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 cell infers the new virtual camera pose, which is used to generate a warping grid that stabilizes the video frames. We adopt a relative motion representation as well as a multi-stage training strategy 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 content for video stabilization. We validate the proposed framework through ablation studies and demonstrate that the proposed method outperforms the state-of-art alternative solutions via quantitative evaluations and a user study. Check out our video results, code and dataset at our website.

1. Introduction

Videos captured with a hand-held device are often shaky. 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.

However, high-quality stabilization remains challenging due to complex camera motions and scene variations. The existing video stabilization systems can be generally categorized into image-based and sensor-based methods. Image-based methods output a smooth camera path by extracting camera motions from sparse image features [7, 14, 18] or dense optical flow [4, 29, 31, 32, 34]. These methods offer nonlinear flexibility on motion compensations. However, they often fail when there are complex motions such as parallax, or no reliable features in the scene, and produce visible non-rigid distortions and artifacts due to lack of reliable rigidity constraints. The sensor-based methods use motion sensor data, e.g., gyroscope and accelerometer, to obtain accurate motion. They are free from scene contents and can achieve impressive stabilization results with effective distortion corrections [13, 28]. However, these methods deliver homographies to stabilize the plane at infinity and do not adapt to the scene depth, leading to more residual parallax motions for close scenes.

In this work, we present an efficient deep Fused Video Stabilization (deep-FVS) framework to fuse the two motion sources (image content and motion sensor) and draw benefits from both ends. On one hand, the network outputs a single virtual camera pose instead of dense warping flow, and the videos are then stabilized by warping the sensor-based real camera motions towards this virtual pose. In this way, the motion rigidity is naturally preserved and rolling shutter distortion is corrected without artifacts (e.g., wobbling). On the other hand, the network learns to jointly minimize both camera pose smoothness and optical flow losses. Thus, it automatically adjusts to different scenes (e.g., depth variations) to minimize the residual motions. 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, 18], recent learning-based approaches [4, 29, 30, 32, 34], and the proposed deep-FVS.

As the existing datasets [18, 29] 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. The sensor data and video frames are accurately aligned through calibration. We evaluate the proposed solution objectively and subjectively and show that it outperforms SOTA 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.
- An unsupervised learning process with multi-stage training and relative motion representation.
- A benchmark dataset that contains videos with gyroscope and OIS sensor data and covers various scenarios. Both the dataset and code are publicly released.

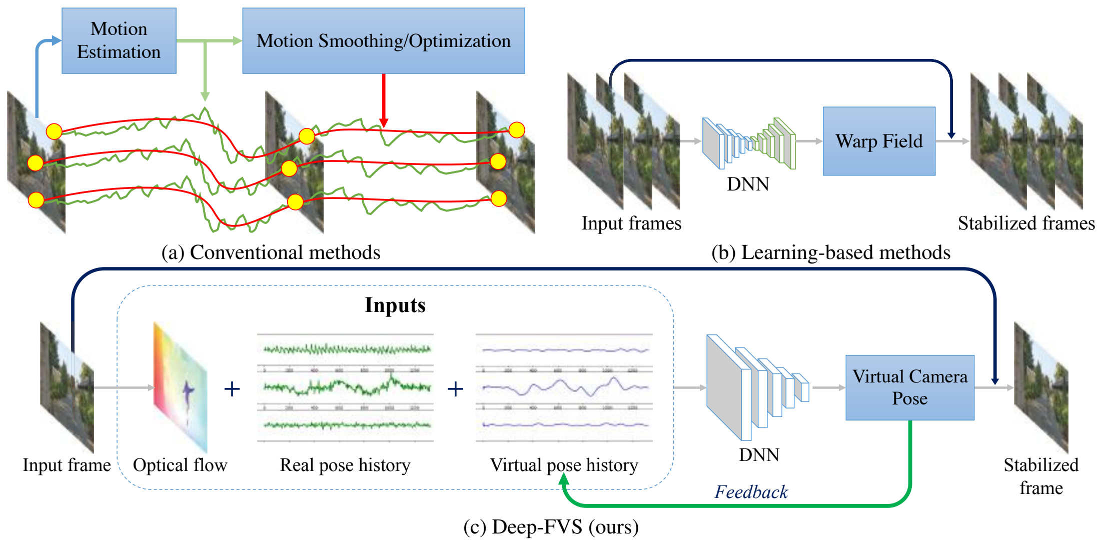


Figure 1: **Comparisons of existing video stabilization methods and the proposed method.** (a) Conventional video stabilization methods [7, 18] estimate camera motions based on image feature trajectories and find a smooth camera path to render a stabilized video. (b) Learning-based approaches [4, 29, 30, 32, 34] learn deep networks to predict warp fields for warping the input video. (c) The proposed Deep-FVS learns to stabilize a video by fusing the optical flow and gyroscope data.

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, 10, 12]. 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 [17], structure from motion [14], and light-field [27]. While 3D approaches can handle parallax and produce high-quality results, they often entail expensive computational costs or require specific hardware devices.

The 2D approaches represent and estimate camera motions as a series of 2D affine or perspective transformations [7, 18, 22]. Robust feature tracking and outlier rejection are applied to obtain reliable estimation [33]. Liu et al. [19] replace feature trajectories with optical flows 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 [18]. Some hybrid 2D-3D approaches exploit the subspace constraints [15] and epipolar geometry [5]. Zhuang et al. [35] 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. [16] 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 [13]. In this work, we utilize the OIS, gyroscope, and optical flow to learn a deep network for stabilization. Our online method has only 10 frames latency and does not require per-video optimization.

Learning-based methods. With the success of deep learning on image recognition [8, 20, 24], 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. [29] 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 [34] learns dense 2D warping fields to stabilize the video. Xu et al. [30] 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 [31] take optical flows as input and optimize the weights of a deep network to generate a warp field for each specific video. They further train a stabilization network that can be generalized test videos without optimization [32]. 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 con-

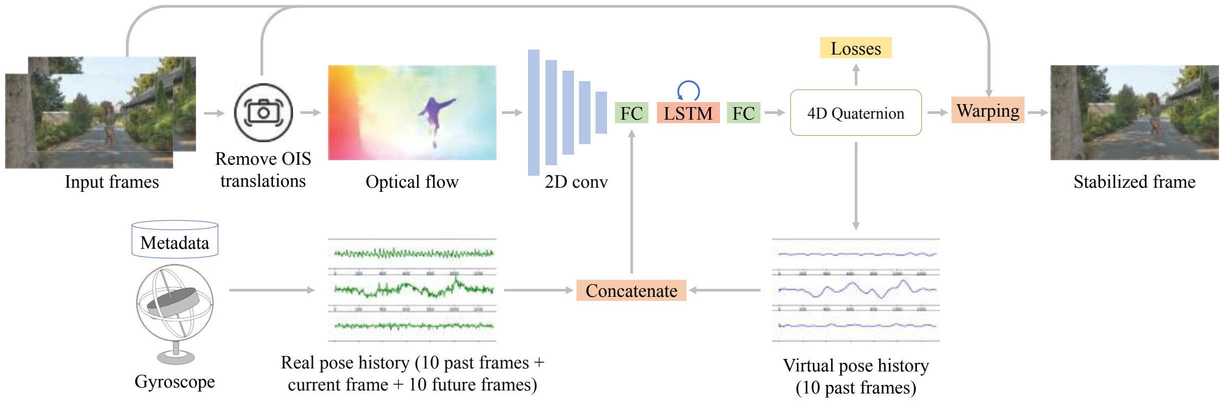


Figure 2: **Overview of deep-FVS.** Given an input video, we first remove the OIS translation to extract the raw optical flow. We also obtain the real camera poses from the gyroscope and convert it to a relative quaternion. An encoder with 2D convolutions embeds optical flows to a latent representation, 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.

trast, we use the gyroscope to compensate camera motions and utilize optical flow to correct the residual motions from scene geometry.

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 use a grid-based warping method similar to Karpenko et al. [10] to warp the input frame to the stabilized rolling shutter corrected domain using the input camera rotations, OIS movement, and the predicted virtual camera poses (supplementary material Section 3). Our solution stabilizes a video frame-by-frame and is suitable for online processing.

During the training stage, we randomly select long subsequences from the training videos, and optimize our DNN with a set of loss functions without any ground-truth videos 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 optical image stabilizer (OIS) translation (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 $t_a \le t_f \le t_b$, 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 camera space to the virtual (stabilized) one is

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

where x_r, x_v are the 2D image coordinates at real and virtual camera spaces, respectively. In all the experiments, we normalize $f = 1511.8$ for both the real and virtual cameras.

4. Unsupervised Learning for Sensor Fusion

This section introduces 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. 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

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 the 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

$$\tilde{F}_n^{n+1} = y_{r,n+1} - x_{r,n}. \tag{4}$$

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

$$\begin{aligned} F_n^{n+1} &= (y_{r,n+1} - O(t_{y_{r,n+1}})) - (x_{r,n} - O(t_{x_{r,n}})) \\ &= \tilde{F}_n^{n+1} - (O(t_{y_{r,n+1}}) - O(t_{x_{r,n}})). \end{aligned} \tag{5}$$

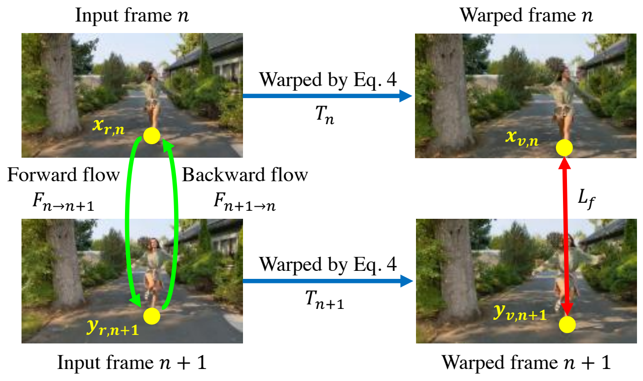


Figure 3: **Optical flow loss.** The optical flow loss aims to minimize the distance between $x_{v,n}$ and $y_{v,n+1}$ in the virtual camera space. By incorporating the forward and backward flows, we define our optical flow loss as in (14).

The backward flow is adjusted similarly. We use the FlowNet2 [9] to extract optical flows in our experiments. Note that with this step, our model can directly handle the case when OIS is not supported.

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

$$H_r = H_{r,\text{absolute}} * R_r^{-1}(t), \tag{6}$$
$$H_v = H_{v,\text{absolute}} * R_r^{-1}(t). \tag{7}$$

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. 5.2).

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

$$L_{C^0} = ||R_v(t) - R_v(t-\Delta t)||^2, \quad (8)$$
$$L_{C^1} = ||R_v(t)R_v^{-1}(t-\Delta t) - R_v(t-\Delta t)R_v^{-1}(t-2\Delta t)||^2. \quad (9)$$

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 [26]:

$$L_p = \sum_{i=0}^{N} w_{p,i} ||\texttt{Min}(\texttt{protrude}(P_v(t), P_r(t+i\Delta t))/\alpha, 1)||^2, \quad (10)$$

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 cropped with a ratio β on each side to the real camera space using (3) and measure the max normalized signed distance between the four warped corners to the frame boundary cropped with a ratio γ. We use β and γ to control the L_p's sensitivity to the camera motion. The protrusion value is clamped to 1 to disregard very large motions and make training stable. We set $\sigma = 2.5$, $N = 10$, $\alpha = 0.2$, $\beta = 0.08$ and $\gamma = 0.04$ in our experiments.

Distortion loss. We measure the warping distortion by:

$$L_d = \Omega(R_v, R_r)/(1 + e^{(-\beta_1(\Omega(R_v,R_r)-\beta_0))}), \quad (11)$$

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 $\beta_0 = 6^\circ$ and $\beta_1 = 100$ in our experiments.

Optical flow loss. We adopt an optical flow loss similar to [31] to minimize the pixel motion between adjacent frames. As shown in Fig. 3, 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 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:

$$x_{v,n} = T_n(x_{r,n}) = T_n(y_{r,n+1} + F_{n+1}^n), \quad (12)$$
$$y_{v,n+1} = T_{n+1}(y_{r,n+1}) = T_{n+1}(x_{r,n} + F_n^{n+1}). \quad (13)$$

Method	Stability	Distortion	Correlation	FOV
YouTube stabilizer	0.834	**0.978**	0.969	0.977
Grundmann et al. [7]	0.818	0.896	0.948	0.635
Wang et al. [29]	0.836	0.850	0.877	0.753
PWStableNet [34]	0.830	0.965	0.973	0.934
Yu et al. [32]	0.842	0.854	0.941	0.793
Choi et al. [4]	0.781	0.875	0.916	**1.0***
Ours (sensor only)	0.846	0.888	**0.976**	0.827
Ours (sensor + flow)	**0.853**	0.937	**0.976**	0.906

Table 1: **Quantitative results.** The best one is marked in **red** and the second best one is marked in blue. *The FOV ratio of Choi et al. [4] is always 1 as they generate full-frame results. For other approaches, the FOV ratio is computed from the scale components of the fitted homography between the input and stabilized frames.

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:

$$L_f = |X_n|^{-1} \sum_{X_n} ||x_{v,n} - T_{n+1}(x_{r,n} + F_n^{n+1})||^2 + |X_{n+1}|^{-1} \sum_{X_{n+1}} ||y_{v,n+1} - T_n(y_{r,n+1} + F_{n+1}^n)||^2, \quad (14)$$

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:

$$L = w_{C^0}L_{C^0} + w_{C^1}L_{C^1} + w_pL_p + w_dL_d + w_fL_f, \quad (15)$$

where $w_{C^0}, w_{C^1}, w_p, w_d$ and w_f are set to $2, 40, 2, 1$ and 1 respectively in our experiments.

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

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 (15) 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.

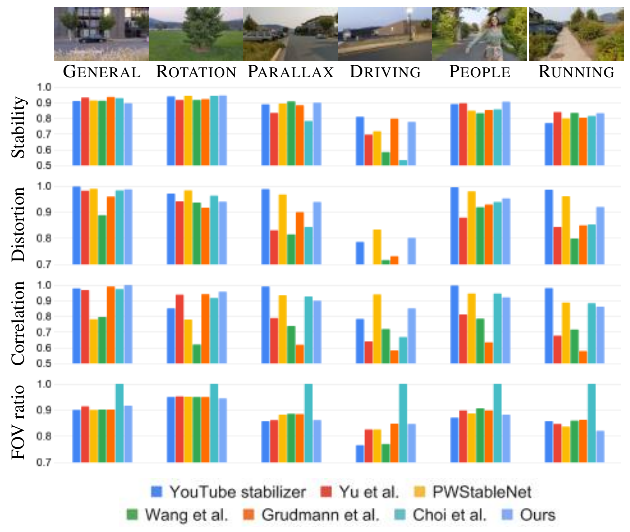


Figure 4: **Per-category quantitative evaluation.** We compare the stability, FOV ratio, distortion, and correlation with state-of-the-art methods [4, 7, 29, 32, 34] on each category.

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. 5.1). We then validate the effectiveness of the key components in the proposed framework by ablation study (Sec. 5.2). We 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 conventional methods [7][1] and YouTube stabilizer (based on [6]), and 4 recent learning-based methods [4, 29, 32, 34][2]. We collect 50 videos with sensor logs using Google Pixel 4 (1920 $\times$ 1080 resolution with variable FPS). The video dataset covers a wide range of variations, such as scenes, illuminations, and motions. The sensor data are accurately calibrated and timestamp-aligned with frames (Google Pixel 4 is ARCore certified [1]). 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. 4 shows a few sample frames from each category. 8-fold cross-validations are used on the training video set for parameter tuning. Due to the lack of training code [4, 32] or requirement on groundtruth data [29, 34], we use the off-the-shelf trained models for comparisons as did in previous works (e.g. [32, 34]). Grundmann et al. [7] does not require training.

Quantitative comparisons. We use three commonly used metrics [4, 18, 29, 32, 34]: *Stability*, *Distortion*, and *FOV ratio*, and define a *Correlation* score, to evaluate the tested methods (please refer to the supplemental materials on their definitions). Note that the distortion measures the global geometry deviation from the input frames, while the correlation evaluates the local deformation.

The results of all test videos are summarized in Table 1, and Fig. 4 plots the average scores for the 6 categories. Overall, our method achieves the best stability and correlation scores. The YouTube stabilizer and Choi et al. [4] generate nearly uncropped or full-frame results, while our FOV ratio is comparable to PWStableNet [34]. We found the quantitative gaps in existing metrics do not fully reflect large visual differences. For example, YouTube stabilizer keeps frames nearly uncropped but outputs relatively unstable videos. PWStableNet [34] produces lots of residual global motions and temporal wobbling, which are not captured by the distortion score. Our method generally obtains better stability and correlation scores on challenging ROTATION, RUNNING, and PEOPLE categories.

Qualitative comparisons. We provide visual comparisons of stabilized frames in Fig. 5. Both Yu et al. [32] and Choi et al. [4] use optical flows to warp the frames and often generate local distortion (03:20 in demo video). Choi et al. [4] produce severe artifacts when the motion is large (04:55 in demo video). Grundmann et al. [7] estimate a global transformation, and Wang et al. [29] 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 (03:56, 06:02 in demo video). In contrast, we fuse both the gyroscope data and optical flow for more accurate motion inference and obtain stable results without distortion or wobbling.

Fig. 6 shows the averaged frame of 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. Our result is the sharpest one, while the averaged frames from other approaches [4, 7, 29, 32, 34] are blurry, demonstrating that our result is more stable than others (03:35 in demo video). We highly encourage readers to watch the full video comparisons in the demo to better evaluate the stabilization quality.

User Study. We conduct a user study to evaluate human preferences on the stabilized videos. As it is easier for a user to make a judgment between two results instead of ranking multiple videos, we adopt the paired comparison [11, 25] to measure the subject preference. In each test, we show two stabilized videos side-by-side and the input video as a reference. We ask the participant the following questions:

1. Which video is more stable?
2. Which video has less distortion?

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

[2]The source code of [4, 29, 34] are publicly available. We obtain the source code of [32] from the authors.

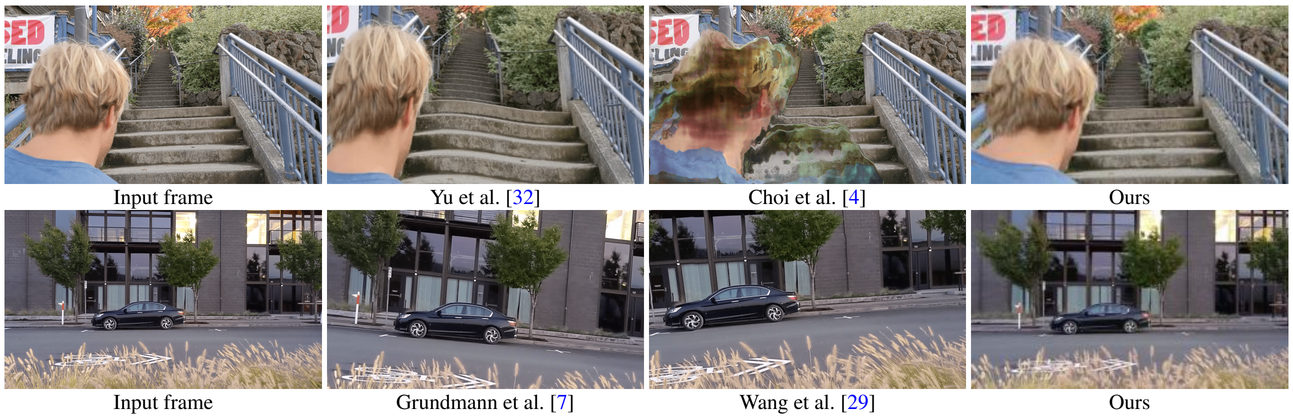


Figure 5: **Visual comparisons.** Non-rigid distortion, local artifacts and temporal wobbling are observed in Yu et al. [32] and Choi et al. [4], and large rotation deviation observed in Grundmann et al. [7] and Wang et al. [29] (which also exhibits local distortions). Our method is free of such issues. Please refer to our supplemental videos on the full results of all the methods.

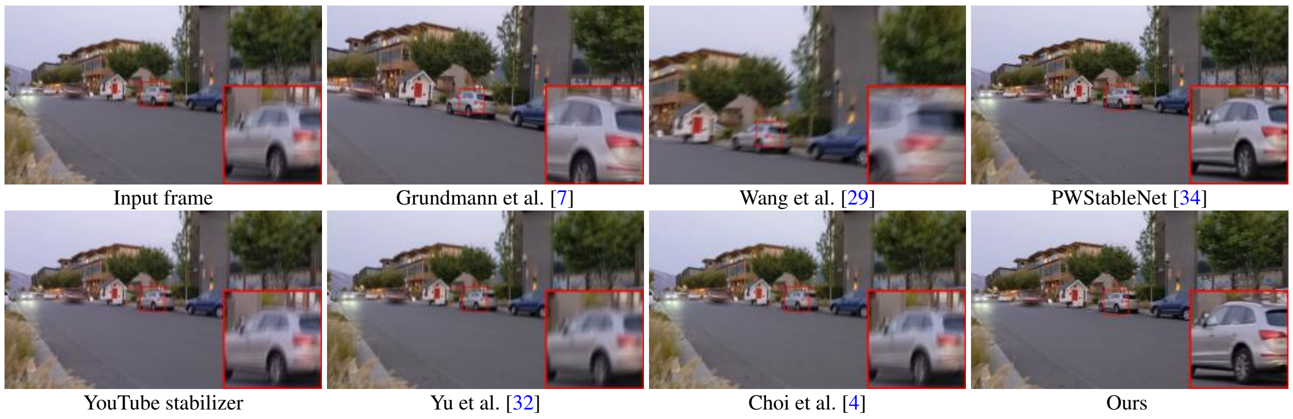


Figure 6: **Stability comparisons.** We take a video with almost no camera motion except handshakes and average 11 adjacent frames. Our average frame is sharper than other methods, indicating that our result is more stable. Please refer to our supplemental videos on the full results of all the methods.

3. Which video has a larger FOV?

In total, we recruit 50 participants online, where each participant evaluates 18 pairs of videos. The videos are shuffled randomly when presenting to each user. All the methods are compared the same number of times.

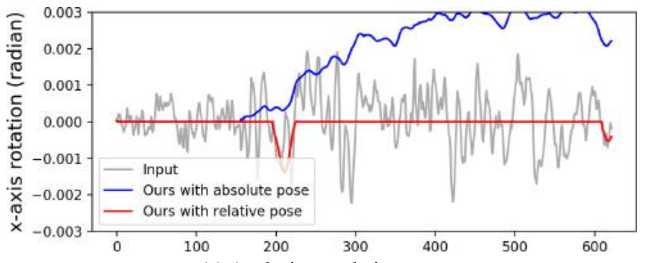
Table 2 shows that our method is preferred in more than 95% of comparisons regarding the stability and 93% regarding the visual quality (less distortion). Our results have larger FOVs than Grundmann et al. [7] and Wang et al. [29] as these two approaches apply excessive cropping to avoid the irregular boundaries. Our FOV is comparable to PWStableNet [34] as users cannot tell the difference ($\simeq$ 50%). Yu et al. [32] generates results with a large FOV in most cases, but applies excessive cropping when the video motion is large (e.g., running). Choi et al. [4] generates full-frame results but at the cost of visible distortion. YouTube stabilizer applies lightweight changes to preserve most of the input content, but the results are less stable. The user study demonstrates that our method generates more stable results with fewer visual artifacts and distortions, while the amount of cropping is similar to other approaches.

	More stable	Less distortion	Larger FOV
vs. Grundmann et al. [7]	98.4±2.3%	95.9±3.5%	82.1±6.9%
vs. Wang et al. [29]	98.4±2.3%	95.1±3.9%	77.2±7.5%
vs. PWStableNet [34]	91.1±5.1%	89.4±5.5%	48.0±9.0%
vs. Yu et al. [32]	92.7±4.7%	93.5±4.4%	38.2±8.7%
vs. Choi et al. [4]	96.7±3.2%	97.6±2.8%	27.6±8.0%
vs. YouTube stabilizer	96.7±3.2%	88.6±5.7%	23.6±7.6%
Average	95.7±1.5%	93.4±1.8%	49.5±3.6%

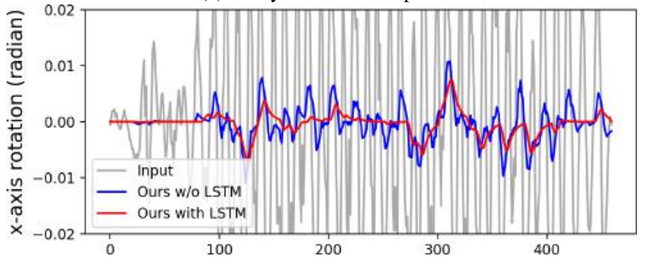
Table 2: **Results of user study.** Our results are more stable with less distortion and overall a comparable field-of-view.

5.2. Ablation Study

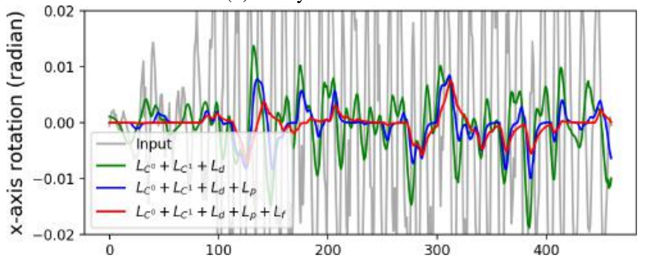
Importance of using relative poses. As the same motion patterns can be converted to similar relative poses, it is easier for the model to infer the motion pattern from rotation deviations instead of the absolute poses. Using the relative



(a) Analysis on relative pose



(b) Analysis on LSTM



(c) Analysis on loss functions

Figure 7: **Ablation studies on relative poses, LSTM and losses.** (a) The model using relative poses can output more stable poses and follow the real camera motion well. (b) Without LSTM, our model cannot learn motion patterns well and often generate unstable prediction. (c) The protrusion loss L_p reduces the undefined region, and the optical flow loss L_f further improves the smoothness.

poses also makes the model training more numerically stable. Fig. 7(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 deviates away from the real motion.

Importance of 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. 7(b), when replacing the LSTM with an FC layer, the output poses contain more jitter, resulting in less stable videos.

Importance of optical flow loss. We compare the sensor-only stabilization results in stage 2 and the full fused results with the optical flow term L_f in stage 3 (Sec. 4.4). The optical flow loss enables the model to adapt to scene content (e.g. parallax), which can improve stability and increase the FOV as shown in the last two rows of Table 1.

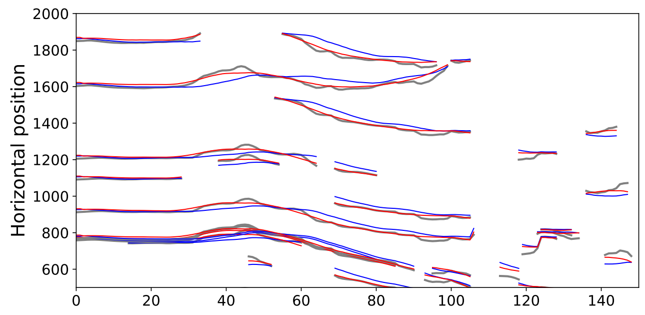


Figure 8: **Feature trajectories before and after stabilization.** Grey: input video. Blue: our method with sensor only (stage 2). Red: our method with sensor and optical flow fusion (stage 3). Our final fused model produces the most stable trajectories which also better follow the real motion.

We also visualize the feature trajectories detected from the KLT tracker [21] in Fig. 8. Our method with sensor and optical flow fusion (red curves) can follow the original camera motion well and maintain the smoothness.

Importance of other losses. Fig. 7(c) shows the x-axis virtual rotation for a RUNNING case. We see a step-by-step stability improvement (smoother motion curves) after introducing each loss.

We also provide the comparisons of execution time in the supplementary material.

6. Limitations and Conclusion

In this work, we present 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, 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. We released the source code and the video dataset to facilitate future research.

Unlike image-based methods, our framework requires additional sensor data. Fortunately, most modern smartphones have a well-synchronized sensor and camera system [1] which makes the requirement minimal. Our model does not support hard FOV preservation. To address this, one can tune the protrusion loss and apply post-processing to pull the virtual camera back toward the real motion when the virtual camera pose deviates from the real pose too much. 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. Also, extending stabilization to further correct relative motions due to parallax is another interesting future work.

References

[1] ARCore supported devices. developers.google.com/ar/discover/supported-devices, 2021. 6, 8

[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, 5, 6, 7

[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. In *ICCP*, 2012. 2, 6

[7] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR*, 2011. 1, 2, 5, 6, 7

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

[9] 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

[10] 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, 3

[11] 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. 6

[12] 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

[13] 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

[14] 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

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

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

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

[18] 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

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

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

[21] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981. 8

[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] 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

[25] Michael Rubinstein, Diego Gutierrez, Olga Sorkine, and Ariel Shamir. A comparative study of image retargeting. In *ACM TOG*. 2010. 6

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

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

[28] 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

[29] 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*, 2018. 1, 2, 5, 6, 7

[30] 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

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

[32] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *CVPR*, 2020. 1, 2, 5, 6, 7

[33] 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

[34] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE TIP*, 2020. 1, 2, 5, 6, 7

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