Learning Video Stabilization Using Optical Flow

Jiyang Yu
University of California, San Diego
jiy173@eng.ucsd.edu

Ravi Ramamoorthi
University of California, San Diego
ravir@cs.ucsd.edu

Abstract

We propose a novel neural network that infers the per-pixel warp fields for video stabilization from the optical flow fields of the input video. While previous learning based video stabilization methods attempt to implicitly learn frame motions from color videos, our method resorts to optical flow for motion analysis and directly learns the stabilization using the optical flow. We also propose a pipeline that uses optical flow principal components for motion inpainting and warp field smoothing, making our method robust to moving objects, occlusion and optical flow inaccuracy, which is challenging for other video stabilization methods. Our method achieves quantitatively and visually better results than the state-of-the-art optimization based and deep learning based video stabilization methods. Our method also gives a ∼3x speed improvement compared to the optimization based methods.

1. Introduction

Video stabilization is a common need in both amateur and professional video capture. On the amateur side, using specific video stabilization equipment is usually difficult and expensive. Therefore, various algorithms have been developed for stabilizing the video. There are two main steps in the video stabilization: understanding the motion pattern and stable frame generation. For the first step, previous video stabilization algorithms resort to explicit physics. Some of these algorithms use image feature detection and tracking to build feature tracks, like Matsushita et.al.[12], Liu et.al.[10] and Grundmann et.al.[5]. Other physically based algorithms consider the real 3D camera motion and scene structures, like Liu et.al.[8] and Goldstein and Fattal[4]. However, the complex nature of videos makes it very difficult for a physically based motion model to cover all the scenarios: the motion blur and occlusions may negatively affect the feature detection and tracking, and reconstructing 3D structure from video with these effects is also difficult. Having observed the limitation of physically based algorithms, in this paper, we use the optical flow to understand the frame motions.

For the second step, traditional video stabilization algorithms use full-frame or grid homography to warp the original frames. However, simple parameterized warping cannot handle complex scenarios like parallax and the rolling shutter effect. Like other optical flow based methods SteadyFlow[11] and Yu and Ramamoorthi[19], we use the per-pixel warp field, which provides the most flexibility in warping the video frames. These methods perform well on handling complex camera effects like rolling shutter. However, the optical flow essentially only provides a pseudo correspondence between two frames. At occlusion boundaries, pixels can either appear in the next frame or be occluded in the next frame. The optical flow is also inaccurate in regions lacking texture. Using the optical flow as the reference can lead to unexpected artifacts in the output warp fields. We propose a pipeline that is specifically designed to handle the inaccuracy in the optical flow. We are the first method that uses the optical flow principal components[17] in video stabilization instead of hand-crafted spatial smoothness constraints. We will discuss the details of our pipeline in Sec. 3.

The core of our algorithm is a deep neural network that takes the optical flow as the input and directly outputs the warp fields. Our neural network based method overcomes the major drawback of optical flow based methods: the computational complexity. Both SteadyFlow[11] and Yu and Ramamoorthi[19] use optimization to minimize an objective function with a significant amount of unknowns(the motion vectors in the warp fields). The optimization process must be performed for each different video. Our pre-trained network is generalizable to any videos; thus we avoid this main overhead compared to traditional optical flow based methods.

We summarize our contribution as follows:

a) An optical flow based video stabilization network: We proposed a novel neural network that takes the optical flow fields as the input and produces a pixel-wise warp field for each frame. Our neural network can be pre-trained and generalized to any videos. The details are discussed in Sec. 5.

b) Frequency domain regularized training: We propose a frequency domain loss function that enables learning with optical flow fields. We will show the necessity of this loss function in Sec. 5.1.

c) Robust video stabilization pipeline: We propose a pipeline that is robust to moving occlusion and optical flow inaccuracy by applying PCA Flow to video stabilization. The design is demonstrated in Sec. 3. In Sec. 7, we will show that our method generates better results compared to the state-of-the-art optimization based and deep learning based video stabilization methods. Our method also achieves ∼3x speed improvement compared to optimization based methods.

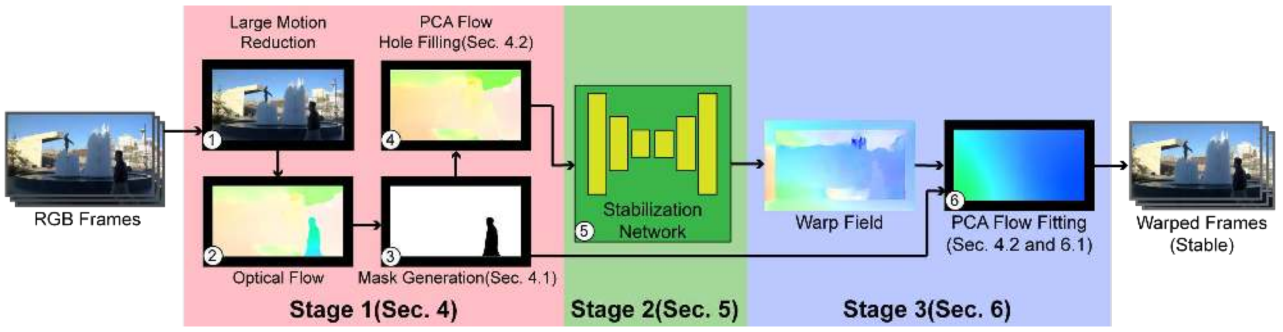


Figure 1. *The 3-stage pipeline of our algorithm. ①In the first stage(Sec. 4) we initially stabilize the video with translation and rotation. ②We compute the optical flow between consecutive frames. ③We generate a mask for each frame to indicate the valid regions for stabilization(Sec. 4.1). ④The invalid regions are inpainted using PCA Flow(Sec. 4.2). ⑤In the second stage(Sec. 5), our stabilization network infers the warp field from the inpainted optical flow fields. ⑥In the third stage(Sec. 6), we fit the PCA Flow to the raw warp field and use the smoothed warp field to warp the input video.*

2. Related Works

In this section, we summarize the traditional physically based video stabilization methods and the recent deep learning based methods. Most existing video stabilization works are 2D physically based methods. The methods below all involve 2D feature tracking. The difference is mainly from the method for feature track smoothing and stable frame generation. Buehler et al.[2] re-render the frames at smoothed camera positions using the non-metric IBR algorithm. Matsushita et al.[12] and Gleicher and Liu[3] use simple 2D full-frame transformations to warp the original frames. Liu et al.[10] uses a grid to warp the frames and smoothes the enclosing feature tracks. Grundmann et al.[5] proposed an L1 optimal camera path for smoothing the feature tracks. Liu et al.[9] extracts and smoothes eigen-trajectories. Goldstein and Fattal[4] constrain the feature track smoothing with the epipolar geometry. Wang et al.[16] also keep the relative position of feature points, but use only 2D constraints.

In addition to these 2D physically based methods, Liu et al.[8] first reconstruct the 3D position of the feature points and camera positions, then smooth the camera trajectory and reproject the feature points to new camera positions. Sun[14] and Smith et al.[13] also use 3D information, but they require depth cameras and light field cameras respectively.

Later works use optical flow and smooth the motion at the pixel level. SteadyFlow[11] smoothes the motion vector changes on each pixel using iterative Jacobi-based optimization. Yu and Ramamoorthi[19] track the pixel motion using the optical flow. They optimize the neural network weights that generate the warp field, instead of solving for the warp field directly. Their optimization must be repeated for each new video. Our method also uses a neural network to infer the pixel-wise warp field, but our network is pre-trained and can be generalized to any videos. Moreover, as we discussed in Sec. 1, using optical flow in video stabilization leads to fundamental problems. Our method is designed specifically to overcome these problems.

Recent works start to apply deep learning to video stabilization. Xu et al.[18] uses the adversarial network to generate a target image to guide the frame warping. Wang et al.[15] uses a two branch Siamese network to generate a grid to warp the video frames. These networks take color frames as input and are trained with the DeepStab dataset, which contains stable and unstable video pairs. Deep learning methods enable near real-time performance in video stabilization. Visually, the results of these works are not as good as traditional methods. There are two potential reasons for the weak performance of deep learning in video stabilization. First, the video stabilization is a spatial transformation problem. The color images contain rich texture information, but the inter-frame spatial relation remains vague. Wang et al.[15] uses ResNet50 directly without any consideration of spatial transformation. Xu et al.[18] added spatial transformer modules to the adversarial network, but training a single network to infer spatial transformation of multiple frames only from color frames is difficult. Second, the dataset used in the training is not large enough. To our knowledge, the DeepStab dataset[18] is the only dataset for the learning of video stabilization and only contains 60 videos. For each video, the color frames are highly similar. Training an RGB based network with this dataset is essentially overfitting. Instead of trying to solve the video stabilization in an end-to-end fashion, we separate the task into two parts. We first use FlowNet2[6] to compute the spatial correspondence between frames, then train a network to smooth the motion fields provided by FlowNet2. This makes the training easier and yields better results compared to networks trained end-to-end.

3. Pipeline

The pipeline of our algorithm is shown in Fig. 1. Stage 1 is the pre-processing. We remove the large motions in the video in the first step. We compute SURF features[1] and their matches between consecutive frames, then compute the affine transformations. The translation and rotation components of the affine transformations are smoothed by a simple moving average with a window size 40. The frames are transformed using the affine transformation to obtain the smoothed positions. The optical flow is computed with the state-of-the-art neural network FlowNet2[6] on the smoothed video sequence. The purpose of removing large motions is to increase the accuracy of the optical flow. In Fig. 2(a), we show a visual comparison of the final result versus only using the raw input. Large motion reduction helps avoid large displacement in the optical flow and warp fields, which usually

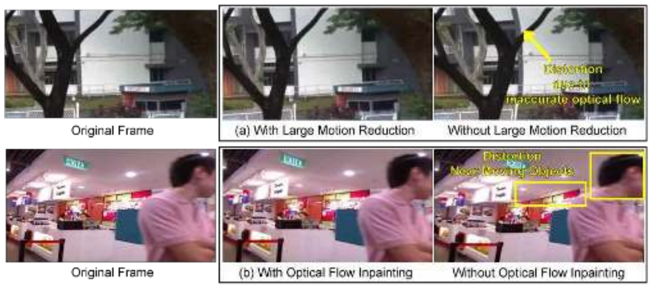

Original Frame
(a) With Large Motion Reduction
Without Large Motion Reduction
Distortion due to inaccurate optical flow
Original Frame
(b) With Optical Flow Inpainting
Without Optical Flow Inpainting
Distortion Near Moving Objects

Figure 2. *The visual comparison of the results (a)with and without large motion reduction, (b)with and without masking and optical flow inpainting. The results contain distortion if large motion is not removed since the optical flow is not accurate. The distortion is also introduced by the moving object, if we do not use masks and inpaint the moving object regions.*

introduce distortion in the results.

In the next step, based on a few criteria which will be discussed in Sec. 4.1, we generate a mask for each frame indicating the region where the optical flow is accurate. We inpaint the inaccurate regions using the first 5 principal components proposed in PCA Flow[17]. The coefficients are computed by fitting the principal components to the valid regions. In Fig. 2(b), we show an example using the raw optical flow without masking and inpainting. The person introduces significant distortion in the background due to the motion discontinuity. The analysis of the cause of this artifact and the details of motion inpainting will be discussed in Sec. 4.2.

The second stage is our stabilization network. The input of the network is the inpainted optical flow field. The network generates a per-pixel warp field for each frame, which compensates for the frame motion. In Sec. 5, we will discuss the loss function(Sec. 5.1) and the training process(Sec. 5.2).

The third stage is the post-processing. Since the optical flow in the invalid regions is inpainted, local discontinuities can be introduced at the valid/invalid boundaries. To ensure the continuity in the warp field, similar to stage 1, we fit the first 5 principal components to the warp fields in the valid regions. However, in stage 3, we replace the raw warp fields with the resulting low-frequency fits. We will discuss the necessity of this step in Sec. 6.1. Finally, we use the low-frequency warp fields to warp the input video. The warped video is cropped to a rectangle as the output.

4. Pre-Processing

In Sec. 3, we introduced the 3 stages of our pipeline: pre-processing(stage 1), stabilization network(stage 2) and warp field smoothing(stage 3). For stage 1, we discussed the large motion reduction and the optical flow computation in Sec. 3. In this section, we demonstrate the mask generation and the PCA Flow fitting in stage 1.

4.1. Mask Generation

As we discussed in Sec. 1, using optical flow as the reference in video stabilization potentially suffers from reliability issues. We summarize these problems into four types which are shown in Fig. 3: **1)** Motions of moving objects do not match frame motion. **2)** Inaccurate at moving object boundaries. **3)** Inaccurate in uniform color regions due to the lack of motion information. **4)** Large motion of still objects due to parallax.

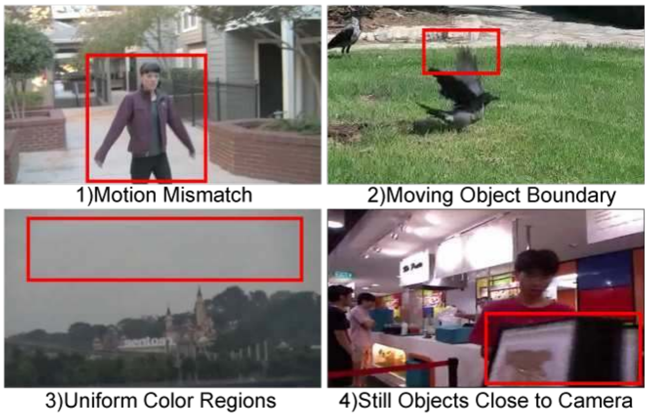


Figure 3. *Four types of scenarios in which optical flow can potentially be inaccurate or cause problems. Red boxes indicate example regions we refer to.*

Our goal is to identify these regions, and generate a mask M so that $M = 0$ for these regions and $M = 1$ otherwise.

Denote the optical flow from frame I_n to frame I_{n+1} as $\mathbf{F}_n$. To detect type 1 regions, we use the pre-trained semantic segmentation network[21, 20] to detect 11 kinds of possible dynamic object regions in I_n: person, car, boat, bus, truck, airplane, van, ship, motorbike, animal and bicycle. Note that these objects are not necessarily moving in the scene. Therefore, in these regions, we set $M_n(\mathbf{p}) = 1$ for any pixel $\mathbf{p}$ that satisfies $\|\mathbf{F}_n(\mathbf{p}) - \overline{\mathbf{F}_n}\|_2 < 5$, where $\overline{\mathbf{F}_n}$ is the mean motion of the entire frame.

In type 2 regions, the value of the optical flow changes significantly, causing a large local standard deviation. We compute the moving standard deviation with a 5×5 window, forming the standard deviation map $\Delta\mathbf{F}_n$. We set $M_n(\mathbf{p}) = 0$ if $\Delta\mathbf{F}_n(\mathbf{p}) > 3\overline{\Delta\mathbf{F}_n}$ where $\overline{\Delta\mathbf{F}_n}$ is the mean standard deviation map value.

To detect type 3 regions, we compute the gradient image of frame I_n, denoted as ∇I_n. We set $M_n(\mathbf{p}) = 0$ if $\nabla I_n < 8$, since a smaller gradient value indicates less color variation.

For type 4 regions, we simply set $M_n(\mathbf{p}) = 0$ if $\|\mathbf{F}_n(\mathbf{p}) - \overline{\mathbf{F}_n}\|_2 > 50$, since the motion can only be very large in a large motion removed video if the object is very close to the camera.

We show sample masks generated using the metrics above in Fig. 4.

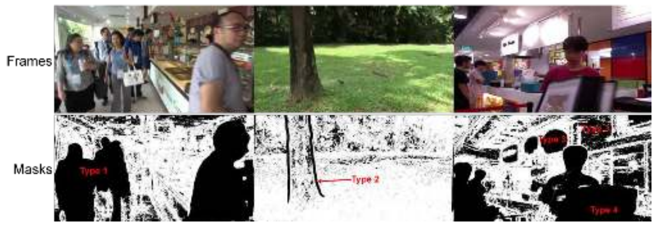


Figure 4. *Sample masks generated using our metrics described in Sec. 4.1. The four types of invalid regions are marked in the mask images.*

4.2. PCA Flow Fitting

To inpaint the motion vectors in the $M_n = 0$ regions, we fit the first 5 principal components proposed by PCAFlow[17] to the $M_n = 1$ regions. Since the first 5 principal components of PCAFlow are spatially smooth, we can expect the $M_n = 0$ regions are filled with reasonable values that obey the overall optical flow field. We reshape and stack the horizontal and vertical principal components into matrices $\mathbf{Q}_x$ and $\mathbf{Q}_y \in \mathbb{R}^{wh\times 5}$ respectively, where wh is the frame size. Similarly, we also reshape the optical flow field to $\mathbf{F}_{n,x}$ and $\mathbf{F}_{n,y} \in \mathbb{R}^{wh\times 1}$. For simplicity, we omit the subscript x and y. The fits below are computed independently for the horizontal and vertical directions. For each frame with mask M_n, we select the corresponding rows in $\mathbf{Q}$ and $\mathbf{F}$ where $M_n = 1$, forming the frame-specific principal components $\tilde{\mathbf{Q}}_n$ and valid optical flow matrix $\tilde{\mathbf{F}}_n$. Finding the coefficients $\mathbf{c_n} \in \mathbb{R}^{5\times 1}$ to fit the valid optical flow $\tilde{\mathbf{F}}_n$ forms a traditional least squares problem:

$$\min_{\mathbf{c}_n} \left\| \tilde{\mathbf{Q}}_n \mathbf{c_n} - \tilde{\mathbf{F}}_n \right\|_2 + \eta \left\| \mathbf{c}_n \right\|_2 \tag{1}$$

where $\eta = 0.1$ is the regularization term. The solution of this problem is:

$$\mathbf{c}_n = (\tilde{\mathbf{Q}}_n^T \tilde{\mathbf{Q}}_n + \eta \mathbf{I})^{-1} \tilde{\mathbf{Q}}_n^T \tilde{\mathbf{F}}_n \tag{2}$$

We replace the optical flow values in $M_n = 0$ regions with the fitted PCA Flow $\mathbf{Q}_n \mathbf{c}_n$. The PCA Flow inpainted optical flow matrices for the horizontal and vertical directions are combined and reshaped back to the inpainted optical flow field $\widehat{\mathbf{F}}_n \in \mathbb{R}^{w\times h\times 2}$.

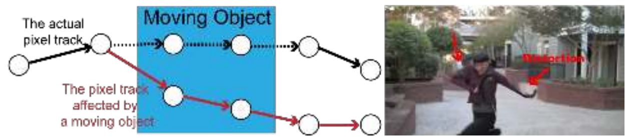


Figure 5. *The effect imposed by moving objects. The circles represent pixels and the arrows represent motion vectors evolving with time. The pixel can deviate from the actual track due to the moving object, resulting in a wrong warp field. The image on the right shows an example. The red arrows point out the distortion introduced by the moving object.*

4.3. Discussion

We demonstrate the necessity of using the mask in Fig. 5, in which we depict a 1D abstraction of the optical flow sequence. The moving object can cause a deviation in the motion vector that enters its region from the background, leading to a different pixel track from the actual motion pattern. Stabilizing the video in this scenario introduces distortion.

Applying the mask and stabilizing the valid regions alone still introduces distortion around moving objects. Figure 6 depicts an example of stabilizing only the valid regions. The mask M_n breaks the pixel track, making the pixels that connect to the masked pixels now only connect to one correspondence. These pixels can move freely, causing distortion artifacts around the masked moving objects. Therefore, we need to inpaint the optical flow in the $M_n = 0$ regions so that the pixels connecting to these regions are constrained properly.

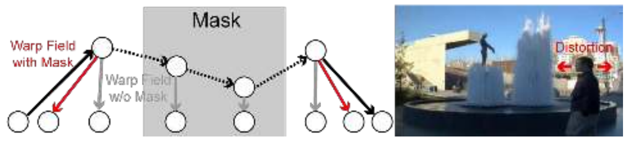


Figure 6. *Only stabilizing the valid regions will cause distortion in the warp field (red arrows) since the pixels are only constrained by the valid pixels connecting to it. The image on the right shows an example of this case.*

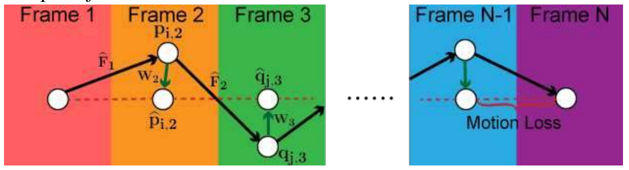


Figure 7. *A 1D abstraction of the motion loss. The loss indicates the average distance between corresponding pixels in each frame.*

5. Network and Training

In this section, we introduce our video stabilization network. Our network follows the structure proposed by Zhou et al.[22]. The network has a fixed number of input channels and can only take a segment of the optical flow sequence. Intuitively, the stabilization can handle low-frequency shake better if more frames are stabilized together since the network can access more global motion information. On the other hand, processing more frames together leads to a larger number of network weights and more difficulty in training. Taking all these factors into consideration, we use 20 frames of optical flow fields as the input of our network(representing the motion of a 21-frame video segment). Our network infers 19-frame warp fields for the video frames, excluding the first and the last frame. In the network structure of Zhou et al.[22], we set the number of input channels of layer *conv1_1* to 20 and the number of output channels of layer *conv7_3* to 19. In Sec. 5.1, we define a loss function that enables the training of this network for our application. We will also introduce the training process of our network in Sec. 5.2. Note that to make our network be able to stabilize arbitrary long videos, we propose a sliding window schedule that will be discussed in Sec. 6.2.

5.1. Loss Functions

Denote a pixel at frame n as $\mathbf{p}_i$, where we unroll the pixel coordinates to index i. As discussed in Sec. 4.2, denote the PCA Flow inpainted optical flow from frame n to frame $n+1$ as $\widehat{\mathbf{F}}_n$. By definition, the correspondence of $\mathbf{p}_{i,n}$ in frame $n+1$, $\mathbf{q}_{j,n+1}$, can be represented as:

$$\mathbf{q}_{j,n+1} = \mathbf{p}_{i,n} + \widehat{\mathbf{F}}_n(\mathbf{p}_{i,n}) \tag{3}$$

Denote the output warp field for frame n as W_n. The warped position of a pixel $\mathbf{p}_{i,n}$ is defined as:

$$\widehat{\mathbf{p}}_{i,n} = \mathbf{p}_{i,n} + \mathbf{W}_n(\mathbf{p}_{i,n}) \tag{4}$$

Similarly, the warped position of its correspondence $\mathbf{q}_{j,n+1}$ an be written as:

$$\widehat{\mathbf{q}}_{j,n+1} = \mathbf{q}_{j,n+1} + \mathbf{W}_{n+1}(\mathbf{q}_{j,n+1}) \tag{5}$$

For a video segment with N frames, the number of optical flow fields between consecutive frames is $N-1$. Intuitively, our goal is to apply the warp field to every pixel so that the distance between correspondences are minimized. In Fig. 7, we depict a 1D abstraction of the motion loss. Note that we must fix the warp field to zero for the first and the last frame, i.e. $\mathbf{W}_1 = \mathbf{W}_N = 0$. In other words, the network only produces the warp field for the intermediate frames. Therefore, we seek to find the shortest path to move the pixels in the first frame to their destination in the last frame instead of aligning all the frames. We define the motion loss as:

$$L_m = \sum_{n=1}^{N-1} \sum_{i} \|\widehat{\mathbf{p}}_{i,n} - \widehat{\mathbf{q}}_{j,n+1}\|_2 \tag{6}$$

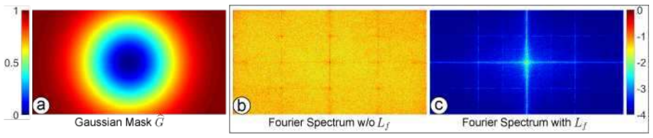


Figure 8. *The inverted Gaussian map ⓐ and an example spectrum of a warp field estimated with ⓒ and without ⓑ the frequency domain loss. The magnitude of the spectrum is shown in the $\log_{10}$ domain. The network can learn to produce a significantly smoother warp field with L_f ⓒ.*

In addition, we also seek to enforce the spatial smoothness of the warp fields. There are various kinds of constraints for enforcing spatial smoothness, e.g. total variation and the linear warp field constraint proposed by Yu and Ramamoorthi[19]. However, the total variation constraint is strong in constraining local noise but weak in constraining distortions. The linear warping constraint is difficult to control since a strong constraint limits the warping flexibility to handle large scale non-linear motions, while a weak constraint will not constrain local distortions properly. In our method, we seek to unify the need for suppressing warp field noise and avoiding local distortion without affecting the flexibility of compensating global motions. Therefore, we propose to constrain the warp field in the frequency domain. Intuitively, the noise usually increases the high-frequency energy, while the local distortion increases the mid-frequency energy. The goal is to increase the low-frequency energy in the warp field, encouraging global warping and suppressing local warping and noise. This can be achieved by weighting the Fourier transform of the warp field in the training process. Computationally, we compute the 2D Fourier transform of each output warp field, then weight the spectrum by an inverted Gaussian map shown in Fig. 8. In our experiment, we generate the Gaussian map $\mathbf{G}$ with $\mu = 0$ and $\sigma = 3$, inverted by its maximum value, and normalized by the maximum value:

$$\widehat{\mathbf{G}} = (\max(\mathbf{G}) - \mathbf{G}) / \max(\mathbf{G})$$

The frequency domain loss is defined as:

$$L_f = \sum_{n=2}^{N-1} \left\| \widehat{\mathbf{G}} \cdot \mathcal{F}\mathbf{W}_n \right\|_2 . \tag{7}$$

In this equation, the Fourier spectrum of the output warp field $\mathcal{F}\mathbf{W}_n$ is also normalized by its maximum value. Also note that the DC term of the inverted Gaussian map $\widehat{\mathbf{G}}$ is not used since we only encourage a low-frequency warp field but not a uniform warp field.

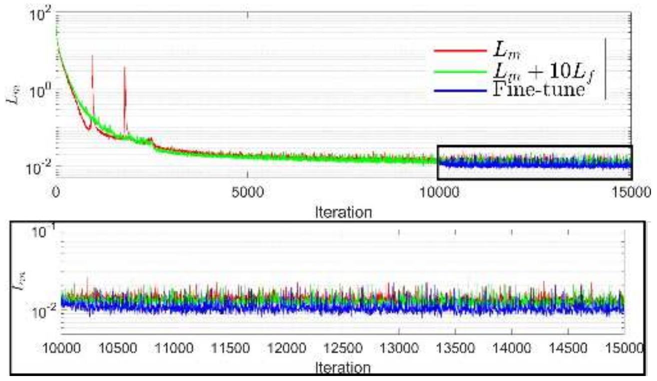


Figure 9. *The comparison of the motion loss L_m in different training schedules. The x-axis is the number of iterations and the y-axis is the L_m value. The figure below is a zoom-in version of the black box region of the upper figure. The red curve represents the training with L_m only. The green curve represents the training with $L_m + 10 * L_f$. The blue curve represents our training schedule. The frequency domain loss helps the first training phase so that the fine-tuning phase can achieve a lower motion loss.*

In the following section, we will discuss the usage of these loss functions in the training process.

5.2. Training

Dataset For the training of our network, we need a dataset with a large number of unstable videos. Existing video stabilization datasets, DeepStab[15](60 videos) and NUS[10](174 videos), do not contain enough motion pattern and color variation. In our training phase, we select the RealEstate10K[22] dataset which contains stable videos with a large number of color variations. For each training sample, we randomly select 20 frames from a random video. To produce an unstable video, we simply perturb every frame other than the first frame and last frame using random 2D affine transformation. The parameters of this random 2D affine transformation are: scaling $U[0.9, 1.1]$, translation (percentage w.r.t the frame size) $U[-5\%, 5\%]$, rotation $U[-5°, 5°]$ and shear $U[-5°, 5°]$. The perturbed video forms the input of our network.

Training Phases We summarize our training process into two phases. In the first phase, we set the loss function as:

$$L_1 = L_m + 10 * L_f \tag{8}$$

After training for 10000 iterations, we enter the second training phase, in which we switch the loss function to:

$$L_2 = L_m \tag{9}$$

and fine-tune the network for another 5000 iterations. We use the Adam optimizer[7] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate is set to 10^{-4} for the first 2500 iterations and fixed to 10^{-5} for the rest of the training process.

To justify this training schedule, in Fig. 9, we plot the value of residual motion L_m which mainly indicates the

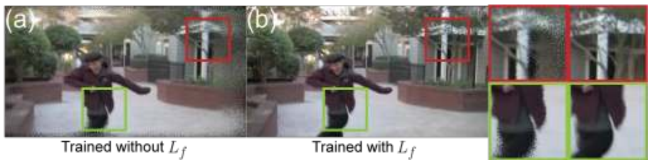


Figure 10. *The visual comparison of (a)the warped frames using the raw outputs of the networks trained with and (b)without L_f. The red and green boxes indicate the noisy regions. The frequency domain loss helps to improve the quality of the warp field.*

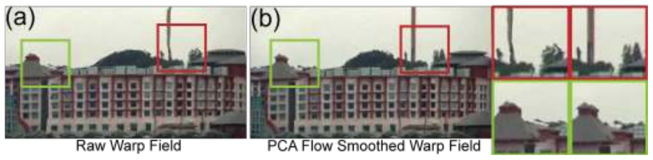


Figure 11. *The visual comparison of (a)the frames warped with the raw warp field and (b)the PCA Flow smoothed warp field. Due to the inpainting of the optical flow, the raw warp field may contain artifacts at the valid/invalid region boundaries.*

training progress. The Case-I (red curve) represents the training with L_m only. The Case-II (green curve) represents the training with loss set to $L_m + 10 * L_f$. It can be observed that using L_f helps in making L_m descend to a lower value and expedite the training process. In Case-I, we observe that although the optical flow is spatially smooth, the output warp field usually contains high-frequency noise. The noise makes the network difficult to train in Case-I, especially in the early stages(spikes appear in the red curve). By introducing L_f to Case-I, we intend to suppress the high-frequency noise and reduce the local minima. After Case-II converges(iteration 10000), we switch back to Case-I to fine-tune the network. The blue curve in Fig. 9 shows that our schedule achieves the lowest loss level. Figure 10 shows an output frame comparison between training with L_m only and our training schedule. Using L_f makes the raw warp field smoother.

6. Testing and Implementation Details

In this section, we will discuss the details in the third stage and testing.

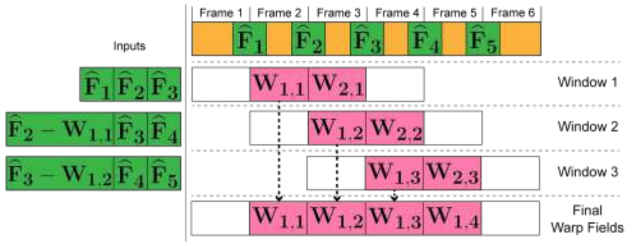
6.1. Warp Field Smoothing

Since the optical flow in the invalid regions is inpainted, our warp fields are only valid for the valid regions. The continuity of the warp field at valid/invalid boundaries is not guaranteed. Using the raw warp field introduces artifacts in the output, as shown in Fig. 11(a). Similar to the PCA Flow hole filling discussed in Sec. 4.2, we fit PCA Flow to the valid regions. In stage 3, we directly use the fitted PCA Flow as the warp field instead of the raw warp field. Figure 11(b) shows the result warped by the PCA Flow smoothed warp field.

6.2. Sliding Window

As discussed in Sec. 5, our network only takes 20 frames as the input. To handle a regular video, we use a sliding window approach for the testing phase.

The sliding window of our method works as shown in Fig. 12. In this figure, we use the notation $\mathbf{W}_{n,k}$ to represent the warp fields from different windows. The first index



Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Inputs
$\widehat{\mathbf{F}}_1$
$\widehat{\mathbf{F}}_2$
$\widehat{\mathbf{F}}_3$
$\widehat{\mathbf{F}}_4$
$\widehat{\mathbf{F}}_5$
$\widehat{\mathbf{F}}_1$ $\widehat{\mathbf{F}}_2$ $\widehat{\mathbf{F}}_3$
$\mathbf{W}_{1,1}$ $\mathbf{W}_{2,1}$
Window 1
$\widehat{\mathbf{F}}_2 - \mathbf{W}_{1,1}$ $\widehat{\mathbf{F}}_3$ $\widehat{\mathbf{F}}_4$
$\mathbf{W}_{1,2}$ $\mathbf{W}_{2,2}$
Window 2
$\widehat{\mathbf{F}}_3 - \mathbf{W}_{1,2}$ $\widehat{\mathbf{F}}_4$ $\widehat{\mathbf{F}}_5$
$\mathbf{W}_{1,3}$ $\mathbf{W}_{2,3}$
Window 3
$\mathbf{W}_{1,1}$ $\mathbf{W}_{1,2}$ $\mathbf{W}_{1,3}$ $\mathbf{W}_{1,4}$
Final Warp Fields

Figure 12. *The sliding window schedule for processing arbitrary long videos. For each window, we only accept the warp field for the second frame, e.g.* $\mathbf{W}_{1,1}$ *in window 1. In the next window(window 2), the inpainted optical flow from the first frame to the second frame(*$\widehat{\mathbf{F}}_2$*) is modified using the accepted warp field from the previous frame(*$\widehat{\mathbf{F}}_2 - \mathbf{W}_{1,1}$*).*

n is the frame number within a window, and the second index k is the window index. For each 20-frame window, the warp field for the first frame is already known from the previous window. We update the original optical flow as $\widehat{\mathbf{F}}_k - \mathbf{W}_{1,k-1}$ since the starting point of the motion vector is moved by $\mathbf{W}_{1,k-1}$ in the previous window. The updated optical flow is concatenated with the other optical flow fields as the input of the network, as shown on the left of window 2 and 3 in Fig. 12. We only use the first warp field produced by the network output to fit the principal component and warp the second video frame. Then the window slides to the next frame and the process above repeats.

The disadvantage with the sliding window is that we are using 20 frames ahead of the current frame. However, the optical flow for these frames will be updated in the future windows, which should influence the current frame as well. For offline video stabilization, we can process the video with the sliding window for multiple passes. Between two passes, we re-compute the optical flow using the warped frames.

7. Results

In this section, we compare the results of our method with the state-of-the-art video stabilization methods. These methods are selected since they represent different approaches to the video stabilization problem. Grundmann et.al.[5] uses the full-frame homography as the motion model and warping method. Liu et.al.[10] uses a grid to analyze the local frame motion and warp the frames. Yu and Ramamoorthi[19] use the dense optical flow as the motion model, and optimize a set of CNN parameters that produce the pixel-wise warp field for each segment of a video. These methods belong to traditional optimization methods since they use traditional optimization and have to be re-run for a new video. We also compare with the most recent deep learning based method, Wang et.al.[15], which uses a pre-trained network to directly infer a warp grid from colored input frames. We compare the results both visually and quantitatively.

Visual Comparison We provide visual comparisons in the supplementary video since most of the artifacts are only visible in the video. For the visual comparison of video stills, we selected a few difficult scenarios for the video stabilization task. Figure 13 shows the comparison of video stills from the comparison methods. Example 1 contains parallax effects with moving occlusions. Since Liu et.al.[10] uses a

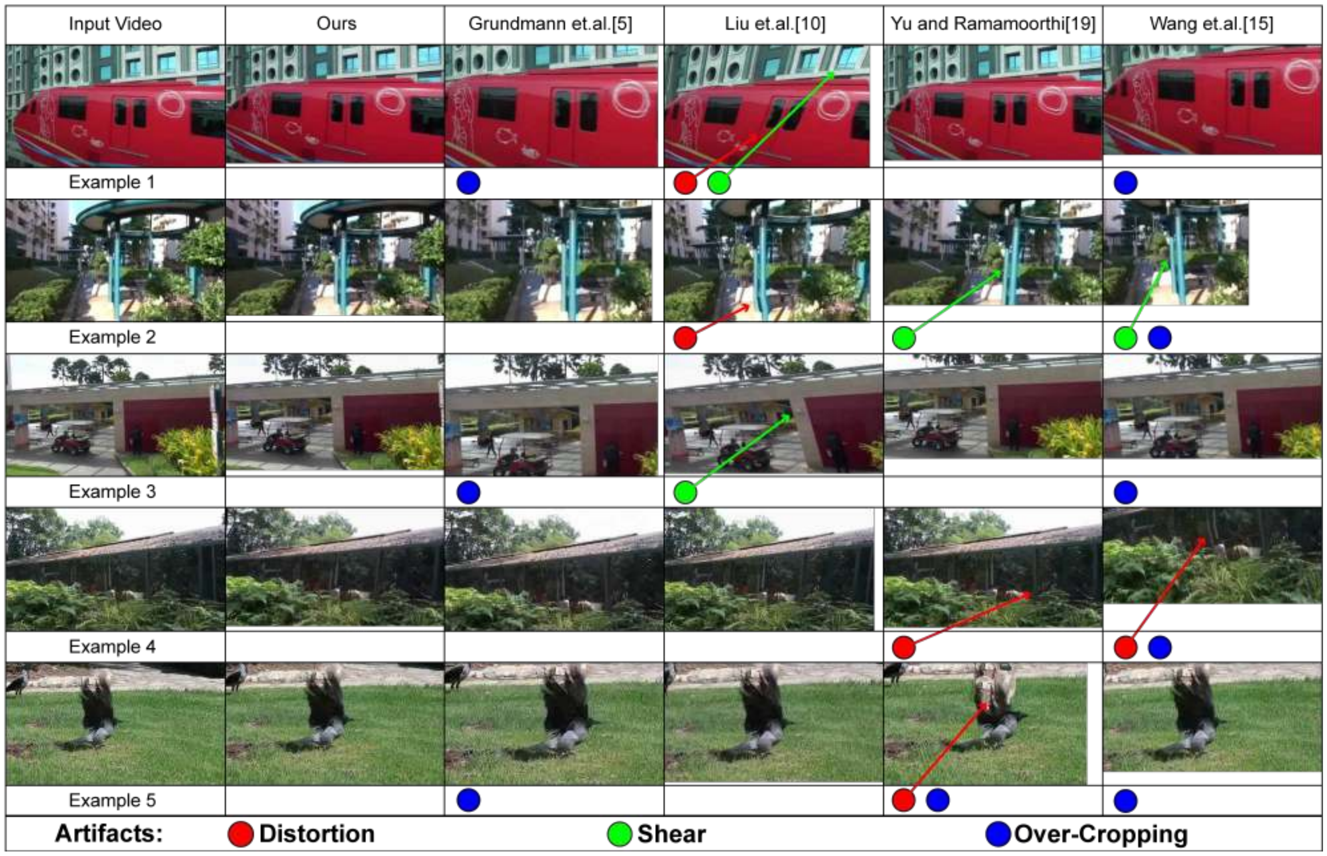
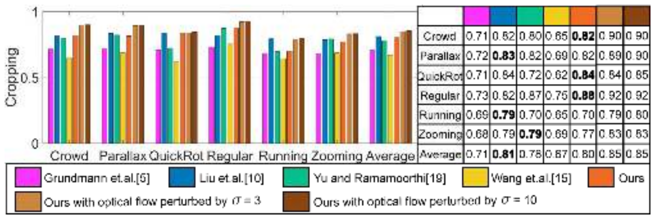


Figure 13. *The visual comparison of Grundmann et.al.[5], Liu et.al.[10], Yu and Ramamoorthi[19], Wang et.al.[15] and our method. The artifacts are noted below the video stills and pointed out by arrows. To avoid introducing extra distortion, all the video stills are scaled while keeping the original aspect ratio.*

grid to warp the video, the region enclosed by a single cell is warped by the same homography. Therefore it generates a shear at the motion boundaries. It also introduces distortion in the uniform color regions(the body of the train), since estimating homography in these regions is difficult. Example 2 involves complex occlusions. Grid warping based methods, Liu et.al.[10] and Wang et.al.[15], produce local distortion due to motion mismatch. Yu and Ramamoorthi[19] introduce shear since their linear warping constraint enforces strong rigidity on the warp field, which tries to compensate for the motion. Our PCA Flow smoothed warp field provides more flexibility in warping compared to the grid used by Liu et.al.[10] and Wang et.al.[15], and the linear warping constrained warp field proposed by Yu and Ramamoorthi[19]. Example 3 provides another example where Liu et.al.[10] produces shear at motion boundaries. Example 4 contains complex structures and Example 5 contains quick object motion. Both are challenging for optical flow based video stabilization methods. Yu and Ramamoorthi[19] produce significant distortion in these cases, since they fail to constrain the local region in the warp field. Our PCA Flow based warp field avoids drastic compensation to optical flows and does not introduce artifacts in local regions.

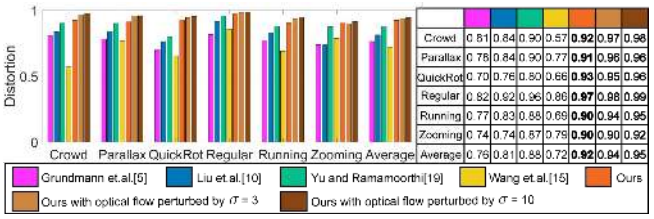
The 2D full-frame homography method of Grundmann et.al.[5] performs well on keeping original frame appearance, but in the supplementary video we will show that their temporal stability is inferior to that of comparison methods in the examples shown in Fig. 13. The pre-trained model proposed by Wang et.al.[15] failed to generate good results in most of the videos, due to the difficulty in generalization.



	🟪	🟦	🟩	🟨	🟧	🟫	🟫
Crowd	0.71	0.82	0.80	0.65	**0.82**	0.90	0.90
Parallax	0.72	**0.83**	0.82	0.69	0.82	0.89	0.90
QuickRot	0.71	0.84	0.72	0.62	**0.84**	0.84	0.85
Regular	0.73	0.82	0.87	0.75	**0.88**	0.92	0.92
Running	0.69	**0.79**	0.70	0.65	0.70	0.79	0.80
Zooming	0.68	0.79	**0.79**	0.69	0.77	0.83	0.83
Average	0.71	**0.81**	0.78	0.67	0.80	0.85	0.85

Figure 14. *The cropping metric comparison of Grundmann et.al.[5], Liu et.al.[10], Yu and Ramamoorthi[19], Wang et.al.[15] and our method. Each value is the result averaged by the category in the NUS dataset[10]. The last bar group is the average over all videos. The quantative values of the bars are shown in the table on the right. A larger value indicates a better result. The actual best result of each category before rounding is marked in bold font.*

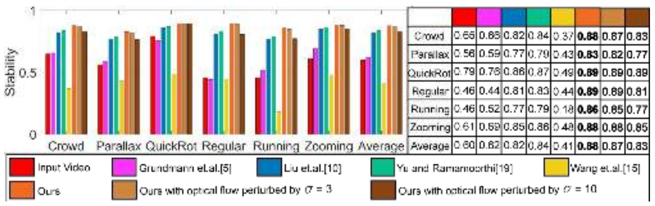
Quantative Comparison For quantative comparison, we use the metrics proposed in Liu et.al.[10] to evaluate the quality of the results over the entire NUS dataset[10]. The values are averaged over each category. The cropping metric measures the frame size loss of the output video due to the warping and cropping. A larger value indicates a better frame size preservation. Figure 14 shows the cropping comparison. Our method maintains a large frame size similar to Liu et.al.[10] and Yu and Ramamoorthi[19], since the PCA Flow smoothed warp field does not introduce sharp warps that affect the cropping size. Our method is slightly worse than Liu et.al.[10] in the Running category since we have the large motion reduction step. The full-frame affine transformation removes large motions in the Running videos, but also leads to a smaller overlapping area and the final frame size. This can be easily avoided by using a smaller window size in the large motion reduction step.



	🟪	🟦	🟩	🟨	🟧	🟫	🟫
Crowd	0.81	0.84	0.90	0.57	**0.92**	0.97	0.98
Parallax	0.78	0.84	0.90	0.77	**0.91**	0.96	0.96
QuickRot	0.70	0.76	0.80	0.66	**0.93**	0.95	0.96
Regular	0.82	0.92	0.96	0.86	**0.97**	0.98	0.99
Running	0.77	0.83	0.88	0.69	**0.90**	0.94	0.95
Zooming	0.74	0.74	0.87	0.79	**0.90**	0.90	0.92
Average	0.76	0.81	0.88	0.72	**0.92**	0.94	0.95

Figure 15. *The distortion metric comparison of Grundmann et.al.[5], Liu et.al.[10], Yu and Ramamoorthi[19], Wang et.al.[15] and our method. Each value is the result averaged by the category in the NUS dataset[10]. The last bar group is the average over all videos. The quantative values of the bars are shown in the table on the right. A larger value indicates a better result. The actual best result of each category before rounding is marked in bold font.*

The distortion metric measures the anisotropic scaling that leads to distortion in the result frames. A larger value indicates better preservation of the original shape of the objects in the video. Figure 15 shows the comparison of the distortion metric. Our per-pixel warp field introduces less distortion than the grid warping used in Liu et.al.[10] and Wang et.al.[15], and the full-frame homography used in Grundmann et.al.[5] in all the categories. Our method has less anisotropic scaling compared to Yu and Ramamoorthi[19], since our warp field is more flexible than their linear warping constrained warp field. Therefore, our method performs better in preserving the original shape of the objects in the video. It can be also seen in the visual comparison that our PCA Flow smoothed warp field introduces less local distortion. Note that the comparison deep learning based method Wang et.al.[15] performs the worst in all the categories, implying the difficulty in generalization.



	🟥	🟪	🟦	🟩	🟨	🟧	🟫	🟫
Crowd	0.65	0.65	0.82	0.84	0.37	**0.88**	0.87	0.83
Parallax	0.56	0.59	0.77	0.79	0.43	**0.83**	0.82	0.77
QuickRot	0.79	0.76	0.86	0.87	0.49	**0.89**	0.89	0.89
Regular	0.46	0.44	0.81	0.83	0.44	**0.89**	0.89	0.81
Running	0.46	0.52	0.77	0.79	0.18	**0.86**	0.85	0.77
Zooming	0.61	0.59	0.85	0.86	0.48	**0.88**	0.88	0.85
Average	0.60	0.62	0.82	0.84	0.41	**0.88**	0.87	0.83

Figure 16. *The stability metric comparison of Grundmann et.al.[5], Liu et.al.[10], Yu and Ramamoorthi[19], Wang et.al.[15] and our method. Each value is the result averaged by the category in the NUS dataset[10]. The last bar group is the average over all videos. The quantative values of the bars are shown in the table on the right. A larger value indicates a better result. The actual best result of each category before rounding is marked in bold font.*

The stability metric measures the stability of the output video. A larger value indicates a more visually stable result. Our method achieves a better stability value compared to optimization based methods Grundmann et.al.[5], Liu et.al.[10] and Yu and Ramamoorthi[19]. Our results are also significantly more robust than the deep learning based method Wang et.al.[15], since their results are even more unstable than the input video from the NUS dataset[10]. We will also show in the supplementary video that we achieve better stability values with less artifacts compared to these methods.

Table 1. Per-frame run time comparison

Grundmann et.al.[5]	480ms
Liu et.al.[10]	1360ms
Yu and Ramamoorthi[19]	1610ms
Wang et.al.[15]	460ms
Ours	570ms

To evaluate the robustness of our method, we also conduct experiments with inaccurate optical flow. We added Gaussian random noise with standard deviation $\sigma = 3$ and $\sigma = 10$ on the input optical flow to the network. In Fig. 14, Fig. 15 and Fig. 16, we observe that the inaccuracy in the optical flow leads to less cropping and distortion but worse stability. This indicates that the more inaccuracy in the optical flow, the more regions are identified as invalid regions in the stage 1 of our method. The network tends to warp the frame less since it receives less motion information. As shown in Fig. 16, our method is robust to inaccurate optical flow. Our method still maintains comparable level of stability even with optical flow perturbed by Gaussian noise with $\sigma = 10$.

Also note that since our method is a deep learning based method, the speed of our method is faster than the optimization based methods. Our network and pipeline are implemented with PyTorch and Python. Table. 1 is a summary of per-frame runtime for comparison methods. All the timing is performed on a desktop with an RTX2080Ti GPU and an i7-8700K CPU. On average, our unoptimized method takes 270ms in stage 1 and 300ms in stage 2 and 3. Our method achieves better visual results and somewhat better quantitative results compared to optimization based methods Liu et.al[10] and Yu and Ramamoorthi[19] but gives $\sim$3x speed up. Our method has only a slight computation time loss compared to the simple 2D method of Grundmann et.al.[5] and deep learning based method of Wang et.al.[15], but generates significantly better visual and quantative results.

8. Conclusions and Future Work

In this paper, we proposed a novel deep learning based video stabilization method that infers the pixel-wise warp field for stabilizing video frames from the optical flow between consecutive frames. We also proposed a pipeline that detects invalid regions in the optical flow field, inpaints the invalid regions and smoothes the output warp field. The results show that our method is more robust than existing deep learning based methods and achieves visually and quantitatively better results compared to the state-of-the-art optimization based methods with a $\sim$3x speed improvement. Future works would be an end-to-end network that directly converts input videos to stabilized videos, and a dataset that enables the training of such a network.

Acknowledgements. We thank Qualcomm for supporting this research with a Qualcomm FMA Fellowship.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *ECCV*, 2006.

[2] Christopher Buehler, Michael Bosse, and Leonard McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, 2001.

[3] Michael L. Gleicher and Feng Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(1), Oct. 2008.

[4] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 31(5), Sept. 2012.

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

[6] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE CVPR*, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[8] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph.*, 28(3), July 2009.

[9] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1), Feb. 2011.

[10] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4), July 2013.

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

[12] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.(PAMI)*, 28(7), July 2006.

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

[14] Jian Sun. Video stabilization with a depth camera. In *IEEE CVPR*, 2012.

[15] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 2019.

[16] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee. Spatially and temporally optimized video stabilization. *IEEE Trans. Visual. and Comput. Graph.*, 19(8), Aug 2013.

[17] Jonas Wulff and Michael J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE CVPR*, 2015.

[18] Sen-Zhe Xu, Jun Hu, Miao Wang, Tai-Jiang Mu, and Shi-Min Hu. Deep video stabilization using adversarial networks. *Computer Graphics Forum*, 2018.

[19] Jiyang Yu and Ravi Ramamoorthi. Robust video stabilization by optimization in cnn weight space. In *IEEE CVPR*, 2019.

[20] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *IEEE CVPR*, 2017.

[21] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.

[22] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM Trans. Graph.*, 2018.