Deep Online Fused Video Stabilization

Zhenmei Shi
University of Wisconsin Madison

Fuhao Shi
Google

Wei-Sheng Lai
Google

Chia-Kai Liang
Google

Yingyu Liang
University of Wisconsin Madison

Abstract

We present a deep neural network (DNN) that uses both sensor data (gyroscope) and image content (optical flow) to stabilize videos through unsupervised learning. The network fuses optical flow with real/virtual camera pose histories into a joint motion representation. Next, the LSTM block infers the new virtual camera pose, and this virtual pose is used to generate a warping grid that stabilizes the frame. Novel relative motion representation as well as a multi-stage training process are presented to optimize our model without any supervision. To the best of our knowledge, this is the first DNN solution that adopts both sensor data and image for stabilization. We validate the proposed framework through ablation studies and demonstrated the proposed method outperforms the state-of-art alternative solutions via quantitative evaluations and a user study. Check out our video results and dataset at our website.

1. Introduction

Videos captured with a hand-held device are often shaky even with the optical image stabilizer (OIS), which can suppress small motion blur but not the large motion as the operating range is bound to 1-2 degrees due to the physic constraint. With the growing popularity of casual video recording, live streaming, and movie-making on hand-held smartphones, effective and efficient video stabilization is crucial for improving overall video quality and user experience. However, high-quality stabilization remains challenging due to complex camera motions such as walking or running and scene variations in lighting and compositions.

Despite decades of research in computer vision, several existing approaches [7, 15, 19] rely on the input video frames for motion estimation (e.g., affine or homography transform), which often fail when the video contains fast or large motion. Predicting dense flow field [4, 30, 32, 33, 35] is able to handle more complex motion such as parallax. However, the video frames warped by optical flow often suffer from visible non-rigid distortion and artifacts. On the contrary, the electronic image stabilization (EIS) in smartphones now all use motion sensor data, e.g., gyroscope and accelerometer, to obtain accurate motion and achieve impressive stabilization results [14, 29]. Nevertheless, the sensor-only solutions [1, 14, 24] cannot distinguish close/far-away subjects, leading to more residual parallax motions for close scenes. Moreover, most existing algorithms are offline and not suitable for online applications such as live-streaming. The lack of public video+sensor datasets hinders research in this direction.

In this work, we present the deep Fused Video Stabilization (deep-FVS) to address the above-mentioned issues. Our goals are to 1) estimate accurate motion from both the sensor and video content, and 2) develop an efficient online stabilization method. Our solution consists of three main stages. First, we use an encoder to fuse optical flows, real camera poses (from gyroscope), and virtual camera poses (from the output of previous frames) into a joint motion representation. Then, we predict virtual camera poses with an LSTM [9] and fully-connected layers. Finally, we apply grid-based warping based on predicted camera poses to stabilize the frame and remove its rolling shutter distortion. Our network is trained with unsupervised learning with carefully designed loss functions and a multi-stage training procedure. Fig. 1 shows an overview of conventional methods [7, 19], recent learning-based approaches [4, 30, 31, 33, 35], and the proposed deep-FVS.

As the existing datasets [30, 19] do not record the sensor data, we collect a new video dataset that contains videos with both gyroscope and OIS data for training and evaluation. Our dataset covers diverse scenarios with different illumination conditions and camera/subject motions. We evaluate the proposed solution objectively and subjectively and show that it outperforms state-of-the-art methods by generating more stable and distortion-free results.

This paper makes the following contributions:

- The first DNN-based framework that fuses motion sensor data and optical flow for online video stabilization.
- The relative motion representation and unsupervised losses to optimize the proposed model.

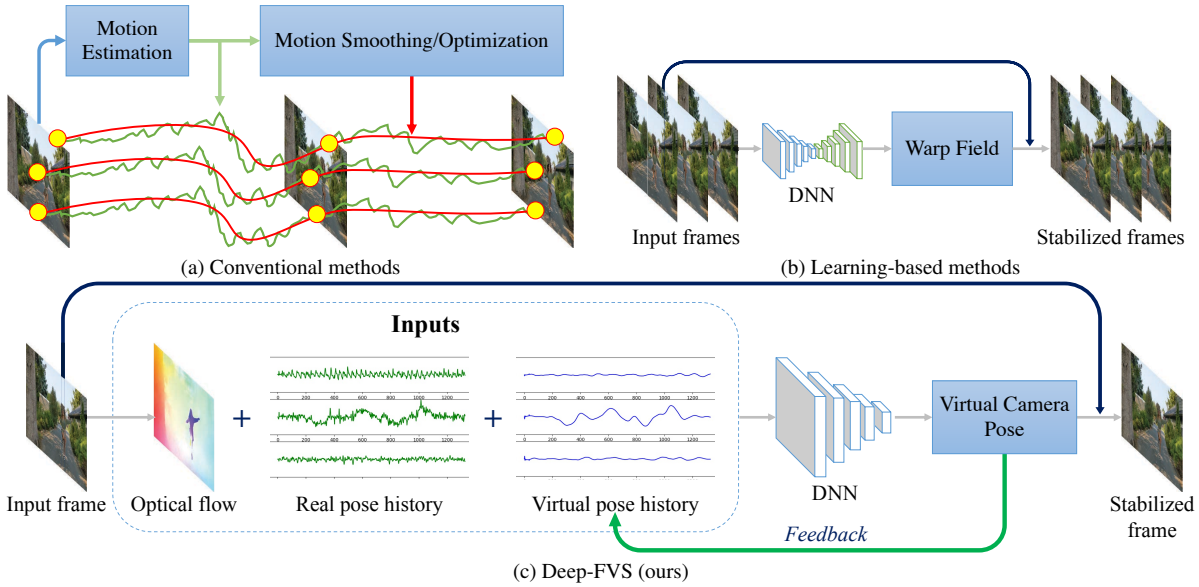


Figure 1: **Comparisons of existing video stabilization methods and the proposed method.** (a) Conventional video stabilization methods [7, 19] estimate camera motions based on image feature trajectories and find a smooth camera path to render a stabilized video. (b) Learning-based approaches [4, 30, 31, 33, 35] learn deep networks to predict warp fields for warping the input video. (c) The proposed method learn to stabilize a video from the optical flow and gyroscope data.

- A video dataset that contains videos with gyroscope and OIS sensor data and covers various scenarios. Both the dataset and code will be publicly released.

2. Related Work

Conventional methods. Classical video stabilization algorithms typically involve motion estimation, camera path smoothing, and video frame warping/rendering steps [23]. Some solutions also correct the rolling shutter distortions [6, 11, 13]. Those methods can be categorized into 3D, 2D, and 2.5D approaches based on motion estimation.

The 3D approaches model the camera poses and estimate a smooth virtual camera trajectory in the 3D space. To find 6DoF camera poses, several techniques have been adopted, including projective 3D reconstruction [2], depth camera [18], structure from motion [15], and light-field [28]. While 3D approaches can handle parallax and produce high-quality results, they often entail expensive computational cost or require specific hardware devices.

The 2D approaches represent and estimate camera motions as a series of 2D affine or perspective transformations [7, 19, 22]. Robust feature tracking and outlier rejection are applied to obtain reliable estimation [34]. Liu et al. [20] replace feature trajectories with optical flow to handle spatially-variant motion. Early approaches apply low-pass filters to smooth individual motion parameters [3, 22], while recent ones adopt $\mathcal{L}_1$ optimization [7] and joint optimization with bundled local camera paths [19] for the entire video. Some hybrid 2D-3D approaches exploit the subspace constraints [16] and epipolar geometry [5]. Zhuang et al. [36] smooth 3D rotation from the gyroscope and stabilize the residual 2D motion based on feature matching.

The above methods often process a video offline, which are not suitable for live-streaming and mobile use cases. Liu et al. [17] propose a MeshFlow motion model with only one frame latency for online video stabilization. A mobile online solution using both the OIS and EIS is developed in [14]. In this work, we utilize the OIS, gyroscope, and optical flow to learn a deep network for stabilization. Our online method has only a few frames latency and does not require per-video optimization.

Learning-based methods. With the success of deep learning on image recognition [8, 21, 25], DNNs have been adopted to several computer vision tasks and achieved state-of-the-art performance. However, DNN based video stabilization still does not attract much attention, mainly due to the lack of proper training data. Wang et al. [30] collect the DeepStab dataset with 60 pairs of stable/unstable videos, and train a deep CNN to predict mesh-grids for warping the video. Instead of predicting low-resolution mesh-grids, the PWStableNet [35] learns dense 2D warping fields to stabilize the video. Xu et al. [31] train a generative adversarial network to generate a steady frame as guidance and use the spatial transformer network to extract the affine transform for warping the video frames. Yu and Ramamoorthi [32] take optical flows as input and optimize the weights of a deep network to generate a warp field for each specific

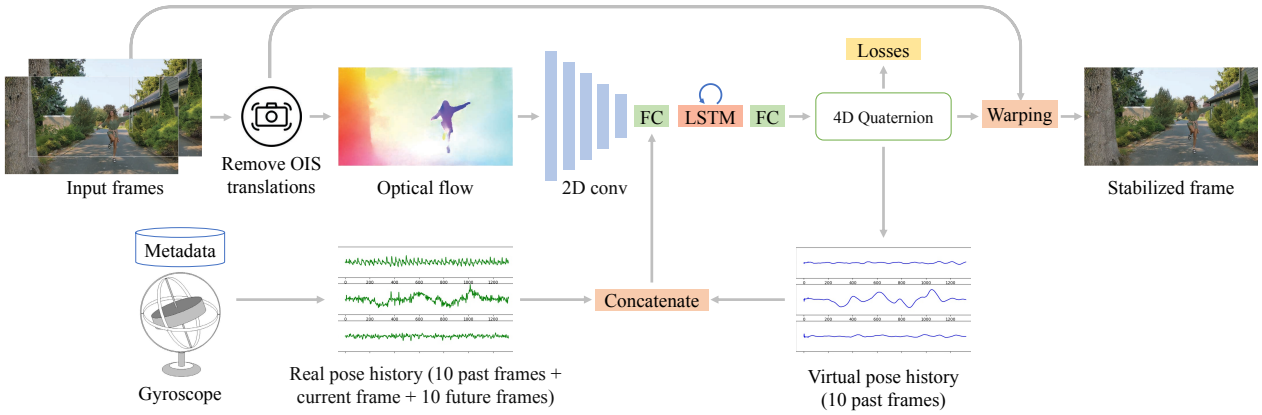


Figure 2: **Overview of deep-FVS.** Given an input video, we first remove the OIS translation to extract the raw optical flow. We also obtain the real camera poses from the gyroscope and convert it to a relative quaternion. An encoder with 2D convolutions embeds optical flows to a latent representation, which is then concatenated with the real and virtual camera poses. This joint motion representation is fed to a LSTM cell and FC layers to predict the new virtual camera pose as a quaternion. Finally, we warp the input frame based on the OIS and virtual camera pose to generate the stabilized frame.

video. They further train a stabilization network that can be generalized test videos without optimization [33]. Choi et al. [4] learn a frame interpolation model to iteratively interpolate the input video into a stable one without cropping.

These learning-based methods learn to stabilize videos from the video content and optical flow. Their performance heavily depends on the training data and can suffer from visible distortion for large motions (e.g., running). In contrast, we use the gyroscope to measure large camera motions and utilize optical flow jointly to achieve video stability.

3. Deep Fused Video Stabilization

The overview of our method is shown in Fig. 2. We first process the gyroscope and OIS reading so that we can query the real camera extrinsic (i.e., rotation) and intrinsic (i.e., principal point offsets) at arbitrary timestamps (Sec. 3.1). We then remove the OIS translations on the input video and extract optical flows from the raw video frames (Sec. 4.1). The optical flows are encoded to a latent space via 2D convolutional layers and concatenated with the real camera poses within a temporal window and the previous virtual camera poses as a joint motion representation (Sec. 3.2 and 4.2). Next, we feed this joint motion representation to an LSTM cell and a few fully-connected layers to predict a virtual camera pose at the current timestamp. Finally, we generate a warping grid from the input camera rotations, OIS movement, and the predicted virtual camera poses to warp the input frame as the stabilized frame (Sec. 3.3). Our solution stabilizes a video frame-by-frame and is suitable for online processing.

In training, we randomly select long sub-sequences from the training videos. We optimize our DNN with a set of loss functions without any ground-truth video or camera poses for supervision (Sec. 4.3). To stabilize the training, we adopt a multi-stage training strategy to constrain the solution space (Sec. 4.4).

3.1. Gyroscope and OIS Pre-processing

In our dataset, the gyroscope $(\omega_x, \omega_y, \omega_z, t)$ and OIS (o_x, o_y, t) are sampled at 200 Hz, where ω is the angular velocity, and o_x, o_y are the OIS movements. The camera rotation is integrated by $R(t) = S\omega(t) * R(t - S)$, where S is the sampling interval (5ms). We represent the rotation as a 4D quaternion and save it in a queue. To obtain the camera rotation at an arbitrary timestamp t_f, we first locate the two consecutive gyro samples a, b in the queue such that $t_a \leq t_f \leq t_b$, and obtain $R(t_f)$ by applying a spherical linear interpolation (SLERP):

$$R(t_f) = \texttt{SLERP}(R(t_a), R(t_b), (t_b - t_f)/(t_b - t_a)). \quad (1)$$

Similarly, $O(t)$ is calculated from a linear interpolation between $O(t_a)$ and $O(t_b)$.

3.2. Camera Pose Representation

We represent a camera pose as $P = (R, O)$, where R is the camera rotation and $O = (o_x, o_y)$ is a 2D offset to the camera principal point (u, v). Given a 3D world coordinate X, the projected point on the 2D image at timestamp t is

$$x = K(t)R(t)X, \quad (2)$$

where $K(t) = [f, 0, u + o_x(t); 0, f, v + o_y(t); 0, 0, 1]$ is the intrinsic matrix with focal length f.

Given a real camera pose $P_r = (R_r, O_r)$ and virtual one $P_v = (R_v, O_v)$, the transformation of a point from the real

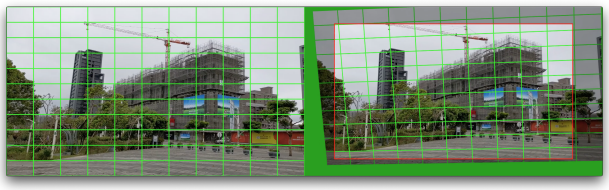


Figure 3: **Grid-based frame warping.** Left: the input frame with a regular 12x12 grid. Right: the warped frame with a virtual camera pose. The red bounding box shows the output frame after cropping. Note the rolling shutter distortion is also corrected (curved lines are straightened).

camera space to the virtual (stabilized) one is

$$x_v = K_v(t)R_v(t)R_r^{-1}(t)K_r^{-1}(t)x_r, \tag{3}$$

where x_r, x_v are the 2D image coordinates at real and virtual camera spaces, respectively. In all the experiments, we normalize $f = 1.27$ for both the real and virtual cameras.

3.3. Grid-based Frame Warping

We use a grid-based warping similar to Karpenko et al. [11] to jointly stabilize video frames and remove the rolling shutter distortion. For each frame, we record the timestamp at the start of frame exposure t_f, length of rolling shutter l_{rs}, exposure duration l_{exp}, and other frame metadata (e.g., focal length, sensor size). We divide a frame into M columns and S horizontal stripes, where each stripe has its unique timestamp (see Fig. 3). By warping all stripes to a virtual camera pose P_v, we can correct the rolling shutter distortion. Specifically, the warping grid is generated as

$$x_v(i,j) = K_v R_v R_r^{-1}(t_i) K_r^{-1}(t_i) x_r(i,j), \tag{4}$$

where $t_i = t_f + l_{exp}/2 + l_{rs}/S * i$ is the stripe timestamp at row i. $x_r(i,j)$ is the 2D location on row i and column j. We set the mesh dimension to 12×12 in all of our experiments.

4. Sensor Fused Model Learning

We now describe the core of our deep fused video stabilization network. As shown in Fig. 2, our network consists of a sequence of 2D convolutional layers to encode the optical flow, an LSTM cell to fuse the latent motion representation and maintain temporal information, and fully-connected layers to decode the latent representation to virtual camera poses. The detailed network configuration is provided in the supplementary material.

We first extract the OIS-free optical flow from the input frames and OIS data (Sec. 4.1) and map it to a low-dimensional representation z. Meanwhile, we extract the past and future real camera rotation history H_r and the past virtual rotation history H_v from the queues (Sec. 4.2). We define the joint motion representation as $[z, H_r, H_v]$ and feed it into the LSTM to predict an incremental rotation $\Delta R_v(t)$ to the previous virtual pose $R_v(t - \Delta t)$, where Δt is fixed to 40ms in our experiments and is invariant to the video frame rate. Note we set the virtual offset O_v to 0. The final virtual pose is then calculated as $P_v = (\Delta R_v(t) R_v(t - \Delta t), O_v)$ and used to generate the warping grid (Sec. 3.3). It is also pushed into the virtual pose queue as the input for later frames. We can interpret the LSTM, virtual pose prediction, and frame warping steps as a decoder that maps the current motion state $[z, H_r, H_v]$ to a stabilized frame.

4.1. OIS-free Optical Flow

Some camera motions in the input videos are compensated by the OIS to reduce the motion blur. Although the OIS movement depends on the hand motion, the offset O_r is different at each scanline due to rolling shutter and more like a random noise (see the supplementary materials for more discussions). It is non-trivial to let the network learn to associate the local offset with the principal point changes.

To address this issue, we remove OIS motions when estimating the optical flow such that the input to our model contains only the camera and object motions. Specifically, we denote the position of a pixel in frame n as $x_{r,n}$ and its corresponding pixel in frame $n+1$ as $y_{r,n+1}$. The raw forward optical flow can be represented as

$$\tilde{F}_n^{n+1} = y_{r,n+1} - x_{r,n}. \tag{5}$$

By reverting the OIS movement at the pixel's timestamp (which depends on the y-coordinate due to the rolling shutter readout), $x_{r,n}$ and $y_{r,n+1}$ are mapped to $x_{r,n} - O(t_{x_{r,n}})$ and $y_{r,n+1} - O(t_{y_{r,n+1}})$, respectively. The forward optical flow is then adjusted to

$$\begin{aligned} F_n^{n+1} &= (y_{r,n+1} - O(t_{y_{r,n+1}})) - (x_{r,n} - O(t_{x_{r,n}})) \\ &= \tilde{F}_n^{n+1} - (O(t_{y_{r,n+1}}) - O(t_{x_{r,n}})). \end{aligned} \tag{6}$$

The backward flow is adjusted similarly. We use the pre-trained FlowNet2 [10] to extract optical flows in our experiments.

4.2. Relative Rotation based Motion History

To obtain the real and virtual pose histories $[H_r, H_v]$ at a timestamp t, we first sample N past and future timestamps from the gyro queue (Sec. 3.1) and obtain the real absolute camera rotations $H_{r,\text{absolute}} = (R_r(t - N\Delta t), ..., R_r(t), ..., R_r(t + N\Delta t))$. Meanwhile, we sample the virtual pose queue to obtain the virtual camera pose history as $H_{v,\text{absolute}} = (R_v(t - N\Delta t), ..., R_v(t - \Delta t))$.

One key novelty here is to convert the absolute poses,

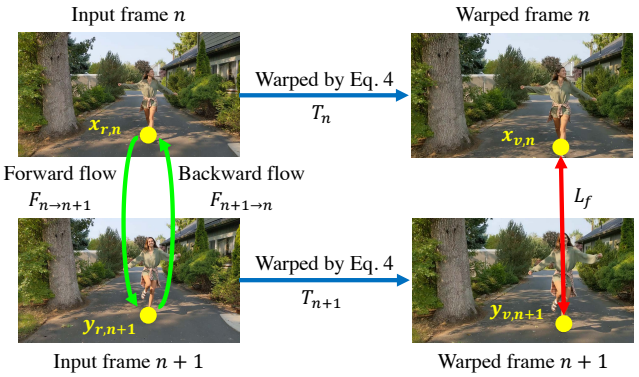


Figure 4: **Optical flow loss.** The optical flow loss aims to minimize the distance between $x_{v,n}$ and $y_{v,n+1}$ in the virtual camera space. By incorporating the forward and backward flows, we define our optical flow loss as in (15).

which are integrated from the very first frame, into a *relative* rotation w.r.t. the current real camera pose:

$$H_r = H_{r,\text{absolute}} * R_r^{-1}(t), \tag{7}$$
$$H_v = H_{v,\text{absolute}} * R_r^{-1}(t). \tag{8}$$

The network output is also a relative rotation to the previous virtual camera pose. Therefore, our model only needs to learn the first order pose changes and is invariant to the absolute poses. Our experiments show that this relative rotation representation leads to more stable predictions and provides a much better generalization (Sec. 5.3).

4.3. Loss Functions

We define the following loss functions to train our network. These loss functions can be evaluated without any ground-truth. Note that we omit the timestamp or frame index in some terms (e.g., L instead of $L(t)$) for simplicity.

Smoothness losses. We measure the C^0 and C^1 smoothness of the virtual camera poses by

$$L_{C^0} = ||R_v(t) - R_v(t-\Delta t)||^2, \tag{9}$$
$$L_{C^1} = ||R_v(t)R_v^{-1}(t-\Delta t) - R_v(t-\Delta t)R_v^{-1}(t-2\Delta t)||^2. \tag{10}$$

These two losses encourage the virtual camera to be stable and vary smoothly.

Protrusion loss. To avoid undefined regions and excessive cropping on the stabilized video, we measure how the warped frame protrudes the real frame boundary [27]:

$$L_p = \sum_{i=0}^{N} w_{p,i} ||\texttt{protrude}(P_v(t), P_r(t+i\Delta t))/\alpha||^2, \tag{11}$$

where N is the number of look-ahead frames, $w_{p,i}$ is the normalized Gaussian weights (with a standard deviation σ) centered at the current frame, and α is a reference protrusion value that we can tolerate. To evaluate `protrude`, we project the virtual frame corners to the real camera space using (3) and measure the max normalized signed distance between the four warped corners to the frame boundary. We set $\sigma = 2.5$, $N = 10$ and $\alpha = 0.2$ in our experiments.

Distortion loss. We measure the warping distortion by:

$$L_d = \Omega(R_v, R_r)/(1 + e^{(-\beta_1(\Omega(R_v,R_r)-\beta_0)}), \tag{12}$$

where $\Omega(R_v, R_r)$ is the spherical angle between the current virtual and real camera poses, β_0 is a threshold and β_1 is a parameter to control the slope of the logistic function. This loss is only effective when the angle deviation is larger than a threshold. We empirically set $\beta_0 = 6°$ and $\beta_1 = 100$ in our experiments.

Optical flow loss. We adopt an optical flow loss similar to [32] to minimize the pixel motion between adjacent frames. As shown in Fig. 4, let $x_{r,n}$ and $y_{r,n+1}$ be the correspondences between frame n and $n+1$ in the real camera space. We define the transform from the real camera space to the virtual camera space in Sec. 3.3 as T, and obtain $x_{v,n} = T_n(x_{r,n})$ and $y_{v,n+1} = T_{n+1}(y_{r,n+1})$ in the virtual camera space. By incorporating the forward flow F_n^{n+1} and backward flow F_{n+1}^n, the warped pixels can be represented as:

$$x_{v,n} = T_n(x_{r,n}) = T_n(y_{r,n+1} + F_{n+1}^n), \tag{13}$$
$$y_{v,n+1} = T_{n+1}(y_{r,n+1}) = T_{n+1}(x_{r,n} + F_n^{n+1}). \tag{14}$$

Our goal is to minimize $||x_{v,n} - y_{v,n+1}||^2$ so they stay close in the stabilized video. This can be measured by:

$$L_f = |X_n|^{-1} \sum_{X_n} ||x_{v,n} - T_{n+1}(x_{r,n} + F_n^{n+1})||^2 + |X_{n+1}|^{-1} \sum_{X_{n+1}} ||y_{v,n+1} - T_n(y_{r,n+1} + F_{n+1}^n)||^2, \tag{15}$$

where X_n is the set of all pixel positions in frame n except those fall into undefined regions after warping.

Overall loss. Our final loss at a timestamp t is the weighted summation of the above loss terms:

$$L = w_{C^0}L_{C^0} + w_{C^1}L_{C^1} + w_pL_p + w_dL_d + w_fL_f, \tag{16}$$

where $w_{C^0}, w_{C^1}, w_p, w_d$ and w_f are set to $10, 5, 0.2, 1$ and 10 respectively in our experiments.

At each training iteration, we forward a sub-sequences with 100 frames to evaluate the losses and accumulate gradients before updating the model parameters.

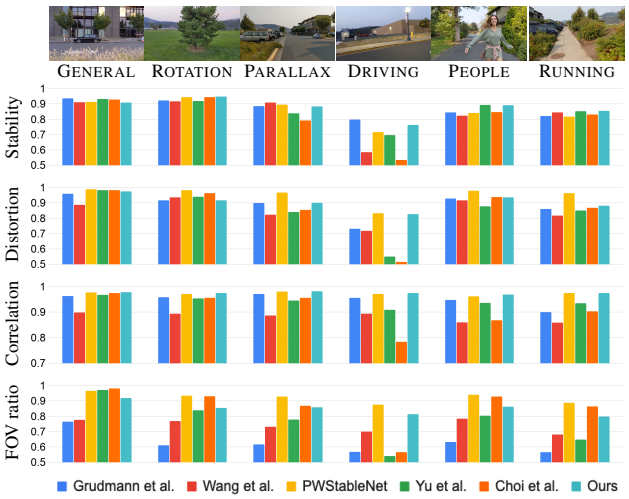


Figure 5: **Per-category quantitative evaluation.** We compare the stability, FOV ratio, distortion, and correlation with state-of-the-art methods [4, 7, 30, 33, 35] on each category.

4.4. Multi-Stage Training

For the virtual camera poses, there is a trade-off between following the real camera motion and staying stable. Although we have defined loss terms in (16) to constrain the solution space, it is difficult for the network to learn this non-linearity - the training cannot converge when we optimize all the loss terms simultaneously.

We adopt a multi-stage training to address this issue. In the first stage, we only minimize L_{C^0}, L_{C^1}, and L_d to ensure that our model can generate a meaningful camera pose. In the second stage, L_p is added to reduce the undefined regions in the output. In the last stage, L_f is included to enhance the overall quality. We train each stage for 200, 100, and 500 iterations. To improve the model generalization, we adopt a data augmentation by randomly changing the virtual camera poses (within ± 6 degrees) to model possible real-virtual pose deviations in the test sequences.

5. Experimental Results

In this section, we show that our deep-FVS achieves state-of-the-art results in quantitative analysis (Sec. 5.1) and a user study (Sec. 5.2). We then validate the effectiveness of the key components in the proposed framework by ablation study (Sec. 5.3). We also strongly encourage readers to watch the source and stabilized videos (by our and existing methods) in the supplemental materials.

5.1. Comparisons with State-of-the-Arts

Experimental settings. We compare our deep-FVS with a conventional method [7][1] and 4 recent learning-based methods [4, 30, 33, 35][2]. We collect 50 videos with sensor logs using Google Pixel 4, which records videos in 1920×1080 resolution with variable FPS. The video dataset covers a wide range of variations, such as scenes, illuminations, and motion. We split our dataset into 16 videos for training and 34 videos for testing, where the test set classified into 6 categories: GENERAL, ROTATION, PARALLAX, DRIVING, PEOPLE, and RUNNING. Fig. 5 shows a few sample frames from each category.

Method	Stability	Distortion	Correlation	FOV Ratio
Grundmann et al. [7]	0.866	0.897	0.949	0.624
Wang et al. [30]	0.859	0.852	0.877	0.739
PWStableNet [35]	0.862	**0.966**	0.973	**0.924**
Yu et al. [33]	0.862	0.856	0.942	0.770
Choi et al. [4]	0.822	0.878	0.918	0.881
Ours	**0.880**	0.911	**0.976**	0.851

Table 1: **Quantitative results.** The best one is marked in **bold red** and the second best one is marked in underline blue.

Quantitative comparisons. We use four metrics: *Stability* [19], *Distortion* [19], *FOV ratio*, and *Correlation*, to evaluate the performance of the tested methods (please refer to the supplemental materials on their definitions). We note that the distortion measures the global geometry distortion, while the correlation evaluates the local deformation.

The results of all test videos are summarized in Table 1, and Fig. 5 plot the average scores for the 6 categories. Overall, our method achieves the best stability and correlation scores. For the distortion score, our method is comparable to PWStableNet [35] on average. Our method generally obtains better stability and correlation scores on challenging ROTATION, RUNNING, and PEOPLE categories. Note that while PWStableNet [35] has high distortion scores and FOV ratios, their results contain lots of residual global motions and temporal wobbling, which cannot be characterized by existing metrics. Please refer to our supplemental videos for the full video comparisons.

Qualitative comparisons. We provide visual comparisons of stabilized frames in Fig. 6. Both Yu et al. [33] and Choi et al. [4] use optical flows to warp the frames and often generate local distortion. Choi et al. [4] produce severe artifacts when the motion is large (e.g., running and driving). Grundmann et al. [7] estimate a global transformation, and Wang et al. [30] predict low-resolution warping grids. The results of both methods have less local distortion but are not temporally stable as the motion is purely estimated from the video content. In contrast, we fuse both the gyroscope data and optical flow for more accurate motion inference and obtain

[1] We use a third-party implementation from `https://github.com/ishit/L1Stabilizer`.

[2] The source code of [4, 30, 35] are publicly available. We obtain the source code of [33] from the authors.

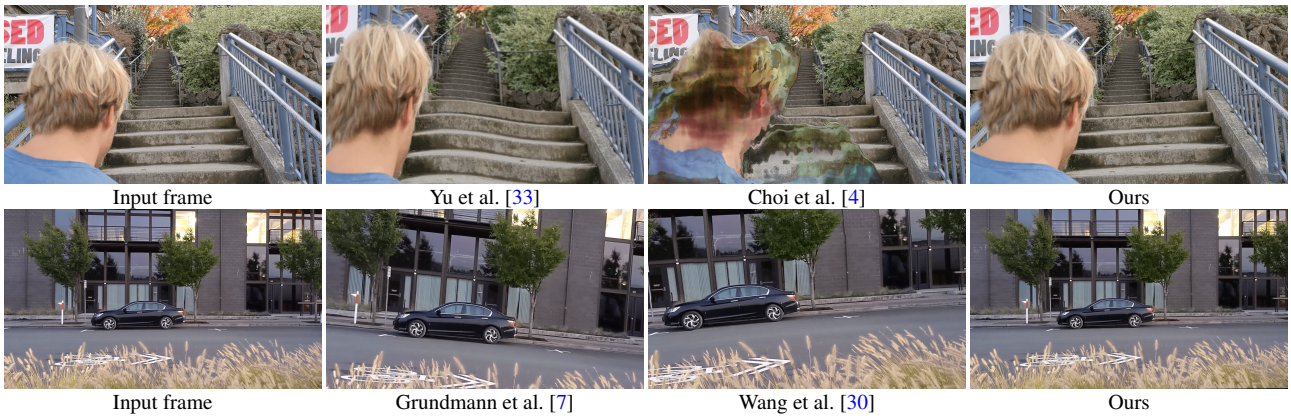


Figure 6: **Visual comparisons.** Non-rigid distortion, local artifacts and temporal wobbling are observed in Yu et al. [33] and Choi et al. [4], and large rotation deviation observed in Grundmann et al. [7] and Wang et al. [30] (which also exhibits local distortions). Our method is free of such issues. Please refer to our supplemental videos on the full results of all the methods.

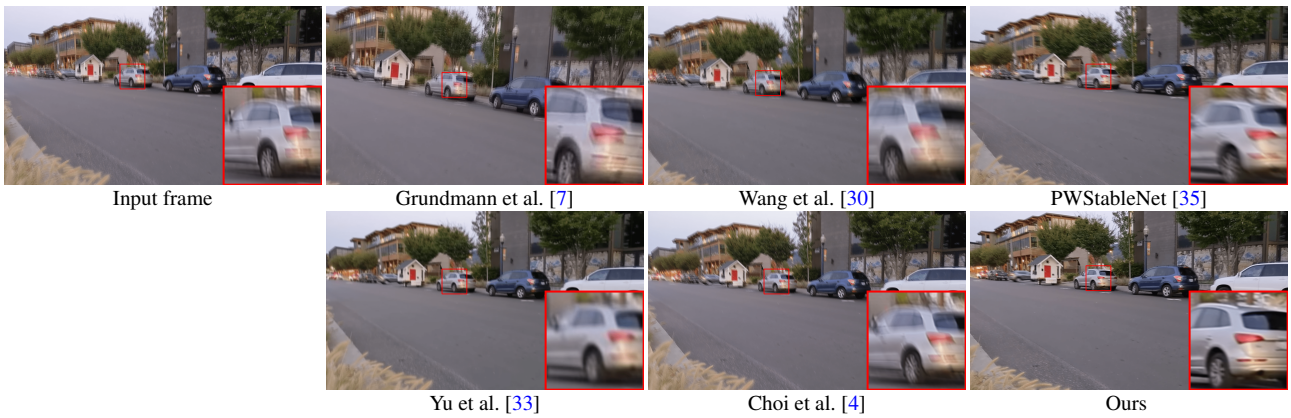


Figure 7: **Stability comparisons.** We take a video with almost no camera motion except handshakes and average 11 adjacent frames. Our average frame is sharper than other methods, indicating that our result is more stable. Please refer to our supplemental videos on the full results of all the methods.

stable results without distortion or wobbling.

To further compare the stability of output videos, we compute the averaged frames (from 11 adjacent frames) from a short clip, where the input video contains only handshake motion. Ideally, the stabilized video should look static as it was captured on a tripod. In Fig. 7, our result is the sharpest one, while the averaged frames from other approaches [4, 7, 30, 33, 35] are blurry, demonstrating that our stabilized video is more stable than others. Please refer to our supplemental videos for the full video comparisons.

5.2. User Study

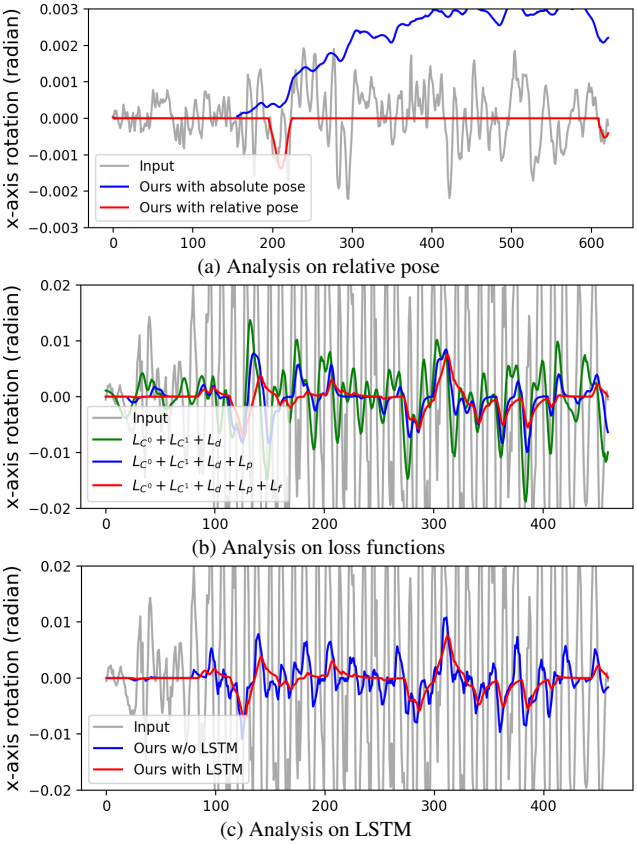
As the evaluation metrics in Sec. 5.1 may not reflect all the artifacts in videos, we conduct a user study to evaluate human's preferences on the stabilized videos. As it is easier for a user to make judgement between two results instead of ranking multiple videos, we adopt the paired comparison [12, 26] to measure the subject preference. In each test, we show two stabilized videos side-by-side and the input video as a reference. The participant is asked to answer the following questions:

1. Which video is more stable?
2. Which video has less distortion?
3. Which video has a larger FOV?

In total, we recruit 44 participants, where each participant evaluates 15 pairs of videos. While the results are shuffled randomly, we ensure that all the methods are compared the same number of times. The results are summarized in Table 2. Overall, our method is selected on more than 91% of comparisons for the first two questions, demonstrating that our results are more stable and have less distortion. Our method is less preferred in FOV comparison, which is consistent with Table 1, and all other methods with a higher FOV preference have much less stability and distortion preferences.

	More stable	Less distortion	Larger FOV
vs. Grundmann et al. [7]	92.4±4.6%	90.9±5.0%	61.4±8.4%
vs. Wang et al. [30]	96.2±3.3%	94.7±3.9%	68.2±8.1%
vs. PWStableNet [35]	93.2±4.4%	90.9±5.0%	31.8±8.1%
vs. Yu et al. [33]	88.6±5.5%	91.7±4.8%	32.6±8.1%
vs. Choi et al. [4]	91.7±4.8%	89.4±5.3%	25.8±7.6%
Average	92.4±2.0%	91.5±2.1%	43.9±3.8%

Table 2: **Results of user study.** Our results are more stable with less distortion, with the cost of field-of-view.



(a) Analysis on relative pose

(b) Analysis on loss functions

(c) Analysis on LSTM

Figure 8: **Ablation studies on relative poses, losses, and LSTM.** (a) The model using relative poses can output more stable poses and follow the real camera motion well. (b) The protrusion loss L_p reduces the undefined region, and the optical flow loss L_f further improves the smoothness. (c) Without LSTM, our model cannot learn motion patterns well and often generate unstable prediction.

5.3. Ablation Study

Relative poses. As the same motion patterns can be converted to similar relative poses, e.g., panning motion with different speed, it is easier for the model to infer the motion pattern from rotation deviations instead of the absolute poses. Using the relative poses also makes the model training more numerically stable. Fig. 8(a) shows that our method with relative poses can follow the real camera poses well for a PANNING case. In contrast, the model using absolute poses does not follow the real motion well.

w/o L_p w/ L_p

Figure 9: **Effect of protrusion loss.** Without the protrusion loss L_p, the undefined regions are larger on the left and bottom, resulting in a larger cropping in the output video.

Losses. Fig. 8(b) shows the x-axis rotation for a RUNNING case. Our baseline model (the green curve) is trained with the smoothness (L_{C^0} and L_{C^1}) and distortion (L_d) losses. Without these three, our model cannot output a valid quaternion, and the training does not converge. With the protrusion loss L_p (the blue curve), the warped frames contain fewer undefined regions, as shown in Fig. 9. Finally, adopting the optical flow loss L_f (the red curve) further improves the motion smoothness and stability.

LSTM. The LSTM unit carries the temporal information (e.g., motion state) and enables the model to output state-specific results. With the temporal information, the LSTM can also reduce high-frequency noise and generate more stable poses. As shown in Fig. 8(c), when replacing the LSTM with an FC layer, the output poses contain more jitter, resulting in less stable output videos.

6. Limitations and Conclusion

The proposed deep-FVS requires both video frames and the sensor data as inputs. It will show artifact if the camera and gyro sensor are not synchronized (e.g., with a gap larger than 10 ms). Fortunately, most modern smartphones have a well-synchronized sensor and camera system for AR and SLAM features. Our experiments also show a discrepancy between the existing metrics and user preference. Closing this gap with more human perception studies will enable more effective learning-based solutions.

In this work, we have presented deep Fused Video Stabilization, the first DNN-based unsupervised framework that utilizes both sensor data and images to generate high-quality distortion-free results. The proposed network achieves high-quality performance using joint motion representation, relative motion history, novel unsupervised loss functions, and multi-stage training. We have demonstrated that our method outperforms state-of-the-art alternatives in both quantitative comparisons and user study. Our source code and the video dataset that includes sensor logs will be publicly released to facilitate future research.

References

[1] Steven Bell, Alejandro Troccoli, and Kari Pulli. A non-linear filter for gyroscope-based video stabilization. In *ECCV*, 2014. 1

[2] Chris Buehler, Michael Bosse, and Leonard McMillan. Non-metric image-based rendering for video stabilization. In *CVPR*, 2001. 2

[3] Hung-Chang Chang, Shang-Hong Lai, and Kuang-Rong Lu. A robust real-time video stabilization algorithm. *Journal of Visual Communication and Image Representation*, 17(3):659–673, 2006. 2

[4] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM TOG*, 39(1):1–9, 2020. 1, 2, 3, 6, 7, 8

[5] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM TOG*, 31(5):1–10, 2012. 2

[6] Matthias Grundmann, Vivek Kwatra, Daniel Castro, and Irfan Essa. Calibration-free rolling shutter removal. 2012. 2

[7] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR*, 2011. 1, 2, 6, 7, 8

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 4

[11] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. *Stanford University Computer Science Tech Report*, 1:2, 2011. 2, 4

[12] Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, Narendra Ahuja, and Ming-Hsuan Yang. A comparative study for single image blind deblurring. In *CVPR*, 2016. 7

[13] Chia-Kai Liang, Li-Wen Chang, and Homer H Chen. Analysis and compensation of rolling shutter effect. *IEEE TIP*, 17(8):1323–1330, 2008. 2

[14] Chia-Kai Liang and Fuhao Shih. Fused video stabilization on the Pixel 2 and Pixel 2 XL. https://ai.googleblog.com/2017/11/fused-video-stabilization-on-pixel-2.html, 2017. 1, 2

[15] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3D video stabilization. *ACM TOG*, 28(3):44:1–44:9, 2009. 1, 2

[16] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM TOG*, 30(1):4:1–4:10, 2011. 2

[17] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *ECCV*, 2016. 2

[18] Shuaicheng Liu, Yinting Wang, Lu Yuan, Jiajun Bu, Ping Tan, and Jian Sun. Video stabilization with a depth camera. In *CVPR*, 2012. 2

[19] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM TOG*, 32(4):78:1–78:10, 2013. 1, 2, 6

[20] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *CVPR*, 2014. 2

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[22] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE TPAMI*, 28(7):1150–1163, 2006. 2

[23] Carlos Morimoto and Rama Chellappa. Evaluation of image stabilization algorithms. In *ICASSP*, 1998. 2

[24] Hannes Ovrén and Per-Erik Forssén. Gyroscope-based video stabilisation with auto-calibration. In *IEEE International Conference on Robotics and Automation*, 2015. 1

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2

[26] Michael Rubinstein, Diego Gutierrez, Olga Sorkine, and Ariel Shamir. A comparative study of image retargeting. In *ACM TOG*. 2010. 7

[27] Fuhao Shi, Sung-Fang Tsai, Youyou Wang, and Chia-Kai Liang. Steadiface: Real-time face-centric stabilization on mobile phones. In *ICIP*, 2019. 5

[28] Brandon M Smith, Li Zhang, Hailin Jin, and Aseem Agarwala. Light field video stabilization. In *ICCV*, 2009. 2

[29] Damien J Thivent, George E Williams, Jianping Zhou, Richard L Baer, Rolf Toft, and Sebastien X Beysserie. Combined optical and electronic image stabilization. US Patent 9,979,889, 2018. 1

[30] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE TIP*, 28(5):2283–2292, 2018. 1, 2, 6, 7, 8

[31] Sen-Zhe Xu, Jun Hu, Miao Wang, Tai-Jiang Mu, and Shi-Min Hu. Deep video stabilization using adversarial networks. *Comput. Graph. Forum*, 37(7):267–276, 2018. 1, 2

[32] Jiyang Yu and Ravi Ramamoorthi. Robust video stabilization by optimization in cnn weight space. In *CVPR*, 2019. 1, 2, 5

[33] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *CVPR*, 2020. 1, 2, 3, 6, 7, 8

[34] Fang-Lue Zhang, Xian Wu, Hao-Tian Zhang, Jue Wang, and Shi-Min Hu. Robust background identification for dynamic video editing. *ACM TOG*, 35(6):1–12, 2016. 2

[35] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE TIP*, 29:3582–3595, 2020. 1, 2, 6, 7, 8

[36] Binnan Zhuang, Dongwoon Bai, and Jungwon Lee. 5D video stabilization through sensor vision fusion. In *ICIP*, 2019. 2