Video Stabilization with a Depth Camera

Shuaicheng Liu[1] Yinting Wang[1] Lu Yuan[2] Ping Tan[1] Jian Sun[2]

[1]National University of Singapore [2]Microsoft Research Asia

Abstract

Previous video stabilization methods often employ homographies to model transitions between consecutive frames, or require robust long feature tracks. However, the homography model is invalid for scenes with significant depth variations, and feature point tracking is fragile in videos with textureless objects, severe occlusion or camera rotation. To address these challenging cases, we propose to solve video stabilization with an additional depth sensor such as the Kinect camera. Though the depth image is noisy, incomplete and low resolution, it facilitates both camera motion estimation and frame warping, which makes the video stabilization a much well posed problem. The experiments demonstrate the effectiveness of our algorithm.

1. Introduction

Videos captured by a handheld camera often suffer from serious frame jitters, which is the most obvious difference from professional videos. Video stabilization techniques can make casually captured videos look professional by removing these undesirable jitters. However, existing methods are limited by two key issues. First, methods rely on homography based frame registration such as [12, 6] suffer from image distortion when there are significant depth changes in a scene. In principle, a homography can register two frames only when the scene is flat, or when there is no camera translation at all. These two conditions are not precisely true in most real videos, and can cause serious distortions in the stabilized results, especially when the distance between scene objects and camera is small such as indoor scenes. Second, long feature tracks are difficult to obtain in scenes with severe occlusion, sudden camera rotation, motion blur, or textureless objects (e.g. white walls in indoor scenes). Hence, methods requiring feature tracking such as [9, 10] tend to fail in these challenging cases.

We propose to use additional depth sensors to solve these two challenging problems. Depth sensors such as the Kinect camera are cheap, compact and widely available in the market. Though their depth measure is noisy, incomplete and low resolution at each frame, this additional information can make the video stabilization much robust. Specifically, we exploit the rough depth measure to improve both the camera motion estimation and frame warping.

We first combine color and depth images to robustly compute 3D camera motion. Since we have depth information, we perform motion estimation between every two consecutive frames without requiring long feature tracks. We extract corresponding points between two neighboring frames, and use their depths to estimate relative camera motion. Our method does not rely on fragile feature tracking, or structure-from-motion algorithms [7]. We then smooth the recovered 3D camera trajectories following cinematography principles [6], which removes both high frequency camera jitters and low frequency shakes. The novel video frames can be generated by projecting 3D scene points (generated from the depth image) according to the new camera poses. However, since the depth image is noisy and incomplete, we further warp the color image to fill-in missing regions caused by incomplete depth to create the final results.

2. Related Work

Previous methods on video stabilization can be roughly divided as 2D and 3D stabilization. 2D video stabilization methods use a series of 2D transformations to represent the camera motion, and smooth these transformations to stabilize the video. Early 2D video stabilization methods such as [13, 12] computed affine or homography transformations between consecutive frames and applied low pass filtering to reduce high frequency camera jitter. To reduce low frequency camera shakes, Chen et al. [4] fit polynomial curves to represent camera trajectories. Gleicher and Liu [5] further broke camera trajectories into segments and fit smooth motion to each of them for better camera motion. More recently, [6] applied cinematography rules and represented camera motion by a combination of constant, linear or parabolic motion. All these methods share a common disadvantage that the assumed 2D motion model (e.g. affine or homography transformations) is insufficient to model frame changes when the scene contains significant depth changes.

3D video stabilization methods reconstruct the 3D camera trajectories and smooth them for stable videos. Buehler et al. [3] proposed a 3D video stabilization method based on a projective reconstruction of the scene with an uncalibrated camera. When a Euclidean reconstruction can be obtained, Zhang et al. [16] smoothed the camera trajectories to minimize its acceleration in rotation, translation and zooming. However, video frames at the new camera poses were generated by applying a per-frame homography to original frames. A full 3D stabilization method was proposed by Liu et al. [9]. After smoothing the 3D camera trajectories, the projections of reconstructed 3D points were used to control a 'content-preserving' warping of the original frames to create the final result. These methods are limited by their employed 3D reconstruction algorithms. Though there is significant progress [14, 1] in 3D reconstruction, reconstructing a general video is still difficult. Videos with zooming, quick rotation and significant moving objects will make the reconstruction algorithm fail. Besides these problems, robust 3D reconstruction requires long feature tracks, which are difficult to obtain in amateur videos.

A trade-off between 2D and 3D stabilization techniques is to directly smooth the trajectories of tracked image feature points. Lee et al. [8] directly searched for a serious of similarity or affine transformations between neighboring frames to minimize the acceleration of image feature points. To better capture the scene 3D structure, Liu et al. [10] proposed to smooth the bases of the subspace formed by these feature trajectories. However, in real amateur videos, feature point tracking is also complicated by occlusion and camera rotation, which makes these methods fragile.

Smith et al. [15] employed specialized hardware, a light field camera, to solve the video stabilization problem. Here, we also employ additional hardware, a depth sensor, for stabilization. Depth cameras are cheap and widely available in the market, such as the Kinect camera and other time-of-flight cameras. A depth camera helps us in camera motion estimation as well as the creation of novel frames. Though we demonstrate our method with the Kinect camera in indoor scenes, we believe the same algorithm can be applied to time-of-flight cameras to work in outdoor environments.

3. Challenges in Video Stabilization

Before going to the details of our method, we first highlight two key challenges to previous video stabilization methods, which commonly exist in indoor scenes. Indoor scenes are particularly important, because many amateur videos (such as family event, party, shopping, etc) are captured in indoors. Many of previous methods employed 2D transformations such as similarity [8], affine or homography [12, 6] transformations to register neighboring frames. However, these simple motion models are invalid when there are large depth changes in the scene, especially when the scene is close to the camera. Figure 1 shows such an example where three cubes in front of a wall are captured by a handheld video camera. The first row shows two frames of the original shaky video. The second row are the corresponding frames from the video stabilized according to [6] [1]. The results are clearly distorted. For example, the first-aid box on the left image is subject to a shearing mapping. This is because the sudden depth change between the cubes and the wall makes homography based registration invalid. For a comparison, the same frames from the video stabilized by our method are shown in the last row. Our method is free from this distortion by exploiting rough depth information from a depth camera.

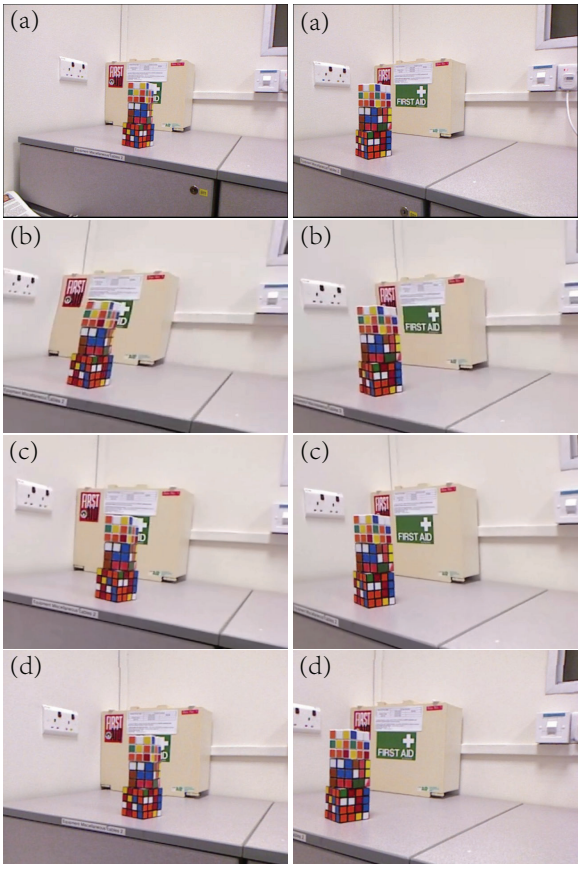
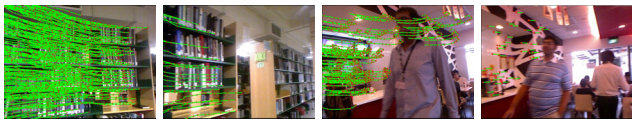


Figure 1. Results on the 'Cube' example. From top to bottom are two sample frames from (a) original video,(b) 2D stabilization method[6], (c) 3D stabilization method[10] and (d) our approach. The results (b) and (c) both have clear shear and wobble distortions at the first-aid box and cubes, while our results are more visually pleasing.

3D video stabilization methods such as [3, 16, 9] require feature correspondence in different frames for robust 3D reconstruction. Methods based on feature track smoothing

[1] We uploaded our videos to Youtube (http://www.youtube.com) with the 'stabilize' feature enabled. The uploaded videos are stabilized by the website server according to the method in [6]. We then downloaded the results for comparison.



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

Figure 2. Feature point tracking in amateur videos is difficult. (a) frame in a video with quick rotation,(c) frame in a video with severe occlusion. (b) and (d) are frames after (a) and (c). Both the number of tracked points and the length of the feature tracks drop significantly in (b) and (d).

such as [8, 10] also need long tracks of feature points. As commented in [10], typically, features should be tracked for about 50 frames to make their algorithm robust. However, robust tracking of feature points is a difficult problem, which could be affected by textureless regions, sudden camera rotation or severe occlusion. The third row of Figure 1 shows the results from [10][2]. Most of the tracked feature points locate on the foreground cubes, which leads to wobble artifacts on the background first-aid box (please refer to the supplementary videos for a clearer comparison).

To further demonstrate the tracking difficulty, we show two typical amateur videos in Figure 2. Figure 2 (a)(b) and (c)(d) show two frames from one video.The video of frame(a)and(b) has quick rotation, while the one of frame(c)and(d) suffers from severe occlusion caused by pedestrians. On the selected video frames, we overlay the trajectories of tracked feature points. Here we used the KLT tracker [11] to trace detected SURF features [2]. On each trajectory, the red points are the feature positions in tracked frames. When rotation or occlusion happens, both the number of tracked feature points and the length of feature tracks drop significantly, which makes feature tracking based video stabilization fragile. The average lengthes of feature tracks in the image (a) and (c) are 10 and 23 frames. In comparison, the average lengthes of (b) and (d) are 6 and 2 frames. The numbers of tracked points are also reduced from 248 in (a) and 158 in (c) to 21 in (b) and 37 in (d). With an additional depth camera, we compute camera motion between any two consecutive frames from corresponding pixels with known depth. This method does not require long feature tracks. Hence, we avoid this challenging tracking problem.

4. Our Method

The input to our method is a video with an accompany depth image for each frame. In developing our algorithm, we use the Kinect camera in indoor scenes for data capturing, though other depth sensors might also be used. Similar to most of the video stabilization methods, our method includes mainly three steps. We first estimate the 3D camera motion from neighboring color and depth images. Since we have depth information, we do not require long feature tracks for 3D reconstruction. Once the 3D camera trajectory is known, we smooth it following [6] to reduce both high frequency jitters and low frequency shakes. We then generate video frames according to the smoothed camera poses, again by combing information from color and depth images.

[2]We used the 'stabilize motion' with 'subspace warp' in the Adobe After Effects CS5.5 with 50% smoothness and Rolling shutter automatic reducing to generate results of [10].

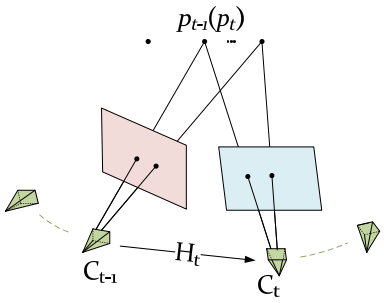


Figure 3. Camera motion estimation from corresponding 3D points between two consecutive frames. p_t and p_{t-1} are coordinates of the same 3D point in two local camera coordinate systems. The Euclidean transformation H_t between two cameras can be estimated from corresponding 3D points.

4.1. Camera motion estimation

We begin by recovering camera motion in the original shaky video. Our input are the video frames I_1, I_2, $\cdots$, I_n, and their corresponding depth images P_1, P_2, $\cdots$, P_n measured in local camera coordinate system. We seek to estimate a 4×4 matrix C_t at each time t that represents the camera pose in a global coordinate system, i.e.

$$C_t = \begin{pmatrix} R_t & O_t \\ 0 & 1 \end{pmatrix}.$$

Here, R_t and O_t are the 3×3 rotation matrix and 3×1 translation vectors representing the camera orientation and position in the global coordinate system respectively.

As shown in Figure 3, the relative camera motion at time t can be represented by a 3D Euclidean transformation H_t satisfying $C_t = C_{t-1}H_t$. H_t has similar form as C_t, where

$$H_t = \begin{pmatrix} \hat{R}_t & \hat{O}_t \\ 0 & 1 \end{pmatrix}.$$

Here, $\hat{R}_t, \hat{O}_t$ are the rotation and translation components of H_t. We set the world coordinate system at the first frame. Hence, camera poses can be computed by chaining the relative motions between consecutive frames as

$$C_t = H_1 H_2 ... H_t.$$

To estimate H_t, we first detect and match SURF features [2] between two frames I_{t-1} and I_t. Since depth images

are incomplete, some matched feature points might not have depth recorded. Here, we only choose those corresponding feature points whose depths in both P_{t-1} and P_t are known. Each pair of correspondence introduces a constraint about H_t as,

$$\hat{R}_t p_{t-1} + \hat{O}_t = p_t.$$

As illustrated in Figure 3, p_t, p_{t-1} are the coordinates of the same 3D point in the two local camera coordinate systems of the frame t and $t-1$ respectively.

Suppose N pairs of features are collected, we can then estimate H_t (i.e. $\hat{R}_t, \hat{O}_t$) by minimizing

$$\sum_{i=1}^{N} \rho(||\hat{R}_t p_{t-1} + \hat{O}_t - p_t||_2). \quad (1)$$

Here, $\rho(\cdot)$ is the M-estimator (we use the Tukey bi-weight function [17]) for robust estimation defined as

$$\rho(x) = \begin{cases} t^2/6\,(1-[1-(\frac{x}{t})^2]^3) & \text{if } |x| \le t \\ t^2/6 & \text{otherwise.} \end{cases}$$

Equation 1 is minimized by the standard iteratively re-weighted least squares (IRLS) method [17]. During the computation, RANSAC is also applied to skip outliers. Specifically, we repetitively draw three random pairs of corresponding points at a time to solve Equation 1 until we find the largest set of inliers. We then solve Equation 1 again with all inliers to decide the camera motion. For computation efficiency, during the random sampling, we set $t = +\infty$ (i.e. without using M-estimator), while we set t as the standard deviation of the fitting residual in all inliers in the final estimation.

4.2. Camera trajectory smoothing

We smooth the estimated camera trajectory for stable motion. We follow [6] to adopt cinematography principles to remove both high frequency jitters and low frequency shakes. The smoothed camera trajectory should be a combination of constant, linear and parabolic motion. Note that the key difference from [6] is that we work with real 3D camera poses (i.e. orientations and positions), while [6] used a series of homogrpahies to indicate the camera motion.

We represent the camera rotation matrix R_t by its quaternions, which offer a better representation for interpolation than Eulerian angles. For notation simplicity, we still denote these quaternions by R_t. We then concatenate the 4D quaternions R_t and the 3D translation vector O_t to a 7D vector F_t to represent the camera pose at time t. The optimal camera trajectory is obtained by minimizing the following objective function,

$$\mathcal{O}(F) = w_1||D(F)||_1 + w_2||D^2(F)||_1 + w_3||D^3(F)||_1 \quad (2)$$

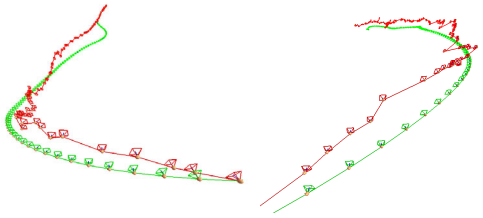


Figure 4. Camera trajectory smoothing results. The red and green curves show trajectories before and after smoothing respectively.

where $||D(F)||_1$, $||D^2(F)||_1$ and $||D^3(F)||_1$ are the L-1 norms of the first order, second order and third order camera pose derivatives respectively. We set $w_1 = 10, w_2 = 1, w_3 = 100$ for all our examples.

The optimization is solved by linear programming with the first camera pose F_1 unchanged. Following [6], we also require new camera poses to be close to the original ones to simplify video generation. Specifically, we require the components in R_t do not change more than 0.05, and the components in O_t do not change more than 20 in all our experiments. In other words, the new camera center should be within 20 mm distance to its original position, the principal axis should not change for more than 3 degrees. Figure 4 shows the camera trajectories before and after smoothing in red and green respectively.

4.3. Video frame generation

Once we obtain the stabilized camera poses, we are ready to synthesize the output video. In principle, if the depth sensor returns a precise depth for each pixel, we can generate the stabilized frame by simply projecting all 3D points according to smoothed camera poses. However, the depth image is often incomplete, as shown by the grayscale images in Figure 5 (a). Figure 5 (b) shows a projection of the 3D points (generated from the color and depth image in Figure 5 (a)) to the stabilized video frame, where many pixels are missing because of the incomplete depth map. Hence, we apply the 'content-preserving' image warping [9] to fill-in these missing regions.

To seamlessly blend results from projecting 3D points and image warping, we use morphological dilation operator to create a r-pixel width ($r = 1.5\%$ of image width in our experiments) buffer band surrounding all missing regions. We use all pixels in this band as 'control points' for image warping, so that the warping will be as consistent as possible with the projection. Figure 5 (d) shows the green control points and the image warping grid (a clearer version is provided in the left of Figure 6). We combine these two methods in the band by linearly interpolating the two motion fields introduced by them.

Motion field from depth images We project pixels with depth measure according to the smoothed camera pose. Given the original camera pose C_t and its smoothed pose

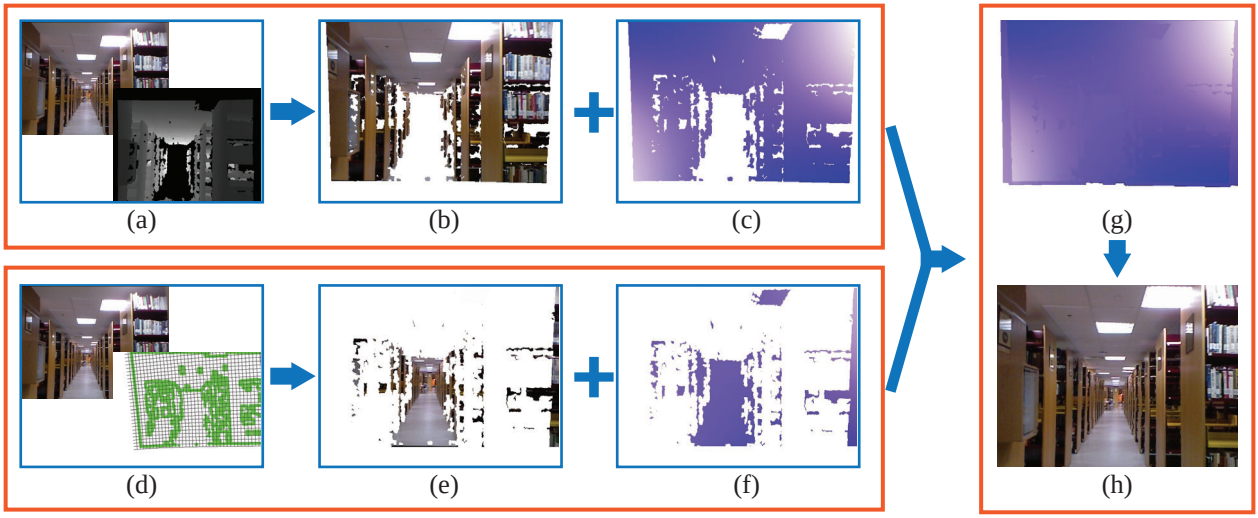


Figure 5. Video frame generation pipeline. We use the color and depth images in (a) to generate the projection in (b) and the motion field in (c). Many pixels are missing because of the incomplete depth image. Hence, we warp the color image by the 'content-preserving' warping [9] in (d) according to the green control points and a regular grid. This warping generate a color image (e) and a motion field (f). We then generate a complete motion field (g) by combining (c) and (f). The final video frame (h) is created by warp the original frame with (g).

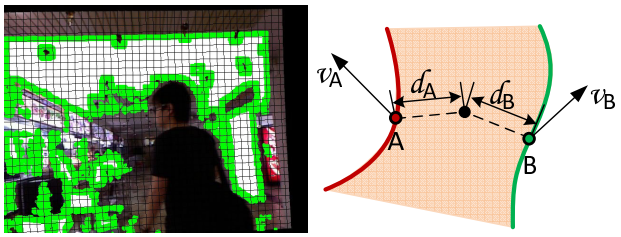


Figure 6. Left: control points and image grid for 'content-preserving' warping. Right: illustration for motion interpolation.

C'_t, we can compute the image coordinates of a 3D point in both original and stabilized video frames (we need to calibrated the intrinsic camera matrix K beforehand). The difference between these two coordinates gives a motion vector, which maps a pixel from original video to the stabilized one. In this way, we obtain a motion field M_t^1 that covers all pixels with depth measure, as shown in Figure 5 (c).

Motion field from image warping To fill in missing regions, we take all pixels in the buffer band as control points for the 'content-preserving' warping. Basically, we partition the original image into 10×10 regular grid. We solve a linear equation [9] to decide the motion of the grid vertices. The motion of a pixel is then computed by bilinear interpolation of the motion vectors at its four grid vertices. This generates another motion field M_t^2, which covers all pixels without depth measure and the buffer band as show in Figure 5 (f).

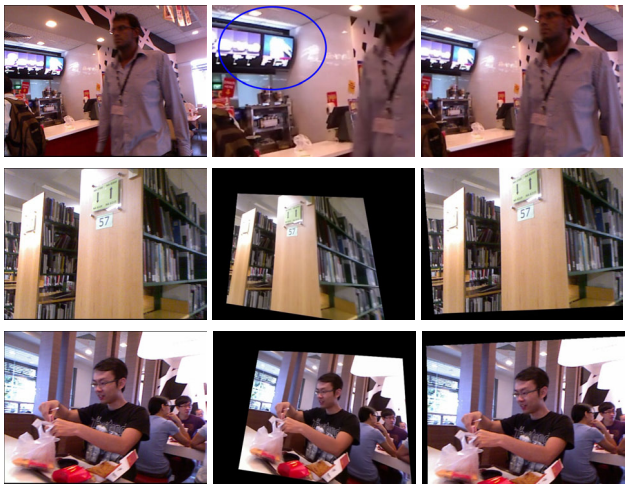
Motion fields blending We then linearly blend M_t^1 and M_t^2 in the buffer band. Specifically, the motion of a pixel in the band is computed by linearly interpolating the motion of its two nearest neighbors at the two sides of the band. As shown on the right of Figure 6, A, B are two pixels on the two sides of the band with minimum distance (d_A, d_B respectively) to the black pixel in consideration. v_A, v_B are the motion vectors of A and B, which are computed from projecting 3D points and image warping respectively. We linearly interpolate these two vectors in the band to blend M_t^1 and M_t^2. For example, the motion of the black pixel is computed as

$$d_A/(d_B + d_A)v_B + d_B/(d_A + d_B)v_A.$$

Figure 5 (g) shows the interpolated motion from (c) and (f). Once the motion field is obtained for the whole frame, we use it to warp the original video frame to create the stabilized frame as shown in Figure 5 (h).

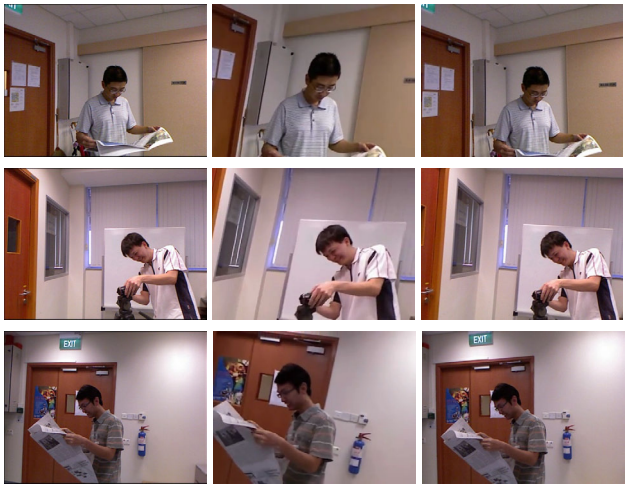
5. Experiments

We evaluated our method with some challenging videos captured by a Kinect camera. To avoid the calibration between the color and depth cameras, we used the embedded color camera in Kinect whose calibration is known. All our videos have resolution of 640×480. Figure 1 and Figure 9 compare our results with the method described in [6] and [10]. In both figures, from top to bottom, the four rows for each example are sample frames of the original video, stabilized video according to [6], [10] and our method respectively. For easy reference, we name these examples in Figure 1 and Figure 9 as 'Cube', 'Boy'. The 'Cube' and 'Boy' examples showed a nearby scene with sudden depth change, which made the homography based frame registration in [6] fail. Hence, severe geometric distortions were observed in these results (please notice the shear distortion on the first-aid box in the 'Cube' example, and on the bookshelf in the



(a) (b) (c)

Figure 7. Comparison with method[10]. Each row shows one example. Columns from left to right: (a)sample frames from original video, (b)stabilized video according to [10] and (c)our method. Please notice the zoom in artifacts circled in blue of the first example and warping distortion of the second and third examples in (b).



(a) (b) (c)

Figure 8. Comparison with method[6]. Each row shows one example. Columns from left to right: (a)sample frames from original video, (b)stabilized video according to [6] and (c)our method. Please notice the wobble on the background in (b).

Figure 10. Additional results under different indoor environment from our video stabilization. Please visit project website for more videos.

'Boy' example). The 'content-preserving' warping in [10] is more robust to depth changes. However, the large textureless wall in the 'Cube' example had few tracked feature points, which caused wobble in the result, e.g. on the first-aid box. (Note that tracked feature points were used as control points for warping in [10]. Similar artifacts were reported in [9] when the image feature points distributed unequally over the image.) Though more feature points can be tracked in the 'Boy' example, it was not stabilized well by [10], perhaps because the dynamic scene confused the subspace analysis. In comparison, our method took advantage of the depth camera and generated better results on all these examples.

Figure 7 provide more comparison with method[10]. The first example of Figure 7 contained severe occlusion, where people walked through and blocked the whole frame. It is challenging for [10] because of tracking failures caused by severe occlusion. The region circled in blue had inconsistent motion in the stabilized video. The second and third example of Figure 7 contained quick camera rotation. Both the number of tracked points and the length of the feature tracks drop significantly. This caused shear artifacts on the whole scene. Furthermore the warping distortion produce a large empty area compared to ours. In Figure 8, three examples illustrate the severe geometric distortion of method [6]. The depth change confused the homography registration. The simple linear model can not describe variations of depth in these scenario. Please notice the shear distortion on the background of Figure 8 (b).

We show additional examples in Figure 10. Please refer to project website for more results. Examples in Figure 10 were captured under various indoor environment including library,canteen,shopping mall, supermarket,gym etc. Some of them contain large depth change and sever occlusion. With the help of a depth camera, our method generated steady video from these challenging examples.

Limitations We observe a number of limitations of our current method during experiments, which point out the direction for future work. First, our method does not consider the rolling shutter effects of both the color camera and the depth camera, which sometimes make the camera motion estimation imprecise and lead to some high frequency jitters in the results. Second, our current implementation is limited to the Kinect camera, which only works in indoor scenes. But we believe the same algorithm can be applied to time-of-flight cameras in outdoor environments. We leave this to the future study.

6. Conclusion

We study two challenges in video stabilization, namely sudden depth change making 2D motion model imprecise

Figure 9. Results on the 'Boy' examples. From top to bottom, the four rows are sample frames from (a) original video, (b) stabilized video according to [6], (c) stabilized video according to[10] and (d) our method.

and tracking failure making 3D stabilization fail. We solve these problems with an additional depth sensor, which provides a depth measure for each video frame. We exploit this rough depth information to improve both camera motion estimation and frame warping. The result is a robust stabilization algorithm that works in challenging cases.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. ICCV*, 2009. 2

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, 2008. 3

[3] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *Proc. CVPR*, 2001. 2

[4] B.-Y. Chen, K.-Y. Lee, W.-T. Huang, and J.-S. Lin. Capturing intention-based full-frame video stabilization. *Computer Graphics Forum*, 27(7):1805–1814, 2008. 1

[5] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camera dynamics of casual video. In *Proc. of ACM Multimedia*, 2007. 1

[6] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *Proc. CVPR*, 2011. 1, 2, 3, 4, 5, 6, 7

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. 1

[8] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung. Video stabilization using robust feature trajectories. In *Proc. ICCV*, 2009. 2, 3

[9] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 28, 2009. 1, 2, 4, 5, 6

[10] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30, 2011. 1, 2, 3, 5, 6, 7

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of International Joint Conference on Artificial intelligence (IJCAI)*, pages 674–679, 1981. 3

[12] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:1150–1163, 2006. 1, 2

[13] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2789 – 2792, 1998. 1

[14] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:756–777, 2004. 2

[15] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala. Light field video stabilization. In *Proc. ICCV*, 2009. 2

[16] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao. Video stabilization based on a 3d perspective camera model. *Vis. Comput.*, 25:997–1008, 2009. 2

[17] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997. 4