Hybrid Neural Fusion for Full-frame Video Stabilization Supplementary Material

Yu-Lun Liu[1] Wei-Sheng Lai[2] Ming-Hsuan Yang[2,4] Yung-Yu Chuang[1] Jia-Bin Huang[3]
[1]National Taiwan University [2]Google [3]Virginia Tech [4]University of California at Merced
https://alex04072000.github.io/FuSta/

1. Overview

In this supplementary material, we present additional results to complement the main manuscript. First, we provide more implementation and training details (Section 2). Next, we introduce the differentiable fusion functions (Section 3), the evaluation metrics used in our experiments (Section 4), and the dataset statistics (Section 5). Then, we present more ablation studies, including the fusion space, path adjustment, blurry frame removal, and residual detail transfers (Section 6), and provide per-category evaluation and cumulative scores on video stabilization datasets (Section 7). In addition, we analyze the temporal coherence of our approach (Section 8) and demonstrate additional applications on video completion and FOV expansion (Section 9). Finally, we provide details of our user study (Section 10), compare the execution time (Section 11), and discuss the limitation of the proposed method (Section 12). In addition to this document, we also provide an interactive HTML interface to compare our video results with state-of-the-art methods.

2. Implementation Details

2.1. Network architecture

As shown in Figure 1, our network consists of the feature extractor, warping layer, fusion function, frame generator, and the final weighted sum step to generate the output frame.

Feature extractor. Before warping and fusion, we encode each source frame $\{I_{n^s}\}_{n\in\Omega_k}$ to a higher dimensional feature map. The image encoder consists of 8 ResNet blocks [4] with a stride of 1. All convolutional layers are followed by a ReLU [11] activation function. The final encoded features are 32-dimensional for each location and have the same resolution as the input image.

Frame generator. The frame generator uses a encoder-decoder U-Net architecture. The encoder has two downsampling stages, where each stage has a ResNet block and an average pooling layer. Similarily, the decoder has two upsampling stages, and each stage has a ResNet block and a bilinear upsampling layer. All the ResNet blocks are constructed by the gated convolution [17] followed by a ReLU [11] activation function.

CNN-based fusion. The fusion module use the same architecture as the frame generator, where the fused feature is a linear weighted sum of all the output features from the fusion module, as shown in Figure 4 in the main manuscript.

Residual detail transfer. We observe that the synthesized frames may not preserve enough high-frequency details as in the input frames. To tackle this problem, we adopt the residual detail transfer technique similar to [10] to recover the missing residual details to the output frames. As shown in Figure 1, we first feed the image feature f_{k^s} (before warping) into the frame generator module to reconstruct an image, and then compute the residual ΔI_{k^s} by subtracting the reconstructed image with the input image I_{k^s}. Next, we warp the residual ΔI_{k^s} using flow $F_{k^t\to k^s}$ to obtain the *warped residual* $\Delta I^{k^t}_{k^s}$. Finally, we add the warped residual back to the generated image $I^{k^t}_{k^s}$ before the final weighted sum stage. This residual detail transfer helps us recover more high-frequency details and improve the overall sharpness of the synthesized frame.

Note that our residual detail transfer differs from [10] in that 1) we transfer the residual to a virtual stabilized frame (as opposed to the original frame) and 2) our method does not require per-scene/video training. Our method is also different from the residual learning used in image restoration tasks. For example, in image super-resolution [7, 6], the upsampled low-resolution input image (i.e., the *base*) is added with the network output, such that the network can focus on *predicting*

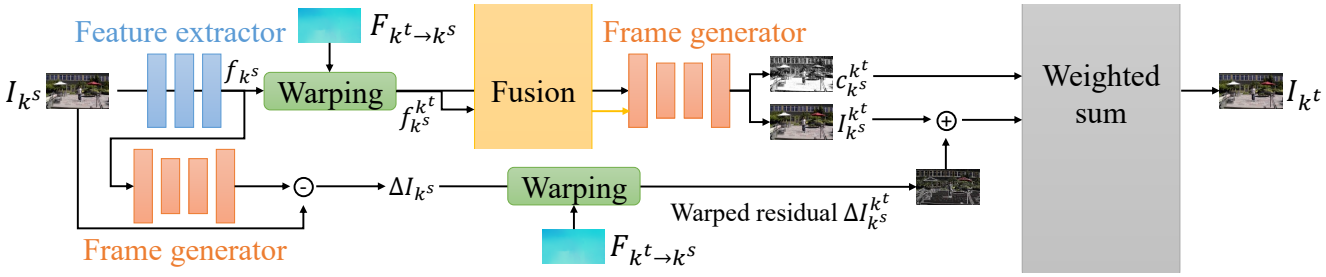


Figure 1: **Network architecture.** The feature extractor first encodes each source frame I_{k^s} into its feature f_{k^s}. The feature is then warped using the given flow field $F_{k^t \to k^s}$. The warped features $f_{k^s}^{k^t}$ are fused into the fused feature $f_{CNN}^{k^t}$ using the proposed CNN-based fusion. For each frame, its warped feature $f_{k^s}^{k^t}$ is concatenated with the fused feature $f_{CNN}^{k^t}$ as the input to the frame generator. The frame generator obtains the rendered frame $I_{k^s}^{k^t}$ and its corresponding confidence map $c_{k^s}^{k^t}$. As rendering (warped) image features using a decoder may suffer from loss of high-frequency details, we compute the residuals and transfer them to the target frames to recover these missing details. We first reconstruct the frame (without warping) by giving the extracted feature f_{k^s} as input to the frame generator. Then, we subtract the reconstructed image with the input image I_{k_s} to obtain the residual ΔI_{k^s}. Next, we warp this residual using flow $F_{k^t \to k^s}$ to obtain the warped residual $\Delta I_{k^s}^{k^t}$. Finally, we add this warped residual back to the generated image $I_{k^s}^{k^t}$. The final weighted sum stage combines the generated images $I_{k^s}^{k^t}$ using the confidence maps $c_{k^s}^{k^t}$ for rendering the output stabilized frame I_{k^s}.

the residual details. However, in our case, we do not have any pixel information of the target frame. Our network has to *predict the base* of the target frame, where the residual details are added back using our proposed approach in Figure 1.

2.2. Blurry frame removal

For videos with rapid camera or object motion, the input frames may suffer from severe blur. Fusing information from these blurry frames inevitably leads to blurry output. We thus use the method of [12] to compute a sharpness score for each neighboring frame and then select the top 50% sharp neighboring frames for fusion. This pre-processing step helps restore sharper contents and improves the runtime.

2.3. Training details

We set the weighting coefficients of pretrained VGG-19 network to: $\lambda_1 = 1/2.6$, $\lambda_2 = 1/4.8$, $\lambda_3 = 1/3.7$, $\lambda_4 = 1/5.6$, $\lambda_5 = 10/1.5$. We train our network using the Adam optimizer with a learning rate of 0.0001. The training patch size is 256×256. We train the model for 50 epochs with a batch size of 1. The training process takes about four days to converge on a single V100 GPU.

2.4. Synthetic data generation

As described in Section 3.4 of the main paper, to generate the training data, we sample a 7-frame sequence from a video and apply random cropping to simulate a shaky input video. We generate the input sequence by randomly cropping 256×256 patches to simulate *unstable* frames. We then apply another random crop at the center frame to obtain the target *stabilized* frame. We sample 7-frame sequences in a sliding window manner and use the center frame as the target keyframe. Figure 2 visualizes the process to synthesize the training and test data. We use the training set of [14] to synthesize training sequences, and use the test set of [14] to generate testing sequences for quantitative evaluation. In total, we generate 5734 training sequences and 940 test sequences.

3. Fusion Functions

In this section, we introduce the alternative fusion functions we have compared with in Section 4.1 of the manuscript.

1) Mean fusion: We compute the mean of all available pixels/features at each location:

$$f_{\text{mean}}^{k^t} = \frac{\sum_{n \in \Omega_k} f_{n^s}^{k^t} M_{n^s}^{k^t}}{\sum_{n \in \Omega_k} M_{n^s}^{k^t} + \tau}, \tag{1}$$

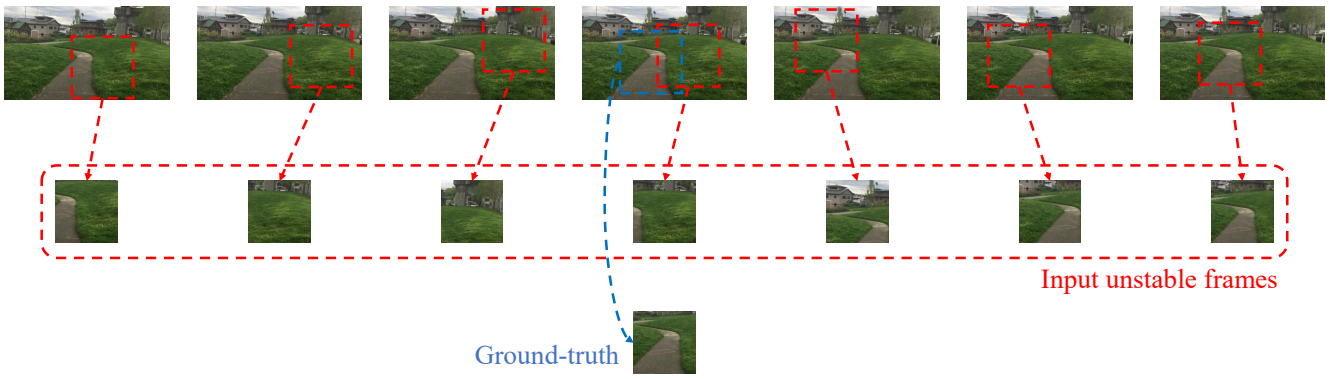


Figure 2: **Shaky video simulation.** We apply random cropping to synthesize unstable input videos and the ground-truth keyframe.

where $f_{n^s}^{k^t}$ and $M_{n^s}^{k^t}$ is the encoded feature map and warping mask of frame n, respectively. The superscript k^t denotes that the encoded feature and warping mask are warped and aligned with the target frame. τ is a small constant to avoid dividing by zero.

2) Gaussian-weighted fusion: We extend the mean fusion by assigning higher weights for neighbor frames that are temporally closer to the keyframe:

$$f_{\text{gauss}}^{k^t} = \frac{\sum_{n\in\Omega_k} f_{n^s}^{k^t} M_{n^s}^{k^t} W_n}{\sum_{n\in\Omega_k} M_{n^s}^{k^t} W_n + \tau}, \tag{2}$$

where $W_n = G(k, \|n-k\|)$ is the Gaussian blending weight.

3) Argmax fusion: Instead of linear blending (which may cause blur), we can take the pixels/features with the maximal weight, where the weight is a linear combination of the warping mask $M_{n^s}^{k^t}$ and the temporal Gaussian weight W_n:

$$f_{\text{argmax}}^{k^t} = f_{\arg\max_n M_{n^s}^{k^t} W_n}. \tag{3}$$

This function is similar to the winner-takes-all labeling strategy.

4) Flow error-weighted fusion: We replace the Gaussian blending weight in (2) with a confidence score based on the flow error:

$$f_{\text{FE}}^{k^t} = \frac{\sum_{n\in\Omega_k} f_{n^s}^{k^t} M_{n^s}^{k^t} W_n^{\text{FE}}}{\sum_{n\in\Omega_k} M_{n^s}^{k^t} W_n^{\text{FE}} + \tau}. \tag{4}$$

The confidence score $W_n^{\text{FE}} = \exp(-e_{n^s}^{k^t}/\beta)$ is calculated from the forward-backward flow consistency error $e_n^{\hat{k}}$:

$$e_{n^s}(p) = \| |F_{k^s\to n^s}(p) + F_{n^s\to k^s}(p + F_{k^s\to n^s})| \|_2, \tag{5}$$

where p denotes pixel coordinate in $F_{k^s\to n^s}$, and we set $\beta = 0.1$ in all our experiments. Note that the forward-backward flow consistency errors are calculated from the input unstabilized frames.

4. Evaluation Metrics

We use the PSNR, SSIM, and LPIPS [20] scores to evaluate the quality of frame synthesis on our synthetic test set for the ablation studies. We use the following metrics to evaluate the performance of video stabilization, which are commonly used in previous work [9, 1, 18, 19].

Cropping ratio. The cropping ratio measures the remaining frame area after cropping off the irregular boundaries and undefined pixels due to warping. A larger cropping ratio indicates less cropping and preserves more video content. We first fit a homography between the input and output frames. The cropping ratio of a video is calculated by averaging the scale component of the homography across the entire video.

Distortion. Distortion is measured by the anisotropic scaling of the homography between the input and output frames. Specifically, it can be computed by the ratio of the two largest eigenvalues of the affine part in a homography matrix. The distortion score is defined as the minimal ratio across all the frames in a video.

Stability. This metric measures the stability and smoothness of the stabilized video. The frequency-domain analysis is performed on the 2D camera motion of the output video. Two 1D temporal signals are extracted from the translation and rotation components from each path. The ratio of the sum of the lowest (the 2nd to the 6th) frequency energy over the total energy is computed. The final score is obtained by taking the minimum across the entire video.

Accumulated optical flow (A). This metric is defined by accumulated optical flow over the entire video:

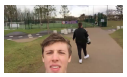
$$\frac{1}{wh(T-1)} \sum_{t=1}^{T-1} \sum_{x=1}^{h} \sum_{y=1}^{w} (\|F_{t\to t+1}(x,y))\|_2 + \|F_{t+1\to t}(x,y))\|_2), \tag{6}$$

where w is the frame width, h is the frame height, t is the frame index of time, T is the total number of frames, x, y are the pixel coordinates, and $F_{t\to t+1}$ is the optical flow from time t to $t+1$. The optical flow is normalized by its frame size in order to compare videos with different resolutions. In this metric, a smaller value means that the video has smaller motion, indicating a more stable result. This score value is further normalized by the score of the input video to show the improvement. We use RAFT [15] to compute the bidirectional optical flow for each video.

5. Video Stabilization Datasets

Table 1 summarizes the statistics of the three video stabilization datasets: the NUS dataset [9], the selfie dataset [18] and the DeepStab dataset [16] used in the quantitative and visual comparisons.

Table 1: **Dataset summarization.** The NUS dataset [9] contains six categories, including *simple*, *quick rotation*, *zooming*, *large parallax*, *crowd*, and *running*. The selfie dataset [18] has 33 selfie video clips collected from the Internet. The DeepStab dataset [16] has 61 video clips taken in a first-person point of view. In total, we evaluate other state-of-the-art methods and our method on these 238 video clips.

Dataset	NUS						Selfie	DeepStab
Category	Simple	Quick Rotation	Zooming	Large Parallax	Crowd	Running	-	-
Sample frame								
#videos	23	29	29	18	23	22	33	61

6. Additional Ablation Studies

6.1. Fusion space

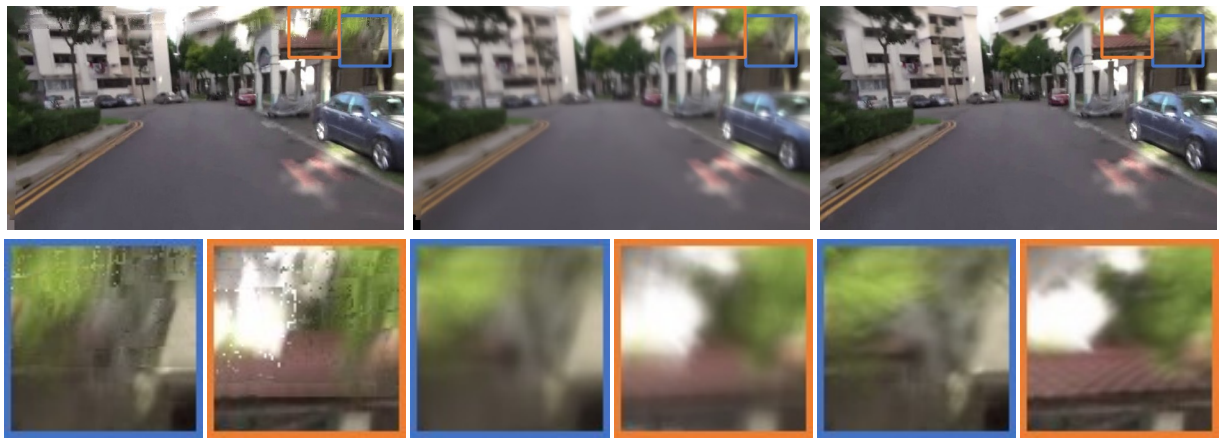
Figure 3 shows an example to demonstrate the rendered frames using different fusion spaces. Image-space fusion often generates glitching artifacts or visible seams. Feature-space fusion typically results in blurry predictions. In contrast, our hybrid-space fusion can avoid visual artifacts and synthesize sharper output frames.

6.2. Path adjustment

Figure 4 demonstrates the effect of our path adjustment, where the synthesized frames without path adjustment contain boundary artifacts (see the red arrows in Figure 4(a)) if those pixels are not visible in any of the neighbor frames. Table 2 shows that path adjustment does not significantly affect the stability, distortion, and cropping ratio of the output videos.

6.3. Blurry frame removal

Figure 5 demonstrates that, by removing blurry frames from fusion, our method can generate sharper frames with more details. At the same time, we can reduce the computational cost of flow estimation and frame synthesis. Note that the proposed fusion method is able to tackle arbitrary input frames.



(a) Image-space fusion (b) Feature-space fusion (c) Hybrid-space fusion

Figure 3: **Effect of different blending spaces.** (a) Blending multiple frames at the pixel-level retains sharp contents but often leads to visible artifacts due to the sensitivity to flow inaccuracy. (b) Blending in the feature level alleviates these artifacts but has difficulty in generating sharp results. (c) The proposed hybrid-space fusion method is robust to flow inaccuracy and produces sharp output frames.

(a) w/o path adjustment (b) w/ path adjustment

Figure 4: **Effect of the path adjustment.** (a) Without path adjustment, the synthesized frames contain visual artifacts on the image boundaries when the pixels are not visible in any of the neighbor frames. (b) With our path adjustment, most the pixels in the target frame can be found in the neighbor frames, which significantly reduce the boundary artifacts.

6.4. Residual detail transfer

Figure 6 demonstrates that our residual detail transfer can recover more high-frequency details to improve the visual quality. Table 3 shows that the quantitative results in terms of PSNR, SSIM and LPIPS scores are improved after applying the residual detail transfer.

Table 2: **Ablation study of path adjustment on the Selfie dataset [18].** With path adjustment, the results of cropping ratio, distortion value, and stability score remain the same. The accumulated optical flow becomes slightly worse, but the output frames contain less boundary artifacts as shown in Figure 4.

	Cropping ratio ↑	Distortion value ↑	Stability score ↑	Accumulated flow ↓
w/o path adjustment	**1.00**	**0.86**	**0.84**	**0.60**
w/ path adjustment	**1.00**	**0.86**	**0.84**	0.64

Table 3: **Ablation study of residual detail transfer.** See Figure 6 for visual comparisons.

	LPIPS ↓	SSIM ↑	PSNR ↑
w/o residual detail transfer	0.073	0.914	27.868
w/ residual detail transfer	**0.056**	**0.942**	**29.255**

7. Quantitative Evaluation

7.1. Per-category evaluation

Figure 7 shows the per-category evaluation on the NUS dataset [9]. Our method and DIFRINT [1] has the highest cropping ratio (close to 1) as both methods generate full-frame results without cropping. For the distortion and stability metrics, our method performs on par with the best state-of-the-art methods (DIFRINT [1] in distortion and Adobe Premiere Pro 2020 warp stabilizer in stability). Our method obtains the lowest accumulated optical flow in most categories except the rotation and crowd category, which have more challenging cases. Generally, the proposed method is robust and performs well in different scenarios.

7.2. Cumulative scores

Figure 8 shows the cumulative scores of the evaluation metrics on the NUS, Selfie, and DeepStab datasets. For the cropping ratio, distortion value, and stability scores, our curves are typically above the other methods. For the accumulated optical flow, our curves are below other methods.

8. Temporal Coherency

We slice the center vertical line of the output videos over time to show the temporal coherency of our method. Figure 9 shows that our method achieves smoother temporal coherency comparing to other methods and the input video.

9. Additional Applications

9.1. Video Completion

We demonstrate that our method can also be applied to video completion. We use the same flow smoothing [19] and apply a state-of-the-art video completion method [2] to fill in the blank regions. Note that [2] can be regarded as an image-based fusion method[1]. Figure 10 shows that [2] generates visible artifacts due to wrong optical flows, while our method produces more visually pleasing results by hybrid-space fusion.

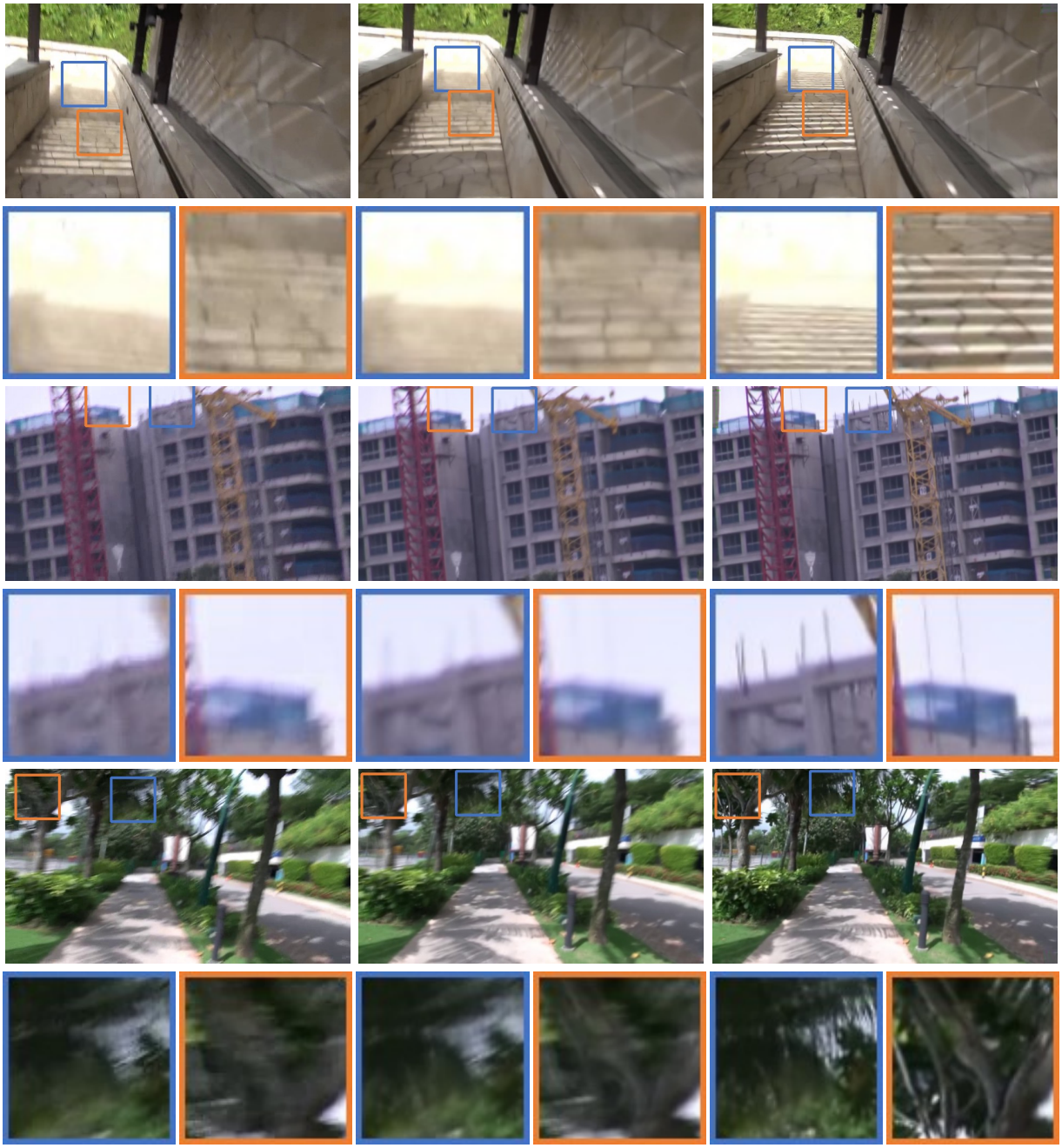
9.2. FOV Expansion

Our method can be extended to generate videos with a larger field of view than the input videos. Figure 11(a) shows a video frame where the camera is moving horizontally to the right. Our method can use information from neighbor frames to expand the horizontal FOV in Figure 11(b). Figure 11(c) shows a video frame where the camera is zooming out. Our method can expand the FOV in all directions to include more content in a single view.

9.3. Comparisons to DeepBlending [5]

We compare our method to a learning-based fusion method DeepBlending [5], which belongs to the image-space fusion. Figure 12 shows our synthesized frames have less artifacts.

[1] As the source code of [2] frequently crashes on these outpainting scenarios, we are not able to conduct a full quantitative evaluation on their dataset.



(a) Blurry keyframe (b) w/o blurry frame removal (c) w/ blurry frame removal

Figure 5: **Effect of selecting and removing blurred frames.** Rapid camera or object motion leads to blurry frames (a). Stabilizing the videos using all these frames inevitably results in blurry outputs (b). By selecting and removing blurred frames in a video, our method re-renders the stabilized video using only *sharp* input frames and thus can avoid synthesizing blurry frames.

10. User Study

As the quality of video stabilization is a subjective matter of taste, we also conduct a user study to compare the proposed method and three approaches: Yu and Ramamoorthi[19], DIFRINT [1], and Adobe Premiere Pro 2020 warp stabilizer. We randomly select two videos from the Selfie dataset [18] and two videos from each category of the NUS dataset [9], resulting

(a) w/o residual detail transfer (b) w/ residual detail transfer

Figure 6: **Effect of the residual detail transfer.** The residual detail transfer restores more high-frequency details in the output frames.

in total $2 \times (6 + 1) = 14$ videos for comparison. We adopt the pairwise comparison [13, 8], where each video pair has three questions:

1. Which video preserves the most content?
2. Which video contains fewer artifacts or distortions?
3. Which video is more stable?

The orders of the videos (including left-right and sequential orders) are randomly shuffled for each subject. We show the

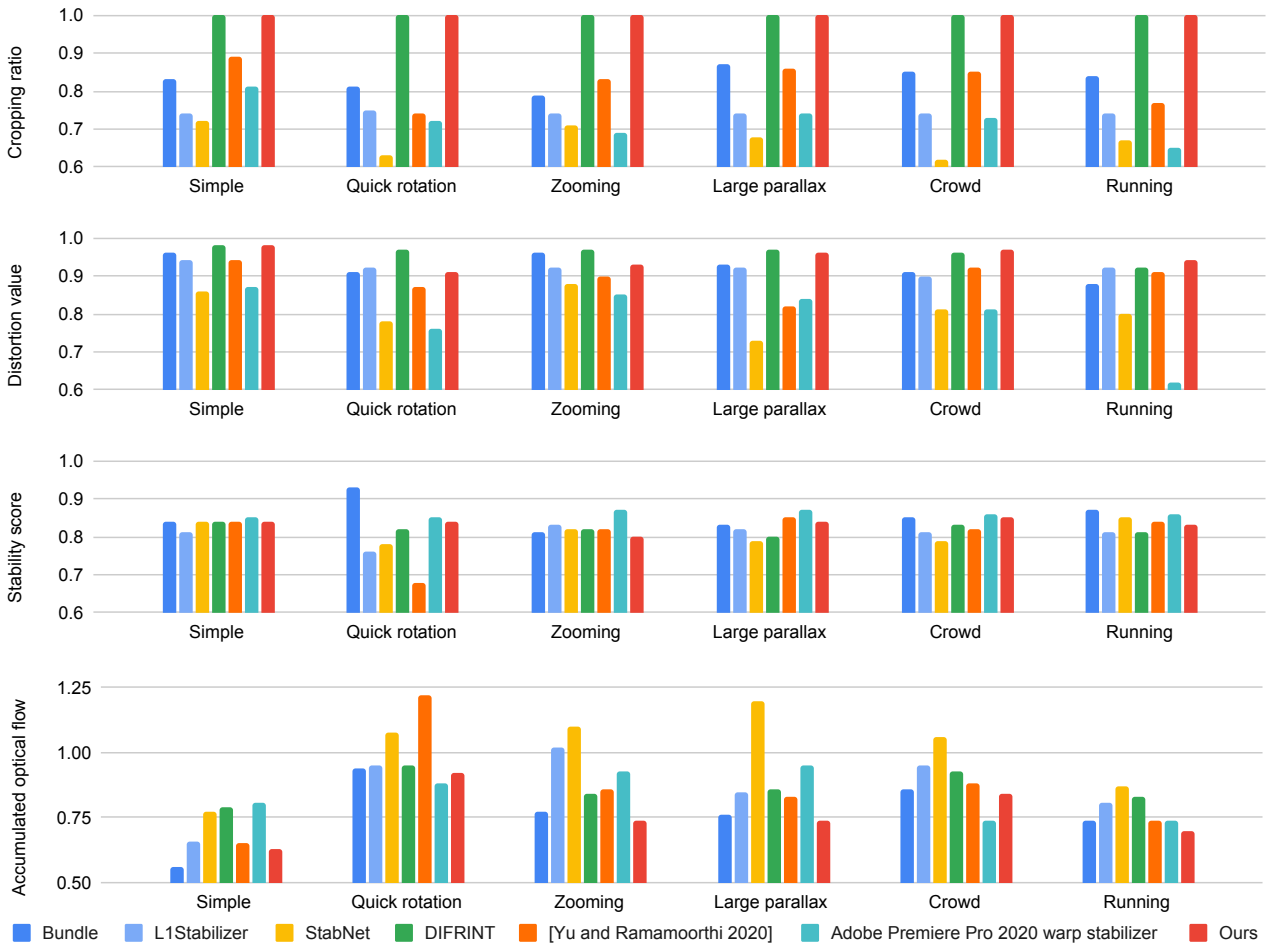


Figure 7: **Evaluation on each category of the NUS dataset [9].** For cropping ratio, distortion value, and stability score, a larger value indicates a better result. For accumulated optical flow, a smaller value indicates a better result. Our method is robust in all categories, achieving state-of-the-art or comparable performance with existing approaches.

input video to users as a reference.

From this user study, we collect the results from 46 subjects, who are recruited online without any domain knowledge. We show the winning rate of our method against the compared approaches in Figure 13. Overall, our results are preferred by the users in more than 60% of the comparisons, demonstrating that the proposed method outperforms existing approaches in the subjective evaluation.

Figure 14 shows a screenshot of our interactive webpage for conducting the user study. For each sequence, the user is asked to select the winner out of two comparing methods in terms of three criterion. One of the comparing videos is from our method. Note that we also provide the unstable input video as a reference. The sequence order and left-right order are randomly shuffled.

11. Runtime Analysis

We measure the runtime of CPU-based approaches [3, 9, 18], on a laptop with i7-8550U CPU. For our method and GPU-based approaches [16, 19, 1], we evaluate on a server with Nvidia Tesla V100 GPU. We use sequence #10 from the Selfie dataset for evaluation, which has a frame resolution of 854×480. We show the runtime comparisons in Table 4. We also provide the runtime profiling of our method in Table 5. The pre-processing step of motion smoothing from [19] takes about 71% of our runtime. Our method can be accelerated by using a more efficient motion smoothing algorithm. However, the quality of warping and fusion also depends on the accuracy of the motion estimation and smoothing.

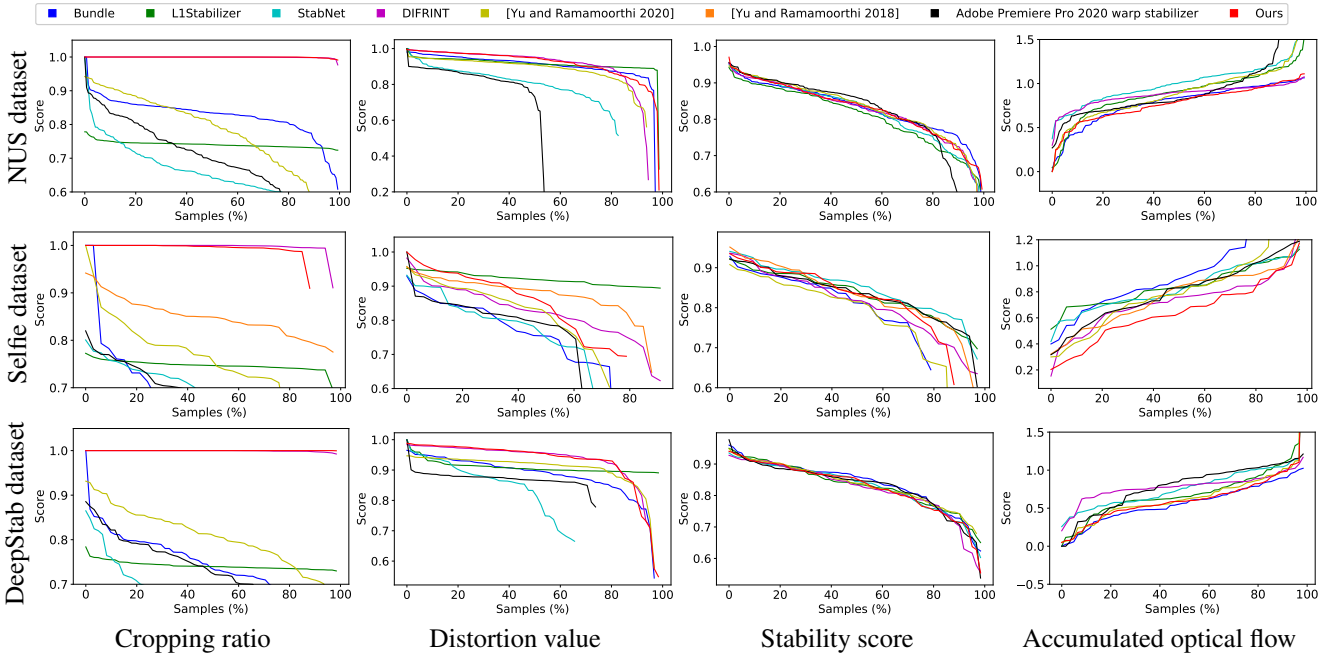
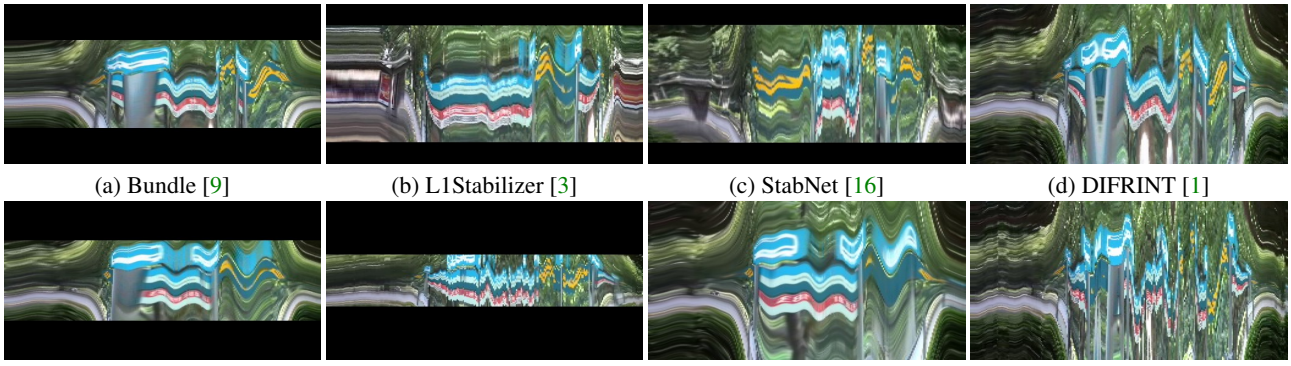


Figure 8: **Quantitative evaluation on the NUS dataset [9], the selfie dataset [18], and the DeepStab dataset [16].** As some of the methods fail to produce results for several sequences, computing averaged scores of successful sequences does not reflect the robustness of the method. For each method, we store all the error metrics for each sequence and plot the *sorted* scores. These plots reflect both the *completeness* and the *performance* (in cropping ratio, distortion, stability, and smoothness). Our approach achieves full-frame stabilization (except for a few sequences where [19] fails to generate results) while maintaining competitive performance compared with the state-of-the-art.



(a) Bundle [9] (b) L1Stabilizer [3] (c) StabNet [16] (d) DIFRINT [1]

(e) Yu and Ramamoorthi [19] (f) Adobe warp stabilizer (g) Ours (h) Input

Figure 9: **Temporal coherency comparisons to state-of-the-art methods.** Bundle [9], Deep online video stabilization [16], and Adobe Premiere Pro 2020 warp stabilizer suffer from large cropping. L1Stabilizer [3] and Yu and Ramamoorthi [19] generate smooth videos with cropping. DIFRINT [1] produces glitching motions. Our stable video is smooth and preserves the most content in the input video.

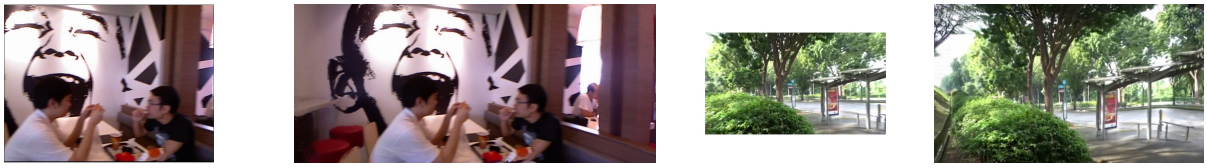
12. Limitations

Video stabilization for videos in unconstrained scenarios remains challenging. Here we discuss several limitations of our approach. We refer to the readers to our supplementary material for video results.

Wobble. When the camera or object motion is too rapid, we observe that the stabilized frame exhibit *rolling shutter wobble* (Figure 15(a)). Integrating rolling shutter rectification into our approach can potentially alleviate such an issue.

(a) Yu and Ramamoorthi [19] (b) Yu and Ramamoorthi [19] + Gao *et al*. [2] (c) Ours

Figure 10: **Comparisons to the video completion method [2].** (a) [19] crops off the blank regions caused by warping. (b) The video completion method [2] utilizes stable neighbor frames to recover the missing pixels. However, the results contain visible discontinuity artifacts due to wrong flows. (c) Our method produces full-frame and artifact-free results.



(a) Input (b) Ours + FoV expansion (c) Input (d) Ours + FoV expansion

Figure 11: **Effect of FoV expansion.** (a) (c) Input unstable frames. (b) (d) Our method utilizes nearby information to recover the missing regions due to motion compensation. It can also expand the field-of-view of the output videos. The example on the left is from the parallax category. The example on the right is from the zooming category.

Visible seams. As our method fuse multiple frames to re-render a stabilized frame, our results may contain visible seams, particularly when there are large lighting variations caused by camera white-balancing and exposure corrections (Figure 15(b)).

Temporal flicker and distortion. Our method builds upon an existing motion smoothing method [19] to obtained stabilized frames. However, the method [19] may produce temporally flickering results due to large non-rigid occlusion and inconstant foreground/background motion (e.g., selfie videos). In such cases, due to the dependency of motion inpainting, our method also suffers from undesired temporal flicker and distortion (Figure 15(c)).

Speed. Our work aims for offline application. Currently, our method runs slowly (about 10 seconds/frame) compared to many existing methods. Speeding up the runtime is important future work.

References

[1] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM TOG*, 2020. 3, 6, 7, 9, 10, 13

(a) DeepBlending [5] (b) Ours

Figure 12: **Comparisons to the learning-based image fusion method DeepBlending [5].**

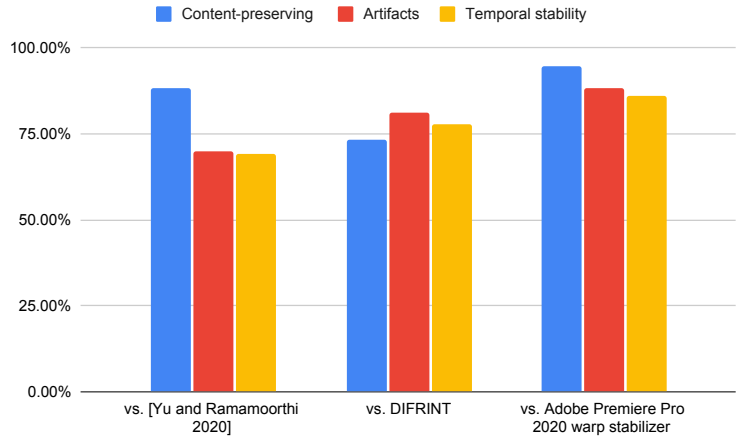


Figure 13: **User preference of our method against other state-of-the-art methods.** We ask the subjects to rate their preferences over the video stabilization results in a randomized paired comparison (ours vs. other) in terms of content preservation, visual artifacts, and temporal stability. Overall our results are preferred by the users in all aspects.

[2] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. In *ECCV*, 2020. 6, 11

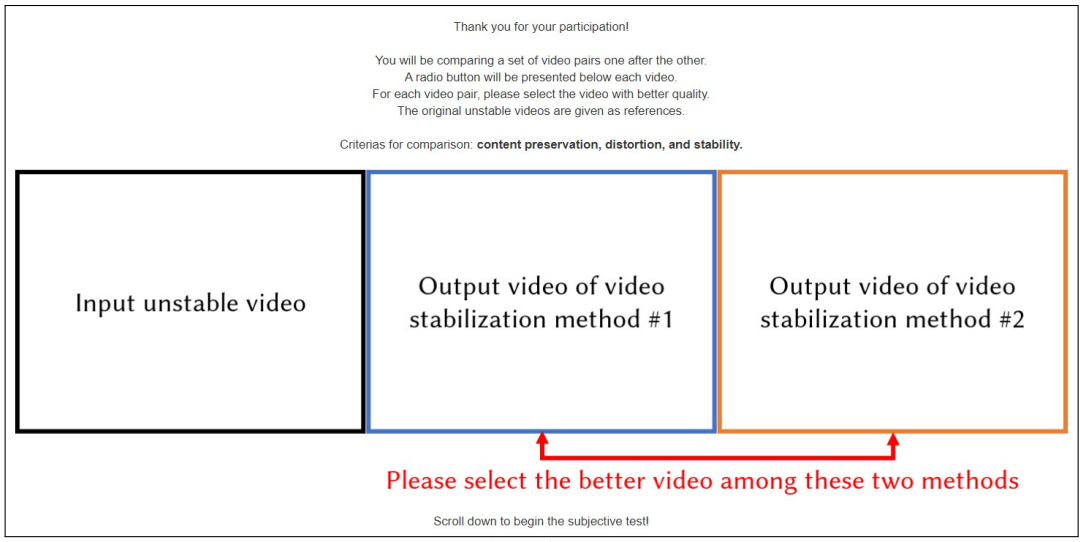
[3] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust L1 optimal camera paths. In *CVPR*, 2011. 9, 10, 13

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

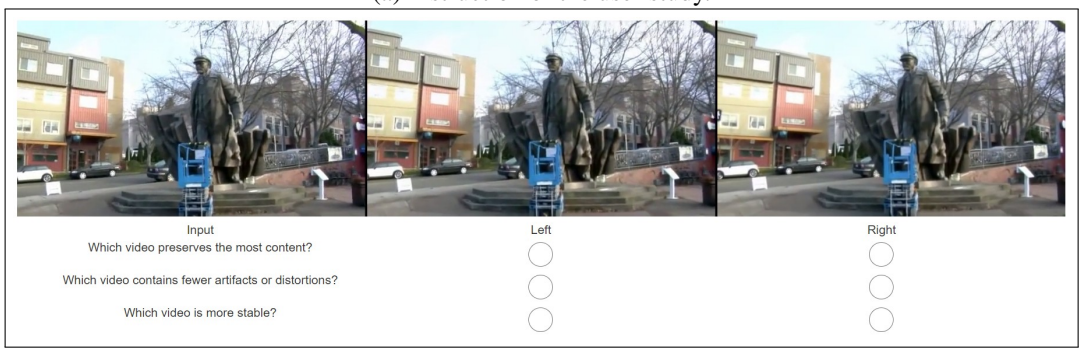
[5] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM TOG*, 2018. 6, 12

[6] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1

[7] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 1



(a) Instruction of the user study.



(b) Interface of the user study.

Figure 14: **User study interface.** We ask the user to answer three questions for each video sequence, which corresponds to the cropping ratio, distortion value, and stability. *Left*: Input unstable video for reference. *Middle*: Randomly selected method #1. *Right*: Randomly selected method #2. Note that one of the compared methods is ours.

Table 4: **Per-frame runtime comparison.** CPU-based methods are evaluated on a laptop with i7-8550U CPU, while GPU-based methods are evaluated on a server with Nvidia Tesla V100 GPU. The testing video has a frame resolution of 854×480. *: Time is reported in [18] as the source code is not available.

Method	Environment	Implementation	Runtime (s)
L1Stabilizer [3]	CPU	Matlab	0.623
Bundle [9]	CPU	Matlab	7.612
StabNet [16]	GPU	TensorFlow	0.073
DIFRINT [1]	GPU	PyTorch	1.627
Yu and Ramamoorthi [19]	GPU	PyTorch	6.501
Yu and Ramamoorthi [18]	CPU	Matlab	15 mins*
Adobe Premiere Pro 2020 warp stabilizer	CPU	-	0.045
Ours excluding [19]	GPU	PyTorch	3.090
Ours	GPU	PyTorch	9.591

[8] Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, Narendra Ahuja, and Ming-Hsuan Yang. A comparative study for single image blind deblurring. In *CVPR*, 2016. 8

[9] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM TOG*, 2013. 3, 4, 6, 7, 9, 10,

Table 5: **Runtime breakdown of the proposed method.**

Stage	Runtime (s)	Percentage
Motion smoothing [19]	6.826	71.03%
Feature extractor	0.724	7.53%
Warping	0.207	2.15%
CNN-based fusion	0.952	9.91%
Image generator	0.881	9.17%
Final fusion	0.020	0.21%

(a) Wobble (b) Visible seams (c) Temporal flicker and distortion

Figure 15: **Limitations.** (a) Our method does not correct rolling shutter and thus may suffer wobble artifacts. (b) Our fusion approach is not capable of compensating large photometric variations across frames, resulting in visible seams. (c) Existing smoothing approach [19] may introduce distortion, particularly when videos contain non-rigid large occlusion (such as selfie videos). As our method depends on the flow produced by [19] for stabilizing the video, we cannot correct such errors.

13

[10] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. *ACM TOG*, 2020. 1

[11] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 1

[12] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *ICPR*, 2000. 2

[13] Michael Rubinstein, Diego Gutierrez, Olga Sorkine, and Ariel Shamir. A comparative study of image retargeting. In *ACM SIGGRAPH Asia*. 2010. 8

[14] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017. 2

[15] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 4

[16] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 2018. 4, 9, 10, 13

[17] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *CVPR*, 2019. 1

[18] Jiyang Yu and Ravi Ramamoorthi. Selfie video stabilization. In *ECCV*, 2018. 3, 4, 6, 7, 9, 10, 13

[19] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *CVPR*, 2020. 3, 6, 7, 9, 10, 11, 13, 14

[20] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 3