LOCAL SUBSPACE VIDEO STABILIZATION

Chengzhou Tang, Ronggang Wang

School of Electronic and Computer Engineering
Peking University Shenzhen Graduate School
cztang@sz.pku.edu.cn, rgwang@pkusz.edu.cn

ABSTRACT

Video stabilization enhances video quality by stabilizing unstable motion. This paper proposes a new video stabilization method that simultaneously factors and smooths motion trajectories. We model the trajectories with a time-variant local subspace constraint. Every column of the trajectory matrix is factored and smoothed in separate local subspace. This model makes our method more flexible and accurate than subspace video stabilization. In addition, we design a novel outlier detection technique that utilizes the relationship between consecutive local subspaces. Experiments on synthetic data validate the numerical performance of our factorization. Quantitative comparisons on real videos show that our local method is better than subspace video stabilization. Moreover, our stabilized videos are comparable with the public results from some other state-of-the-art methods.

***Index Terms*—** Video stabilization, Matrix factorization, Local subspace constraint

1. INTRODUCTION

Taking satisfactory video sequences with hand-held devices is difficult for amateur recorders. One of the main differences between videos shot by the professional and the non-professional is camera motion. In film industry and other professional fields, cameramen use dollies or steadicams to get stable motion. These hardware solutions are impractical for ordinary users, so video stabilization softwares are usually used in post processing stage.

Previous video stabilization methods first estimate and smooth 2D camera motion [11, 6, 4] or 3D camera motion [7, 14], then synthesize a stabilized video using the correspondence between raw and smoothed motion. The 2D methods model camera motion between consecutive frames as affine transformation or homography. Generally, 2D methods are faster and more robust than 3D methods. However, they often fail in scenes with large parallax for lack of model flexibility. In contrast, the 3D methods can handle parallax and produce visual plausible results, but they are rarely used in commercial softwares because they are unreliable under various conditions such as motion blur, large moving object, camera zoom and rolling shutter. Furthermore, the computational complexity of recovering 3D information is unacceptable. As reported in [7], structure-from-motion with Voodoo Tracker [1] takes several hours for a short video.

Recently, both 2D and 3D methods have been greatly improved. In [10], S. Liu et al. proposed bundled camera paths, which are more flexible than a single global camera path. They stabilize estimated camera paths using space-time optimization and warp frames with spatially-variant homographies. Wang et al. [13] represent each trajectory as a Bézier curve. They formulate video stabilization as a spatial-temporal optimization problem that finds smooth trajectories as well as preserves offsets of neighboring curves. Goldstein and Fattal [3] avoid 3D reconstruction by using 'epipolar transfer' to construct and smooth virtual trajectories.

In this paper, we focus on subspace video stabilization [8], where Liu et al. adopt a low-rank subspace constraint to motion trajectories. They smooth the trajectories using a global subspace constraint rather than filter each trajectory separately. Their method was later extended to stereoscopic video stabilization in [9]. However, Liu et al.'s model is overstrict because in a video captured by a perspective camera, motion trajectories lie on a non-linear manifold but not a linear subspace [2], and this manifold can be approximated as linear subspace only locally [5]. Motivated by this fact, we novelly smooth the columns of a trajectory matrix in different local subspaces. Compared with subspace video stabilization, our local method is more flexible and accurate. In our experiments, it has also generated results comparable with some other state-of-the-art methods.

2. SUBSPACE VIDEO STABILIZATION

For a better understanding of the basic concept, we first revisit subspace video stabilization in this section. Most video stabilization methods track a set of feature points through a video in the first step. In subspace video stabilization [8], the

[1] http://www.digilab.uni-hannover.de/docs/manual.html

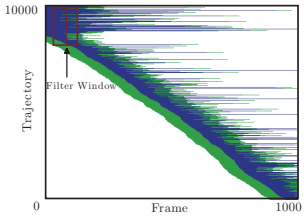


Fig. 1. A typical trajectory matrix from real video. About 84% data in this matrix are missing. *Blue*: tracked trajectories. *Green*: extended segments by the radius of filter kernel. *Red*: low-pass filter window frames with radius of k. *Blank area*: missing data.

trajectories of these points are assembled into a matrix M:

$$\mathrm{M}_{2t\times f} = \begin{bmatrix} \mathrm{x}_{11} & \cdots & \mathrm{x}_{1f} \\ \vdots & \ddots & \vdots \\ \mathrm{x}_{t1} & \cdots & \mathrm{x}_{tf} \end{bmatrix}, \tag{1}$$

where t is the number of the trajectories tracked across f frames, x_{ij} is the i-th trajectory's position in the j-th frame. As shown in Fig 1, M is highly incomplete because trajectories will appear and disappear in a video. Liu et al. model this matrix using a rank r subspace constraint:

$$\mathrm{M}_{2t\times f} = \mathrm{W} \odot \mathrm{C}_{2t\times r}\mathrm{E}_{r\times f}, \tag{2}$$

where W is a binary mask with 0 for missing data, 1 for existing data, and $\odot$ indicates the element-wise multiplication. Liu et al. call the r row vectors of E eigen-trajectories, they are the basis vectors that can be linearly combined to form a 2D motion trajectory. The coefficient matrix C represents each observed trajectory as such a linear combination. A stable trajectory matrix $\widetilde{\mathrm{M}}$ is then obtained by adopting a gaussian kernel K to E:

$$\widetilde{\mathrm{M}} = \mathrm{W} \odot (\mathrm{CE})\mathrm{K} = \mathrm{W} \odot \mathrm{C}(\mathrm{EK}) = \mathrm{W} \odot \mathrm{C}\widetilde{\mathrm{E}}, \tag{3}$$

and a stabilized video is synthesized by content-preserving warps [7] with the guidance of $\widetilde{\mathrm{M}}$.

In order to factor M as in (2), Liu et al. developed a streamable factorization method named moving factorization [8]. They said that to support the low-pass filter, factorization needs to be able to extend each trajectory forwards and backwards in time by the radius of the filter kernel. In (3), M is smoothed by projecting filtered eigen-trajectories back to 2D space. It is equivalent to smoothing M after recovering the missing data locally. The points near the start and the end of the original trajectories are filtered with both tracked and recovered neighbors. So the factorization should be accurate for not only the tracked trajectories but also the extended (recovered) segments. As discussed in [8], the factorization should also preserve sufficient number of points for content-preserving warps. To achieve these goals, we present our model and approach in the next section.

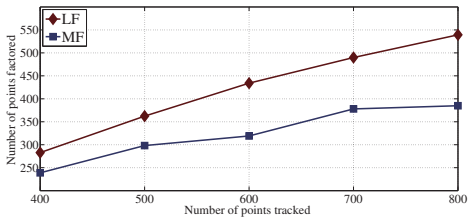


Fig. 2. *X-axis*: number of points tracked per-frame. *Blue*: number of per-frame points after Liu et al.'s moving factorization. *Red*: number of per-frame points after our local factorization.

3. LOCAL SUBSPACE VIDEO STABILIZATION

3.1. Local subspace constraint

Motion trajectories captured by a perspective camera lie on a non-linear manifold instead of a linear subspace [2]. This manifold can be approximated with several linear subspaces locally. Irani [5] showed that for instantaneous motion, a trajectory matrix has at most rank 9. In subspace video stabilization, this property is assumed to be held over a short window of frames, and this window is at least as large as the filter kernel. However, Liu et al. still factor and smooth the trajectory matrix in a single global subspace as in (2) and (3). For the entire trajectory matrix, their global subspace constraint is overstrict. As shown in Fig 2, many correct trajectories are detected as outliers and removed after moving factorization. These correct trajectories' factorization error exceeds a threshold (Liu et al. set the threshold to 3 pixels) because the global subspace constraint can not fit the manifold well.

In our work, we use multiple local subspaces rather than a single global subspace. We model a local window of $2k+1$ frames as:

$$\mathrm{M}^i_{2m\times(2k+1)} = \mathrm{W} \odot \mathrm{C}^i_{2m\times r}\mathrm{E}^i_{r\times(2k+1)}, \tag{4}$$

where i denotes that this window centers on the i-th column of M, m is the trajectory number in this window, and k is the same with the filter kernel's radius. Then we smooth M^i similarly to (3). Since only the center column of $\widetilde{\mathrm{M}}^i$ is obtained by full range filtering, we put it back to the corresponding position in $\widetilde{\mathrm{M}}$ while the other columns are dropped. Thus, we can smooth the columns of M in different local subspaces as:

$$\widetilde{\mathrm{M}} = \mathrm{W} \odot \left[\mathrm{C}^1\widetilde{E}^1, \mathrm{C}^2\widetilde{E}^2 \cdots \mathrm{C}^f\widetilde{E}^f\right], \tag{5}$$

where $\widetilde{E}^i$ is the filtered eigen-trajectories of the i-th column.

This time-variant model is more flexible than (2). Different from subspace video stabilization, we do not require the coefficient matrix fixed. The low-rank representation of a trajectory varies in different local subspaces as in (5), which gives a better approximation to the non-linear manifold. To factor and smooth M as in (4) and (5), we now introduce our local factorization algorithm.

3.2. Local factorization

We start the factorization by assembling m trajectories that span the first $2k+1$ frames into a complete matrix M^1. This matrix can be factored as follows:

$$\mathrm{M}^1_{2m\times(2k+1)} = \mathrm{C}^1_{2m\times r}\mathrm{E}^1_{r\times(2k+1)}, \tag{6}$$

where C and E are obtained by truncating the output of Singular Value Decomposition (SVD) corresponding to the largest r singular values and distributing the square root of each singular value to the left and right orthogonal matrices. Then the first $k+1$ columns of E are filtered and projected back to 2D space as:

$$\widetilde{\mathrm{M}}^1_{1:(k+1)} = \mathrm{C}^1(\mathrm{E}^1_{1:(k+1)}\mathrm{K}) = \mathrm{C}^1\widetilde{\mathrm{E}}^1_{1:(k+1)}. \tag{7}$$

Here, we only smooth the first $k+1$ columns of M^1 because the last k columns of E^1 can not be filtered by the radius of k, and they will be filtered in the following windows.

Besides the m trajectories collected in (6), we compute the coefficients for the other trajectories that start in the first $k+1$ frames by minimizing $\|\mathrm{X}-\mathrm{C}^1_{\mathrm{X}}\mathrm{E}^1_{\mathrm{X}}\|^2_F$ as:

$$\mathrm{C}^1_{\mathrm{X}} = \mathrm{X}{\mathrm{E}^1_{\mathrm{X}}}^{\top}(\mathrm{E}^1_{\mathrm{X}}{\mathrm{E}^1_{\mathrm{X}}}^{\top})^{-1}, \tag{8}$$

where X is a trajectory's segment lies in current window and $\mathrm{E}^1_{\mathrm{X}}$ consists the columns of E^1 that cover X in time. This trajectory's segment in the first $k+1$ frames is then smoothed similarly to (7).

Now we move the window forward by 1 frame. Trajectories that span from the 2nd to the $(2k+2)$-th frame are assembled and factored as the same in (6). For trajectories that cross the $(k+2)$-th frame but do not span the entire window, we compute their coefficients as in (8). In this window, we only smooth the center column of M^2 by filtering and projecting back the corresponding column of eigen-trajectories:

$$\widetilde{\mathrm{M}}^2_{k+1} = \mathrm{C}^2(\mathrm{E}^2_{k+1}\mathrm{K}) = \mathrm{C}^2\widetilde{\mathrm{E}}^2_{k+1}, \tag{9}$$

because full range filtering is invalid for the other columns. Therefore, we can get the stable trajectory matrix $\widetilde{\mathrm{M}}$ by repeating the above procedure until all columns have been processed. For the last k columns of M, we also factor and smooth them together in a similar way to (6) and (7).

The proposed local factorization preserves more trajectories for content-preserving warps as shown in Fig 2, and it also factors points more accurately than Liu et al.'s moving factorization as demonstrated in Section 4.1. Both the sufficient number of trajectories and factorization accuracy are crucial to video stabilization.

3.3. Outlier detection with subspace transform

Incorrect trajectories and the trajectories from moving objects will mislead our local factorization. We first detect these outliers using the outlier detection method in [8]. However, to handle the detected outliers, we cut them instead of removing them entirely. For example, we only remove a trajectory's points after the 30-th frame (including the 30-th frame), where it is detected as an outlier for the first time. But we still discard the entire trajectory when its remaining segment is shorter than $2r$ frames.

During our experiments, we found using the strategy from [8] only could miss some outliers. The reason is that factorization by SVD is not robust, it will be influenced greatly even when a little outliers occur. Although some robust factorization methods [1, 12] have been proposed in the past years, they are impractical for video stabilization because of high computational cost.

To tackle this problem without losing computational efficiency and model flexibility, we design a subspace transform based technique. Here, we use a trajectory's segment between the 2nd and the $(2k+1)$-th frame as an example to introduce our outlier detection method. This trajectory X has different projection on two basis E^1 and E^2 as:

$$\mathrm{X}_{2:2k+1} = \mathrm{C}^1_{\mathrm{X}}\mathrm{E}^1_{2:2k+1} = \mathrm{C}^2_{\mathrm{X}}\mathrm{E}^2_{1:2k}. \tag{10}$$

Since the low-rank representation of $\mathrm{X}_{2:2k+1}$ is not unique, $\mathrm{C}^1_{\mathrm{X}}$ is unequal to $\mathrm{C}^2_{\mathrm{X}}$, so is $\mathrm{E}^1_{2:2k+1}$ to $\mathrm{E}^1_{1:2k}$, but they have the following relationships:

$$\begin{aligned} \mathrm{C}^1_{\mathrm{X}}\mathrm{T} &= \mathrm{C}^2_{\mathrm{X}} \\ \mathrm{E}^1_{2:2k+1} &= \mathrm{T}\mathrm{E}^2_{1:2k}, \end{aligned} \tag{11}$$

where T is a $r\times r$ matrix that transforms $\mathrm{C}^1_{\mathrm{X}}$ to $\mathrm{C}^2_{\mathrm{X}}$, and we solve for it by minimizing $\|\mathrm{E}^1_{2:2k+1}-\mathrm{T}\mathrm{E}^2_{1:2k}\|^2_F$ as:

$$\mathrm{T} = \mathrm{E}^1_{2:2k+1}{\mathrm{E}^2_{1:2k}}^{\top}(\mathrm{E}^2_{1:2k}{\mathrm{E}^2_{1:2k}}^{\top})^{-1}. \tag{12}$$

Then we predict the trajectory's point in the $(2k+2)$-th frame as X'_{2k+2} using $\mathrm{C}^1_{\mathrm{X}}$:

$$\mathrm{X}'_{2k+2} = \mathrm{C}^2_{\mathrm{X}}\mathrm{E}^2_{2k+1} = \mathrm{C}^1_{\mathrm{X}}\mathrm{T}\mathrm{E}^2_{2k+1}, \tag{13}$$

and detect the actually tracked X_{2k+2} as an outlier if $\|\mathrm{X}_{2k+2}-\mathrm{X}'_{2k+2}\|_2$ exceeds 10 pixels. The large error indicates that the low-rank representation of trajectory X changes greatly in the $(2k+2)$-th frame, i.e., X_{2k+2} is incorrectly tracked. Otherwise, X'_{2k+2} should be a good assumption to X_{2k+2}.

This outlier detection method inherits the advantage of global subspace constraint by fixing the coefficient matrix between consecutive frames, but it does not affect the model flexibility because every column is still factored and smoothed separately.

3.4. Summary

Now we summarize the proposed method as follows. Since we use a local linear subspace constraint in this paper, we call our method local subspace video stabilization.

1. Track 2D motion trajectories in an input video and assemble them into a matrix M.

2. Factor and smooth columns of M in local subspace using (6)-(9). The first k columns of M are smoothed in a same subspace as in (7); the last k columns are processed in a similar way; and every other column is smoothed in separate local subspace as in (9).

3. Before factoring for every local subspace, detect outliers using the method in [8]. After local factorization in every local window, detect and handle outliers using subspace transform based technique as in (10)-(13).

4. Warp the input video guided by the correspondence between M and $\widetilde{\mathrm{M}}$ using content-preserving warps.

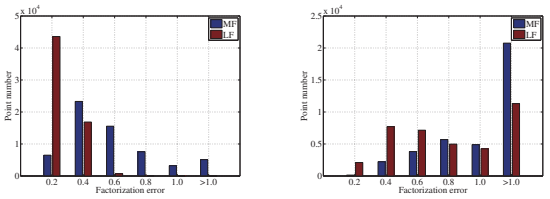
4. EXPERIMENTS

4.1. Synthetic data

As discussed in Section 2, for video stabilization, factorization should be accurate for both tracked trajectories and extended segments. In real videos, there is no access to the missing data of a trajectory matrix. So we randomly generated 500 points within a $100 \times 100 \times 100$ cube, and synthesized a sequence of 300 frames (the size of these frames are 600×600) by a moving perspective camera. The focal length of the camera was set to 500 and rotation angles changed from $-\pi/4$ to $+\pi/4$. Then we added gaussian noise ($\sigma = 3$ and $\mu = 0$) to every captured point in synthesized frames. The factorization window size and moving step of moving factorization were set to 50 and 5, which were the same as in [8]. The factorization window size of our local factorization was 51. Since the synthetic sequence was captured by a perspective camera, we set the rank r to 9. These parameters were also used in experiments on real videos, but we did not employ outlier detection because the synthetic sequence was free from outliers.

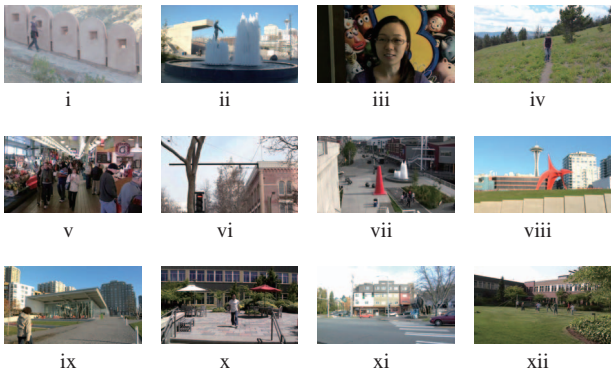
After factoring the synthesized trajectory matrix by moving factorization (MF) and our local factorization (LF), we compared their factorization error on both captured and extended points. Fig 3(a) shows that there are much more points with error less than 0.2 pixels in our method. The average error is 0.0836 pixels by our local factorization and 0.4105 pixels by moving factorization.

For extended points, our method is also much more accurate than moving factorization. The average error on extended points is 0.7076 pixels by our method, which is about half of the error 1.4743 pixels by moving factorization. As shown in Fig 3(b), most extended points in our method have error less than 1 pixel while moving factorization has generated more points with larger error. It is because our method better approximates the non-linear manifold using local subspace constraint. As discussed in Section 2, accurately extended points are crucial to motion trajectories smoothing, and will further improve the quality of stabilized videos.



(a) (b)

Fig. 3. Comparisons on synthesized data. (a) The bar graph of captured point number by factorization error. (b) The bar graph of extended point number by factorization error.



i ii iii iv

v vi vii viii

ix x xi xii

Fig. 4. Snapshots of the 12 video sequences used in experiments.

4.2. Video stabilization

In this experiment, we tested our method on shaky videos. We quantitatively compared the proposed method with subspace video stabilization. Subjective comparisons with some other state-of-the-art methods are given by snapshots as well as a supplementary video.

For quantitative comparisons, 12 videos shown in Fig 4 were used. They cover various scene types including camera zoom, dynamic scene, large parallax, large moving object, etc. These videos were also used by many previous works [7, 8, 3, 4, 10]. To evaluate the quality of stabilized videos, we redefined the three objective metrics used in [10]:

Cropping Since the scope of every synthesized frame is different, videos will have blank areas after stabilization. Matsushita et al. proposed motion inpainting to fill the blank [11], but their method is rarely used in real applications because of complexity and robustness issues. Most state-of-the-art methods unify frames' size by cropping stabilized frames to their overlapping region. The cropping is computed as the ratio of cropped frames' size to the original. A higher cropping ratio means the video content is better preserved.

Distortion In content-preserving warps [7], warps mesh is obtained by minimizing the following objective function:

$$E = E_d + \alpha E_s, \tag{14}$$

where α is a scalar weight between the data term E_d and the similarity term E_s. E_s is the constraint that controls the geometric distortion of a warped frame. In our experiments, we used $\log(E_s + 1)$ to measure the distortion of a stabilized frame. The log function reduces the difference between different high distortion levels because this difference is less notable for severe distortion, and the added 1 avoids negative values. A lower distortion value indicates that the geometric structure of a frame is better preserved after stabilization. For a video, we compute its distortion by averaging the distortion values of all the frames. Please refer to the Section 4.1 of [7] for more details about the energy function (14).

Stability Similar to distortion, we also use the objective function from previous method as a metric to measure the stability of smoothed trajectories. In [4], Grundmann et al. stabilize a video by minimizing the combination of the the 1st, the 2nd, and the 3rd deviation of the target camera path. In our experiments, stability is measured by the combination of the 1st and the 2nd deviation of stabilized trajectories, which represent speed and acceleration respectively. In practice, we multiply the 2nd deviation by 10 and add it to the 1st deviation as the metric for stability. It is because unstable motion is mainly caused by the change on speed (acceleration).

With the three metrics defined, we made quantative comparisons sequence by sequence. Our local subspace video stabilization and subspace video stabilization shared the same gaussian kernel, whose deviation and radius were 32 and 25.

As shown in Fig 5(c), the stability of smoothed trajectories in our method is slightly better than the results from subspace video stabilization. But this subpixel-wise difference is visually unconspicuous in some sequences because we used the same gaussian kernel for both of the two methods. However, our method outperforms Liu et al.'s in terms of cropping and distortion. See Fig 5(a) and 5(b), our local subspace video stabilization crops less video content and better preserves the geometric structure of video frames for all the test sequences. Moreover, we have successfully stabilized the sequence iii — a failure case in [8], where subspace video stabilization is failed by the outliers on the girl's moving face. In our method, the trajectories on the face have been detected and rejected by our novel outlier detection technique, and the remaining trajectories after local factorization are still sufficient for content-preserving warps. As shown in Fig 6(a), the public result from Liu et al.'s method is severely distorted, but our stabilized video is much more visual plausible.

At the same time, our results are also comparable or even better than the ones from some other state-of-the-art methods. As shown in Fig 6(b), much more video content is preserved in our method than L1 optimal camera paths [4]. In Fig 6(c), the background trees shift a little to the right in the result of bundled camera paths [10] while our result is a little better. Note that our method preserves more video content than all of the three competitors. **Please see the supplementary video for more comparisons.**

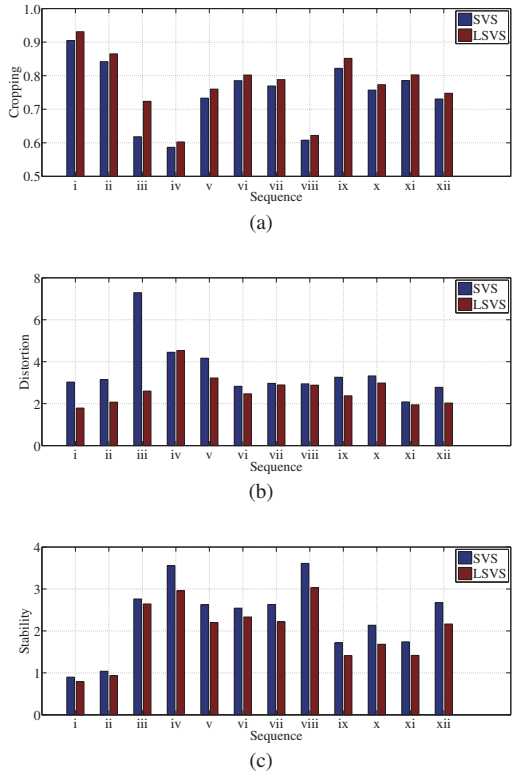


Fig. 5. Comparisons with our implementation of subspace video stabilization. Comparisons on (a) cropping, (b) distortion, and (c) stability. The sequence iii is a failure case in [8], but we have successfully stabilized it using our local subspace video stabilization method.

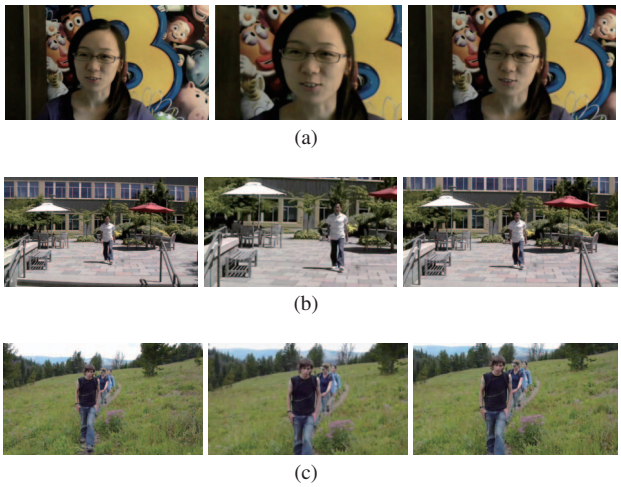


Fig. 6. Comparisons with some public results. Left: input frames. Middle: stabilized frames from (a) subspace video stabilization, (b) L1 optimal camera paths and (c) bundled camera paths. Right: our stabilized frames.

Fig. 7. Failure cases. From left to right: motion blur, severe occlusion, and large plane area.

4.3. Performance

We implemented our local subspace video stabilization in C++ and did all the experiments on a MacBook Air with 1.7GHz Intel Core i5 CPU and 4GB RAM. The SVD was implemented using Lanczos based algorithm.We tracked about 500 points in a 640×360 frame using KLT tracker, which runs in 9 fps. Our local factorization achieves 30 fps when factors the tracked trajectory matrix, and our implementation of content-preserving warps takes 40 ms to warp a stabilized frame. Overall, our stabilization system runs at about 5 fps.

4.4. Discussion

In general, the proposed method has three superiorities over subspace video stabilization. First, it preserves more points for content-preserving warps. Second, it factors the trajectory matrix more accurately for both existing and missing data. At last, it can reject some outliers that fail subspace video stabilization. Although our local factorization is slower than moving factorization (30 fps vs. 500 fps [8]), the overall speed of our system is comparable with subspace video stabilization because the main bottleneck is feature tracking. Benefiting from the locality of the model, our method is highly parallel, we can easily accelerate it on GPU platforms. However, like subspace video stabilization, our method is fragile in videos with sever occlusion, motion blur, and large plane area. It is caused by the limitation of the KLT tracker. In the sequences shown in Fig 7, we can not accumulate enough trajectories for factorization in some frames. In the future, we will try to deal with this problem by splitting a video into several segments and stabilizing them separately.

5. CONCLUSION

In this paper, we have proposed a novel method for video stabilization. It smooths every column of a trajectory matrix in separate local subspace. By utilizing the local subspace constraint, our factorization is more accurate than moving factorization [8]. It also preserves more points for content-preserving warps, which is crucial to the quality of stabilized videos. Moreover, we have designed a novel outlier detection technique that utilizes the relationship between consecutive local subspaces, and it can reject the outliers that fail subspace video stabilization [8] while maintain computational efficiency. The experiments show that our method not only outperforms subspace video stabilization but also is comparable with some other state-of-the-art methods.

Acknowledgements. This work is partly supported by the grant of NSFC 61360115 and Shenzhen Peacock Plan.

6. REFERENCES

[1] F. De la Torre and M. Black. Robust principal component analysis for computer vision. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 362–369 vol.1, 2001. 3

[2] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–6, 2007. 1, 2

[3] A. Goldstein and R. Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 31(5):126:1–126:10, Sept. 2012. 1, 4

[4] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 225–232, 2011. 1, 4, 5

[5] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 626–633 vol.1, 1999. 1, 2

[6] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung. Video stabilization using robust feature trajectories. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1397–1404, 2009. 1

[7] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3):44:1–44:9, July 2009. 1, 2, 4, 5

[8] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1):4:1–4:10, Feb. 2011. 1, 2, 3, 4, 5, 6

[9] F. Liu, Y. Niu, and H. Jin. Joint subspace video stabilization. In *Computer Vision, 2013 IEEE 14th International Conference on*, 2013. 1

[10] S. Liu, L. Yuan, P. Tan, and J. Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4):78:1–78:10, July 2013. 1, 4, 5

[11] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1150–1163, 2006. 1, 4

[12] Y. Shen, Z. Wen, and Y. Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization, 2011. 3

[13] Y.-S. Wang, F. Liu, P.-S. Hsu, and T.-Y. Lee. Spatially and temporally optimized video stabilization. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1354–1361, Aug. 2013. 1

[14] Z. Zhou, H. Jin, and Y. Ma. Plane-based content preserving warps for video stabilization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2299–2306, 2013. 1