Self-Supervised Real-time Video Stabilization

Jinsoo Choi
jinsc37@kaist.ac.kr

Jaesik Park
jaesik.park@postech.ac.kr

In So Kweon
iskweon77@kaist.ac.kr

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...
Focusing on the real-time aspect of video stabilization, the authors propose an approach that combines a coarse stabilizer for global adjustments and a fine stabilizer for detailed corrections. These modules operate on moving average filters to reduce noise without explicit camera trajectory optimization. After stabilization, an inpainting module fills in the margins.

**Another highlight of the method is its self-supervised nature,** allowing it to produce strong stabilization effects without requiring ground truth videos for training. This distinguishes it from supervised methods that necessitate capturing simultaneously one unstable and another stabilized footage using gimbals.

**In summary,** the method offers a real-time solution with minimal cropping, low visual distortion, and high stability, achieved through a combination of coarse and fine stabilizers and an inpainting module. The self-supervised training enables it to produce strong stabilization effects without requiring ground truth videos, making it a promising approach for real-time applications.
• 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) \cite{12}, depth information \cite{16}, 3D plane constraints \cite{33}, 3D reconstruction \cite{3}, light fields \cite{27}, gyroscopes \cite{1, 11, 24}, and partial 3D information \cite{7, 13, 14}. Image-based methods have also shown sufficient quality \cite{3, 6, 10, 22, 32}. Grundmann et al. \cite{8} apply L1-norm optimization for camera path computation and extend the approach to handle rolling shutter effects \cite{9}. Liu et al. \cite{17} model camera paths for each image patches, and extend the idea to model the entire pixel profiles \cite{18}. This work shows that spatially smoothed flow is useful for stabilizing frames, which our approach builds upon. Recently, Choi and Kweon \cite{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. \cite{19} proposes computing sparse mesh profiles via applying median filters to pixel profiles. This idea is extended further to video coding \cite{20}. Limitations of these methods are the slight wobbling artifacts and relatively low stabilization quality compared to offline methods. Recently, deep learning-based algorithms \cite{21} take advantage of fast feed-forward capabilities. Wang et al. \cite{29} propose a supervised learning approach to video stabilization by defining the novel stability and temporal loss terms. Similarly, Xu et al. \cite{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 \cite{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 transformation estimation and image inpainting. Deep homography estimation \cite{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 \cite{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
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 \(F_{j \rightarrow i}\) 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 \(p_{j \rightarrow i}\) of the adjacent time stamps \(j \in \mathcal{N}\). They are averaged to produce a transformation matrix \(T\). It is obtained from \(\bar{p}_i\) as defined below:

\[
\bar{p}_i := \frac{1}{2N} \sum_{j=-N}^{N} p_{i+j \rightarrow i}. \tag{1}
\]

\(T\) adjusts the target frame to produce \(C_i = T(I_i)\) for stabilization.\(^2\) This procedure is equiva-

\(^1\)We empirically found the 3-DoF rigid transformation \([R(\theta) \, 0 \, 1]\) 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 \(T\).
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 $I_{im}$. The Margin-Net concatenates the original $I_{im}$ 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, 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 $p$ (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 $T(I_i)$. Then, the flow map between $I_i$ and $T(I_i)$ is computed and fed to the Coarse-Net which estimates the parameter vector $p'$. $I_i$ is then

---

4It 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 $p'$ and results in $C_i$. We use the L1 loss function between the randomly transformed frame $T(I_i)$ and the estimated transformed frame $C_i$ defined as:

$$\mathcal{L}_t = \|T(I_i) - C_i\|_1$$  \hspace{1cm} (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 $p$ and the estimated vector $p'$. The loss is $\mathcal{L}_p = \|p - p'\|_1$.

**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 $I_{i-K}, \ldots, I_{i-1}$ and $I_{i+1}, \ldots, I_{i+K}$. In this training phase, the optical flow is computed via inputs $(I_i, I_{i+k})$, and the Coarse-Net estimates the transform parameters $p_{i+k}$. $p_{i+k}$ transforms $I_i$ and produces $C_{i+k}$. Now we apply the sum of L1 losses using $I_i$ and $C_{i+k}$, $\mathcal{L}_n = \sum_{k \in \{1, K\}} \|I_i - C_{i+k}\|_1$. 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 $T(I_i)$ in Eq. (2) and compute the optical flow between $T(I_i)$ 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 $W(I_i)$. The L1 loss is applied between $T(I_i)$ and the estimated frame $W(I_i)$, such as $\mathcal{L}_f = \|T(I_i) - W(I_i)\|_1$.

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: $\mathcal{L}_N = \sum_{k \in \{-1, 1\}} \|I_i - F_{i+k}\|_1$, 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 $T(I_i)$ 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 $T(I_i)$, this procedure preserves the content at the frame boundaries, as shown in Fig 5. Let us call this image $TC(I_i)$.

\(^5\)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 $I^n_i$, utilizing the information from the given adjacent frames. The Margin-Net is trained by the L1 loss between the transformed frame $TC(I_i)$ and the inpainted image $I^n_i$: $L_g = \|TC(I_i) - I^n_i\|_1$.

Furthermore, we can isolate the loss to focus only on the inpainted region by applying a mask to both $TC(I_i)$ and $I^n_i$: $L_I = \|m \odot (TC(I_i) - I^n_i)\|_1$, 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: $L_M = L_g + L_I$.

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 ($640 \times 360$), HD ($1280 \times 720$), and FHD ($1920 \times 1080$). 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 \cite{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.

Evaluation result. As shown in Fig. 6, the comparison with offline methods \cite{7,8,12,13,17,18}, online (real-time) methods \cite{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 \cite{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 \cite{4} and by Yu and Ramanmoorthi \cite{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 \cite{8,17,19,29} 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


