PWStableNet: Learning Pixel-Wise Warping Maps for Video Stabilization

Minda Zhao and Qiang Ling, *Senior Member, IEEE*

***Abstract*— As the videos captured by hand-held cameras are often perturbed by high-frequency jitters, stabilization of these videos is an essential task. Many video stabilization methods have been proposed to stabilize shaky videos. However, most methods estimate one global homography or several homographies based on fixed meshes to warp the shaky frames into their stabilized views. Due to the existence of parallax, such single or a few homographies can not well handle the depth variation. In contrast to these traditional methods, we propose a novel video stabilization network, called PWStableNet, which comes up pixel-wise warping maps, i.e., potentially different warping for different pixels, and stabilizes each pixel to its stabilized view. To our best knowledge, this is the first deep learning based pixel-wise video stabilization. The proposed method is built upon a multi-stage cascade encoder-decoder architecture and learns pixel-wise warping maps from consecutive unstable frames. Inter-stage connections are also introduced to add feature maps of a former stage to the corresponding feature maps at a latter stage, which enables the latter stage to learn the residual from the feature maps of former stages. This cascade architecture can produce more precise warping maps at latter stages. To ensure the correct learning of pixel-wise warping maps, we use a well-designed loss function to guide the training procedure of the proposed PWStableNet. The proposed stabilization method achieves comparable performance with traditional methods, but stronger robustness and much faster processing speed. Moreover, the proposed stabilization method outperforms some typical CNN-based stabilization methods, especially in videos with strong parallax. Codes will be provided at https://github.com/mindazhao/pix-pix-warping-video-stabilization.**

***Index Terms*— Video stabilization, pixel-wise warping, cascade networks.**

I. Introduction

NOWADAYS, with the popularity of hand-held cameras including mobile phones [1], [2], more and more videos are generated everyday. Due to the lack of professional stabilizing instruments, the camera jitter causes serious undesirable motion in amateur videos and thus leads to visual discomfort. Many stabilization methods [3], [4], [5], [6] have been proposed on the basis of the traditional computer vision in the past decade. Most of these methods first extract feature trajectories [7]–[9] or use optical flow [10] to estimate the camera motion, then smooth them for stabilization, and finally calculates one global homography or a few homographies based on fixed meshes to warp shaky frames into stabilized ones. Due to the existence of parallax, the depth variation prevents these methods from producing stable videos. Moreover, these methods are usually challenged by insufficient robustness and may fail in low-quality videos, such as night-scene videos, blurry videos, noisy videos.

In recent years, the convolutional neural network(CNN) has achieved great successes in computer vision tasks, such as image recognition [11]–[13], object detection [14], [15] and segmentation [16], [17]. Particularly some traditional video processing problems have been reconsidered with deep neural networks, e.g. image denoising [18] and video deblurring [19]. By creating a practical dataset, CNN is also implemented to video stabilization [20], [21]. Given a sequence of frames, these methods calculate one or several homographies (or affine matrices) which are applied for stabilized frame synthesis. The CNN-based methods are more robust than traditional stabilization methods since they are adept in extracting high-dimensional abstract features, instead of relying on artificial feature extraction and matching whose accuracy strongly depends on the quality of videos. However, they are quite similar with the traditional methods in terms of the implemented global or mesh-based transformation, which can be quite fragile in the case of strong parallax. Generally speaking, warping with a certain number of homographies is reasonable approximation of the actual inter-frame transformation. But its approximation error can vary significantly for different videos, especially the ones with strong parallax. To our best knowledge, no deep-learning-based pixel-wise warping map has been proposed for video stabilization. For this purpose, we propose a novel network, called PWStableNet, to learn pixel-wise warping maps for frame synthesis and remove the jitter for various unstable videos.

The pipeline of the proposed method is shown in Fig. 1. To stabilize a certain frame, a set of neighbor frames are taken as the input of PWStableNet, and two warping maps, *i.e.,* horizontal and vertical warping maps, are estimated. For each pixel, the corresponding values in two warping maps represent the mapping which transforms the original position of that pixel to the desired position under the stabilized view. We design a multi-stage cascade encoder-decoder architecture

Manuscript received June 23, 2019; revised November 7, 2019 and December 3, 2019; accepted December 27, 2019. Date of publication January 7, 2020; date of current version January 30, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC0201003, in part by the Technological Innovation Project for New Energy and Intelligent Networked Automobile Industry of Anhui Province, and in part by the Fundamental Research Funds for the Central Universities. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chia-Kai Liang. *(Corresponding author: Qiang Ling.)*

The authors are with the Department of Automation, University of Science and Technology of China, Hefei 230027, China (e-mail: qling@ustc.edu.cn).

Digital Object Identifier 10.1109/TIP.2019.2963380

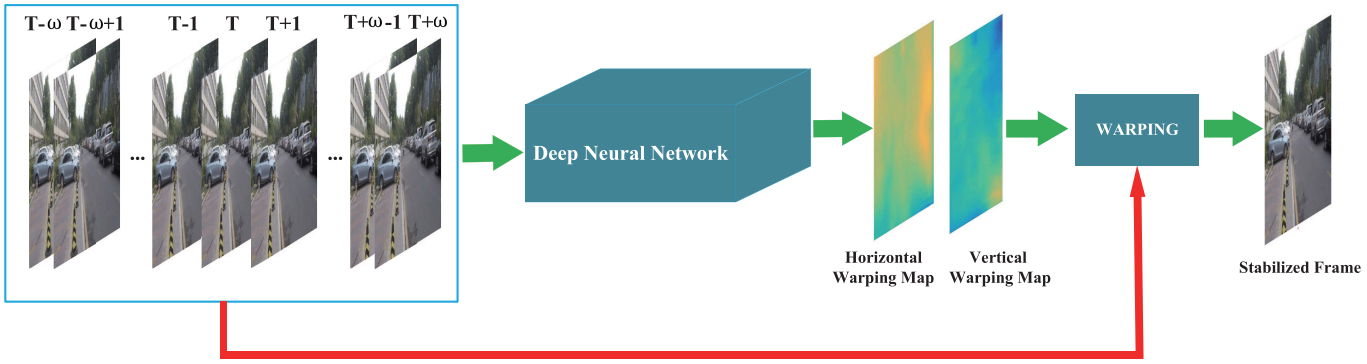


Fig. 1. The procedure of the proposed video stabilization method. To stabilize a certain frame, a set of neighbor frames are considered as input and the PWStableNet estimates two warping maps (*i.e.,* horizontal and vertical warping maps), with which we generate the stabilized view of a certain frame.

where each stage receives the same sequence of unstable frames as input. Between two adjacent stages, we introduce inter-stage connection to add feature maps of a former stage to the corresponding feature maps of latter stages, which enable latter stages to learn the residual with respect to former stages. This cascade architecture can produce more precise warping maps at latter stages. Since the estimated warping maps of the proposed PWStableNet have much higher degree of freedom than the ones of conventional methods, which just calculate several homographies, we design a loss function to guide the training procedure of PWStableNet and avoid the distortion of warped frames.

We train our PWStableNet on *DeepStab* dataset [21]. With the well-designed architecture and loss function, the network can analyze the motion characteristics sufficiently and extract the low-frequency components of unstable frame sequences. Experiments on public unstable videos show the proposed PWStableNet can achieve comparable performance with many existing stabilization methods for many unstable videos. Since CNNs can extract more abstract features from frames, the proposed method can perform robustly on low-quality videos, such as night-scene videos, blurring video, while artificial feature-matching based stabilization methods may fail. Moreover, instead of calculating one or a set of homographies (or affine matrices), our PWStableNet estimates pixel-wise warping maps and transforms each pixel to the desired position under the stabilized view. It does not require any small regions to fit the same homography, which means much more freedom is allowed. So the proposed method can better solve video stabilization problems with discontinuous depth variation. To our best knowledge, this is the first deep learning based pixel-wise video stabilization. PWStableNet is trained on NIVDIA GTX 1080Ti and can stabilize videos at the speed of about 56 FPS. The latency of PWStableNet is determined by the length of concerned neighbor frames, which is closely related to the frequency of jitter we expect to remove. The proposed PWStableNet has great potential for real-time video stabilization and may yield better performance after being trained with larger datasets.

The remainder of this paper is organized as follows. Section II presents the related work. Section III presents the architecture of PWStableNet and the designed loss function. Section IV compares our PWStableNet with some state-of-the-art stabilization methods through several public videos. Section V discusses the limitation of PWStableNet. Concluding remarks are placed in Section VI.

II. Related Work

A. Digital Video Stabilization

Traditional video stabilization methods estimate the inter-frame motion by extracting feature points or calculating optical flow, then smooth the estimated motion and warp the unsteady frames to their stabilized view. These methods can be roughly classified into three types, 2D, 3D and 2.5D methods.

2D methods aim to stabilize videos mainly containing planar motion. Through image matching technologies, such as feature matching, 2D methods usually model camera motion with inter-frame transformation matrix sequences [22], [23]. Then different methods are proposed to smooth them [4], [24]. Grundmann *et al.* [8] limited different order derivatives of calculated camera paths to smooth each frame. Liu *et al.* [25] and Zhang *et al.* [26] introduced the *as-similar-as-possible* constraint to make the estimation robust to nonplanar motions. 3D methods can solve the problems caused by parallax. Such 3D methods record the feature trajectories and reconstruct the 3D locations with Structure from Motion (SFM) technique [27]. Buehler *et al.* [28] got the smoothed location of feature points by limiting the speed of the projected feature points to be constant. Liu *et al.* [2] developed a content-preserving warping for 3D video stabilization. Smith *et al.* [29] employed a space-time optimization to create the virtual camera path according to the camera array. Generally speaking, 3D methods are much slower than 2D methods due to the implementation of SFM.

Recently, inspired by 2D and 3D methods, 2.5D methods have been developed based on continuous feature trajectories. Liu *al.* [5] extracted 2D feature trajectories and utilized the subspace constraints [30] on feature trajectories. In order to solve the parallax problem, content preserving warping [31] is also adopted. However, this method can not handle the dynamic scenes with rapidly moving background. Liu *et al.* designed the steadyflow [10] and further extended it to meshflow [32] for motion estimation. These methods can perform

well for most videos, but may fail when large foreground objects exist. Feedback-based methods [3], [33] can distinguish background and foreground feature trajectories, even when large foreground objects occupy more than half of a frame. In [34], a method was proposed to stabilize videos with both background and foreground feature trajectories. However, these methods only estimate a global homography to warp a frame and may produce undesirable local errors caused by depth variation.

In summary, digital video stabilization can solve many shaky videos. They can handle errors caused by parallax with post-processing, like mesh-based warping. Dividing a frame into more regions usually yields more accurate stabilization results, but at the cost of greater time consumption. Moreover, the fixed region dividing patterns may not work well for all videos. Automatically warping different regions with different transformations is the most ideal, but hard to be realized, which is the main motivation of the present paper.

B. CNNs for Video Processing

In recent years, CNNs have made huge progresses in computer vision tasks. CNNs can predict optical flow [35], camera pose [36], frame interpolation [37] and deblurring [19]. However, few research results regarding video stabilization with deep neural networks have been reported due to the lack of training video datasets. Fortunately, Wang *et al.* built a novel dataset called *DeepStab* dataset which includes 61 pairs of stable and unstable videos [21]. They also designed a network for multi-grid warping transformation learning, which can achieve comparable performance with traditional methods and is more robust for low-quality videos. In [20], adversarial networks are taken to estimate an affine matrix, with which steady frames can be generated. However, these methods estimate one or a set of affine matrices based on fixed meshes, which is not flexible and accurate enough for scene with different depth variation. Actually, all videos contain parallax which will surely influence the video stabilization results. In this paper, we design a novel network, called PWStableNet, to calculate warping maps of the same size of input frames and warp each pixel to its desired position under the stabilized view.

III. The Proposed PWStableNet

Given a sequence of unstable video frames, PWStableNet is expected to estimate the jitter of the inter-frame motion and learn pixel-wise warping maps, with which we can transform each pixel in the shaky view to its stabilized view. In contrast to the previous methods which estimate a homography or a few homographies based on fixed meshes, the pixel-wise warping maps by PWStableNet can give a desirable individual position for each pixel in the stabilized view and well handle parallax and even discontinuous depth variation among different objects.

PWStableNet is built upon a multi-stage cascade encoder-decoder architecture. To process an unstable frame I_t with the width W and the height H, the input of PWStableNet is a sequence of consecutive unstable frames $S_t = \{I_{t-\omega}, \ldots, I_t, \ldots, I_{t+\omega}\}$, where $2\omega + 1$ is the length of the considered temporal window. For the proposed PWStableNet with n stages, the output at the n-th stage is two warping maps with the same size of the input frame I_t^n, $T_{t,x}^n$ and $T_{t,y}^n$. With these two warping maps, we transform I_t to the stabilized frame $\hat{I}_t^n$. In the training procedure, the warping maps and stabilized frames of all stages are taken to supervise our cascade network. However, for testing, we only utilize the warping maps obtained at the last stage. The motivation and analysis of this operation will be presented in details in the experiment section. Here we introduce how the stabilized frames are generated with the obtained warping maps. At the n-th stage, for each pixel (i, j) ($\{i = 1, 2, \cdots, W; j = 1, 2, \cdots, H\}$) of the stabilized frame $\hat{I}_t^n$, we get its corresponding values in $T_{t,x}^n$ and $T_{t,y}^n$. Specifically, $x_0 = T_{t,x}^n(i, j)$ indicates the horizontal coordinate of the pixel in the original unstable frame I_t and should be warped to (i, j) in the stabilized frame $\hat{I}_t^n$. Similarly $y_0 = T_{t,y}^n(i, j)$ indicates the vertical coordinate of the same pixel in I_t. Since the values of $T_{t,x}^n$ and $T_{t,y}^n$ are not necessarily integers, the value of each pixel, $\hat{I}_t^n(i, j)$, is usually generated by the four nearest pixels, i.e., $I_t(\lfloor x_0 \rfloor, \lfloor y_0 \rfloor)$, $I_t(\lfloor x_0 \rfloor, \lceil y_0 \rceil)$, $I_t(\lceil x_0 \rceil, \lfloor y_0 \rfloor)$, $I_t(\lceil x_0 \rceil, \lceil y_0 \rceil)$, with bilinear interpolation, where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ stands for the flooring and ceiling operations, respectively. The procedure is also shown in Fig. 2. The training data, network architecture and loss function will be introduced in the following subsections.

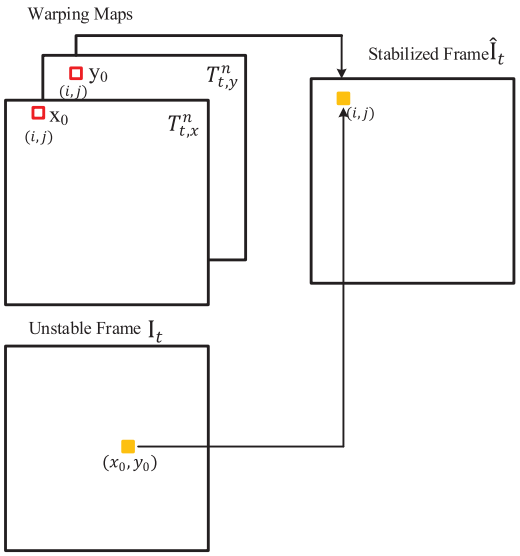


Fig. 2. Generation of stabilized frames with warping maps. In order to get the value of pixel (i, j) in the stabilized frame $\hat{I}_t^n$, we assemble $T_{t,x}^n(i, j)$ and $T_{t,y}^n(i, j)$ into (x_0, y_0), and calculate $\hat{I}_t^n(i, j)$ with the nearest pixels of (x_0, y_0) in the unstable frame I_t.

A. Training Data and Pre-Processing

The dataset to train our network is generated by Wang *et al.* [21]. This dataset contains 61 stable and unstable video pairs with outdoor scenes, including roads, buildings and vegetation. Each pair of videos is captured by two portable GoPro Hero 4 Black cameras and a hand-held stabilizer. We extract each frame from videos in the dataset and resize

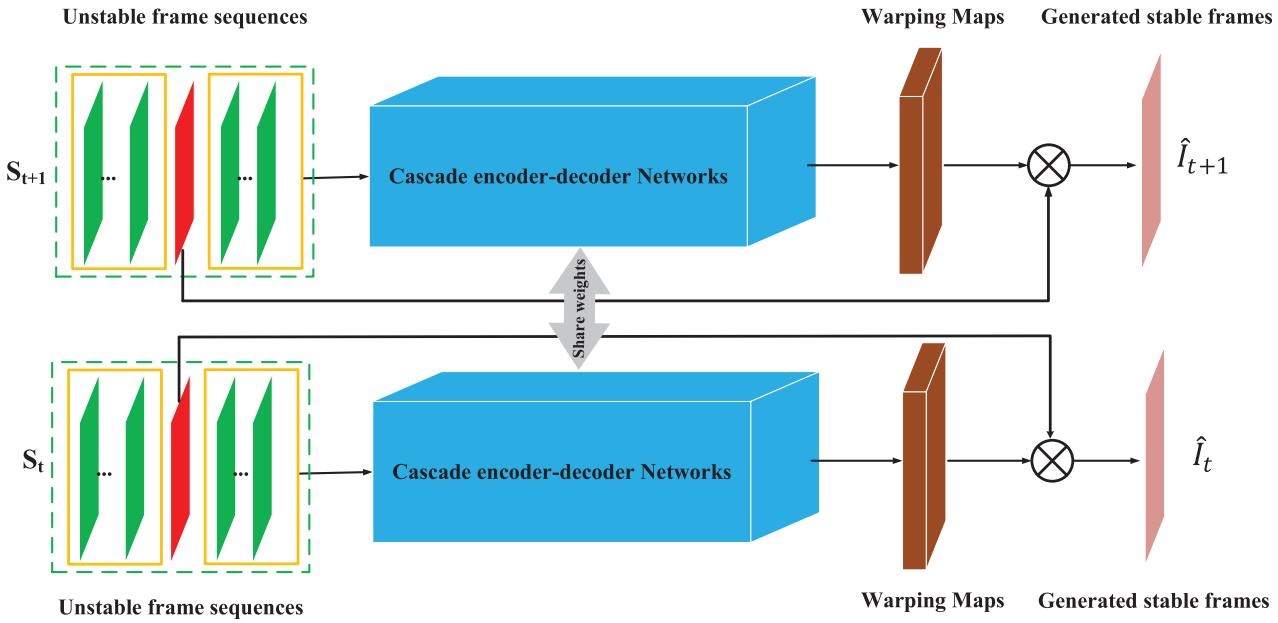


Fig. 3. The architecture of PWStableNet, which is based on the siamese network that has two branches to share the network parameters. During the training procedure, two consecutive sequences of temporal successive unstable frames are fed into the two branches as input, and two sets of warping maps are estimated. For testing, only one branch is used.

each frame to the size of 256×256. It is also converted to gray-scale. Pixel values and image axes are normalized to the range of $[-1,1]$ before being fed into the network.

B. Network Architecture and Loss Function

The architecture of PWStableNet is shown in Fig. 3. It is based on the siamese network that has two branches sharing the same parameters. This siamese architecture can ensure the temporal consistency of successive stabilized frames, and significantly improve the stability of generated videos. For testing, only one branch is used to produce the stabilized videos. We also propose a method to estimate and crop the invalid borders automatically in the post-processing procedure.

By Fig. 3, there are two branches in the siamese network. Each branch is made up of multiple stages. Since the structures of these multiple stages are similar, we first introduce the details of one stage in Fig. 4 and then introduce how these stages are cascaded into a multi-stage structure in Fig. 5. As shown in Fig. 4, each stage is mainly an encoder-decoder framework being similar to the architecture of U-Net [38]. The input is the resized frame sequence with the size of $(2\omega + 1) \times W \times H$. Its encoder is composed of several *conv* layers which produce smaller feature maps with more channels at deeper layers. Its decoder is composed of *deconv* layers which have the same sizes and the same channel numbers as the corresponding *conv* layers of the encoder and are connected with corresponding *conv* layers through skip connection. The stage in Fig. 4 generates two warping maps, including T_0 and T_1, which are of the same size.

T_0 is inspired by the STN modules in [20], which are designed to regress the inter-layer transformations and multiply these inter-layer transformations into the final transformation. In contrast, we take a similar module to roughly estimate the pixel-wise relationship between the unstable and stabilized frames. In Fig. 4, the generation of T_0 depends on an affine transformation H_t, which is produced by a down-sampling *conv* layer and a 1×1 *conv* layer after the last layer of the encoder. H_t is multiplied with a constant matrix $A \in \mathbb{R}^{W \times H \times 3}$, which is specified as

$$\begin{cases} A(i,j,1) = 2 * i/W - 1 \\ A(i,j,2) = 2 * j/H - 1 \\ A(i,j,3) = 1 \end{cases}, \quad i = 1, \cdots, W; \; j = 1, \cdots, H. \tag{1}$$

H_t and A produce T_0, which is made up of T_0^x and T_0^y and computed as

$$\begin{aligned} T_0^x(i,j) &= \sum_{n=1,2,3} H_t(1,n) * A(i,j,n), \\ T_0^y(i,j) &= \sum_{n=1,2,3} H_t(2,n) * A(i,j,n), \\ & \quad i = 1, \cdots, W; \; j = 1, \cdots, H. \end{aligned} \tag{2}$$

In Fig. 4, the first output T_0 is only a rough estimate of the motion between unstable and stabilized frames because of its limited freedom. For example, T_0 cannot even characterize the perspective transformation. So we need the second output T_1, which is a more precise pixel-wise warping map and generated by passing the output of the decoder through a *conv* layer with $stride = 1$. T_1 is expected to recover the residual motion which is not caught by T_0. T_0 and T_1 are added to produce the final warping map T, which generates the stabilized frames.

The above combination of T_0 and T_1 is motivated by decomposing the motion between unstable and stabilized frames into the global motion and the pixel-wise residual motion. That global motion can be efficiently and easily learned by

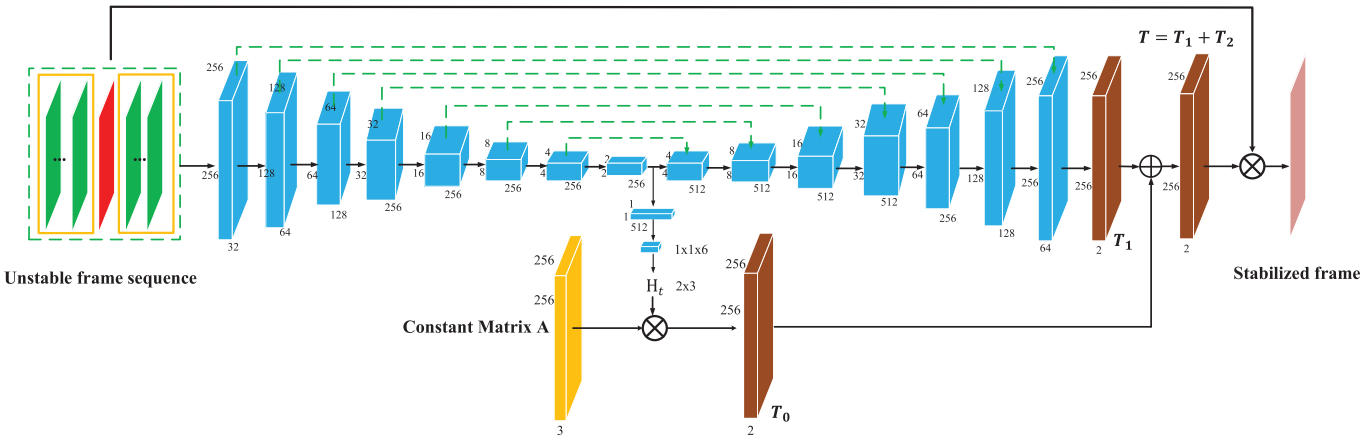


Fig. 4. The detailed architecture of one stage in one branch of PWStableNet. The input is the unstable frame sequence. The learned warping maps are made up of two parts. One is rigid warping maps obtained by the learned affine transformation, the other is the pixel-wise warping maps. The stabilized frame is generated by the sum of these two parts.

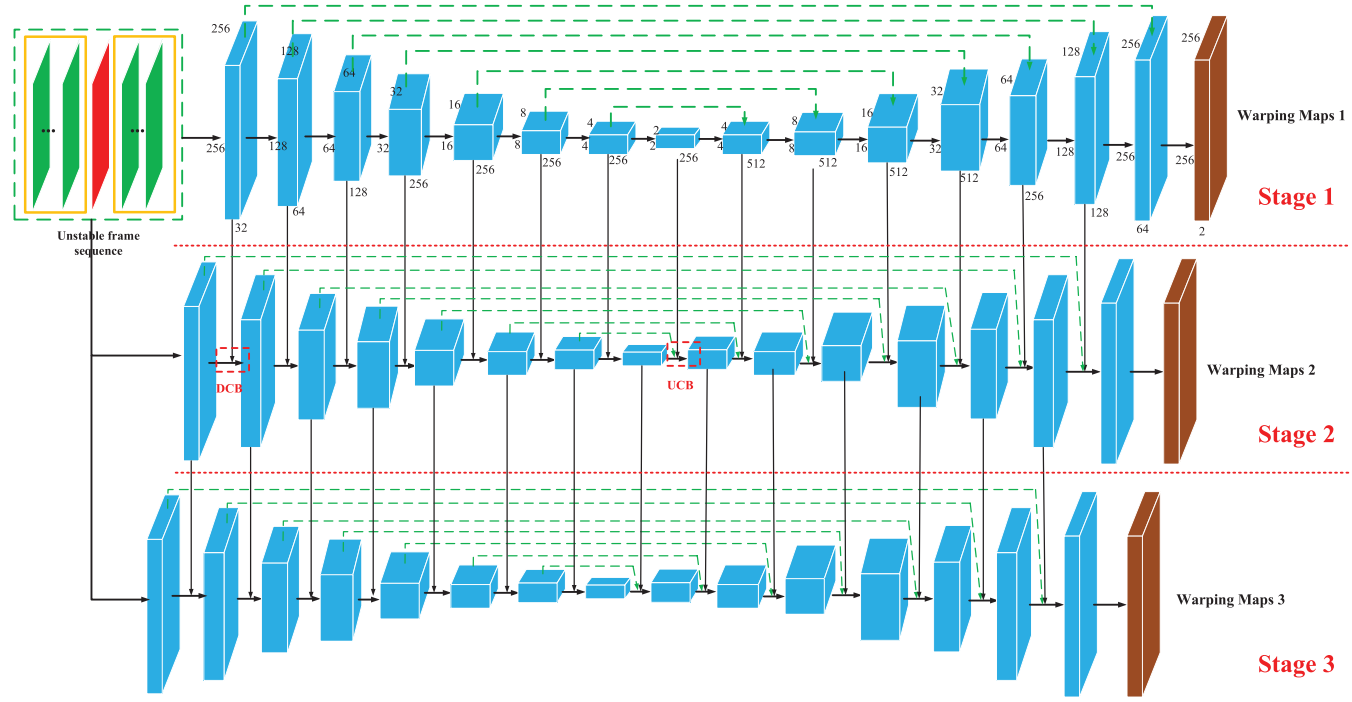
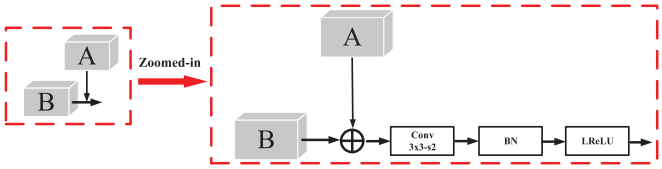


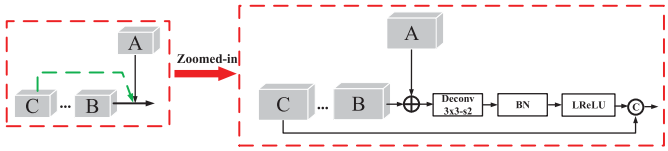
Fig. 5. The cascade architecture of PWStableNet. Here we omit the branch to generate T_0 at each stage for briefness. The inter-stage connection is indicated by red dotted rectangles, whose detailed structures are shown in Fig. 6.

T_0 through the *global* affine transformation H_t. The residual motion mainly results from parallax and is estimated by T_1. Based on the above motion decomposition, the residual motion is not expected to be large and can be precisely and efficiently learned by T_1 without much distortion. Moreover, the training loss function in Eq (3) is designed to enforce local rigidity in T_1 and further avoid distortion.

The overall multi-stage cascade architecture of PWStableNet is shown in Fig. 5. For briefness, the blocks to generate T_0 are not shown in Fig. 5. The first stage, *i.e.*,, Stage 1, takes exactly the same architecture as Fig. 4. However, the latter stages, including Stage 2 and Stage 3, take a slightly different architecture from the first stage by receiving feature maps from their former stages. The inter-stage connection is represented by arrows in Fig. 5. There are two types of inter-stage connection blocks, including the down-connection block (DCB) in Fig. 6(a) and the up-connection block (UCB) in Fig. 6(b). DCB first adds the feature maps of the former stage to the corresponding feature maps of the current stage with a shortcut connection, and then down-samples them with a Conv-BatchNorm-LeakyRelu module. Similarly UCB adds the feature maps of the former stage to the corresponding feature maps of the current stage, and then dilates the feature maps twice with a Deconv-BatchNorm-LeakyRelu module. UCB finally takes a skip connection to concatenate corresponding feature maps at



(a) Structure of the down-connection block (DCB).



(b) Structure of the up-connection block (UCB).

Fig. 6. Two inter-stage connection blocks.

the down-sampling layer. DCB and UCB enable the layers of the current stage to learn the residual feature maps of its former stage. Based on the architecture in Fig. 5, latter stages can learn more abstract features of the input frame sequence and estimate the transformation more precisely.

At stage n, the output of the proposed PWStableNet is the pixel-wise warping map T^n, which is made up of T_x^n and T_y^n. With T_x^n and T_y^n, we can transform the unstable frame I_t to its stabilized view $\hat{I}_t^n$. During the training procedure, the outputs of all three stages are used to calculate the loss function. We take the n-th stage as an example to define the loss function as follows.

$$L_{stab}^n = \sum_{i\in\{t-1,t\}} L_{content}(\hat{I}_i^n, \tilde{I}_i) + L_{shape}(T_i^n, \tilde{I}_i, I_i) + L_{temporal}(\hat{I}_{t-1}^n, \hat{I}_t^n), \quad (3)$$

where $L_{content}$ is the *content loss*, L_{shape} is the *shape loss* and $L_{temporal}$ is the *temporal loss*, whose details will be presented below. Then the loss functions of all stages are summed up as

$$L_{stab} = \sum_{n\in\{1,2,3\}} L_{stab}^n. \quad (4)$$

Now we define the loss terms in Eq (3). For notation simplicity, the superscript n is omitted here.

1) Content Loss: It measures how the stabilized frame aligns with the ground-truth stable frame and is defined as

$$L_{content}(\hat{I}_t, \tilde{I}_t) = \lambda_1 * L_{MSE}(\hat{I}_t, \tilde{I}_t) + \lambda_2 * L_{VGG}(\hat{I}_t, \tilde{I}_t), \quad (5)$$

where $\lambda_1 = 1$, $\lambda_2 = 1$ are constant weights.

Firstly, we push the stabilized frame to approximate its ground-truth through the mean squared error (MSE) loss, denoted as L_{MSE}. By [39], L_{MSE} may cause blurring and excessive smoothing, and can be alleviated by perceptual loss. Here the perceptual loss is defined as the VGG loss L_{VGG}, which comes from the max-pooling layer before the fully connection layers of the pre-trained VGG16 network [40]. With L_{VGG}, high-frequency parts of the stabilized frame can be well preserved.

2) Shape Loss: It aims to preserve the shapes of warped frames and avoid distortion, and is defined as

$$L_{shape}(T_t, \tilde{I}_t, I_t) = \lambda_3 * L_{feature}(T_t, \tilde{I}_t, I_t) + \lambda_4 * L_{grid}(T_t), \quad (6)$$

where $\lambda_3 = 1$, $\lambda_4 = 1$ are constant weights.

As we start our training process with randomly initialized weights, the warped frames and the ground-truth stable frames may not be well aligned and a long time will be spent to guide the warping maps towards the correct transformation. Since the estimated pixel-wise warping maps represent the pixel coordination mapping between unstable and stabilized frames, we require some sparse keypoints to be mapped to their desired locations under the given warping maps, which can accelerate the training process. Specifically we introduce the feature alignment term $L_{feature}$ as [21]. At frame t, the coordinates of a pair of matching feature points are denoted as $P_{i,t}$ and $\tilde{P}_{i,t}$, where $P_{i,t}$ is the one in the unstable frame I_t and $\tilde{P}_{i,t}$ is the one in the correspond ground-truth stable frame $\tilde{I}_t$. Based on the estimated warping maps T_x and T_y, we can calculate the corresponding position of feature point $\tilde{P}_{i,t}$ in the unstable frame as

$$\omega(\tilde{P}_{i,t}) = (T_x(\tilde{P}_{i,t}^x), T_y(\tilde{P}_{i,t}^y)), \quad (7)$$

where $\tilde{P}_{i,t}^x$ and $\tilde{P}_{i,t}^y$ are the horizontal and vertical components of $\tilde{P}_{i,t}$, respectively. Then $L_{feature}$ is computed as the average alignment error of matched feature points,

$$L_{feature}(T_t, \tilde{I}_t, I_t) = \frac{1}{m}\sum_{i=1}^{m} \| P_{i,t} - \omega(\tilde{P}_{i,t}) \|_2^2. \quad (8)$$

To compute the feature loss, we extract the SIFT features [41] for each pair feature points in the stable and unstable frames. In order to further avoid mismatching between feature points in the two frames, the spectral matching technique [42] and RANSAC algorithm [43] are implemented. Note that the above feature extraction and feature matching are only performed during training the network and are not required for testing.

As our network learns pixel-wise warping maps, it have much higher degree of freedom than networks which just calculate one or several homographies (or affine matrices). To avoid distortion, we place an authentic constraint on the shape of warped frames in Eq (6), i.e., the pixel-wise warping transformation in the same area is required to be similar. More specifically, inspired by [2], we require that the global warping is close to a homography and allow more flexibility in local regions. Each time we sample a region $[x_r, y_r, w_r, h_r]$ in T_x and T_y, where (x_r, y_r) is the coordinate of the left top point of that region and w_r and h_r stand for the width and height of that region, respectively. x_r, y_r, w_r, and h_r are randomly chosen in some appropriate ranges. One region example is shown in Fig. 7. K sampling points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_K, y_K)\}$ are uniformly picked up in region $[x_r, y_r, w_r, h_r]$ and their corresponding values in T_x, T_y are denoted as $\{(x_1^t, y_1^t), (x_2^t, y_2^t), \ldots, (x_K^t, y_K^t)\}$. We calculate the homography H by solving the following overdetermined

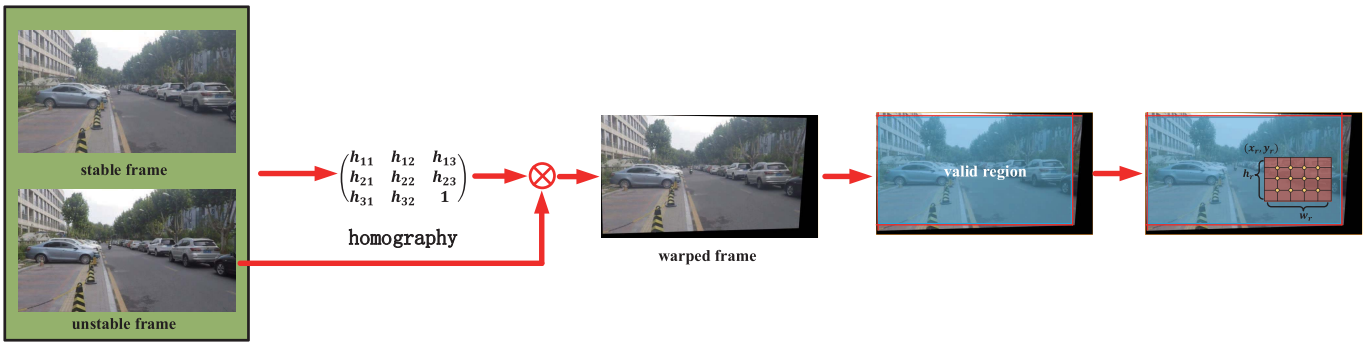


Fig. 7. The procedure of sampling random regions for L_{grid}. Given a pair of stable and unstable frames, we first calculate the homography between them and warp the unstable frame to its stabilized view. The valid region is defined as the inscribed rectangle of the stabilized frame. A region $[x_r, y_r, w_r, h_r]$ is randomly sampled inside the valid region.

equation,

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1^t x_1 & -x_1^t y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1^t x_1 & -y_1^t y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2^t x_2 & -x_2^t y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2^t x_2 & -y_2^t y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_K & y_K & 1 & 0 & 0 & 0 & -x_K^t x_K & -x_K^t y_K \\ 0 & 0 & 0 & x_K & y_K & 1 & -y_K^t x_K & -y_K^t y_K \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}}_{\beta}$$

$$= \underbrace{\begin{bmatrix} x_1^t \\ y_1^t \\ x_2^t \\ y_2^t \\ \vdots \\ x_K^t \\ y_K^t \end{bmatrix}}_{B}, \tag{9}$$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}. \tag{10}$$

Eq (9) can be solved with the least square method. Eq (9) is represented as $A\beta = B$ and its approximate solution is $\beta = (A^T A)^{-1} A^T B$ with the residual error of $\|A\beta - B\|^2$. The final loss is defined as

$$\begin{aligned} L_{grid} &= \gamma * \|A\beta - B\|_2^2 \\ &= \gamma * \|A(A^T A)^{-1} A^T B - B\|_2^2, \end{aligned}$$

where $\gamma = e^{min(\frac{w_r}{2W}, \frac{h_r}{2H})}$. For a small region, i.e., w_r and h_r are small, γ places a weaker constraint and allows more flexibility.

Since the warping map T_0 is calculated as an affine transformation, its rigidity can be guaranteed. So L_{grid} is mainly utilized to constrain the pixel-wise warping map T_1.

3) Temporal Loss: The motivation of this *temporal loss* is to require the smoothness of adjacent stabilized frames. We build our network with a siamese network which takes adjacent unstable frames as input and shares weights among its two branches. Denote the sequence of temporally successive unstable frames as $S_t = \{I_{t-\omega}, \ldots, I_t, \ldots, I_{t+\omega}\}$. At time t, S_{t-1} and S_t are fed into the two branches of the PWStableNet and two successive stabilized frames, $\hat{I}_{t-1}$ and $\hat{I}_t$, are generated. As low-frequency motion dominates in stable videos, adjacent stabilized frames should vary slowly. Then the temporal loss is defined as the mean square error between the adjacent stabilized frames,

$$L_{temporal}(\hat{I}_{t-1}, \hat{I}_t) = \lambda_5 * \frac{1}{WH} \|\hat{I}_t - \Psi(\hat{I}_{t-1})\|_2^2, \tag{11}$$

where $\Psi(\cdot)$ warps the stabilized frame $\hat{I}_{t-1}$ to the stabilized frame $\hat{I}_t$ according to the pre-computed optical flow with TV-L1 algorithm [44] and $\lambda_5 = 10$ is a constant.

C. Adversary With a Discriminator

To further improve the low-frequency motion of long-term sequences of stabilized frames, we can make our PWStableNet to be adversarial with a discriminator and achieve a little better performance. The discriminator has two inputs, including the real sequences of ground-truth stable frames $A_t^R = \{\tilde{I}_{t-\omega}, \ldots, \tilde{I}_t, \ldots, \tilde{I}_{t+\omega}\}$, and the fake sequences ground-truth stable frames $A_t^F = \{\tilde{I}_{t-\omega}, \ldots, \tilde{I}_{t-1}, \hat{I}_t, \tilde{I}_{t+1}, \cdots, \tilde{I}_{t+\omega}\}$ where $\tilde{I}_t$ is particularly replaced with the generated frame $\hat{I}_t$. We use *Patch*GAN for the discriminator, whose penalty is mainly placed at the scale of patches. We divide the generated frame into $N \times N$ patches and adopt a fully convolution network (FCN) to classify whether each patch is real (stable) or fake (unstable). The loss function of the discriminator is defined as

$$L_D = \sum_{i \in (t-1,t)} \frac{1}{2N^2} (\|D(A_t^R) - \mathbb{R}\|_2^2 + \|D(A_t^F) - \mathbb{F}\|_2^2), \tag{12}$$

where $\mathbb{R}$ and $\mathbb{F}$ are matrices whose all elements are set as 1 and -1, respectively. Being adversarial with this discriminator, the overall loss of the generator is finally chosen as

$$L_G = L_{stab} + \gamma \sum_{i \in (t-1,t)} \|D(A_t^F) - \mathbb{R}\|_2^2, \tag{13}$$

where $\gamma = 1$ is a constant weight.

D. Implementation Details

We trained our PWStableNet on *DeepStab* dataset which contains 61 pairs of stable and unstable videos. These video pairs are split into 45 training pairs, 8 validation pairs and 8 testing pairs. Some video thumbnails are shown in Fig. 8. To train our PWStableNet, we resize all videos to the spatial dimension of $W = 256$ and $H = 256$. Weights are initialized according to a normal distribution (μ is 0 and σ is 0.02). Adam optimizer is used for our PWStableNet with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We use the mini-batch size of 8 and train our network for 30 epochs. The initial learning rate is set to 0.002 and multiplied by 0.1 every 10 epochs. In the training process, $2\omega + 1$ successive unstable frames are fed into PWStableNet. Similar to [21], ω is set at 15 to handle the jitters of different frequencies within one second. In testing, the first and last frames are repeated for ω times and added to the head and tail of videos, respectively. This repeating operation will help in generating the first and last ω stabilized frames. For online video stabilization, this operation means ω frames are required as latency. The whole training process takes about 30 hours on two NVIDIA GTX 1080Ti graphics cards.

E. Post-Processing

Our PWStableNet generates stabilized frames with the estimated pixel-wise warping maps. As the final step, these stabilized frames have to be cropped into a common region to avoid empty regions or holes. It is not an easy task. Here we propose a simple but effective method to automatically determine the cropping region, which is characterized by its left top vertex and its right bottom vertex with the coordinates of (x_s, y_s) and (x_e, y_e), respectively. We will first calculate a homography at each frame, then determine the cropping region with all homographies.

At each frame, the coordinates of pixels in the stabilized frame are denoted as $S = \{S_x, S_y\}$ in Eq (14). Their corresponding coordinates in the original unstable frame are represented by the obtained warping map $T = \{T_x, T_y\}$.

$$S_x = \begin{bmatrix} -1 & \frac{2-W}{W} & \cdots & \frac{W-2}{W} \\ -1 & \frac{2-W}{W} & \cdots & \frac{W-2}{W} \\ \vdots & \vdots & \ddots & \vdots \\ -1 & \frac{2-W}{W} & \cdots & \frac{W-2}{W} \end{bmatrix},$$

$$S_y = \begin{bmatrix} -1 & -1 & \cdots & -1 \\ \frac{2-H}{H} & \frac{2-H}{H} & \cdots & \frac{2-H}{H} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{H-2}{H} & \frac{H-2}{H} & \cdots & \frac{H-2}{H} \end{bmatrix}. \quad (14)$$

By taking p columns and q rows, S and T formulate the following $p \times q$ matrices,

$$\widehat{S} = (\widehat{S}_x, \widehat{S}_y), \quad (15)$$

$$\widehat{T} = (\widehat{T}_x, \widehat{T}_y), \quad (16)$$

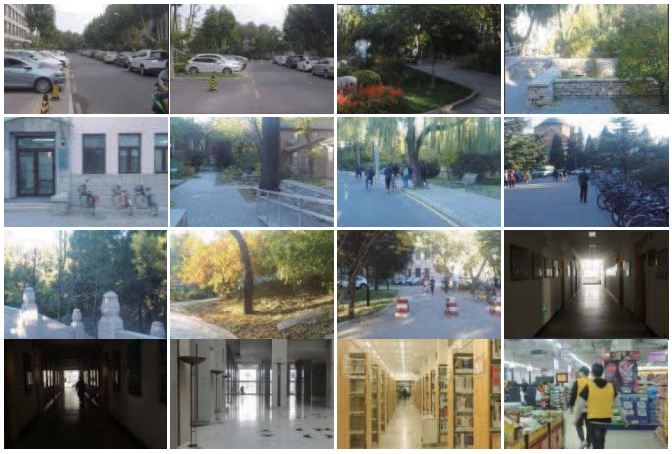


Fig. 8. Example video thumbnails of the *DeepStab* dataset.

$$\begin{aligned} \widehat{S}_x(i,j) &= S_x(i \times \lfloor \frac{W}{p} \rfloor, j \times \lfloor \frac{H}{q} \rfloor), \\ \widehat{S}_y(i,j) &= S_y(i \times \lfloor \frac{W}{p} \rfloor, j \times \lfloor \frac{H}{q} \rfloor), \\ \widehat{T}_x(i,j) &= T_x(i \times \lfloor \frac{W}{p} \rfloor, j \times \lfloor \frac{H}{q} \rfloor), \\ \widehat{T}_y(i,j) &= T_y(i \times \lfloor \frac{W}{p} \rfloor, j \times \lfloor \frac{H}{q} \rfloor), \\ & i = 1, \cdots, p; j = 1, \cdots, q, \end{aligned}$$

where $\lfloor \cdot \rfloor$ stands for the flooring operation. Both $\widehat{S}$ and $\widehat{T}$ consist of $p \times q$ pixels, whose coordinates in $\widehat{S}$ and $\widehat{T}$ are denoted as $\widehat{S}_k$ and $\widehat{T}_k$ ($k = 1, \cdots, p \times q$), respectively. With these pixels, a global homography is determined as

$$H = \underset{H}{\operatorname{argmin}} \sum_{k=1}^{p \times q} \| \widehat{S}_k - H * \widehat{T}_k \|, \quad (17)$$

where $H * \widehat{T}_k$ represents the transformed coordination of $\widehat{T}_k$ under the homography H, and $\| \cdot \|$ represents the Euclidean norm of coordinate vectors.

At frame t, the obtained H in Eq (17) is denoted as H_t and the four vertexes of the original unstable frame are $p^1 = (-1, -1)$, $p^2 = (1, -1)$, $p^3 = (-1, 1)$, $p^4 = (1, 1)$. Under H_t, the warped positions of these vertexes are $\hat{p}_t^1 = H_t * p^1$, $\hat{p}_t^2 = H_t * p^2$, $\hat{p}_t^3 = H_t * p^3$, $\hat{p}_t^4 = H_t * p^4$. Then the parameters of the expected cropping region are calculated as

$$\begin{aligned} x_s &= \max(-1, \max_{t \in (1,N)} p_t^1(x), \max_{t \in (1,N)} p_t^3(x)), \\ x_e &= \min(1, \min_{t \in (1,N)} p_t^2(x), \min_{t \in (1,N)} p_t^4(x)), \\ y_s &= \max(-1, \max_{t \in (1,N)} p_t^1(y), \max_{t \in (1,N)} p_t^2(y)), \\ y_e &= \min(1, \min_{t \in (1,N)} p_t^3(y), \min_{t \in (1,N)} p_t^4(y)), \end{aligned}$$

where N stands for the number of frames of the concerned video. Finally we crop all stabilized frames into the above region with the left top vertex (x_s, y_s) and the right bottom vertex (x_e, y_e).

TABLE I

ABLATION STUDY OF ARCHITECTURE. EACH ROW SHOWS THE STABILIZATION STATISTICS IN THREE CATEGORIES OF VIDEOS FROM [25], REGULAR, PARALLAX AND CROWD, IN THREE METRICS: CROPPING RATIO (C), DISTORTION VALUE (D) AND STABILITY SCORE (S)

Architecture	Regular			Parallax			Crowd			inference time (ms)
	C	D	S	C	D	S	C	D	S	
Stage-1 (1x)	0.57	0.70	0.53	0.51	0.61	0.64	0.67	0.56	0.63	6.1
Stage-2 (1x)	0.64	0.74	0.69	0.54	0.73	0.73	0.73	0.70	0.66	13.6
Stage-4 (1x)	0.67	0.78	0.81	0.63	0.75	0.77	0.77	0.78	0.75	22.6
Stage-1 (2x)	0.67	0.75	0.74	0.59	0.74	0.72	0.77	0.77	0.72	12.1
Stage-2 (2x)	0.68	0.79	0.76	0.63	0.76	0.76	0.78	0.77	0.73	24.4
Proposed (Stage 1)	0.66	0.76	0.78	0.61	0.74	0.76	0.74	0.76	0.76	17.8
Proposed (Stage 2)	0.67	0.77	0.79	0.63	0.75	0.77	0.76	0.77	0.76	17.8
Proposed (Stage 3)	0.69	0.79	0.80	0.64	0.76	0.78	0.77	0.78	0.77	17.8

IV. EXPERIMENTAL RESULTS

We train our PWStableNet on *DeepStab* dataset and test it on various videos of [25]. To quantitatively compare our network with previous methods, we calculate three objective metrics as [32], including *cropping ratio*, *distortion score* and *stability score*, which are briefly explained below.

Cropping ratio measures the remaining area after cropping away empty regions. Larger cropping ratio means more meaningful contents are preserved. For each pair of unstable and stable frames, a global homography is estimated and then the cropping ratio can be calculated with the scale components of this homography. We average the ratios of all frames for the final cropping ratio.

Distortion score is estimated from the affine part of homography. Similar to cropping ratio, the global homography is calculated for each pair of unstable and stable frames and the ratio of two largest eigenvalues of the affine part of the homography is extracted. Distortion score is defined as the smallest eigenvalue ratio among all frames of the whole video.

Stability score measures the smoothness of the stabilized videos. As [21], we segment each frame into 4×4 grids, and the vertex profiles between successive frames are viewed as 1D temporal signals for frequency domain analysis. The ratio of each profile is computed as the lowest frequency components (2nd to 6th) over the full frequencies by excluding the DC component. The stability score of the stabilized videos is the average ratio of all profiles.

A. Ablation Study

In order to evaluate the effectiveness of our proposed framework, we test the performance of the PWStableNet through changing the architecture of the network, the loss function and the input frame sequences. All experiments are executed on public stabilization dataset from [25] which consists of several video categories, including $Regular$, $Parallax$ and $Crowd$.

1) Network Architecture: PWStableNet is constructed with three-stage cascade encoder-decoder modules. To show the effectiveness of the proposed framework, we implement the variations using the same encoder-decoder module with different numbers of stages: one stage (Stage-1), two stages (Stage-2) and four stages (Stage-4). Same channels are retained at each convolutional layer. The results are listed in Table I, which shows more stages are beneficial for better results. However, more modules also mean more parameters to learn and slower processing speed. Four or more modules for our PWStableNet lead to insignificant performance improvement with inference time increasing. So we employ three stages for our frameworks. Moreover, we also change the channels of each convolutional layer. $2\times$ means that we double the channels of all layers . The corresponding results are also listed in Table I. Generally speaking, better performance can be achieved with more feature maps at each layer, but at the cost of low speed. The built cascade framework is more effective for generating stabilized videos. For the proposed three-stage cascade structure, we also list the performance of generating the stabilized frames with the estimated warping maps at each stage (Stage 1, Stage 2, Stage 3) in this table. Best results are achieved by Stage 3. This is quite reasonable because the latter stage learns the residual with respect to feature maps at former stage. The latter stage can learn more abstract features of the input frame sequence and regress more precise transformation. Therefore, we take the estimated warping maps at last stage to stabilize videos.

2) Loss Functions: Table II shows the ablation studies for different loss functions. We select several pivotal loss terms and train the network without them. Without L_{vgg}, the generated frames are blurred. However, this issue will not seriously influence the quantitative metrics since they mainly measure the stability performance. When L_{grid} is removed, the performance greatly degrades due to the lack of shape constraint for warping maps, which yields more distortion during generating stabilized frames. We observed that without $L_{temporal}$, the network training will not converge. When L_D are removed, our PWStableNet will not be adversarial with the discriminator and the stability declines a little because we lose long-term temporal supervision. However, since PWStableNet is based on a siamese network, it can constrain the inter-frame consistency of adjacent frames and ensure our video stabilization still works.

3) Input Variation: The *DeepStab* dataset is composed of videos containing jitters of different frequencies. The period of jitters caused by hand-held camera is usually within one second. In other words, we mainly focus on removing the jitter

TABLE II

ABLATION STUDY OF LOSS FUNCTION. EACH ROW SHOWS THE STABILIZATION STATISTICS IN THREE CATEGORIES OF VIDEOS FROM [25], REGULAR, PARALLAX AND CROWD, IN THREE METRICS: CROPPING RATIO (C), DISTORTION VALUE (D) AND STABILITY SCORE (S). "−" MEANS THE CORRESPONDING NETWORK DOES NOT CONVERGE

Loss function	Regular			Parallax			Crowd		
	C	D	S	C	D	S	C	D	S
w/o L_{vgg}	0.66	0.78	0.78	0.62	0.72	0.69	0.74	0.75	0.75
w/o L_{grid}	0.64	0.59	0.77	0.56	0.69	0.61	0.73	0.67	0.64
w/o $L_{temporal}$	-	-	-	-	-	-	-	-	-
w/o L_D	0.68	0.79	0.79	0.65	0.75	0.77	0.77	0.79	0.76
w/o T_1	0.67	0.80	0.78	0.63	0.75	0.76	0.76	0.78	0.74
Proposed	0.69	0.79	0.80	0.64	0.76	0.78	0.77	0.78	0.77

TABLE III

ABLATION STUDY OF INPUT VARIATIONS. EACH ROW SHOWS THE STABILIZATION STATISTICS IN THREE CATEGORIES OF VIDEOS FROM [25], REGULAR, PARALLAX AND CROWD, IN THREE METRICS: CROPPING RATIO (C), DISTORTION VALUE (D) AND STABILITY SCORE (S)

Input variation	Regular			Parallax			Crowd		
	C	D	S	C	D	S	C	D	S
$\omega = 3$	0.67	0.69	0.67	0.60	0.67	0.61	0.74	0.72	0.71
$\omega = 7$	0.65	0.74	0.76	0.61	0.69	0.69	0.73	0.75	0.73
$\omega = 11$	0.68	0.77	0.78	0.62	0.72	0.72	0.75	0.75	0.74
Proposed	0.69	0.79	0.80	0.64	0.76	0.78	0.77	0.78	0.77

higher than this frequency. Considering that the experimental videos play at 30 FPS, we set ω as 15. Different ω is set for training and tested with several videos. The results are shown in Table III. Under short temporal frame sequences, high frequencies can be removed with possible performance degradation. On the other hand, when ω is larger than 15, little performance improvement can be improved by further increasing ω.

B. Quantitative Evaluation

1) Comparison With State-of-the-Art Methods: We compare our PWStableNet with several traditional methods, including subspace [5], epipolar [9], bundled-paths [25], meshflow [32] through several public available videos in terms of the aforementioned objective metrics. The results are shown in Fig. 9 and left blank for videos whose results are not provided by authors. Compared with traditional offline stabilization methods, our PWStableNet achieves comparable performance and its stability scores are slightly lower on some videos, which mainly results from the fact that the irregular visual wobbling or other artifacts of these videos do not appear in our training videos. These issues can be resolved by training our PWStableNet with more videos. Nevertheless, as shown in Table IV, our PWStableNet is faster than most traditional stabilization methods and can work in real time.

2) Comparison With Some CNN-Based Methods: For a more comprehensive comparison, we compared our PWStableNet with two state-of-the-art CNN-based stabilization methods, including StabNet in [21] and the one in [20], through all types of test videos (including *Regular*, *QuickRotation*, *QuickZooming*, *Parallax*, *Running* and *Crowd*). The results of these methods are shown in Fig. 10.

TABLE IV

AVERAGE PROCESSING SPEED (FPS) OF DIFFERENT METHODS FOR TEST VIDEOS

Method	FPS
Bundled [25]	3.5
MeshFlow [32]	22.0
StabNet [21]	35.5
Xu *et al.* [20]	30.1
PWStableNet	56.2

It can be seen that the proposed PWStableNet achieves higher indices in most videos and particularly produces much better results than them on *Parallax* and *Crowd* videos. This performance superiority mainly results from the fact that our PWStableNet regresses pixel-wise warping maps to transform unstable frames to stabilized frames and is much more precise than simply estimating a global affine transformation or a set of transformations in [21] and [20]. *Parallax* and *Crowd* videos usually contain more parallax or discontinuous depth variation, which can not be described by a few affine matrices or homographies. The comparison results through *Parallax* and *Crowd* videos are particularly shown in Fig. 11 to illustrate the effectiveness of our PWStableNet.

In summary, our PWStableNet is an online stabilization method with several frames latency and can run much faster than the aforementioned traditional and CNN-based stabilization methods, which is confirmed by Table IV.

3) Stabilizing Low-Quality Videos: Another superiority of our PWStableNet to traditional stabilization methods is its robustness to low-quality videos caused by noise, motion blurring, etc. Here we also compare the performances of different

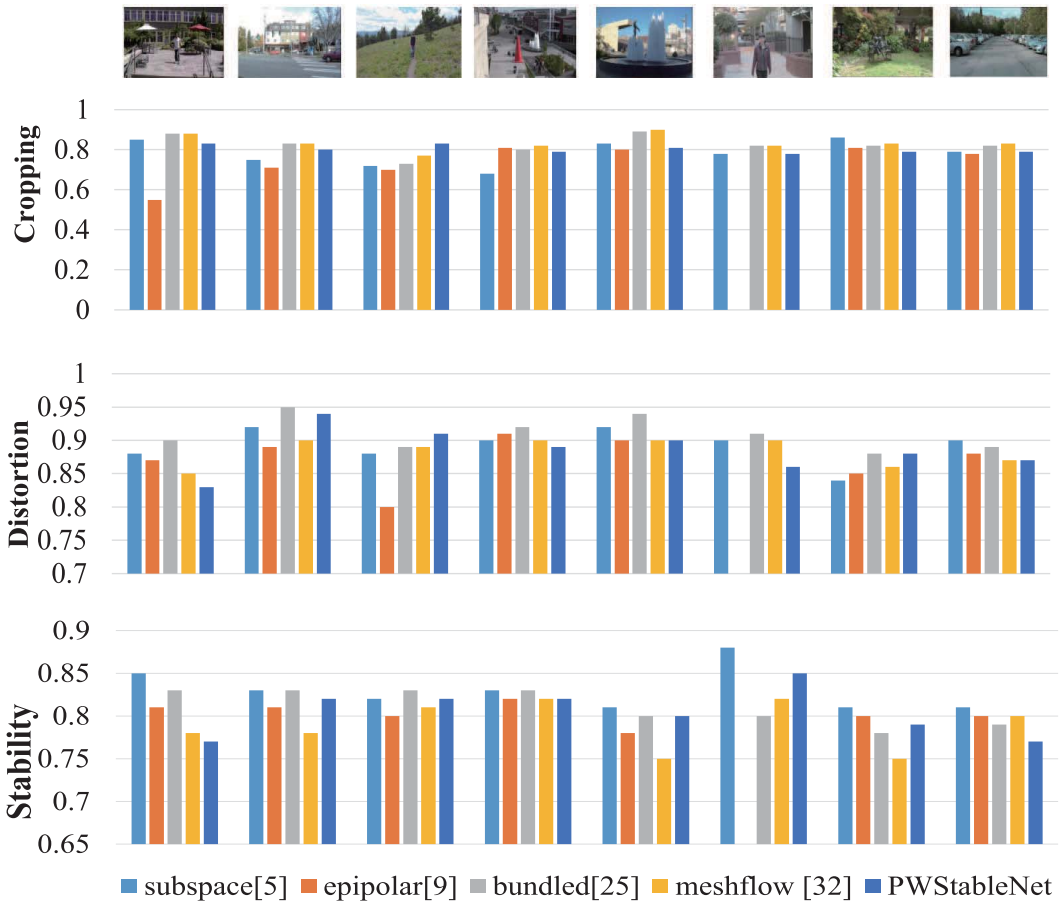


Fig. 9. Comparison with other traditional methods in terms of three metrics through several publicly available videos.

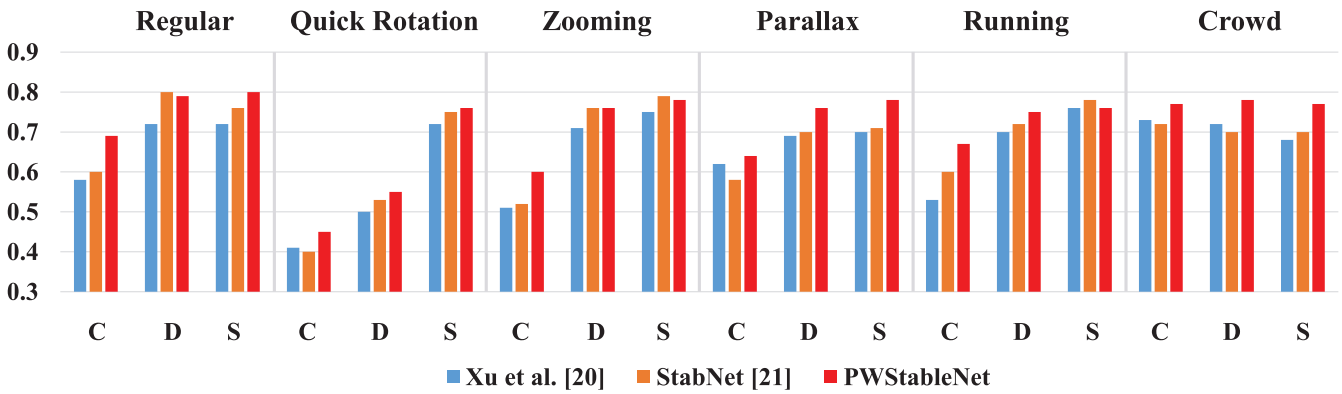


Fig. 10. Comparison with some CNN-based video stabilization methods in terms of three metrics through all six video categories [25].

methods through several low-quality videos, including blurry videos, noisy videos, watermarked videos and night-scene videos, whose thumbnails are shown in Fig. 12.

Blurry videos are generated due to the shakiness of hand-held cameras and this shakiness is harmful for extracting effective feature points. Noisy videos are produced by poor illumination and manually added Gaussian noise. Watermarked videos are those videos which are overlaid with logos or repetitive patterns. These interferences of watermark videos may also disturb feature matching from the original video contents. Night-scene videos are typically blurry and contain severe noise. Traditional methods may fail due to the feature matching difficulty.

To demonstrate the performance and robustness of the PWStableNet, we establish a $Low-quality$ category with 12 videos (3 videos of each type) and compare our method with a state-of-the-art traditional method, subspace [5], which is the stabilizer of Adobe Premiere. The results are shown in Fig. 13. Compared with the traditional method, PWStableNet can achieve more robust performance.

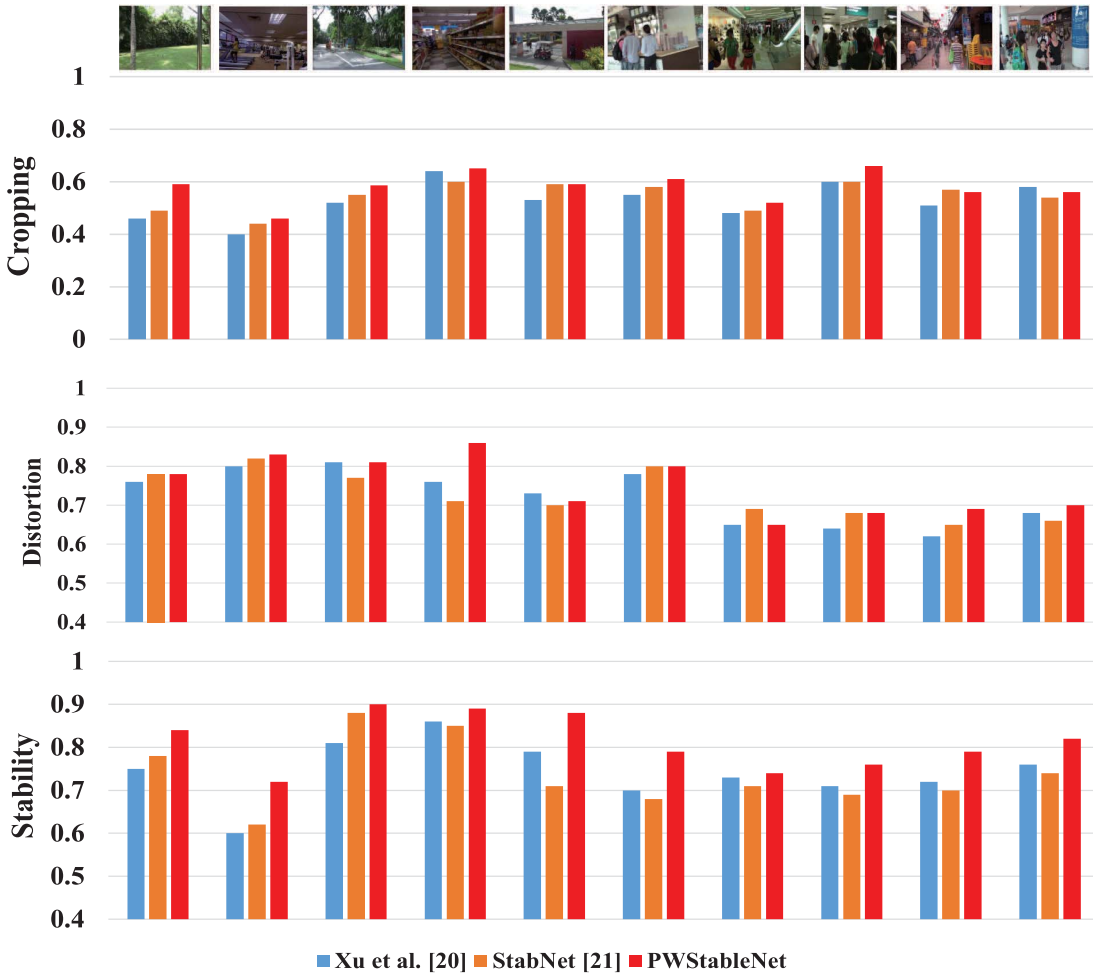


Fig. 11. Comparison with some CNN-based video stabilization methods in terms of three metrics through *Parallax* and *Crowd* videos. The first five videos are from *Parallax* and the last five videos are from *Crowd*.

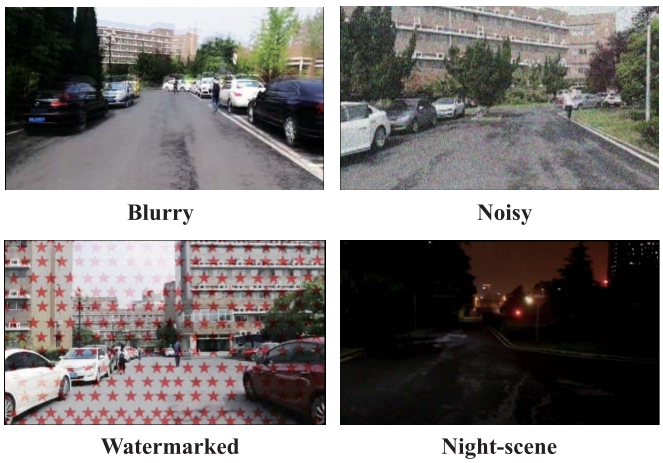


Fig. 12. Typical frames of four kinds of $low-quality$ videos.

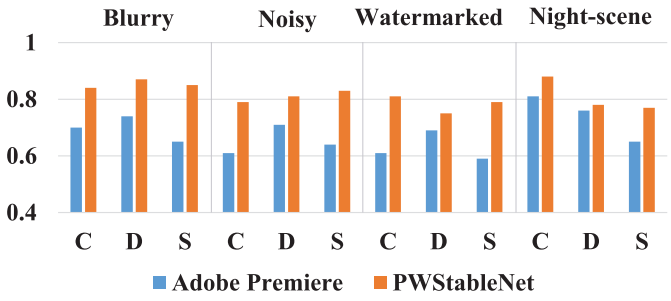


Fig. 13. Quantitative comparison between subspace(Adobe Premiere stabilizer) and PWStableNet through the $Low-quality$ video category.

Traditional methods may fail to stabilize some low-quality videos due to the nature of feature-based matching. For thorough comparison, we provide a supplemental demo at http://home.ustc.edu.cn/ zmd1992/PWStableNet.html.

We further conduct a user study with 20 participants to compare our PWStableNet with Adobe Premiere stabilizer which is based on the subspace stabilization method. A test set is the constructed $Low-quality$ video category with 12 videos. In the user study, every participant is asked to pick the better one from the results of the two methods or mark them "indistinguishable". The original videos and their stabilized results are displayed in a random order and participants are unaware of the methods which generate the stabilized videos. We show the results in Fig. 14. It can be seen our PWStableNet can achieve much better performance

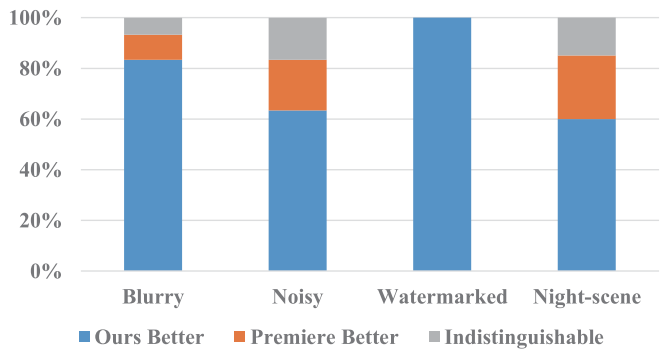


Fig. 14. User study results by comparing our PWStableNet with the Adobe premiere stabilizer.

than the Adobe Premiere stabilizer in these low-quality videos. Moreover, the Adobe Premiere stabilizer failed to stabilize some watermarked and night-scene videos due to its feature matching failure. These experimental results again confirm the robustness of our PWStableNet.

V. Limitation

Our PWStableNet has its own limitations. Firstly, since the training dataset is collected with all videos captured by hand-held camera, the trained network can mainly handle the jitters within the frequency and amplitude caused by human videographers. The proposed network may suffer performance degradation when processing the movement patterns which have not been trained before, especially in the existence of large quick motion or strong parallax. The same problem also exists in some CNN-based video stabilization methods. Although this problem can be mitigated by increasing λ_4, the weight of the rigidity loss L_{grid}, in Eq (6), too large λ_4 may disturb the training of the proposed network, or even yield training failure. In the future, more efficient methods will be pursued to handle the problem.

Secondly, we propose a simple but effective method to determine the cropping region. However, this cropping method may be influenced by seriously inaccurate estimation in some rare cases. So an inspection of stabilized videos is necessary to detect possible over-cropping or under-cropping. Since our PWStableNet learns pixel-wise warping maps to transform frames, it has more freedom than traditional methods, which warp unstable frames with several homograpies in fixed meshes, and can expect better performance than traditional methods. The bottleneck of our PWStableNet lies in the capacity of its training dataset. We can expect better performance through training with more videos in various situations.

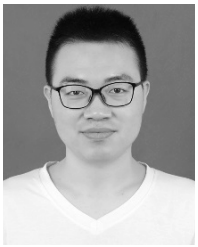
VI. Conclusion

In this paper, we proposed a CNN-based method to solve the traditional video stabilization problem. The proposed method, called PWStableNet, estimates pixel-wise warping maps to transform the unstable frames into their stabilized views. Since PWStableNet stabilizes unstable frames with pixel-wise transformation, we can get more precise transformation for each pixel than just calculating one or several homographies (or affine matrices), especially for videos with strong parallax. PWStableNet is built upon a multi-stage cascade encoder-decoder network, in which a latter stage learns the residual feature maps with respect to the corresponding feature maps of its former stage. This cascade architecture can produce more precise results at latter stages. We train our PWStableNet on a public dataset with the well-designed loss function. Experiments demonstrate that our PWStableNet achieves comparable performance with several feature-based state-of-the-art methods and better results than some CNN-based stabilization methods, especially in the situation of significant depth variation. Furthermore, the proposed PWStableNet has stronger robustness and higher processing speed than most previous methods.

References

[1] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 63:1–63:9, Jul. 2015.
[2] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graph.*, vol. 28, no. 3, p. 44, 2009.
[3] Q. Ling, S. Deng, F. Li, Q. Huang, and X. Li, "A feedback–based robust video stabilization method for traffic videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 3, pp. 561–572, Mar. 2018.
[4] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1150–1163, Jul. 2006.
[5] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Trans. Graph.*, vol. 30, no. 1, p. 4, 2011.
[6] Y. J. Koh, C. Lee, and C.-S. Kim, "Video stabilization based on feature trajectory augmentation and selection and robust mesh grid warping," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5260–5273, Dec. 2015.
[7] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012.
[8] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 225–232.
[9] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Trans. Graph.*, vol. 31, no. 5, p. 126, 2012.
[10] S. Liu, L. Yuan, P. Tan, and J. Sun, "SteadyFlow: Spatially smooth optical flow for video stabilization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 4209–4216.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[14] R. Girshick, "Fast R–CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
[17] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R–CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.
[18] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 341–349.
[19] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep video deblurring for hand–held cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1279–1288.
[20] S. Xu, J. Hu, M. Wang, T. Mu, and S. Hu, "Deep video stabilization using adversarial networks," *Comput. Graph. Forum*, vol. 37, no. 7, pp. 267–276, 2018.

[21] M. Wang *et al.*, "Deep online video stabilization with multi–grid warping transformation learning," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2283–2292, May 2019.
[22] B. Chen, K. Lee, W. Huang, and J. Lin, "Capturing intention-based full-frame video stabilization," *Comput. Graph. Forum*, vol. 27, no. 7, pp. 1805–1814, Oct. 2008.
[23] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camerawork of casual video," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 5, no. 1, pp. 1–28, Oct. 2008.
[24] J. Yang, D. Schonfeld, and M. Mohamed, "Robust video stabilization based on particle filter tracking of projected camera motion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 945–954, Jul. 2009.
[25] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Trans. Graph.*, vol. 32, no. 4, p. 1, Jul. 2013.
[26] L. Zhang, Q.-Z. Zheng, and H. Huang, "Intrinsic motion stability assessment for video stabilization," *IEEE Trans. Visual. Comput. Graph.*, vol. 25, no. 4, pp. 1681–1692, Apr. 2019.
[27] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[28] C. Buehler, M. Bosse, and L. Mcmillan, "Non-metric image-based rendering for video stabilization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2005, p. 2.
[29] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala, "Light field video stabilization," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 341–348.
[30] M. Irani, "Multi-frame correspondence estimation using subspace constraints," *Int. J. Comput. Vis.*, vol. 48, no. 3, pp. 173–194, 2002.
[31] Y.-S. Wang, F. Liu, P.-S. Hsu, and T.-Y. Lee, "Spatially and temporally optimized video stabilization," *IEEE Trans. Visual. Comput. Graph.*, vol. 19, no. 8, pp. 1354–1361, Aug. 2013.
[32] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "Meshflow: Minimum latency online video stabilization," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 800–815.
[33] M. Zhao, S. Deng, and Q. Ling, "A fast traffic video stabilization method based on trajectory derivatives," *IEEE Access*, vol. 7, pp. 13422–13432, 2019.
[34] Q. Ling and M. Zhao, "Stabilization of traffic videos based on both foreground and background feature trajectories," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2215–2228, Aug. 2019.
[35] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.
[36] Y. Nakajima and H. Saito, "Robust camera pose estimation by viewpoint classification using deep learning," *Comp. Vis. Media*, vol. 3, no. 2, pp. 189–198, Jun. 2017.
[37] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4463–4471.
[38] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput.-Assist. Intervent.* Munich, Germany: Springer, 2015, pp. 234–241.
[39] C. Ledig *et al.*, "Photo–realistic single image super–resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[41] D. G. Lowe, "Distinctive image features from scale–invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[42] J. Tang, L. Shao, X. Li, and K. Lu, "A local structural descriptor for image matching via normalized graph Laplacian embedding," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 410–420, Feb. 2016.
[43] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[44] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TV–L1 optical flow estimation," *Image Process.*, vol. 3, pp. 137–150, Jul. 2013.



Minda Zhao received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include computer vision, image processing, and machine learning.

Qiang Ling (Senior Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1997, the M.E. degree from Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree from the University of Notre Dame, Notre Dame, IN, USA, in 2005. He was a Research Staff Member with Seagate Technology from 2005 to 2008. He joined the University of Science and Technology of China in 2008. He is currently a Professor with the Department of Automation, University of Science and Technology of China. His research interests include networked control systems and image processing. He is also serving as an Associate Editor on the IEEE Control Systems Society Conference Editorial Board.