_{j\in\phi_f} w_{f,j} \cdot \rho\left(\mathbf{u}_{f,j} - \tfrac{1}{2}\left(\mathbf{u}_{f-1,j_{\texttt{prev}}} + \mathbf{u}_{f+1,j_{\texttt{next}}}\right)\right), \tag{2}$$

where the subscript `acc` stands for acceleration, $\mathbf{u}_{f-1,j_{\texttt{prev}}}$, $\mathbf{u}_{f,j}$, and $\mathbf{u}_{f+1,j_{\texttt{next}}}$ are the locations of a triple of matched features $(j_{\texttt{prev}}, j, j_{\texttt{next}})$ in frames $f-1$, f, $f+1$, respectively, computed using Eq. (1) with their corresponding pose parameters and 3D positions, ρ is an error metric function, and $w_{f,j}$ is the weight.

In our current implementation, we use the Blake-Zisserman robust cost function [17] for ρ, where

$$\rho(\delta) = -\log(\exp(-\|\delta\|^2) + \epsilon). \tag{3}$$

This robust function helps to reduce the influence of outliers in feature matching. Without this robust function, the outliers usually deviate dramatically from normal matches and can confound the result, even if they are few in number. We set the weight $w_{f,j}$ to be the magnitude of the image intensity gradient, which forces the objective function to favor points on more dominant edges. Additional weighting schemes are discussed in Section 5.

We must emphasize that our simple feature matching is only used to compute the acceleration of feature motion over a sliding window of three consecutive frames; it is not sought to track features persistently over a long period of time. We have found it to be sufficient for our purposes and robust in all the challenging sequences we have experimented with, as shown in Section 4.

3.2.3 Regularizing Virtual Camera Motion

Eq. (2) itself imposes no restriction on the placement of the virtual camera other than encouraging it to minimize the acceleration of feature point motion. In practice, this lack of restriction on $\mathbf{R}$ and $\mathbf{t}$ sometimes yields a virtual camera motion that deviates significantly from the actual camera path. Indeed, for a long sequence this objective function may favor moving the virtual camera position far away from the scene, which does minimize the acceleration of the feature points (they are very close together and move slowly in the image plane when the virtual camera is distant), but the result is hardly what we want.

To address this issue, we add an additional set of terms to restrict the virtual camera path. Specifically, we penalize translation $\mathbf{t}_f$ from being too large and $\mathbf{R}_f$ from deviating too much from the identity matrix. Mathematically, the penalty is written as

$$E_{\texttt{reg}}\left(\{\mathbf{R}_f, \mathbf{t}_f\}_{f=1}^F\right) = \sum_{f=1}^{F} W_f \cdot \left(\|\mathbf{t}_f\|_\alpha^2 + \|\mathbf{R}_f - \mathbf{I}\|_\beta^2\right) \tag{4}$$

where the subscript `reg` stands for regularization, $W_f = \sum_{j\in\phi_f} w_{f,j}$ is a weight used to balance $E_{\texttt{acc}}$ and $E_{\texttt{reg}}$, $\|\cdot\|_\alpha^2$ and $\|\cdot\|_\beta^2$ are weighted L_2 vector and matrix norms, respectively. For a vector $\mathbf{v}$, we use $\|\mathbf{v}\|_\alpha^2 = \sum_{i=1}^{3} \alpha_i v_i^2$; for a matrix $\mathbf{M}$, we use $\|\mathbf{M}\|_\beta^2 = \sum_{i=1}^{3}\sum_{j=1}^{3} \beta_{ij} m_{ij}^2$. We can choose to weight

each component of the translation and rotation uniformly, or use component-specific weights. The latter is useful, for example, if we want to penalize translations in the z-direction more than the x- and y-directions, or if we want to restrict vertical rotations more than horizontal rotations.

In addition to serving as a regularization term, the weights in Eq. (4), namely α and β, control how far the virtual camera can deviate from the array. One can adjust the weights to balance two conflicting goals: On one hand one wants them to be small to maximize the ability of the cost function to stabilize large camera shake; On the other hand, one wants them to be large so that the virtual camera viewpoints stay close to the camera array, which avoids large cracks at depth discontinuities and large missing border regions in the synthesized result. In our experiments, we set them manually to $\alpha \approx 3$ and $\beta \approx 3$, in particular. We remark that it is possible to set the weights automatically with respect to a single parameter based on the maximally acceptable number of unknown pixels in the result, which might be more intuitive for a user.

3.2.4 The Overall Objective Function

The final cost function that we minimize is then

$$E = E_{\mathtt{acc}} + E_{\mathtt{reg}}, \tag{5}$$

where $E_{\mathtt{acc}}$ and $E_{\mathtt{reg}}$ are defined in Eq. (2) and Eq. (4), respectively. Each rotation in the objective function is parameterized with three variables using the Rodrigues rotation formula [17]. To minimize Eq. (5), we implemented the Levenberg-Marquardt (LM) nonlinear least squares method [27] in Matlab, starting with zero translations and identity rotations. The LM method iterates until the change in the cost function is less than a fraction ϵ of the current cost. In our experiments, ϵ is chosen to be 0.001. The result of this minimization is a set of camera rotations and translations that, when applied to the 3D feature points in the scene, minimize the acceleration of the 2D reprojected versions of these 3D points.

3.2.5 Property of This Objective Function

Now we discuss an interesting property of the objective function in Eq. (5). Unlike the conventional video stabilization approach that stabilizes cameras with respect to static backgrounds, our objective function is a balance between a static background and moving targets. For example, if a person jumps upward in the video, the virtual camera will try to follow her moving direction. The exact balance depends on the salient feature distribution in the image content. This formulation makes our approach robust to complex dynamic scenes where multiple moving targets are close to the camera and occlude most of the static background. It is worth mentioning that if a user prefers to indicate which targets (or background) the camera should be stabilized to, she can guide the solution by manually modulating the weight $w_{f,j}$. However, all of the results in this paper are obtained automatically without any user interaction.

Figure 2. A single frame from a shaky video of a remote control toy moving on top of a highly textured surface. We use this simple, relatively easy video to validate the objective function, Eq. (2), in our novel approach (Section 3.2) and ensure that it produces qualitatively similar results to the straightforward approach (Section 3.1). Please see Section 4.1 for details.

3.3. New View Synthesis

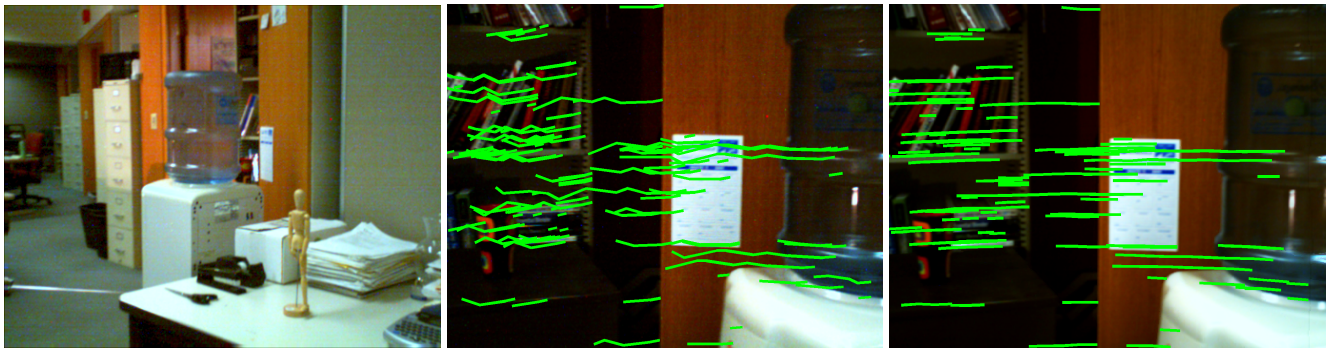
Once we have the virtual camera poses and the depth maps for all input images, we are ready to synthesize the output video. View synthesis is a well studied area in computer graphics and is not the focus of this paper. We describe our current implementation below and give references to more advanced techniques.

At each frame, we first (forward) warp all input depth maps to the virtual viewpoint and fuse the warped depth maps using an approach similar to [22]. We then use the fused depth map to (inverse) warp all images in the camera array to the virtual viewpoint. Pixel intensities are combined by averaging; more advanced view-dependent combinations [10] can also be used to improve results.

Cracks may form at object boundaries if the virtual camera translation is outside the coverage area of cameras in the array. These cracks are filled in using a simple weighted average of surrounding pixel intensities, although more advanced inpainting techniques can also be used [33]. During image warping, missing pixels often exist near image boundaries. A simple approach that we employ to avoid this distracting artifact is to crop each frame just enough to remove these missing pixels. This simple approach is also used in commercial software such as Apple iMovie '09 [2] and 2d3 SteadyMove Pro [1] which we compare against. Recent work on motion inpainting [21] can be used instead to improve the visual quality of stabilized videos.

4. Experimental Results

We now discuss the results of our method on a variety of scenes, as well as comparisons to state-of-the-art methods, namely Apple iMovie '09 [2] and 2d3 SteadyMove Pro [1]. Overall, we find that our method works significantly bet-



(a) (b) (c)

Figure 3. Light field video stabilization on a static scene. (a) A single frame from a shaky video of a static scene with large depth variation and significant camera shake. (b) A frame in the original video overlayed with green lines representing point trajectories traced over time. (c) The corresponding frame in the stabilized video, in which the point trajectories are significantly smoother. Please see Section 4.2 for details. Best viewed electronically.

ter than competing methods on challenging videos (*e.g.*, with high magnitude camera shake, significant scene motion, or dynamic elements very close to the camera), and as well or subtly better on easier cases. **Please see our supplemental video at http://pages.cs.wisc.edu/~lizhang/projects/lfstable/ for a clearer demonstration of our results.**

4.1. Validating the Objective Function Eq. (2)

Our first experiment compares the straightforward approach in Section 3.1 to the novel approach in Section 3.2 for a simple scene for which structure-from-motion successfully computes the 3D input camera path. A selected frame from this video is shown in Figure 2. The goal of this experiment is to validate that minimizing feature acceleration in the image plane generates similar results to the straightforward approach, which explicitly smooths the input 3D camera path. Our results show that the two methods produce qualitatively similar results.

4.2. A Static Scene

Our second experiment is a video of a static scene with significant depth variation (Figure 3(a)). Since the scene is highly non-planar, large camera shake results in parallax between the input and output viewpoints, which is challenging for 2D warping methods. Our result is very successful, iMovie performs poorly, and SteadyMove Pro performs slightly worse than ours. Figure 3(b,c) visualizes the stabilization quality by tracing a subset of feature points through time; the traces are much smoother for our result. Recent research methods [7, 9, 13] have shown excellent results for video stabilization of static scenes, and would likely work well for this sequence. However, none of them will work for dynamic scenes, which we consider next.

4.3. Dynamic Scenes

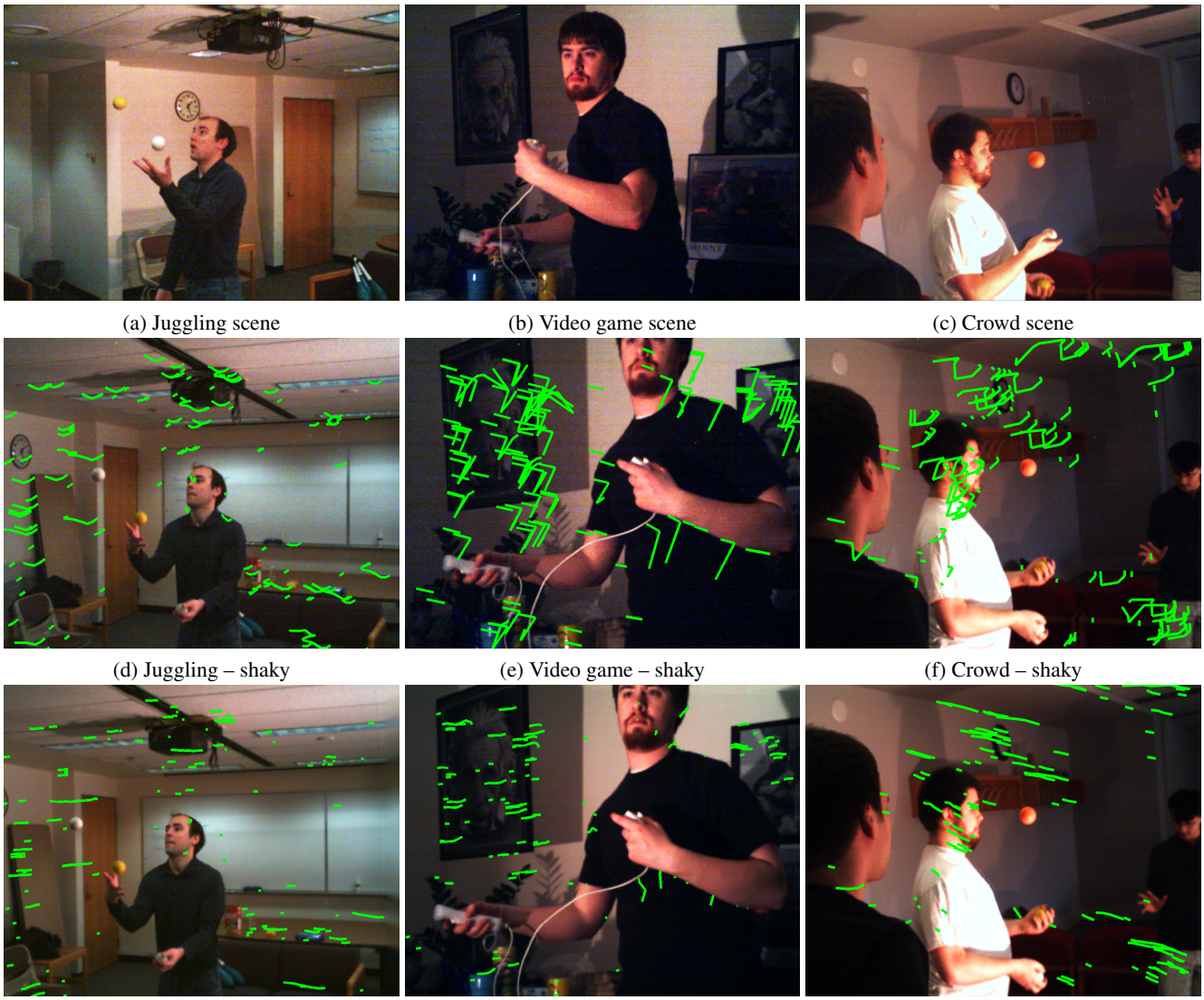
Our first dynamic scene is of a single person juggling (Figure 4 (a)). The camera shake is small for this input, so all three methods (iMovie, SteadyMove Pro, and ours) work well. However, our result is slightly more stable, especially near the ceiling areas. The second dynamic scene, of a person playing a video game with a Wii Remote (Figure 4 (b)), is more challenging; iMovie is not effective, and SteadyMove Pro only works for the first half of the sequence while the camera is still far away from the subject. As the camera moves closer, noticeable shaking is still present in the background. Our result, on the other hand, is more stable.

Our final dynamic scene consists of a crowd of people (Figure 4 (c)). Due to the close proximity of the subjects relative to the camera, a large amount of dynamic activity, and significant camera shake, both iMovie and SteadyMove Pro are not successful on this video. Our method works well on this challenging scene, as shown in Figure 4(i).

4.4. Observations

The performance of our method depends on the success of the constituent standard vision techniques, but is surprisingly resilient to their pitfalls. The optical flow estimation and stereo reconstruction algorithms are worth discussing in this context. In the presence of large viewpoint shifts, poor lighting conditions, weak scene texture, *etc*. the optical flow estimate may not be accurate. One consequence of inaccurate flow in our application is that few feature matches will be found within affected regions. This phenomenon sometimes occurs on fast moving objects (*e.g.*, the arms in Figure 4), in which case the corresponding regions will contribute less to the overall stabilization. The resulting videos in this case, however, are often visually acceptable: fast moving foreground targets remain moving fast. Similarly, not all failure cases in the stereo algorithm are noticeable. For example, it is especially difficult to recover accurate depth in uniform regions of the image. However, problems in these regions are easily disguised in the final rendering precisely because they are uniform.

In all our experiments, the virtual viewpoint is almost always within the camera array region, and never farther than



(a) Juggling scene (b) Video game scene (c) Crowd scene

(d) Juggling – shaky (e) Video game – shaky (f) Crowd – shaky

(g) Juggling – stabilized (h) Video game – stabilized (i) Crowd – stabilized

Figure 4. Light field video stabilization on dynamic scenes. Each column shows one dynamic scene in our experiments. The top row shows a frame in the original videos. The middle row shows a frame in the original video overlayed with green lines representing point trajectories traced over time. The bottom row shows a frame in the stabilized video, in which the point trajectories are significantly smoother. These examples demonstrate that our method is able to handle severe camera shake for complex dynamic scenes with large depth variation and nearby moving targets. Please see Section 4.3 for details. Best viewed electronically.

twice the dimension of the array, due to our α and β weights. As a result, our new view synthesis step only has to fill in small gaps at depth discontinuities.

We have found that the accuracy of depth estimation does not improve significantly with more than five cameras (which is the number we used). However, despite using fewer cameras, one weakness of our current method is its computation cost. The depth maps are the main bottleneck, requiring about 10 minutes of computation time for each frame at 480×360 resolution, followed by the optical flow fields, which require 2-3 minutes for each pair of consecutive frames. The run time of the LM optimization for Eq. (5) depends on the number of frames and the number of feature points. On a typical 10-second video with about 1000 matched feature points per frame, it completes in under one hour. However, we believe this time can be reduced to several minutes if the optimization is implemented in C++.

5. Future Work

There are a number of areas for future work. In our current implementation, the weight $w_{f,j}$ is a scalar. Ideally, the weight should account for the directionality of the edges, and therefore take the form of an inverse covariance matrix which measures the uncertainty of feature points. This type of uncer-

tainty has been used in structure-from-motion research [17].

In our current work, we have not considered the motion blur issue in input videos. We sidestep this issue by taking videos under good lighting conditions. In the future, we would be interested in studying how motion deblurring may benefit from having a camera array.

If the hand motion is too violent, the input videos do not capture any image content that we wish to see. As a result, the simple cropping will not work to create the output video. To address this issue, we need either better dynamic scene reconstruction methods or a hand-held array with wide field-of-view lenses.

Finally, another area for future exploration is to evaluate the range of camera baselines that still allow high-quality video stabilization. Smaller baselines will allow smaller cameras, but limit the range of viewpoints that can be synthesized.

6. Conclusion

Camera shake is a major detriment to the quality of consumer-level video, and current stabilization techniques can only go so far in improving the quality of camera motion in hand-held shots. We have shown that a multi-viewpoint device has the capability to dramatically improve the appearance of consumer-level video. The design of camera arrays and other forms of light field cameras are advancing rapidly, both in research and commercially. For example, FujiFilm recently announced the consumer-level FinePix Real3D stereo camera [3]. While the size of such a device for video stabilization will be larger than a single-viewpoint camera, since a reasonable baseline is necessary, the improvement in video quality as well as the other benefits of capturing light fields may well make small video camera arrays a significant design point in the near future.

7. Acknowledgements

We are grateful to Adobe Systems Incorporated for supporting this work, Tuo Wang for contributing to the initial implementation of the straightforward approach described in Section 3.1, Stefan Roth for generously sharing his implementation of the 2D-CLG optical flow estimation method by Bruhn *et al*. [8], and the anonymous reviewers for their constructive feedback.

References

[1] 2d3 SteadyMove Pro, http://www.2d3.com/product/?v=5.

[2] Apple iMovie '09, http://www.apple.com/ilife/imovie.

[3] Fujifilm finepix real3d system, http://www.fujifilm.com/photokina2008/list/#h2-15.

[4] ViewPLUS Inc. ProFUSION 25 5x5 camera array system, http://www.viewplus.co.jp/products/profusion25/index-e.html.

[5] E. Adelson and J. Wang. Single lens stereo with a plenoptic camera. In *PAMI*, 1992.

[6] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, 2007.

[7] P. Bhat, C. L. Zitnick, N. Snavely, and A. Agarwala. Using photographs to enhance videos of a static scene. In *Proceedings of Eurographics Symposium on Rendering*, 2007.

[8] A. Bruhn, J. Weickert, and C. Schnoerr. Lucas/kanade meets horn/schunck: Combining local and global ptic flow methods. In *IJCV*, 2005.

[9] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *CVPR*, 2001.

[10] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001.

[11] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. In *DARPA Image Understanding Workshop, Monterrey*, 1994.

[12] J. F. Canny. A computational approach to edge detection. *PAMI*, 1986.

[13] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *IJCV*, 2005.

[14] T. Georgiev, K. C. Zheng, B. Curless, D. Salesin, S. Nayar, and C. Intwala. Spatio-angular resolution tradeoffs in integral photography. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*.

[15] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2008.

[16] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *DARPA Image Understanding Workshop, Monterrey*, 1994.

[17] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition.

[18] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *CVPR*, 1994.

[19] S. B. Kang and H.-Y. Shum. A review of image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing*, 2002.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2004.

[21] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. In *PAMI*, 2006.

[22] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, D. N. R. Yang, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, 2007.

[23] C. Morimoto and R. Chellappa. Automatic digital image stabilization. In *ICPR*, 1996.

[24] C. Morimoto and R. Chellappa. Fast 3d stabilization and mosaic construction. In *CVPR*, 1997.

[25] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998.

[26] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005-02, Stanford University Computer Science, 2005.

[27] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Science+Business Media, LLC, 2nd edition.

[28] Y. Nomura, L. Zhang, and S. Nayar. Scene Collages and Flexible Camera Arrays. In *Proceedings of Eurographics Symposium on Rendering*, June 2007.

[29] B. M. Smith, L. Zhang, and H. Jin. Stereo matching with nonparametric smoothness priors in feature space. In *CVPR*, 2009.

[30] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin. Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Trans. Graph.*, 2007.

[31] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Transactions on Graphics*, 2005.

[32] C. Zhang and T. Chen. A self-reconfigurable camera array. In *Proceedings of Eurographics Symposium on Rendering*, 2004.

[33] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH*, 2004.