Spatially and Temporally Optimized Video Stabilization

Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu and Tong-Yee Lee

Abstract—Properly handling parallax is important for video stabilization. Existing methods that achieve the aim require either 3D reconstruction or long feature trajectories to enforce the subspace or epipolar geometry constraints. In this paper, we present a robust and efficient technique that works on general videos. It achieves high-quality camera motion on videos where 3D reconstruction is difficult or long feature trajectories are not available. We represent each trajectory as a Bézier curve and maintain the spatial relations between trajectories by preserving the original offsets of neighboring curves. Our technique formulates stabilization as a spatial-temporal optimization problem that finds smooth feature trajectories and avoids visual distortion. The Bézier representation enables strong smoothness of each feature trajectory and reduces the number of variables in the optimization problem. We also stabilize videos in a streaming fashion to achieve scalability. The experiments show that our technique achieves high-quality camera motion on a variety of challenging videos that are difficult for existing methods.

Index Terms—Video stabilization, Warping, Optimization, Bézier curve

1 Introduction

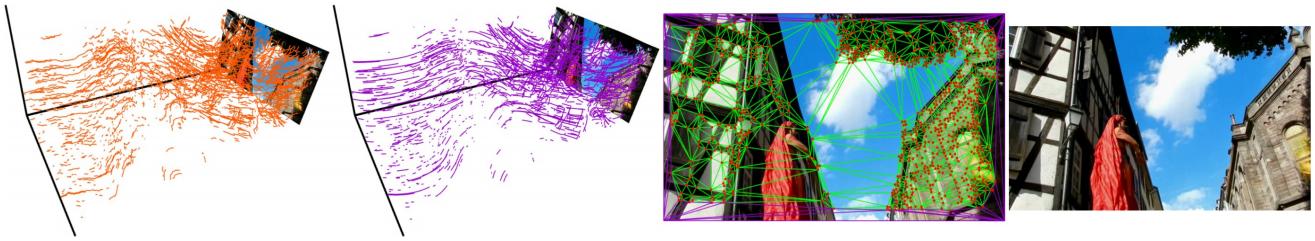
Most amateur videos are captured using hand-held cameras. They are often very shaky and difficult to watch. Therefore, video stabilization techniques have been developed to smooth shaky camera motion. Most existing methods work in 2D and are efficient and robust. However, these methods use homography, which cannot model parallax induced by a moving camera imaging a 3D scene, so they cannot aggressively stabilize shaky camera motions.

To handle parallax well, 3D video stabilization methods apply 3D reconstruction from structure from motion (SFM) [1] or depth acquisition from sensors [2]. However, SFM solves a highly non-linear optimization problem to reconstruct a 3D scene, which is computationally expensive and is often brittle. Depth acquisition is limited to depth cameras and thus is not applicable to general videos. Closely related to our technique, where scene modeling is not required, Liu *et al.* [3] developed a method that makes use of "eigen-trajectories", a middle-level representation between 2D and 3D, and smoothes camera motion in the subspace of feature trajectories. However, the method requires a certain number of long feature trajectories. It potentially breaks down when such trajectories are not enough, which usually occurs in videos suffering from serious blurriness. Goldstein and Fattal developed a method that uses a projective scene reconstruction [4] to handle parallax without 3D reconstruction. Their method, however, cannot work well on videos with strong occlusions and non-Lambertain surfaces.

In this paper, we introduce a video stabilization method that can handle parallax and is robust to poor motion estimation results. Our key insight is that the tracked features are spatially and temporally correlated. Solving a video stabilization problem in separate passes, say, computing spatial relations (such as planar assumption, SFM and subspace extraction) followed by smoothing the reduced parameters, cannot make use of all the feature trajectories. Accordingly, we formulate stabilization as a spatial-temporal optimization problem that considers both smoothing and spatial rigidity simultaneously to handle parallax. This formulation enables our system to utilize both long and short feature trajectories and approach global optimal solutions when stabilizing videos. To achieve high performance, we represent each smoothed feature trajectory using a Bézier curve when minimizing the objective function. Since the unknowns are reduced from feature positions to only curve control points, heavy computational cost and memory consumption can be prevented. By setting a lower degree to Bézier curves, this representation also achieves strong stabilization because each smoothed feature trajectory is interpolated from the optimized control points. Although the Bézier curve idea may be not new [3], [5], [6], we embed this reduced model in the spatial-temporal optimization problem and enable high-performance video stabilization that creates high-quality camera motion.

During the optimization, we explicitly constrain the

- *Y. S. Wang is with the Department of Computer Science, National Chiao-Tung University, Taiwan E-mail: yushuen@cs.nctu.edu.tw*
- *F. Liu is with the Department of Computer Science, Portland State University, OR, 97207-0751 E-mail: fliu@cs.pdx.edu*
- *P. S Hsu and T. Y. Lee is with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan E-mail: tonylee@mail.ncku.edu.tw, prindrock@gmail.com*



(a) (b) (c) (d)

Fig. 1. The original (a) and the stabilized (b) feature trajectories are shown in the spatial-temporal coordinate system. We smooth feature trajectories while retaining the neighboring feature offsets (c) in each frame to handle parallax. We then warp each video frame based on the stabilized features and crop the maximal overlapping area to produce the result (d).

spatial offsets of neighboring features in each frame when smoothing their motions. Each local region is enforced to undergo a rigid transformation because the stabilization will typically not change the viewpoint too much and thus will not distort the interior content, unless the video suffers from rolling shutter artifacts. In addition, the Bézier representation enables our method to directly smooth feature trajectories and thus, our system can handle parallax well. Although the number of variables in each trajectory has already reduced, the computational cost of our system would still increase as more and more trajectories are added when the video becomes long. We therefore optimize the feature trajectories within the moving time window in a streaming manner while constraining the smoothness in the overlapping frames to achieve scalability. Once all trajectories are optimized, we consider the smoothed features as positional constraints and apply the content preserving methods to warp video frames sequentially. Finally, a cropping window is determined to capture the maximal area containing only valid information to produce the stabilizing result.

We demonstrate the effectiveness of our technique by comparing it to state-of-the-art techniques [3], [4], [7] on a variety of videos containing large parallax, strong occlusions, and rolling shutter artifacts. Experiments show that our method can handle parallax and is robust to challenging examples. Since the L1-Optimization method provided by Youtube [7] is the most robust technique among these recent works, we also evaluate the performance of our framework by conducting a user study with 74 participants, which showed strong preference of our method over [7].

2 Related Work

Video stabilization techniques have been developed to smooth shaky camera motions. Two-dimensional stabilization methods estimate a homography sequence from the input video, dampen this sequence, and warp video frames based on the smoothed sequence [8], [9], [10], [11], [12], [13]. A recent method achieves aggressive smoothing using L1 optimization [7], in which the gradient of the camera motion at each frame can be strongly suppressed, to produce very high quality results. However, the use of 2D motion models still limits its ability to handle parallax.

Three-dimensional stabilization methods use SFM to estimate 3D camera motion and scene structure, followed by smoothing the camera motion. These methods then render a new video according to the smoothed camera path using image-based rendering [14], [15], homography approximation [16], or content-preserving warp [1], [17]. However, SFM is a difficult and time-consuming computer vision problem. It often fails when the video 1) lacks parallax, 2) contains camera zooming, or 3) suffers from rolling shutter artifacts. Therefore, these three-dimensional methods are not practical in many examples.

Liu *et al.*'s method [3] smoothes feature trajectories with subspace constraints to achieve robustness and efficiency. The tracked feature trajectories are factorized into low dimensional key bases and reconstruction coefficients. Their method then smoothes the bases followed by trajectory reconstruction to stabilize videos. However, the streaming factorization requires long feature trajectories to cover the entire factorization window, which do not usually appear in videos containing dynamic backgrounds and large foreground motions. Goldstein *et al.* [4] applied the projective reconstruction to account for simple geometric relations between points and epipolar lines, which raises the robustness of scene modeling. They also derived a time-view point reprojection to stabilize trajectories of moving objects so that the defined warping constraints can be spread more uniformly across the frame to capture the scene shape. However, the projective reconstruction relies on relative positions of corresponding features between images, so the method fails at examples that have strong occlusions and non-Lambertian surfaces. Liu *et al.* [2] stabilized videos that are captured by depth cameras, in which the relative motions of neighboring content can be estimated and then preserved when smoothing the

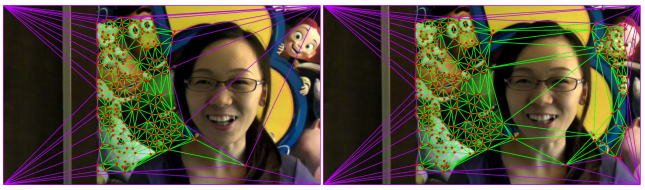


Fig. 2. We show the features and the neighboring relations using red dots and green triangles. Clearly, the spatial preservation based on only green triangles potentially causes instability artifacts due to the sudden appearance of right features. We thus further retain the relative offsets of four corners to features (purple triangles) in each frame to prevent this problem.

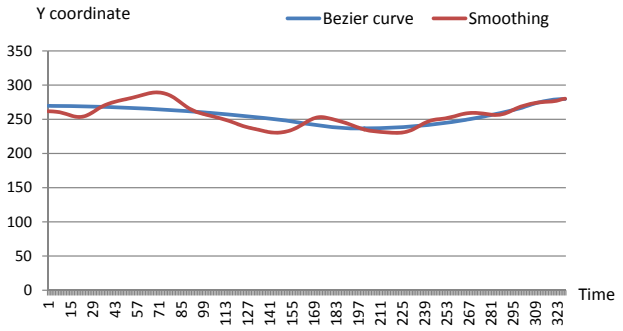


Fig. 3. We compare the smoothness of stabilized feature trajectories using acceleration minimization and Bézier curve fitting.

camera motion. As this method requires depth acquisition, it cannot work for regular videos. In contrast, our method, which smoothes each feature motion while retaining the offsets to neighboring features in the spatial-temporal coordinate system, achieves high-quality stabilization without requiring long feature trajectories, scene reconstruction, or depth acquisition.

3 Feature Trajectory Smoothing

We stabilize videos by smoothing the trajectories that are integrated from the KLT features [18]. Similar to the subspace method [3], the tracked features on moving objects are ruled out in advance using the epipolar constraint [19]. We then smooth the remaining feature trajectories while retaining their spatial relations to stabilize videos. Note that this outlier rejection is not necessarily perfect because the remaining foreground features have nearly static motions and those regions can be treated as backgrounds.

Let the i^{th} trajectory be $\mathbf{P}_i = \{\mathbf{p}_i^m, \mathbf{p}_i^{m+1}, ..., \mathbf{p}_i^n\}$, where $\mathbf{p}_i = (x_i, y_i) \in \mathcal{R}^2$ is the feature position, and m and n are the start and the end frames of $\mathbf{P}_i$, respectively. Our goal is to solve an optimization problem that can minimize the acceleration of $\mathbf{P}_i$ in each frame while constraining the offsets of neighboring trajectories to be consistent within the input video. To achieve high performance, we represent each smoothed trajectory using a Bézier curve and reduce the unknown variables from all feature positions to Bézier control points. This reduced model also achieves strong stabilization because the smoothed feature positions are interpolated from the control points. We show the details of our technique in the following subsections.

3.1 Objective function

Spatial rigidity preservation. We compute the neighbor relations between features in each frame using the Delaunay triangulation. To achieve spatial rigidity when stabilizing a video, we enforce each triangle to undergo a rigid transformation. This constraint works well in most videos that are free from rolling shutter artifacts because the change to the camera motion is typically small in video stabilization. That is, triangles are allowed to move and rotate but their sizes and shapes should be retained. Since features are not uniformly distributed, they may be very different in consecutive frames (see Figure 2) and thus, preserving only relative positions of features is not sufficient to stabilize the regions that have no features in the current frame but have some previously. We thus add four corners of each frame when computing the Delaunay triangulation and introduce the energy term

$$\Omega_c = \sum_t \sum_f \sum_{\{i,j\} \in \mathcal{E}(f)} \left| (\mathbf{r'}_i^t - \mathbf{r'}_j^t) - \mathbf{R}^t (\mathbf{r}_i^t - \mathbf{r}_j^t) \right|^2, \quad (1)$$

where t is the frame index, $\mathcal{E}(f)$ denotes the edges of triangle f, $\mathbf{r} = \{\mathbf{c}, \mathbf{p}\}$, $\mathbf{c}$ is the original corner position, $\mathbf{R}$ is the unknown rotation matrix and $\mathbf{r}'$ is the smoothed version of $\mathbf{r}$.

Original camera motion approximation. Clearly, video stabilization cannot deviate the smoothed camera motion from its original trajectory too much because the motion usually has important meanings. This requirement also reduces the content loss because video frames are transformed during the stabilization and only the overlapped areas throughout the entire video will remain. Therefore, let $\mathbf{p}'$ be the smoothed feature position, we give the energy term:

$$\Omega_p = \sum_{\mathbf{P}_i} \sum_t \left| \mathbf{p'}_i^t - \mathbf{p}_i^t \right|^2. \quad (2)$$

Individual trajectory smoothing. To achieve video stabilization, we can minimize the acceleration of each feature motion using the energy term

$$\Omega_s = \sum_{\mathbf{P}_i} \sum_t \left| \mathbf{p'}_i^{t-1} - 2\mathbf{p'}_i^t + \mathbf{p'}_i^{t+1} \right|^2. \quad (3)$$

However, minimizing the objective function $\Omega_c + \Omega_p + \Omega_s$ requires solving all feature positions in one step,

where the number of unknown variables is tremendous. This requirement would lead to heavy computational cost and large memory consumption when a long and high-resolution video is processed. In addition, as pointed out by [7], least squares optimization satisfies Equation 3 on average, which would result in small but non-zero gradients between consecutive frames. It means that the optimized video is only less shaky but still not stable (see Figure 3 and our supplemental video *Bezier.mp4*). To solve these two problems, we represent the smoothed trajectory using a Bézier curve so that feature motions can be strongly smooth and the unknown variables are greatly reduced. Specifically, each node on a trajectory is written as $\mathbf{p}_i^t = \sum_{\ell=0}^{d} \omega_i^{t,\ell} \mathbf{q}_i^\ell$, where $\omega^{t,\ell} = \binom{d}{\ell}(1-r)^{d-\ell} r^\ell$, d is the degree of Bézier curve, $\mathbf{q}_i$ are the control points and $r = \frac{t-n}{m-n}$ is the interpolation coefficient. Thus, our goal becomes approximating each jittering feature trajectory using a Bézier curve while retaining the offsets of neighboring features in each frame. Formally, we rewrite the unknown feature position $\mathbf{p'}_i^t$ in Equations 1 and 2 to $\sum_{\ell=0}^{d} \omega_i^{t,\ell} \mathbf{q}_i^\ell$ and neglect Equation 3 when solving the optimization.

Generally, a larger value of d can retain more content after cropping while a smaller value enjoys stronger camera stabilization due to the different degrees of Bézier curve fitting. For generality, we set $d = 2$ in all our experimental results, except the supplemental video *DegreeComparison.mp4* that we used to illustrate the effect of different degrees.

Reliability constraints. When video frames containing no features, there is no way to determine their smoothed positions. The minimization of Equations 1 and 2 subject to the Bézeir constraint is not sufficient to stabilize such jittering videos because the unknowns are not correlated. Therefore, we slightly enforce the corners of these extremely difficult frames to have smooth transitions between consecutive time steps, so that our system can be robust to all challenging examples. Specifically, we add the energy term

$$\Omega_r = \sum_{t \in \mathbf{U}} \sum_{i=0}^{3} \left| \mathbf{c}_i^{t-1'} - 2\mathbf{c}_i^{t'} + \mathbf{c}_i^{t+1'} \right|^2, \tag{4}$$

where i is the corner index and $\mathbf{U}$ is the set of frames that have no features.

3.2 Optimization

By integrating the mentioned energy terms, we search for the control points of Bézier curves that can minimize the objective function

$$\Omega = \mathrm{w}_c \Omega_c + \mathrm{w}_p \Omega_p + \mathrm{w}_r \Omega_r \tag{5}$$

to stabilize videos, where w_c, w_p and w_r are the weighting factors ($\mathrm{w}_c = 10$, $\mathrm{w}_p = 1$ and $\mathrm{w}_r = 0.01$ in all our experiments). The optimization is transformed to a linear system $\mathbf{Ax} = \mathbf{b}$ and we solve for the unknown variables $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. Thanks to the Bézier curve representation, the size of $\mathbf{x}$ is linear to the number of control points. The system will be greatly reduced when $\mathbf{A}^T$ is multiplied to its both sides. We solve the objective function by iteratively updating the unknown corners, the control points of Bézier curves and the rotation matrixes because they are correlated. Specifically, we first set $\mathbf{R} = \mathbf{I}$ and compute $\mathbf{q}$ and $\mathbf{c}$ by solving a linear system. $\mathbf{R}$ is then determined by first computing a homography transformation between the original and the smoothed features and corners, and followed by using the singular value decomposition to eliminate the shear and the scale components. Please refer to [20] for more optimization details. Once the control points of each Bézier curve are obtained, we compute the stabilized feature positions at each frame using $\hat{\mathbf{p}}_i^t = \sum_{\ell=0}^{d} \omega_i^{t,\ell} \mathbf{q}_i^\ell$.

It deserves noting that the Bézier and the rigidity constraints conflict when handling videos that have parallax. Expecting each region to undergo an exact rigid transformation cannot stabilize the video since features with different depth values have different shaking magnitudes. When this conflict happens, our system can still smooth feature motions to handle parallax because our Bézier representation is a hard constraint but Ω_c is a soft one. While the triangles may be slightly distorted, each stabilized feature trajectory is definitely smooth because it is interpolated from the optimized Bézier control points.

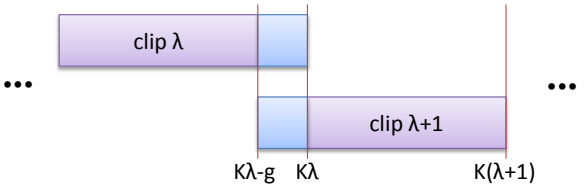


Fig. 4. We streamingly stabilize a long video to achieve scalability. The feature positions in the overlapping frames are smoothed to prevent the discontinuity artifacts.

3.3 Implementation details.

Scalability. We solve for a subset of video frames instead of the whole video cube at a time to prevent the scalability problem, even though each feature trajectory is already transformed into a reduced model. This streaming strategy is very helpful to our system because the captured videos are usually long and the computational cost would increase rapidly. Therefore, we segment the video into shorter clips, with each clip containing k frames, and with g frames overlapping between consecutive clips ($k = 200$ and $g = 50$ in all our experiments). During the optimization of the λ^{th} clip, we solve for the smoothed feature positions from

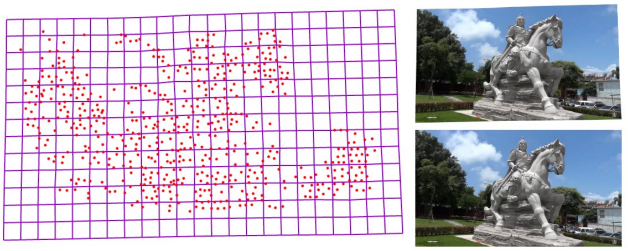


Fig. 5. (left) The warped quad mesh and the smoothed features. (top right) The warped frame. (bottom right) The original frame. We warp each video frame by enforcing features to be located at the smoothed positions so that some grid lines bend although each local region is transformed homographically.

frame $k\times\lambda-g$ to frame $k\times(\lambda+1)$ (see Figure 4). Hence, a long trajectory will be approximated using several Bézier curves. We thus set positional constraints in the overlapping frames to prevent the discontinuity artifacts caused by our streaming approach. Specifically, when stabilizing the λ^{th} ($\lambda > 0$) clip, we replace $\mathbf{p}_i^t$ by $\hat{\mathbf{p}}_i^t$ in Equation 2 if $k \times \lambda - g \leq t \leq k \times \lambda$, where $\hat{\mathbf{p}}$ is the stabilized feature position obtained in the $(\lambda-1)^{th}$ clip. Since some features are computed twice in consecutive clips, they may have very similar but different positions. We determine their smoothed positions by linear blending.

Bézier curve fitting. Apparently, a Bézier curve may not well fit the shape of a long and twisting trajectory. Although our streaming implementation that solves a subset of video frames at a time can ease the problem, it potentially happens to some complicated camera motions and results in waving artifacts. Therefore, before the minimization of Ω shown in Equation 5, we compute a best fit Bézier curve for each trajectory without considering spatial rigidity. We then determine the fitting error by computing the largest distance between the pair of the original and the smoothed feature positions. For the trajectory having a fitting error larger than μ pixels, we partition it into two sub-trajectories at that position. This fitting detection repeats until each partitioned trajectory can be well fit by a Bézier curve. We set $\mu = \sqrt{\mathcal{A}}$ in our system, where $\mathcal{A}$ is the averaged triangle area.

4 Video Warping

We warp video frames according to the optimized feature positions to determine the result. We render each video frame based on the deformed grid mesh, rather than the triangular one shown in Figure 2 because the appearing and the missing features are less reliable (pointed out by Liu et al. [1]), and additional stability constraints are needed.

Specifically, we represent each video frame using a regular grid mesh $\mathbf{M}^t = \{\mathbf{V}^t, \mathbf{E}^t\}$ with each quad covering roughly 10×10 pixels, where $\mathbf{v}_i^t \in \mathbf{V}^t$ and $\{i,j\} \in \mathbf{E}$ are the grid vertices and edges, respectively. During the frame warping, we expect each quad to undergo a homography transformation $\mathbf{H}^t$ while constraining each feature to locate at the smoothed position according to the feature reliability. The transformation $\mathbf{H}^t$ can be considered as a 2D stabilization and it is computed according to the original and the smoothed features and corners. We use this homography to stabilize the regions with no features, including homogeneous regions and moving foregrounds. Although the transformation $\mathbf{H}^t$ treats each moving foreground as a plane, our system does not introduce noticeable visual artifacts because the foreground motions are usually much larger than the jittering caused by depths.

Due to the discrete representation, we constrain each feature position using the linear combination of surrounding vertices when warping a video frame. We also compute the reliability value of each feature using $\gamma_i^t = \min(1, \frac{t-m}{10}, \frac{n-t}{10})$, which is mainly based on distances to end points of a trajectory. By considering the above criteria, we minimize the objective function $D_h^t + D_p^t$ to warp each video frame, where

$$D_h^t = \sum_{\{i,j\}\in\mathbf{E}^t} \left|(\hat{\mathbf{v}}_i^t - \hat{\mathbf{v}}_j^t) - \mathbf{H}^t(\mathbf{v}_i^t - \mathbf{v}_j^t)\right|^2, \quad (6)$$

$$D_p^t = \sum_{\hat{\mathbf{p}}_i^t} \gamma_i^t \left|\textstyle\sum_{j\in\mathcal{L}_i^t} \delta_j^t \hat{\mathbf{v}}_j^t - \hat{\mathbf{p}}_i^t\right|^2, \quad (7)$$

$\mathcal{L}_i^t$ are the indexes of vertices surrounding $\hat{\mathbf{p}}_i^t$ and δ is the combination weight. Note that solving the mentioned objective function does not degenerate our algorithm to a 2D stabilization even though the energy term D_h^t treats each video frame as a plane, because the features are still constrained at the smoothed positions by D_p^t. As Figure 5 shows, the quad deformations are similar but not identical to ensure all features moving smoothly. Once all meshes are transformed to achieve stabilization, we linearly interpolate pixel colors in each quad and determine the maximal window that covers only valid information to produce the stabilized video.

5 Results and Discussions

We implemented and tested our algorithm on a desktop PC with Core i7 3.0 GHz CPU. We use the direct solver with Cholesky factorization to minimize Ω in Equation 5 and use the conjugate gradient method to warp each video frame because the frame can be pre-warped using the homography transformation and used as an initial guess for fast convergence. In general, the performance of our algorithm mainly depends on the number of feature trajectories. The trajectory lengths do not affect the performance too much because we have already transformed each trajectory into a Bézier curve and only three control points are solved to obtain its smoothed version. Table

Fig. 7. We show the warped video frames (top row) and the cropped results (bottom row) that are determined by our system.

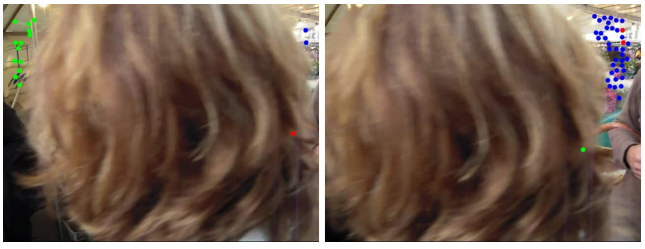


Fig. 6. From left to right are the consecutive frames. In this example, the appearing, existing and the missing features are shown in blue, red and green, respectively. The subspace [3] and the epipolar [4] methods fail at this example due to short feature trajectories. Our approach successfully stabilizes the video since the global optimization is solved.

resolution	# frame	# trajectory	smoothing (sec.)	fps
640 × 360	609	6366	6.791	90
720 × 480	389	18670	10.81	36
720 × 480	331	12319	5.931	56
1280 × 720	775	42229	26.06	30

TABLE 1
Video information and the corresponding timing statistics. Note that the timing of KLT feature extraction is not included.

1 shows the timing statistics. Our method uses the Voodoo camera tracker [21] to track KLT features. While the KLT implementation used in our method is not real-time, feature trajectory estimation is a common step in existing video stabilization methods. Our method achieves real-time performance after we preprocess the input video to estimate the KLT feature trajectories.

We have tested diverse examples containing dynamic backgrounds, strong occlusions and large parallax to demonstrate the effectiveness of our system. All the results are generated automatically using the default parameters. Many of the examples involve large foreground objects occupying almost the whole video frame, making the stabilization rather challenging. While previous methods [3], [4], [7] all fail in such difficult examples, our technique successfully stabilizes the videos with no visual artifacts. To achieve a fair comparison, we also tested the videos chosen from previous methods to demonstrate that our system not only works well in our selected examples but also in theirs. Please refer to Figures 1, 5, 6, 7, the accompanying videos and our *project webpage*[1] for more results and comparisons, especially as the temporal stabilities are difficult to visualize and appreciate in still images.

Robustness. Our system can leverage very short feature trajectories because we solve a global optimization to achieve video stabilization. When sometimes the frame number of a trajectory $n-m+1$ may be smaller than the number of Bézier control points $d+1$, we represent that trajectory using a Bézier curve with degree $n-m$ in our implementation. Although our stabilizing result will be equal to the original video if all trajectories are two frames long, this extreme case rarely appears. Moreover, our reliability constraint allows some video frames to have even no features because constraining corner vertices to have smooth transitions ensures all unknowns being correlated when solving the optimization. Although the frames with no features may not be effectively stabilized in this scenario, our system can produce the result for such a challenging example, which is much more stable than its original version (see Figure 8 and our supplemental video *limitation.mp4*).

Comparisons. We have compared our results with those stabilized by state-of-the-art techniques, including L1-optimization [7], the subspace [3] and the epipolar [4] methods. The 3D stabilization [1] is not included because the latter two approaches [3], [4] can also handle parallax and have been demonstrated to be more robust. We show that our method can work well on videos that are shown in the pa-

1. http://people.cs.nctu.edu.tw/~yushuen/VideoStabilization/

Fig. 8. Although our algorithm is robust to all challenging examples, the stabilization is not effective if there are no background features in some frames.

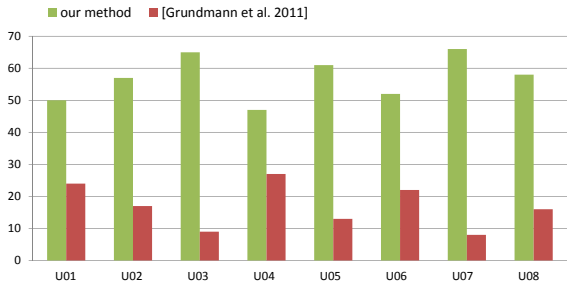


Fig. 9. The bar chart shows the participants' preferences among examples from U01 to U08.

per of [3], [4], [7] in our supplemental materials (*CompToLiu.mp4*, *CompToGrundmann.mp4*, and *CompToGoldstein.mp4*). Meanwhile, we focused on the selected challenging examples containing large parallax, strong occlusions and fast camera motions to demonstrate the effectiveness of our technique. Since the methods of [3], [4] fail at producing many results among these examples, we thus mainly compare our technique to [7] in our demo video *Main.mp4*. For those examples that the subspace and epipolar methods [3], [4] can stabilize, we show their results as well as ours in *challenging.mp4*.

The L1-optimization results we used for the comparison are from the YouTube Video Stabilizer[2], which is implemented based on Grundmann et al.'s technique [7]. This strategy strongly smoothes camera paths by solving inequality constraints. Hence, it achieves only 10–20 fps in low resolution videos even though only a transformation of each frame is computed. In addition, although their system is robust to challenging examples, its 2D nature limits the ability to handle parallax, in which case global shaking potentially appears after stabilization, as demonstrated in the accompanying video. In contrast, we smooth feature trajectories in the spatial-temporal coordinate system to coherently transform features with different depth values. Although we solve all feature trajectories simultaneously to handle parallax, our system achieves higher stabilization performance than [7].

User study. Based on the selected challenging examples, we conducted a user study with 74 participants coming from diverse backgrounds and ages to evaluate our method. The participants were presented with an original video and two stabilizing results side by side, and they were asked to answer which result they prefer. During the study, the questions are presented in a random order to avoid bias. The results are automatically stabilized using [7] and our technique. The methods of [3], [4] were not included in the study because they fail to produce results for the examples due to the lack of long feature trajectories.

Figure 9 shows the summary of the obtained results, which shows the significant preference of our method. Overall, it was favored over the method of [7] in 77% of 616 comparisons. In particular, the participants prefer our results that contain large parallax (see Figure 9, *U03.mp4, U07.mp4 and U08.mp4* in our supplemental material) due to the less noticeable jittering artifacts. This obtained statistic is not unexpected, because our stabilization system can handle parallax well.

Rolling shutter artifacts. Our system is robust against rolling shutter artifacts caused by camera shaking because Bézier curves are always smooth. The offset deformations of neighboring trajectories will be stabilized by our system and the objects can thus become rigid. Unlike the videos free from rolling shutter artifacts, we compute optical flows [22] between consecutive frames and seed trajectories to track the wobbling motion of each local region. The presented trajectory smoothing and video frame warping steps are then used without any modification. Although optical flows are less accurate than KLT features, they are not easily interrupted by content deformations so as to capture the artifacts. In addition, we chose the features that have only backgrounds because ruling out foreground but keeping wobbling features automatically is nearly impossible. Although our method is robust against rolling shutter artifacts, it alone cannot correct all kinds of rolling shutter artifacts because it does not estimate the object geometries of the video [23], [24]. For the skew artifacts caused by a large but smooth camera motion, our system requires an additional rectification of each video frame after the stabilization to achieve a high-quality result.

Limitations. Our system relies on the tracked features to stabilize videos. Since even state-of-the-art trackers are not perfect, our stabilizing results are not always satisfactory. Some video frames potentially fall back to the shaky states due to excessive blur caused by extremely fast motions or lack of rigid background objects (see Figure 8). In addition, as video frames are warped to achieve stabilization, our system may crop too much information if the given video is aggressively stabilized. We consider solving this problem using motion inpainting as our future work.

While our method is robust against rolling shutters, it cannot eliminate these artifacts because it only smoothes the offset deformations of neighboring tra-

2. http://youtube.com/editor

jectories. Even when the video can be stabilized, it is also hard to find out the real shape of an object because their captured appearances are different in frames. We intend to investigate the possibility of estimating object geometries to improve the stabilization quality.

6 Conclusions and Future Works

We have introduced a robust, efficient and streamable technique to video stabilization. Thanks to the spatial preservation of features in each video frame, our system successfully handles parallax without the reconstruction of a 3D scene. Although we achieve high robustness by solving a global optimization, our Bézier curve fitting greatly reduces the computational cost when stabilizing a video. In addition, this strategy strongly stabilizes the video because the feature positions are linearly interpolated based on the optimized control points. Our technique matches or performs even better than the quality of state-of-the-art stabilizing systems while being robust to wide and diverse set of examples. We show the experimental results in our accompanying and the supplemental videos to verify our technique.

Our streaming implementation achieves scalability and real-time performance when stabilizing long and high-resolution videos, as long as KLT features are pre-computed. Since real-time KLT implementations are currently available[3], we plan to embed this GPU-accelerated feature extraction into our system in the near future.

Acknowledgements

We would like to thank Dr. Michael Adams for narrating the demo video and to Pei-Chih Wen for helping us with the experiments.

References

[1] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3d video stabilization," *ACM Transactions on Graphics*, pp. 44:1–44:9, 2009.
[2] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *CVPR*, 2012.
[3] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Transactions on Graphics*, vol. 30, no. 1, pp. 1–10, 2011.
[4] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Transactions on Graphics*, pp. 1–10, 2012.
[5] S. Wu and Z. Ren, "Video stabilization by multi-trajectory mapping and smoothing," *Information, Communications and Signal Processing*, pp. 542 – 545, 2005.
[6] M. Long, L. Xinggang, X. Lidong, and F. Fangwen, "Video stabilizing system for digital camera," in *International Conference on Singal Processing*, 2004, pp. 1119–1122.
[7] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *CVPR*, 2011.
[8] C. Morimoto and R. Chellappa, "Evaluation of image stabilization algorithms," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998, pp. 2789–2792.
[9] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, 2006.
[10] J. Yang, D. Schonfeld, C. Chen, and M. A. Mohamed, "Online video stabilization based on particle filters." in *ICIP*, 2006, pp. 1545–1548.
[11] B.-Y. Chen, K.-Y. Lee, J.-S. Lin, and W.-T. Huang, "Capturing intention-based full-frame video stabilization," in *Proceedings of Pacific Graphics*, 2008, pp. 1805 –1814.
[12] M. Gleicher and F. Liu, "Re-cinematography: Improving the camerawork of casual video," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 5, no. 1, pp. 1–28, 2008.
[13] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung, "Video stabilization using robust feature trajectories," in *ICCV*, 2009, pp. 1397–1404.
[14] A. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-based rendering using image-based priors," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 141–151, 2005.
[15] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, M. Cohen, B. Curless, and S. B. Kang, "Using photographs to enhance videos of a static scene," in *Proceedings of Eurographics Workshop on Rendering*, 2007, pp. 327–338.
[16] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao, "Video stabilization based on a 3D perspective camera model," *The Visual Computer*, vol. 25, no. 11, pp. 997–1008, 2009.
[17] Z. Huang, F. He, X. Cai, Y. Chen, and X. Chen, "A 2d-3d hybrid approach to video stabilization," in *Proceedings of International Conference on Computer-Aided Design and Computer Graphics*. IEEE Computer Society, 2011, pp. 146–150.
[18] J. Shi and C. Tomasi, "Good features to track," in *CVPR*, 1994, pp. 593–600.
[19] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
[20] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, 2007, pp. 109–116.
[21] "Voodoo camera tracker: A tool for the integration of virtual and real scenes," http://www.digilab.uni-hannover.de/docs/manual.html.
[22] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic Huber-L1 optical flow," in *Proc. British Machine Vision Conference (BMVC)*, September 2009.
[23] S. Baker, E. P. Bennett, S. B. Kang, and R. Szeliski, "Removing rolling shutter wobble," in *CVPR*, 2010, pp. 2392–2399.
[24] E. Ringaby and P.-E. Forssén, "Efficient video rectification and stabilisation for cell-phones," *International Journal of Computer Vision*, 2011.

3. http://cs.unc.edu/∼ssinha/Research/GPU_KLT/