Real-Time Selfie Video Stabilization
Supplementary Material

Jiyang Yu[1,3] Ravi Ramamoorthi[1] Keli Cheng[2] Michel Sarkis[2] Ning Bi[2]
[1]University of California, San Diego [2]Qualcomm Technologies Inc. [3]JD AI Research, Mountain View
jiy173@eng.ucsd.edu ravir@cs.ucsd.edu {kelic,msarkis,nbi}@qti.qualcomm.com

Table a. Notations in the paper and supplementary material

Symbols	Explanation
t	Frame index
$\mathbf{M}_t$	Foreground mask
$\mathbf{P}_t$	Background feature points
$\mathbf{Q}_t$	Correspondence of $\mathbf{P}_{t-1}$ in frame t
$\mathbf{F}_t$	Face vertices
$\widehat{\mathbf{Q}}_t$	Target coordinate of $\mathbf{Q}_t$
$\mathbf{v}$	Coordinate of a pixel
$W(\mathbf{v};\mathbf{Q}_t,\widehat{\mathbf{Q}}_t)$	Rigid MLS warping function
$\widehat{\mathbf{v}}$	Warped coordinate of pixel $\mathbf{v}$
$\mathbf{q}_i$	ith column of $\mathbf{Q}_t$
w_i	MLS weight of $\mathbf{q}_{i,t}$ to pixel $\mathbf{v}$
α	MLS parameter
$\mathbf{c}$	Weighted centroid of $\mathbf{Q}_t$
$\widehat{\mathbf{c}}$	Weighted centroid of $\widehat{\mathbf{Q}}_t$
$\mathbf{q}_i^*$	Vector from $\mathbf{c}$ to $\mathbf{q}_{i,t}$
$\widehat{\mathbf{q}}_i^*$	Vector from $\widehat{\mathbf{c}}$ to $\widehat{\mathbf{q}}_i$
$\mathbf{A}_i$	Transformation matrix of $\widehat{\mathbf{q}}_i^*$
$\mathbf{g}_j$	jth grid vertex
$\mathbf{G}$	Grid vertices enclosing $\mathbf{v}$
$\mathbf{D}$	Bilinear weights of $\mathbf{v}$ with respect to $\mathbf{G}$

Algorithm 1: The rigid MLS warping algorithm $W(\mathbf{v};\mathbf{Q},\widehat{\mathbf{Q}})$

```
Input  : Source coordinates of a pixel $\mathbf{v}$, source
         node coordinates $\mathbf{Q}$ and target node
         coordinates $\widehat{\mathbf{Q}}$
Output: Target coordinates of a pixel $\widehat{\mathbf{v}}$
for $i \leftarrow 1$ to 512 do
    $w_i = 1/\left|\mathbf{v}-\mathbf{q}_i\right|^{2\alpha}$
end
$\mathbf{c} = \left(\sum_{i=1}^{512} w_i\mathbf{q}_i\right) / \left(\sum_{i=1}^{512} w_i\right)$
$\widehat{\mathbf{c}} = \left(\sum_{i=1}^{512} w_i\widehat{\mathbf{q}}_i\right) / \left(\sum_{i=1}^{512} w_i\right)$
for $i \leftarrow 1$ to 512 do
    $\mathbf{q}_i^* = (\mathbf{q}_i-\mathbf{c})^T \quad \widehat{\mathbf{q}}_i^* = (\widehat{\mathbf{q}}_i-\widehat{\mathbf{c}})^T$
    $\mathbf{A}_i = w_i\begin{pmatrix}\mathbf{q}_i^* \\ -\mathbf{q}_i^{*\perp}\end{pmatrix}\begin{pmatrix}\mathbf{v}-\mathbf{c} & -(\mathbf{v}-\mathbf{c})^\perp\end{pmatrix}$,
    where $\perp$ is an operator on 2D vector
      $(x,y)^\perp = (-y,x)$
end
$\widehat{\mathbf{v}} = \left|\mathbf{v}-\mathbf{c}\right|\left(\sum_{i=1}^{512}\mathbf{A}_i\widehat{\mathbf{q}}_i^*\right) / \left|\sum_{i=1}^{512}\mathbf{A}_i\widehat{\mathbf{q}}_i^*\right| + \widehat{\mathbf{c}}$
```

A. Implementation Details

A.1. MLS warping process

Table a summarizes the notations used in the main paper and this supplementary material. Algorithm 1 provides the MLS warping process referred in Sec. 3.2. Since the MLS warping is not related to the time dimension, we omit the time subscript t for simplicity. In Algorithm 1, we use relatively small $\alpha = 0.3$ to maintain a smooth warp field and avoid artifacts.

A.2. Sliding Window

Since the stabilization network only takes fixed length video segments, to apply to arbitrary length selfie videos, we apply a sliding window scheme. In our experiment, we use a sliding window with length $T = 5$. We demonstrate our sliding window scheme in Fig. a. Each window is marked by the same color, which is the input to our network for the window. Consider window 1 as an example. The outputs of our stabilization network are the displacements of the warp nodes $\widehat{\mathbf{Q}}_2$, $\widehat{\mathbf{Q}}_3$ and $\widehat{\mathbf{Q}}_4$ as we discussed in the network structure. At this point, if we warp all these frames in the current window and set the next window starting from frame 5, the result will be smooth within each window but not globally smooth. Therefore, we only warp the second frame in the current window and shift one frame for the next window. This scheme ensures temporal consistency between consecutive windows. Specifically, in this example, we use the MLS warp function $W(\mathbf{v};\mathbf{Q}_2,\widehat{\mathbf{Q}}_2)$ to warp frame 2. We then warp the feature points and face vertices using $W(\mathbf{P}_1;\mathbf{Q}_1,\widehat{\mathbf{Q}}_1)$ and $W(\mathbf{F}_1;\mathbf{Q}_1,\widehat{\mathbf{Q}}_1)$, since warping the frame leads to updated positions of the original feature points and face vertices. The updated feature points and

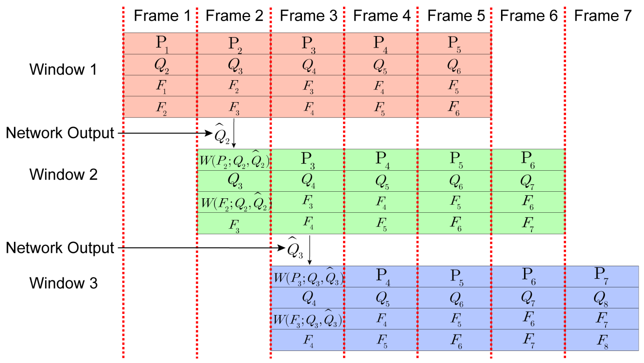

Figure a. *The sliding window scheme of our method. The inputs of our network for each window are marked with the same color. For each window, the second frame is stabilized. The background feature points and the foreground face vertices are updated accordingly and become the next window's input.*

face vertices become a part of window 2, which is the next window starting at frame 2.

B. Network Design

In this section, we extend the discussion regarding the linear network design. We first provide the complete list of parameters in the stabilization network in Sec. B.1. We then discuss the necessity of using a network instead of formulating a linear optimization problem in Sec. B.2. In Sec. B.3, we compare the performance of linear network and direct optimization of the loss function(Eq. 3). In Sec. B.4, we compare the performance of linear network and non-linear network in terms of the quantitative metrics discussed in main paper Sec. 6.3. Finally, we compare the performance using different number of filters in the network in Sec. B.5.

B.1. Network Parameters

Table b lists the network parameters in main paper Fig. 5. The number of filters in each layer is multiple of a base number C, which will be discussed in Sec. B.5 in this supplementary material.

B.2. Necessity of the linear network

In Eq. (1) and Eq. (2) in the main paper, we define the loss function directly on feature points detected in the image. This requires linear relationship between the input and the output of the stabilization network, i.e. scaling of feature point coordinates should lead to the same scaling of the output displacement to compensate the motion. Note that this linear relationship between input and output can be posed as a matrix-vector product, i.e., $\mathbf{n} = \mathbf{A}\mathbf{m}$ where $\mathbf{A} \in \mathbb{R}^{1024(T-1)\times 4096(T-1)}$ is a large matrix that transforms concatenated and reshaped input feature points and face vertices $\mathbf{m} \in \mathbb{R}^{4096(T-1)\times 1}$ to reshaped warp node displacements $\mathbf{n} \in \mathbb{R}^{1024(T-1)\times 1}$. The optimization problem

Table b. Network parameters in main paper Fig. 5

Layer id	Layer Type	Input Size	Output Size	Kernel Size	Stride	Dilation	Padding
1	Conv1d	4(T-1)x512	Cx512	3	1	1	1
2	Conv1d	Cx512	2Cx256	4	2	1	1
3	Conv1d	2Cx256	2Cx256	3	1	1	1
4	Conv1d	2Cx256	4Cx128	4	2	1	1
5	Conv1d	4Cx128	4Cx128	3	1	1	1
6	Conv1d	4Cx128	4Cx128	3	1	1	1
7	Conv1d	4Cx128	8Cx64	4	2	1	1
8	Conv1d	8Cx64	8Cx64	3	1	1	1
9	Conv1d	8Cx64	8Cx64	3	1	2	2
10	Conv1d	8Cx64	8Cx64	3	1	2	2
11	ConvT1d	32Cx64	8Cx128	4	2	2	2
12	Conv1d	8Cx128	8Cx128	3	1	1	1
13	Conv1d	8Cx128	8Cx128	3	1	1	1
14	ConvT1d	16Cx128	4Cx256	4	2	1	1
15	Conv1d	4Cx256	4Cx256	3	1	1	1
16	ConvT1d	8Cx256	2Cx512	4	2	1	1
17	Conv1d	2Cx512	2Cx512	3	1	1	1
18	Conv1d	2Cx512	2(T-2)x512	1	1	1	0

Table c. Linear Network vs. Direct Optimization

Methods	Cropping	Distortion	Stability
Direct Optimization	0.91	0.93	0.40
Our Linear Network	0.88	0.97	0.60

equivalent to our network training can be defined as:

$$\min_{\mathbf{A}} L(\mathbf{m}, \mathbf{n}), \tag{1}$$

where L is the loss function defined in Eq. 3 in the main paper. Solving this problem directly is difficult and prohibitive in the video stabilization for the following reasons. First, the matrix $\mathbf{A}$ is dense and the problem is highly underdetermined. Second, the loss function we defined involves non-linear moving least squares warping; the problem cannot be solved using a simple linear system solver as in the bundled camera paths [4]. Finally, the problem has to be solved for each sliding window in the online video stabilization, making it impossible to achieve real-time performance. On the other hand, the linear neural network has two advantages compared to posing the problem as an optimization. First, the convolutional layers contain only small kernels; the concatenation of layers is equivalent to decomposing the dense matrix into a series of sparse matrices which is easier to solve through backpropagation and gradient descent. Second, the network implicitly provides regularization by training on a large dataset; using a pretrained network avoids the overfitting problem in the optimization and also enables computational real-time performance.

B.3. Direct optimization

Since our network is linear, an obvious question is whether we need a convolutional network at all. A way to pose the stabilization process as an optimization problem is to directly solve for the warp node displacement $\widehat{\mathbf{Q}}_t - \mathbf{Q}_t$

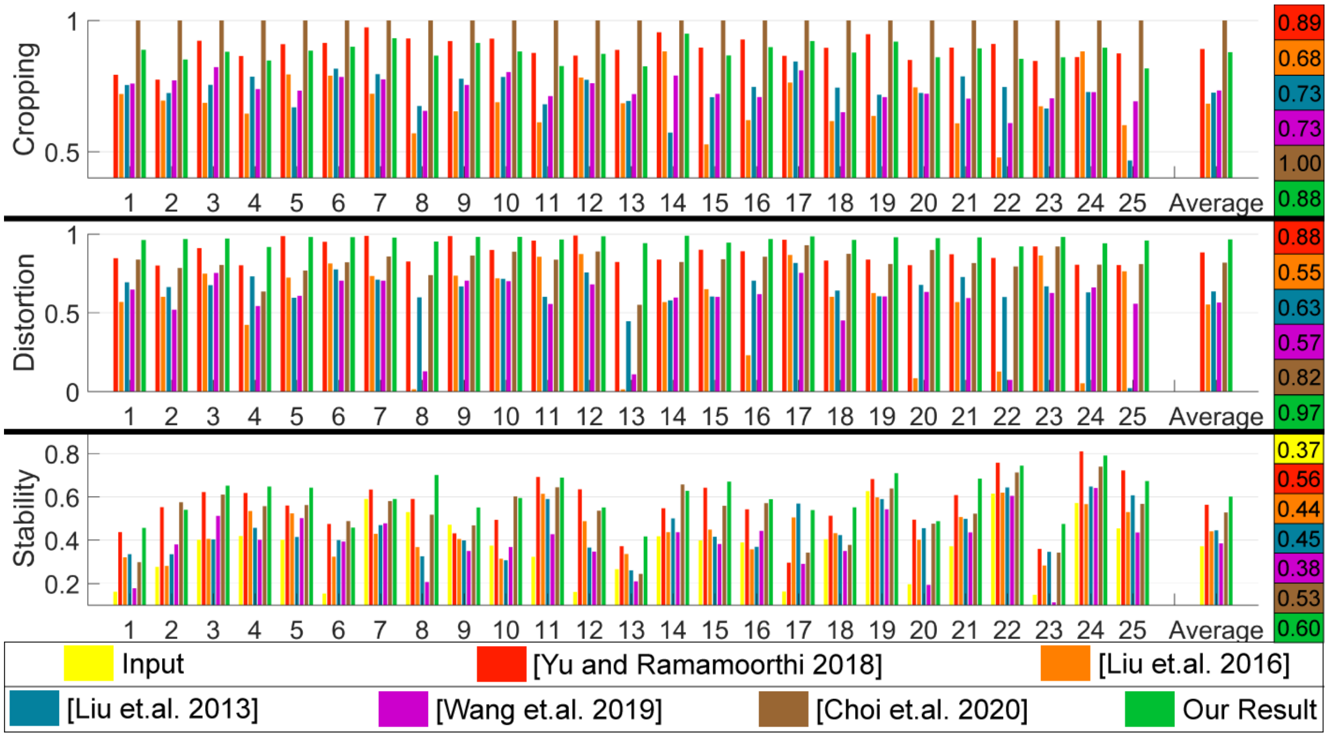

Figure b. *Complete quantitative comparison of bundled camera paths [4], selfie video stabilization [7], MeshFlow [3], deep online video stabilization [6], deep iterative frame interpolation [1] and our method. In these metrics, a larger value indicates a better result.*

Table d. Quantitative results from different network designs. In this table, C is the number of filters in the first layer of our network depicted in Fig. 5

C=32	Cropping	Distortion	Stability
No activation	0.85	0.95	0.56
Leaky ReLU	0.90	0.97	0.48
Tanh	0.87	0.97	0.50

C=64	Cropping	Distortion	Stability
No activation	0.86	0.96	0.57
Leaky ReLU	0.92	0.98	0.52
Tanh	0.85	0.96	0.52

C=128	Cropping	Distortion	Stability
No activation	0.88	0.97	0.60
Leaky ReLU	0.91	0.97	0.57
Tanh	0.89	0.96	0.52

to minimize the non-linear loss function L. Note that the objective function L is non-linear, so a simple least squares linear solver such as in bundled camera paths [4] cannot be used. We conduct an experiment in which we optimize our loss function Eq. 3 in the main paper directly over the feature points (warp nodes) instead of network weights. We optimize 1000 iterations using Adam optimizer [2] with

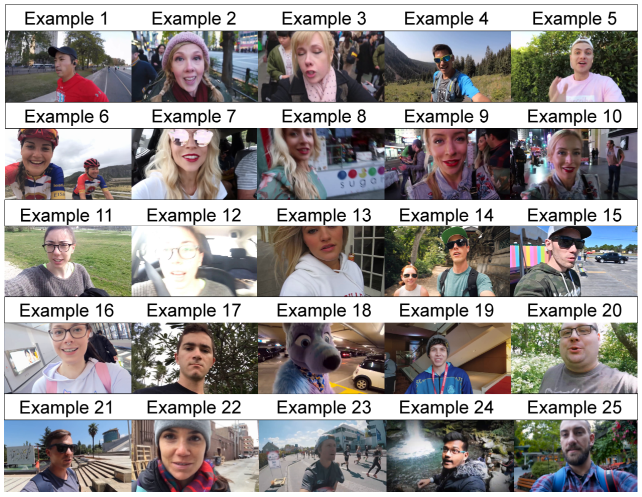

Figure c. *The 25 selfie video examples used for testing, referred to in Fig. b*

$lr = 10^{-1}$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for each 5-frame sliding window. Note that although this formulation is tractable comparing to Sec. B.2, the runtime of this optimization is prohibitive for pratical use since it requires an average of 20 seconds to stabilize each frame. We show the quantitative comparison of this optimization result with the result generated by our linear network in Table c. Although our network is linear, it performs significantly better than direct optimization. This is expected; since the input feature points are sparsely distributed and the distribution varies

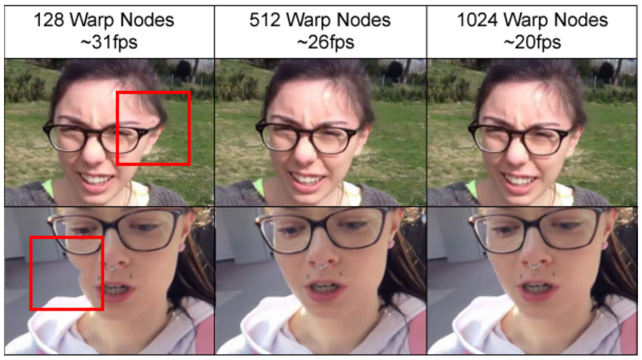

Figure d. *The visual comparison and stabilization speed comparison of different number of warp nodes in our method. The artifacts are marked by the red box.*

frame from frame, blindly overfitting to the feature points in each sliding window will result in temporal inconsistency. Our linear network provides implicit regularization for this process since it is trained over a variety of feature point distributions. Therefore, this comparison proves that using the linear network is necessary and can produce significantly better results than optimization.

B.4. Comparison with non-linear network

To justify our linear network design, we added different types of activation layers after each convolutional layer in our network and compare the result with our original network design. To allow negative values in the network feature vectors, we select leaky ReLU and Tanh in our experiments. Table d shows the averaged quantitative result over the examples in Fig. c using the networks with leaky ReLU (with negative slope 0.2), tanh and no activation layers (our original design). For the stability metric that is the most important, it can be observed that non-linear activation layers undermine the performance comparing to our original network design with the same base number of filters C. The reason for this performance degradation is that the non-linear layers break the linear input/output relationship requirement.

B.5. Number of filters

To show the effect of the number of filters used in each layer of the network, in Table d we include the quantitative results with different numbers of filters in the input layer, i.e., $C = 32, 64, 128$. In general, the larger number of filters in the network, the better the results. This conclusion also applies to the networks with non-linear activation layers, but the effect is more significant for the leaky ReLU activated network. For the even more non-linear network with tanh layers, the performance saturates quickly with a greater number of filters C. In this paper, we use $C = 128$ in all the experiments.

Table e. Ablation Study

Ablation	Cropping	Distortion	Stability
No Foreground Detection	0.89	0.95	0.52
Full Pipeline	0.88	0.97	0.60

Table f. Input Video Frame Size Comparison

Frame Sizes	Cropping	Distortion	Stability
HD (1280×720)	0.87	0.95	0.59
FHD (1920×1080)	0.87	0.96	0.58
832×448	0.88	0.97	0.60

C. Additional Results

In this section, we provide additional results to Sec. 6 in the main paper.

C.1. Complete Quantitative Result

The complete list of video index and sample frames are shown in Fig. c. In Fig. b, we provide complete quantitative comparison with bundled camera paths [4], selfie video stabilization [7], MeshFlow [3], deep online video stabilization [6], deep iterative frame interpolation [1]. Note that we also modify the optimization based method bundled camera paths [4] by including the same mask detection procedure used in our pipeline, but it doesn't improve their result.

C.2. The number of warp nodes (feature points)

The runtime performance of our method greatly depends on the number of warp nodes. Note that we use the feature points as the warp nodes, therefore the number of warp nodes is equivalent to the number of feature points. In the motion detection stage, tracking more feature points requires more processing time, leading to slower stabilization speed. However, if the warp nodes are too sparse in the frame, the possibility of local distortion increases. We provide the average per-frame stabilization time using 128, 512 and 1024 warp nodes and the corresponding warped frames in Fig. d. In Fig. d, using 128 warp nodes results in distortion near the foreground/background bundaries. This is because in the MLS warping, the warp nodes are implicitly constrained by each other. Fewer constraints reduce the robustness of the warping. An isolated warp node, if tracked mistakenly, introduces local distortion. In our experiment, we select 512 warp nodes since it is a good balance between computational speed and warp quality.

C.3. Ablation Study

We performed an ablation study by removing the foreground mask detection stage in our pipeline. This experiment means that we are essentially using all the feature points from both foreground and background, even if the foreground feature tracking is not reliable. Table e shows the comparison with the full pipeline. The stability score is significantly smaller than our full pipeline that separates the foreground and background. However, note that even without foreground mask detection, we still outperform comparison optimization based methods [3, 4]. This also indicates that using the network is necessary for the video stabilization task.

C.4. Video Frame Size

The previously discussed results are tested with videos with frame size 832×448. Since our network only takes feature point/head vertices as the input, it is scalable with different frame sizes. We tested our network with standard video resolutions (i.e., HD 1280×720 and Full HD 1920×1080) and compare the quantitative results with the 832×448 input, shown in Table f. In these experiments, we resize the frame to 832×448 for faster feature detection and foreground/face detection. In the warping stage, we rescale the feature points and the output of our network. Our network is able to handle higher resolution videos, and the result quality is similar to previously discussed results with frame size 832×448.

References

[1] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM Trans. Graph.*, 39(1), Jan. 2020. 3, 4

[2] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014. 3

[3] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *European Conference on Computer Vision (ECCV)*, 2016. 3, 4, 5

[4] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4), July 2013. 2, 3, 4, 5

[5] Fuhao Shi, Sung-Fang Tsai, Youyou Wang, and Chia-Kai Liang. Steadiface: Real-time face-centric stabilization on mobile phones. In *IEEE International Conference on Image Processing (ICIP)*, 2019.

[6] M. Wang, G. Yang, J. Lin, S. Zhang, A. Shamir, S. Lu, and S. Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 28(5):2283–2292, 2019. 3, 4

[7] Jiyang Yu and Ravi Ramamoorthi. Selfie video stabilization. In *European Conference on Computer Vision (ECCV)*, 2018. 3, 4