SPARSE MOVING FACTORIZATION FOR SUBSPACE VIDEO STABILIZATION

Chengzhou Tang, Ronggang Wang

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

ABSTRACT

This paper presents a new method for calculating the low-rank approximation of a highly incomplete trajectory matrix for subspace video stabilization. We extend moving factorization proposed in [1], which is a streamable method based on least squares. By utilizing sparse representation of trajectories, the proposed factorization method is more accurate while still streamable. We test our sparse moving factorization on synthetic data as well as real videos. Experiments on synthetic sequence demonstrate the numerical properties of our method, and stabilized videos show that our method outperforms moving factorization for subspace video stabilization. In addition, our results are also better than the ones from some other state-of-the-art video stabilization methods.

Index Terms— Sparse Representation, Matrix Factorization, Video Stabilization

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 the camera motion. In film industry and other professional fields, cameramen use dollies or steadicams to get stable motion. These hardware solutions are not practical for ordinary users, so video stabilization software is usually used in post processing stage.

Previous video stabilization methods first estimate and smooth 2D camera motion [2, 3, 4] or 3D camera motion [5, 6], then synthesize a stabilized video using 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 parallax for lack of model flexibility. In contrary, the 3D methods can handle parallax and produce visual plausible results, but they are rarely used in commercial softwares for unreliability under various conditions such as motion blur, large moving object, camera zoom and rolling shutter. On another hand, the computational complexity of recovering 3D information is unacceptable. As reported in [5], structure-from-motion using Voodoo Tracker [1] takes several hours for a short video.

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

Recently, both 2D and 3D methods have been greatly improved. In [7], S. Liu et al. use bundled camera paths, which are more flexible than a single global camera path. They smooth the estimated camera paths using space-time optimization and warp frames with spatially-variant homographies. Wang et al. [8] 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 [9] avoid 3D reconstruction by using the 'epipolar transfer' technique to construct and smooth virtual trajectories.

In this work, we focus on subspace video stabilization proposed in [1], where Liu et al. develop a streamable method named moving factorization to factor a highly incomplete trajectory matrix. They obtain a stable trajectory matrix by smoothing and projecting the factored eigen-trajectories back to original 2D coordinate space. Differently, we novelly factor new trajectories from calculated results by utilizing sparse representation of trajectories. Compared with moving factorization, our method extends trajectories more accurately, which is beneficial to the quality of stabilized videos.

In the rest of this paper, we first revisit subspace video stabilization to give a better understanding of the basic concept. Then we introduce the sparse representation of trajectories and describe the proposed algorithm. At last, we give experimental results on synthetic data and real videos to validate our method.

2. SUBSPACE VIDEO STABILIZATION

In subspace video stabilization, a set of feature points are tracked through a video. The trajectories of these points are assembled into a trajectory matrix $\mathbf{M}$:

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

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

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

where $\mathbf{W}$ is a binary mask with 0 for missing data, 1 for existing data, and $\odot$ indicates the element-wise multiplication. Then the stable

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

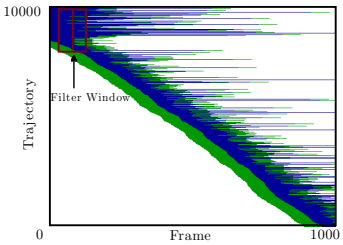


Fig. 1. A typical trajectory matrix from real video. About 84% data in this matrix are missing. *Blue*: tracked trajectories. *Green*: extended segments. *Red*: low-pass filter window of w frames. Since there are much more trajectories than frames, we scale the y-axis for visualization.

trajectory matrix $\widetilde{\mathbf{M}}$ is obtained by adopting a gaussian kernel $\mathbf{K}$ to $\mathbf{E}$:

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

and stabilized frames are synthesized using content-preserving warps [5] with the correspondence between $\mathbf{M}$ and $\widetilde{\mathbf{M}}$.

In [1], Liu et al. use the moving factorization technique to factor $\mathbf{M}$ into coefficient matrix $\mathbf{C}$ and eigen-trajectories $\mathbf{E}$. They present that to support the low-pass filter, the factorization needs to be able to extend each trajectory forwards and backwards in time by the radius of the smoothing kernel. In order to get $\widetilde{\mathbf{M}}$, $\mathbf{M}$ is smoothed by projecting filtered eigen-trajectories back to the original 2D coordinate space as in (3). It is equivalent to filtering the points near the start and the end of original trajectories with both tracked and extended neighbors. So the factorization should be accurate for not only the tracked trajectories but also the extended segments. To achieve this goal, we present our sparse moving factorization in the next section, which extends points more accurately as described in Section 4.1.

3. SPARSE MOVING FACTORIZATION

3.1. Sparse representation of trajectories

Sparse representation is based on the idea that a vector can be represented as a sparse linear combination of a set of basis vectors. For motion segmentation, Elhamifar et al. represent each trajectory as a sparse linear combination of the other trajectories from the same subspace [10]. In subspace video stabilization, we only consider the trajectories of static background, which belong to a single subspace. Thus, we can represent a trajectory $\boldsymbol{y}$ as:

$$\boldsymbol{y} = \sum_{i=1}^{n} a_i \boldsymbol{x}_i, \tag{4}$$

where $\boldsymbol{x}_i$ is one of the other n trajectories that cover $\boldsymbol{y}$ in time. As reported in [11], one can recover the K-sparse (have no more than K non-zero values) vector $\boldsymbol{a} = [a_1 \cdots a_n]^\top$ if $K \lesssim m/\log(D/m)$. And $\boldsymbol{a}$ is usually obtained by solving the Lasso problem:

$$\min \|\boldsymbol{y} - \mathbf{X}\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1. \tag{5}$$

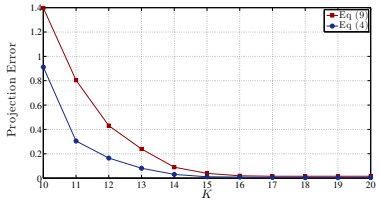


Fig. 2. Projection error under different sparsity level K. *Blue*: projection error by representing a trajectory as a sparse linear combination of all the other trajectories as in (4). *Red*: projection error by representing a new trajectory as a sparse linear combination of factored trajectories as in (9).

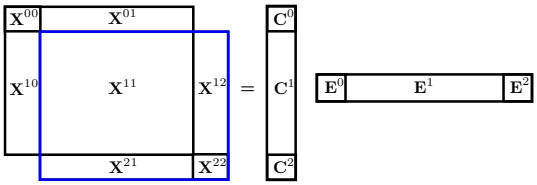


Fig. 3. Moving factorization via sparse representation. To compute the factorization for the next window forward δ frames, we fix $\mathbf{C}^0$, $\mathbf{C}^1$, $\mathbf{E}^0$ and $\mathbf{E}^1$, then compute $\mathbf{C}^2$ from $\mathbf{C}^1$ by utilizing sparse representation of trajectories.

In practice, we investigate this property in a filter window of w frames (we set w to 50). Here, the ambient space dimension D equals to the trajectory number n in the window, and the measurement number $m = 2r$ if trajectories are projected onto lower dimensional subspace as in (2). In real sequences, D is usually larger than 50. If we project the trajectory matrix onto a rank 9 subspace as in [1], $\boldsymbol{a}$ will be at least $18/\log(50/18) \approx 18$-sparse.

To validate the sparse representation of trajectories, sparsity level K is altered by truncating $\boldsymbol{a}$ except for the entries that have the K largest absolute values. Then we investigate how the average projection error $\|\boldsymbol{y} - \mathbf{X}\boldsymbol{a}\|_2$ varies along with the change of K. Fig 2 shows that the error decreases as K increases, and converges to 0.062 when K=17. So in our application, the sparse representation of trajectories is available both theoretically and experimentally.

3.2. Moving factorization via sparse representation

As the same with moving factorization, we start our algorithm by assembling k trajectories that span the first w frames into a complete matrix $\mathbf{M}^0$. This matrix can be factored as follows:

$$\mathbf{M}^0_{2k\times w} = \mathbf{C}_{2k\times r}\mathbf{E}_{r\times w}, \tag{6}$$

where $\mathbf{C}$ and $\mathbf{E}$ are obtained by truncating the output of Singular Value Decomposition (SVD) [12] to the rows, columns, and values corresponding to the largest r singular values, and then distributing the square root of singular value to the left and right orthogonal matrices.

Now we move the factorization window forward δ frames (we set δ to 5) to reach the next complete matrix $\mathbf{M}^1$. As shown in Fig

3, $\mathbf{X}^{11}$ is shared by $\mathbf{M}^0$ and $\mathbf{M}^1$ after matrix permutation. Since $\mathbf{X}^{11} = \mathbf{C}^1\mathbf{E}^1$ is already known when $\mathbf{M}^0$ is factored, we keep this value fixed and factor $\mathbf{M}^1$ as:

$$\mathbf{M}^1 = \begin{bmatrix} \mathbf{X}^{11} & \mathbf{X}^{12} \\ \mathbf{X}^{21} & \mathbf{X}^{22} \end{bmatrix} = \begin{bmatrix} \mathbf{C}^1 \\ \mathbf{C}^2 \end{bmatrix} \begin{bmatrix} \mathbf{E}^1 & \mathbf{E}^2 \end{bmatrix}. \quad (7)$$

In moving factorizaiton, Liu et al. estimate $\mathbf{C}^2$ by minimizing the projection error $\|\mathbf{C}^2\mathbf{E}^1 - \mathbf{X}^{21}\|_F^2$:

Different from their method, our algorithm calculates $\mathbf{C}^2$ from $\mathbf{C}^1$ by utilizing the sparse representation of trajectories. We write $\mathbf{X}^1([\mathbf{X}^{11}, \mathbf{X}^{12}])$ and $\mathbf{X}^2([\mathbf{X}^{21}, \mathbf{X}^{22}])$ as:

$$\mathbf{X}^1 = \begin{bmatrix} \boldsymbol{x}_1^1 \\ \vdots \\ \boldsymbol{x}_m^1 \end{bmatrix}_{2m\times w} \text{and } \mathbf{X}^2 = \begin{bmatrix} \boldsymbol{x}_1^2 \\ \vdots \\ \boldsymbol{x}_n^2 \end{bmatrix}_{2n\times w}, \quad (8)$$

where $\boldsymbol{x}_j^1$ is a $2\times w$ matrix of a trajectory factored in $\mathbf{M}^0$ while $\boldsymbol{x}_i^2$ belongs to a new trajectory in $\mathbf{M}^1$. Since n is much smaller than m, $\boldsymbol{x}_i^2$ can be represented as:

$$\boldsymbol{x}_i^2 = \sum_{j=1}^{m} a_{ij}\boldsymbol{x}_j^1, \quad (9)$$

which is a good approximation to (4) as shown in Fig 2.

Then we transfer (9) to $\mathbf{C}^1$ and $\mathbf{C}^2$ because they are the linear projection of $\mathbf{X}^1$ and $\mathbf{X}^2$ on the same basis $[\mathbf{E}^1, \mathbf{E}^2]$:

$$\boldsymbol{c}_i^2 = \sum_{j=1}^{m} a_{ij}\boldsymbol{c}_j^1. \quad (10)$$

Therefore, $\mathbf{C}^2$ can be computed from $\mathbf{C}^1$ once every sparse vector $\boldsymbol{a}_i = [a_{i1}\cdots a_{im}]^\top$ is given.

In order to calculate $\{\boldsymbol{a}_i\}_{i=1\cdots n}$, we project $\mathbf{M}^1$ onto the first-r principal subspace by SVD as:

$$\mathbf{M}^1_{2(m+n)\times w} = \mathbf{U}_{2(m+n)\times r}\mathbf{\Sigma}_{r\times r}\mathbf{V}^\top{}_{r\times w}, \quad (11)$$

and recover the sparse representation of trajectories on $\mathbf{U}$. Since $\mathbf{U} = \begin{bmatrix} \mathbf{U}^1 \\ \mathbf{U}^2 \end{bmatrix}$ also represents $\begin{bmatrix} \mathbf{X}^1 \\ \mathbf{X}^2 \end{bmatrix}$ in lower dimensional subspace, $\boldsymbol{a}_i$ can be obtained by solving

$$\min\|\bar{\boldsymbol{u}}_i^2 - \bar{\mathbf{U}}^1\boldsymbol{a}_i\|_2^2 + \lambda\|\boldsymbol{a}_i\|_1, \quad (12)$$

where $\bar{\boldsymbol{u}}_i^2$ is a $2r\times 1$ vector reshaped from the $2\times r$ matrix $\boldsymbol{u}_i^2$, and $\bar{\mathbf{U}}^1 = [\bar{\boldsymbol{u}}_1^1\cdots\bar{\boldsymbol{u}}_m^1]$.

Finally, we construct $\mathbf{C}^2$ by computing $\{\boldsymbol{c}_i^2\}_{i=1\cdots n}$ using (10), and solve for $\mathbf{E}^2$ as the same in moving factorization:

$$\mathbf{E}^2 = \left(\begin{bmatrix} \mathbf{C}^1 \\ \mathbf{C}^2 \end{bmatrix}^\top \begin{bmatrix} \mathbf{C}^1 \\ \mathbf{C}^2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{C}^1 \\ \mathbf{C}^2 \end{bmatrix}^\top \begin{bmatrix} \mathbf{X}^{12} \\ \mathbf{X}^{22} \end{bmatrix}. \quad (13)$$

In this way, we can get the factorization of $\mathbf{M}$ in (2) by moving the factorization window forward until all the trajectories have been processed. For the trajectories that are too short to be included in a window, their coefficients are computed by minimizing the projection error. Before factoring every window, outliers are detected and rejected using the same method in [1].

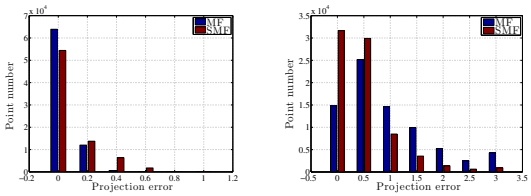


Fig. 4. Comparison on synthesized data. (a) The bar graph of in-frame point number by projection error. (b) The bar graph of extended point number by projection error.

4. EXPERIMENTAL RESULTS

4.1. Synthetic data

This section aims at obtaining a better understanding of why our method outperforms moving factorization for subspace video stabilization. We randomly generated 500 points within a $100\times100\times100$ 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 were set from $-\pi/4$ to $+\pi/4$ randomly. Then we added gaussian noise with $\sigma = 3$ and $\mu = 0$ to every projected point in frames.

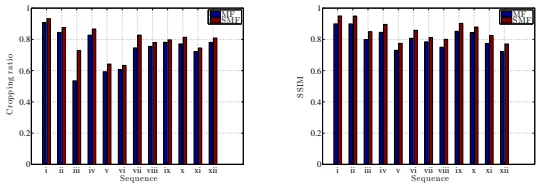
After factoring the synthesized trajectory matrix by moving factorization (MF) and sparse moving factorization (SMF), we counted point number according to the projection error of every point. Since our algorithm doesn't explicitly minimize projection error when calculates coefficient matrix, it performs slightly worse than moving factorization on the in-frame points. Fig 4(a) shows that there are more points with projection error larger than 0.1 pixels in our method. But this difference is negligible — the average projection error is 0.0637 pixels by moving factorization and 0.0984 pixels by our method. They are both accurate enough for subspace video stabilization as reported in [1].

However, our method outperforms moving factorization greatly on extended points. As shown in Fig 4(b), most extended points in our method have projection error less than 1 pixels while moving factorization generates more points with larger error. The average projection error of extended points is 0.5089 by our method, which is about half of the error by moving factorization. It is because our method better preserves the relationship between trajectories by adopting the sparse representation of trajectories into factorization. As discussed in Section 2, accurately extended points are crucial to smoothing the trajectory matrix.

4.2. Video Stabilization

In this section, we test our method on video stabilization with 12 video sequences shown in Fig 5. These videos cover various scene types including camera zoom, dynamic scene, large parallax, large moving object, etc. We quantatitively compare the proposed method and our implementation of moving factorization in subspace video stabilization. Additionally, subjective comparisons with other methods are given by snapshots and online supplementary video.

Fig. 5. Snapshots of the 12 video sequences used in experiments.



(a) (b)

Fig. 6. Comparisons with subspace video stabilization. (a) Cropping ratio. (b) SSIM between consecutive frames.

For quantitative comparisons, we use two objective metrics:

Cropping Since the scope of every synthesized frame is different, videos will have blank areas after stabilization. Matsushita et al. propose motion inpainting to fill the blank [2], but their method is rarely used in real applications for complexity and robustness issues. Most state-of-the-art methods unify frames' size by cropping 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.

SSIM Although some sophisticated metrics have been proposed to measure the quality of stabilized videos [7, 13], similarity between consecutive frames by PSNR is still popular for its simplicity [14, 15]. In our experiments, we measure the similarity of consecutive frames by SSIM [16], which is more advanced than PSNR and still easy to use. The more stable a video is the more similar its consecutive frames are, and less temporal distortions between frames will also increase the SSIM value. So satisfactory stabilized videos have high SSIM values.

We set the parameters of the gaussian kernel to σ=24 and w=50 when smooth the eigen-trajectories factored by moving factorization and our method. The two metrics are used in sequence-by-sequence comparisons on the videos synthesized by content-preserving warps. As Fig 6 shows, our method preserves more image content than moving factorization, and the higher SSIM values indicate that stabilized videos by our method are more stable and less distorted.

Then we compare our stabilized videos with some public results. In Fig 7(a), we successfully stabilize a failure case of subspace video stabilization [1]. As shown in Fig 7(b), much more video content is preserved in our method than L1 optimal camera path [4]. In Fig 7(c), the light pole in the result of epipolar video stabilization [9] is distorted while ours is better. Note that our method preserves more video content than all of the three competitors. Please see the supplementary video (http://youtu.be/ljCXEqv_1rg) for more comparisons.

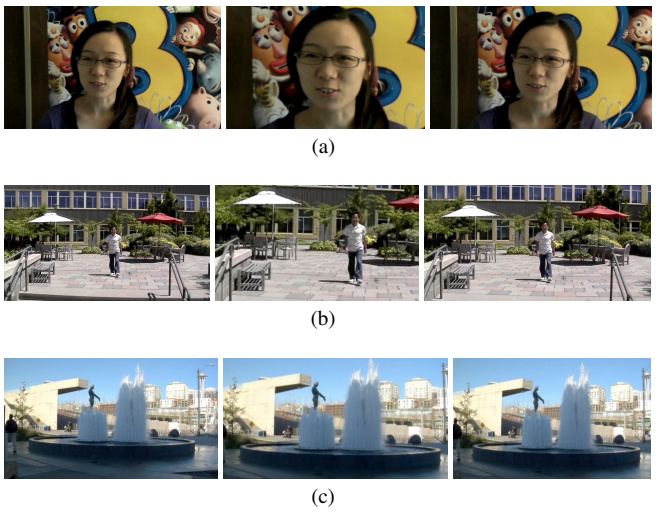


Fig. 7. Comparisons with public results. Left: input frames. Middle: stabilized frames from (a) subspace video stabilization, (b) L1 optimal camera path and (b) epipolar video stabilization. Right: our stabilized frames.

4.3. Performance

We implemented the proposed algorithm in C++ and did all the experiments on a MacBook Air with 1.7GHz Intel Core i5 CPU and 4GB RAM. The minimization in (12) is solved by Homotopy solver [17], and Lanczos based algorithm from SVDLIBC[2] is used in (6) and (11). In video stabilization, we track feature points using KLT tracker in OpenCV[3], which achieves 9 fps when tracks about 500 points in a 640×360 frame. The sparse moving factorization achieves 35 fps when factor tracked trajectory matrix, and our implementation of content-preserving warps takes about 40 ms to warp a frame. Overall, our video stabilization system runs at about 5 fps.

5. CONCLUSIONS

In this paper we have proposed a factorization method that outperforms moving factorization for subspace video stabilization. We use sparse representation to model the relationship between trajectories and novelly adopt this model into factorization. Our method follows the streamable framework of moving factorization, but it better extends the trajectories by the radius of the filter window. Consequently, our stabilization method generates more visual plausible results than the original subspace video stabilization and some other existing methods.

[2] http://tedlab.mit.edu/~dr/SVDLIBC/
[3] http://opencv.org

6. REFERENCES

[1] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1):4:1–4:10, February 2011.

[2] Y. Matsushita, E. Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1150–1163, 2006.

[3] Ken-Yi Lee, Yung-Yu Chuang, Bing-Yu Chen, and Ming Ouhyoung. Video stabilization using robust feature trajectories. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1397–1404, 2009.

[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.

[5] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3):44:1–44:9, July 2009.

[6] Zihan Zhou, Hailin Jin, and Yi Ma. Plane-based content preserving warps for video stabilization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2299–2306, 2013.

[7] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4):78:1–78:10, July 2013.

[8] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee. Spatially and temporally optimized video stabilization. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1354–1361, August 2013.

[9] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 31(5):126:1–126:10, September 2012.

[10] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797, 2009.

[11] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[13] Chao Zhang, P. Chockalingam, A. Kumar, P. Burt, and A. Lakshmikumar. Qualitative assessment of video stabilization and mosaicking systems. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, pages 1–6, 2008.

[14] Junlan Yang, D. Schonfeld, and M. Mohamed. Robust video stabilization based on particle filter tracking of projected camera motion. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(7):945–954, 2009.

[15] Sanjeev Kumar, H. Azartash, M. Biswas, and Truong Nguyen. Real-time affine global motion estimation using phase correlation and its application for digital image stabilization. *Image Processing, IEEE Transactions on*, 20(12):3406–3418, 2011.

[16] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[17] A.Y. Yang, S.S. Sastry, A. Ganesh, and Yi Ma. Fast l1-minimization algorithms and an application in robust face recognition: A review. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1849–1852, 2010.