Survey of 2D Video Stabilization Techniques

Olivia Loh
olivia77@stanford.edu

Abstract—Video stabilization plays a crucial role in enhancing the quality of visual media by mitigating the effects of camera shake. This project undertakes a comprehensive survey of iterative 2D video stabilization methods, focusing on signal processing techniques to denoise shaky camera paths. The experimentation involved the exploration of various spatial filters, Kalman filters, and a newly proposed hybrid filter to effectively smoothen the camera trajectory. Moreover, this study incorporates an optimization framework from existing literature, aimed at optimizing a stable camera path that adheres to cinematic principles and preserves content from the original video. The survey reveals that iterative methods achieve adequately stable videos while maintaining computational efficiency. These methods are particularly relevant to commercial applications, such as professional video stabilization software, which demand the capability for real-time implementation.

Index Terms—Cinematography, Computer graphics, Filtering, Linear programming, Trajectory optimization

1 Introduction

Video stabilization serves many purposes, from professional filming to casual vlogging and even applications in UAV filming. Manually shooting a video often results in jittery footage, especially prevalent in hand-held or UAV shots. Stabilizing footage becomes imperative, particularly in film production, where the objective is to deliver smooth, captivating content while minimizing distractions caused by jerky camera movements.

In film production, stabilization equipment, such as dollies, jibs, gimbals, and camera rigs, are employed to achieve fluid cinematic motion. However, some amateur filmmakers or UAV operators have no access to this specialized and costly equipment. Consequently, digital video stabilization techniques offers an alternative to correct unstable footage in post-production.

For this final project, we surveyed a myriad of iterative 2D video stabilization techniques. We stabilized videos by employing spatial filters, Kalman filters, and a hybrid filter. We also assess a popular L1 optimization algorithm [1], deployed in the YouTube Video Edits stabilizer. This project aims to highlight the relevance of iterative methods for commercial use or other time-sensitive applications.

2 Related Work

2.1 Background

Video stabilization is traditionally performed in a pipeline shown in Figure 1. Some state of the art deep learning methods target and improve specific portions in the pipeline, such as optical flow [2], motion smoothing [3], or motion impainting [4]. Recent advancements in dynamic view synthesis methods, such as DynIBaR [5] and NSFF [6], enable video stabilization by synthesizing novel views from video frames and rendering a smooth path. However, this approach is impractical due to lengthy training times required to create a 3D representation for each video. In this project, we focus on motion smoothing and using iterative methods.

2.2 Filtering

Previous research employ many signal processing techniques to denoise and smoothen the camera trajectory [7], [8]. Ertuerk proposed low pass filtering using a raised-cosine filter [9]. Rawat and Sawale surveyed spatial filters, using temporal Gaussian, mean, and median kernels [10]. Kejriwal and Singh proposed using Kalman filtering, and a hybrid filter combination of Kalman, lowpass, followed by another Kalman filter [11]. For this project, we implemented a variety of filters, as well as a new hybrid filter combining Gaussian smoothing and Kalman filtering in section 3.2.

2.3 L1 Optimization

Grundmann et al. proposed an algorithm to derive the most optimal camera path [1]. Despite published more than ten years ago, this video stabilization method is still regarded as one of the state of the art methods [12]. The algorithm aims to optimize a shaky camera trajectory based on cinematic principles. They theorize, in professional cinematography, a stable video has a camera path that can be divided into segments with either zero first, second, or third derivatives. For instance, the camera path of a static shot is a linear segment with zero first derivative. A panning shot of constant velocity is a linear segment with zero second derivative, while a panning shot of constant acceleration is a parabolic segment with zero third derivative. In addition to formulating an objective using cinematic rules, they proposed proximity and inclusion constraints in their optimization framework. This allows for an optimized path contained within bounds of the original path, so as to not develop undesirable crop artifacts in the stabilized video. This algorithm and its linear programming framework is further elaborated in section 3.3.

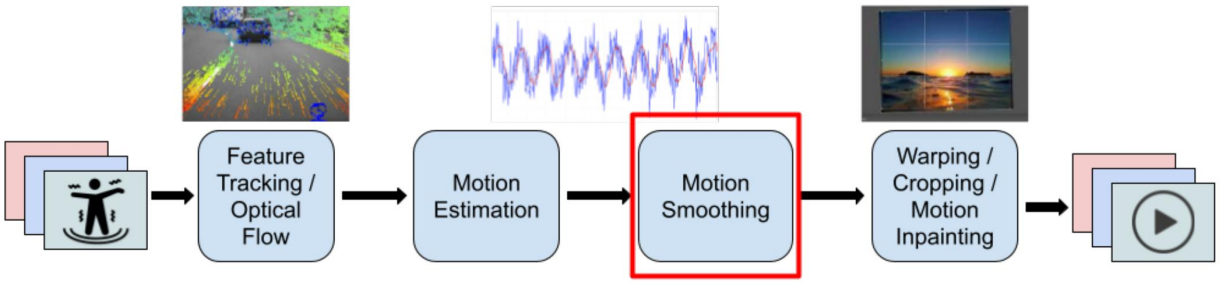

Fig. 1. Video Stabilization Pipeline

3 Proposed Method

3.1 Stabilization Pipeline

3.1.1 Feature Tracking and Optical Flow

For this project, we employ feature tracking using the Shi-Tomasi corner detection method [13]. In each video frame, we track 500 features with a minimum quality level of 0.01 and a minimum euclidean distance between corners of 10 pixels. We then compute optical flow between frames using pyramidal Lucas-Kanade method [14].

3.1.2 Motion Estimation

Given the features and optical flow between frames, we estimate the 2D affine transformation between frames by RANSAC [15]. We estimate a camera trajectory from aggregating a series of these transformations over frames.

3.1.3 Motion Smoothing

For motion smoothing, we denoise the shaky camera trajectory by implementing a variety of filters, as well as a hybrid filter shown in section 3.2. In addition, we also utilize the Robust L1 Optimal Camera Paths algorithm from this paper [1] to obtain an optimal path corresponding to cinematic principles and other constraints shown in section 3.3.

3.1.4 Warping and Cropping

After obtaining the stabilized camera trajectory, we compute transforms that warp the original frame to the new stabilized frame corresponding to the new trajectory. We apply another transform to crop the frame using ratio of 0.8.

3.2 Filtering

As mentioned earlier, we obtain transforms between frames to estimate the camera trajectory. A 2D affine transformation defines the change in position between consecutive frames.

$$M_t = \begin{bmatrix} cos(\Delta\theta) & -sin(\Delta\theta) & \Delta x \\ sin(\Delta\theta) & cos(\Delta\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

We obtain the change in position in (x, y) and rotation in θ by extracting , , and $\Delta\theta$ from the transformation matrix into a vector. This vector is defined as m_t, where m_t defines this change in position and rotation for every frame t and its consecutive frame $t + 1$.

$$m_t = \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\theta_t \end{bmatrix}$$

The change in position (x, y) and rotation θ between each frame is cumulatively aggregated to form an estimated camera trajectory, where c_t represents the point in the trajectory corresponding to a frame t and c_{t+1} represents the point in the trajectory for the next frame.

$$c_{t+1} = c_t + m_t \rightarrow c_t = m_1 + m_2 + ... + m_t.$$

We then employ various filters to stabilize the trajectory, c_t, and obtain the stable trajectory, denoted as p_t. With the stable trajectory, we compute transformations warping the input frames into stable frames. To do so, we take the difference between the stable trajectory, p_t, and the original trajectory, c_t. We then apply this to m_t in order to obtain n_t, a vector representing the translation, (Δx,Δy), and rotation, $\Delta\theta$, in the transform to warp the original frame into stable frames.

$$n_t = m_t + (p_t - c_t)$$

To construct the final warp matrix, N_t, we construct the affine transform from the translation, (Δx,Δy), and rotation, $\Delta\theta$, in the n_t vector. N_t denotes the warp matrix that will warp original frames to stable frames for each frame t.

$$N_t = \begin{bmatrix} cos(n_{t2}) & -sin(n_{t2}) & n_{t0} \\ sin(n_{t2}) & cos(n_{t2}) & n_{t1} \\ 0 & 0 & 1 \end{bmatrix}$$

3.2.1 Gaussian

We perform Gaussian smoothing by convolving a one-dimensional Gaussian kernel with t_x, t_y, and θ respectfully, from the p_t vector. We take kernel size to be $5\sigma + 1$, where σ denotes standard deviation. We experiment with various kernel sizes on one video to obtain an optimal kernel size. We evaluate this Gaussian filter with the optimal kernel size on a video dataset, keeping the kernel size proportional to the number of frames in each video from the dataset. Figure 2 shows stabilized trajectories corresponding to various kernel sizes for a video of 630 frames.

3.2.2 Mean

Similarly to Gaussian filtering, we perform mean smoothing by convolving a one-dimensional mean kernel with t_x, t_y, and θ respectfully. After experimenting with various kernel sizes, we determine one optimal kernel size and evaluate the Gaussian filter with that kernel size on the dataset, keeping the kernel size proportional to number of frames in a video. Figure 3 shows stabilized trajectories corresponding to various kernel sizes for the same video as before.

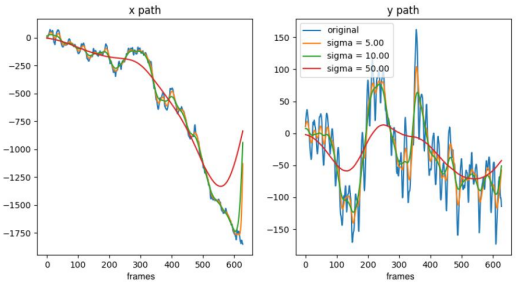

Fig. 2. Gaussian Filtering

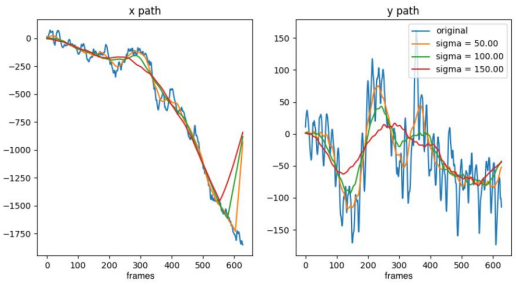

Fig. 3. Mean Filtering

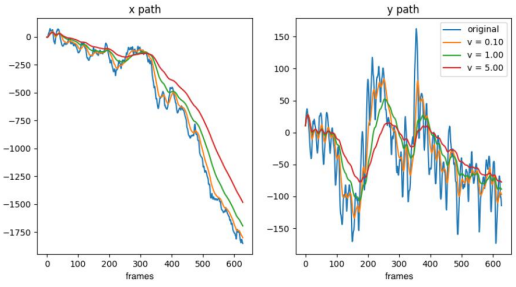

Fig. 4. Kalman Filtering

3.2.3 *Kalman*

We perform Kalman filtering by adopting a position-based model. The intial state value is constructed as the initial point (x,y) of the camera trajectory.

$$\begin{bmatrix} t_{x0} \\ t_{y0} \end{bmatrix}$$

We experiment with various levels of measurement noise on one video to obtain an optimal value. We evaluate the Kalman filter with this measurement noise value on the video dataset. Figure 4 shows stabilized trajectories corresponding to various noise values for the same video.

3.2.4 *Hybrid*

In our evaluation process in section 4, we deemed Gaussian filtering to produce the highest ITF PSNR and Kalman filtering the highest GTF PSNR, so we experiment with a combination of these two filters. Using a Gaussian filter of optimal standard deviation and kernel size, followed by a Kalman filter of optimal measurement noise coefficient, we were able to obtain the following stabilized trajectory shown in Figure 5 for the same video.

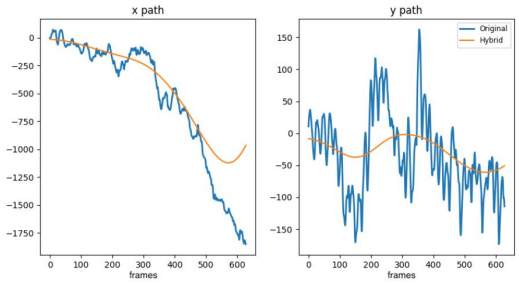

Fig. 5. Hybrid Filter

3.3 L1 Optimization

The linear programming framework proposed in the paper by Grundmann et al. [1] is as follows. We need to obtain the affine transforms between consecutive frames, F_t, which is defined as the affine transform estimating the motion between frame $t+1$ to frame t.

A point on the camera path C_t, corresponding to the point at frame t, is determined by the aggregated transforms of $F_1, F_2, .., F_t$.

$$C_{t+1} = C_t F_{t+1} \rightarrow C_t = F_1 F_2 ... F_t$$

The authors defined the optimized path as P_t, a matrix multiplication between the original camera path C_t and B_t, where B_t is the update transform.

$$P_t = C_t B_t$$

They proposed a linear programming framework to minimize the L1 norm of the first, second, and third derivatives of the optimized path. This objective is illustrated by the objective function below.

$$O(P) = w_1 |D(P)|_1 + w_2 |D^2(P)|_1 + w_3 |D^3(P)|_1$$

With forward differencing, they derive $|D(P)|$ to be the following.

$$|D(P)| = \sum_t P_t |P_{t+1} - P_t| = \sum_t |C_t F_{t+1} B_{t+1} - C_t B_t|$$

The expression is further simplified to the following, with R_t defined as the residual.

$$|D(P)| \leq \sum_t |R_t|,$$
$$\text{where } |R_t| := F_{t+1} B_{t+1} - B$$

This gives residuals for all three objectives:

$$|D(P)|_1 \leq \sum_t |R_t| \tag{1}$$

$$|D^2(P)|_1 \leq \sum_t |R_{t+1} - R_t| \tag{2}$$

$$|D^3(P)|_1 \leq \sum_t |R_{t+2} - 2R_{t+1} + R_t| \tag{3}$$

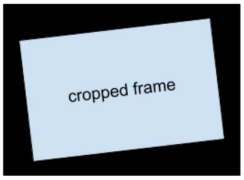

Fig. 6. Cropped stable frame vs. original frame

The final objective function to minimize the L1 norms is the following, where e represents a vector of N slack variables to bound the residual and N is 6 to represent the 6 degrees of freedom (DOF) in an affine transform. c^T is a vector with weights, w_1, w_2, and w_3, for each derivative. For instance, assigning w_1 a higher weight will result in a path with more piece-wise segments of camera path with zero first derivative.

$$\min c^T e$$
$$-e \leq R_t \leq e$$

The paper also proposed proximity and inclusion constraints. The proximity constraints are bounds for the variables, a_t, b_t, c_t, and d_t, in the update transforms, B_t, such that the stable trajectory, P, will not deviate too much from the original, C.

$$B_t = \begin{bmatrix} a_t & b_t & dx_t \\ c_t & d_t & dy_t \end{bmatrix}$$

These variables, a_t, b_t, c_t, d_t, dx_t, and dy_t can be encapsulated by the vector p_t. A matrix U encapsulates the linear combinations with p_t, such that it limits zoom, rotation, skew, and non-uniform scale of the transform. The final proximity constraint is defined below.

$$lb \leq Up_t \leq ub$$

In the paper, the authors also defined the linear combinations as the following. These equations can be formulated into U, the system of equations matrix.

$$0.9 \leq a_t, d_t \leq 1.1$$
$$0.1 \leq b_t, c_t \leq 0.1$$
$$0.05 \leq b_t + c_t \leq 0.05$$
$$0.1 \leq a_t d_t \leq 0.1$$

Stabilization will induce loss of pixel information from the original frame, hence we need to crop the newly stabilized frame. The inclusion constraints proposed by the paper limits the corners of the cropped stable frame, such that they are all contained within the original frame as shown in Figure 6. all the information will be retained. CR_i denotes the 4 corner points, while (w, h) denotes the width and height of the original frame.

$$(0,0)^T \leq CR_i p_t \leq (w,h)^T$$

The final linear programming framework from the paper is shown in Figure 7 Figure 8 shows the original and stabilized camera trajectories of varying weights, w_1, w_2, w_3. The most optimal weight ratio, as proposed by the authors, was $w1 = 10$, $w2 = 1$, $w3 = 100$ show in Figure 9.

Algorithm 1: Summarized LP for the optimal camera path.

Input: Frame pair transforms $F_t,\ t = 1..n$
Output: Optimal camera path $P_t = C_t B_t = C_t A(p_t)$

$$\begin{aligned}
\text{Minimize} \quad & c^T e \\
\text{w.r.t.} \quad & p = (p_1, ..., p_n) \\
\text{where} \quad & e = (e^1, e^2, e^3), e^i = (e^i_1, ..., e^i_n) \\
& c = (w_1, w_2, w_3) \\
\text{subject to} & \\
\text{smoothness} \quad & \begin{cases} -e^1_t \leq R_t(p) \leq e^1_t \\ -e^2_t \leq R_{t+1}(p) - R_t(p) \leq e^2_t \\ -e^3_t \leq R_{t+2}(p) - 2R_{t+1}(p) + R_t(p) \leq e^3_t \\ e^i_t \geq 0 \end{cases} \\
\text{proximity} \quad & \text{lb} \leq Up_t \leq \text{ub} \\
\text{inclusion} \quad & (0,0)^T \leq \text{CR}_i p_t \leq (w,h)^T
\end{aligned}$$

Fig. 7. L1 Optimal Path Linear Programming algorithm

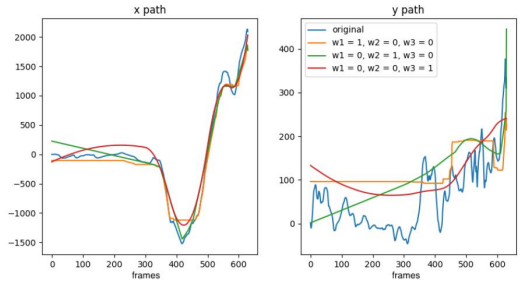

Fig. 8. L1 Optimal Path Weights Comparison

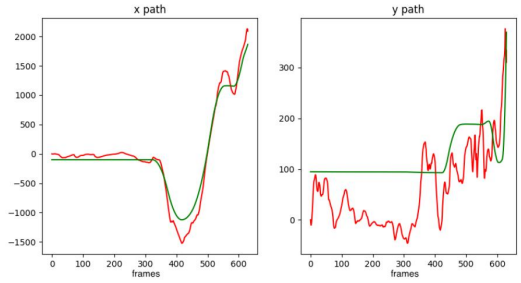

Fig. 9. L1 Optimal Path With Optimal Weights

4 Experimental Results

The metrics we used to quantitatively assess the stabilized video quality was the interframe tranformation fidelity (ITF) and global transformation fidelity (GTF) mentioned in this paper [16]. ITF averages the PSNR between consecutive frames of the stabilized video. GTF averages the PSNR between the input video frame and the stabilized output frame.

ITF is a measure of pixel similarity between consecutive frames and a potential metric for stability. ITF, however, is not an ideal metric for stability. This is due to PSNR being

inconsistent in a video with varying illumination or moving objects. ITF is still widely used for video stabilization quality assessment (VSQA), and can be source of comparison for different methods given that those methods are evaluated on the same videos. GTF is a measure of similarity between original and stabilized frame and shows how much of the information from the input frames are retained.

We evaluate the ITF and GTF metrics across all filtering methods and the L1 optimization method on the DeepStab dataset from the StabNet paper [3].

TABLE 1
Overview of Filtering Methods

	Gaussian	Mean	Kalman	Hybrid
ITF (PSNR)	**27.12** dB	26.65 dB	23.03 dB	19.62 dB
GTF (PSNR)	12.14 dB	12.41 dB	12.35 dB	**12.61 dB**
Crop Ratio (%)	93.57%	94.98%	97.19%	98.05 %

TABLE 2
L1 Optimization

	L1
ITF (PSNR)	25.18 dB
GTF (PSNR)	13.46 dB

4.1 Filtering

The filtering methods do not employ any inclusion or proximity constraints. Due to this, the filtered trajectories can deviate from the original camera path so much that the warp transform will take the stabilized frame out-of-bounds of the original frame. This is the opposite of Figure 6, which shows a stable frame within bounds of the original frame. As a result, we often observe ”blackout” artifacts in the compilation of videos stabilized using filtering in this project.

We introduce a crop ratio metric as a measure of these ”blackout”s. The crop ratio is defined as the ratio of non-black pixels over total pixels. Higher crop ratios are desired, as it means a larger percentage of the video frame can been viewed.

4.1.1 Gaussian

Evidenced by a high ITF in Figure 1, Gaussian performs well in deshaking the video, but it has lower GTF and crop ratio. This is due to a higher amount of blackouts, as some Gaussian-filtered trajectories severely deviate from the original camera trajectory. Observing the X-T slices of a Gaussian-filtered video in Figure 10, we can see the frames are well stabilized. This is a static shot from a YouTube video, so we expect to see nearly the same X-T slice for each frame.

4.1.2 Mean

Compared to Gaussian filtering, mean filtering yields minor improvements in GTF and crop ratio and minor degradation in ITF. Like Gaussian filtering, mean filtering yields higher ITF and performs just as well in deshaking the video. It also suffers in GTF and has more blackouts in stabilized videos. Figure 11 shows the X-T slices from mean filtering, which by visual inspection is nearly the same as the X-T slices from Gaussian filtering.

4.1.3 Kalman

Kalman filtering yields higher GTF and crop ratio than both spatial filters, as well as lower ITF. The Kalman-filtered trajectories adheres to the original camera trajectory more. As a result, Kalman filtering reduces the amount of ”blackout”s, but at the expense of visibly yielding little to no improvement in video jitter. As seen in Figure 12, the X-T slices look visibly different frame to frame.

4.1.4 Hybrid

The hybrid filter was conceived as an attempt to produce higher ITF and GTF given it is a combination of the highest GTF and highest ITF yielding filters. Although GTF is high for the hybrid-filtered paths, ITF is significantly lower than its predecessors. The hybrid filter produces a video that is visually no different from the original shaky video. Like Kalman filtering, the X-T slices in Figure 13 also look visibly different frame to frame.

4.2 L1 Optimization

The evaluation of the L1 optimization method shows its excellence in both ITF and GTF. Due to the proximity and inclusion constraints in linear programming framework, there are no ”blackout” artifacts. Visually, L1 stabilized videos are the smoothest. In Figure 14, the X-T slices are almost identical frame to frame. One observation we made is how unlike the filtered videos, the L1 optimized video is off-centered in relation to the original frame. The scan line for the L1 stabilized X-T slices was raised higher to capture the same X-T slices as in the original video This could also be due to proximity and inclusion constraints. The most optimal path for a smooth trajectory within bounds of the original trajectory has to result in off-centering.

5 Conclusion

In this paper, we discover the challenges surrounding quantitative evaluation of video stabilization (VSQA). These challenges include the absence of ground truth, the scarcity of unstable video datasets, and the qualitative nature of video stabilization, as the notion of stability is also subject to human perception. Additionally, PSNR-based metrics such as ITF exhibit inconsistencies, particularly in scenarios involving moving objects and varying illumination. In the future, we hope to explore alternative stability metrics that better assess stabilized video content. The crop, distortion, and stability metrics, as mentioned in this survey [7], are promising benchmarks for future evaluations of video stabilization efficacy.

Our survey of filtering methods highlights a trade-off between trajectory smoothness and information retention. While spatial filters excel in trajectory smoothing, they often

sacrifice input video information and suffer from "black-out" artifacts under extreme motion. The L1 optimization framework proposed by Grundmann et. al [1] is a more superior method, achieving higher information retention due to inclusion and proximity constraints. However, even this method has its drawbacks, such as a mandatory crop limitation and sensitivity to extreme motion [7]. Regardless, despite the drawbacks, iterative methods are inherently more efficient and more applicable to commercial applications.

In the future, we would like to experiment with other Kalman filter models, such as position velocity models, and explore other components in the stabilization pipeline, such as local outlier rejection in feature tracking and motion imprinting. Given more time and compute resources, we also hope to explore the potential of state of the art deep learning methods and dynamic view synthesis approaches. We can gain further insights on the best directions for video stabilization by investigating deep learning methods, comparing their performance to iterative methods, and evaluating the trade-offs.

REFERENCES

[1] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *CVPR 2011*, 2011, pp. 225–232.

[2] J. Yu and R. Ramamoorthi, "Learning video stabilization using optical flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] M. Wang, G.-Y. Yang, J.-K. Lin, A. Shamir, S.-H. Zhang, S.-P. Lu, and S.-M. Hu, "Deep online video stabilization," 2018.

[4] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, 2006.

[5] Z. Li, Q. Wang, F. Cole, R. Tucker, and N. Snavely, "Dynibar: Neural dynamic image-based rendering," 2023.

[6] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," 2021.

[7] Y. Wang, Q. Huang, C. Jiang, J. Liu, M. Shang, and Z. Miao, "Video stabilization: A comprehensive survey," *Neurocomputing*, vol. 516, pp. 205–230, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122201270X

[8] M. Roberto e Souza, H. d. A. Maia, and H. Pedrini, "Survey on digital video stabilization: Concepts, methods, and challenges," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3494525

[9] S. Ertuerk, "Image sequence stabilization by low-pass filtering of interframe motion," in *Visual Communications and Image Processing 2001*, B. Girod, C. A. Bouman, and E. G. Steinbach, Eds., vol. 4310, International Society for Optics and Photonics. SPIE, 2000, pp. 434 – 442. [Online]. Available: https://doi.org/10.1117/12.411820

[10] P. Rawat and M. D. Sawale, "Gaussian kernel filtering for video stabilization," in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, 2017, pp. 142–147.

[11] L. Kejriwal and I. Singh, "A hybrid filtering approach of digital video stabilization for uav using kalman and low pass filter," *Procedia Computer Science*, vol. 93, pp. 359–366, 2016, proceedings of the 6th International Conference on Advances in Computing and Communications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916314624

[12] W. Guilluy, L. Oudre, and A. Beghdadi, "Video stabilization: Overview, challenges and perspectives," *Signal Processing: Image Communication*, vol. 90, p. 116015, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596520301697

[13] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[14] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," Intel Corporation-Microprocessor Research Labs, Tech. Rep., January 2001.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[16] C. Morimoto and R. Chellappa, "Evaluation of image stabilization algorithms," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, vol. 5, 1998, pp. 2789–2792 vol.5.

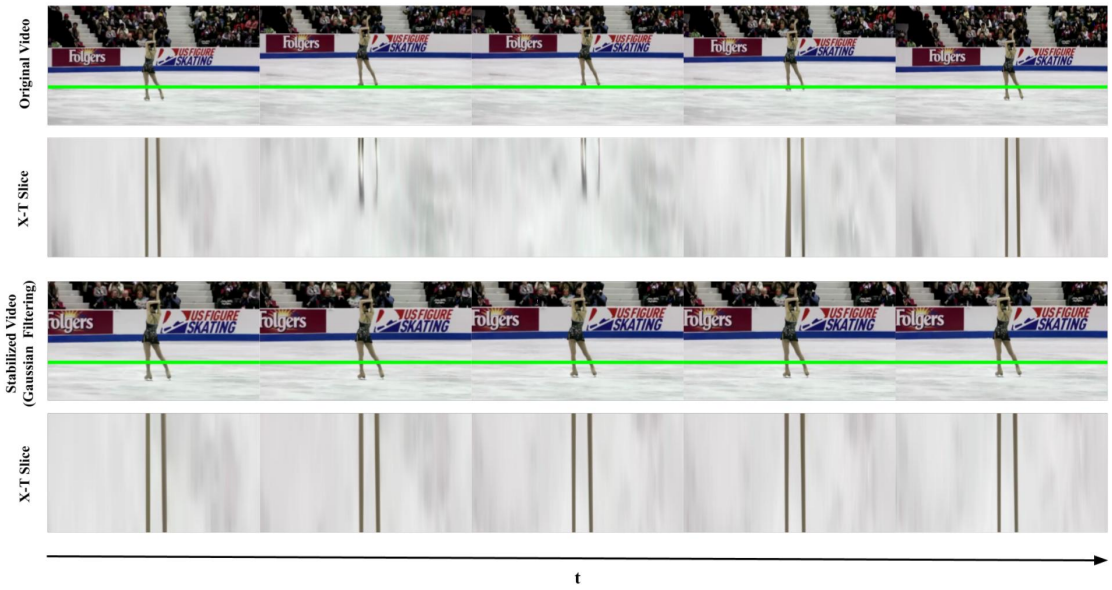

Fig. 10. X-T Slice of Gaussian filtered video

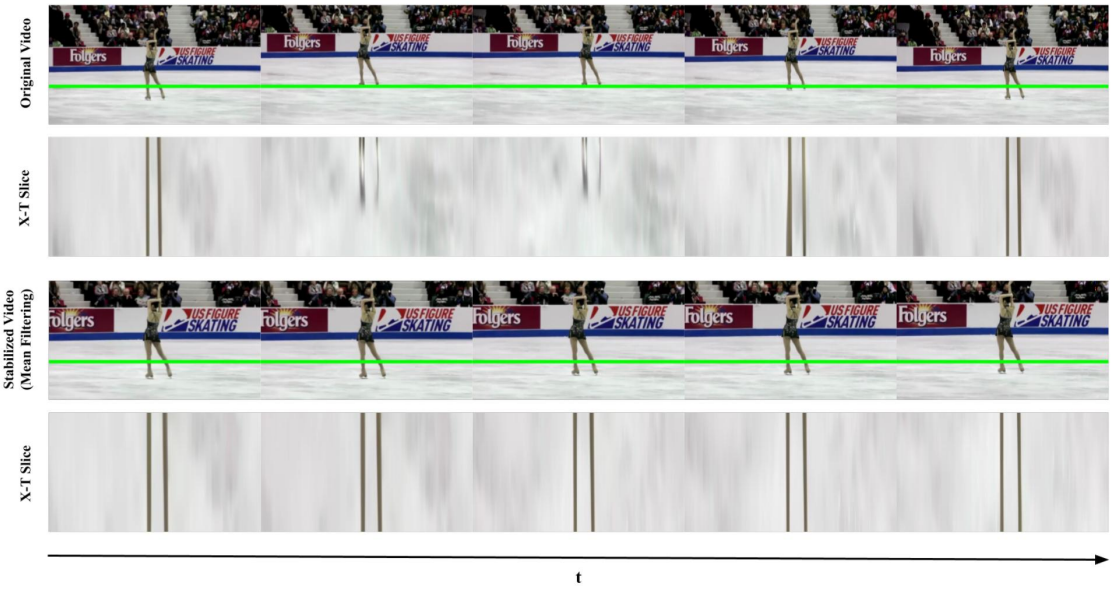

Fig. 11. X-T Slice of Mean filtered video

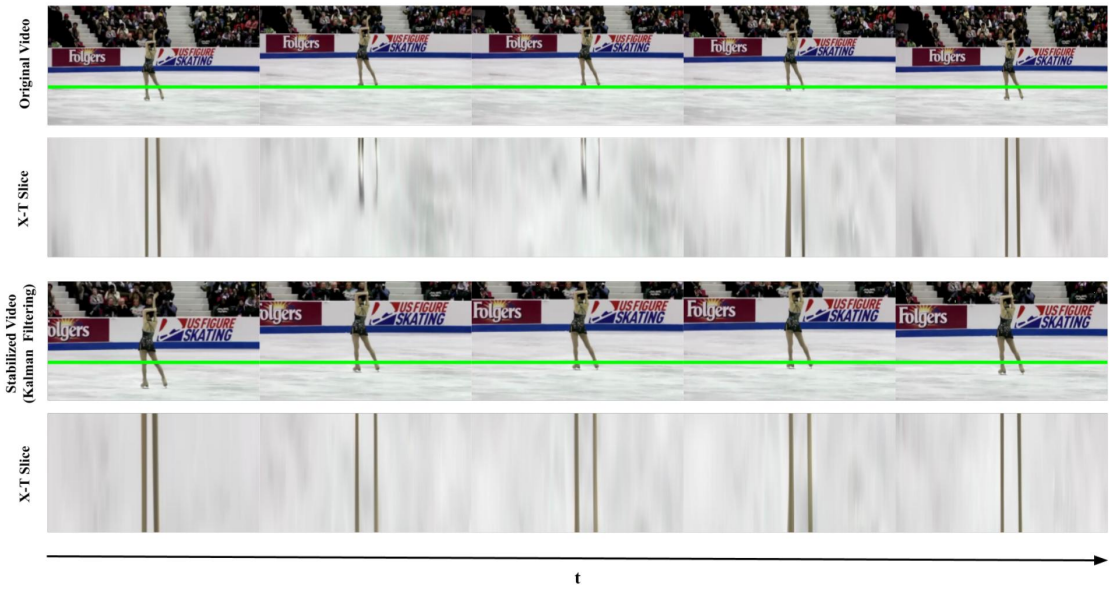

Fig. 12. X-T Slice of Kalman filtered video

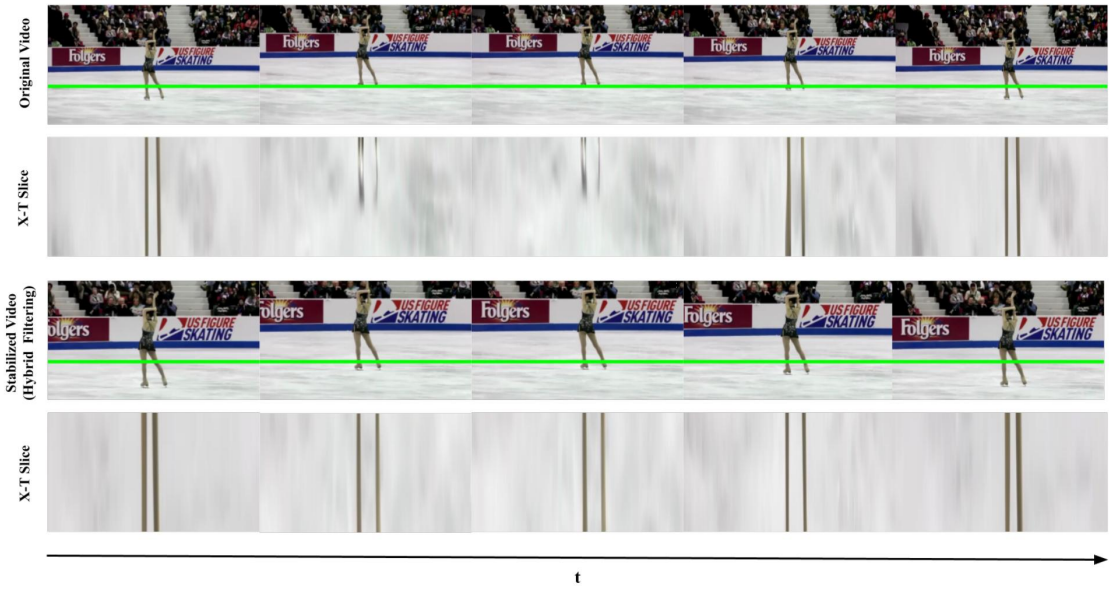

Fig. 13. X-T Slice of Hybrid filtered video

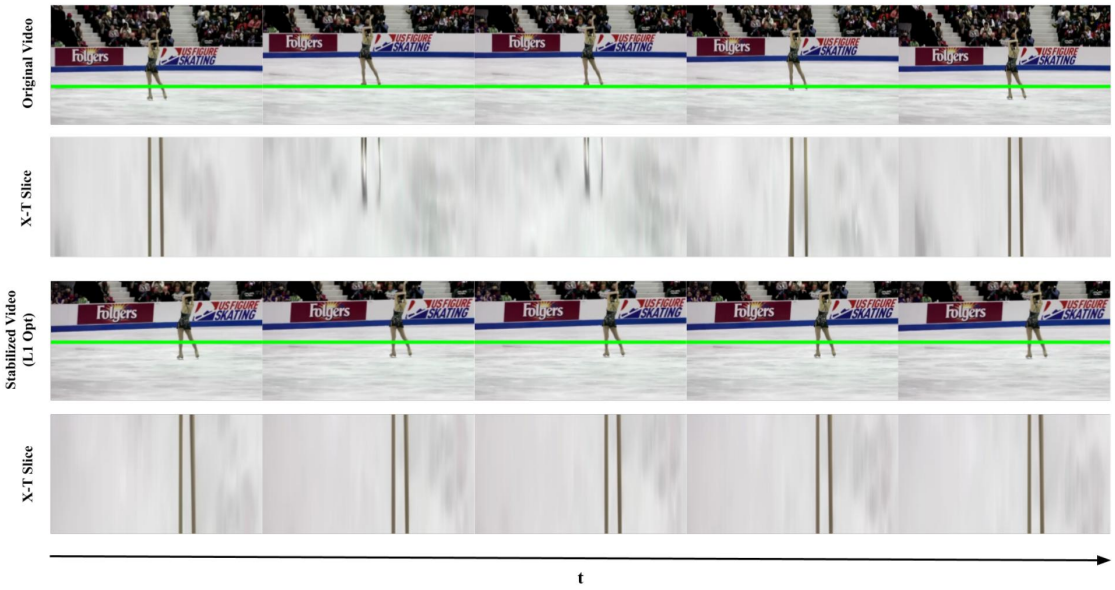

Fig. 14. X-T Slice of L1 optimized video