Minimum Latency Deep Online Video Stabilization

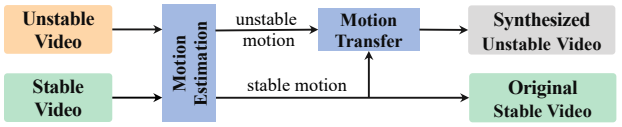
Zhuofan Zhang[1*] Zhen Liu[2*] Ping Tan[3] Bing Zeng[1] Shuaicheng Liu[1,2†]
[1]University of Electronic Science and Technology of China [2]Megvii Technology
[3]The Hong Kong University of Science and Technology
{zhangzf98@std., eezeng@, liushuaicheng@}uestc.edu.cn,
liuzhen03@megvii.com, pingtan@ust.hk

Abstract

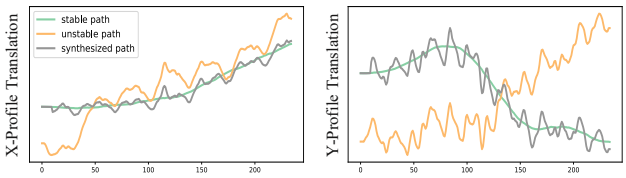
We present a novel camera path optimization framework for the task of online video stabilization. Typically, a stabilization pipeline consists of three steps: motion estimating, path smoothing, and novel view rendering. Most previous methods concentrate on motion estimation, proposing various global or local motion models. In contrast, path optimization receives relatively less attention, especially in the important online setting, where no future frames are available. In this work, we adopt recent off-the-shelf high-quality deep motion models for motion estimation to recover the camera trajectory and focus on the latter two steps. Our network takes a short 2D camera path in a sliding window as input and outputs the stabilizing warp field of the last frame in the window, which warps the coming frame to its stabilized position. A hybrid loss is well-defined to constrain the spatial and temporal consistency. In addition, we build a motion dataset that contains stable and unstable motion pairs for the training. Extensive experiments demonstrate that our approach significantly outperforms state-of-the-art online methods both qualitatively and quantitatively and achieves comparable performance to offline methods. Our code and dataset are available at https://github.com/liuzhen03/NNDVS.

1. Introduction

Video stabilization methods aim at removing unwanted shaky motions of a video caused by unsteady moving capture [6]. Traditional methods often take three main steps: 1) camera motion estimating for trajectory recovery; 2) camera path smoothing; and 3) steady frame synthesis. According to the adopted motion model in the first step, these methods can be broadly classified as 2D or 3D. 2D methods adopt planar motion models such as affine [5], or homography [27], or mesh [25, 22, 33], or flow [26, 20, 41], while 3D methods resort to 3D depth [23, 17], or reconstructed points [18], or epipolar geometry constraints [3].



(a) Illustration of the dataset synthesis process.



(b) Illustration of the synthesized camera path.

Figure 1. Illustration of the dataset synthesis process (a) and the synthesized camera path (b). We transfer an unstable path to a stable path, creating a synthesized path, which shares high frequencies of the unstable path with low frequencies of the stable path.

In comparison, deep learning-based methods learn stabilization models from stable and unstable video pairs [33, 41, 35] without explicit steps of motion estimation and smoothing. However, the results of deep methods are often visually inferior to those of traditional ones. A potential reason is that these methods try to learn the three steps all in a unified framework, each of which has different characteristics but is learned in the network blindly.

On the other hand, deep affine [12], deep homography [38, 8], and deep mesh models [39, 24, 21, 28] have demonstrated high-quality image registration, even under adverse cases. They are robust to scenes with large depth variation, large dynamic objects, and poor textures, fitting perfectly for the camera motion estimation. In this work, we argue that it is not necessary to enclose all three steps of the stabilization into the learning pipeline, but let the motion estimation to the recent deep motion models [39, 24].

*Equal contribution
†Corresponding author

In this way, our network only focuses on learning to stabilize the shaky camera motion, which makes the learning process much more efficient and effective.

Camera path smoothing can be performed offline or online, where the former optimizes the path globally while the latter smooths the path on-the-fly. In other words, offline approaches have access to all past and future frames during optimization, as it often stabilizes a video after it is captured. In contrast, online methods aim to stabilize a video during the capture. Note that, the concept of real-time differs from that of online, as an online method must be in real-time whereas an offline method can run at real-time speed. The main distinction is the availability of future frames.

Most existing methods are offline [25, 5, 4, 34, 26]. However, the online setting is critical as many applications desire instant visual feedback based on live video streams [22, 33]. Normally, with limited or even no access to future frames, online methods cannot achieve equal stability as offline methods, particularly in the suppression of the low-frequency camera shake, which frequently necessitates a long camera path. One may argue that online methods can have all the past frames for processing. However, future motions are important, if not more important than the past. Because, for one thing, fast camera motions can happen at any time suddenly, e.g., quick rotation or zooming. The reaction time is short. Inappropriate processing may either decrease the stability or create artifacts such as excessive cropping. For another, all the past frames have already been shown to the audience, meaning that they are fixed, and thus cannot be modified. We can only adjust the unstabled future frames to the frozen past ones.

To this end, we propose a deep online camera path smooth network that takes a pre-estimated short 2D camera trajectory in a sliding window as input and outputs the motion compensation warp field of the last frame in the window for the stabilized view, which is the setting of online stabilization with the minimum latency. A video can be stabilized on the fly by applying our model repeatedly for every incoming frame. To deal with depth changes and moving objects, we employ the mesh-based motion model [39] which demonstrates excellent robustness in various scenes. Instead of directly inputting the mesh structure or mesh cells as local homographies to the network, we convert the estimated motion to a dense flow field, which can be fed into the convolutional neural network naturally. We show that it is a more convenient representation compared to other alternatives, e.g., vertex sparse motion vectors or homography matrix, which cannot be easily adapted as input to CNNs. Based on that, we further propose a hybrid loss, which consists of the motion-consistency loss, the shape-consistency loss, and the scale-preserving loss, to maintain the spatial coherence and temporal continuity of the stabilized video.

Besides, we create a motion dataset, **MotionStab**, to train our network. In particular, we capture 110 stable videos with a cell phone mounted on a hand-held physical stabilizer. Then, we create a shaky video by transforming each frame of the stable video according to the motion of another irrelevant shaky video. In this way, the stable and unstable motion pairs can be constructed. Fig. 1 shows our idea. Note that, our dataset is different from DeepStab [33], which is captured by a stable and a shaky camera simultaneously. We directly transfer the motion of a shaky video to a stable video. The contents of the two videos are not important but the motion does. That means DeepStab [33] offers frame pairs, while our MotionStab offers motion pairs. We feed the motions to the network instead of RGB frames.

The benefits are three folds: 1) image contents are diverse, while motion mappings are much easier to learn; 2) motion as mesh is lightweight, so does the network; 3) the network can be more focused by excluding the task of the motion estimation. As a result, we show that this motion learning pipeline is effective enough, such that it can work with straightforward network architectures, e.g., UNet, without help from more advanced designs or sophisticated modules. Complex motion types can be learned by the network successfully, as long as they exist in the training data. As such, the network can be kept simple yet effective.

In summary, the main contributions of this paper are concluded as follows:

- We propose a novel deep camera path optimization framework for online video stabilization.
- We propose a hybrid loss to enable robust supervision for maintaining the spatial and temporal coherence of the stabilized video.
- We build a comprehensive MotionStab dataset that covers various motions for camera path optimization.
- We conduct extensive experiments to demonstrate the effectiveness of the proposed approach against existing state-of-the-art online and offline methods.

2. Related Works

2.1. Traditional Methods

Early 2D approaches track image features for several frames and then smooth these feature trajectories to stabilize a video [16, 34]. However, long feature tracks [31] are hard to be obtained when large camera motions exist. Later, motion models, such as affine [5] or homography [27, 2], are calculated between neighboring frames, relaxing the requirement of long feature tracks into feature matches [30] between adjacent frames. Then, these motion models are accumulated to generate a 2D camera trajectory, which is shown to be a good replacement against

track-based approaches, owning largely improved robustness. With respect to the motion representation, homography mixture [4], mesh [25, 22, 34], and optical flow [26, 20] based motion models are proposed to deal with scenes that contain large depth variations. Besides, some approaches focus on special stabilization tasks, such as Selfie [42], 360 [15], and hyperlapse videos [13].

On the other hand, 3D-based methods require 3D camera motions or scene structure for stabilization. The 3D structure can either be calculated from the video by structure-from-motion (SfM) [18] or acquired from additional hardware, such as a depth camera [23], a gyroscope sensor [14], or a light field camera [32]. However, full 3D stabilization is fragile and computationally expensive [18]. Partial 3D methods proposed 3D constraints, such as subspace projection [19] and epipolar geometry [3], to alleviate the requirement of full 3D reconstruction [7]. Normally, 3D methods can better handle the scene parallax compared with 2D approaches due to the physical correctness, as long as the 3D information can be available.

2.2. Deep Learning Methods

Deep methods take the video frames as input and directly output the stabilized frames, which are often trained with stable and unstable frame pairs acquired by special hardware, e.g., DeepStab dataset [33]. Xu *et al.* used the adversarial network to generate a target image to guide the warping [35]. Yu *et al.* learned the flow fields from initial optical flow estimation for the accurate per-pixel motion compensation [41]. Deep methods suffer from the generalization problem, given that DeepStab only contains 60 videos. Yu *et al.* used the CNN as the optimizer instead of learning from data to overfit each input example [40].

Recently, deep motion models have achieved good results for motion estimation, such as deep homography [38, 8] and deep meshflow [39], which is more robust in adverse cases compared to traditional solutions, such as scenes of low texture and low light. In this work, we adopt the deep meshflow [39] for our camera motion estimator and design a network concentrating on camera path smoothing.

2.3. Online Approaches

Most of the video stabilization approaches are offline, where a video is processed after it has been captured. Online methods stabilize a video during the capture [37, 33, 22]. One may argue that global path optimization can be easily modified to online methods by applying a sliding window scheme. However, the stability would inevitably decrease, let alone the existence of challenging camera motions, such as quick rotation and zooming, which may introduce artifacts such as wobble and excessive cropping. Previously, these difficult motions can only be solved satisfactorily by global camera path smoothing. In this work, we propose to solve these problems with our proposed deep online camera path optimization framework.

3. Methodology

3.1. Problem Formulation

Our method is built upon convolutional neural networks. As illustrated in Fig. 3 (a), given an incoming unsteady frame I_t at timestamp t, our deep online video stabilization aims to predict the corresponding steady frame $I_t^{'}$ with no future frames. Here, we use '$'$' to represent predicted quantities. Note that, previous learning-based methods learn the mapping $f(\cdot)$ from the input shaky RGB frames and predict a warp field $B_t^{'}$ of frame I_t, which can be formulated as

$$\{B_t^{'}\} = f\left(\{I_t\}\right), \tag{1}$$

where '$\{\cdot\}$' represents a set of elements. However, our experiments show that directly using complex RGB video frames as input and mixing motion estimation and path smoothing in a network frequently causes estimation errors, resulting in wobbling and distortion artifacts.

In this work, we propose our approach from a novel perspective, where we separate the motion estimation step from the network alone and leave the network focus on camera path smoothing. Specifically, we consider using a fixed window of r past frames $\{I_t\}_r = \langle I_{t-r}, ..., I_{t-1}, I_t\rangle$ of minimum latency (i.e. without using future frames) to stabilize the incoming target frame I_t. The camera motion $\{F_t\}$ is estimated by an off-the-shelf deep motion estimation model. Our camera path smooth network uses only the estimated motion as input to predict the corresponding warp field $B_t^{'}$, i.e.,

$$\{B_t^{'}\} = \Phi(\{F_t\}; \theta), \tag{2}$$

where $\Phi(\cdot)$ denotes the camera path smoothing network and θ is the network parameters to be optimized. We empirically set r to 15 in our experiments.

3.2. Deep Online Camera Path Optimization

As shown in Fig. 2, the overall pipeline of the proposed deep online camera path optimization framework mainly consists of three stages. Firstly, we adopt a deep motion extraction model for robust camera motion estimation. Secondly, the camera path smooth network takes the unsteady camera motion as input and predicts the smoothed warp field. Finally, the original shaky frame is transformed by the predicted warp field to synthesize the target steady frame.

Camera Motion Estimation The first step is to estimate the camera motion. We employ an optimized deep meshflow model [39] to extract the inter-frame motion. Specifically, given the incoming frame I_t and past adjacent frame

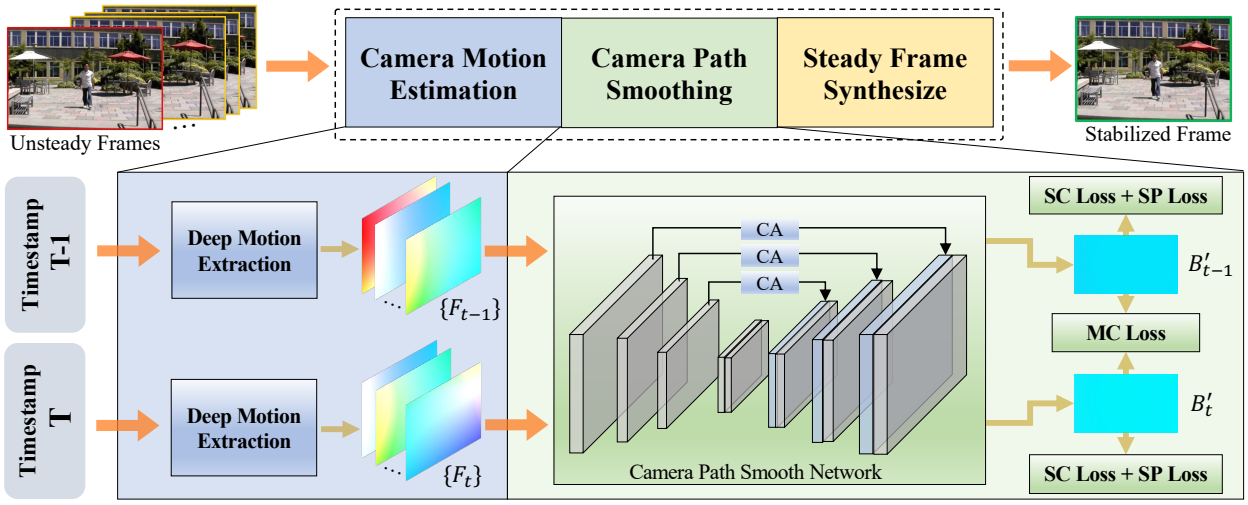


Figure 2. The overall pipeline of the proposed deep online camera path optimization framework. (a) We first employ a deep motion extraction model to estimate the unsteady camera motion. (b) Then, the estimated motion is fed into the camera path smooth network, yielding the smoothed warp field. (c) Finally, the target steady frame is synthesized by the predicted smooth warp field.

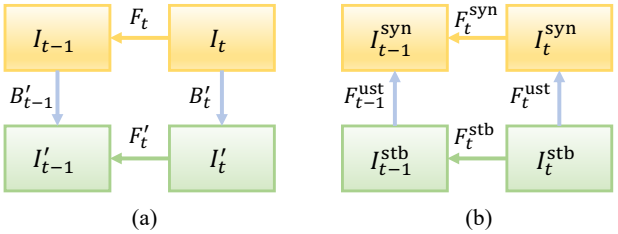


Figure 3. Relationships between the unstable/synthesized video and the stable video. (a) depicts the basic relationships of the unstable video $\{I_t\}$ and stabilized video $\{I'_t\}$, and (b) illustrates the main idea of our dataset synthesis strategy that transfers collected unstable motions $\{F^{\text{ust}}_t\}$ to stable video frames $\{I^{\text{stb}}_t\}$, generating synthesized unstable video frames $\{I^{\text{syn}}_t\}$.

I_{t-1} with size of 640×360, we estimate F_t by model [39] (Fig. 3 (a), upper yellow part). Repeat this process, we accumulate a set of inter-frame motions, $\{F_t\}_{r-1} = \langle F_{t-r+1}, ..., F_{t-1}, F_t \rangle$. In particular, we keep the most recent $r-1$ motions in our buffer, where historical motions that are beyond range $r-1$ are simply dropped. The deep meshflow motion estimation is defined as:

$$F_t = \downarrow_s \mathcal{M}_F(I_t, I_{t-1}), \tag{3}$$

where $\mathcal{M}_F(\cdot)$ is the pre-trained deep meshflow model. '$\downarrow_s$' indicates that we collect the sparse motions at the mesh vertexes, and shrink them pixel by pixel to form a downsampled dense motion field, for the purpose of feeding it to the CNNs. The scale factor s is set to 8, and the size of the dense motion field F_t is 80×45.

Camera Path Smooth Network Given the estimated camera motion $\{F_t\}_{r-1}$ as input, the camera path smooth network predicts the target smoothed warp field B'_t as Eq. 2. Note that, the meshflow model is spatially-variant, and so does the F_t. Therefore, we provide spatial and temporal optimizations. The camera path smooth network is designed as an encoder-decoder architecture and we implement it as a 4-stage UNet with channels of 64, 128, 256, and 512. We further employ the channel attention mechanism (CA) [9] to adaptively learn the weights of different historical motions.

Steady Frame Rendering After obtaining the warp field B'_t, we inverse the process of shrinking, where we put the motions back to the sparse vertex locations to create a mesh, based on which we warp the shaky frame I_t to its stabilized position I'_t:

$$I'_t = \mathcal{W}(I_t, B'_t), \tag{4}$$

where $\mathcal{W}(\cdot)$ denotes the backward warping function. The first parameter is an image and the second is a warp field.

3.3. Training Dataset

Existing training datasets are either captured simultaneously by two cameras [33] or synthesized by simulated injected noise [10, 29, 11]. The former would naturally suffer from parallax problems, while the latter makes the synthesized video paths unrealistic. To address this issue, we offer a novel synthetic method that can build realistic unstable/stable motion pairs from existing shaky videos.

Our main idea is to transfer the motion of an **unstable** shaky video $\boldsymbol{V}_{\text{ust}}$ to a **stable** video $\boldsymbol{V}_{\text{stb}}$, yielding another **synthesized** shaky video $\boldsymbol{V}_{\text{syn}}$ with motions from $\boldsymbol{V}_{\text{ust}}$ but image contents as $\boldsymbol{V}_{\text{stb}}$. The videos $\boldsymbol{V}_{\text{ust}}$ and $\boldsymbol{V}_{\text{stb}}$ are irrelevant. Let's denote $\{I^{\text{ust}}_t\}$, $\{I^{\text{stb}}_t\}$, $\{I^{\text{syn}}_t\}$ as video frames and

$\{F_t^{\text{ust}}\}, \{F_t^{\text{stb}}\}, \{F_t^{\text{syn}}\}$ as motion between frames of video $\boldsymbol{V}_{\text{ust}}, \boldsymbol{V}_{\text{stb}}, \boldsymbol{V}_{\text{syn}}$, accordingly.

To achieve this, we first estimate the motions $\{F_t^{\text{ust}}\}$, $\{F_t^{\text{stb}}\}$ between adjacent frames of $\boldsymbol{V}_{\text{ust}}$ and $\boldsymbol{V}_{\text{stb}}$ by deep meshflow [39]. Next, we warp the frame $\{I_t^{\text{stb}}\}$ by motion $\{F_t^{\text{ust}}\}$ to produce $\{I_t^{\text{syn}}\}$, for each timestamp t:

$$I_t^{\text{syn}} = \mathcal{W}(I_t^{\text{stb}}, F_t^{\text{ust}}). \tag{5}$$

Applying Eq. 5 to every frame of $\boldsymbol{V}_{\text{stb}}$ creates the desired $\boldsymbol{V}_{\text{syn}}$. The $\boldsymbol{V}_{\text{stb}}$ and $\boldsymbol{V}_{\text{syn}}$ become a pair of stable/unstable videos, where the following motion equality holds (Please refer to Fig. 3 (b) for the relation), for each timestamp t:

$$F_t^{\text{ust}} + F_t^{\text{syn}} = F_t^{\text{stb}} + F_{t-1}^{\text{ust}}, \tag{6}$$

where F_t^{ust}, F_t^{stb} have been calculated which are known. Therefore, F_t^{syn} can be calculated as:

$$F_t^{\text{syn}} = F_t^{\text{stb}} + F_{t-1}^{\text{ust}} - F_t^{\text{ust}}. \tag{7}$$

During the training, our network takes a subset of motions $\{F_t^{\text{syn}}\}_{r-1} = \langle F_{t-r+1}^{\text{syn}}, ..., F_{t-1}^{\text{syn}}, F_t^{\text{syn}} \rangle$ as input and output a motion field that is as close as the ground-truth label '$-F_t^{\text{ust}}$', which is the inverse motion of 'F_t^{ust}' as shown in Fig. 3 (b). On the other hand, the generated videos $\boldsymbol{V}_{\text{stb}}$ and $\boldsymbol{V}_{\text{syn}}$ are not important, as all we need are motions that can be calculated and derived by relations.

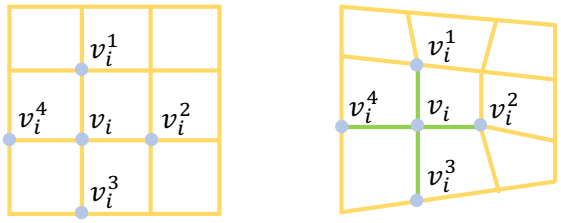
To further establish the training dataset, we collected 110 stable videos of five categories (including regular, rotation, zooming, parallax, and crowd) using a cell phone mounted on a hand-held physical stabilizer. Next, we collected over 150 unstable videos from the website. Subsequently, we performed data synthesis on the collected stable and unstable videos by employing the aforementioned method, constructing the **MotionStab** dataset, which contains paired unstable/stable camera motion of various complex scenarios. As illustrated in Fig. 1 (b), the camera path of the synthesized video pairs is well-registered and realistic as the original unstable one.

3.4. Training Loss

To train the camera path smooth network, we define several losses in our framework: the motion-consistency loss for maintaining motion continuity between consecutive frames, the shape-consistency loss for preserving the geometry of the predicted warp field, and the scale-preserving loss for maintaining the scale of the predicted warp field.

Motion-consistency Loss To keep the continuity of the predicted consecutive video frames, the inter-frame motion of the stabilized video $F_t^{'}$ should be as similar as the ground truth one $\hat{F}_t$, i.e.,

$$\mathcal{L}_{\text{MC}} = \left\| F_t^{'} - \hat{F}_t \right\|_1. \tag{8}$$



(a) original meshgrid (b) transformed meshgrid

Figure 4. Illustration of the shape-consistency loss and the scale-preserving loss. The shape-consistency loss is employed to preserve the geometry shape of a meshgrid and the scale-preserving loss is used to maintain the scale.

According to the relationship of Fig. 3 (a), Eq. 8 can be written as

$$\mathcal{L}_{\text{MC}} = \left\| (F_t + B_{t-1}^{'} - B_t^{'}) - (F_t + \hat{B}_{t-1} - \hat{B}_t) \right\|_1 \tag{9}$$

$$= \left\| B_{t-1}^{'} - B_t^{'} - \hat{B}_{t-1} + \hat{B}_t \right\|_1, \tag{10}$$

where $B_t^{'}$ denotes network output of the predicted warp field. That is to say, the motion continuity can be restrained by the ground truth warp field $\hat{B}_t$ and $\hat{B}_{t-1}$.

Shape-consistency Loss The shape-consistency loss is effective to prevent the distortion problem of the predicted warp field. Following Wang *et al.* [33], we implement the SC loss as an intra-grid loss term and an inter-grid loss term:

$$\mathcal{L}_{\text{SC}} = \mathcal{L}_{\text{intra}} + \mathcal{L}_{\text{inter}}. \tag{11}$$

Note that each pixel of our dense warp field is actually a vertex of the mesh grid. As depicted in Fig. 4, the intra-grid loss term is defined as:

$$\mathcal{L}_{\text{intra}} = \frac{1}{N} \sum_{v_i} \left\| (v_i^1 - v_i) \cdot (v_i^2 - v_i) \right\|_1, \tag{12}$$

where i denotes the i-th vertice and N is the total number of vertices. Besides, the inter-grid loss term is used to maintain the geometric consistency of adjacent grids:

$$\mathcal{L}_{\text{inter}} = \frac{1}{N} \sum_{v_i} \left\| (v_i^1 - v_i) - (v_i - v_i^3) \right\|_1. \tag{13}$$

Scale-preserving Loss Since we convert the sparse motions at the mesh vertexes as a dense motion field and predict the mesh warp field, we introduce a scale-preserving loss to maintain the scale consistency of the predicted warp field:

$$\mathcal{L}_{\text{SP}} = \frac{1}{N} \sum_{v_i} \left\| \frac{\left\| v_i^1 - v_i \right\|_2}{s} - 1 \right\|_1. \tag{14}$$

Table 1. Quantitative comparisons on the NUS dataset [25]. We use the Cropping ratio (C), the Distortion value (D), and the Stability score (S) as evaluation metrics. All of these metrics are in the range of 0 to 1, and the higher the better. The '$*$' denotes the results of Bundled [25] are computed directly from the NUS dataset. The bests are marked in red and the second bests are in blue.

	Methods	Regular			Rotation			Zooming		
		C	D	S	C	D	S	C	D	S
Offline	$*$ Bundled [25]	0.6658	0.9409	0.9048	0.6477	0.8692	0.9194	0.5773	0.9073	0.9236
	Robust L1 [5]	0.7157	0.9252	0.8454	0.7206	0.8231	0.8617	0.7107	0.8329	0.7730
	DIFRINT [1]	0.9854	0.9555	0.8177	0.9413	0.8813	0.8701	0.9528	0.8531	0.8763
	Yu *et al.* [41]	0.9486	0.9752	0.8427	0.7661	0.7616	0.9317	0.8971	0.8630	0.9107
	PWStableNet [43]	0.9319	0.9852	0.8162	0.8801	0.9707	0.9251	0.9303	0.9792	0.8906
	DUT [36]	0.9485	0.9600	0.8717	0.7609	0.6940	0.9203	0.8986	0.8606	0.9221
Online	Meshflow [22]	0.8081	0.9168	0.8386	0.7578	0.7679	0.9092	0.7773	0.8715	0.8839
	StabNet [33]	0.7491	0.8393	0.8406	0.7417	0.7205	0.8384	0.7376	0.7807	0.8697
	Ours	0.9433	0.9917	0.8030	0.9021	0.9847	0.9341	0.9185	0.9736	0.8982

	Methods	Crowd			Parallax			Avg.		
		C	D	S	C	D	S	C	D	S
Offline	$*$ Bundled [25]	0.6685	0.8910	0.8744	0.7158	0.8965	0.8964	0.6550	0.9010	0.9037
	Robust L1 [5]	0.7208	0.9077	0.8504	0.7132	0.8054	0.8778	0.7162	0.8589	0.8417
	DIFRINT [1]	0.9662	0.8713	0.8456	0.9746	0.8823	0.8600	0.9641	0.8887	0.8539
	Yu *et al.* [41]	0.9081	0.9191	0.8660	0.9209	0.9219	0.8896	0.8882	0.8872	0.8881
	PWStableNet [43]	0.9181	0.9790	0.8162	0.9244	0.9802	0.8549	0.9170	0.9789	0.8606
	DUT [36]	0.8986	0.8617	0.9042	0.9254	0.8774	0.9127	0.8860	0.8507	0.9062
Online	Meshflow [22]	0.7888	0.8246	0.8280	0.8121	0.8429	0.8488	0.7882	0.8762	0.8617
	StabNet [33]	0.7247	0.7725	0.7912	0.6967	0.7660	0.8289	0.7280	0.7758	0.8338
	Ours	0.9247	0.9816	0.8408	0.9326	0.9827	0.8762	0.9242	0.9829	0.8705

where s is the scale factor in Eq. 3. Eventually, the overall loss is the combination of the above three loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{MC}} + \alpha\mathcal{L}_{\text{SC}} + \beta\mathcal{L}_{\text{SP}}, \tag{15}$$

where α and β are empirically set to 0.01.

4. Experimental Results

4.1. Implementation Details

Evaluation Metrics For quantitative comparison, we employ three widely used metrics: **cropping ratio**, **distortion value**, and **stability score**, as in prior methods [25, 36, 43, 22]. For cropping ratio and distortion value, we first fit a global homography between the input and output videos at each frame. The cropping ratio measures the remaining frame area after cropping out undefined pixels due to motion compensation and is computed as the average scale component of the entire video. The distortion value, which is defined as the worst ratio of the two largest eigenvalues of the affine component across all frames, measures the anisotropic scaling of the homography between the input and output frames. The stability score measures the smoothness of the stabilized videos and the camera path is utilized to determine the value. We compute this metric by using a frequency domain analysis as in Bundled [25].

Implementation Details Our network is implemented by PyTorch. We use Adam as the optimizer for training, with an initial learning rate of 1×10^{-4} and no weight decay. We set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$, respectively. We train the network for 100,000 iterations. The overall training time on two NVIDIA 1080Ti GPUs is 20 hours. During training, two adjacent frames and their corresponding historical frames are selected as a training sample. The MotionStab dataset comprises 65238 training samples in total. For inference, the overall framework can process a frame of size 640×360 in real-time. Specifically, we spend 12 ms, 10 ms, and 4 ms to extract camera motion, smooth the camera path, and synthesize stable frames, respectively.

4.2. Comparisons with Existing Methods

To evaluate the proposed approach, we compare it with the state-of-the-art methods on the NUS dataset [25]. The compared methods include three traditional methods, Bundled [25], Robust L1 [5], Meshflow [22], and five deep learning based methods, DIFRINT [1], Yu *et al.* [41], PWStableNet [43], DUT [36], and StabNet [33]. It is worth noting that Meshflow [22], StabNet [33], and ours are online, while the rest are offline. The results of the compared methods are obtained from publicly accessible implementations with default parameters or pre-trained models.

Quantitative results Table 1 presents the quantitative results for each of the five categories (including regular, quick rotation, zooming, crowd, and parallax), as well as the average results across all categories. From Table 1, we can see that offline approaches generally outperform online meth-

Figure 5. Qualitative comparisons with the online methods Meshflow [22] and StabNet [33]. We keep the original warped frames for comparison. The red boxes indicate the preserved frame content after cropping. The red arrows highlight the shear and distortion artifacts.

ods, especially in challenging scenarios. In comparison to existing online approaches, our method achieves significant performance improvements. Taking Meshflow [22] as the baseline, the cropping ratio and the distortion value of our method increase by 17.25% and 12.18%, respectively. Notably, our method can achieve approaching results to several compared offline methods.

Qualitative results Fig. 5 illustrates the flaws that can be noticed directly from the video frames, including shear, distortion, and over-cropping. As can be seen, Meshflow suffers from shear and distortion in some scenes due to its reliance on well-optimized parameter tweaking. The shear and over-cropping problems in StabNet [33] are more severe because they directly employ video frames as input for mixed camera motion estimation and path smoothing, making the performance and generalization ability difficult to even approach the traditional approach Meshflow [22]. On the contrary, since our network solely smooths and optimizes the camera path, regardless of image content, it is more generic on diverse challenging scenes and can avoid the aforementioned issues in an online setting.

User Study We also conduct a user study comparing our method to the PWStableNet [43], which is one of the deep offline approaches that achieves overall good performances according to Table 1. Note that we do not compare with the online methods [22, 33], because our qualitative results are significantly better than theirs. Specifically, we prepare a test set containing 15 videos (3 videos per category), and each of the 20 participants is asked to choose the better one from the results of the two compared methods. The videos are arranged randomly and the original unstable videos are provided. The results of the user study are shown in Fig. 7.

4.3. Ablation Studies

To analyze the capability of each component, we conduct extensive ablation experiments on the NUS dataset [25].

Ablation on losses We first conduct experiments on the motion-consistency loss, the shape-consistency loss, and the scale-preserving loss. In the first row of Table 2, we take the vanilla UNet optimized by the MC loss as our baseline. In the next two rows, we add the SC loss and the SP loss to train the model, respectively. Comparing the first three rows

(a) MC (b) MC + SC (c) MC + SC + SP (d) Input frame

Figure 6. Qualitative results of the ablation study on the motion-consistency loss (MC), the shape-consistency loss (SC), and the scale-preserving loss (SP). The red arrows show the shear and distortion regions, and the red boxes show the preserved frame region.

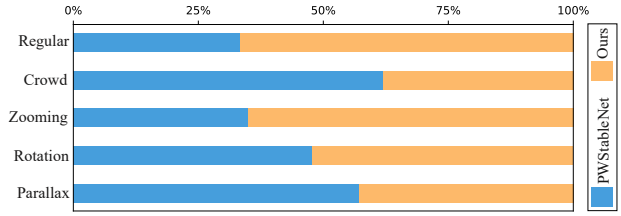


Figure 7. User study results comparing with PWStableNet [43].

Table 2. Quantitative results of the ablation studies on the hybrid loss and the channel attention mechanism (CA).

	MC	SC	SP	CA	C	D	S
Baseline	✓				0.80	0.84	0.87
Variant 1	✓	✓			0.85	0.90	0.86
Variant 2	✓	✓	✓		0.90	0.95	0.86
Ours	✓	✓	✓	✓	0.92	0.98	0.87

reveals that adding the SC loss and the SP loss respectively improves performance. The best performance is gained by combining them together. Fig. 6 shows some qualitative results. As seen, when only using the MC loss, substantial shear, and distortion occur (Fig. 6 (a), highlighted by red arrows.). The distortion artifacts can be effectively resolved by adding the SC loss (Fig. 6 (b)). However, the scale of the stabilized frame is not well preserved, resulting in larger cropped regions. This issue can eventually be solved by the proposed SP loss (Fig. 6 (c), distinguished by red boxes).

Analysis of the quality of the dataset To verify the impact of the dataset, we train the network on the DeepStab [33] and our MotionStab dataset, respectively. Fig. 8 shows the quantitative results on the NUS dataset. As seen, when trained using the proposed MotionStab dataset, both our method and StabNet [33] yield better results than training on the DeepStab dataset, respectively, especially in terms of cropping ratio and distortion value. We conclude that the main reasons are twofold. Firstly, our MotionStab dataset contains richer scenarios and unstable motions, improving the performance and generalization ability of the network.

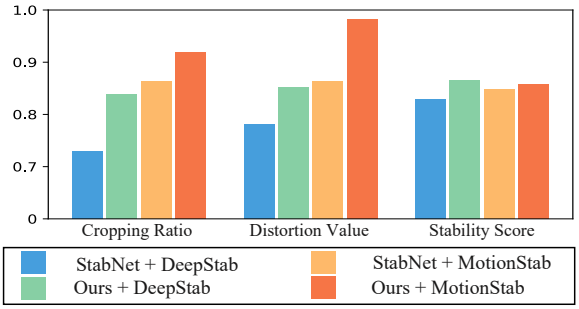


Figure 8. Comparisons of the quantitative results between StabNet [33] and our method when trained with different datasets.

Secondly, the synthesized pairs in MotionStab suffer no parallax issue inherent in DeepStab, leading to fewer distortion artifacts (i.e., higher distortion value) in the stabilized frames. Additionally, our method outperforms StabNet [33] on both datasets, demonstrating its superiority over directly predicting stable warp fields from input video frames.

5. Conclusions

This work presents a deep camera path optimization framework for online video stabilization. We leave the motion estimation to recent off-the-shelf deep motion models and concentrate on path smoothing. Our model takes a 2D camera path in a sliding window as input and outputs a warp field for the last frame in that window. A video can be stabilized online by applying our model repeatedly for every incoming frame. We introduce a hybrid loss to enhance the spatial and temporal consistency of the stabilized video. Moreover, we created a motion dataset to train our model. Results show that our method outperforms previous approaches both qualitatively and quantitatively.

Acknowledgements This work was supported by Sichuan Science and Technology Program of China under grants Nos. 2023NSFSC0462, 2023NSFSC1972, 2022YFQ0079, 2021YFG0001, 2022YFG0050, and National Natural Science Foundation of China under grant No.62031009.

References

[1] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM Trans. on Graphics (TOG)*, 39(1):1–9, 2020. 6

[2] Michael L Gleicher and Feng Liu. Re-cinematography: Improving the camerawork of casual video. *ACM transactions on multimedia computing, communications, and applications (TOMM)*, 5(1):1–28, 2008. 2

[3] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM Trans. on Graphics (TOG)*, 31(5):1–10, 2012. 1, 3

[4] Matthias Grundmann, Vivek Kwatra, Daniel Castro, and Irfan Essa. Calibration-free rolling shutter removal. In *Proc. ICCP*, pages 1–8, 2012. 2, 3

[5] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *Proc. CVPR*, pages 225–232, 2011. 1, 2, 6

[6] Wilko Guilluy, Laurent Oudre, and Azeddine Beghdadi. Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication*, 90:116015, 2021. 1

[7] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[8] Mingbo Hong, Yuhang Lu, Nianjin Ye, Chunyu Lin, Qijun Zhao, and Shuaicheng Liu. Unsupervised homography estimation with coplanarity-aware gan. In *Proc. CVPR*, pages 17663–17672, 2022. 1, 3

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, pages 7132–7141, 2018. 4

[10] Chia-Hung Huang, Hang Yin, Yu-Wing Tai, and Chi-Keung Tang. Stablenet: semi-online, multi-scale deep video stabilization. *arXiv preprint arXiv:1907.10283*, 2019. 4

[11] Maria Silvia Ito and Ebroul Izquierdo. A dataset and evaluation framework for deep learning based video stabilization systems. In *Proc. VCIP*, pages 1–4. IEEE, 2019. 4

[12] Dengchao Jin, Jianjun Lei, Bo Peng, Wanqing Li, Nam Ling, and Qingming Huang. Deep affine motion compensation network for inter prediction in vvc. *IEEE Trans. on Circuits and Systems for Video Technology*, 32(6):3923–3933, 2021. 1

[13] Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015. 3

[14] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. *CSTR*, 1(2):13, 2011. 3

[15] Johannes Kopf. 360 video stabilization. *ACM Transactions on Graphics (TOG)*, 35(6):1–9, 2016. 3

[16] Ken-Yi Lee, Yung-Yu Chuang, Bing-Yu Chen, and Ming Ouhyoung. Video stabilization using robust feature trajectories. In *Proc. CVPR*, pages 1397–1404, 2009. 2

[17] Yao-Chih Lee, Kuan-Wei Tseng, Yu-Ta Chen, Chien-Cheng Chen, Chu-Song Chen, and Yi-Ping Hung. 3d video stabilization with depth estimation by cnn-based optimization. In *Proc. CVPR*, pages 10621–10630, 2021. 1

[18] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. on Graphics (ToG)*, 28(3):1–9, 2009. 1, 3

[19] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. on Graphics (TOG)*, 30(1):1–10, 2011. 3

[20] Shuaicheng Liu, Mingyu Li, Shuyuan Zhu, and Bing Zeng. Codingflow: Enable video coding for video stabilization. *IEEE Trans. on Image Processing*, 26(7):3291–3302, 2017. 1, 3

[21] Shuaicheng Liu, Yuhang Lu, Hai Jiang, Nianjin Ye, Chuan Wang, and Bing Zeng. Unsupervised global and local homography estimation with motion basis learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2022. 1

[22] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *Proc. ECCV*, pages 800–815, 2016. 1, 2, 3, 6, 7

[23] Shuaicheng Liu, Yinting Wang, Lu Yuan, Jiajun Bu, Ping Tan, and Jian Sun. Video stabilization with a depth camera. In *Proc. CVPR*, pages 89–95, 2012. 1, 3

[24] Shuaicheng Liu, Nianjin Ye, Chuan Wang, Kunming Luo, Jue Wang, and Jian Sun. Content-aware unsupervised deep homography estimation and beyond. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2022. 1

[25] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Trans. on Graphics (TOG)*, 32(4):1–10, 2013. 1, 2, 3, 6, 7

[26] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *Proc. CVPR*, pages 4209–4216, 2014. 1, 2, 3

[27] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(7):1150–1163, 2006. 1, 2

[28] Lang Nie, Chunyu Lin, Kang Liao, Shuaicheng Liu, and Yao Zhao. Depth-aware multi-grid deep homography estimation with contextual correlation. *IEEE Trans. on Circuits and Systems for Video Technology*, 32(7):4460–4472, 2022. 1

[29] Hui Qu, Li Song, and Gengjian Xue. Shaking video synthesis for video stabilization performance assessment. In *Proc. VCIP*, pages 1–6. IEEE, 2013. 4

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proc. ICCV*, pages 2564–2571, 2011. 2

[31] Jianbo Shi et al. Good features to track. In *Proc. CVPR*, pages 593–600, 1994. 2

[32] Brandon M Smith, Li Zhang, Hailin Jin, and Aseem Agarwala. Light field video stabilization. In *Proc. ICCV*, pages 341–348, 2009. 3

[33] 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 Trans. on Image Processing*, 28(5):2283–2292, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[34] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee. Spatially and temporally optimized video stabiliza-

tion. *IEEE Trans. on Visualization and Computer Graphics*, 19(8):1354–1361, 2013. 2, 3

[35] Sen-Zhe Xu, Jun Hu, Miao Wang, Tai-Jiang Mu, and Shi-Min Hu. Deep video stabilization using adversarial networks. In *Computer Graphics Forum*, volume 37, pages 267–276, 2018. 1, 3

[36] Yufei Xu, Jing Zhang, Stephen J Maybank, and Dacheng Tao. Dut: Learning video stabilization by simply watching unstable videos. *IEEE Trans. on Image Processing*, 31:4306–4320, 2022. 6

[37] Junlan Yang, Dan Schonfeld, Chong Chen, and Magdi Mohamed. Online video stabilization based on particle filters. In *Proc. ICIP*, pages 1545–1548, 2006. 3

[38] Nianjin Ye, Chuan Wang, Haoqiang Fan, and Shuaicheng Liu. Motion basis learning for unsupervised deep homography estimation with subspace projection. In *Proc. ICCV*, pages 13117–13125, October 2021. 1, 3

[39] Nianjin Ye, Chuan Wang, Shuaicheng Liu, Lanpeng Jia, Jue Wang, and Yongqing Cui. Deepmeshflow: Content adaptive mesh deformation for robust image registration. *arXiv preprint arXiv:1912.05131*, 2019. 1, 2, 3, 4, 5

[40] Jiyang Yu and Ravi Ramamoorthi. Robust video stabilization by optimization in cnn weight space. In *Proc. CVPR*, pages 3800–3808, 2019. 3

[41] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *Proc. CVPR*, pages 8159–8167, 2020. 1, 3, 6

[42] Jiyang Yu, Ravi Ramamoorthi, Keli Cheng, Michel Sarkis, and Ning Bi. Real-time selfie video stabilization. In *Proc. CVPR*, pages 12036–12044, 2021. 3

[43] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE Trans. on Image Processing*, 29:3582–3595, 2020. 6, 7, 8