

# StabStitch++: Unsupervised Online Video Stitching with Spatiotemporal Bidirectional Warps

Lang Nie, Chunyu Lin, Kang Liao, Yun Zhang, Shuaicheng Liu, *Senior Member, IEEE*,  
Yao Zhao, *Fellow, IEEE*

**Abstract**— We retarget video stitching to an emerging issue, named *warping shake*, which unveils the temporal content shakes induced by sequentially unsmooth warps when extending image stitching to video stitching. Even if the input videos are stable, the stitched video can inevitably cause undesired warping shakes and affect the visual experience. To address this issue, we propose *StabStitch++*, a novel video stitching framework to realize spatial stitching and temporal stabilization with unsupervised learning simultaneously. First, different from existing learning-based image stitching solutions that typically warp one image to align with another, we suppose a virtual midplane between original image planes and project them onto it. Concretely, we design a differentiable bidirectional decomposition module to disentangle the homography transformation and incorporate it into our spatial warp, evenly spreading alignment burdens and projective distortions across two views. Then, inspired by camera paths in video stabilization, we derive the mathematical expression of stitching trajectories in video stitching by elaborately integrating spatial and temporal warps. Finally, a warp smoothing model is presented to produce stable stitched videos with a hybrid loss to simultaneously encourage content alignment, trajectory smoothness, and online collaboration. Compared with *StabStitch* that sacrifices alignment for stabilization, *StabStitch++* makes no compromise and optimizes both of them simultaneously, especially in the online mode. To establish an evaluation benchmark and train the learning framework, we build a video stitching dataset with a rich diversity in camera motions and scenes. Experiments exhibit that *StabStitch++* surpasses current solutions in stitching performance, robustness, and efficiency, offering compelling advancements in this field by building a real-time online video stitching system. The code and dataset are available at <https://github.com/nie-lang/StabStitch2>.

**Index Terms**—Image/video stitching, Video stabilization.

## I. INTRODUCTION

Lang Nie, Chunyu Lin, and Yao Zhao are with the Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China, and also with Visual Intelligence +X International Cooperation Joint Laboratory of MOE, Beijing 100044, China (e-mail: nielang@bjtu.edu.cn, cylin@bjtu.edu.cn, yzhao@bjtu.edu.cn).

Kang Liao is with the School of Computing and Data Science, Nanyang Technological University, Singapore (e-mail: kang.liao@ntu.edu.sg).

Yun Zhang is with the School of Media Engineering, Communication University of Zhejiang, Hangzhou 310018, China (e-mail: zhangyun@cuz.edu.cn).

Shuaicheng Liu is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: liushuaicheng@uestc.edu.cn).

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants U2441242 and 62172032, as well as by the Open Fund of Zhejiang Key Laboratory of Film and TV Media Technology. (Corresponding author: Chunyu Lin.)

VIDEO stitching techniques are commonly employed to create panoramic or wide field-of-view (FoV) displays from different viewpoints with limited FoV. Due to their practicality, they are widely applied in autonomous driving [1], video surveillance [2], virtual reality [3], etc. Our work lies in the most common and challenging case of video stitching with hand-held cameras. It does not require camera poses, motion trajectories, or temporal synchronization. It merges multiple videos, whether from multiple cameras or a single camera, capturing multiple videos to create a more immersive representation of the captured scene. Moreover, it transforms video production into an enjoyable and collaborative endeavor among a group of individuals.

Compared with video stitching, image stitching has been studied more extensively and profoundly, which inevitably throws the question of whether existing image stitching solutions can be directly extended to video stitching. Pursuing this thought, we initially leverage existing image stitching algorithms [4] [5] to process hand-held camera videos. Subsequently, we observe that although the stitched results for individual frames are remarkably natural, there is apparent content jitter among temporally consecutive frames. It is also important to note that the jitter effect does not originate from the inherent characteristics of the source video itself, although these videos are captured by hand-held cameras. In fact, due to the advancements and widespread adoption of video stabilization in both hardware and software nowadays, the source videos obtained from hand-held cameras are typically stable unless deliberately subjected to shaking. For clarity, we define such content jitter as *warping shake*, which describes the temporal content instability induced by temporally unsmooth warps, irrespective of the stability of source videos. Fig. 1 (left and middle) illustrates the occurrence process of warping shakes.

Existing video stitching solutions [6] [7] [8] [9] [10] follow a strong assumption that each source video from freely moving hand-held cameras suffers from heavy and independent shakes. Consequently, every source video necessitates stabilization via warping, contradicting the current prevalent reality that video stabilization technology has already been widely integrated into various portable devices (*e.g.*, cellphones, DV cameras, and UAVs). In addition, these approaches, to jointly optimize video stabilization and stitching, often establish a sophisticated non-linear solving system consisting of various energy terms. To find the optimal parameters, an iterative solving strategy is typically employed. Each iteration involves several steps dedicated to optimizing different parameters separately, resulting

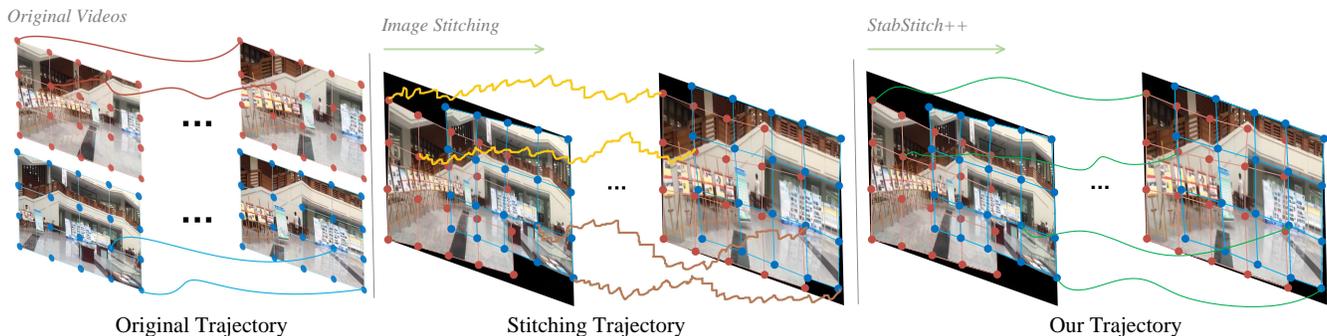


Fig. 1. The occurrence and elimination of warping shakes. Left: stable camera trajectories for input videos. Middle: warping shakes are produced by image stitching, yielding unsmooth stitching trajectories. Right: the proposed *StabStitch++* eliminates these shakes successfully.

in a rather slow inference speed. The complicated optimization procedures also impose stringent requirements on the quality of input videos (*e.g.*, sufficient, accurate, and evenly distributed matching points), making video stitching systems fragile and not robust in practical applications.

To solve the above issues, we present a novel unsupervised online video stitching framework (termed *StabStitch++*) to simultaneously realize spatial stitching and temporal stabilization. First, existing learning-based image stitching solutions typically warp one image to align with another, concentrating all the alignment challenges and projective distortions into a single view. In contrast, our spatial warp introduces a differentiable bidirectional decomposition module to evenly distribute the burdens across both views, thereby boosting alignment and reducing distortions. It determines a virtual midplane by disentangling the global homography transformation and then carries out local spatial stitching by projecting the original image planes onto it. Then, building upon the current condition that source videos are typically stable, we simplify this task to stabilize the warped videos by removing warping shakes as illustrated in Fig. 1 (middle and right). To get the shaken trajectories, we derive the mathematical formulation of stitching trajectories in the warped video from the experience of camera paths in video stabilization by ingeniously combining spatial and temporal warps. Finally, to get stable stitched videos, we propose a warp smoothing model to simultaneously encourage content alignment, trajectory smoothness, and online collaboration within a hybrid loss. It is worth mentioning that this joint optimization aims to find the optimal solution that satisfies all of these conditions simultaneously, rather than sacrificing one condition to enhance the others.

Diverging from conventional offline video stitching approaches that require complete videos as input, *StabStitch++* stitches and stabilizes the current online frame to composite a high-quality stitched video only with historical frames. Besides, its efficient designs further contribute to a real-time online video stitching system with minimum latency.

As there is no proper dataset readily available, we build a holistic video stitching dataset to train the proposed framework. Moreover, it could serve as a comprehensive benchmark with rich camera motions and scene diversity to evaluate image/video stitching methods. In summary, the main con-

tributions of this paper are concluded as follows:

- We retarget video stitching to an emerging issue, termed *warping shake*, and reveal its occurrence when extending image stitching to video stitching.
- We present *StabStitch*, the first online video stitching framework, with a pioneering step to integrate video stitching and stabilization with unsupervised learning.
- We propose a holistic video stitching benchmark dataset with diverse scenes and camera motions, which we hope can promote other related research work.
- Compared with state-of-the-art image/video stitching solutions, our method achieves superior performance in terms of scene robustness, inference speed, and stitching/stabilization effect.

In comparison to our previous conference version [11], we make the following new contributions substantially:

- We propose a differentiable bidirectional decomposition module to carry out bidirectional warping on a virtual middle plane, evenly spreading warping burdens across both views. It benefits both image and video stitching, demonstrating universality and scalability.
- A new warp smoothing model is presented to simultaneously encourage content alignment, trajectory smoothness, and online collaboration using a hybrid loss. Different from our conference version that sacrifices alignment for stabilization, the new model searches for a joint optimum in online mode.
- With the above new contributions, we extend *StabStitch* to *StabStitch++* with better alignment, fewer distortions, and higher stability. It can deal with not only common stable videos but also unstable videos.

## II. RELATED WORK

Here, we briefly review image stitching, video stabilization, and video stitching techniques, respectively.

### A. Image Stitching

Traditional image stitching methods usually detect keypoints [12] or line segments [13] and then minimize the projective errors to estimate a parameterized warp by aligning these geometric features. To eliminate the parallax misalignment

[14], the warp model is extended from global homography transformation [15] to other elastic representations, such as mesh [16], TPS [17], superpixel [18], and triangular facet [19]. Meanwhile, to keep the natural structure of non-overlapping regions, a series of shape-preserving constraints is formulated with the alignment objective. For instance, SPHP [20] and ANAP [21] linearized the homography and slowly changed it to the global similarity to reduce projective distortions; DFW [22], SPW [23], and LPC [4] leveraged line-related consistency to preserve geometric structures; GSP [24] and GES-GSP [25] added a global similarity before stitching multiple images together so that the warp of each image resembles a similar transformation as a whole; etc. Besides, Zhang *et al.* [26] re-formulated image stitching task with regular boundaries by simultaneously optimizing alignment and rectangling [27] [28].

Recently, learning-based image stitching solutions emerged. They feed the entire images into the neural network, encouraging the network to directly predict the corresponding parameterized warp model (*e.g.*, homography [29] [30] [31], multi-homography [32], TPS [5] [33] [34], and optical flow [35] [36]). Compared with traditional methods based on sparse geometric features, these learning-based solutions train the network parameters to adaptively capture semantic features by establishing dense pixel-wise optimization objectives. They show better robustness in various cases, especially in challenging cases where hand-craft features are few to detect.

### B. Video Stabilization

Traditional video stabilization can be categorized into 3D [37] [38], 2.5D [39] [40], and 2D [41] [42] [43] methods, according to different motion models. The 3D solutions model the camera motions in 3D space or require extra scene structure for stabilization. The structure is either calculated by structure-from-motion (SfM) [37] or acquired from additional hardware, such as a depth camera [38], a gyroscope sensor [44], or a lightfield camera [45]. Given the intensive computational demands of these 3D solutions, 2.5D approaches relax the full 3D requirement to partial 3D information. To this end, additional 3D constraints are established, such as subspace projection [39] and epipolar geometry [40]. Compared with them, the 2D methods are more efficient with a series of 2D linear transformations (*e.g.*, affine, homography) as camera motions. To deal with large-parallax scenes, spatially varying motion representations are proposed, such as homography mixture [46], mesh [47], vertex profile [48], optical flow [49], etc. Moreover, some special approaches focus on special scenes and specific input (*e.g.*, selfie [50] [51], 360 [52] [53], and hyperlapse [54] videos).

In contrast, learning-based video stabilization methods directly regress unstable-to-stable transformation from data. Most of them are trained with stable and unstable video pairs acquired by special hardware in a supervised manner [55] [56] [57]. To relieve data dependence, DIFRINT [58] proposed the first unsupervised solution via neighboring frame interpolation. To get a stable interpolated frame, only stable videos are used to train the network. Different from it, DUT [59] established unsupervised constraints for motion estimation and trajectory

smoothing, learning video stabilization by simply watching unstable videos.

### C. Video Stitching

Video stitching has received much less attention than image stitching. Early works [1] [60] stitched multiple videos frame-by-frame and focused on the temporal consistency of stitched frames. But the input videos were captured by cameras fixed on rigs. For hand-held cameras with free and independent motions, there is a significant increase in temporal shakes. To deal with it, videos were first stitched and then stabilized in [61], while [9] did it in an opposite way (*e.g.*, videos were firstly stabilized, and then stitched). Both of them accomplished stitching or stabilization in a separate step. Later, a joint optimization strategy was commonly adopted in [7] [8] [6], where [6] further considered the dynamic foreground by background identification. However, solving such a joint optimization problem regarding stitching and stabilization is fragile and computationally expensive. To this end, we rethink the video stitching problem from the perspective of warping shake and propose the first (to our knowledge) unsupervised online solution for hand-held cameras.

## III. METHODOLOGY

The framework of *StabStitch++* is illustrated in Fig. 2, where we take consecutive video frames (*i.e.*, the reference frames and target frames) as input and output the stable stitched frames. It consists of three trainable warp models: spatial warp, temporal warp, and warp smoothing models. We first introduce our spatial and temporal warp models, of which the spatial warp model includes a bidirectional decomposition module. Then, we derive the expression of stitching trajectories for video stitching by integrating spatial and temporal warps, which implicitly represent the warping shakes. Thereafter, a warp smoothing model is proposed to produce stable stitched video with smooth stitching paths.

Here, we briefly review the preliminaries of UDIS++ [5] to distinguish the unidirectional warps from ours. UDIS++ employs a two-stage warping process: the first stage estimates a global homography transformation [62], while the second stage refines it using local Thin-Plate Spline (TPS) transformation [63]. TPS is a principal warp [64] that describes the deformations specified by finitely many point-correspondences in an irregular spacing between a flat image and a warped one. UDIS++ reparameterizes the two different transformations in a uniform representation for joint learning. Concretely, the homography is parameterized as the motions of four vertices, while the TPS is represented as the motions of a uniform grid of  $(U + 1) \times (V + 1)$  control points predefined across the image. A key limitation of UDIS++ is its reliance on unidirectional warping, where only the target image is warped toward the reference, concentrating distortions in a single view. In our work, we address this limitation through bidirectional decomposition, as detailed in Sec. III-A2.

### A. Spatial Warp

1) *Network*: Given a reference and target image ( $I_{ref}^t$  and  $I_{tgt}^t$ ), the spatial warp model aims to estimate the spatial

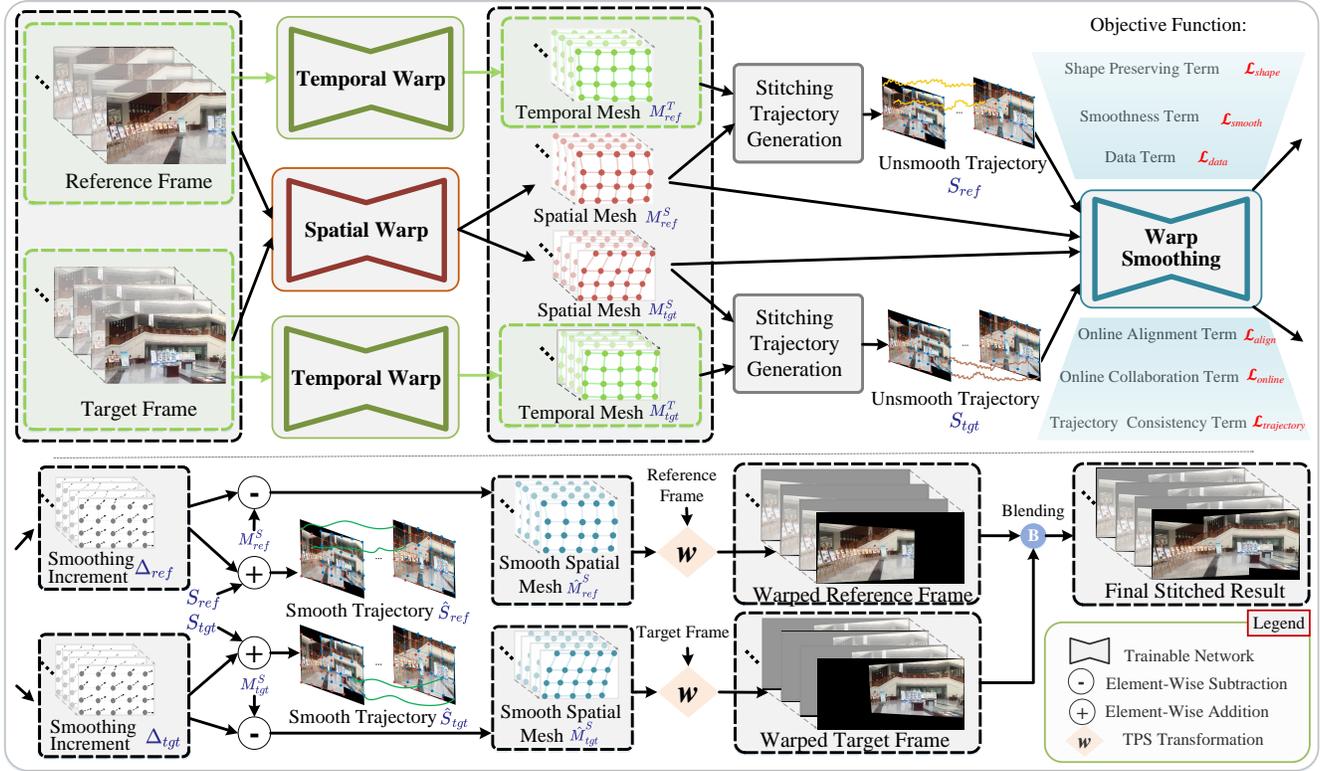


Fig. 2. The overview of *StabStitch++*. We first get the spatial and temporal meshes through spatial and temporal warp models. Then stitching trajectories can be derived by integrating spatial and temporal warps. Finally, a warp smoothing model is leveraged to produce stable stitched frames.

deformation that can naturally align the two images. The network is shown in Fig. 3 (a). Similar to UDIS++ [5], it has a global-to-local structure to combine homography and TPS in a common network. Particularly, the images are first input into a backbone network [65] with shared weights to capture high-level semantic features. Then, the global part calculates the global correlation [66] with the feature maps of the last layer and regresses the global homography. Next, we design a bidirectional decomposition module to disentangle the estimated homography  $H(t)$  into  $H_{ref}(t)$  and  $H_{tgt}(t)$ . It supposes there is a virtual middle and projects the feature maps of the last but one layer onto it. Finally, we adopt the local correlation layer (*i.e.*, cost volume [67]) to capture short-range matching information and regress residual control point motions. The disentangled homography can be transformed into the representations of control point motions and combined with the local residual motions to get final spatial warps (denoted as the sum of control point motions  $m_{ref}^S(t)$ ,  $m_{tgt}^S(t)$ ).

2) *Bidirectional Decomposition*: We intend to find a virtual middle view between the reference and target views. However, it is hard to obtain the camera extrinsics (*i.e.*, translation and rotation) [68] due to limited overlapping regions and the lack of intrinsic. To this end, we simplify the image view as a plane and leverage homography to represent the transformation from one plane to another. This planar transformation can be characterized by a  $3 \times 3$  matrix with eight degrees of freedom: two each for translation, rotation, (an)isotropic scale, and perspectivity. To obtain the middle plane, an intuitive approach is to ensure the magnitude of each attribute (*e.g.*, translation, scale, etc.) is halved from the original transformation. But

these components are intricately coupled within a  $3 \times 3$  matrix (*e.g.*, scaling and rotation are intertwined) and the change of their magnitude is non-linear. To this end, we propose to use the 4-pt representation of deep homography [69] as an alternative, as outlined below:

$$H(t) = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \sim \begin{pmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \\ \Delta u_3 & \Delta v_3 \\ \Delta u_4 & \Delta v_4 \end{pmatrix}, \quad (1)$$

where  $(\Delta u, \Delta v)$  denotes the motions of four vertices on the image. Using the four vertices and their motions, we can get four pairs of matched points, uniquely determining a homography transformation.

In the 4-pt representation, all elements can be uniformly interpreted as spatial displacements. Different from the matrix representation, each element is independent and its magnitude change is linear. Based on the above properties, we present to determine the virtual middle plane by halving all the displacements from the original 4-pt representation:

$$H_{tgt}(t) \sim \begin{pmatrix} \Delta u_1/2 & \Delta v_1/2 \\ \Delta u_2/2 & \Delta v_2/2 \\ \Delta u_3/2 & \Delta v_3/2 \\ \Delta u_4/2 & \Delta v_4/2 \end{pmatrix}. \quad (2)$$

Then, the homography mapped from the reference plane to the middle one is formulated as:

$$H_{ref}(t) = H^{-1}(t)H_{tgt}(t). \quad (3)$$

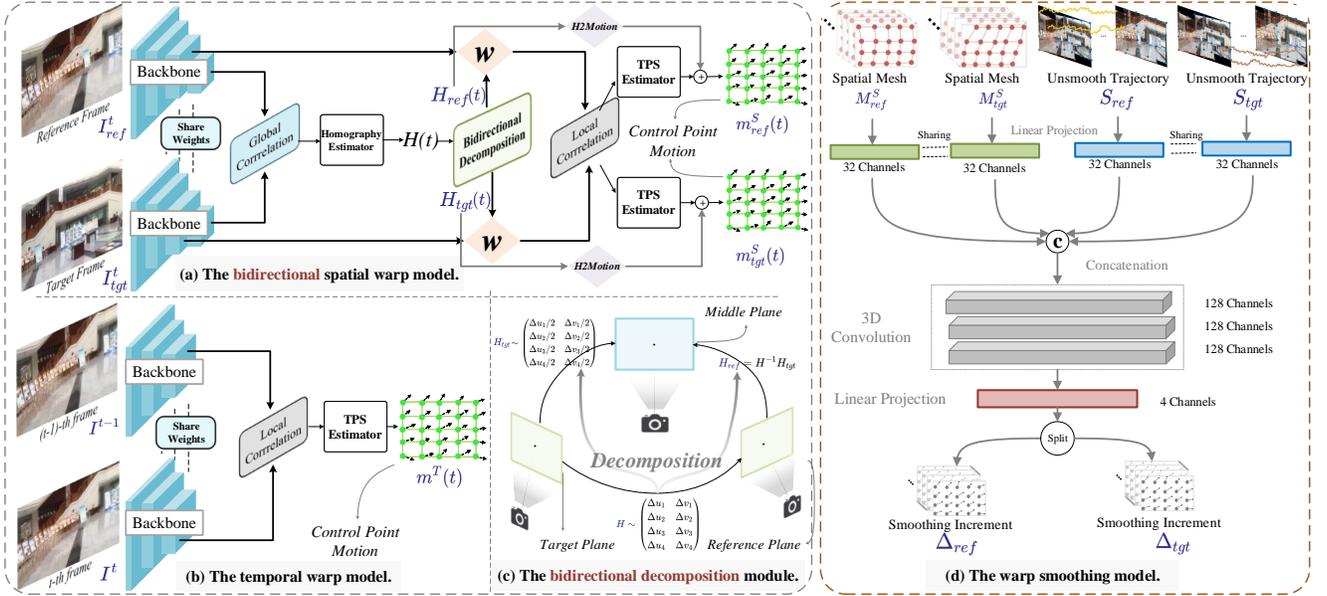


Fig. 3. The network structures of our warping models. The spatial warp, temporal warp, and warp smoothing models are depicted in (a)(b)(d).

Fig. 3 (c) shows the whole decomposition process and all operations are differentiable, making it easy to incorporate into deep learning frameworks.

It is worth noting that our bidirectional decomposition is scalable and not limited to the middle plane. For example, we can manually control the virtual plane closer to the reference or target plane, allocating more distortion to the less salient view and thus producing a more natural appearance in visual perception.

3) *Loss Function*: The total loss function of our spatial model consists of an alignment loss and a shape-preserving loss as follows:

$$\mathcal{L}^S = \mathcal{L}_{align}^S + \lambda \mathcal{L}_{shape}^S. \quad (4)$$

The alignment component leverages photometric errors to encourage the predicted homography, and TPS warps can align the input images as:

$$\begin{aligned} \mathcal{L}_{align}^S = & \omega \| \mathcal{W}(I_{ref}^t, H_{ref}(t)) - \mathcal{W}(I_{tgt}^t, H_{tgt}(t)) \|_1 \cdot OP_h \\ & + \| \mathcal{W}(I_{ref}^t, m_{ref}^S(t)) - \mathcal{W}(I_{tgt}^t, m_{tgt}^S(t)) \|_1 \cdot OP_{tps}. \end{aligned} \quad (5)$$

Here,

$$OP_h = \mathcal{W}(\mathbb{1}, H_{ref}(t)) \cdot \mathcal{W}(\mathbb{1}, H_{tgt}(t)), \quad (6)$$

and

$$OP_{tps} = \mathcal{W}(\mathbb{1}, m_{ref}^S(t)) \cdot \mathcal{W}(\mathbb{1}, m_{tgt}^S(t)). \quad (7)$$

$\mathcal{W}(\cdot, \cdot)$  is the warping process, and  $\mathbb{1}$  is an all-one matrix with the same resolution as the input image.  $OP_h$  and  $OP_{tps}$  denote the overlapping regions in the global and local parts, respectively. We adopt the mesh term [28] (named  $\mathcal{L}_{distortion}(\cdot)$  in our formulations) to calculate our shape-preserving loss, which can be written as:

$$\mathcal{L}_{shape}^S = \mathcal{L}_{distortion}(m_{ref}^S(t)) + \mathcal{L}_{distortion}(m_{tgt}^S(t)). \quad (8)$$

## B. Temporal Warp

1) *Network*: *StabStitch* warps target frames to align with reference frames, so we only need to obtain temporal warps between consecutive target frames. In contrast, *StabStitch++* simultaneously warp the reference and target frames, which requires the temporal warps from consecutive reference and target frames. To this end, we simplify our temporal warp model to reduce the elapsed time. The network is illustrated in Fig. 3 (b), where we take  $(t-1)$ -th and  $t$ -th frame ( $I^{t-1}$ ,  $I^t$ ) as input and output the temporal warps  $m^T(t)$ . We also use the control point motions of TPS transformation to represent the temporal warps, which hold the same resolution of control points as that of the spatial model. Compared with the spatial warp model, our temporal model removes the global homography estimation and only predicts a unidirectional warp to align  $I^t$  with  $I^{t-1}$ . For brevity, we omit the subscript of reference or target frames (*ref/tgt*) in this section.

2) *Loss Function*: Similar to the spatial warp model, the loss function of our temporal model also consists of two components as follows:

$$\mathcal{L}^T = \mathcal{L}_{align}^T + \lambda \mathcal{L}_{shape}^T. \quad (9)$$

Here,

$$\mathcal{L}_{align}^T = \| I^{t-1} - \mathcal{W}(I^t, m^T(t)) \|_1 \cdot \mathcal{W}(\mathbb{1}, m^T(t)), \quad (10)$$

and,

$$\mathcal{L}_{shape}^T = \mathcal{L}_{distortion}(m^T(t)). \quad (11)$$

## C. Stitching Trajectory Generation

1) *Camera Trajectory*: Camera trajectory is widely used in video stabilization and can be defined as a chain of relative motions, such as Euclidean transformations [38], homography transformations [47], etc. Representing the transformation of

the initial frame as an identity matrix  $F(1)$ , the camera trajectories are written as:

$$C(t) = F(1)F(2) \cdots F(t), \quad (12)$$

where  $F(t)$  is the relative transformation from the  $t$ -th frame to the  $(t-1)$ -th frame. Considering that our temporal model directly predicts 2D motions of each control point, we adopt the motion representation of vertex files like MeshFlow [48]. Concretely, we chain the temporal motions of control points as camera trajectories for a straightforward representation:

$$C(t) = m^T(1) + m^T(2) + \cdots + m^T(t), \quad (13)$$

where  $m^T(1)$  is set to all-zero matrix. Note each control point in  $m^T(t)$  is anchored at every vertex in a rigid mesh.

2) *Stitching Trajectory*: Video stabilization leverages the chain of temporal motions as camera paths, whereas in our video stitching system, it throws a question of how to represent the stitching paths of a warped video. We delve into this question by combining the spatial and temporal warp models. Here, we only derive the formulation of stitching paths for the reference view, and the trajectories for the target view can be obtained similarly. Particularly, we first reach the spatial/temporal motions  $m_{ref}^T(t)$ ,  $m_{ref}^S(t-1)$ ,  $m_{ref}^S(t) \in \mathbb{R}^{2 \times (U+1) \times (V+1)}$  and their corresponding meshes  $M_{ref}^T(t)$ ,  $M_{ref}^S(t-1)$ ,  $M_{ref}^S(t) \in \mathbb{R}^{2 \times (U+1) \times (V+1)}$  as follows:

$$\begin{cases} m_{ref}^T(t) = TNet(I_{ref}^{t-1}, I_{ref}^t), \\ m_{ref}^S(t-1), m_{tgt}^S(t-1) = SNet(I_{ref}^{t-1}, I_{tgt}^{t-1}), \\ m_{ref}^S(t), m_{tgt}^S(t) = SNet(I_{ref}^t, I_{tgt}^t), \\ \begin{cases} M_{ref}^T(t) = M^{Rig} + m_{ref}^T(t), \\ M_{ref}^S(t-1) = M^{Rig} + m_{ref}^S(t-1), \\ M_{ref}^S(t) = M^{Rig} + m_{ref}^S(t), \end{cases} \end{cases} \Rightarrow \quad (14)$$

where  $SNet/TNet(\cdot, \cdot)$  represents the spatial/temporal warp model, and  $M^{Rig} \in \mathbb{R}^{2 \times (U+1) \times (V+1)}$  is defined as the pixel positions of control points in a rigid mesh.

Then, we need to derive the stitching motions of warped videos from the spatial/temporal meshes. To align the  $t$ -th frame with the  $(t-1)$ -th frame in the warped video, the temporal meshes from the  $t$ -th frame to the  $(t-1)$ -th frame  $M_{ref}^T(t)$  should also undergo the same transformations as the spatial warps of the  $(t-1)$ -th frame  $M_{ref}^S(t-1)$ . Assuming  $\mathcal{T}(\cdot)$  is the TPS transformation, the desired stitching motions could be represented as the difference between the desired meshes and the actual spatial meshes  $M_{ref}^S(t)$ :

$$s_{ref}(t) = \mathcal{T}_{M^{Rig} \rightarrow M_{ref}^S(t-1)}(M_{ref}^T(t)) - M_{ref}^S(t), \quad (15)$$

Finally, we attain the stitching trajectories by chaining the relative stitching motions between consecutive warped frames as follows:

$$S_{ref}(t) = s_{ref}(1) + s_{ref}(2) + \cdots + s_{ref}(t), \quad (16)$$

where we define  $s_{ref}(1)$  is an all-zero matrix.

Temporal shakes typically arise from discontinuities in sequential trajectories. As evidenced by Eq. 15, we can observe: (1) When the spatial warp degenerates to a constant or

becomes non-existent, the stitching trajectories reduce to conventional camera trajectories [48] used in video stabilization. At this time, the shakes are only from the source videos. (2) When the spatial warps at different timestamps are different or varying irregularly, even if the source videos are stable, temporal shakes will be produced from warping.

#### D. Warp Smoothing

To get a stable warped video, we need to smooth the stitching trajectories and preserve the natural shapes after warping.

1) *Model Achitecture*: In this stage, a warp smoothing model is designed to achieve the above goals. As depicted in Fig. 2, it takes sequences of ( $N$  frames) stitching paths ( $S_{ref}$ ,  $S_{tgt}$ ) and spatial meshes ( $M_{ref}^S$ ,  $M_{tgt}^S$ ) as input, and then outputs the smoothing increments ( $\Delta_{ref}$ ,  $\Delta_{tgt}$ ) as described in the following equation:

$$\Delta_{ref}, \Delta_{tgt} = SmoothNet(S_{ref}, S_{tgt}, M_{ref}^S, M_{tgt}^S), \quad (17)$$

where  $S_{ref}, M_{ref}^S, \Delta_{ref} \in \mathbb{R}^{2 \times N \times (U+1) \times (V+1)}$ .

The network architecture is shown in Fig. 3(d). This model first embeds  $S_{ref}$ ,  $S_{tgt}, M_{ref}^S$ ,  $M_{tgt}^S$  into 32 channels through separate linear projections. Then, these embeddings are concatenated and fed into three 3D convolutional layers to model the spatiotemporal dependencies. Finally, we reproject the hidden results back into 4 channels to get  $\Delta_{ref}$  and  $\Delta_{tgt}$ . The network architecture is designed so compact that it can accomplish efficient smoothing inference. In addition, this simple architecture better highlights the effectiveness of the proposed unsupervised learning scheme.

With the smoothing increment  $\Delta_{ref}$ , we define the smooth stitching paths for the reference view as:

$$\hat{S}_{ref} = S_{ref} + \Delta_{ref}. \quad (18)$$

Then we can expand Eq. 18 based on Eq. 16 and Eq. 15, and obtain:

$$\begin{aligned} \hat{S}_{ref}(t) &= S_{ref}(t-1) + s_{ref}(t) + \Delta_{ref}(t) = S_{ref}(t-1) + \\ &\quad \mathcal{T}_{M^{Rig} \rightarrow M_{ref}^S(t-1)}(M_{ref}^T(t)) - \underbrace{(M_{ref}^S(t) - \Delta_{ref}(t))}_{\text{Smooth spatial mesh}}. \end{aligned} \quad (19)$$

In this case, the last term in Eq. 19 can be regarded as the smooth spatial mesh  $\hat{M}_{ref}^S(t)$ . Hence, the sequences of smooth spatial meshes for the reference view are written as:

$$\hat{M}_{ref}^S = M_{ref}^S - \Delta_{ref}. \quad (20)$$

As for the target view, the corresponding smooth stitching paths and meshes can be calculated in a similar way.

2) *Objective Function*: Built on the warp smoothing model as described in Eq. 17, we define a comprehensive objective function as the balance of different unsupervised constraints:

$$\mathcal{L} = \mathcal{L}_{data} + \omega_1 \mathcal{L}_{smooth} + \omega_2 \mathcal{L}_{shape} + \omega_3 \mathcal{L}_{trajectory}. \quad (21)$$

**Data Term**: The data term encourages the final paths (*i.e.*,  $\hat{S}_{ref}$ ,  $\hat{S}_{tgt}$ ) to be close to the original paths. This constraint alone does not contribute to stabilization. The stabilizing effect

of *StabStitch++* is realized in conjunction with the data term and the subsequent smoothness term. It is formulated as follows:

$$\mathcal{L}_{data} = \|\hat{S}_{ref} - S_{ref}\|_2 + \|\hat{S}_{tgt} - S_{tgt}\|_2. \quad (22)$$

**Smoothness Term:** In a smooth path, each motion should not contain sudden large-angle rotations, and the amplitude of translations should be as consistent as possible. To this end, we encourage the trajectory position at a certain moment to be located at the midpoint between its positions in the preceding and succeeding moments, which implicitly satisfies the above two requirements. It is defined as:

$$\begin{aligned} \mathcal{L}_{smooth} = & \sum_{i=1}^{\bar{m}-1} \alpha_i \|\hat{S}_{ref}(\bar{m}+i) + \hat{S}_{ref}(\bar{m}-i) - 2\hat{S}_{ref}(\bar{m})\|_2 \\ & + \sum_{i=1}^{\bar{m}-1} \alpha_i \|\hat{S}_{tgt}(\bar{m}+i) + \hat{S}_{tgt}(\bar{m}-i) - 2\hat{S}_{tgt}(\bar{m})\|_2, \end{aligned} \quad (23)$$

where  $\bar{m}$  is the middle index of  $N$  ( $N$  is required to be an odd number) and  $\alpha_i$  is a constant between 0 and 1 to impose smoothing constraints of varying significance on trajectories at different temporal intervals.

**Shape Preserving Term:** The warping shakes can be effectively removed under the balance of data and smoothness terms. However, the trajectory of each control point is optimized individually. Actually, each warped video has  $(U+1) \times (V+1)$  control points, which means there are  $(U+1) \times (V+1)$  independently optimized trajectories. When these trajectories are changed inconsistently, significant distortions will be produced. To remove the distortions and encourage different paths to share similar changes, we introduce a shape-preserving term as:

$$\begin{aligned} \mathcal{L}_{shape} = & \frac{1}{N} \sum_{t=1}^N \mathcal{L}_{distortion}(\hat{M}_{ref}^S(t) - M^{Rig}) + \\ & \frac{1}{N} \sum_{t=1}^N \mathcal{L}_{distortion}(\hat{M}_{tgt}^S(t) - M^{Rig}), \end{aligned} \quad (24)$$

where  $\mathcal{L}_{distortion}(\cdot)$  takes mesh motions as input and calculates the mesh distortion as used in Eq. 8 and Eq. 11.

**Trajectory Consistency Term:** The shape-preserving term encourages the trajectories of all control points in a warped video change consistently. However, there might be inconsistent trajectories between the reference warped video and the target one. To address this potential issue, we design a trajectory consistency term. Concretely, we only constrain the trajectory consistency in overlapping areas between different views. However, under the joint effect of the shape-preserving term, the trajectory consistency of overlapping areas can be extended to the complete views.

Besides, these trajectories are anchored to sparse control points, and the control points from different views in the overlapping area hardly overlap after warping. Therefore, these sparse trajectories from different views are not located in the same position. To this end, we resample the sparse trajectories

to pixel-level dense trajectories and then apply the following constraints:

$$\begin{aligned} \mathcal{L}_{trajectory} = & \|\mathcal{W}(\uparrow(\hat{S}_{ref}(t)), \hat{M}_{ref}^S(t) - M^{Rig}) - \\ & \mathcal{W}(\uparrow(\hat{S}_{tgt}(t)), \hat{M}_{tgt}^S(t) - M^{Rig})\|_1 \cdot \hat{O}P_{tps}, \end{aligned} \quad (25)$$

where  $\uparrow(\cdot)$  is the resampling operation and  $\hat{O}P_{tps}$  denotes the overlapping regions that can be obtained following Eq. 7.

#### IV. ONLINE STITCHING

Existing video stitching methods [6] [7] [8] [9] [10] are offline solutions, which smooth the trajectories after the videos are completely captured. Besides, the final result typically takes a long time (much longer than the duration of the video). Unlike them, *StabStitch++* is an online video stitching solution that does not require the subsequent frames after the current frame and stitches videos in real time.

##### A. Online Smoothing

To realize online inference, we define a fixed-length sliding window ( $N$  frames) to cover previous  $N-1$  frames and the current frame, as shown in Fig. 4. Then, the local stitching trajectory inside this window is extracted and smoothed according to Sec. III. Next, warped current frames are rendered using the smooth spatial meshes  $\hat{M}_{ref}^S(N)$ ,  $\hat{M}_{tgt}^S(N)$ . Finally, we blend them to get the stable stitched result (of the current frame) and display it when the next frame arrives. With this mode and efficient architectures, *StabStitch++* can achieve minimal latency with only one frame. To support online stitching, we redesign the objective function of the warp smoothing model (Eq. 21) with two online terms as follows:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{data} + \omega_1 \mathcal{L}_{smooth} + \omega_2 \mathcal{L}_{shape} + \\ & \omega_3 \mathcal{L}_{trajectory} + \omega_4 \mathcal{L}_{online} + \omega_5 \mathcal{L}_{align}. \end{aligned} \quad (26)$$

**Online Collaboration Term:** This first term is an online collaboration term. Compared with the previous offline mode, the online mode could introduce a new issue, wherein the smoothed trajectories in different sliding windows (with partial overlapping sequences) may be inconsistent. It could produce subtle jitters if we chain the last frame of all sliding windows to form a complete stitched video. Therefore, we design this online collaboration term to address the above issue as:

$$\begin{aligned} \mathcal{L}_{online} = & \frac{1}{N-1} \sum_{t=2}^N \|\hat{S}_{ref}^{(\xi)}(t) - \hat{S}_{ref}^{(\xi+1)}(t-1)\|_2 + \\ & \frac{1}{N-1} \sum_{t=2}^N \|\hat{S}_{tgt}^{(\xi)}(t) - \hat{S}_{tgt}^{(\xi+1)}(t-1)\|_2, \end{aligned} \quad (27)$$

where  $\xi$  is the absolute time ranging from  $N$  to the last frame of the videos and also implies the index of sliding windows. In contrast,  $t$  can be regarded as the relative time in a certain sliding window ranging from 1 to  $N$ .

**Online Alignment Term:** In our previous conference version [11], *StabStitch* first carried out pre-alignment and then struggled to preserve the alignment performance while

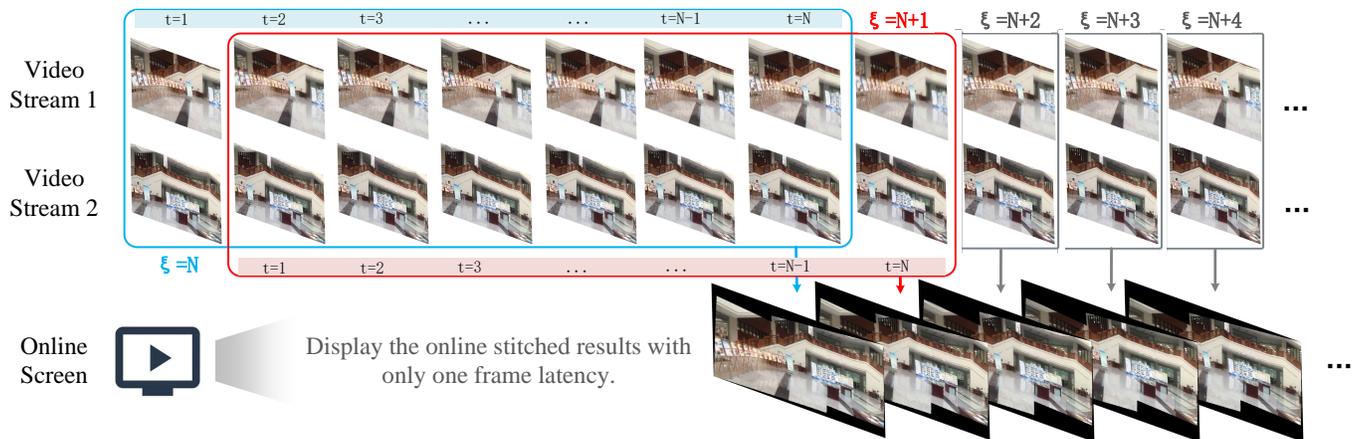


Fig. 4. The online stitching mode. We define a sliding window to process a short sequence and display the last frame on the online screen.

smoothing the stitching trajectories. Different from that, *StabStitch++* makes no compromise and simultaneously optimizes both of them in the warp smoothing model. Although it only takes the low-resolution trajectories and meshes as input to ensure inference efficiency, we leverage high-resolution input images to impose implicit guidance on the deformed meshes, which is proved to be effective in our experiments. We call it an online alignment term and formulate it as follows:

$$\mathcal{L}_{align} = \|\mathcal{W}(I_{ref}^N, \hat{M}_{ref}^S(N) - M^{Rig}) - \mathcal{W}(I_{tgt}^N, \hat{M}_{tgt}^S(N) - M^{Rig})\|_1 \cdot \hat{O}P_{tps}. \quad (28)$$

It is worth noting that we only apply this alignment constraint to the last frame of a sliding window. The effect is comparable to imposing this constraint on all frames, but it reduces massive redundant gradients in the training process.

### B. Offline and Online Inference

Offline smoothing takes the whole trajectories as input, outputs the optimized whole trajectories, and then renders all the video frames. It conducts smoothing after receiving the whole input videos and can be regarded as a special online case in which the sliding window covers all video frames. By contrast, online smoothing takes local trajectories as input, outputs the smoothed local trajectories, and then only renders the last frame in the sliding window. The future frames beyond the current frames are unseen to the online smoothing process. It is more challenging than offline inference because the future frames beyond the current frames are not available in the online smoothing process, and this mode allows real-time playback of the stitched video while capturing input videos.

## V. EXPERIMENT

In this section, we first introduce the proposed dataset and other datasets used in our paper, and then describe the experimental details and metrics. Then we carry out extensive comparative experiments with SoTA solutions. Subsequently, the ablation studies are depicted to show our effectiveness.

### A. Dataset Preparation

a) *StabStitch-D*: Considering the lack of dedicated datasets for video stitching, we establish *StabStitch-D*, a comprehensive dataset for training and evaluation. Our dataset comprises over 100 video pairs, consisting of over 100,000 images, with each video lasting from approximately 5 seconds to 35 seconds. To holistically reveal the performance of video stitching methods in various scenarios, we categorize videos into four classes based on their content, including regular (RE), low-texture (LT), low-light (LL), and fast-moving (FM) scenes. In the testing split, 20 video pairs are divided for testing, with 5 videos in each category. Fig. 5(left) illustrates some examples for each category, where FM is the most challenging case with fast irregular camera movements (rotation or translation). The distribution statistics of video duration are demonstrated in Fig. 5(right). Each video's resolution is resized into  $360 \times 480$  for efficient training, and arbitrary resolutions are supported in the testing phase.

b) *Traditional Dataset*: The videos in *StabStitch-D* are typically stable due to the advancement of video stabilization in software and hardware. Considering that, we also collect some unstable videos from traditional video stitching datasets [6]–[9] to further validate our performance. These videos are captured by different smartphones and drones with a wide range of resolutions such as  $540 \times 960$ ,  $720 \times 1280$ ,  $1080 \times 1920$ , etc. There are a total of 31 unstable video pairs, each of which lasts for more than 10 seconds. These videos are challenging cases that are extremely shaky, low overlapped, or contain dynamic objects.

### B. Implementation Detail

a) *Detail*: We implement the whole framework in PyTorch with one RTX 4090Ti GPU. The spatial warp, temporal warp, and warp smoothing models are trained separately, with the epoch number set to 40, 100, and 50. The parameter sizes of each model are 42.56MB, 24.51MB, and 8.45MB, respectively. Initially, we train the first two warp models (spatial and temporal) and adopt the pre-trained models to predict spatial and temporal meshes (i.e.,  $M_{ref}^S(t)$ ,  $M_{tgt}^S(t)$ ,  $M_{ref}^T(t)$ ,  $M_{tgt}^T(t)$ ) for each frame. These predicted meshes are then leveraged to generate stitching trajectories, which

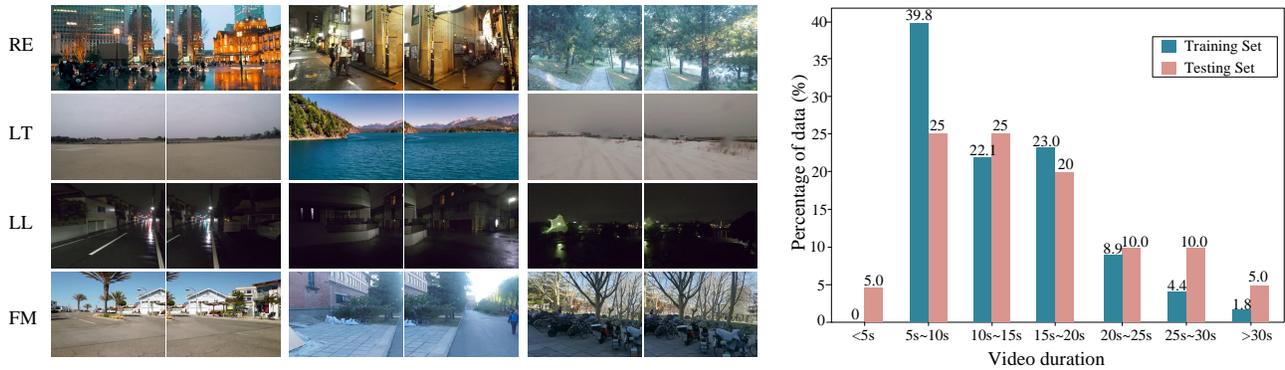


Fig. 5. The proposed *StabStitch-D* dataset. Left: several video examples from diverse scenes. Right: the distribution of video duration time.

further serve as the input of the warp smoothing model. We finally train the warp smoothing models with multiple objective terms.  $\lambda$  and  $\omega$  are defined as 10 and 3. The weights for data, smoothness, shape preservation, online collaboration, trajectory consistency, and online alignment terms are set to 1, 50, 10, 0.1, 10, and 1000.  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  are set to 0.9, 0.3, and 0.1. The control point resolution and sliding window length are empirically set to  $(6 + 1) \times (8 + 1)$  and 7.

b) *Augmentation*: When training the warp smoothing model, we carry out data augmentation by randomly selecting  $N = 7$  frames as the sliding window from a buffer of 12 frames, which could allow more diverse stitching trajectories.

### C. Metric

To evaluate the proposed solution quantitatively, we suggest three metrics: alignment, distortion, and stability.

**Alignment Score**: Following the criterion of UDIS [70] and UDIS++ [5], we also adopt PSNR and SSIM of the overlapping regions to evaluate the alignment performance. We average the scores in all video frames.

**Distortion Score**: The final warps in the online stitching mode can be described as a series of meshes:  $\hat{M}^{S(N)}(N)$ ,  $\hat{M}^{S(N+1)}(N)$ ,  $\dots$ ,  $\hat{M}^{S(\xi)}(N)$ ,  $\dots$ . Then we adopt  $\mathcal{L}_{shape}(\cdot)$  to measure the distortion magnitude. Because any distortion in a single frame will destroy the perfection of the whole video, we choose the mean value of the maximum distortion values of each video as the distortion score.

**Stability Score**: The smoothed trajectories in the online stitching mode can also be described as a series of positions:  $\hat{S}^{(N)}(N)$ ,  $\hat{S}^{(N+1)}(N)$ ,  $\dots$ ,  $\hat{S}^{(\xi)}(N)$ ,  $\dots$ . Then we adopt  $\mathcal{L}_{smooth}(\cdot)$  to measure the stability. The stability score is the mean value of the average smoothness loss of each video.

To intuitively compare with our conference version (*i.e.*, *StabStitch* [11]), we only calculate the metrics of target frames for distortion and stability scores.

### D. Compared with Current Solutions

We compare our method with image and video stitching solutions, respectively.

1) *Compared with Image Stitching*: Two representative SoTA image stitching methods are selected to compare with our solution: LPC [4] (traditional method), and UDIS++ [5] (learning-based method). The quantitative comparison results are illustrated in Tab. I, where ‘./.’ indicates the PSNR/SSIM values. ‘-’ implies the approach fails in this category (*e.g.*, program crash or extremely severe distortion). The results show that our solution achieves better alignment performance than the current SoTA image stitching methods.

2) *Compared with Video Stitching*: Next, we compare our method with video stitching methods, including Nie *et al.*’s video stitching solution [6] (traditional method) and *StabStitch* [11] (learning-based method). To our knowledge, they are the latest and best video stitching methods for hand-held cameras. For *StabStitch* [11], we conduct quantitative and qualitative comparisons with it. As illustrated in Tab. III and Fig. 6, the proposed *StabStitch++* gains comprehensive performance improvements. As for Nie *et al.*’s video stitching solution [6], it tends to fail in specific challenging cases, such as low texture or low light. Therefore, we replace the quantitative experiments with user studies and demonstrate the qualitative results in Tab. II and Fig. 6.

In particular, in the testing set of *StabStitch-D* dataset [11] (20 video cases in total), Nie *et al.* [6] fail in 10 video cases because of program crashes. Hence, we exclude these failure cases and conduct the user study only on the successful cases. For a stitched video, different methods may perform differently at different times. To this end, we segment each complete stitched video into one-second clips (we omit the last clip of a stitched video that is shorter than one second in practice), resulting in 128 clips in total. Then, we invite 20 participants, including 10 researchers/students with computer vision backgrounds and 10 volunteers outside this community. In each test session, two clips from different methods are presented randomly, and every volunteer is required to indicate their overall preference for alignment, distortion, and stability. We average the preference rates and exhibit the results in Tab. II, where our results are clearly preferred.

### E. Analysis

1) *More Result*: We further evaluate our solution on traditional datasets, where the videos are collected from previous video stitching works [6]–[9]. Due to the limitations of pre-

TABLE I  
 QUANTITATIVE COMPARISONS WITH IMAGE STITCHING METHODS ON *StabStitch-D* DATASET. \* INDICATES THE MODEL IS RE-TRAINED ON THE PROPOSED DATASET.

	Method	Regular	Low-Light	Low-Texture	Fast-Moving	Average
1	LCP [4]	24.22/0.812	-	-	23.88/0.813	-
2	UDIS++ [5]	23.19/0.785	31.09/0.936	29.98/0.906	21.56/0.756	27.19/0.859
3	UDIS++ * [5]	24.63/0.829	34.26/0.957	32.81/0.920	24.78/0.819	29.78/0.891
4	StabStitch++	<b>25.51/0.837</b>	<b>35.10/0.958</b>	<b>34.23/0.928</b>	<b>25.64/0.830</b>	<b>30.80/ 0.897</b>



Fig. 6. Qualitative comparison with Nie *et al.*'s video stitching [6] and *StabStitch* [11] on the *StabStitch-D* dataset. The arrows indicate artifacts or distortions, and the numbers below the images exhibit the time at which the frame appears in the video. Please zoom in for the best view.

TABLE II  
 USER STUDY OF THE PREFERENCE ON *StabStitch-D* DATASET. WE EXCLUDE THE FAILURE CASES OF NIE *et al.* [6] FOR FAIRNESS.

StabStitch++	Nie <i>et al.</i> [6]	No preference
34.38%	3.91%	61.71%

TABLE III  
 QUANTITATIVE COMPARISONS WITH VIDEO STITCHING METHODS ON *StabStitch-D* DATASET.

Method	Alignment $\uparrow$	Stability $\downarrow$	Distortion $\downarrow$
1 StabStitch [11]	29.89/0.890	48.74	0.674
2 StabStitch++	<b>30.88/0.898</b>	<b>41.70</b>	<b>0.371</b>

vious video stabilization technologies, almost all input videos are shaky, and some are even intentionally shaky. Even though this situation does not conform to the current technology state, our method can still stably stitch them together, especially in various challenging scenes such as deliberate shakes, low overlap rates, dynamic objects, etc. The results are simply exhibited in Fig. 7. Please refer to the supplementary video for more details.

2) *Inference Speed*: A comprehensive analysis of the inference speed is provided in Tab. IV with one RTX 4090Ti GPU, where ‘Blending’ represents the average blending. We conduct the experiment on the third case of category ‘RE’ in Fig. 5. As shown in Tab. IV, *StabStitch* [11] only takes about 28.2ms to stitch one frame, yielding a real-time online video stitching system. When stitching a video pair with higher resolution, only the time for warping and blending steps will slightly increase. In contrast, Nie *et al.*'s solution [6] takes over 40 minutes to get such a 26-second stitched video with an Intel i7-9750H 2.60GHz CPU, making it impractical to be applied to online stitching. For *StabStitch++*, it takes 35.3ms to deal with one frame, which is only a little longer than that of *StabStitch* [11]. Theoretically, *StabStitch++* would double the inference time to generate bidirectional warps. However, due to the bidirectional decomposition module of the spatial warp model and the lightweight structure of the temporal warp model, the total running time only increases by 7.1ms.



Fig. 7. Our video stitching results on traditional datasets. Refer to the supplementary video for more details.

TABLE IV  
A COMPREHENSIVE ANALYSIS OF INFERENCE SPEED.

Component	SNet	TNet	Trajectory generation	SmoothNet	Warping	Blending	Total
1 StabStitch [11]	11.5ms	10ms	1.1ms	1ms	4.4ms	0.2ms	28.2ms
2 StabStitch++	16.1ms	6.9ms	2.2ms	1.4ms	8.5ms	0.2ms	35.3ms

TABLE V  
THE EFFECTIVENESS OF THE PROPOSED BIDIRECTIONAL WARPS ON ALIGNMENT PERFORMANCE.

Method	UDIS-D [70]	StabStitch-D [11]
1 UDIS++ [5]	25.43/0.838	29.78/0.891
2 UDIS++ (Bidirectional Warps)	<b>26.13/0.852</b>	<b>31.34/0.907</b>



Fig. 8. The effectiveness of the proposed bidirectional decomposition module on distortion performance. The red rectangles indicate projective distortions. The gray regions in the second row represent the space saved by projecting source views onto the middle plane.

### F. Ablation Study

Here, we conduct extensive ablation studies to reveal the effectiveness of the proposed solution.

**Bidirectional Warps:** In addition to video stitching, the proposed bidirectional decomposition module can also be applied in the more common field (*i.e.*, image stitching) and bring significant gains. We demonstrate the benefit of alignment in Tab. V, where we validate its effectiveness on both image stitching (UDIS-D [70]) and video stitching (*StabStitch-D* [11]) datasets. Moreover, we display the qualitative comparisons in Fig. 8, where we further report the output

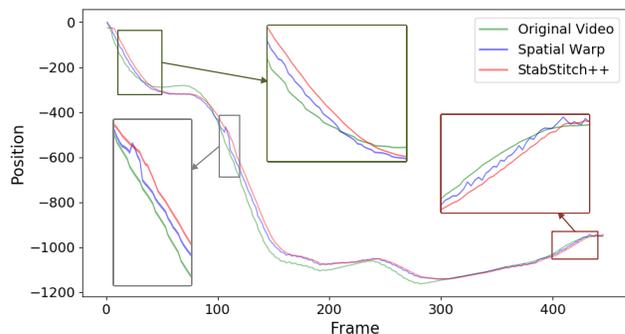


Fig. 9. Trajectory visualization. It exhibits the complete trajectories of the original video and warped videos (by our spatial warp and *StabStitch++*) to show the occurrence and elimination of warping shakes.

resolution and the rate of invalid areas. Compared with the unidirectional warp (*i.e.*, UDIS++), the proposed bidirectional decomposition module evenly spreads projective distortions across both views, yielding more natural stitched results with smaller output resolutions and fewer invalid areas.

**Objective Terms:** *StabStitch++* is an unsupervised framework that is trained with a hybrid loss. Hence, we evaluate the effect of each objective term, and the results are demonstrated in Tab. VI. (Experiment 1) When there is only the data term, *StabStitch++* degenerates into an ordinary image stitching model. (Experiment 2) Then, we add the smoothness term. With their joint action, the stability of stitched videos improves but alignment and distortion worsen. (Experiment 3) The shape-preserving term significantly preserves the natural structures but further decreases the alignment performance. (Experiment 4) Next, we introduce the online collaboration term, which ensures the warping consistency among the sliding windows and brings comprehensive improvements. (Experiment 5) Compared with *StabStitch* [11], we design a trajectory consistency for the bidirectional warps from different views. Although the metrics are slightly decreased, it ensures the consistency of two views after warping. (Experiment 6) One of the biggest differences with *StabStitch* [11] is that the proposed *StabStitch++* can simultaneously optimize the alignment and

TABLE VI  
ABLATION STUDIES ON DIFFERENT OBJECTIVE TERMS.

	$\mathcal{L}_{data}$	$\mathcal{L}_{smooth}$	$\mathcal{L}_{shape}$	$\mathcal{L}_{online}$	$\mathcal{L}_{trajectory}$	$\mathcal{L}_{align}$	Augmentation	Alignment $\uparrow$	Stability $\downarrow$	Distortion $\downarrow$
1	✓							30.30/0.893	51.78	0.338
2	✓	✓						29.38/0.876	41.60	0.594
3	✓	✓	✓					28.83/0.869	41.57	0.313
4	✓	✓	✓	✓				28.91/0.871	<b>41.54</b>	<b>0.299</b>
5	✓	✓	✓	✓	✓			28.24/0.857	41.81	0.319
6	✓	✓	✓	✓	✓	✓		30.77/0.897	41.88	0.469
7	✓	✓	✓	✓	✓	✓	✓	<b>30.88/0.898</b>	41.70	0.371

stabilization. With the online alignment term, *StabStitch++* can find the optimal solution that satisfies both conditions in the online mode rather than sacrificing one condition to enhance another. (Experiment 7) Finally, we validate the effect of our data augmentation strategy (mentioned in Sec. V-B). Similar to the online collaboration term, it also brings in comprehensive improvements including alignment, stability, and distortion.

**Trajectory Visualization:** We visualize the camera trajectories of the original video and the stitching trajectories of warped videos from the testing set of *StabStitch-D* dataset in Fig. 9. Here, the trajectories are extracted from a certain control point in the horizontal direction. It can be observed that even if the input video is stable, image stitching can introduce undesired warping shakes, whereas *StabStitch++* minimizes these shakes as much as possible during stitching.

## VI. LIMITATION

While *StabStitch++* demonstrates impressive performance across diverse scenarios, several limitations remain open for future exploration.

Although our method uses semantic features to improve robustness in low texture compared to traditional feature-based approaches, alignment quality may degrade when overlapping regions lack sufficient visual cues (e.g., uniform walls or skies). This is a common challenge for vision-based methods relying on appearance consistency. In the future, the pre-trained foundational models or other geometric priors may be leveraged to infer structural information in such cases, potentially improving alignment in feature-sparse environments. Besides, to ensure computational efficiency, *StabStitch++* processes downsampled versions of high-resolution videos to estimate spatiotemporal transformations, which are then up-scaled to the original resolutions and applied to the source videos. While this approach avoids excessive GPU resource consumption, it may introduce precision loss during the scaling process, particularly for ultra-high-resolution inputs (e.g., 4K or beyond). Future work could explore dynamic resolution adjustments, where the system processes high-resolution videos at lower scales for efficiency and selectively refines critical regions at full resolution for accuracy.

## VII. CONCLUSION

In this paper, we retarget video stitching to an emerging issue, named *warping shake*, considering the current development of stabilization technology. It describes the undesired content instability caused by temporally unsmooth warps when

image stitching technology is directly applied to videos. To solve this problem, we present *StabStitch++*, an unsupervised online video stitching framework with spatiotemporal bidirectional warps. Concretely, we first estimate spatial bidirectional warps by determining the virtual middle plane and then integrate them with temporal warps to formulate the mathematical expression of stitching trajectories. Next, a warp smoothing model is proposed to simultaneously achieve content alignment, trajectory smoothness, and online collaboration using a hybrid objective loss. Moreover, a video stitching dataset with various camera motions and scenes is built, which we hope can work as a benchmark and promote other related research work. Compared with our conference version [11], we extend *StabStitch* [11] to *StabStitch++* through bidirectional decomposition and joint optimization, yielding better alignment, fewer distortions, and higher stability.

## REFERENCES

- [1] W.-S. Lai, O. Gallo, J. Gu, D. Sun, M.-H. Yang, and J. Kantz, "Video stitching for linear camera arrays," in *BMVC*, 2019, p. 130.
- [2] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection—a new baseline," in *CVPR*, 2018, pp. 6536–6545.
- [3] I.-C. Lo, K.-T. Shih, and H. H. Chen, "Efficient and accurate stitching for 360° dual-fisheye images and videos," *IEEE TIP*, vol. 31, pp. 251–262, 2021.
- [4] Q. Jia, Z. Li, X. Fan, H. Zhao, S. Teng, X. Ye, and L. J. Latecki, "Leveraging line-point consistency to preserve structures for wide parallax image stitching," in *CVPR*, 2021, pp. 12 186–12 195.
- [5] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, "Parallax-tolerant unsupervised deep image stitching," in *ICCV*, 2023, pp. 7399–7408.
- [6] Y. Nie, T. Su, Z. Zhang, H. Sun, and G. Li, "Dynamic video stitching via shakiness removing," *IEEE TIP*, vol. 27, no. 1, pp. 164–178, 2017.
- [7] T. Su, Y. Nie, Z. Zhang, H. Sun, and G. Li, "Video stitching for handheld inputs via combined video stabilization," in *SIGGRAPH ASIA 2016 Technical Briefs*, 2016, pp. 1–4.
- [8] H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, "Joint video stitching and stabilization from moving cameras," *IEEE TIP*, vol. 25, no. 11, pp. 5491–5503, 2016.
- [9] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng, "Seamless video stitching from hand-held camera inputs," in *Computer Graphics Forum*, vol. 35, no. 2. Wiley Online Library, 2016, pp. 479–487.
- [10] W. Jiang and J. Gu, "Video stitching with spatial-temporal content-preserving warping," in *CVPRW*, 2015, pp. 42–48.
- [11] L. Nie, C. Lin, K. Liao, Y. Zhang, S. Liu, R. Ai, and Y. Zhao, "Eliminating warping shakes for unsupervised online video stitching," in *ECCV*, 2024, pp. 390–407.
- [12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.
- [13] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE TPAMI*, vol. 32, no. 4, pp. 722–732, 2008.
- [14] F. Zhang and F. Liu, "Parallax-tolerant image stitching," in *CVPR*, 2014, pp. 3262–3269.
- [15] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *IJCV*, vol. 74, no. 1, pp. 59–73, 2007.

- [16] J. Zaragoza, T.-J. Chin, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving dlt," in *CVPR*, 2013, pp. 2339–2346.
- [17] J. Li, Z. Wang, S. Lai, Y. Zhai, and M. Zhang, "Parallax-tolerant image stitching based on robust elastic warping," *IEEE TMM*, vol. 20, no. 7, pp. 1672–1687, 2017.
- [18] K.-Y. Lee and J.-Y. Sim, "Warping residual based image stitching for large parallax," in *CVPR*, 2020, pp. 8198–8206.
- [19] J. Li, B. Deng, R. Tang, Z. Wang, and Y. Yan, "Local-adaptive image alignment based on triangular facet approximation," *IEEE TIP*, vol. 29, pp. 2356–2369, 2019.
- [20] C.-H. Chang, Y. Sato, and Y.-Y. Chuang, "Shape-preserving half-projective warps for image stitching," in *CVPR*, 2014, pp. 3254–3261.
- [21] C.-C. Lin, S. U. Pankanti, K. Natesan Ramamurthy, and A. Y. Aravkin, "Adaptive as-natural-as-possible image stitching," in *CVPR*, 2015, pp. 1155–1163.
- [22] S. Li, L. Yuan, J. Sun, and L. Quan, "Dual-feature warping-based motion model estimation," in *ICCV*, 2015, pp. 4283–4291.
- [23] T. Liao and N. Li, "Single-perspective warps in natural image stitching," *IEEE TIP*, vol. 29, pp. 724–735, 2019.
- [24] Y.-S. Chen and Y.-Y. Chuang, "Natural image stitching with the global similarity prior," in *ECCV*, 2016, pp. 186–201.
- [25] P. Du, J. Ning, J. Cui, S. Huang, X. Wang, and J. Wang, "Geometric structure preserving warp for natural image stitching," in *CVPR*, 2022, pp. 3688–3696.
- [26] Y. Zhang, Y.-K. Lai, and F.-L. Zhang, "Content-preserving image stitching with piecewise rectangular boundary constraints," *IEEE TVCG*, vol. 27, no. 7, pp. 3198–3212, 2020.
- [27] K. He, H. Chang, and J. Sun, "Rectangling panoramic images via warping," *ACM TOG*, vol. 32, no. 4, pp. 1–10, 2013.
- [28] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, "Deep rectangling for image stitching: A learning baseline," in *CVPR*, 2022, pp. 5740–5748.
- [29] L. Nie, C. Lin, K. Liao, M. Liu, and Y. Zhao, "A view-free image stitching network based on global homography," *Journal of Visual Communication and Image Representation*, vol. 73, p. 102950, 2020.
- [30] L. Nie, C. Lin, K. Liao, and Y. Zhao, "Learning edge-preserved image stitching from multi-scale deep homography," *Neurocomputing*, vol. 491, pp. 533–543, 2022.
- [31] Z. Jiang, Z. Zhang, X. Fan, and R. Liu, "Towards all weather and unobstructed multi-spectral image stitching: Algorithm and benchmark," in *ACM MM*, 2022, pp. 3783–3791.
- [32] D.-Y. Song, G.-M. Um, H. K. Lee, and D. Cho, "End-to-end image stitching network via multi-homography estimation," *SPL*, vol. 28, pp. 763–767, 2021.
- [33] M. Kim, Y. Lee, W. K. Han, and K. H. Jin, "Learning residual elastic warps for image stitching under dirichlet boundary condition," in *WACV*, 2024, pp. 4016–4024.
- [34] Y. Zhang, Y. Lai, N. Lang, F.-L. Zhang, and L. Xu, "Recstitchnet: Learning to stitch images with rectangular boundaries," *Computational Visual Media*, 2024.
- [35] H. Kweon, H. Kim, Y. Kang, Y. Yoon, W. Jeong, and K.-J. Yoon, "Pixel-wise warping for deep image stitching," in *AAAI*, vol. 37, no. 1, 2023, pp. 1196–1204.
- [36] Q. Jia, X. Feng, Y. Liu, X. Fan, and L. J. Latecki, "Learning pixel-wise alignment for unsupervised image stitching," in *ACM MM*, 2023, pp. 1392–1400.
- [37] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3d video stabilization," *ACM TOG*, p. 1–9, 2009.
- [38] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *CVPR*. IEEE, 2012, pp. 89–95.
- [39] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM TOG*, vol. 30, no. 1, pp. 1–10, 2011.
- [40] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM TOG*, vol. 31, no. 5, pp. 1–10, 2012.
- [41] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE TPAMI*, vol. 28, no. 7, pp. 1150–1163, 2006.
- [42] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *CVPR*. IEEE, 2011, pp. 225–232.
- [43] T. Ma, Y. Nie, Q. Zhang, Z. Zhang, H. Sun, and G. Li, "Effective video stabilization via joint trajectory smoothing and frame warping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3163–3176, 2019.
- [44] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," *CSTR*, vol. 1, no. 2, p. 13, 2011.
- [45] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala, "Light field video stabilization," in *ICCV*. IEEE, 2009, pp. 341–348.
- [46] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Calibration-free rolling shutter removal," in *IEEE International Conference on Computational Photography*. IEEE, 2012, pp. 1–8.
- [47] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM TOG*, vol. 32, no. 4, pp. 1–10, 2013.
- [48] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "Meshflow: Minimum latency online video stabilization," in *ECCV*. Springer, 2016, pp. 800–815.
- [49] S. Liu, L. Yuan, P. Tan, and J. Sun, "Steadyflow: Spatially smooth optical flow for video stabilization," in *CVPR*, 2014, pp. 4209–4216.
- [50] J. Yu and R. Ramamoorthi, "Selfie video stabilization," in *ECCV*, 2018, pp. 551–566.
- [51] J. Yu, R. Ramamoorthi, K. Cheng, M. Sarkis, and N. Bi, "Real-time selfie video stabilization," in *CVPR*, 2021, pp. 12 036–12 044.
- [52] J. Kopf, "360 video stabilization," *ACM TOG*, vol. 35, no. 6, pp. 1–9, 2016.
- [53] C. Tang, O. Wang, F. Liu, and P. Tan, "Joint stabilization and direction of 360° videos," *ACM TOG*, vol. 38, no. 2, pp. 1–13, 2019.
- [54] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM TOG*, vol. 34, no. 4, pp. 1–9, 2015.
- [55] M. Wang, G.-Y. Yang, J.-K. Lin, S.-H. Zhang, A. Shamir, S.-P. Lu, and S.-M. Hu, "Deep online video stabilization with multi-grid warping transformation learning," *IEEE TIP*, vol. 28, no. 5, pp. 2283–2292, 2018.
- [56] S.-Z. Xu, J. Hu, M. Wang, T.-J. Mu, and S.-M. Hu, "Deep video stabilization using adversarial networks," in *Computer Graphics Forum*, vol. 37, no. 7. Wiley Online Library, 2018, pp. 267–276.
- [57] Z. Zhang, Z. Liu, P. Tan, B. Zeng, and S. Liu, "Minimum latency deep online video stabilization," in *ICCV*, 2023, pp. 23 030–23 039.
- [58] J. Choi and I. S. Kweon, "Deep iterative frame interpolation for full-frame video stabilization," *ACM TOG*, vol. 39, no. 1, pp. 1–9, 2020.
- [59] Y. Xu, J. Zhang, S. J. Maybank, and D. Tao, "Dut: Learning video stabilization by simply watching unstable videos," *IEEE TIP*, vol. 31, pp. 4306–4320, 2022.
- [60] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross, "Panoramic video from unstructured camera arrays," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 57–68.
- [61] A. Hamza, R. Hafiz, M. M. Khan, Y. Cho, and J. Cha, "Stabilization of panoramic videos from mobile multi-camera platforms," *Image and Vision Computing*, vol. 37, pp. 20–30, 2015.
- [62] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [63] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE TPAMI*, vol. 11, no. 6, pp. 567–585, 1989.
- [64] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, "Semi-supervised coupled thin-plate spline model for rotation correction and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
- [66] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, "Depth-aware multi-grid deep homography estimation with contextual correlation," *IEEE TCSVT*, vol. 32, no. 7, pp. 4460–4472, 2021.
- [67] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *CVPR*, 2018, pp. 8934–8943.
- [68] K. Liao, L. Nie, S. Huang, C. Lin, J. Zhang, Y. Zhao, M. Gabbouj, and D. Tao, "Deep learning for camera calibration and beyond: A survey," *arXiv preprint arXiv:2303.10559*, 2023.
- [69] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.
- [70] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, "Unsupervised deep image stitching: Reconstructing stitched features to images," *IEEE TIP*, vol. 30, pp. 6184–6197, 2021.