Self-Supervised Real-time Video Stabilization

Jinsoo Choi[1]
jinsc37@kaist.ac.kr
Jaesik Park[2]
jaesik.park@postech.ac.kr
In So Kweon[1]
iskweon77@kaist.ac.kr

[1] KAIST
Republic of Korea
[2] POSTECH
Republic of Korea

Abstract

Videos are a popular media form, where online video streaming has recently gathered much popularity. In this work, we propose a novel method of real-time video stabilization - transforming a shaky video to a stabilized video as if it were stabilized via gimbals in real-time. Our framework is trainable in a self-supervised manner, which does not require data captured with special hardware setups (i.e., two cameras on a stereo rig or additional motion sensors). Our framework consists of a transformation estimator between given frames for global stability adjustments, followed by scene parallax reduction module via spatially smoothed optical flow for further stability. Then, a margin inpainting module fills in the missing margin regions created during stabilization to reduce the amount of post-cropping. These sequential steps reduce distortion and margin cropping to a minimum while enhancing stability. Hence, our approach outperforms state-of-the-art real-time video stabilization methods as well as offline methods that require camera trajectory optimization. Our method procedure takes approximately 24.3 ms yielding 41 fps regardless of resolution (e.g., 480p or 1080p).

1 Introduction

Due to the recent popularity in social networking services (SNS), videos have become a popular media form, demanding higher visual quality as time progresses. Recently, cameras (including smartphones) make use of hardware configurations to produce stabilized videos. One such method is the Optical Image Stabilizer (OIS), which negates instability caused by hand movements via adjusting the optical lens positions. Another mechanism is the Electronic Image Stabilizer (EIS) that is designed to compensate for more substantial motion. However, these methods require specialized motion-sensing hardware synchronized with image capture, and may lead to significant cropping of the frame boundaries of the original video, which results in an inevitable zoom-in effect. Fig. 2 shows a real example of a video captured with OIS and EIS.

To cover such limitations, software approaches have been developed typically for offline purposes, namely post-processing of existing videos [8, 17, 18]. Offline approaches have

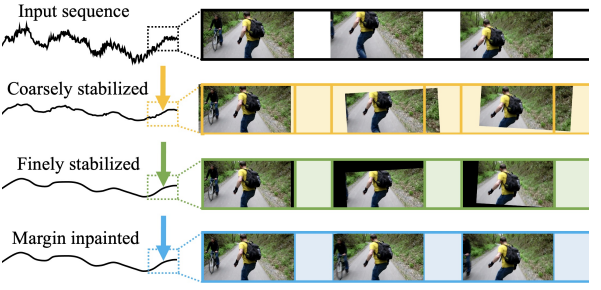

Figure 1: Illustration of approach. Given a shaky camera path (black line profile), our method applies global transforms(top row). Then, spatially smoothed flow warping is applied for further stability(2nd row). Lastly, the image margins are inpainted to minimize missing regions (last row).

Figure 2: Example of the cropping ratios among OIS+EIS, our method and original frames taken by a Galaxy S10. OIS+EIS exhibits *zoom-in effects*.

shown success with stabilization quality, but the computational speed is an issue. More recently, due to the growing popularity in online streaming, works on real-time methods [19, 20, 29, 30] have been introduced. Especially, deep learning-based methods [29, 30] utilize the fast feed-forward mechanism and a supervised approach to realize real-time performance. Compared to offline methods, however, the real-time approaches exhibit a slight loss of stability and increased distortion artifacts. Moreover, supervised learning methods require capturing simultaneous ground truth footages for supervised learning, which is a time-consuming process.

A desirable video stabilization can be characterized by minimal cropping, low visual distortion, and high stability. In this work, we propose a deep learning-based approach to enable real-time video stabilization with these aspects. The idea is to introduce a cascade module which consists of a coarse (global) stabilizer and a fine (detail) stabilizer to handle complex camera motion in the wild effectively. The coarse stabilizer estimates the rigid inter-frame transformations among the multiple (unstable) frame inputs for global adjustment of subsequent frames. Next, a fine stabilizer applies the spatially smoothed optical flow between input frames to handle parallax and spatially varying instability. Both stabilizers use moving average filters to suppress noisy camera motion without explicitly optimizing camera trajectories. After the stabilization step, the inpainting module fills in the blank margins induced by the stabilization process. Fig. 1 shows an illustration of the process.

A notable aspect of our method is that our approach does not require any ground truth videos for deep architecture training, but yet manages to produce strong stabilization effects in a self-supervised manner. This distinguishes our method with supervised methods [29, 30] that require sets of two videos, one of which is unstable while the other is physically stabilized with gimbals, captured simultaneously. The proposed network is fully convolutional and is trained end-to-end, where our model achieves real-time regardless of input video size (e.g., 480p or 1080p) with a single GPU. The contributions of our work are listed as follows.

- We propose a real-time approach for video stabilization. The input video is stabilized via a single feed-forward through the constituent modules.

- We introduce a self-supervised training method for video stabilization. Therefore, existing videos can be used for network training.
- Our method induces low levels of cropping, low visual distortion, and high stability. The proposed approach outperforms the state-of-the-art methods both quantitatively and qualitatively.

2 Related Work

Video stabilization approaches consist of offline and real-time methods. Offline methods are characterized by strong stabilization effects while having a high computational load. On the other hand, real-time methods are computationally efficient but convey weaker stabilization effects compared to offline methods. We briefly address offline and real-time approaches in the following.

Previous works on offline methods deal with techniques including Structure from Motion (SfM) [12], depth information [16], 3D plane constraints [33], 3D reconstruction [2], light fields [27], gyroscopes [1, 11, 24], and partial 3D information [7, 13, 14]. Image-based methods have also shown sufficient quality [3, 6, 10, 22, 32]. Grundmann *et al*. [8] apply L1-norm optimization for camera path computation and extend the approach to handle rolling shutter effects [9]. Liu *et al*. [17] model camera paths for each image patches, and extend the idea to model the entire pixel profiles [18]. This work shows that spatially smoothed flow is useful for stabilizing frames, which our approach builds upon. Recently, Choi and Kweon [4] introduce an iterative frame interpolation approach to video stabilization.

Real-time methods typically employ more efficient computations while using historical frames to estimate stabilization parameters. Liu *et al*. [19] proposes computing sparse mesh profiles via applying median filters to pixel profiles. This idea is extended further to video coding [20]. Limitations of these methods are the slight wobbling artifacts and relatively low stabilization quality compared to offline methods. Recently, deep learning-based algorithms [21] take advantage of fast feed-forward capabilities. Wang *et al*. [29] propose a supervised learning approach to video stabilization by defining the novel stability and temporal loss terms. Similarly, Xu *et al*. [30] proposes training an adversarial network in a supervised manner, which estimates transformation parameters to generate stabilized frames. Compared with those approaches, our approach does not require stable videos for training. Recently, Yu and Ramamoorthi [32] propose a learning method using principal components of optical flow information. Our approach estimate smooth warping field via neural network and produces high-quality stabilization.

The core module of our pipeline is related to deep image tranformation estimation and image inpainting. Deep homography estimation [5, 23] enables transformation matrix estimation between two frames without explicitly extracting image features. On the other hand, advances in image inpainting tasks employ specific deep architecture components for natural inpainting quality [15, 31]. Although image boundary inpainting is a difficult task, our deep margin inpainter utilizes multiple adjacent frames to fill in frame boundaries.

3 Architecture

We aim to achieve high-quality stabilization and high computational speed via three sequential modules, namely the coarse stabilizer, fine stabilizer, and margin inpainter. The overall

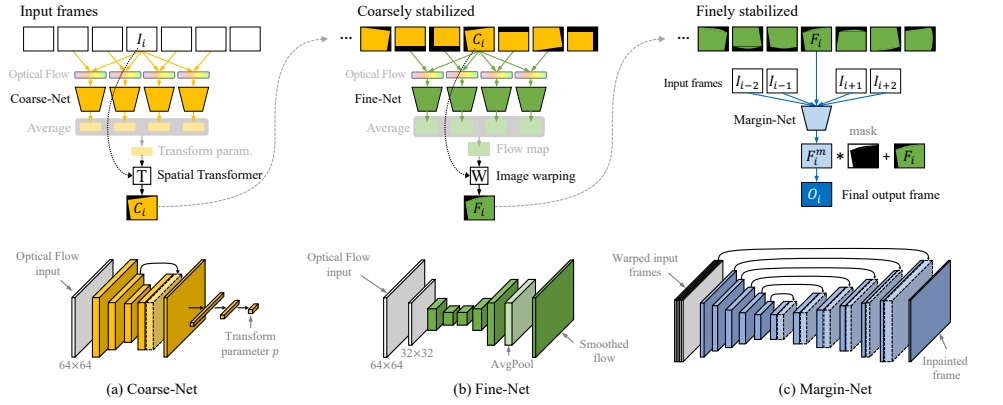

Figure 3: An overview of the Coarse-Net, Fine-Net and Margin-Net during testing. The Coarse-Net computes the transform parameters among input frames with respect to the center image I_i, which are then averaged to transform I_i to C_i. The Fine-Net compute the mean of the flow maps to warp C_i, generating F_i. Finally, the Margin-Net takes F_i and neighboring input frames to generate F_i^m, which is applied with the margin mask to generate the final output frame O_i.

process of our method is illustrated in Fig. 3. The coarse (first) stabilizer estimates the rigid transformations among frames to globally transform a frame for stabilization. The fine (second) stabilizer estimates the spatially smoothed flow among frames to adjust the remaining instability. Finally, the margin inpainter fills in the frame boundaries to reduce the need for cropping. Unlike prior work that runs offline, our approach does not involve any optimization, camera pose estimation. Instead, sequential feed-forward passes of the coarse and fine networks produce the stabilized frames, which has been shown to be effective [32].

3.1 Coarse stabilizer

Our approach first estimates the image transformations between video frames.[1] The network for coarse motion estimation (Coarse-Net) is built upon the U-Net architecture [26] followed by fully connected (FC) layers to estimate the transformation parameters. The Coarse-Net takes an optical flow $\mathcal{F}_{j\to i}$ from image pairs (I_j, I_i) and produces the transformation parameters that can relieve abrupt motion. The network is shown in Fig. 3 (a).

With this procedure, an i-th image I_i has the estimated transformation parameters $\mathbf{p}_{j\to i}$ of the adjacent time stamps $j \in \mathcal{N}$. They are averaged to produce a transformation matrix T. It is obtained from $\bar{\mathbf{p}}_i$ as defined below:

$$\bar{\mathbf{p}}_i := \frac{1}{2N} \sum_{j=-N, j\neq i}^{N} \mathbf{p}_{i+j\to i}. \tag{1}$$

T adjusts the target frame to produce $C_i = \mathrm{T}(I_i)$ for stabilization.[2] This procedure is equiva-

[1] We empirically found the 3-DoF rigid transformation $\begin{bmatrix} R(\theta) & \mathbf{t} \\ 0 & 1 \end{bmatrix}$ is surprisingly effective for video stability, compared with transformations having higher DoF. The supplement summarizes our experiment results on other type of transformations that include scale and/or shear. Please note that the combination of Coarse stabilizer and Fine stabilizer and their moving average can effectively handle challenging motions (zoom, parallax, and so on), as shown in Fig. 6, 7.

[2] For convenience, we denote T as a transformation matrix as well as a function that transforms image using T.

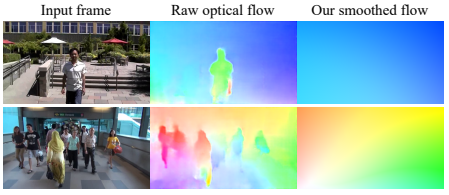

Figure 4: The Fine-Net produces spatially smoothed flow maps via a single feed forward operation that is purely induced by camera movement. The raw optical flow responds to the foreground, while our network is agnostic to it.

lent to applying the moving average filter [28] to the relative displacements between frames within the window $2N$. Since the moving average filter is a low-pass filter, it leads to noise suppression effects. This procedure enables finding the transformation to stabilize the target frame, while avoiding expensive camera pose estimation and optimization.

3.2 Fine stabilizer

The image transformations estimated by the coarse stabilizer alone is not enough to handle parallax and spatially varying instability. To stabilize such spatially varying instability, we employ *spatially smoothed optical flow*. This stage is inspired by Liu *et al.* [18]'s work. It demonstrates spatially smoothed pixel trajectories are what essentially underlies between stabilized frames.

In practice, the smooth flow should be robust to foreground object movement. For example, a moving person in a scene cannot interfere with the optical flow that is induced solely by camera movement. Prior methods implement the smooth optical flow by identifying pixels with discontinuous flow vectors [18], which can be a slow process. Instead, we utilize a simple neural network generating the smoothed optical flow through a single feed-forward.

In this regard, the architecture of the fine stabilizer (Fine-Net) produces smooth flow with a tiny U-shaped network. Fine-Net applies subsequent down-sampling of the features for filtering noisy motions. In addition, the average pooling layers in the network further induces spatial smoothness. Thus, by design, the Fine-Net does not produce high-frequency flow. Fine-Net is shown in Fig. 3 (b), and examples of smooth flow are shown in Fig. 4. It shows that output flow is agnostic to small foreground movement.[3]

Likewise in the coarse stabilizer, the moving average is applied to a group of smooth flow maps in a sliding window manner as Eq. (1). Averaged flow map generated from a window of C_i frames, produce a *stabilizing warp* W for frame C_i, resulting in F_i. Unlike the recent work by Yu and Ramamoorthi [32], our approach does not require an additional PCA component, and solves the problem with simple network architectures.

3.3 Margin inpainter

As frames are adjusted to stabilized positions, missing blank regions are inevitably created at the frame margins. This is due to temporally unseen content during camera shakes. Thus, a typical post-processing, cropping the boundary of frame was needed in previous approaches [17, 19, 29] to conceal missing blank regions. However, excessive cropping leads to loss of original content and the zoom-in effect as a consequence. We attempt to minimize such cropping via the margin inpainter.

Our approach takes five frames, namely I_{i-2}, I_{i-1}, F_i, I_{i+1}, and I_{i+2}, where F_i is the warped image obtained from Fine-Net. For motion compensation of the adjacent frames, the

[3] In Fig. 4, we use the full-sized images for the raw optical flow computation for the illustration. Fine-Net uses resized video instead for optical flow computation as discussed in Sec. **??**.

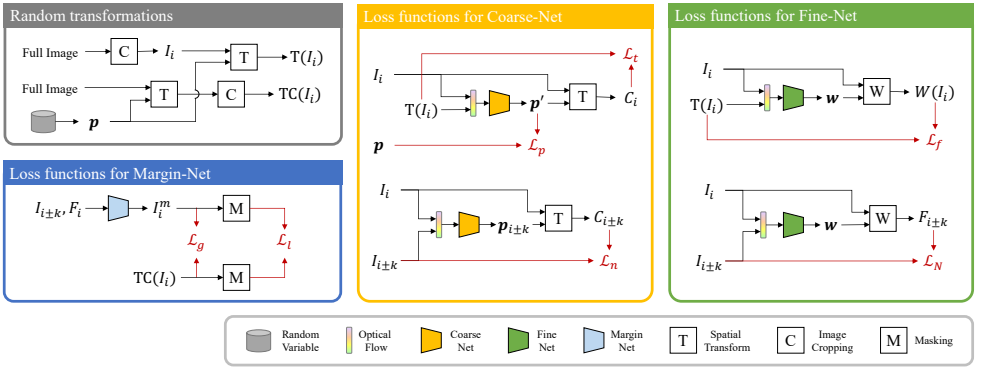

Figure 5: Overview of all loss functions for self-supervised training. Losses for Coarse-Net (yellow box), Fine-Net (green box), and Margin-Net (blue box) are shown. For each network, the original image I_i is being self-augmented using the random image transforms (gray box), and it is used for the loss function shown in red. For the data augmentation, 256×256 patches are used for training from the original image in the training phase.

smooth flow maps estimated from the Fine stabilizer are used to warp adjacent frames to the stabilized center frame F_i. The aligned image frames are fed as input to the Margin-Net, as shown in Fig. 3 (c).

Inspired by the gated convolutions [31], we implement the pixel gates via convolutional layers that act as spatial attention modules. Adopting gated convolutions leads to effectively handling the arbitrary shapes of the boundary masks, which is beneficial compared to the equivariant nature of vanilla convolutional layers.

As a result, the Margin-Net learns how to combine adjacent frames to fill in the missing area of F_i, and produces the inpainted image I_i^m. The Margin-Net concatenates the original I_i^m and warped frames F_i as input to generate the inpainted frame O_i.

4 Self-Supervised Learning

To train Coarse-Net, Fine-Net, and Margin-Net appropriately, we propose a novel self-supervised learning scheme. This distinguishes our approach with supervised methods [29, 30]. We carefully design the training methods for each module[4], which is a crucial aspect of making our approach work. An overview of the loss functions is depicted in Fig. 5.

4.1 Training Coarse-Net

To train the Coarse-Net, we generate a pair of an original image and a randomly transformed image. The random transformation parameters $\mathbf{p}$ (e.g., rotation, translation) is sampled from a bounded uniform distribution ranging up to 30 degrees and 50 pixels respectively.

Learning by random augmentation. As the first step, the image I_i is transformed via the random rigid transformation matrix, yielding $\mathrm{T}(I_i)$. Then, the flow map between I_i and $\mathrm{T}(I_i)$ is computed and fed to the Coarse-Net which estimates the parameter vector $\mathbf{p}'$. I_i is then

[4]It is worth to note that all three networks in our pipeline can handle an arbitrary size of images – Coarse-Net and Fine-Net take rescaled flow fields as input, and Margin-Net is fully convolutional. With this in mind, we use smaller patches of 256 × 256 pix. of the video frames, to augment data and to reduce memory consumption during training. The cropped frames can be flipped horizontally or vertically for further augmentations.

transformed with $\mathbf{p}'$ and results in C_i. We use the L1 loss function between the randomly transformed frame $\mathrm{T}(I_i)$ and the estimated transformed frame C_i defined as:

$$\mathcal{L}_t = \|\mathrm{T}(I_i) - C_i\|_1 \tag{2}$$

In addition, since we know the exact (randomly generated) ground truth (GT) parameter values, we can penalize the discrepancy between the GT parameter vector $\mathbf{p}$ and the estimated vector $\mathbf{p}'$. The loss is $\mathcal{L}_p = \|\mathbf{p} - \mathbf{p}'\|_1$.

Learning by predicting rigid transformation. $\mathcal{L}_t$ and $\mathcal{L}_p$ show losses using a rigidly transformed version of the same frame. In addition to these synthetic augmentations, another type of loss teaches how to roughly align temporally adjacent image pairs.

For this purpose, we collect adjacent frames of I_i towards each of the K neighboring frames from both sides[5], such as $I_{i-K}, ..., I_{i-1}$ and $I_{i+1}, ..., I_{i+K}$. In this training phase, the optical flow is computed via inputs (I_i, I_{i+k}), and the Coarse-Net estimates the transform parameters $\mathbf{p}_{i+k}$. $\mathbf{p}_{i+k}$ transforms I_i and produces C_{i+k}. Now we apply the sum of L1 losses using I_i and C_{i+k}, $\mathcal{L}_n = \sum_{k \in \pm[1,K]} \|I_{i+k} - C_{i+k}\|_1$. This loss function makes Coarse-Net to learn rigid transform parameters for roughly aligning adjacent frames. This loss term increases the robustness for handling dynamic scenes.

Thus, the overall training loss for the Coarse-Net is the weighted sum of the three losses $L_C = \alpha\mathcal{L}_t + \beta\mathcal{L}_p + \gamma\mathcal{L}_n$, where $\alpha = \beta = 1.0$ and $\gamma = 0.1$.

4.2 Training Fine-Net

Learning by random augmentation. Training Fine-Net is similar to the training steps for Coarse-Net. We reuse the randomly transformed image $\mathrm{T}(I_i)$ in Eq. (2) and compute the optical flow between $\mathrm{T}(I_i)$ and I_i. This flow map is fed through the Fine-Net, producing a *smoothed flow map*. Frame I_i is then warped by performing the backward warping technique, using the computed smooth flow, producing the warped frame $\mathrm{W}(I_i)$. The L1 loss is applied between $\mathrm{T}(I_i)$ and the estimated frame $\mathrm{W}(I_i)$, such as $\mathcal{L}_f = \|\mathrm{T}(I_i) - \mathrm{W}(I_i)\|_1$.

This loss helps the network to learn a relatively simple warping map which provides a warm start for training. Note that the *spatially smoothed* flow maps are produced by the Fine-Net's architecture design, as shown in Fig. 3 (b).

Learning by predicting smoothed flow. The Fine-Net aims to estimate smooth flow maps between neighboring frames and I_i. We apply the sum of L1 losses to the K frames as follows: $\mathcal{L}_N = \sum_{k \in \pm[1,K]} \|I_{i+k} - F_{i+k}\|_1$, where F_{i+k} is the frame I_i warped towards I_{i+k} using the Fine-Net's predicted smoothed flow. The overall training loss for the Fine-Net is the weighted sum of the two losses $\mathcal{L}_F = \alpha\mathcal{L}_f + \gamma\mathcal{L}_N$, where $\alpha = 1.0$ and $\gamma = 0.1$.

4.3 Training Margin-Net

The task of the Margin-Net is to fill in natural content in the blank margins of F_i, as shown in Fig. 3 (c). To train this network, we prepare a pair of images, with or without blank margins. In order to obtain such images, we reuse $\mathrm{T}(I_i)$ in Eq. (2). In addition, the same random transform is applied to the *full-sized* image of the dataset, and then cropping is applied. In contrast to $\mathrm{T}(I_i)$, this procedure *preserves* the content at the frame boundaries, as shown in Fig 5. Let us call this image $\mathrm{TC}(I_i)$.

[5] We use $K = 2$ in our experiment.

Figure 6: Quantitative comparison with the state-of-the-art video stabilization methods [7, 8, 12, 13, 17, 18, 19, 20, 29, 30]. The results are evaluated with the three metrics: cropping ratio, distortion value, and stability score. Higher values indicate better performance.

Learning to inpaint. Using the stabilized frame F_i and warped images of $[I_{i-2}, I_{i-1}, I_{i+1}, I_{i+2}]$ as input, the Margin-Net outputs an inpainted frame I_i^m, utilizing the information from the given adjacent frames. The Margin-Net is trained by the L1 loss between the transformed frame $\mathrm{TC}(I_i)$ and the inpainted image I_i^m: $\mathcal{L}_g = \|\mathrm{TC}(I_i) - I_i^m\|_1$.

Furthermore, we can isolate the loss to focus only on the inpainted region by applying a mask to both $\mathrm{TC}(I_i)$ and I_i^m: $\mathcal{L}_l = \|m \odot (\mathrm{TC}(I_i) - I_i^m)\|_1$, where m is the inpainting margin mask and $\odot$ denotes the element-wise product. Since the stabilized frame F_i is used as input, a mask indicating the inpainting region can be computed via the estimated transform parameter and flow map from the preceding Coarse and Fine-Net. Using these two losses together increases the training stability: $\mathcal{L}_M = \mathcal{L}_g + \mathcal{L}_l$.

5 Experiments

For thorough evaluation, we conduct an extensive quantitative comparison to both offline and state-of-the-art real-time methods, visual comparisons, analysis against a commercial product that runs offline, and ablation studies. We also present results on full resolution videos, namely nHD (640×360), HD (1280×720), and FHD (1920×1080). For approaches without public code, we refer to the reported numbers in each paper. For quantitative comparison, we use three commonly used metrics [17, 19, 20, 29, 30] to assess video stabilization quality, namely the *cropping ratio*, *distortion value*, and *stability score*. For details including implementation details, please refer to the supplementary material.

Test videos. Prior arts use videos that are publicly available from Liu *et al*. [17]. However, it is important to note that *no prior work validates all videos* provided by Liu *et al*. [17] for video stabilization. Instead, each method uses an arbitrary subset of the videos. To configure the same experimental setup and to respect the numbers reported by state-of-the-arts, we use the *union* of the video sets that were tested by them [7, 8, 12, 13, 17, 18, 19, 20, 29, 30]. Therefore, some videos that are not tested by an approach is not displayed in this paper.

The union of video set consists of 19 video clips, and it covers forward/backward motion (#1-5, 9, 10, 18), side motion (#6, 7, 11, 12, 14) zoom-in and zoom-out (#8, 17), the rolling-

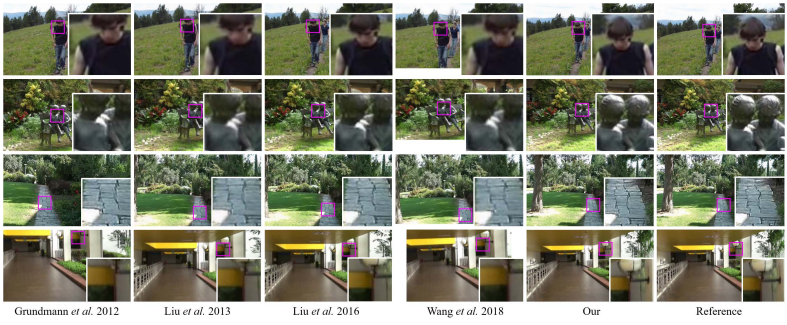

Figure 7: Visual comparison with *offline* approaches [9, 17, 19, 29] in terms of zoom-in effect. Pink bounding boxes of the same size are magnified to show image regions at the same respective locations. Our approach shows the least amount of distortion and zoom-in effect. Please see our supplementary video for visual results.

shutter artifact (#13, 15, 16, 19). The thumbnails of the video clips are shown in Fig. 6. Note that our approach is validated with the union of the video sets tested by all prior works, and we emphasize that this is the *most extensive collection of comparisons* in the video stabilization literature. Furthermore, our network is trained on the DAVIS dataset [25] different from the 19 test clips, which verifies the generalization of our approach.

Evaluation result. As shown in Fig. 6, the comparison with offline methods [7, 8, 12, 13, 17, 18], online (real-time) methods [19, 20, 29, 30], and ours indicates that the proposed method shows favorable performance for the majority of videos. In particular, our real-time method shows comparable cropping ratio to the recent approach proposed by Choi and Kweon [4], although their approach runs offline and is designed to prevent any cropping.

We also conduct comparisons with a widely used commercial product, Adobe Premiere (Pro CC 2017), and two recent approaches by Choi and Kweon [4] and by Yu and Ramamoorthi [32]. Please refer to the supplementary material.

Ablation study. We conduct an extensive analysis of stabilization scores with various combination of proposed modules (Coarse-Net, Fine-Net, Margin-Net), window sizes, regarding resizing the frame inputs, types of transformations for Coarse-Net (translation, rotation, scale, and shear). Please see the supplement for the details.

Visual comparison. Although it is not fair to directly compare our real-time method to offline methods, we present visual comparisons to the state-of-the-art offline methods [9, 17, 19, 29] in Fig. 7. Notice that our method closely resembles the input content, while state-of-the-art methods convey enlargements due to the zoom-in effect. Furthermore, we can observe moderate levels of blur introduced by the other methods, whereas our approach does not exhibit such artifacts. For detailed visual results, please refer to the supplement.

6 Conclusion

We propose an unsupervised learning algorithm for video stabilization that runs in real-time. Our approach consists of simple and efficient modules, such as Coarse-Net for image transform estimation, and Fine-Net for estimating the smoothed flow, and Margin-Net to compensate cropped contents. By combining all the modules, our approach outperforms both offline and real-time state-of-the-art methods on various videos (that shows zoom, rotation, parallax, and so on) and is robust to severe camera movements.

7 Acknowledgement

This work was supported by the Institute for Information & Communications Technology Promotion (2017-0-01772) grant funded by the Korea government. This work was also supported by Institute of Information and communications Technology Planning and evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, AI Innovation Hub)

References

[1] Steven Bell, Alejandro Troccoli, and Kari Pulli. A non-linear filter for gyroscope-based video stabilization. In *European Conference on Computer Vision (ECCV)*, 2014.

[2] Chris Buehler, Michael Bosse, and Leonard McMillan. Non-metric image-based rendering for video stabilization. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 609–614, 2001.

[3] Bing-Yu Chen, Ken-Yi Lee, Wei-Ting Huang, and Jong-Shan Lin. Capturing intention-based full-frame video stabilization. In *Computer Graphics Forum*, volume 27, pages 1805–1814, 2008.

[4] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM Transactions on Graphics (TOG)*, 39(1):1–9, 2020.

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[6] Michael L Gleicher and Feng Liu. Re-cinematography: improving the camera dynamics of casual video. In *ACM Multimedia*, pages 27–36, 2007.

[7] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM Transactions on Graphics (TOG)*, 31(5):126, 2012.

[8] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 225–232, 2011.

[9] Matthias Grundmann, Vivek Kwatra, Daniel Castro, and Irfan Essa. Calibration-free rolling shutter removal. In *IEEE ICCP*, pages 1–8, 2012.

[10] Hua Huang, Xiao-Xiang Wei, and Lei Zhang. Encoding shaky videos by integrating efficient video stabilization. *IEEE TCSVT*, 2018.

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

[12] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics (TOG)*, 28(3):44, 2009.

[13] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Transactions on Graphics (TOG)*, 30(1):4, 2011.

[14] Feng Liu, Yuzhen Niu, and Hailin Jin. Joint subspace stabilization for stereoscopic video. In *IEEE International Conference on Computer Vision (ICCV)*, pages 73–80, 2013.

[15] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision (ECCV)*, 2018.

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

[17] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Transactions on Graphics (TOG)*, 32(4):78, 2013.

[18] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 4209–4216, 2014.

[19] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *European Conference on Computer Vision (ECCV)*, pages 800–815, 2016.

[20] Shuaicheng Liu, Mingyu Li, Shuyuan Zhu, and Bing Zeng. Codingflow: enable video coding for video stabilization. *IEEE Transactions on Image Processing (TIP)*, 26(7):3291–3302, 2017.

[21] Yu-Lun Liu, Wei-Sheng Lai, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Hybrid neural fusion for full-frame video stabilization. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2021.

[22] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2006.

[23] Ty Nguyen, Steven W Chen, Shreyas S Shivakumar, Camillo Jose Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *IEEE Robotics and Automation Letters*, 3(3):2346–2353, 2018.

[24] Hannes Ovrén and Per-Erik Forssén. Gyroscope-based video stabilisation with auto-calibration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2090–2097, 2015.

[25] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241. Springer, 2015.

[27] Brandon M Smith, Li Zhang, Hailin Jin, and Aseem Agarwala. Light field video stabilization. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[28] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[29] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Ariel Shamir, Song-Hai Zhang, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing (TIP)*, 2018.

[30] 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, 2018.

[31] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[32] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 8159–8167, 2020.

[33] Zihan Zhou, Hailin Jin, and Yi Ma. Plane-based content preserving warps for video stabilization. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 2299–2306, 2013.