3D Video Stabilization with Depth Estimation by CNN-based Optimization

Yao-Chih Lee[1] Kuan-Wei Tseng[2] Yu-Ta Chen[2] Chien-Cheng Chen[2]
Chu-Song Chen[2] Yi-Ping Hung[2]
[1]Academia Sinica, Taiwan [2]National Taiwan University
[1]yclee1231@iis.sinica.edu.tw, [2]{kwtseng,r07922120,r09944015,chusong,hung}@csie.ntu.edu.tw

Abstract

Video stabilization is an essential component of visual quality enhancement. Early methods rely on feature tracking to recover either 2D or 3D frame motion, which suffer from the robustness of local feature extraction and tracking in shaky videos. Recently, learning-based methods seek to find frame transformations with high-level information via deep neural networks to overcome the robustness issue of feature tracking. Nevertheless, to our best knowledge, no learning-based methods leverage 3D cues for the transformation inference yet; hence they would lead to artifacts on complex scene-depth scenarios. In this paper, we propose Deep3D Stabilizer, a novel 3D depth-based learning method for video stabilization. We take advantage of the recent self-supervised framework on jointly learning depth and camera ego-motion estimation on raw videos. Our approach requires no data for pre-training but stabilizes the input video via 3D reconstruction directly. The rectification stage incorporates the 3D scene depth and camera motion to smooth the camera trajectory and synthesize the stabilized video. Unlike most one-size-fits-all learning-based methods, our smoothing algorithm allows users to manipulate the stability of a video efficiently. Experimental results on challenging benchmarks show that the proposed solution consistently outperforms the state-of-the-art methods on almost all motion categories.

1. Introduction

Video stabilization removes the undesired shaky motion and preserves the primary motion from a video, which is a critical module to video acquisition and pre-processing. Video stabilization methods often estimate the motion between frames at first, and then the inter-frame motion estimated can be used in a smoothing algorithm for stabilizing the input video. Traditional video stabilization methods can be categorized into two types according to the dimensionality of the motion model. The 2D-motion methods model the tracked features with 2D transformations (*e.g.* affinity, homography). Earlier 2D-based approaches (such as [15, 10]) adopt full-frame 2D transformations and show promising stabilization qualities. Nonetheless, the full-frame 2D motion would suffer from spatially variant motion (*e.g.* parallax effect) caused by complex 3D scene structures. Some 2D-based methods [21, 19] divide the frame into grids and extract the 2D local motion of each grid to handle parallax effect. Despite the local motion models show better results than full-frame models, the local motion methods are still challenging since it is difficult to coordinate the grid-wise transformations to avoid the local distortions thus introduced. The 3D approaches [16, 14, 8, 20] acquire 3D information to model the frame motion in 3D space by using Structure-from-Motion (SfM) or additional sensors such as inertial measurement unit (IMU) and depth sensors. Hence, the 3D approach often requires expensive costs but shows superior results in contrast to the 2D approach.

The performance of earlier 2D and 3D approaches often rely on successful feature tracking for motion estimation. Yet it is challenging to obtain robust and long feature tracks, particularly from shaky videos. This restricts the performance of early approaches. The convolution neural network (CNN)-based video stabilization methods [29, 31, 35] learn to extract the frame motion in CNN parameter space and infer the stabilization transformation directly. Some learning-based methods train the networks on synchronized stable/unstable video pairs [29] to find frame transformations from ground-truth stable video directly. However, the application would be limited by the generalization capability as it is demanding to collect all types of unsteady videos for training in advance. On the other hand, these prior studies are based on 2D transformation and thus they still suffer from parallax effect and introduce distortions.

We introduce a novel 3D-based learning method for video stabilization. To the best of our knowledge, our proposed Deep3D Stabilizer is the first learning method that handles the stabilization process with learned 3D information. The pipeline of our method is provided in Figure 1. The CNNs jointly learn the scene depth and 3D camera motion for the input video during test time in an optimization

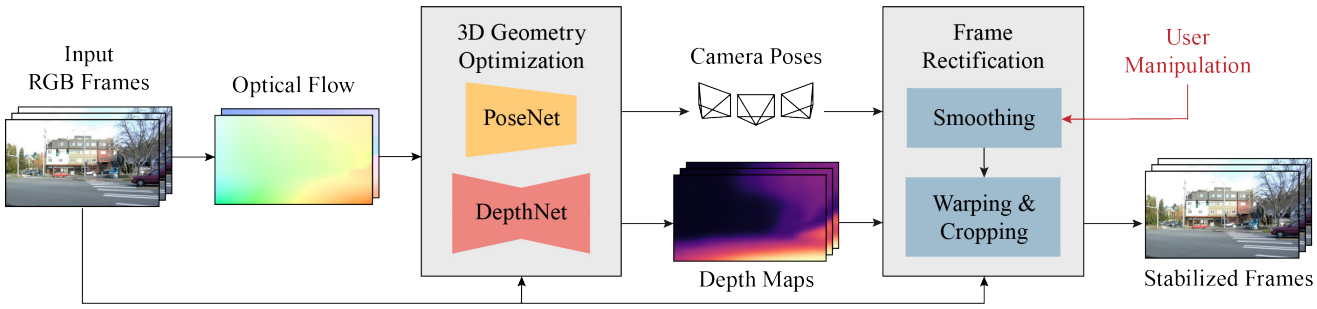

Figure 1. **Pipeline of the proposed method:** The pipeline consists of two stages. Firstly, the 3D geometry optimization stage estimates the 3D camera trajectory and the dense scene depth of the input RGB sequence with PoseNet and DepthNet, respectively, via test-time training. The optimization stage takes the input sequence and the corresponding optical flows as the guidance signal for learning the 3D scene. Secondly, the frame rectification stage takes the estimated camera trajectory and scene depth as input to perform view synthesis on a smoothed trajectory. The smoothing process enable users to manipulate the parameter of smoothing filter to attain different levels of stability of the resulted video, which is then wrapped and cropped to produce the stabilized video.

framework without needing training data. Our solution only requires input frames and the precomputed optical flow between consecutive frames as guidance signals for 3D geometry optimization. Consequently, the rectification stage takes the optimized 3D information to perform stabilization via smoothing the camera motion and reprojecting the input frames with scene depth. Our method can model the 3D scene structure and lead to low distortion, particularly when dealing with the videos with parallax effect. Moreover, our smoothing algorithm enables users to manipulate or alter the stability of the stabilized video online, which is a critical functionality for video stabilization while is overlooked by most previous learning-based studies since their methods are difficult to provide such freedom of manipulation. In sum, main contributions of our approach include:

- We introduce the first 3D-based deep CNN method for video stabilization without needing training data.
- Our approach can handle parallax effect more properly leveraging 3D motion model.
- Our stabilization solution allows users to manipulate the stability of a video in real-time (34.5 fps).

2. Related Works

Our approach is related to previous studies on *video stabilization*, *monocular depth estimation*, and test-time training. They are briefly reviewed as follows.

2.1. Video Stabilization

Video stabilization methods can be categorized into 2D, 3D, and deep learning approaches.

The 2D-based methods, which smooth the 2D linear transformations (*e.g.* affinity, homography) estimated between consecutive frames, require only a low computational complexity in general. Grundmann *et al.* [10] apply a full-frame 2D motion model and employ cinematography rules for motion design by using L1-norm optimization. Liu *et al.* [21] divide frames into grids to acquire bundled local transformations for spatially variant motions. Liu *et al.* [22] propose pixel profiles as a novel motion model rather than tranditional feature tracks so as to prevent from lost-tracking of features; they continue to present more efficient mesh profiles to enable real-time processing [19] and extend the work to video coding [18]. However, 2D approaches usually suffer from tackling parallax effects as 2D transformations are insufficient to model the entire 3D scene structure.

The 3D-based methods estimate 3D camera motions or scene structure from videos. Liu *et al.* [16] utilize 3D camera trajectory and sparse 3D point cloud reconstructed by Structure-from-Motion (SfM) to guide the warping. Liu *et al.* [17] utilize subspace constraints [13] on feature trajectories to enable more robust feature extraction. Goldstein and Fattal [8] exploit epipolar geometry to enhance the length of feature tracks. Some 3D methods utilize additional hardware sensors to capture 3D information. Smith *et al.* [27] and Liu *et al.* [20] employ light field and depth cameras for 3D projection, respectively. Karpenko *et al.* [14] use gyroscope for 3D rotation estimation. However, the 3D methods would either require expensive costs for 3D reconstruction or suffer from the robustness issue in SfM. The results would be easily fragile due to the failure of feature tracking.

Recent works explore deep learning models for video stabilization. Wang *et al.* [29] propose a supervised learning framework with inferring multi-grid 2D transformations. Xu *et al.* [31] apply adversarial networks to supervised learning with full-frame affine transformations. Zhao *et al.* [35] estimate pixel-wise warping maps in a supervised manner. These supervised methods are trained with DeepStab dataset [29] that contains 61 synchronized stable and unstable pairs of videos. However, they often suffer from

the confined generalization capability because of the limited variation of training videos in DeepStab.

In contrast, Yu and Ramamoorthi [33] directly perform test-time training with optical flow for stabilization but require extremely expensive computation time. They further propose a self-supervised method trained with only unstable videos based on optical flow field [34]. Choi and Kweon [4] train a frame interpolation network to smooth the frame motion in a self-supervised manner. The approach exploits the relationship between video stabilization and inter-frame interpolation, but an issue is that the interpolation easily yields severe distortion when the input video contains large motion. Moreover, these methods are all based on 2D information so that they could still suffer from parallax effect.

2.2. Monocular Depth Estimation

Besides the related works of video stabilization, our solution is also related to the automatic scene depth estimation with a monocular camera. Traditional depth estimation relies on the disparity estimated among multi-view images. With the recent development of deep neural networks, learning-based depth estimation methods have been introduced. Eigen *et al*. [5] employ a coarse-to-fine model with the ground truth depths captured by depth sensors as the supervision signal for single-image depth estimation. However, this method relies on the known depth for training, which restricts its applicability to unknown scene classes.

Garg *et al*. [6] and Gordard *et al*. [7] propose to solve the problem in a self-supervised manner, which use view synthesis with stereo training image pairs for single-image depth estimation problem. However, the settings require two cameras for self-supervision, which is unsuitable for a monocular video. Zhou *et al*. [37] propose a jointly self-supervised learning framework, where the monocular depth estimation and relative pose estimation can be learned from monocular sequences simultaneously. Further studies improve the jointly learning self-supervised framework via additional cues, such as optical flow [32, 38, 36], motion segmentation [25, 2, 9] and geometric constraints [24, 1, 3].

2.3. Test-Time Training

Learning directly from the testing data has been used to reduce the distribution difference between training and testing domains. Yu and Ramamoorthi [33] treat CNN as an optimizer to overfit a smoothing objective function to solve the video stabilization problem. Casser *et al*. [2] and Chen *et al*. [3] use testing sequences to fine-tune the pre-trained models to improve the monocular depth estimation results. Luo *et al*. [23] refine the pre-trained models with temporal consistency for sequential depth estimation.

A limitation of test-time training is its expensive computation time in test phase. Our method leverages the continuity of video, which propagates the learned weights for scene reconstruction to the subsequent snippets in a video, so as to reduce the optimization time.

3. Proposed Method

Given an unsteady sequence $\{I_t\}_{t=1}^N$ of N frames, the goal is to synthesize steady frames $\{I'_t\}_{t=1}^N$. The proposed pipeline mainly comprises two stages: *3D geometry optimization* and *frame rectification*. The *3D geometry optimization* stage learns the 3D geometry information including dense scene depth $\{D_t\}_{t=1}^N$ and 3D camera trajectory $\{P_t\}_{t=1}^N$ from $\{I_t\}_{t=1}^N$. The *frame rectification* stage reconstructs the stabilized output $\{I'_t\}_{t=1}^N$ by smoothing $\{P_t\}_{t=1}^N$ and performing view synthesis with $\{I_t\}_{t=1}^N$ and $\{D_t\}_{t=1}^N$. They are respectively presented in the following.

3.1. 3D Geometry Optimization

Inspired by the self-supervised framework of Zhou *et al*. [37], scene depth and camera ego-motion can be jointly trained by monocular RGB sequences. However, due to the difference between training and testing data distributions, a learned model on the training set often fails to be applied to the testing video. To address this issue, we exploit test-time training to the framework (as shown in Figure 2).

Moreover, inference of the depth and pose on separated clips cannot retain temporal consistency for an input sequence. To get temporally consistent sequential estimation, our framework contains two CNNs, DepthNet and PoseNet; both take a snippet $\{I_t\}_{t=k}^{k+T}$ of the input sequence with length T at each time. The length-N sequence is divided into multiple length-T snippets, with length-Ω overlap between nearby snippets. The networks learn the scene depth $\{D_t\}_{t=k}^{k+T}$ and camera pose $\{P_t\}_{t=k}^{k+T}$, respectively.

Consider one frame I_s in $\{I_t\}_{t=k}^{k+T}$ as a source view. When the scene depths and camera poses are estimated, we can use the source view to synthesize a novel target view in $\{I_t\}_{t=k}^{k+T}$ $(t \neq s)$ via the re-projection of D_s with the relative pose transformation $P_{s\to t} = P_t P_s^{-1}$. Let x_t and x_s denote the homogeneous coordinates of a pixel in I_t and I_s, respectively. The point transformation via 3D projection can be formulated as follows:

$$x_t = K P_{s\to t} D_s(x_s) K^{-1} x_s, \tag{1}$$

where K denotes the camera intrinsic parameters. Following this transformation, we can synthesize the entire novel view of any t from s, and we denote the synthesized view as $I_{s\to t}$. Similarly, we can also synthesize the optical-flow map from s to t. The estimated 2D projection of 3D scene flow (*i.e*. rigid flow) $F_{s\to t}$ from I_s to I_t can be obtained by

$$F_{s\to t}(x_s) = x_t - x_s. \tag{2}$$

Besides, via the rigid transformation $P_{s\to t}$, the depth map estimated for the source view s can be converted to the target view t as well, resulting in a synthesized depth map

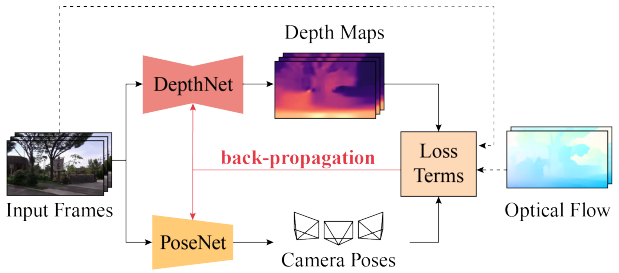

Figure 2. **3D geometry optimization framework:** The DepthNet and PoseNet estimate dense scene depth and camera pose trajectory based on a snippet of the input sequence. The loss terms measured via 3D reprojection are used to update the parameters of DepthNet and PoseNet via back-propagation in test-time.

$D_{s\to t}$ with the viewpoint of t's frame. The synthesized novel images ($I_{s\to t}$), optical-flow maps ($F_{s\to t}$), and depth maps ($D_{s\to t}$) are employed for conducting the loss terms enabling an end-to-end trainable model for simultaneous 3D scene reconstruction and camera ego-motion estimation.

Loss terms. The loss is composed of three terms: photometric loss $\mathcal{L}_P(t,s)$, geometric loss $\mathcal{L}_G(t,s)$, and optical-flow loss $\mathcal{L}_F(t,s)$. The photometric loss $\mathcal{L}_P(t,s)$ measures the appearance difference between the synthesized $I_{s\to t}$ and actual frame I_t using an L_1 component and structural similarity loss [30] with a trade-off weight $\gamma(=0.5)$,

$$\mathcal{L}_P(t,s) = \frac{1}{|V|}\sum_{x_t\in V}(1-\gamma)\|I_{s\to t}(x_t) - I_t(x_t)\|_1 + \gamma\frac{1-SSIM_{I_{s\to t},I_t}(x_t)}{2}, \tag{3}$$

where V represents the valid points successfully projected onto the image plane of I_t, and $|V|$ is the number of points.

The geometric loss proposed by Bian *et al.* [1] enforces the scale consistency of depth and pose estimation among consecutive frames. The geometric loss is formulated as:

$$\mathcal{L}_G(t,s) = \frac{1}{|V|}\sum_{x_t\in V}\frac{\|D_{s\to t}(x_t) - D_t(x_t)\|_1}{D_{s\to t}(x_t) + D_t(x_t)}, \tag{4}$$

where the difference of $D_{s\to t}$ and D_t is normalized by their sum to make $\mathcal{L}_G$ numerically steady in training [1].

In addition to DepthNet and PoseNet, inspired by Zhou *et al.* [36], we adopt a pre-trained CNN model for the optical flow estimation (as shown in Figure 2), which can help regularize the solutions found for depth and pose. The purpose of optical-flow loss is to strengthen the supervision on texture-less or small displacement areas, which are the weakness of photometric loss. The optical-flow loss directly computes the difference between the estimated $F_{s\to t}$ and the precomputed flow $\hat{F}_{s\to t}$ using pre-trained PWC-Net [28]. We use L_1 loss to build our optical flow error function,

$$\mathcal{L}_F(t,s) = \frac{1}{|V|}\sum_{x_t\in V}\|F_{s\to t}(x_t) - \hat{F}_{s\to t}(x_t)\|_1. \tag{5}$$

Multiple source views. Single-source projection consistency is fragile due to occlusion, illumination variation, *etc.* Therefore, we adopt a set of source views $\mathbf{S_t}$ for the 3D reconstruction and reprojection. When the frame interval between the source and target views is large, the common region between them could be small and yield easily unreliable reconstruction. Thus, we set the farthest frame interval as Ω. Given the target view I_t, its source view are chosen in $\mathbf{S_t} = \{I_{t-\Omega} : I_{t+\Omega}\}$ and note that Ω also serves as the overlapping length of the neighboring snippets ($\Omega = 9$ by default). The loss function $\mathcal{L}(I_t, \mathbf{S_t})$ becomes

$$\mathcal{L}(I_t,\mathbf{S_t}) = \sum_{s_i\in\mathbf{S_t}}\omega_\alpha^{s_i}(\lambda_P\mathcal{L}_P(t,s_i) + \lambda_G\mathcal{L}_G(t,s_i)) + \omega_\beta^{s_i}\lambda_F\mathcal{L}_F(t,s_i), \tag{6}$$

where λ_P, λ_G, λ_F are some constant weights. $\omega_\rho^{s_i}$ is an exponential weight varying with the frame intervals.

$$\omega_\rho^{s_i} = \frac{\rho^{|s_i-t|}}{\sum_{s_j\in\mathbf{S_t}}\rho^{|s_j-t|}}. \tag{7}$$

As for the photometric and depth losses, we use $\rho = \alpha(=1.2)$ to increase the weights of the farther source views, which benefits from the learning of 3D scene with larger baselines. As for the optical-flow loss, since the accuracy of precomputed flow $\hat{F}_{t\to s_i}$ is often higher for small frame intervals but tends to collapse for the farther views, we use $\beta(=0.85)$ to emphasize the flow influence of closer frames. Finally, the optimal depths $\{D_t^*\}_k^{k+T}$ and camera poses $\{P_t^*\}_k^{k+T}$ are obtained by minimizing the overall loss:

$$\{D_t^*, P_t^*\}_k^{k+T} = \arg\min_{\{D_t,P_t\}_k^{k+T}}\frac{1}{T}\sum_{t=k}^{k+T}\mathcal{L}(I_t,\mathbf{S_t}). \tag{8}$$

Snippet coherence. To enforce the temporal consistency of the sequential snippets, for each snippet $\{I_t\}_{t=k}^{k+T}$, we freeze the depth and pose estimated already in the overlapping area of the previous snippet (*i.e.* $\{D_t^*, P_t^*\}_{t=k}^{k+\Omega}$), as shown in Figure 3. The optimization framework computes the loss of $\{D_t, P_t\}_{t=k+\Omega}^{k+T}$ with $\{D_t^*, P_t^*\}_{t=k}^{k+\Omega}$ fixed, so as to enhance the temporal consistency of depth and poses. The test-time training with snippet coherence is thus formulated as:

$$\{D_t^*, P_t^*\}_{k+\Omega}^{k+T} = \arg\min_{\{D_t,P_t\}_{k+\Omega}^{k+T}}\frac{1}{T}\sum_{t=k}^{k+T}\mathcal{L}(I_t,\mathbf{S_t}). \tag{9}$$

Parameter propagation. Unlike 2D optical-flow based testing-time training methods (*e.g.* [33]) for video stabilization, the 3D depth map employed in our approach often holds strong temporal continuity in a video, and thus

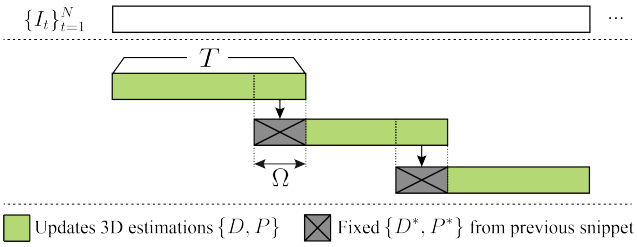

Figure 3. **Illustration of overlapping snippets:** An input sequence $\{I_t\}_{t=1}^N$ is sequentially divided into length-T snippets with Ω frames overlap. For a current snippet (except for the first one), the 3D depth and pose estimations of the overlapping area (gray with $\times$) are frozen as the optimal $\{D^*, P^*\}$ estimated for the previous snippet. The learning framework only updates the non-overlapping $\{D, P\}$ (green) to enforce the temporal coherence.

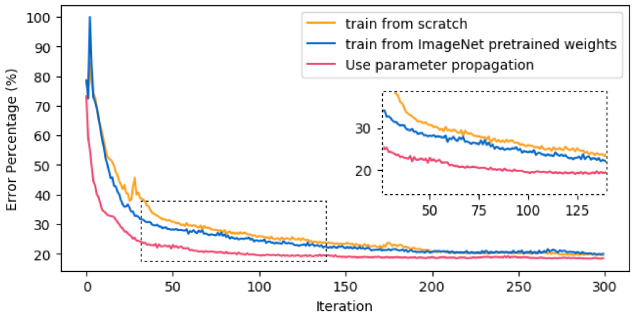

Figure 4. **Convergence speeds comparison:** The convergence curves averaged on all snippets (except for the first snippet) are shown for different parameter initialization strategies. Among them, parameter propagation shows the fastest convergence speed compared to training from scratch and ImageNet [26] pre-trained weights, which increases the test-time training speed better.

the previously estimated depth is easily transferable to the next snippet and serves as good initialization for better optimization. In our test-time training, the snippets are learned one by one according to their sequential order, and the optimal estimation $\{D_t^*, P_t^*\}_{t=1}^N$ are obtained through only a single-round process of the image sequence $\{I_t\}_{t=1}^N$. For the first snippet, both DepthNet and PoseNet are initialized with ImageNet [26]. Afterwards, the trained parameters of the current snippet are propagated to the next snippet for initializing the optimization with fewer iterations to converge. Note that the optical-flow maps between consecutive frame pairs [33] of an unsteady video would not have such continuity property to be exploited. As shown in Figure 4, training with parameter propagation shows a far faster convergence speed and a further loss descent, compared with training from scratch or ImageNet pre-trained weights. We also show the speed comparison with the test-time video stabilization method [33] in the experiments.

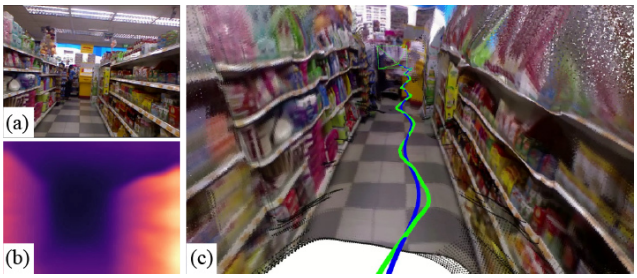

Figure 5. **Example of camera trajectory smoothing:** (a) Input video. (b) Depth D^* of (a). (c) The 3D model of the input video reconstructed by $\{D_t^*, P_t^*\}_{t=1}^N$. The green/blue paths represent the original/smoothed camera trajectories in 3D model, respectively.

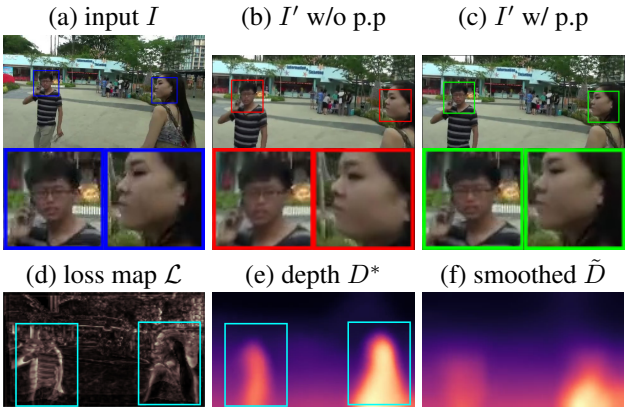

Figure 6. **Visual comparison tackling dynamic objects:** The input (a) contains two dynamic pedestrians, yielding large values in the loss map $\mathcal{L}$ and unreliable estimate of D^* (cyan boxes in (d) and (e)). The resulted (b) synthesized with D^* contains distortions in dynamic area (red box). Post-processing D^* to obtain $\tilde{D}$ with an adaptive smoothing filter based on $\mathcal{L}$. (c) synthesized with $\tilde{D}$ eliminates the distortions. The blue, red and green boxes present the details of moving pedestrians in (a), (b) and (c), respectively.

3.2. Frame Rectification.

Once we obtain the solutions of depths and camera poses of the entire sequence, each stabilized frame I_t' is synthesized with I_t and a compensated warp field F_t' which is computed by the depth D_t^* and a smoothed camera pose P_t'. The smoothed camera pose $\{P_t'\}_{t=1}^N$ (as illustrated in Figure 5) is typically obtained with a Gaussian filter. Our approach thus enables users to manipulate the stability by adjusting the smoothness degree (σ of Gaussian) of the trajectory. To our best knowledge, most previous learning-based methods [29, 31, 34, 35] do not enable the freedom of manipulation since their models are one-size-fits-all for inferring transformation directly. Hence, when the users hope to change the smoothness degree, they need to retrain the entire networks for finding the new solution.

Post-processing tackling dynamic objects. The 3D opti-

mization framework assumes the scene of input videos is static. Thus the networks will produce unreliable results on moving-object regions, yielding artifacts in stabilized videos (*e.g*. Figure 6e). Some recent joint depth and pose learning methods improve the outcomes based on optical flow [32, 38] or motion segmentation [2, 25, 9]. Yet these additional techniques would be either unreliable enough or increase heavily computational costs in test-time training.

To tackle this issue, we perform spatially variant smoothing based on the loss map obtained. As with test-time training, the map of loss $\mathcal{L}$ (Eq. 6) is available in our approach. We adaptively smooth the depth map $\{D_t^*\}_{t=1}^N$ with a spatially variant filter to obtain $\{\tilde{D}_t\}_{t=1}^N$, where the smoothing degree is varying with the sites in the loss map (the larger is the loss value, the larger is the smoothing degree and vice versa). The adaptively smoothed map $\{\tilde{D}_t\}_{t=1}^N$ is then employed for the view synthesis. As the large-loss areas are often caused by the dynamic objects since they violate the rigid-motion constraint, adopting a stronger smoothing tends to fit planar warping to these areas, so as to alleviate the artifacts. An illustration is given in Figure 6.

3.3. Implementation Details

We realize DepthNet and PoseNet using ResNet18 [12]. The optimization framework is implemented in PyTorch with Adam, where $\beta_1 = 0.9$, $\beta_2 = 0.99$, and learning rate $= 2 \times 10^{-4}$. The input frames are downsampled to 192×128 with the purpose of enlarging the snippet length T ($= 80$ as default). For each input unsteady video, we optimize the first snippet for 300 forward-backward iterations and 100 iterations for each of the subsequent snippets. The Gaussian filter in rectification stage is set with $\sigma = 12$ and window size 59 as default for the following experiments.

4. Experiments

We compare our Deep3D Stabilizer with several state-of-the-art methods on NUS dataset [21], which contains 144 short unsteady videos of 6 categories based on the camera motion and scene types: *regular*, *large parallax*, *crowd*, *running*, *quick rotation* and *zooming*. We use publicly available codes of the state-of-the-art method in the experiments. The stabilizer of commercial software *Adobe Premiere Pro CC 2020* is known as the traditional 3D-based method [17] using subspace constraints. Bundled [21] divides frames into grids to extract local homography transformations in offline processing. DIFRINT [4] performs learning-based frame interpolation on unsteady frames without cropping in offline. As for online stabilization methods, including MeshFlow [19] and StabNet [29], which aim to stabilize streaming videos and do on-the-fly thus cannot refer to the future frames. MeshFlow [19] computes pixel profiles as 2D local motion. StabNet [29] uses a pre-trained feed-forward network to infer grid-wise 2D transformations. We conduct both visual and quantitative comparisons.

Visual comparison. We highly encourage readers to see visual comparisons in supplementary video[1] since it is difficult to compare the visual quality of stabilized videos in thumbnails. As shown in Figure 7, we visualize the stability of resulted videos with optical flow tracking. Examples 1 and 2 are videos with parallax effects. Bundled [21] uses grid-wise homography transformation and the different motions of parallax effects in a single cell cause the wrong motion estimation and shear in resulted video. MeshFlow [19] derives local distortion in Example 2 because their grid-based transformation does not coordinate the spatial continuity of warping with neighboring grids. Our method estimates the frame motion in 3D motion model with dense depth maps to tackle parallax effect properly. Example 3 contains horizontal rotation motion. Adobe [17] and StabNet [29] show horizontal overcropping due to the large rotation. On the other hand, as DIFRINT [4] uses frame interpolation with nearby frames, the result suffers from severe local distortion in rotation. Example 4 also contains multiple moving pedestrians. DIFRINT [4] produces severe distortions in the pedestrians on the frame boundaries. Our method smooths the warping of non-rigid objects spatially in the adaptive filtering of post-processing to prevent from distortion. Example 5 is captured in running with severely shakes. Our method shows the best stability with the smoothest motion trajectory.

Quantitative comparison. We evaluate the the results on NUS dataset [21] following the metrics proposed by Liu *et al*. [21]. Specifically, the metrics include three aspects: cropping ratio, global distortion and stability.

The cropping ratio measures the view preservation from input to output caused by warping and cropping. We compute the homography H_t between the input frame I_t and output frame I_t'. The region that I_t have been warped onto I_t' by H_t indicates the preserved view V_t. The cropping ratio of the entire stabilized video is the average ratio of the preserved area, $\frac{1}{N}\sum_{t=1}^{N}|V_t|$. The score ranges from 0 to 1 and the larger represents the better view preservation.

The global distortion metric measures the distortion in output videos caused by non-isotropic scaling transformation from input to output, which can be extracted from the affine part of H_t by SVD decomposition [11]. The distortion score of the entire stabilized video, ranging from 0 to 1, is determined by the worst frame. The larger value indicates the better preservation of the shape in the input video.

The cropping-ratio and global-distortion metrics above prefer invariant images. The original image I_t gets the highest scores on these metrics, and thus they are only circumstantial. As for stability metric, we follow the idea of Liu *et al*. [21] to perform frequency analysis on estimated motion of a video. The assumption is that low frequency part

[1] Supplementary video: https://youtu.be/pMluFVA7NDQ

Figure 7. **Visual comparison of the state-of-the-art methods:** The artifacts are noted below thumbnails. The magenta paths on the thumbnails indicate the optical flow motion of the resulted videos. Smoother paths represent the more steady results. To visualize the cropping loss of each result, we rescale each video to the input scale. The distortion and shear are pointed out by yellow and red arrows, respectively. Zoom in for a better (more detailed) visualization.

of the estimated motion represents the dominant motion of a video. By using an off-the-shelf optical flow estimation network (PWC-Net [28]) to extract the motion of resulted video, we measure the SNR value to represent the stability score of the video, where the energy of the lowest 5% frequency components and the rest of the energy are treated as the signal and noise, respectively. A larger value indicates a more stable result. Note that there is a trade-off between the stability and cropping metrics, while one is larger, the other is low. Since a stabilized video often yields poor content preservation when pursuing high stability, the stability metric is often the main metric on experimental comparison.

The quantitative comparisons to state-of-the-art methods are shown in Table 1. As can be seen, our method performs the best (gets rank-1) on the stability metric for all categories except the *regular* category. Since the regular category contains simpler scene and motion types, many approaches perform averagely well on this category. Despite the stability score of our method is slightly lower than that of Bundled [21] and MeshFlow [19] in this category, our approach provides more steady videos. DIFRINT [4] shows the highest score in cropping ratio and global distortion metrics since they utilize frame interpolation to keep the view content, whereas resulting in the worst stability.

For *parallax* and *crowd* categories that contain parallax effects and dynamically moving human in the videos respectively, our method achieves the best stability score while still holds very well cropping and global distortion effects. It means that our method can produce more steady videos with smaller content distortion. In contrast, 2D Grid-based methods especially MeshFlow [19] and StabNet [29] introduce much larger distortion.

As for *running* category, Bundled [21] provides less cropping loss but with poor stability. Again, our method shows the highest stability and distortion scores with fairly well cropping value. Our approach provides a freedom of online manipulation, as they are trade-offs to each other. Furthermore, MeshFlow [19] shows the second highest stability but has the most distorted effects. In general, our method is more favorable to preserve the original shape of the videos with severe shaky motion.

For *quick rotation* and *zooming* categories, our method also achieves the least distortion effects comparing to frame interpolation-based DIFRINT [4] and the best stability. In sum, our method is able to provide not only the best visual quality results for almost all categories but also the flexibility of adjustable stability. As for learning-based methods, StabNet [29] shows the worst results in overall due to the difficulty of generalization. DIFRINT [4] uses a novel frame interpolation method to preserve the view content and shape but the stability is restricted.

Category	Metric	Input	Adobe [17]	Bundled [21]	DIFRINT [4]	MeshFlow[19]	StabNet [29]	Deep3D
Regular	Cropping	-	$0.70_{(3)}$	$0.67_{(4)}$	$0.98_{(1)}$	$0.67_{(4)}$	$0.54_{(6)}$	$0.79_{(2)}$
	Distortion	-	$0.94_{(5)}$	$0.95_{(3)}$	$0.98_{(1)}$	$0.95_{(3)}$	$0.82_{(6)}$	$0.97_{(2)}$
	Stability	11.08	$14.27_{(4)}$	$\mathbf{16.31}_{(1)}$	$13.14_{(6)}$	$15.72_{(2)}$	$13.85_{(5)}$	$15.53_{(3)}$
Large Parallax	Cropping	-	$0.60_{(4)}$	$0.74_{(3)}$	$0.98_{(1)}$	$0.55_{(5)}$	$0.46_{(6)}$	$0.78_{(2)}$
	Distortion	-	$0.83_{(4)}$	$0.91_{(3)}$	$0.94_{(1)}$	$0.75_{(5)}$	$0.71_{(6)}$	$0.93_{(2)}$
	Stability	14.49	$16.07_{(4)}$	$16.16_{(3)}$	$15.33_{(5)}$	$16.68_{(2)}$	$15.07_{(6)}$	$\mathbf{16.81}_{(1)}$
Crowd	Cropping	-	$0.53_{(4)}$	$0.70_{(2)}$	$0.98_{(1)}$	$0.50_{(5)}$	$0.41_{(6)}$	$0.70_{(2)}$
	Distortion	-	$0.86_{(4)}$	$0.89_{(3)}$	$0.94_{(1)}$	$0.78_{(5)}$	$0.65_{(6)}$	$0.93_{(2)}$
	Stability	16.71	$18.45_{(3)}$	$18.25_{(4)}$	$17.63_{(5)}$	$18.76_{(2)}$	$17.58_{(6)}$	$\mathbf{18.80}_{(1)}$
Running	Cropping	-	$0.41_{(5)}$	$0.67_{(2)}$	$0.96_{(1)}$	$0.44_{(4)}$	$0.41_{(5)}$	$0.50_{(3)}$
	Distortion	-	$0.85_{(3)}$	$0.85_{(3)}$	$0.89_{(1)}$	$0.80_{(5)}$	$0.77_{(6)}$	$0.89_{(1)}$
	Stability	10.30	$14.85_{(3)}$	$13.14_{(4)}$	$12.50_{(6)}$	$15.62_{(2)}$	$12.88_{(5)}$	$\mathbf{15.81}_{(1)}$
Quick Rotation	Cropping	-	$0.38_{(6)}$	$0.70_{(2)}$	$0.98_{(1)}$	$0.39_{(4)}$	$0.39_{(4)}$	$0.58_{(3)}$
	Distortion	-	$0.85_{(3)}$	$0.86_{(2)}$	$0.83_{(4)}$	$0.41_{(6)}$	$0.67_{(5)}$	$0.87_{(1)}$
	Stability	19.51	$18.05_{(6)}$	$20.44_{(2)}$	$20.12_{(3)}$	$19.01_{(4)}$	$18.13_{(5)}$	$\mathbf{21.91}_{(1)}$
Zooming	Cropping	-	$0.48_{(5)}$	$0.61_{(3)}$	$0.98_{(1)}$	$0.52_{(4)}$	$0.47_{(6)}$	$0.62_{(2)}$
	Distortion	-	$0.87_{(4)}$	$0.93_{(2)}$	$0.89_{(3)}$	$0.84_{(5)}$	$0.71_{(6)}$	$0.95_{(1)}$
	Stability	17.09	$18.28_{(5)}$	$20.19_{(3)}$	$19.06_{(4)}$	$20.47_{(2)}$	$17.25_{(6)}$	$\mathbf{20.90}_{(1)}$

Table 1. **Quantitative comparison of the state-of-the-art methods:** The scores are averaged over each category. The higher value of cropping ratio, global distortion and stability the better. The subscript of each value indicates the ranking in the methods compared.

Metric	w/o $\mathcal{L}_G$ and $\mathcal{L}_F$	w/o $\mathcal{L}_F$	full
Cropping	0.57	0.62	**0.66**
Distortion	0.87	0.89	**0.92**
Stability	16.95	17.35	**18.29**

Table 2. **Ablation study on the loss terms.**

	Methods	Runtime
online (streaming)	MeshFlow [19]	25 ms
	StabNet [29]	118 ms
offline	Bundled [21]	392 ms
	Yu and Ramamoorthi [33]	1610 ms
	DIFRINT [4]	67 ms
	Yu and Ramamoorthi [34]	570 ms
	Deep3D (Ours)	670 ms
	– optimization stage	641 ms
	– rectification stage	29 ms

Table 3. **Per-frame runtime comparison.**

Ablation study. We conduct an ablation study of the loss terms ($\mathcal{L}_P$, $\mathcal{L}_G$, $\mathcal{L}_F$) on entire NUS [21] (shown in Table 2). The $\mathcal{L}_G$ and $\mathcal{L}_F$ are the objectives of geometric regularization to acquire consistent 3D estimations and prevent from distortion. With the aids of both $\mathcal{L}_G$ and $\mathcal{L}_F$, the results show considerably improvements in all metrics.

Runtime comparison. We analyze the runtime of the proposed method on NUS [21] and compare with the state-of-the-art methods. The time is measured on an RTX2080Ti GPU. Table 3 summaries the runtime comparison. Despite our method takes a longer period on the optimization stage in contrast to some other offline methods, we simply use parameter-propagation that leverages the scene continuity of input video to give more than twice times speedup in contrast to the test-time training method [33]. Moreover, the rectification stage takes only 29 ms per-frame (34.5 fps) so that our algorithm enables real-time stability manipulation to users.

5. Conclusions and Future Work

We propose Deep3D Stabilizer, a novel 3D-based self-supervised method for video stabilization, The depths and camera poses of a video are estimated by a 3D optimization framework without pre-training data. The rectification stage incorporates the estimated 3D geometry to perform smoothing for generating a stable video. To our knowledge, this is the first study on learning-based stabilizer leveraging 3D cues. Our method can handle parallax effect and severe camera shakes, while attaining high stability and low distortion. Experimental results reveal that our Deep3D Stabilizer outperforms the state-of-the-art methods in general.

Our future directions include tackling the undesired intra-frame effects (*e.g.* motion blur, rolling shutter) and performing frame extrapolation outside the cropped area of the stabilized videos to further enhance the visual quality.

Acknowledgement. This work was partially supported by MediaTek Inc., and Ministry of Science and Technology in Taiwan (MOST 109-2221-E-002-207-MY3 and MOST 110-2634-F-002-047).

References

[1] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, pages 35–45, 2019.

[2] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *AAAI*, 33:8001–8008, 2019.

[3] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, pages 7063–7072, 2019.

[4] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. In *SIGGRAPH Asia*, 2019.

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 3(January):2366–2374, 2014.

[6] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, pages 740–756. Springer, 2016.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pages 270–279, 2017.

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

[9] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *ICCV*, pages 8977–8986, 2019.

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

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

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] Michal Irani. Multi-frame correspondence estimation using subspace constraints. *IJCV*, 48(3):173–194, 2002.

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

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

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

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

[18] Shuaicheng Liu, Mingyu Li, Shuyuan Zhu, and Bing Zeng. Codingflow: Enable video coding for video stabilization. *IEEE TIP*, 26(7):3291–3302, 2017.

[19] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *ECCV*, pages 800–815. Springer, 2016.

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

[21] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM TOG*, 32(4):1–10, 2013.

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

[23] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM TOG*, 39(4), 2020.

[24] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, pages 5667–5675, 2018.

[25] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *CVPR*, 2019-June(1):12232–12241, 2019.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

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

[28] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018.

[29] 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 TIP*, 28(5):2283–2292, 2019.

[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

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

[32] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *CVPR*, pages 1983–1992, 2018.

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

[34] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *CVPR*, pages 8159–8167, 2020.

[35] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE TIP*, 29:3582–3595, 2020.

[36] Junsheng Zhou, Yuwang Wang, Kaihuai Qin, and Wenjun Zeng. Moving indoor: Unsupervised video depth learning in challenging environments. In *ICCV*, pages 8618–8627, 2019.

[37] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858, 2017.

[38] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, pages 36–53, 2018.