Stabilizing First Person 360 Degree Videos

Chetan Arora
IIIT Delhi

Vivek Kwatra
Google Research

Abstract

The use of 360 degree cameras, enabling one to record and share full-spherical $360^\circ \times 180^\circ$ *view without any cropping in the viewing angle, is on the rise. Shake in such videos is problematic, especially when used in conjunction with VR headsets causing cybersickness to the viewer. The current state-of-the-art video stabilization algorithm [17] designed specifically for 360 degree videos considers the special geometrical constraints in such videos. However, the specific steps in the algorithm can abruptly change the viewing direction in a video leading to unnatural experience for the viewer. In this paper, we propose to fix this anomaly by the use of* $L1$ *smoothness constraints on the camera path, as suggested by Grundmann et al. [7]. The modified algorithm is generic and our experiments indicate that the proposed algorithm not only gives a more natural and smoother stabilization for 360 degree videos but can be used for stabilizing normal field of view videos as well.*

1. Introduction

Advances in processor and sensor technologies have enabled 360 degree cameras to shrink and become affordable. This has made their usage practical for wider consumer adoption. The 360 degree cameras typically record full-spherical $360^\circ \times 180^\circ$ view without any cropping, thus giving unmatched freedom to a videographer to focus on experiencing the moment rather than worrying about the camera capturing direction (Fig. 1). On the other hand, the viewers are also free to explore the viewing perspective of their choice using a keyboard based input or by simply moving their head while wearing a VR headset. Infact, one of the reasons for the popularity of 360 degree videos could also be attributed to virtual reality (VR) systems becoming mainstream. 360 degree videos are by far one of the most popular forms of content for such devices. Availability of commodity cameras such as Ricoh Theta, Samsung Gear 360, Giroptic and viewing interfaces on popular content distribution websites like YouTube and Facebook have only helped further.

While, recording and viewing of 360 degree videos is fairly easy now, the processing of such videos for optimal viewing experience is still a challenge. In the last few years alone, researchers have addressed problems ranging from inpainting [34, 33, 16], tracking [3], stereo [26], virtual reality[9], upright adjustment [10], visual odometry [35], background estimation [15], to hyperlapse [19], and stabilization [12, 17]. Some of the novel applications specific to 360 degree videos have been explored as well [31, 30, 8, 32, 14]. For example, researchers are developing new algorithms to navigate in 360 degree videos. The algorithms help a viewer decide where and what to look at by controlling the direction of virtual camera [31, 30, 8, 32].

In this paper, we focus on the problem of 360 video stabi-

Figure 1: 360 cameras capture the entire 360 degree field of view (FOV) around them, as shown in the *equirectangular* view on the top. The bottom four perspective FOV images are obtained by resampling the full 360 frame on the top. 360 video players allow the user to select the desired FOV through an interactive UI, or via their head motion in VR headsets. However, a videographer may want to drive the user's focus towards a desirable FOV by intentionally controlling the camera orientation during capture.

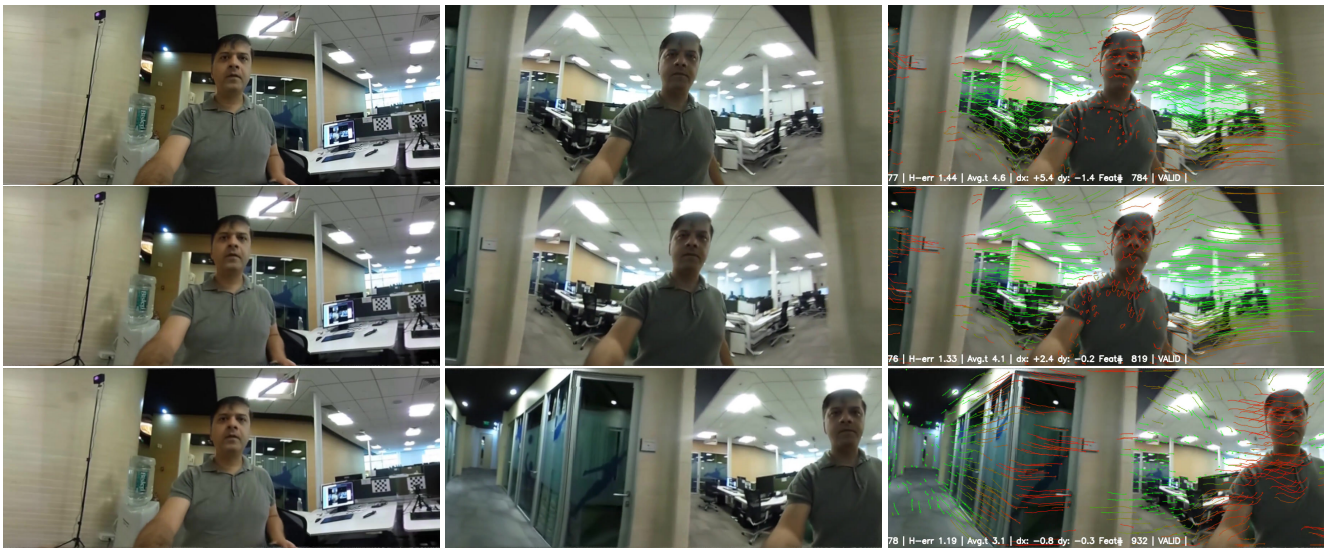

Figure 2: Demonstration of our 360 stabilization technique on OFFICE sequence. Top row: frames from input sequence. Middle row: Frames from our stabilized sequence. Bottom row: Result from our implementation of Kopf *et al.*. First column shows a frame from the beginning of sequence. 2nd column shows a frame 5 seconds into the sequence. The last column shows feature tracks computed over the respective sequence overlaid onto the frame from the 2nd column. Notice that the feature trajectories from our result and Kopf *et al.*'s result are smooth, indicating successful stabilization (as opposed to the input trajectories, which are jittery) – the color of the trajectories indicates whether they agree with the dominant (background) feature motion in the frame (green), or are likely to be outliers or foreground (red); hence greener trajectories tend to be more relevant in evaluating smoothness. The distinguishing feature of our approach is that our result preserves the original intended camera path (demonstrated by the person staying in the middle of the frame), even while achieving stabilization. Kopf *et al.*'s approach on the other hand, eliminates all camera rotation, which may cause it to deviate from the intended camera path (the person moves to the side of the frame in this example). Note that, looking at the output of Kopf *et al.* from a VR headset, a wearer will experience taking a turn and start looking sideways.

lization. Stabilization is a standard problem for narrow field of view (NFOV) videos, but becomes even more acute for 360 degree videos. The 360 degree videos are usually consumed through VR headsets, where viewing shaky videos can even lead to physical discomfort, known as 'cybersickness' [13]. Most existing state of the art stabilization techniques are designed with NFOV videos in mind and cannot be translated to the spherical wraparound view in 360 degree videos.

In some aspects, video stabilization is an easier problem for 360 degree videos. A simple algorithm, which stabilizes the viewing direction by warping every frame to the reference frame using the estimated frame to frame homography, may be sufficient in many cases. This is not a desirable technique for NFOV videos instances because of excessive cropping it may lead to. In 360 degree videos, cropping is not a concern since we have full spherical view available. In fact, Kopf *et al.*[17] has proposed a technique very similar to the above outline, specifically for 360 degeree videos. They first compute 3D rotations across the entire sequene, and once estimated, each frame is simply warped to the reference frame using this computed transformation. Their technique leads to excellent results, but has a problem whenever the videographer takes a large turn. Because of alignment with the reference frame direction, the output video now awkwardly starts looking sideways.

In our work, we specifically address this issue of broadly preserving the videographer's intended camera orientation in the stabilized video. This requires computing a stabilized camera orientation path that does not deviate significantly from the original orientations. To achieve this, we solve a constrained optimization problem over camera rotations. Our approach is inspired by the work of Grundmann *et al.* [7]. However, while their formulation optimizes over 2D linear transformations that can be solved using linear programming techniques, we optimize over 3D camera rotations using non-linear optimization over quaternions.

Contributions: We propose a novel algorithm for 360 degree video stabilization that computes the optimal 3D camera rotations for stabilization, without destroying the originally intended camera orientation. We find 3D rotations, first between keyframes and then between each consecutive

frames, as suggested by Kopf *et al.*[17]. However, rather than warping to reference frame, we use the computed 3D rotations, and stabilize the camera orientation by minimizing the first, second, and third derivatives of the resulting camera path. Warping to this smooth camera path keeps the output video view point near to original even when the videographer takes a turn. Figure 2 compares an output from our algorithm with that of Kopf *et al.*. It may be noted that the idea of computing steady optimal path based upon 3D rotations rather than homography is itself novel and can be applied to regular NFOV videos as well. Though, our focus is on 360 degree videos, we show some experiments on such videos as well.

2. Related Work

Stabilization techniques generally vary on the spectrum of flexibility and robustness. Simpler 2D techniques based on smoothing linear models [25] or fitting linear camera paths [4] trade off flexibility for robustness, while 3D approaches based on SfM [20, 18] or image-based rendering [2] can be more expressive but prone to failure cases. Hybrid approaches include imposing subspace constraints [21] from 3D to 2D domain, or use of models that extend to 2.5D via homographies [7] or homography mixtures [6]. Higher flexibility is achieved by increasing the degrees of freedom, *e.g*. by mesh warps [21, 5, 22] or going to dense flow fields [23]. Ringaby and Forssen [28] employ 3D rotations for rolling shutter removal and stabilization of cell-phone videos, using matrix averaging for temporal smoothing.

Stabilization for 360 degree videos necessitates 3D estimation, which is needed for spherical warping, making the problem challenging. Past work is based on frame-to-frame estimation of relative rotation [13] or use omnidirectional SfM [11]. More recently, Kopf *et al.* [17] have suggested a simple video stabilization technique for 360 degree videos by first finding 3D rotations between certain pairs of consecutive key frames. For the intermediate frames, an optimization technique is used to maximize feature trajectory smoothness, coupled with a deformed-rotation model. The latter allows handling some degree of translational motion, parallax, lens deformations, and rolling shutter wobble. We employ this approach for estimating rotations in our work.

Our approach for camera motion smoothing is inspired by [7]. They compute optimal steady camera paths, within an optimization framework, by moving a crop window of fixed aspect ratio along this path while minimizing an L1-smoothness constraint based on the cinematography principles. Their formulation is limited to 2D linear transformations, which allows for solutions via linear programming. We extend this path optimization approach to work for 3D rotations and 360 videos, by using a quaternion-based representation, coupled with a non-linear solver.

Figure 3: Left: A sample frame from a 360 degree video in equirect format. Note the distortions near the pole. Right: Corresponding cubemap representation. Each face image is equivalent to one taken from a pinhole camera.

3. Tracking and Estimating Camera Pose

3.1. Cubemap Representation

360 degree videos are usually represented in an equirect format. This format is suitable for viewing but hard to process using computer vision techniques. For processing by our algorithm, we convert it to a cubemap representation. The representation is obtained by projecting the viewing sphere to the six faces of a unit cube. Each face is equivalent to an image captured by a pinhole camera at the center of the cube having a unit focal length. The face images follow the standard epipolar geometry and most of the computer vision algorithm for camera pose estimation can be applied to these images. Figure 3 shows a sample equirect image and its corresponding cubemap representation.

3.2. Tracking and KeyFrame Pose

We track GFTT [29] features between a pair of cubemaps using LK tracker [24] and some minor additional consistency checks for robust tracking. We track features points on each face of the cubemap independently. It is possible that feature points near the boundary of a face becomes visible in an adjoining face in the next frame. The proposed independent face tracking loses track for such features. We note that it is possible to come up with a more complicated feature tracking to track such features also, but, experimentally, did not feel the need to do so.

The tracked features points are converted to a 3D vector representation (assuming a unit cube, i.e. $z = 1$), and transformed to the coordinate system of a reference face, chosen arbitrarily. The tracked 3D points from all faces are then *jointly* fed into a RANSAC based camera relative pose estimator [27]. We discard the relative translation returned by the pose estimator and only use the rotation component for our computation ahead.

As noted by Kopf *et al.*[17] also, the relative motion and the parallax between two consecutive frames may be small leading to noisy estimates of the camera relative pose. We designate certain frames as keyframes when the relative motion is large enough to estimate pose reliably. In our experi-

ments, we average the magnitude of optical flow vector and whenever the average exceeds beyond 20 pixels, the frame is marked as a keyframe. We estimate the relative rotation, $\hat{R}$, between each two consecutive keyframes independently, and then chain them to compute the rotation of a keyframe relative to the first frame as: $R_{k_i} = \left(\prod_i \hat{R}_{k_i}^{-1}\right)^{-1}$

3.3. Pose Estimation for Intermediate Frames

Direct camera pose estimation between two consecutive frames is noisy because of lack of motion and parallax between them. Since the rotation estimation is ultimately used to smoothen the camera trajectory, we compute the relative rotation between the consecutive frames that leads to maximally smooth feature trajectories. To achieve this, we divide the input frames in batches with a keyframe at the beginning and end of the batch, and containing all the intermediate frames between these two keyframes. The rotations for all intermediate frames of a batch are computed together by solving an optimization problem, describe below.

Similar to [17], we formulate an optimization problem containing two energy components. The first component, E_f minimizes first order smoothness of feature tracks, while the second, E_s, minimizes second order jitter. The complete expression for the batch with keyframes j and k at its two ends can be written as follows:

$$\arg\min_{R_i|i\in I} \sum_i \left[E_f(i, i-1) + E_s(i, i-1, i-2)\right],$$

where I is the set of intermediate frames, and R denotes their rotation relative to first frame. E_f and E_s can be written as follows:

$$E_f(i, i-1) = \sum_{p\in\mathcal{C}} f(R_i p - R_{i-1} p), \text{ and}$$

$$E_f(i, i-1, i-2) = \sum_{p\in\mathcal{C}} f(R_i p + R_{i-2} p - 2 * R_{i-1} p).$$

Here $\mathcal{C}$ denotes the set of corresponding points between a pair of frames or a triplet, as is clear from the context, and f is some loss function penalizing non-smoothness. We have experimented with L1 and L2 norms in our experiments. We note that more robust functions which reduce the sensitivity to outliers as used by Kopf *et al.* could have been used.

In an important difference from Kopf *et al.*'s algorithm, we do not create feature trajectories spanning the whole batch. Such feature trajectories are rare. Using paired or triplet correspondence allows us to work with many more points making our system more robust.

Another important difference between our system and the one by Kopf *et al.* is in the rotation representation. It is not clear from their paper, but we speculate that the authors have derotated the frames to the reference frame before finding the rotations for the intermediate frames. This results in relatively smaller rotations for the intermediate frames. In our case we do not derotate, which results in large value of rotations with respect to the reference frame. We observe quaternion representation of rotation to be more stable than axis-angle representation for larger angles and use that in our optimization problem. However, we noted that our solver does not always maintain the required unit magnitude for the quaternion. Therefore, we also add an additional cost in the optimization, penalizing magnitude divergence from the ideal value. We use Ceres solver [1] for solving the proposed optimization problem.

4. Stabilizing 360 Degree Videos

An advantage with 360 degree cameras is that they capture the whole viewing sphere. Therefore, one can arbitrarily warp the frame for stabilizing the video, without worrying about the frame cropping that would happen in narrow field-of-view videos. However, we observe that this is a trap. Consider a scenario where a videographer is moving forward and looking straight through a 360 degree camera. At an intersection, the videographer makes a 90 degree turn to the right. If one stabilizes by warping the frame to the reference, then in the output video, the viewpoint of the camera will shift to looking left of the road rather than forward. Ofcourse, a viewer can rotate the viewing perspective and bring it back to the forward direction, but this is awkward and degrades the experience.

Grundmann *et al.* have proposed a technique to stabilize narrow FOV videos by computing an L1 optimal camera path. They suggest finding an *update* affine transformation, which when applied to the frame-to-frame transition transformation of the input unstabilized video, makes the composite transformation (transition times the update) smooth. Restriction to affine allows them to convert the optimization problem to a standard linear program. Their approach does not apply to our case directly. Using affine transforms to warp a 360 degree video frame in equirect representation is geometrically incorrect. Converting to cubemap allows one to use pinhole camera geometry but then the update transforms for the six faces of the cube must reconcile geometrically. We propose a new stabilization algorithm which is inspired from the idea of smooth camera path as suggested by Grundmann *et al.* However, the detailed formulation and technique required to make it work on 360 degree video is novel and our contribution.

4.1. Stabilizing with L1 Optimal Camera Path

It is well known that camera rotation affects video stabilization more than the translation. This is because translation does not induce much optical flow in the far away points, whereas rotation affects near and far points similarly making the effect of rotational camera shake more pronounced. Therefore, we propose to find a steady camera tra-

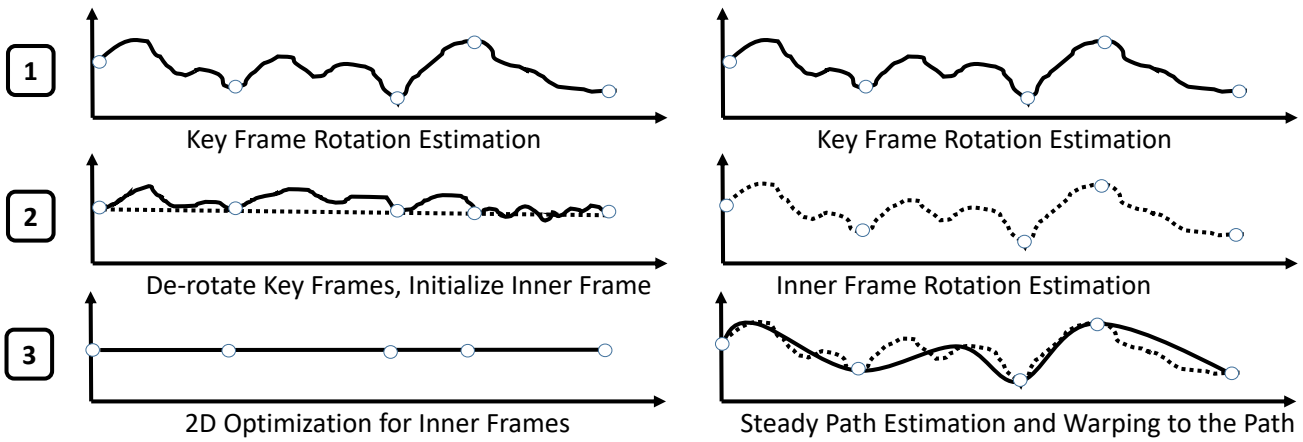

Figure 4: Schematic comparison between approach by Kopf *et al.* [17] (Left) and Ours (right). In the first step both the approaches find 3D rotation between keyframes followed by the estimation for the inner frames. In the second step, Kopf has suggested to de-rotate the frames to reference frame. This may abruptly change the viewpoint in the output video if videographer takes a turn. To fix the anomaly, we pose the 360 degree video stabilization problem as an optimization problem which finds an update transformation for each frame which when applied to the frame smoothes the trajectory. Intuitively this is equivalent to warping to the nearest smooth camera trajectory as suggested by Grundmann *et al.* [7].

jectory based upon rotational component and ignoring the translation. Since, all faces of the cubemap in a 360 degree video frame are constrained by the same 3D rotation, this does not introduce any inter-face distortion. Further, warping a face for 3D rotation does not require any depth information of the scene, thus simplifying the analysis.

For the purpose of the following discussion, we assume that the 3D rotations, R_i, for all the frames, keyframes and intermediate frames, have been computed. All the rotations are represented with respect to the first frame using quaterions. Some of the reasons for using quaternion over axis angle representation has already been described in the last section. We discuss some more later in this section. We compute an update rotation, B_i, for each frame, which when applied to a frame results in a smooth camera trajectory (in terms of rotation). Specifically, we minimize the following energy values:

- First order smoothness:

$$E_f = \sum_{i \in I} f(R_i B_i - R_{i-1} B_{i-1})$$

- Second order smoothness:

$$E_s = \sum_{i \in I} f(R_i B_i + R_{i-2} B_{i-2} - 2R_{i-1} B_{i-1}),$$

where, as in pose estimation, f is a robust loss function penalizing non-smoothness of trajectories. Here again, we have experimented with L1 and L2 norms, however other loss functions could have equivalently applied as well. The overall optimization can be written as follows:

$$B^* = \arg\min_{B_i | i \in I} \lambda_f E_f + \lambda_s E_s$$

Note that there exists a trivial solution to this problem, obtained by setting B_i as the inverse of R_i. In that case, our solution becomes the same as the one suggested by Kopf *et al.* [17]. To prevent such a solution, and also to keep the update transformations smaller in general, we impose an additional constraint on the maximum permissible angular rotation in update transformation. Since we use quaternions as representation for rotation matrices, where the first element is $cos(\theta/2)$ (where θ is the angular rotation), such a constraint becomes simply a lower bound on the first element, and easy to handle in any optimization problem solver.

It is important to state here that using quaternions instead of axis angles for representation of rotations has an additional signficant advantage. Our problem formulation requires composing rotations by multiplying the 3D rotation computed from camera pose estimation with the unknown update rotation that will be estimated as part of the optimization. For two rotation matrices, R_1 and R_2, represented as quaternions, $q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$ and $r = r_0 + \mathbf{i}r_1 + \mathbf{j}r_2 + \mathbf{k}r_3$, respectively, the quaternion, t, corresponding to their composition $R_2 R_1$ can be simply written as:

$$\begin{aligned}
t_0 &= (r_0 q_0 - r_1 q_1 - r_2 q_2 - r_3 q_3) \\
t_1 &= (r_0 q_1 + r_1 q_0 - r_2 q_3 + r_3 q_2) \\
t_2 &= (r_0 q_2 + r_1 q_3 + r_2 q_0 - r_3 q_1) \\
t_3 &= (r_0 q_3 - r_1 q_2 + r_2 q_1 + r_3 q_0).
\end{aligned}$$

Corresponding to our optimization problem, $R2$ represents the rotation from the camera pose and $R1$, the update transformation to be computed. Since, camera pose, and thus r is a constant in the optimization problem, the overall expression is linear in the parameters, q. In contrast, an axis angle representation would have required complicated terms involving angle sines and cosines. From the optimization perspective, using quaternion greatly simplifies the problem.

We use Ceres [1] to solve this optimization problem as well. Similar to pose estimation, we need to put an extra term in the optimization assigning a high cost to the solution when norm of the update vector, B_i, is not identity. This is to make sure that the returned vector is a valid quaternion – this is the single non-linear term in the optimization other than the robust cost function.

Figure 4 gives a schematic diagram describing key steps of our algorithm. Along with, we show the similar steps for the algorithm given by Kopf *et al.* to highlight the difference between the two algorithms. The schematic for Kopf *et al.*'s algorithm has been reproduced from [17].

5. Generalization to NFOV Videos

Many stabilization techniques for narrow FOV cameras compute and exploit 3D camera poses, but most of them fit the camera trajectory to simple linear or quadratic curves. Grundmann *et al.*have suggested a principled technique to automaticlly split the trajectory into portions of static, panning or constantly accelerating camera motion. However, their technique is limited to 2D linear transformations. The proposed approach of stabilization by smoothing 3D rotations is novel by itself even for narrow FOV cameras.

Keeping this in mind, we also explore the applicability of proposed technique for regular cameras. Figure 9 shows one sample result. The algorithm remains exactly the same as described for 360 degree cameras except for the fact that the 3D rotations are now computed on a single frame rather than over six faces together as described in Section 3 .

6. Experiments and Results

We have implemented our algorithm in C++. We use GFTT features [29] and LK tracker [24] for feature tracking. Keyframes are marked whenever the average optical flow between the frames increases beyond 20. We use our own implementation for finding relative camera pose between keyframes which is based on Nister's algorithm [27]. For intermediate frames we use the formulation described in Section 3 and use Ceres [1] solver for solving the optimization problem. The optimization problem for the stabilization is also solved using the Ceres solver.

The implementation of Kopf *et al.*'s algorithm is not publicly available. We use our own implementation for camera pose estimation but warp each frame to the reference. Results generated using this implementation is referred to as the output of Kopf *et al.*'s algorithm throughout.

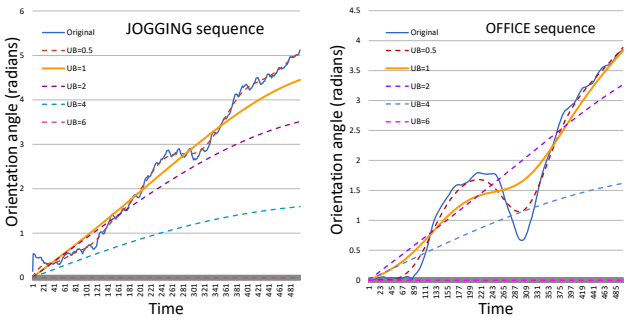

Figure 5: Stabilized orientation angles overlaid on the original orientation angles over time for two video sequences (best viewed in color). The angle for each frame is computed as the magnitude of the axis-angle representation of the rotation w.r.t the first frame. The solid blue curve corresponds to the original shaky orientation path, while the remaining curves show the stabilized orientation angles corresponding to different upper-bounds on the stabilization update, which limits the degree by which the original orientation can be changed. The smoother curve after the stabilization indicates that shake in camera orientations have stabilized. Our video results use an upper bound UB=1 radian. At UB=6 radians, our result coincides with Kopf *et al.*'s result: all frames become fully registered with the first frame (curve merged into x axis).

We have applied our technique on select 360 degree videos available on YouTube and have used them for testing and comparisons. Figures 7 and 8 sample results on two 360 degree video sequences. Figure 9 shows a result on a sequence from an NFOV camera. Since the advantage of our algorithm becomes pronounced in the presence of turns, we have chosen to highlight them in this section. We refer the reader to supplementary material for more results.

To better visualize the effect of stabilization, we analyze the angular rotations (with respect to the reference frame), before and after the stabilization. Figure 5 show stabilized orientation angles overlaid on the original orientation angles over time for two video sequences corresponding to different upper-bounds on the stabilization update, which limits the degree by which the original orientation can be changed. The smoother curve after the stabilization indicates that shake in camera orientations have stabilized. At UB=6 radians, our result coincides with Kopf et al.'s result: all frames become fully registered with the first frame.

We evaluate our approach quantitatively by analyzing the smoothness of motion trajectories computed on the videos. We compute fresh feature trajectories over the stabilized video, the original video, as well as for Kopf *et al.*'s result. We then compute the first, second and third-order smooth-

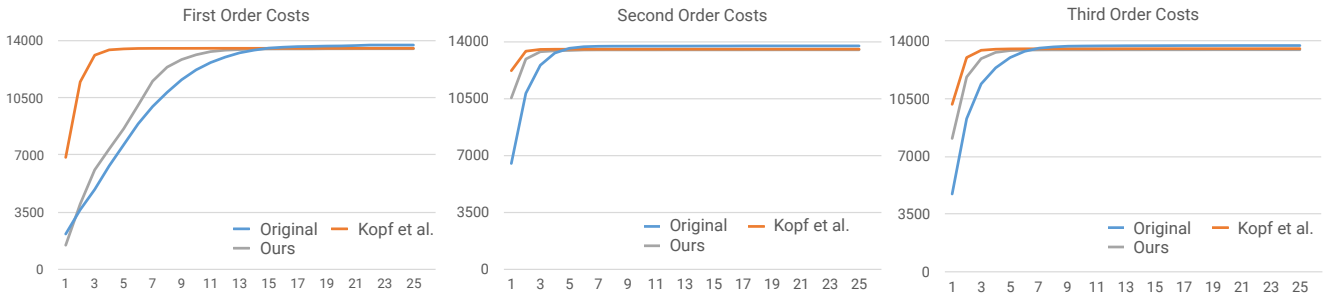

Figure 6: Cumulative Distribution Functions (CDF) for the first, second and third order smoothness of feature trajectories, corresponding to the OFFICE video from Figure 2. We compare against the original video as well as Kopf *et al.*'s result.

Figure 7: Stabilization result on JOGGER sequence. 1st (top) row: select frames from input sequence. 2nd row: our stabilization result – frames do not deviate too far from the original, preserving the intented camera motion. 3rd row: jittery feature tracks computed from input sequence. 4th row: smooth feature tracks computed from our stabilized sequnce. 5th row: Kopf *et al.*'s result – frames deviate from original camera path, turning by a large angle relative to the original path.

ness costs, defined as the norm of the corresponding temporal derivates of the feature trajectories, and plot the cumulative distribution functions (CDFs) of these costs. As shown in Figure 6, the original video has the highest costs, Kopf *et al.* has the lowest, while our costs fall in the middle. This is expected, since unlike Kopf *et al.*, we do not completely eliminate the frame rotation. Interestingly though, the gap between ours and Kopf *et al.*'s CDFs decrease when comparing second and third order costs. This happens because the first-order cost measures perfect registration, which we do not strive for, while the second and third order costs measure smoothness of the motion, where we do well.

Figure 8: Stabilization result on SKATEBOARD sequence. 1st (top) row: select frames from input sequence. 2nd row: our stabilization result. 3rd row: Kopf *et al.*'s result. In the original sequence, the user makes a U-turn. Kopf *et al.*'s technique does not take that into account. As a result, the camera starts moving backwards w.r.t its original orientation (notice in the last bottom-right frame, the building is on the right, instead of on the left). On the other hand, in our result, the camera makes a U-turn even in the stabilized sequence.

Figure 9: Our results on an NFOV sequence. Left column shows original frames; right column shows our result, demonstrating its usefulness in the presense of sharp 3D rotations in the video – in this case, the videographer pans the camera quickly while walking, evident by the motion of the fountain wall on the left. We stabilize this motion reasonably well despite the presence of strong 3D rotation.

7. Conclusion

In this paper we have presented a new video stabilization algorithm for 360° videos. Unlike the current state of the art which may abruptly change the camera orientation in the output whenever a videographer takes a large turn, our approach does not significantly change the original orientation, leading to a more natural experience for the viewer. We pose the problem as warping to the closest smooth camera trajectory. We make important contributions in formulating the problem as an optimization over 3D rotations represented as quaternions. Though our focus is 360 degree videos, the proposed technique is generic enough to be used for regular narrow FOV videos as well. In future we would like to explore a combination of 3D and 2D techniques, such as ours and [7] respectively, for improved stabilization accounting for jitter in camera translation as well.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver: http://ceres-solver.org.

[2] C. Buehler, L. McMillan, and M. Bosse. Non-metric image-based rendering for video stabilization. *IEEE Conference on Computer Vision and Pattern Recognition*, 02:609, 2001.

[3] A. Delforouzi, S. A. H. Tabatabaei, K. Shirahama, and M. Grzegorzek. Unknown object tracking in 360-degree camera images. In *23rd International Conference on Pattern Recognition (ICPR)*, pages 1798–1803, Dec 2016.

[4] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(1):2:1–2:28, Oct. 2008.

[5] A. Goldstein and R. Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 31(5), Sept. 2012.

[6] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Calibration-free rolling shutter removal. In *International Conference on Computational Photography*, 2012.

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

[8] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360deg sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *IEEE Virtual Reality (VR)*, pages 37–44, March 2017.

[10] J. Jung, B. Kim, J.-Y. Lee, B. Kim, and S. Lee. Robust upright adjustment of 360 spherical panoramas. *Vis. Comput.*, 33(6-8):737–747, June 2017.

[11] M. Kamali, A. Banno, J.-C. Bazin, I. S. Kweon, and K. Ikeuchi. Stabilizing omnidirectional videos using 3d structure and image warping. In *12th IAPR Conference on Machine Vision Applications (MVA)*, 2011.

[12] S. Kasahara, S. Nagai, and J. Rekimoto. Livesphere: Immersive experience sharing with 360 degrees head-mounted cameras. In *Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology*, pages 61–62, 2014.

[13] S. Kasahara, S. Nagai, and J. Rekimoto. First person omnidirectional video: System design and implications for immersive experience. In *ACM International Conference on Interactive Experiences for TV and Online Video*, pages 33–42, 2015.

[14] S. Kavanagh, A. Luxton-Reilly, B. Wüensche, and B. Plimmer. Creating 360° educational video: A case study. In *28th Australian Conference on Computer-Human Interaction*, pages 34–39, 2016.

[15] N. Kawai, N. Inoue, T. Sato, F. Okura, Y. Nakashima, and N. Yokoya. Background estimation for a single omnidirectional image sequence captured with a moving camera. *IPSJ Transactions on Computer Vision and Applications*, 2014.

[16] N. Kawai, K. Machikita, T. Sato, and N. Yokoya. Video completion for generating omnidirectional video without invisible areas. *IPSJ Transactions on Computer Vision and Applications*, 2:200–213, 2010.

[17] J. Kopf. 360° video stabilization. *ACM Trans. Graph.*, 35(6):195:1–195:9, Nov. 2016.

[18] J. Kopf, M. F. Cohen, and R. Szeliski. First-person hyperlapse videos. *ACM Trans. Graph.*, 2014.

[19] W. S. Lai, Y. Huang, N. Joshi, C. Buehler, M. H. Yang, and S. B. Kang. Semantic-driven generation of hyperlapse from 360° video. *IEEE Transactions on Visualization and Computer Graphics*, PP(99), 2017.

[20] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. In *ACM SIGGRAPH*, pages 44:1–44:9, 2009.

[21] 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.

[22] 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.

[23] S. Liu, L. Yuan, P. Tan, and J. Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4209–4216, 2014.

[24] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence - Volume 2*, pages 674–679, 1981.

[25] 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(7), 2006.

[26] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski. Low-cost 360 stereo photography and video capture. *ACM Trans. Graph.*, 36(4), 2017.

[27] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[28] E. Ringaby and P.-E. Forssén. Efficient Video Rectification and Stabilisation for Cell-Phones. *International Journal of Computer Vision*, 2012.

[29] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[30] Y.-C. Su and K. Grauman. Making 360deg video watchable in 2d: Learning videography for click free viewing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[31] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2vid: Automatic cinematography for watching 360° videos. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[32] Y. Won Kim, C.-R. Lee, D.-Y. Cho, Y. Hoon Kwon, H.-J. Choi, and K.-J. Yoon. Automatic content-aware projection for 360deg videos. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[33] B. Xu, S. Pathak, H. Fujii, A. Yamashita, and H. Asama. Optical flow-based video completion in spherical image sequences. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 388–395, Dec 2016.

[34] B. Xu, S. Pathak, H. Fujii, A. Yamashita, and H. Asama. Spatio-temporal video completion in spherical image sequences. *IEEE Robotics and Automation Letters*, 2(4):2032–2039, Oct 2017.

[35] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808, May 2016.