Deep Online Video Stabilization With Multi-Grid Warping Transformation Learning

Miao Wang, *Member, IEEE*, Guo-Ye Yang, *Member, IEEE*, Jin-Kun Lin, *Member, IEEE*, Song-Hai Zhang, *Member, IEEE*, Ariel Shamir, *Member, IEEE*, Shao-Ping Lu, *Member, IEEE*, and Shi-Min Hu, *Senior Member, IEEE*

***Abstract*— Video stabilization techniques are essential for most hand-held captured videos due to high-frequency shakes. Several 2D-, 2.5D-, and 3D-based stabilization techniques have been presented previously, but to the best of our knowledge, no solutions based on deep neural networks had been proposed to date. The main reason for this omission is shortage in training data as well as the challenge of modeling the problem using neural networks. In this paper, we present a video stabilization technique using a convolutional neural network. Previous works usually propose an off-line algorithm that smoothes a holistic camera path based on feature matching. Instead, we focus on low-latency, real-time camera path smoothing that does not explicitly represent the camera path and does not use future frames. Our neural network model, called *StabNet*, learns a set of mesh-grid transformations progressively for each input frame from the previous set of stabilized camera frames and creates stable corresponding latent camera paths implicitly. To train the network, we collect a dataset of synchronized steady and unsteady video pairs via a specially designed hand-held hardware. Experimental results show that our proposed online method performs comparatively to the traditional off-line video stabilization methods without using future frames while running about 10 times faster. More importantly, our proposed *StabNet* is able to handle low-quality videos, such as night-scene videos, watermarked videos, blurry videos, and noisy videos, where the existing methods fail in feature extraction or matching.**

***Index Terms*— Video stabilization, video processing.**

Manuscript received March 14, 2018; revised August 30, 2018 and October 21, 2018; accepted November 19, 2018. Date of publication November 30, 2018; date of current version January 30, 2019. This work was supported in part by the National Natural Science Foundation of China under Project 61561146393 and Project 61521002, in part by the China Postdoctoral Science Foundation under Project 2016M601032, in part by the Research Grant of the Beijing Higher Institution Engineering Research Center, and in part by the Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. The work of A. Shamir was supported by the Israel Science Foundation as part of the ISFNSFC Joint Program under Project 2216/15. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chia-Kai Liang. *(Corresponding author: Song-Hai Zhang.)*

M. Wang is with Beihang University, Beijing 100083, China.

G.-Y. Yang, J.-K. Lin, and S.-H. Zhang are with Tsinghua University, Beijing 100084, China (e-mail: shz@tsinghua.edu.cn).

A. Shamir is with the Interdisciplinary Center Herzliya, Herzliya 46150, Israel.

S.-P. Lu is with Nankai University, Tianjin 300071, China.

S.-M. Hu is with Tsinghua University, Beijing 100084, China, and also with Beihang University, Beijing 100084, China.

This paper has supplementary downloadable material at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TIP.2018.2884280

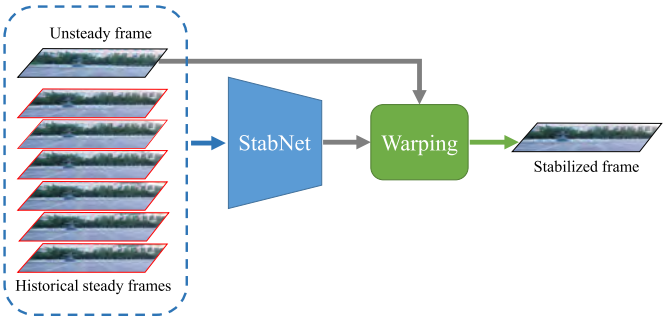


Fig. 1. Deep online video stabilization. We propose *StabNet*, a neural network that learns to predict transformations for each incoming unsteady frame, given the history of steady frames. Applying the predicted transformations to the original unsteady frame generates the stabilized output frame. The stabilized frames then act as historical frames for stabilizing the following unsteady frames.

I. Introduction

VIDEOS captured by hand-held camera are often not easy to watch due to shaky content. Several digital video stabilization techniques have been proposed in the past decade to improve the visual quality of hand-held videos, by removing high-frequency camera movements [1]–[5]. The majority of the proposed methods deal with this problem using a global view, by first estimating and then smoothing the camera path using offline computation. The very few online stabilization methods follow a 'capture→ compute→display' procedure for each incoming video frame in real time with low latency. Due to the real-time requirement in such methods, the camera motion is estimated by an Affine transformation, homography or using meshflow. In this paper, we focus on the online stabilization problem. Different from existing approaches, that must explicitly model the camera path to smooth it, we use a learning-based approach to directly compute a target steady transformation, with guidance from historical stabilized frames (see Figure 1).

In recent years, we have witnessed how convolutional neural networks (CNNs) changed Computer Vision and Computer Graphics fields. Methods that are based on CNNs perform more accurately and more efficiently. For example, several traditional video processing topics such as video stylization [6] and video deblurring [7] are re-addressed using CNNs. To our knowledge, there are no CNN-based methods published for digital video stabilization, although it is an important topic in video processing. We observed two main obstacles that prevent a CNN-based stabilization solution. First, the lack of

training data: pairs of steady and unsteady synchronized videos with an identical capturing route and content are required for training a CNN model. While this is not necessary for traditional methods, it is essential for a learning-based stabilization approach. Second, the challenge of correct problem definition: traditional stabilization methods compute and smooth a camera path, which cannot be easily adapted to a CNN-based solution. A somewhat different problem definition is required.

Based on these observations, we propose to solve the corresponding issues by creating a practical data set for training a neural network, and modifying the formulation of the problem by defining a progressive online stabilization algorithm. First, to collect training data, we captured synchronized hand-held steady/unsteady video pairs using a special hardware. We remodeled a hand-held stabilizer with two cameras, where only one camera is stabilized by the stabilizer while the other camera is fixed to the stabilizer grip, moving consistently with the hand motions. Second, in our modified formulation of the stabilization problem, instead of estimating and smoothing a virtual camera path, we learn transformations for spatially distributed regular mesh grids from each unsteady frame progressively along the time-line, and generate a steady output video in an online fashion.

We present *StabNet*, a CNN model to stabilize frames with light-weighted feed-forward operations through the network. The learning process is driven by the information of historically stabilized frames with the supervised ground-truth steady frame. Figure 1 shows the overview of our deep video stabilization. The proposed deep stabilization method performs comparably well on test videos collected from existing works. The main merit of our algorithm is the ability to run in real-time at 35.5 FPS with minumum latency (1 frame) on a NVIDIA GTX 1080Ti graphic card, being about $10\times$ faster than offline methods. More importantly, our method is superior to existing methods with the ability to handle low-quality videos, such as night-scene videos, watermarked videos, blurry videos and noisy videos, where existing feature-matching based methods may totally fail. To our knowledge, the proposed *StabNet* is a pioneer in using convolutional network for digital video stabilization.

We also built the *DeepStab* dataset consisting of pairs of synchronized steady/unsteady videos. We have released the dataset and believe that it will benefit the community for future research on stabilization using data-driven methods.

II. Related Work

Our work is closely related to digital video stabilization approaches and deep learning video manipulation.

A. Digital Video Stabilization

Existing *offline* stabilization techniques estimate the camera trajectory from 2D, 2.5D or 3D perspective and then synthesize a new smooth camera trajectory to remove the undesirable high-frequency motion. 2D video stabilization methods estimate (bundled) homography or affine transformations between consecutive frames and smooth these transformations temporally. In pioneer works, low-pass filters were applied to smooth parameters of models [1], [8]. An L^1-norm optimization-based method was proposed by Grundman *et al.* [3] with a path synthesis consisting of simple cinematography motions. Later, a bundled camera path based model was proposed by Liu *et al.* [5], estimating and smoothing multiple local camera paths. Zhang *et al.* [9] proposed to optimize geodesics on the Lie group embedded in transformation space to stabilize video. Liang *et al.* [10] analyzed the rolling shutter effect via global motion estimation and velocity estimation, and corrected the distortion via local motion refinement and scanline realignment. 3D-based video stabilization approaches reconstruct the 3D scene [11] from video, then estimate and smooth the 3D camera trajectory. Content-preserve warping [2] was proposed as the first 3D stabilization method. Later, subspace video stabilization [4] was proposed with long tracked features smoothed using subspace constraints. Goldstein and Fattal [12] proposed to enhance the length of feature tracks with epipolar transfer. Generally speaking, 2D stabilization methods perform efficiently and robustly, and 3D-based methods are able to generate visually better results.

Real-time *online* stabilization is specifically desired for live stream applications. Solutions combining the gyroscope hardware and image contents were applied on mobile phones [13]. Liu *et al.* [14] proposed an online stabilization method which only use historical camera path to compute warping functions for incoming frames. Inspired by their idea, we present a deep online stabilization approach which performs stabilization given a few historical stabilized frames. The novelty of our approach is that we avoid explicitly estimating and smoothing camera path, instead, we use a CNN model to directly predict warping functions.

B. CNNs for Video Applications

In recent years, CNNs have made huge improvements in computer vision tasks such as image recognition [15], [16] and generation [17], [18]. When feeding multiple successive frames from videos, CNNs can predict optical flow [19], camera motion [20], or semantics [21]. There are several works which use CNNs to directly produce video contents, such as scene dynamic generation [22], frame interpolation [23] and deblurring [7], [24]. Because predicting a long video sequence is still a challenging problem, all of the above works used only two or very few successive frames as training samples. The proposed StabNet also considers a temporal neighborhood at each time. The stabilization problem cannot be solved using a generation-based model because of the severe vibration of the input video content. To generate visually pleasing result, our *StabNet* learns the warping transformations instead of generating pixel values.

III. Training Dataset

Generating training data is one of the key challenges for digital video stabilization, where ground truth data cannot be easily collected/labeled. To train StabNet, two synchronized video sequences of the same scene are required: one sequence captures a steady camera movement, while the other is unstable. One possible way to generate such data is to render a

Fig. 2. Exemplar frames of DeepStab dataset. The dataset includes pairs of synchronously captured videos. Each pair consists of an unsteady video and a stabilized video, with the same content. Camera motions include forward movement, pan movement, spin movement and complex movements including combinations of the above, at various speed.

Fig. 3. Hardware and training data capturing process.

virtual scene with two camera path configurations: smooth and jumpy. However, CNN models trained using rendered virtual scene may not generalize well due to the domain gap between training synthetic video and testing real videos captured by hand-held camera. To generate authentic data, we designed a specialized hardware with two portable GoPro Hero 4 Black cameras and a hand-held stabilizer,[1] where the cameras lay horizontally next to each other with small disparity (Figure 3). When capturing videos, the two cameras shoot synchronously, with only one camera stabilized, while the other moves consistently with the hand/body motion of the holder. We turned off the auto-focus and auto-exposure functions of the cameras and used the synchronous remote control for synchronization.

Training videos are obtained by holding the designed hardware while taking videos in a first-person point of view. We present the *DeepStab* dataset, containing pairs of synchronized videos with diverse camera movements. The dataset includes indoor scenes with large parallax, and common outdoor scenes with buildings, vegetation, crowd, etc. Camera motions include forward movement, pan movement, spin movement and complex movements including combinations of the above, at various speed. We remove the fish-eye distortion of the videos in post-processing. We trim parts with large lighting difference between the cameras pair, and videos with non-overlapping field of views of the cameras by aligning the frame content and cropping a new rectangular view for each camera.

In total, we collected 60 pairs of synchronized videos whose length is within 30 seconds, at 30 FPS. The videos are split into 44 training pairs, 8 validation pairs and 8 testing pairs. Figure 2 shows representative sampled frames from the dataset. The recorded video pairs are augmented to provide more training samples by horizontally flipping the frames, reversing the video sequences and combining both flipping and reversing.

[1]https://www.youtube.com/watch?v=8vu7IDuDD64

IV. The StabNet

Overview: We propose to stabilize the video without using future frames, relying on how the hand-held stabilizer works during capturing paired steady and unsteady videos. We convert the online stabilization problem to a supervised learning problem of conditional transformation regression without explicitly computing a camera path. Our goal is to learn to warp the input video from an unstable camera to a virtually stable camera horizontally next to the unstable camera with a small parallax, as in the training data.

The inputs to *StabNet* are an incoming unsteady frame I_t and six conditional historical steady frames sampled from approximately one second $S_t = \langle \dot{I}_{t-32}, \dot{I}_{t-16}, \dot{I}_{t-8}, \dot{I}_{t-4}, \dot{I}_{t-2}, \dot{I}_{t-1} \rangle$ for time-stamp t. The sampling of historical frames are denser near the incoming frame and sparser far from the incoming frame. Inspired by [5] which uses a bundled camera model for stabilization, we propose to regress a transformation $f_t^{i,j}$ for the (i, j)-th regularly divided meshgrid $g_t^{i,j}$, where a 4×4 mesh $G_t = \{g_t^{i,j} | 1 \leq i, j \leq 4\}$ are spatially distributed on frame I_t. The output of our model is consequentially a set of transformations $F_t = \{f_t^{i,j} | 1 \leq i, j \leq 4\}$ for frame I_t. The steady frame is then created by applying

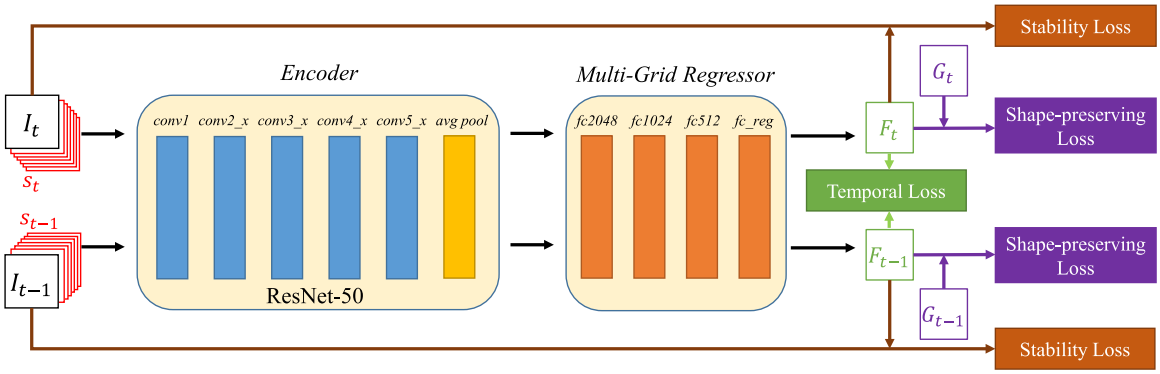


Fig. 4. Network Architecture. *StabNet* is a two-branch Siamese network with shared parameters in each branch. It consists of an *Encoder* and a *Multi-Grid Regressor*. The Encoder is an adapted ResNet-50 backbone model, which encodes input concatenated frames into a $1 \times 1 \times 2048$ feature vector. The Multi-Grid Regressor consists of a sequence of FC layers where the last *fc_reg* layer regresses grid vertex positions with dimension $(h+1) \times (w+1) \times 2$, and h, w are grid numbers along x-axis and y-axis respectively. During training, samples $\langle I_t, s_t \rangle$ and $\langle I_{t-1}, s_{t-1} \rangle$ of two successive incoming frames with corresponding historical frames fed to the network. The transformations F_t and F_{t-1} are then predicted. The network is trained with *stability loss*, *shape-preserving loss* and *temporal loss*.

$\widehat{I}_t = F_t * I_t$, where $*$ is the warping operator. We use the desired vertices $\{(\hat{x}_t^i, \hat{y}_t^j), (\hat{x}_t^{i+1}, \hat{y}_t^j), (\hat{x}^i, \hat{y}_t^{j+1}), (\hat{x}^{i+1}, \hat{y}^{j+1})\}$ of the deformed mesh grid $\hat{g}_t^{i,j}$ to represent each transformation $f_t^{i,j}$. Our network can regress the mesh grid vertex transformation representation, and can drive the warping of image content located inside the grid. The learning process is supervised by our ground-truth steady frames I'_t. When training *StabNet*, the conditional inputs S_t are the ground-truth steady frames $\langle I'_{t-32}, I'_{t-16}, I'_{t-8}, I'_{t-4}, I'_{t-2}, I'_{t-1} \rangle$, while when testing, S_t are the historical stabilized frames $\langle \widehat{I}_{t-32}, \widehat{I}_{t-16}, \widehat{I}_{t-8}, \widehat{I}_{t-4}, \widehat{I}_{t-2}, \widehat{I}_{t-1} \rangle$.

A. Network Architecture

Our *StabNet* is a Siamese network [25] that has two branches sharing the network parameters. We use a Siamese architecture to preserve temporal consistency of successive transformed frames $\widehat{I}_{t-1} = F_{t-1} * I_{t-1}$ and $\widehat{I}_t = F_t * I_t$. Each branch of *StabNet* is a two-stage network consisting of a backbone *encoder*, that extracts high-level features from the inputs and a multi-grid transformation *regressor*, that predicts the stabilization transformations from the extracted feature map. Figure 4 shows the architecture of *StabNet*. The inputs are seven concatenated grayscale frames, each with dimension $W \times H \times 1$, consisting of six conditional steady frames S_t and one unsteady frame I_t. Frames are sent to an *encoder* to extract features. This encoder adapts ResNet-50 [16] as the backbone feature extractor, using the *conv_1* as the input channel, modified to meet our inputs, and removing all layers after *average pooling*. The extracted feature map from the *encoder* is of dimension $1 \times 1 \times 2048$. Next, we use a sequence of *FC layers* with output feature dimensions $\langle 2048, 1024, 512, (h+1) \times (w+1) \times 2 \rangle$, where $w = 4$ and $h = 4$ are grid sizes along x-axis and y-axis respectively. The output dimension corresponds to the total number of grid vertex points.

B. Stabilization Loss Functions

StabNet training process is driven by three types of loss functions: stability loss, shape-preserving loss and temporal smoothness loss. The comprehensive loss function is based on neighboring input frames I_t and I_{t-1}, and is defined as:

$$L = \sum_{i \in \{t, t-1\}} L_{\text{stab}}(F_i, I_i) + L_{\text{shape}}(F_i, G_i) + L_{\text{temp}}(F_t, F_{t-1}, I_t, I_{t-1}), \quad (1)$$

where L_{stab} is the *stability loss*, L_{shape} is the *shape-preserving loss* and L_{temp} is the *temporal loss*.

1) Stability Loss: The *stability loss* drives the warped unsteady frames to the ground-truth steady frames using cues of pixel alignment and feature point alignment. It is defined as:

$$L_{\text{stab}}(F_t, I_t) = \alpha_1 L_{\text{pixel}}(F_t, I_t) + \alpha_2 L_{\text{feature}}(F_t, I_t), \quad (2)$$

where L_{pixel} is the pixel alignment term, L_{feature} is the feature alignment term, and $\alpha_1 = 50.0$, $\alpha_2 = 1.0$ are constant weights.

The pixel alignment term L_{pixel} measures how the transformed frame $\widehat{I}_t = F_t * I_t$ aligns with the ground-truth steady frame I'_t, using mean squared error (MSE):

$$L_{\text{pixel}}(F_t, I_t) = \frac{1}{D} ||I'_t - F_t * I_t||_2^2, \quad (3)$$

where D is the spatial dimension of frame. The transformation $F_t * I_t$ operates in the image domain. To make the warping function differentiable, we used spatial transformer layer [26]. L_{pixel} loss will be small if the transformed frame $\widehat{I}_t$ aligns well with the ground-truth frame I'_t. However, during training $\widehat{I}_t$ can not converge well to I'_t. During early training stages, unsteady and steady frames are not aligned and the loss term is less correlated. For better convergence during training, we further introduce a feature alignment loss.

The feature alignment term L_{feature} is computed as the average alignment error of matched *feature points* after transforming the unsteady frame I_t using the predicted transformation F_t:

$$L_{\text{feature}}(F_t, I_t) = \frac{1}{m} \sum_{i=1}^{m} ||p_t'^i - F_t * p_t^i||_2^2. \quad (4)$$

where $P_t = \{\langle p_t^i, p_t'^i \rangle \mid i \in \{1, \cdots, m\}\}$ are the m pairs of matched feature points between each steady/unsteady frame

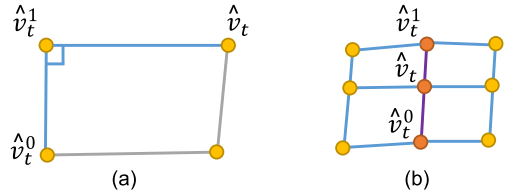


Fig. 5. Shape-preserving loss terms. (a) illustrates the intra-grid distortion term associating three triangular grid vertices. (b) shows the inter-grid consistency term on three consecutive mesh vertices along an edge.

pair, and p_t^i and $p_t'^i$ are the i-th matched feature points from unsteady frame I_t and ground-truth steady frame I_t' respectively.

To compute the feature loss, all pairs P_t are computed in a pre-processing stage between steady and unsteady frame pairs. We extract SURF features [27] from both I_t and I_t', then calculate the matching between them by dividing the frames into 2×2 sub-images, and using a RANSAC algorithm [28] to fit a Homography in each corresponding sub-image. We match features in 2×2 sub-images instead of 4×4 as in [5], because of the large camera pose and content variation between the steady and unsteady cameras. Please note that the feature extraction and feature matching processes are *only* performed for training the network and not needed during online stabilization.

2) Shape-Preserving Loss: Because our model regresses mesh vertex positions of the stabilized video, it is important to preserve the shapes of grids to avoid distortion artifact and to encourage neighboring grids to transform consistently. Our shape-preserving loss thus consists of an intra-grid distortion term L_{intra} and an inter-grid consistency term L_{inter}.

Inspired by [2], we introduce an intra-grid loss L_{intra} to encourage the triangle of neighboring deformed vertices $\{\hat{v}_t, \hat{v}_t^0, \hat{v}_t^1\} \subset f_t * g_t$ to follow a similarity transformation:

$$L_{\text{intra}}(F_t, G_t) = \frac{1}{N} \sum_{\hat{v}_t} ||\hat{v}_t - \hat{v}_t^1 - sR\vec{v}_t^{01}||_2^2, R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{5}$$

where $\vec{v}_t^{01} = \hat{v}_t^0 - \hat{v}_t^1$, $\hat{v}_t^0$ and $\hat{v}_t^1$ are neighboring vertices and $s = ||v_t - v_t^1||/||v_t^0 - v_t^1||$, $\{v_t, v_t^0, v_t^1\} \subset g_t$ is the ratio of original grid side lengths, N is the total amount of triangular vertices.

To encourage the neighboring grids to transform consistently, we introduce an inter-grid loss L_{inter}. For each vertex v_t and its neighboring vertices v_t^0, v_t^1 along an edge of two original neighboring grids, the two vectors $\vec{v}_t = \hat{v}_t^1 - \hat{v}_t$ and $\vec{v}_t^0 = \hat{v}_t - \hat{v}_t^0$ formed by deformed vertices are encouraged to be identical:

$$L_{\text{inter}}(F_t, G_t) = \frac{1}{M} \sum_{\langle \hat{v}_t^0, \hat{v}_t, \hat{v}_t^1 \rangle} ||\hat{v}_t^1 - \hat{v}_t - (\hat{v}_t - \hat{v}_t^0)||_2^2, \tag{6}$$

where $\langle \hat{v}_t^0, \hat{v}_t, \hat{v}_t^1 \rangle$ are three successive deformed grid vertices belonging to $F_t * G_t$, along an original mesh edge, M is the total amount of successive vertex tuples of the mesh. Figure 5 shows an illustration of the loss terms.

The shape-preserving loss is then defined as the combination of the above two terms:

$$L_{\text{shape}}(F_t, G_t) = \gamma_1 L_{\text{intra}}(F_t, G_t) + \gamma_2 L_{\text{inter}}(F_t, G_t), \tag{7}$$

with the weights set as $\gamma_1 = 1.0$, $\gamma_2 = 20.0$.

3) Temporal Loss: Simply applying the transformations separately to every video frame can create wobble artifacts in the video. Therefore, we incorporate a temporal loss term to enforce temporal coherency between adjacent frames using the Siamese network architecture. Each time two successive samples $\langle I_t, s_t \rangle$ and $\langle I_{t-1}, s_{t-1} \rangle$ are fed into *StabNet*, two successive transformations F_t and F_{t-1} are predicted. The temporal loss is defined as the mean square error between the successive output frames:

$$L_{\text{temp}}(F_t, F_{t-1}, I_t, I_{t-1}) = \lambda \frac{1}{D} ||F_t * I_t - w(F_{t-1} * I_{t-1})||_2^2, \tag{8}$$

where D is the spatial dimension of frame, $w(\cdot)$ is a function that warps the steady frame at $t-1$ to the steady frame t according to pre-computed optical flow, $\lambda = 10.0$ is a constant. In our experiments we use TV-L1 algorithm [29] to compute the optical flow, but alternative methods for optical flow calculation can also be used.

C. Implementation Details

To train *StabNet*, we resize the videos to a spatial dimension of $W = 512$ and $H = 288$ for efficiency. Pre-trained ResNet-50 model on ImageNet [15] without the *Conv_1* layer is loaded, and is fine-tuned during the training process. We use mini-batch size of 8 and ADAM [30] for optimization with $\beta_1 = 0.9$, $\beta_2 = 0.999$. Initial learning rate is set to 2e-5, and multiplied by 0.1 every 30,000 iterations. The training is initialized to learn identity transformations for 300 iterations before introducing aforementioned losses. The training process is terminated when reaching 90,000 iterations. The whole training process takes about 20 hours on an NVIDIA GTX 1080 Ti graphics card.

In training process, we feed two successive samples to the two branches (with shared network parameters) of *StabNet* so that temporal coherency is aware during learning. However, during testing, the network is used to stabilize a single frame at a time; temporal consistency is automatically preserved. Further, the stabilization processing is self-driven for a test video as follows: we start by duplicating the first frame and regard the duplicated frames as S_1. After stabilizing frame I_t, historical stabilized frames $\langle \widehat{I}_{t-31}, \widehat{I}_{t-15}, \widehat{I}_{t-7}, \widehat{I}_{t-3}, \widehat{I}_{t-1}, \widehat{I}_t \rangle$ are regarded as S_{t+1} for stabilizing the next frame I_{t+1}. This process is repeated through the time-line.

The stabilization results inevitably have meaningless frame borders introduced by the warping function. As *StabNet* uses stabilized frames as the inputs for future frames, we need to make *StabNet* robust to such borders. During training, we add some black borders produced by Homography perturbation around the Identity transformation to the ground-truth historical frames. The Homography perturbances are

TABLE I

ABLATION STUDY OF ARCHITECTURE, LOSS FUNCTION AND INPUT VARIATIONS. EACH ROW (EXCEPT THE FIRST COLUMN) FROM LEFT TO RIGHT SHOWS THE STABILIZATION STATISTICS IN SIX SUB-SET OF VIDEOS FROM [5]: *Regular*, *Quick Rotation*, *Quick Zooming*, *Parallax*, *Running* AND *Crowd*, IN THREE METRICS: *Cropping Ratio* (C), *Distortion Value* (D) AND *Stability Score* (S). THE FIRST TWO ROWS OF STATISTICS SHOW PERFORMANCES OF ALTERNATIVE ARCHITECTURES, THE NEXT FIVE ROWS COMPARE THE RESULTS WITHOUT *Feature*, *Pixel*, *Temporal*, *Distortion* AND *Consistency* LOSS FUNCTIONS, THEN THE RESULTS OF THREE INPUT VARIATIONS ARE SHOWN. THE LAST ROW SHOWS THE PERFORMANCE OF THE PROPOSED NETWORK. SYMBOL "-" MEANS THE CORRESPONDING NETWORK DOES NOT CONVERGE

	Regular			Rotation			Zooming			Parallax			Running			Crowd		
	C	D	S	C	D	S	C	D	S	C	D	S	C	D	S	C	D	S
SGR	0.66	0.92	0.70	0.52	0.41	0.83	0.58	0.88	0.76	0.57	0.62	0.67	0.53	0.81	0.73	0.47	0.69	0.62
MGR-2	0.63	0.91	0.72	0.48	0.39	0.84	0.55	0.81	0.77	0.59	0.87	0.67	0.52	0.82	0.75	0.46	0.77	0.65
w/o feat.	0.70	0.90	0.69	0.42	0.47	0.81	0.51	0.87	0.71	0.56	0.66	0.66	0.44	0.73	0.70	0.34	0.48	0.66
w/o pix.	0.56	0.86	0.72	0.49	0.40	0.75	0.65	0.84	0.70	0.54	0.72	0.67	0.52	0.81	0.76	0.48	0.62	0.67
w/o temp.	0.59	0.86	0.65	0.37	0.39	0.78	0.46	0.78	0.76	0.54	0.62	0.70	0.42	0.78	0.72	0.36	0.56	0.63
w/o dist.	0.59	0.86	0.71	0.40	0.32	0.83	0.51	0.83	0.70	0.57	0.78	0.68	0.43	0.78	0.73	0.34	0.60	0.68
w/o cons.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Var. 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Var. 2	0.62	0.89	0.70	0.43	0.47	0.83	0.50	0.87	0.77	0.55	0.65	0.69	0.47	0.75	0.74	0.39	0.44	0.62
Var. 3	0.60	0.47	0.72	0.46	0.56	0.82	0.57	0.52	0.77	0.63	0.25	0.70	0.43	0.70	0.76	0.40	0.40	0.66
Ours	0.61	0.83	0.75	0.41	0.54	0.83	0.52	0.80	0.79	0.57	0.70	0.71	0.60	0.80	0.77	0.50	0.71	0.68

randomly sampled between $\mathcal{H}_{\min} = \begin{bmatrix} 0.9 & -0.1 & -0.5 \\ -0.1 & 0.9 & -0.5 \\ -0.1 & -0.1 & 1 \end{bmatrix}$ and $\mathcal{H}_{\max} = \begin{bmatrix} 1.1 & 0.1 & 0.5 \\ 0.1 & 1.1 & 0.5 \\ 0.1 & 0.1 & 1 \end{bmatrix}$, where the image axis is normalized to $[-1, 1]$. For testing, we crop and trim the borders in post-processing. We plan to release source code and pre-trained *StabNet* model.

V. EXPERIMENTAL RESULTS

We train the *StabNet* model on the *DeepStab* dataset, and test it on various video sources. Testing videos are from our *DeepStab* testing set, previous dataset [5] and mobile phone cameras. On average, testing runs at 35.5 FPS on a graphics card, which meets the requirement of real-time online stabilization with 1 frame latency.

We use quantitative evaluation metrics, computed following [14] to evaluate stabilization methods. The three metrics are *cropping ratio*, *distortion* and *stability*.

Cropping Ratio: This metric measures the area of the remaining content after stabilization. Larger cropping ratio with less cropping is favored. Per frame cropping ratio is computed as the scale component of the global Homography H_t estimated from input frame I_t to output frame $\widehat{I}_t$. Ratio values of video frames are averaged to generate the cropping ratio value of the whole video.

Distortion Value: Distortion value evaluates the distortion degree introduced by stabilization. Per frame distortion value is computed by the ratio of the two largest eigenvalues of the affine part of the Homography H_t. The minimum value which represents the worst distortion is chosen as the distortion value for the whole video.

Stability Score: Stability score measures how stable a video is. Following [14], we use frequency-domain analysis of camera paths to estimate the stability score. Spatially distributed camera paths are computed as vertex profiles for 4×4 mesh grid vertices between successive frames. The vertex profiles are then presented as 1D temporal signals for frequency domain analysis. We take each of their lowest frequencies components over full frequencies (DC component is excluded) as the stability score [14]. Averaging from all profiles gives the final score.

A. Ablation Study

In order to evaluate the effectiveness of our proposed framework, we experiment with other possible network architectures, loss functions and alternative input solutions. We conduct an ablation study on public stabilization dataset from [5] which consists of several video categories according to scene type and camera motion, including *Regular*, *Quick Rotation*, *Quick Zooming*, *Large Parallax*, *Running* and *Crowd*.

1) Network Architectures: Because our network uses *multi-grid regressor* (denoted as *MGR*) to learn transformations for each input frame, here we evaluate how the proposed *MGR* performs against the *single-regressor* one (denoted as *SGR*) and the alternative *MGR* variations with various grid divisions. We implement the variations using the same backbone ResNet-50 encoder, and similar regressor architectures with the output channels $(h + 1) \times (w + 1) \times 2$ adapted to mesh division choices. In *SGR*, four frame vertex positions are regressed. In *MGR* variations, 2×2 and 8×8 mesh vertex positions are regressed, with architectures denoted as *MGR-2* and *MGR-8* respectively. As a result, the stability level of results from *SGR* and *MGR-2* are inferior to our *MGR*, as they regress coarser grids; at the same time *MGR-2* and *SGR*'s cropping ratio values are generally higher than our method. This is because in a method with a better stability, a more flexible warping must be performed, with larger warping borders. We also observe that although the mesh division of *MGR-8* is finer than our model, regressing transformations in such granularity ended in failure.

2) Loss Functions: We test the proposed *StabNet* with some of the loss terms turned off to validate the loss function setup. We observed that without *consistency loss*, the network training will not converge, and other alternative results are worse than the proposed one.

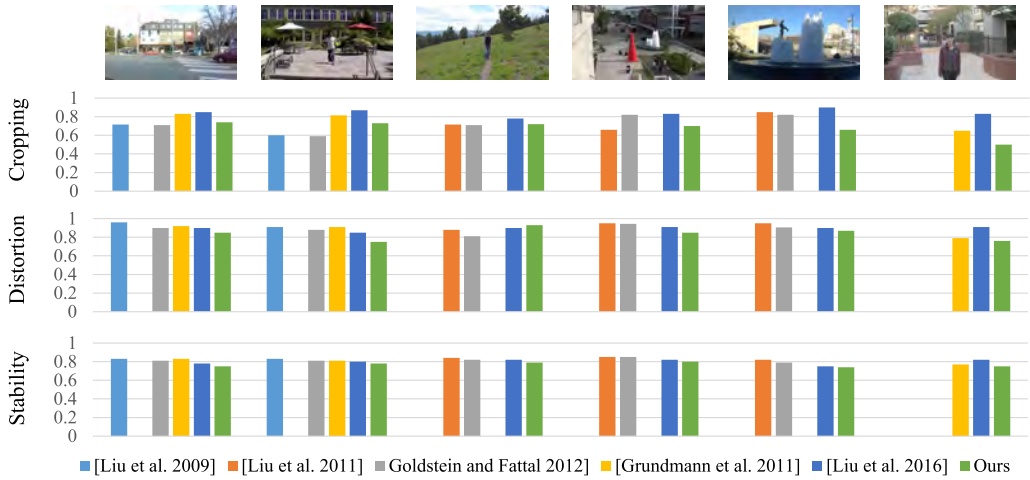


Fig. 6. Compare with publicly available videos from [5] in terms of three metrics.

3) Input Variations: We experiment with different stacked input frame sequences during training the model. The variations are: 1) training one frame supervised by neighboring historical frames as $\langle I'_{t-5}, I'_{t-4}, \cdots, I'_{t-1}, I\rangle$; 2) training one frame supervised by uniformly distributed historical frames as $\langle I'_{t-31}, I'_{t-25}, I'_{t-19}, I'_{t-13}, I'_{t-7}, I'_{t-1}, I\rangle$; 3) training current frame and a few future frames with historical guidance as $\langle I'_{t-32}, I'_{t-16}, \ldots, I'_{t-2}, I'_{t-1}, I, I_{t+1}, I_{t+2}, \ldots, I_{t+16}, I_{t+32}\rangle$.

Corresponding performances are reported in Table I, as a conclusion, results from alternative inputs are inferior to the proposed one in terms of stability. We also experiment with the back-bone of ResNet-101, however the improvement is not apparent, with running time increased.

B. Comparison With Publicly Available Results

We compare with [2]–[4], [12], and [14] using six publicly available videos in terms of the objective metrics, based on results provided by corresponding authors. Comparing with offline stabilizations is slightly unfair for our method because future-frames information is not available for our online stabilization method in real-time. As a result, the stability score of our method is slightly lower, occasionally with probable visual artifacts of unnatural cross-frame wobbling and distortion. This is mainly because our online method only uses historical frames without holistic knowledge of the full camera path. Nevertheless, our method performs in real time while being visually comparable to all existing methods. Comparison details are shown in Figure 6, for videos that we were not able to find the result, we leave it blank.

C. Comparison With the State-of-the-Art Software

We further compare our method with commercial offline stabilization software *Adobe Premiere CS6* on dataset [5]. As far as we know, Adobe Premiere stabilizer is developed based on subspace stabilization [4]. We choose the default parameters for Adobe Premiere (smoothness: 50%, 'Smooth Motion' and 'Subspace Warp') to produce results. Figure 7 shows an visualization of feature trajectories before and after

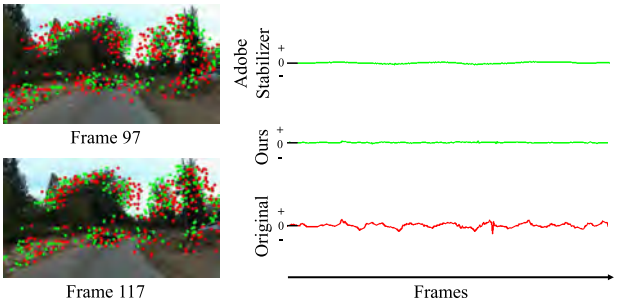


Fig. 7. Visualization of feature trajectories from an unsteady video and the corresponding stabilized videos. Left: two stabilized frames by *StabNet*, with feature trajectories (green) and the feature trajectories from the original video (red) highlighted. Right: visualization of the average horizontal feature offsets between neighboring frames along the time-line, from the original unsteady video, Adobe Stabilizer and our *StabNet*.

TABLE II
RUNNING TIME COMPARISON. FPS STATISTICS ARE GIVEN IN THE SECOND COLUMN. THIRD COLUMN SHOWS WHETHER FUTURE FRAMES ARE REQUIRED FOR STABILIZATION

Method	FPS	Future Frames
Bundled Cameras	3.5	Yes
Adobe Premiere	4.2	Yes
MeshFlow	22.0	No
Ours	35.5	No

stabilization. Further evaluation on the test dataset is reported in Figure 8. Please note that the online stabilization problem is inherently harder than offline stabilization, because only historical frames are available in online stabilization, without the global sense of the camera path. Hence, the quantitative performance statistics for online stabilization methods would be inferior to offline ones. However the average running time performance of our method is superior to all existing methods. The running time performance is given in Table II.

D. Stabilizing Low-Quality Videos

One promising feature of *StabNet* is its robustness to low-quality videos caused by noise, motion blur, etc. When dealing with such videos, traditional methods could fail because of

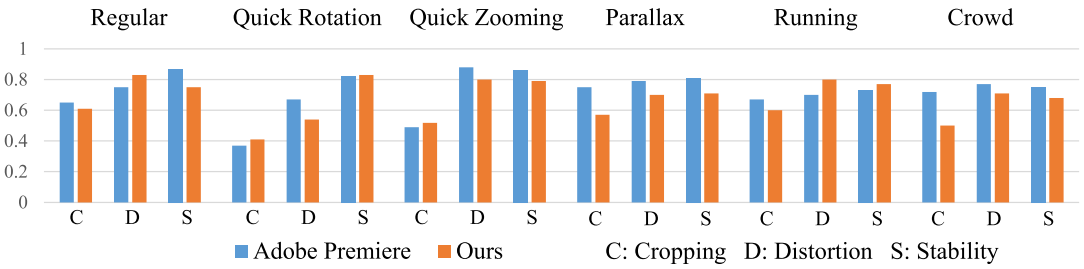


Fig. 8. Quantitative comparison with Adobe Premiere CS6 stabilizer on six categories of hand-held videos.

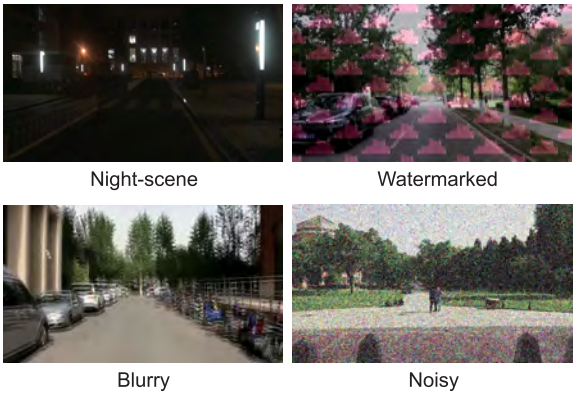


Fig. 9. Representative frames of low-quality videos. Please refer to the supplementary video for stabilization result comparison.

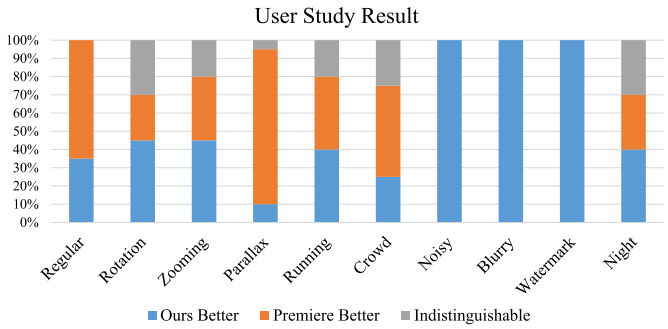


Fig. 10. User study result by comparing our method with Adobe Premiere Stabilizer.

either feature extraction failures from one frame or feature-mismatches between frames. We demonstrate the superiority of *StabNet* to traditional feature-based methods via four types of low-quality videos captured by mobile phones: night-scene videos, watermarked videos, blurry videos and noisy videos, whose representative frames are shown in Figure 9.

1) Night-Scene Videos: We test the proposed network on night-scene videos. Such videos are typically blurry and contain severe noise, where wobble distortions can appear from traditional feature matching-based stabilization methods.

2) Watermarked Videos: Watermarks such as logos or repetitive patterns can be overlaid on video frames. We synthesize watermarked videos using repetitive patterns and overlay the patterns at the same spatial positions across video frames. Such repetitive watermark patterns can disturb feature matching process from the original video content, resulting in false feature matchings from existing stabilization methods.

3) Blurry Videos: Motion blur can appear in shaky frames and cause uncomfortable viewing experience. Such motion blur makes it difficult to extract and match features using existing stabilization methods.

4) Noisy Videos: Videos could be noisy if capturing is effected by poor illumination, high temperature, etc. Gaussian noise could be introduced from multiple noise sources. In our experiments, we synthesize noisy videos by adding Gaussian noise to each video frame, and apply stabilization algorithms to the videos. The match of features would fail using existing stabilization methods.

In our experimental results, on low quality video cases, *StabNet* performs robustly, while traditional stabilization methods such as subspace stabilization [4] fail to generate stable results. Please refer to the supplementary video for visual comparison.

E. User Study

To visually compare our method with Adobe Premiere stabilizer, we further conduct a user study with 20 participants aged from 18 to 32. We provide 18 videos from [5], 3 from each aforementioned category; and 12 low-quality videos, 3 from each aforementioned type. In each testing case, we simultaneously show the original input video, our result, and the result from Adobe Premiere stabilizer to the subjects. The two stabilization results are displayed horizontally in random order. Every participant is asked to pick the more stable result from the results of our method and Adobe Premiere stabilizer, or mark them “indistinguishable”, while disregarding differences in aspect ratio, or sharpness. We show the average percentage of user preference for each category in Figure 10. It can be concluded that for *low-quality videos*, our method performs much better, and for videos from *Quick Zooming*, *Quick Rotation*, *Running* categories, our results are comparable with those from offline approach. For other categories that were harder to process without future frames, our results are slightly worse, which coincides with our aforementioned discussion.

VI. Limitation and Conclusion

StabNet has limitations. First, controlling cropping ratio is not supported by our network, which may generate warping borders in the stabilized video. However, like some existing offline video stabilization methods, with an automatic processing of warping border trimming off after holistic path

stabilization, the final rendered videos do not contain borders. In our case, the stabilization (with warping borders) and cropping bounding box position are computed and updated progressively in an online stage. The warping borders can be further trimmed off with an automatic post-processing cropping stage, based on the computed bounding box. Nevertheless, one possible way to control cropping ratio is to train a network conditioned with a required specific cropping ratio, which we regard as a future work. Second, in scenes with drastic motion or with extreme near-range foreground objects, our method may fail, this is because our model learns to warp the unstable camera to a virtual stable camera with parallax. We note that these scenarios are also challenging for previous methods [3]–[5], [14]. Third, our solution is purely based on software. Fused video stabilization with additional gyro signals using CNNs [13] is an interesting future research direction.

To summarize, we have presented *StabNet*, a convolutional network for digital online video stabilization. Unlike traditional methods which calculate estimated camera paths, *StabNet* learns warping transformations of multi-grids for each unsteady frame, using only historical stabilized frames as condition. It runs in real time by fast feed-forward operations. We also present the *DeepStab* dataset–a dataset consisting of pairs of synchronized steady/unsteady videos for training. This dataset was created using a practical method to generate training videos with synchronized steady/unsteady frames, which could benefit future deep stabilization methods. To our knowledge, *StabNet* is the first CNN model for video stabilization. We have demonstrated the power of *StabNet* for handling typical types of hand-held videos and its advantage in stabilizing low-quality videos. We believe CNN-based methods are a promising direction for digital video stabilization.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers.

REFERENCES

[1] 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.

[2] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 44:1–44:9, Aug. 2009.

[3] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. IEEE CVPR*, Jun. 2011, pp. 225–232.

[4] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Trans. Graph.*, vol. 30, no. 1, pp. 4:1–4:10, Feb. 2011.

[5] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 78:1–78:10, Jul. 2013.

[6] H. Huang *et al.*, "Real-time neural style transfer for videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7044–7052.

[7] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. (2016). "Deep video deblurring." [Online]. Available: https://arxiv.org/abs/1611.08387

[8] H.-C. Chang, S.-H. Lai, and K.-R. Lu, "A robust real-time video stabilization algorithm," *J. Vis. Commun. Image Represent.*, vol. 17, no. 3, pp. 659–673, 2006.

[9] L. Zhang, X.-Q. Chen, X.-Y. Kong, and H. Huang, "Geodesic video stabilization in transformation space," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2219–2229, May 2017.

[10] C.-K. Liang, L.-W. Chang, and H. H. Chen, "Analysis and compensation of rolling shutter effect," *IEEE Trans. Image Process.*, vol. 17, no. 8, pp. 1323–1330, Aug. 2008.

[11] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.

[12] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Trans. Graph.*, vol. 31, no. 5, pp. 126:1–126:10, Sep. 2012.

[13] C.-K. Liang, "Fused video stabilization on the pixel 2 and pixel 2 xl," Google Res. Blog, Mountain View, CA, USA, Tech. Rep. 11, 2017.

[14] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "MeshFlow: Minimum latency online video stabilization," in *Proc. IEEE Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 800–815.

[15] 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. (CVPR)*, Jun. 2009, pp. 248–255.

[16] 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.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. (2016). "Image-to-image translation with conditional adversarial networks." [Online]. Available: https://arxiv.org/abs/1611.07004

[18] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.

[19] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.

[20] Y. Nakajima and H. Saito, "Robust camera pose estimation by viewpoint classification using deep learning," *Comput. Vis. Media*, vol. 3, no. 2, pp. 189–198, 2017.

[21] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1, 2014, pp. 568–576.

[22] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," in *Proc. NIPS*, 2016, pp. 91–99.

[23] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. (2017). "Video frame synthesis using deep voxel flow." [Online]. Available: https://arxiv.org/abs/1702.02463

[24] T. H. Kim, K. M. Lee, and B. Schölkopf, and M. Hirsch. (2017). "Online video deblurring via dynamic temporal blending network." [Online]. Available: https://arxiv.org/abs/1704.03285

[25] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4353–4361.

[26] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. NIPS*, 2015, pp. 2017–2025.

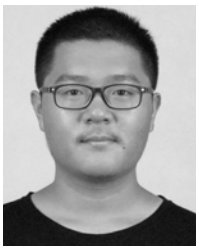
[27] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.

[28] 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.

[29] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TVL1-optical flow estimation," *Image Process. On Line*, vol. 3, pp. 137–150, 2013.

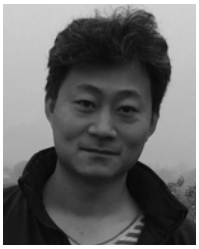
[30] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

Miao Wang received the Ph.D. degree from Tsinghua University in 2016. From 2013 to 2014, he visited the Visual Computing Group, Cardiff University, as a Joint Ph.D. Student. From 2016 to 2018, he was a Post-Doctoral Researcher with Tsinghua University. He is currently an Assistant Professor with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His research interests lie in computer graphics with a particular focus on interactive image and video editing and video in virtual reality.



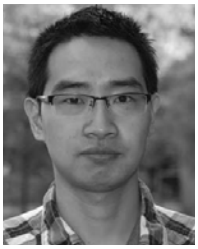
Guo-Ye Yang is currently pursuing the bachelor's degree with Tsinghua University. His research interests include computer graphics, image analysis, and computer vision.

Jin-Kun Lin is currently pursuing the bachelor's degree with Tsinghua University. His research interests include computer graphics, image analysis, and computer vision.



Song-Hai Zhang received the Ph.D. degree from Tsinghua University, Beijing, China, in 2007. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include image and video processing and geometric computing.

Ariel Shamir received the B.Sc. and M.Sc. degrees *(cum laude)* in mathematics and computer science from Hebrew University, Jerusalem, and the Ph.D. degree in computer science in 2000. He spent two years as a Post-Doctoral Fellow with the Computational Visualization Center, The University of Texas in Austin. He was a Visiting Scientist at Mitsubishi Electric Research Labs, Cambridge, MA, USA, in 2006, and at Disney Research in 2014. He is currently a Professor with the Efi Arazi School of Computer Science, Interdisciplinary Center Herzliya, Israel. His research interests include geometric modeling, computer graphics, fabrication, visualization, and machine learning. He is a member of the ACM SIGGRAPH and the IEEE Computer and Eurographics societies.



Shao-Ping Lu received the Ph.D. degree from Tsinghua University, China, in 2012. From 2013 to 2017, he was a Post-Doctoral Researcher and a Senior Researcher at Vrije Universiteit Brussel. He is currently an Associate Professor with Nankai University, Tianjin, China.

His research interests lie primarily in the intersection of visual computing, computer vision, and computer graphics, with a particular focus on 2D/3D image and video processing, computational photography and representation, visual scene analysis, machine learning, and mathematical optimization.

Shi-Min Hu received the Ph.D. degree from Zhejiang University in 1996. He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, and the Director of the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing. He has authored over 100 papers in journals and refereed conference. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is the Editor-in-Chief of *Computational Visual Media* and on the Editorial Board of several journals, including IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, *Computer-Aided Design*, and *Computer & Graphics*.