Real-Time 3D Rotation Smoothing for Video Stabilization

Chao Jia, Zeina Sinno, and Brian L. Evans
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX USA
Email: cjia@utexas.edu, zeina@utexas.edu, bevans@ece.utexas.edu

***Abstract*—We propose two real-time motion smoothing algorithms for video stabilization using a pure 3D rotation motion model with known camera projection parameters. Both proposed algorithms aim at smoothing 3D rotation matrix sequences in a causal way. The first algorithm smooths the 3D rotation sequences in a way similar to 1st-order IIR filtering. The second algorithm uses sequential probabilistic estimation under a constant angular velocity model. These two algorithms are generalized from classical 2D motion smoothing algorithms. We exploit the manifold structure of the rotation matrices so that the proposed algorithms directly smooth the 3D rotation sequences on the manifold. In addition, we introduce a simple projection step in order to guarantee that no black borders intrude into the stabilized video frames. Experimental results show that the proposed algorithms are able to effectively stabilize video sequences and outperform their 2D counterparts with less jitter and distortion.**

I. Introduction

Digital video stabilization seeks to remove the unwanted frame-to-frame jitter and generate visually stable and pleasant videos. In general, digital video stabilization consists of three major steps, namely motion estimation, motion smoothing and frame synthesis. This paper focuses on the second step.

Given the estimated camera motion for each frame, motion smoothing aims at designing a new smooth camera motion path. Most existing works address motion smoothing as an offline processing after the entire video sequence has been recorded. However, real-time video stabilization is necessary for applications such as video conferencing and broadcasting. Besides, for just video recording, real-time stabilization can greatly improve the user experience with the stabilized videos displayed in real-time on the viewfinders. Real-time video stabilization is also able to reduce the memory requirements with frames stabilized before compression. In real-time video stabilization, camera motion is required to be smoothed in a causal way. This is more difficult than offline motion smoothing because we are missing information of how camera motion changes afterward.

In this paper, we focus on real-time motion smoothing based on a 3D rotational camera motion model for a calibrated camera. Compared to 2D (translational, similarity, or affine) motion models, 3D motion models can reflect the real camera projection more accurately, and thus give more realistic motion smoothing and avoid image distortion in frame synthesis. We ignore 3D translation of the camera because (1) the unwanted jitter in videos are primarily caused by camera rotation, and (2) frame synthesis with 3D camera translation would need the depth value at every pixel, which is very difficult to obtain accurately. To estimate the 3D camera rotation we use a gyroscope that is available in many smart phones and tablets. Current gyroscopes in smart phones have very high precision and can return more reliable 3D camera rotation estimates compared to vision-based camera motion estimation, especially when there are many moving objects in the scene or there is severe motion blur or illumination changes.

Under a 3D rotational model, camera motion for a video can be considered as a sequence of 3D rotation matrices. We propose two algorithms for real-time 3D rotation smoothing, generalized from two classical 2D motion smoothing algorithms. The first algorithm smooths the 3D rotation sequence in a way similar to 1st-order IIR filtering. The second algorithm works as sequential estimation with a constant angular velocity model. We exploit the manifold structure of the rotation matrices so that the proposed algorithms directly smooth the 3D rotation sequences on the manifold.

Due to the camera motion change from motion smoothing, some areas in the synthesized frame will be undefined. This is known as black border problem. In practice we have to crop the resulting video frames. Still, in motion smoothing, we have to constrain the change of camera motion in order to guarantee that no black borders intrude into the stabilized video frames. This is achieved by adding a projection step for each frame.

II. Related Work

Camera motion has been commonly modeled using 2D models. Using full 3D models including both rotation and translation for calibrated cameras was first proposed in [1] and further discussed in [2]. In both papers complicated approximations are used in frame synthesis to handle the problem of missing depth values. In [3] and [4] pure 3D rotational models were shown to generate high-quality results while only needing homography-based warping in frame synthesis.

Whether 2D or 3D rotational motion models were used, many (if not most) existing motion smoothing algorithms are offline smoothing. Local Gaussian window filtering was used under 2D affine model in [5] and under 3D rotational model in [6]. Another kind of algorithms smooth the camera motion via minimizing a certain objective function that represents the smoothness of the camera motion trajectory. Such methods were applied on both 2D motion models [7] and 3D rotational

motion models [8]. Some other work directly smooths the feature trajectories instead of camera motion [9].

Prior work on real-time motion smoothing was only restricted to 2D motion models. In [10] IIR filtering was proposed for online motion smoothing based on 2D translational motion model. In [11], the intentional motion parameters (under 2D translational motion model) were modeled by a constant velocity linear system so Kalman filtering could be used to optimally estimate them. The same Kalman-filtering motion smoothing framework was extended to 2D affine motion model in [12], leading to a better performance. The black-border constraints were rarely considered in online motion smoothing. In [13] the authors first proposed to use constrained Kalman filtering for 2D translational motion model.

III. 3D Rotation and Geodesic Distance

All of the 3×3 rotation matrices constitute the Special Orthogonal Group $\mathbf{SO(3)}$, in which any element $\mathbf{R}$ satisfies the constraint $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$. $\mathbf{SO(3)}$ can be also considered as an embedded Riemannian submanifold of Euclidean space $\mathbb{R}^9$ (represented as 3×3 real matrices). A natural extension of Euclidean distance in Euclidean space to the Riemannian manifold $\mathbf{SO(3)}$ is the geodesic distance

$$d_g(\mathbf{R}_i, \mathbf{R}_j) = ||\mathrm{logm}(\mathbf{R}_i^{\mathrm{T}}\mathbf{R}_j)||_F, \tag{1}$$

where $\mathrm{logm}(\cdot)$ is the matrix logarithm operator and $||\cdot||_F$ is the Frobenius norm of a matrix. In fact, $\mathrm{logm}(\mathbf{R}_i^{\mathrm{T}}\mathbf{R}_j)$ is a skew-symmetric matrix representing a tangent vector in the tangent space $T_{\mathbf{R}_i}\mathbf{SO(3)}$ that indicates the non-normalized direction from $\mathbf{R}_i$ to $\mathbf{R}_j$ on $\mathbf{SO(3)}$. Usually we also write $\mathrm{logm}(\mathbf{R}_i'\mathbf{R}_j)$ as $\log_{\mathbf{R}_i}\mathbf{R}_j$ and name it the logarithmic mapping. Inversely, given a vector $\xi \in T_{\mathbf{R}_i}\mathbf{SO(3)}$, we can define the exponential mapping as $\exp_{\mathbf{R}_i}\xi = \mathbf{R}_i\mathrm{expm}(\xi)$, where $\mathrm{expm}(\cdot)$ is the matrix exponential operator. This is used to move $\mathbf{R}_i$ along the direction defined by ξ on $\mathbf{SO(3)}$. The logarithmic mapping and exponential mapping together define a curve

$$t \in [0,1] \mapsto \gamma(t) = \exp_{\mathbf{R}_i}\left(t \cdot \log_{\mathbf{R}_i}\mathbf{R}_j\right), \tag{2}$$

which is known as the minimizing geodesic from $\mathbf{R}_i$ to $\mathbf{R}_j$. It is a generalization of the notion of "straight line" in Euclidean space to Riemannian manifolds, representing the shortest path between two points on the manifolds. The length of the minimizing geodesic is defined in (1).

IV. Black-Border Constraint and Estimate Projection

In the last step of video stabilization, the synthesized frames may contain black borders since not every pixel in the synthesized frame is visible in the original frame due to the change of camera orientation. Therefore, we have to crop the synthesized frames into a smaller size so that there are no black borders. Given a preferred size of the stabilized video, the video stabilization system must guarantee that every pixel in the cropped stabilized frames is visible in the original frames. This is a hard constraint that has to be considered in the camera motion smoothing algorithm.

Assume the intrinsic projection matrix of the camera is given as $\mathbf{K}$. Under pure 3D camera rotation, for any pixel $[u_{kj}, v_{kj}]^{\mathrm{T}}$ in the stabilized frame k, its corresponding 2D pixel location in the original frame $[\tilde{u}_{kj}, \tilde{v}_{kj}]^{\mathrm{T}}$ can be computed as

$$\begin{bmatrix}\tilde{u}_{kj}\\ \tilde{v}_{kj}\end{bmatrix} = g\left(\mathbf{K}\mathbf{R}_k\hat{\mathbf{R}}_k^{-1}\mathbf{K}^{-1}\begin{bmatrix}u_{kj}\\ v_{kj}\\ 1\end{bmatrix}\right), \tag{3}$$

where the function

$$g([x,y,z]^{\mathrm{T}}) = [x/z, y/z]^{\mathrm{T}} \tag{4}$$

is used to convert the homogeneous coordinates into inhomogeneous coordinates. Assume that the frame size in the original video is $w \times h$, and the coordinates of the top left corner and bottom right corner of the cropped rectangle in the stabilized video are $[c_1, d_1], [c_2, d_2]$, the hard constraint for video stabilization can be represented as

$$\begin{cases}0 \le \tilde{u}_{kj} \le w\\ 0 \le \tilde{v}_{kj} \le h\end{cases}, \forall \begin{bmatrix}u_{kj}\\ v_{kj}\end{bmatrix} s.t. \begin{cases}c_1 \le u_{kj} \le c_2\\ d_1 \le v_{kj} \le d_2\end{cases} \tag{5}$$

No matter what real-time rotation smoothing method is used, the estimated smoothed rotation for each frame may violate the constraint (5). Therefore, the original rotation estimate has to be projected onto the constraint set. The constraint (5) is very complex with respect to the rotation matrices that we want to compute and no algorithms as far as we know are guaranteed to handle it efficiently. We propose an simple approximate projection method:

$$\hat{\mathbf{R}} = \mathbb{P}(\hat{\mathbf{R}}^*) = \mathbf{R}\mathrm{expm}(\beta^*\mathrm{logm}(\mathbf{R}^{-1}\hat{\mathbf{R}}^*)), \tag{6}$$

where $\beta^* \in [0,1]$ is the maximum possible value so that the projection result satisfies constraint (5). $\mathbf{R}$ is the original rotation matrix and $\hat{\mathbf{R}}^*$ is the initial estimate of the smoothed rotation matrix. In other words, we only search along the direction defined by the tangent vector $\mathrm{logm}(\mathbf{R}^{-1}\hat{\mathbf{R}}^*)$. The proposed projection returns the exact solution if the constraint set is a geodesic ball around $\mathbf{R}$, which is a good approximation of the constraint set (5). In practice, β^* can be efficiently found by bisection search. During the search, whether the constraint is satisfied can be evaluated only using the four corner points of the cropped rectangle, because (3) is a homography transformation. The projection (6) can thus be implemented very fast. In the following sections we will use it to keep the online smoothing results satisfying the black-border constraint.

V. Rotation Smoothing via IIR Filtering

Assuming the 2D motion parameter for each frame k in the original video is $\boldsymbol{\theta}_k$, 1st-order IIR smoothing calculates the smoothed motion parameter $\hat{\boldsymbol{\theta}}_k$ by

$$\hat{\boldsymbol{\theta}}_k = \alpha\hat{\boldsymbol{\theta}}_{k-1} + (1-\alpha)\boldsymbol{\theta}_k, \tag{7}$$

where $\alpha \in [0,1]$ is the smoothing coefficient. $\boldsymbol{\theta}_k$ is a vector in multi-dimensional Euclidean space. For instance, if 2D affine motion model is used, the dimension of $\boldsymbol{\theta}_k$ is six. If we

use 3D rotational camera motion model, however, (7) is not appropriate because it is defined based on distance measure in Euclidean space instead of $\mathbf{SO(3)}$ manifold. In fact, the weighted sum on the right hand side of (7) does not necessarily return a valid 3D rotation matrix.

The 1st-order IIR filtering in (7), however, can be interpreted in another way as

$$\hat{\boldsymbol{\theta}}_k = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \alpha||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{k-1}||^2 + (1-\alpha)||\boldsymbol{\theta} - \boldsymbol{\theta}_k||^2. \quad (8)$$

In other words, $\hat{\boldsymbol{\theta}}_k$ is a linear interpolation between $\hat{\boldsymbol{\theta}}_{k-1}$ and $\boldsymbol{\theta}_k$ based on Euclidean distance. Given the geodesic distance defined on $\mathbf{SO(3)}$, we can naturally extend (8) as

$$\hat{\mathbf{R}}_k = \underset{\mathbf{R}}{\operatorname{argmin}}\, \alpha d_g(\mathbf{R}, \hat{\mathbf{R}}_{k-1})^2 + (1-\alpha) d_g(\mathbf{R}, \mathbf{R}_k)^2. \quad (9)$$

This is just a linear interpolation between $\hat{\mathbf{R}}_{k-1}$ and $\mathbf{R}_k$ on $\mathbf{SO(3)}$ based on the geodesic distance. It has been show that such interpolation is equivalent to spherical linear interpolation (slerp) on unit quaternion representation of 3D rotation matrices [14], which can be computed very fast. The interpolation result for every frame is projected by (6) before being used for the next frame. Algorithm 1 shows the proposed IIR-like 3D rotation smoothing. We use unit quaternions to represent the 3D rotations instead of matrices, but the representations can be easily converted from one to the other.

Algorithm 1 IIR-like 3D Rotation Smoothing

1: **Input:** $\mathbf{q}_1, \cdots, \mathbf{q}_K$ (original rotations)
2: **Output:** $\hat{\mathbf{q}}_1, \cdots, \hat{\mathbf{q}}_K$ (smoothed rotations)
3: $\hat{\mathbf{q}}_1 = \mathbf{q}_1$
4: **for** $k = 2$ **to** K **do**
5: $\quad \hat{\mathbf{q}}_k = \text{slerp}(\mathbf{q}_k, \hat{\mathbf{q}}_{k-1}, \alpha)$
6: $\quad \hat{\mathbf{q}}_k \leftarrow \mathbb{P}(\hat{\mathbf{q}}_k)$
7: **end for**

VI. Rotation Smoothing via Unscented Kalman Filtering

A constant-velocity model defines the motion parameters and their velocities as state variables. The velocities are assumed to be constant in propagation except for a small random acceleration (usually modeled as Gaussian noise). The measurements are the original motion parameters and the smoothed parameters are just the estimated states. For 2D motions, a linear system is sufficient to model the dynamics and measurements so state estimate can be obtained precisely using Kalman filtering (assuming independent Gaussian process and measurement noise). Here we use the same idea to design a constant-velocity model for 3D rotation. Similarly, the state variable for each stage (frame) consists of the 3D rotation and the angular velocity. We still use unit quaternion representation of 3D rotations. The dynamic model is

$$\begin{bmatrix} \mathbf{q}_k \\ \boldsymbol{\omega}_k \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{k-1} \otimes q(\boldsymbol{\omega}_{k-1}) \\ \boldsymbol{\omega}_{k-1} + \mathbf{w}_k \end{bmatrix}, \quad (10)$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the process noise corresponding to angular acceleration. $\otimes$ is the quaternion multiplication and the function $q(\boldsymbol{\omega}_{k-1})$ is used to convert $\boldsymbol{\omega}_{k-1}$ to a unit-quaternion-represented rotation. If the 3D rotation is represented by matrices then $\mathbf{q}_{k-1} \otimes q(\boldsymbol{\omega}_{k-1})$ is equivalent to $\mathbf{R}_{k-1}\text{expm}(\boldsymbol{\omega}_{k-1})$. The measurement model is then

$$\tilde{\mathbf{q}}_k = \mathbf{q}_k \otimes q(\mathbf{v}_k), \quad (11)$$

where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the measurement noise and $\tilde{\mathbf{q}}_k$ is the original camera rotation for frame k. The system defined by (10) and (11) is actually defined based on geodesic distance on $\mathbf{SO}(3)$ and can be considered as a "linear" system on the manifold though linearity is not defined on manifolds. However, so far there is no efficient iterative algorithm to solve the estimation problem for such system exactly. In practice, we can solve the estimation problem on its embedded Euclidean space. In Euclidean space the system is clearly nonlinear. In this paper we use unscented Kalman filtering (UKF) to solve the problem. UKF picks a minimal set of sample points (sigma points) to represent the posterior probability of the state vector and propagate them through the non-linear dynamic and measurement functions, from which the mean and covariance of the estimate are then recovered. The UKF usually outperforms extended Kalman filter (EKF) for sequential nonlinear estimation with a small increase in computational complexity. More details of UKF can be found in [15].

We summarize the proposed UKF-based rotation smoothing algorithm in Algorithm 2.

Algorithm 2 UKF-based 3D Rotation Smoothing

1: **Input:** $\mathbf{q}_1, \cdots, \mathbf{q}_K$ (original rotations)
2: **Output:** $\hat{\mathbf{q}}_1, \cdots, \hat{\mathbf{q}}_K$ (smoothed rotations)
3: **Parameters:** $\mathbf{Q}$, $\mathbf{R}$ (process and measurement noise variance)
4: **for** $k = 1$ **to** K **do**
5: $\quad$ Obtain unconstrained UKF estimate $\hat{\mathbf{q}}_k$, $\hat{\boldsymbol{\omega}}_k$, $\mathbf{P}_k$
6: $\quad \hat{\mathbf{q}}_k \leftarrow \hat{\mathbf{q}}_k / ||\hat{\mathbf{q}}_k||_2$ (normalization)
7: $\quad \hat{\mathbf{q}}_k \leftarrow \mathbb{P}(\hat{\mathbf{q}}_k)$
8: $\quad$ (Mean and covariance estimate to pass to the next stage are $\hat{\mathbf{q}}_k$, $\hat{\boldsymbol{\omega}}_k$, $\mathbf{P}_k$)
9: **end for**

VII. Parameter Selection

How to select the parameters for the proposed online motion smoothing algorithms is important. For IIR-like rotation smoothing, the only parameter to tune is the smoothing weight α in Algorithm 1. Clearly larger α generates smoother rotation sequences without the black-border constraint. However, rotation smoothing with larger α deviates farther from the original camera motion, and thus triggers estimate projection in Section IV more frequently. The constraints are actually determined by the original (unsmooth) rotation and therefore differ across different frames. Frequent estimate projection may add the unwanted camera shake back and reduce the smoothness of the rotation smoothing output. In the following experiments,

we fix the $\alpha = 0.95$ if not mentioned, which does not triggers estimate projection very often.

For the UKF-based rotation smoothing we need to choose the process and measurement noise covariance $\mathbf{Q}$ and $\mathbf{R}$. We smooth the test videos with the recent proposed offline rotation smoothing method [16] and use the results as the ground truth of the intentional (smooth) camera motion. The measurement noise covariance $\mathbf{R}$ can then be learned from the differences between the original and the smoothed rotation sequences. We fix $\mathbf{R}$ as diag(0.002, 0.002, 0.002). The process noise covariance $\mathbf{Q}$ reflects the expected angular acceleration range and works similarly as α in the IIR-like rotation smoothing. We fix $\mathbf{Q}$ as diag(3e-10, 3e-10, 3e-10) to reach a balance between smoothness and less frequent estimate projection.

VIII. Experimental Results

We show the video stabilization results on two real videos. We refer the readers to the webpage of this paper (http://users.ece.utexas.edu/~bevans/papers/2014/stabilization/) to view the original videos and all our results. Both videos are captured by a walking person with Nexus S smartphone. The original frame size is 720×480 and we use a 540×360 cropping size for the stabilized video. The camera rotation is obtained by integrating the gyroscope readings after sensor calibration. Fig. 1 and Fig. 2 show a comparison between the original and the stabilized videos by the proposed algorithms. We use feature trajectories as a visualization of the continuous frames. We detect feature points in a certain frame and track them for 20 frames. The feature trajectories are plotted as black curves on top of the starting frame (the frame is plotted using alpha channel 0.5 (more transparent) to make the curves clearer). Both proposed algorithms are able to effectively smooth the feature trajectories. Note that we detect and track the feature points independently in the videos so the location and number of the feature points can be different.

We compare original and smoothed camera rotation using the proposed IIR-like smoothing algorithm for video no. 2 in Fig. 3. In this figure, the camera rotation corresponding to each frame in the video sequence is shown in the form of Tait-Bryan angles. Note that, however, our motion smoothing is performed directly on the manifold $\mathbf{SO(3)}$ instead of the Euclidean space of rotation angle representations.

To evaluate the the video stabilization algorithms numerically we compare the mean of L^1 norm of the angular velocity and acceleration of the rotation sequences. The numerical comparison is shown in Table I.

A. Comparison Against 2D motion Smoothing

We compare the proposed 3D rotational real-time motion smoothing algorithms with existing 2D motion smoothing algorithms. Compared to 2D models, the 3D rotational model can reflect the real camera motion more accurately and results in smoother results. Fig. 4 shows a comparison of feature trajectories between the proposed IIR-like rotation smoothing and IIR motion smoothing with 2D affine model (both using $\alpha = 0.95$). In the stabilized videos using 2D affine model

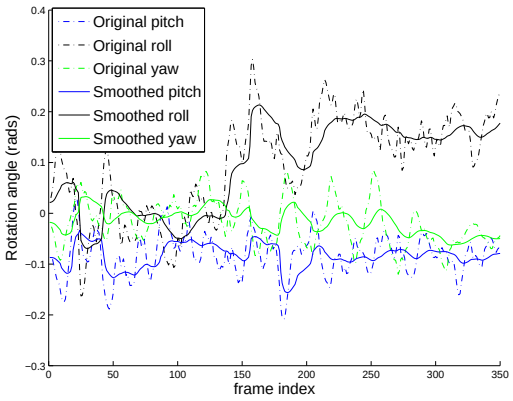


Fig. 3. Comparison of the original and the smoothed camera rotation using the proposed IIR-like rotation smoothing algorithm for video no. 2.

TABLE I
Numerical comparison between orignal and smoothed videos

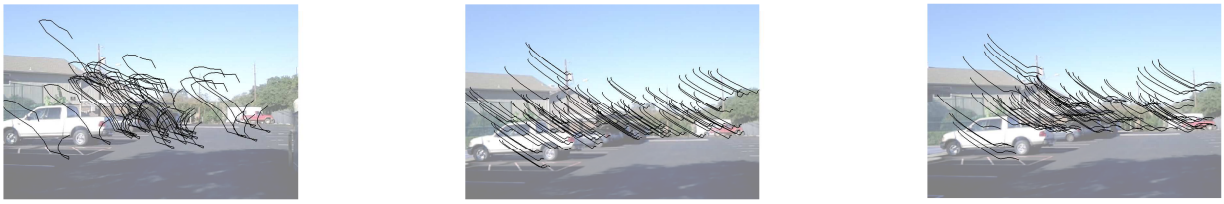
Video no. 1		
	mean angular velocity	mean angular acceleration
Unsmoothed	0.0328	0.0270
IIR smoothing	0.00730	0.00408
UKF smoothing	0.00957	0.00413
Video no. 2		
	mean angular velocity	mean angular acceleration
Unsmoothed	0.0293	0.0256
IIR smoothing	0.00805	0.00331
UKF smoothing	0.00884	0.00303

there are always wavy distortion because of the inaccuracy of the motion model. This distortion cannot be found in the stabilized videos using 3D rotational model.

B. IIR vs. UKF

The IIR-based and UKF-based methods have long been used to smooth 2D camera motion sequences (for 2D models the constant-velocity-based smoothing is solved via Kalman filtering instead of UKF because the system is linear). Both algorithms have been shown to effectively smooth camera motion in real time.. In terms of computational speed, the IIR-like smoothing algorithm is much faster. With MATLAB implementation on a 2.3GHz Intel i5 processor machine, the IIR-like smoothing algorithm takes only 1.54ms/frame while the UKF-based smoothing algorithm takes 6.97ms/frame.

If there are abrupt changes in the intentional camera motion, the UKF-based method tends to perform better with less frequent triggering of estimate projection. The reason is that in UKF-based smoothing algorithm we only assume that the angular velocity is almost constant, not that the angular velocity is almost zero. By estimating the angular velocity together with the smoothed rotation, we can better keep track of the change in intentional camera motion. In this Table II we compare the two algorithms for video no. 2, in which there is a sudden change (panning) of the intentional camera rotation. To make a fair comparison we tune the parameter α in the IIR-like smoothing algorithm from 0.95 to 0.9 so that it triggers the same times of estimate projection as the default UKF-based smoothing algorithm. We can find that under the

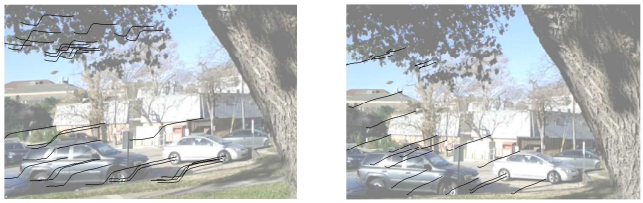


(a) Original video (b) Proposed IIR-like smoothing (c) Proposed UKF-based smoothing

Fig. 1. Stabilization comparison for video no. 1. Features are tracked from frame 31 to frame 50.

(a) Original video (b) Proposed IIR-like smoothing (c) Proposed UKF-based smoothing

Fig. 2. Stabilization comparison for video no. 2. Features are tracked from frame 46 to frame 65.



(a) 2D affine smoothing (b) 3D rotational smoothing

Fig. 4. Stabilization comparison for video no. 2 between 2D affine smoothing and 3D rotational smoothing. Features are tracked from frame 246 to 265.

TABLE II
NUMERICAL COMPARISON BETWEEN IIR-LIKE AND UKF-BASED ROTATION SMOOTHING ALGORITHMS FOR VIDEO NO. 2

	mean angular velocity	mean angular acceleration
IIR smoothing	0.01020	0.00367
UKF smoothing	0.00884	0.00303

same times of estimate projection the UKF-based smoothing algorithm generates a smoother rotation sequence.

IX. CONCLUSIONS

In this paper we propose two real-time motion smoothing algorithms for video stabilization using a pure 3D rotation motion model. Both algorithms directly smooth the 3D rotation sequences on the $\mathbf{SO(3)}$ manifold. The first algorithm is similar to 1st-order IIR filtering and requires only one slerp step for each frame. The second algorithm assumes a constant angular velocity model of the smooth rotation sequences and obtain the smooth rotation matrix for each frame via sequential probabilistic estimation. The estimation problem is solved efficiently using unscented Kalman filter. We also add a simple projection step to guarantee that no black borders intrude into the stabilized video frames. We have demonstrated in experiments that our algorithms are very fast and can generate better video stabilization results than their 2D counterparts.

REFERENCES

[1] C. Buehler, M. Bosse, and L. McMillan, "Non-metric image-based rendering for video stabilization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Dec. 2001, pp. 609–614.

[2] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graphics*, vol. 28, no. 3, 2009.

[3] C. Morimoto and R. Chellappa, "Fast 3D stabilization and mosaic construction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 1997.

[4] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," Stanford University, Tech. Rep., Mar. 2011.

[5] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, Jul. 2006.

[6] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer, "Stabilizing cell phone video using inertial measurement sensors," in *Proc. IEEE Intl. Workshop Mobile Vision*, Nov. 2011.

[7] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2011.

[8] C. Jia and B. L. Evans, "3D rotational video stabilization using manifold optimization," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, May 2013.

[9] Y.-S. Wang, F. Liu, P.-S. Hsu, and T.-Y. Lee, "Spatially and temporally optimized video stabilization," *IEEE Trans. Visualization and Computer Graphics*, vol. 19, no. 8, pp. 1354–1361, 2013.

[10] S. Ertürk, "Image sequence stabilization: motion vector integration (MVI) versus frame position smoothing (FPS)," in *Proc. Intl. Symp. Image and Signal Processing and Analysis*, Jun. 2001.

[11] ——, "Real-time digital image stabilization using Kalman filters," *Real-Time Imaging*, vol. 8, pp. 317–328, 2002.

[12] A. Litvin, J. Konrad, and W. Karl, "Probabilistic video stabilization using Kalman filtering and mosaicking," *Proc. IS&T/SPIE Symp. Electronic Imaging, Image and Video Comm. and Proc.*, pp. 663–674, 2003.

[13] M. Tico and M. Vehvilainen, "Constraint motion filtering for video stabilization," in *Proc. IEEE Intl. Conf. Image Processing*, Sep. 2005.

[14] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. ACM SIGGRAPH*, 1985, pp. 245–254.

[15] E. Wan and R. Van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Oct. 2000.

[16] C. Jia and B. L. Evans, "Constrained 3D rotation smoothing via global manifold regression for video stabilization," *IEEE Trans. Signal Processing*, vol. 62, pp. 3293–3304, 2014.