Sim2RealVS: A New Benchmark for Video Stabilization with a Strong Baseline

Qi Rao[1], Xin Yu[2], Shant Navasardyan[3], Humphrey Shi[3]
[1]ReLER Lab, AAII, University of Technology Sydney, [2]University of Technology Sydney, [3]Picsart AI Research (PAIR)

Abstract

Video stabilization is highly desirable when videos undergo severe jittering artifacts. The difficulty of obtaining sufficient training data obstructs the development of video stabilization. In this work, we address this issue by presenting a Sim2RealVS benchmark with more than 1,300 pairs of shaky and stable videos. Our benchmark is curated by an in-game simulator with diverse scenes and various jittering effects. Moreover, we propose a simple yet strong baseline approach, named Motion-Trajectory Smoothing Network (MTSNet), by fully exploiting our Sim2RealVS data. Our MTSNet consists of three main steps: motion estimation, global trajectory smoothing and frame warping. In motion estimation, we design a Motion Correction and Completion (MCC) module to rectify the optical flow with low confidence, such as in textureless regions, thus providing more consistent motion estimation for next steps. Benefiting from our synthetic data, we can explicitly learn a Trajectory Smoothing Transformer (TST) with ground-truth supervision to smooth global trajectories. In training TST, we propose two fully-supervised losses, i.e.*, a motion magnitude similarity loss and a motion tendency similarity loss. After training, our TST is able to produce smooth motion trajectories for the shaky input videos. Extensive qualitative and quantitative results demonstrate that our MTSNet achieves superior performance on both synthetic and real-world data.*

1. Introduction

Videos captured with unintentional motions (*e.g.*, severe shakes, quick rotations) often lead to unstable visual contents. Video stabilization is highly desirable to eliminate the shaky effects and produce visually stabilized videos. Notwithstanding a great success of the traditional [15, 8, 16, 6, 20, 19, 21, 18] and deep-learning based [33, 37, 35, 31, 29] stabilization techniques, a large variety of scenes or jittering effects are still difficult to be covered by existing benchmarks, and state-of-the-art video stabilization methods might fail in those cases.

Collecting large-scale stable and unstable video pairs synchronously is challenging in the real world. In fact, we cannot place two physical cameras at the same position to collect synchronized views, and the camera position deviations will cause undesired parallaxes. Even though the works [36, 29] have attempted to capture video pairs using special hardware, their collected datasets still have the aforementioned issues. Besides, the amount of their collected data is small for deep learning based methods.

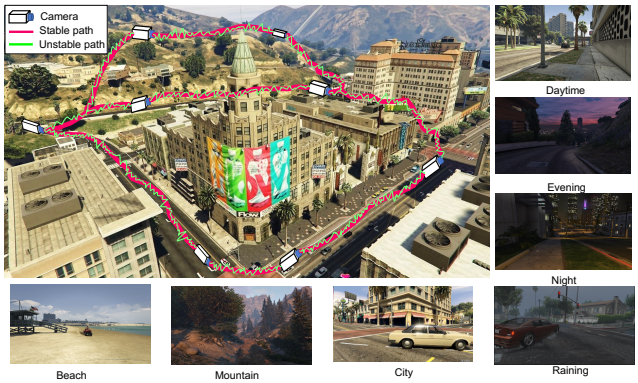

Figure 1. We present a new large-scale video stabilization benchmark dubbed Sim2RealVS. An in-game simulator is utilized to generate synchronized stable and unstable views by two cameras at the same position without parallaxes. The red trajectory represents the stable camera path, and the green trajectory represents the synchronized unstable camera path. A diverse range of scenes and environments is demonstrated.

To overcome the aforementioned shortcomings, we present a new large-scale benchmark dubbed Sim2RealVS. involving diverse scenes and jittering effects via an in-game simulator Grand Theft AutoV (GTA5), as shown in Figure 1. Particularly, in GTA5 we can put two simulated cameras at the same position without parallaxes. Our Sim2RealVS benchmark has several advantages: (i) it covers diverse scenes with various weather conditions and jittering effects. In some scenarios, such as under extreme weather or underwater, it is even dangerous for humans to capture data. (ii) it has the largest scale of training data and provides view-synchronized ground-truth stable references.

In order to verify the potential benefits brought by our Sim2RealVS benchmark, we propose a simple yet strong baseline method, namely Motion-Trajectory Smoothing

Table 1. Summary of existing benchmarks for video stabilization.

Benchmark	Year	Scale. (#)	GT Availability	Parallax between Unstable/ GT views	Jittering effects
Liu et al. [15]	2009	32	✗	-	real
Grundmann et al. [8]	2011	-	✗	-	real
Liu et al. [16]	2011	109	✗	-	real
Goldstein and Fattal [6]	2012	42	✗	-	real
Liu et al. [20]	2013	174	✗	-	real
Koh et al. [12]	2015	162	✗	-	real
Yu and Ramamoorthi [33]	2018	33	✗	-	real
Zhang et al. [36]	2018	45	✓	parallax caused by the camera position bias	real
Ito and Izquierdo [11]	2019	421	✓	no parallaxes	generated
Wang et al. [29]	2019	60	✓	parallax caused by the camera position bias	real
Ours	2021	**1327**	✓	no parallaxes	simulated

Network (MTSNet). MTSNet aims to estimate accurate motions at sparse locations and then remove the unintentional motions from the trajectories. Specifically, MTSNet consists of a Motion Correction and Completion (MCC) step, a Global Trajectory Smoothing (GTS) step, and a frame warping step.

Firstly, our proposed MCC samples motions at grid vertices from dense optical flow in order to preserve the rigidity of stabilized frames. Considering optical flow at vertices might not be robust, especially in some textureless areas, we design a self-supervised motion correction network to correct and complete the motions of the sampled vertices by estimating an offset adjustment for each vertex motion. Secondly, the completed motions at vertices are temporally assembled into trajectories. Then, in GTS, we design a Trajectory Smoothing Transformer (TST) to stabilize the wobbly trajectories by regressing their stable ground truth and constraining motion similarity among neighboring frames. To be specific, we enforce the motion magnitudes and tendency of the smoothed trajectories to be similar to the ground-truth ones by our proposed motion magnitude similarity loss and motion tendency similarity loss. Finally, stabilized grid vertices are calculated by propagating the smoothed motion trajectories to the original vertices. Then, different affine transformations are applied to all the grid-cells independently to produce the stabilized frames.

Experimental results show that our MTSNet significantly benefits from the proposed Sim2RealVS benchmark in several aspects: (i) MTSNet trained on our Sim2RealVS benchmark performs better on the synthetic testing videos compared with the state-of-the-art methods, indicating our baseline is powerful. (ii) Existing methods also achieve performance gains after being trained on Sim2RealVS (we take DUT [31] as an example in our experiments). (iii) MTSNet trained on the combined data of Sim2RealVS and a real-world small-scale benchmark achieves superior performance compared with existing methods on real-world testing videos. This indicates that our Sim2RealVS benchmark significantly facilitates video stabilization for real-world applications.

Our main contributions are summarized as below:

- We present the first large-scale Sim2RealVS benchmark with various shaky scenes to facilitate video stabilization development and evaluation.
- We provide a simple yet effective baseline method, dubbed MTSNet to demonstrate the significance of our benchmark and improve the state-of-the-art video stabilization performance.

2. Related Work

Benchmarks and Metrics: Video stabilization benchmarks have undergone a long revolution. As summarized in Table 1, traditional methods [15, 8, 16, 6, 20, 12, 33] usually provide benchmarks with a limited scale and only for evaluation. Three objective metrics without stable references are proposed in [20] to quantitatively evaluate the stabilization performance. The first stabilization assessment metric with stable references is proposed along with an evaluation benchmark in [36]. With the development of deep learning, the need of training data has spawned the first training benchmark [29] for video stabilization. However, it only contains 60 stable and unstable video pairs, which are too few to satisfy the learning needs. Besides, their video capturing method introduces undesirable parallaxes. On the basis of the previous benchmark [29], the work [11] presents a benchmark with stable references and evaluation framework. However, their shaky effects are not caused by camera motion but a result of applying random transformations on the original videos. Thus, the shaky videos cannot mimic real-world jitterings.

Traditional Video Stabilization: Traditional video stabilization methods are typically designed to estimate the warping field from unstable frames and then generate stable frames. Liu et al. [15] render the stabilized frames by using content-preserving warps. Grundmann et al. [8] propose an L1 norm optimization to remove undesired camera motions. Liu et al. [20] propose to use a bundle of camera paths to

make motion estimation more robust. Later, Liu et al. [21] firstly propose to smooth trajectories based on pixel profiles rather than feature points, and then they make a series of improvements, including more sparse MeshFlow [18], a faster version CodingFlow [17] and a photometric alignment [14]. These methods highly rely on the long stable tracks of local features and might fail when the local features are missing due to occlusion or local blurs.

Liu et al. [16] smooth the tracked features by enforcing subspace constraints. Later, in [6], the epipolar geometry is exploited to optimize stabilization via enhancing the long feature tracks. Liu et al. [19] utilize a depth camera to solve the stabilization with extra depth information. However, the 3D depth information may not always be attainable.

Deep Learning Based Video Stabilization: Benefiting from deep learning techniques [24, 9, 3, 27, 5, 4, 23], video stabilization performance significantly improves. Xu et al. [30] propose to stabilize videos by generative adversarial networks [7]. Wang et al. [29] present a benchmark with ground-truth stable references and design a CNN model to learn to regress the vertex positions from stable supervision. Yu and Ramamoorthi [34] leverage dense optical flow to solve the stabilization problem. Zhao and Ling [37] then propose to generate stable frames by learning pixel-wise warping maps in a multi-stage manner. Yu and Ramamoorthi [35] attempt to learn the stabilization directly using optical flow [10]. Later, Xu et al. [31] first obtain the motions at sparse keypoints by applying an optical flow network PWCNet [25] and a CNN-based keypoint detector RFNet [23], then propagate sparse motions to dense vertex grids, and smooth trajectories using a learned kernel. Liu et al. [22] propose a DNN-based fusion approach to stabilize videos by aggregating neighboring frames. Xu et al. [32] present an out-of-boundary view synthesis method to help warp-based approaches achieve full frame stabilization. Lee et al. [13] propose to solve stabilization by leveraging 3D cues.

In general, deep learning based video stabilization research requires a large-scale benchmark for training and testing. Moreover, the generalization ability in video stabilization has not been investigated. Transferring the stabilization knowledge from synthetic data to real-world scenarios also needs to be studied. All these issues motivate us to design a large-scale benchmark for training and evaluating video stabilization networks.

3. Proposed Sim2RealVS Benchmark

In the video stabilization task, it is very difficult to collect large-scale stable and unstable video pairs simultaneously due to camera parallaxes. To tackle this issue, we opt to utilize easy-collected synthetic data from the video game GTA5. By employing the in-game simulator, we present a Sim2RealVS benchmark including various scenes and shaky effects. The visualization of some sample scenes is shown in Figure 1. More videos are demonstrated in supplementary materials.

3.1. Choices on Scenes and Environments

Our Sim2RealVS benchmark has diverse scenes and environmental conditions. A wide range of scenes covers city streets, mountain roads, parks, beach streets, and etc. Environmental conditions are diverse in time (*i.e*., morning, noon, afternoon, evening and midnight), weather (*i.e*. clear, rainy, cloudy, snowy and thundering) and the density of the pedestrians and cars.

In particular, we provide some extreme condition scenarios where the collection might be dangerous for humans in the real world. These extreme scenarios include diving, speedboat, parachute, roller coaster, and aerial views. However, these camera trajectories obtained from the simulator already have shaky effects in order to follow physics. Thus, we only utilize these for evaluation.

3.2. Collecting Stable and Unstable Video Pairs

Collecting synchronized views with two cameras at the same time will produce undesired parallaxes because two physical cameras cannot be placed at exactly the same position in the real world. In contrast, it is easy to reduce parallaxes if the two cameras are virtual and placed without position deviations. A convenient tool G2D [2] has been proposed for capturing scenes from GTA5 using a virtual camera with six degrees of freedom (6DoF).

Based on G2D, we further develop a simulator that enables two cameras to capture stable and unstable video pairs synchronously. Specifically, we first set up a path in the simulator and then let the protagonist walk according to the path. We record the 6DoF camera poses at all the timestamps and save them as stable trajectories. Various shaky effects are generated by applying different degrees of random perturbations to the stable trajectories, achieving diverse shaking effects. Here, the perturbations are synthesized by random noise and low-pass filtering, where random noise (*i.e*., jittering magnitudes and frequencies) can cover the range of real-world jittering, and low-pass filtering is employed to make perturbations realistic.

Our collecting approach effectively eliminates the impact of parallax. When scene objects are close to cameras, where the camera baselines cannot be neglected, such as indoor scenes or camera mounted on robotics, parallaxes will lead to inaccurate motion transfer from stable videos to shaky counterparts, and a method may produce understabilized results. In contrast, our Sim2RealVS benchmark does not have this limitation as the parallax is zero during the entire collecting procedure.

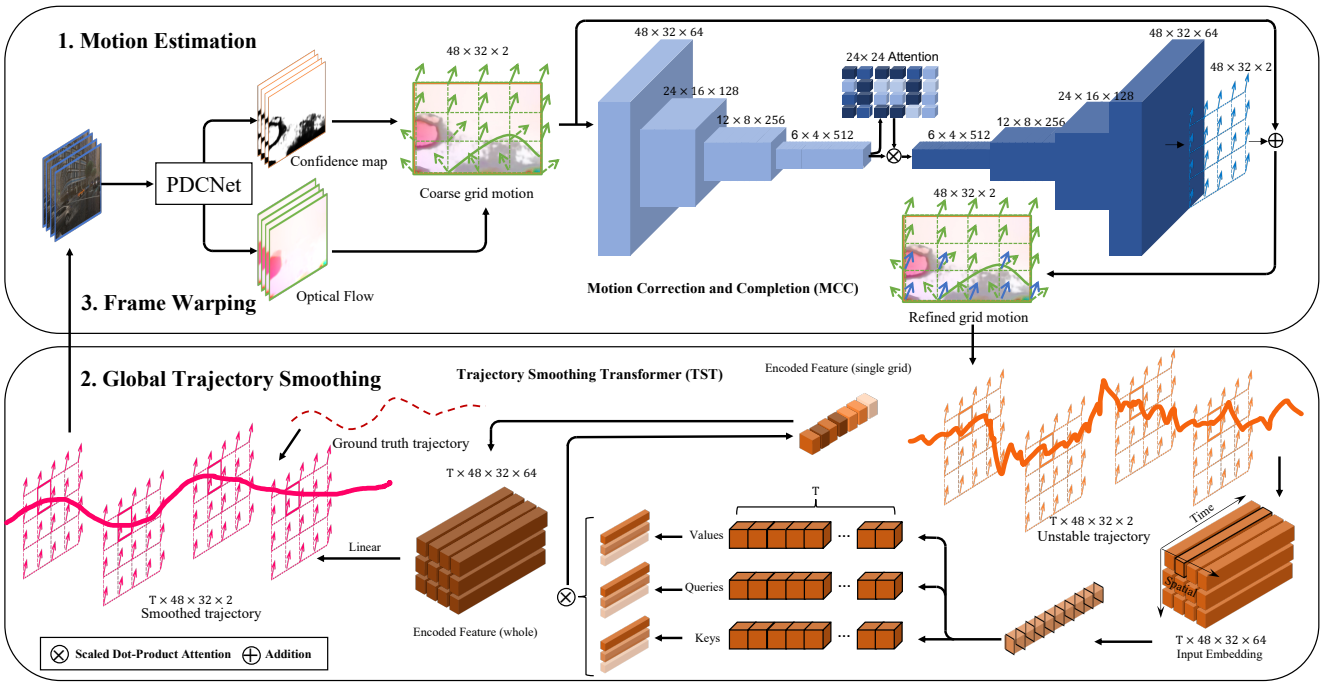

Figure 2. The pipeline of our MTSNet contains three main steps: (i) Motion Estimation. (ii) Global Trajectory Smoothing. (iii) Frame warping. In (i) motion estimation, we utilize the optical flow with a confidence map from PDCNet [27] to estimate a coarse grid motion. Next, we devise a Motion Correction and Completion (MCC) module to estimate an offset adjustment, obtaining the refined grid motion. MCC consists of a stack of down-sampling blocks, a spatial attention block, and a stack of up-sampling blocks. In (ii) global trajectory smoothing, we design a Trajectory Smoothing Transformer (TST) to smooth the wobbly trajectories by regressing their stable ground-truth reference and constraining motion similarity among neighboring frames. In (iii) frame warping, grid vertex positions are re-projected according to the stabilized trajectories and then conduct homography to each grid-cell to generate stabilized frames.

4. Proposed Baseline: MTSNet

We introduce a Motion-Trajectory Smoothing Network (MTSNet) as a strong baseline, which significantly benefits from our Sim2RealVS benchmark. MTSNet involves three steps to stabilize videos: (i) estimating video motions, (ii) removing unintentional motions, and (iii) warping frames based on smoothed motions. Particularly, we present a Motion Correction and Completion (MCC) network for accurate motion estimation, and a Trajectory Smoothing Transformer (TST) for global trajectory smoothing. The pipeline of MTSNet is illustrated in Figure 2.

4.1. Local Motion Estimation

To preserve the rigidity of frame transformation and avoid distortions, existing methods [20, 18, 31, 29] adopt grid-based representation to control frame warping. Similarly, we divide each video frame into grid-cells of size (*i.e.* $M \times N$). Instead of propagating keypoint-based motions to grid vertices as in [20, 18, 31], we directly estimate motion for grid vertices. We can obtain motions from optical flow at grid vertices but it might be unreliable in some areas, such as textureless regions. Therefore, we present a Motion Correction and Completion (MCC) module to correct inconsistent motions and complete missing motions at grid vertices.

4.1.1 Motion Correction and Completion (MCC)

We utilize PDCNet [27] to estimate optical flow for each video frame. PDCNet is a state-of-the-art dense matching network that provides not only optical flow estimation but also a confidence map of the estimated flow. It is applicable even for large appearance and view-point changes, which are common in unstable videos.

We denote the unstable input video as a set of consecutive image frames $I_u = \{i_u^t\}$, $t \in \{1, 2, ..., T\}$, $i_u^t \in \mathbb{R}^{3\times H\times W}$, where T indicates the number of frames, and H and W are the height and width of frames, respectively. The optical flow $o_u^t \in \mathbb{R}^{2\times H\times W}$[1] and its confidence map $c_u^t \in \mathbb{R}^{1\times H\times W}$ estimated by PDCNet (denoted by $\mathcal{P}$) are expressed as:

$$\{(o_u^t, c_u^t)\}_{t=1}^{T-1} = \mathcal{P}(I_u). \quad (1)$$

As aforementioned, we divide each frame by an $M \times N$

[1]The subscripts u, s represent unstable and stable videos, respectively. Otherwise, a variable can represent both cases without confusion.

grid and sample motions at the grid vertices:

$$m_u^t(i,j) = o_u^t\left(\frac{iH}{M}, \frac{jW}{N}\right),\ i \in \{1,...,M\},\ j \in \{1,...,N\}, \tag{2}$$

where the sampled grid motion field $m_u^t \in \mathbb{R}^{2\times M\times N}$ is then refined by MCC to produce locally consistent motion estimation. Specifically, MCC learns to regress a motion offset for each vertex:

$$\Delta r_u^t = \mathcal{N}_{\mathbf{MCC}}(m_u^t), \tag{3}$$

$$\hat{m}_u^t = m_u^t + \Delta r_u^t, \tag{4}$$

where $\mathcal{N}_{\mathbf{MCC}}$ denotes the MCC module, and $\hat{m}_u^t \in \mathbb{R}^{2\times M\times N}$ is the refined motion field. As shown in Figure 2, the network architecture of MCC consists of a stack of down-sampling blocks, a spatial attention block, and a stack of up-sampling blocks.

4.1.2 Training Loss

As indicated in Eqs. (3) and (4), our MCC takes m_u^t as input and outputs Δr_u^t for each vertex. Since the motions m_u^t are at grid vertices which might be noisy, we employ the average motion of a local patch as the ground-truth motions to adjust m_u^t. Specifically, we average local motions estimated by optical flow and the confidence map, and downsample them to the grid size. Then, the flow offset for grid vertices is written as:

$$\begin{gathered}\overline{o}^t, \overline{c}^t = \mathcal{B}(o^t, (M,N)), \mathcal{B}(c^t, (M,N)), \\ \Delta\overline{o}^t = \overline{o}^t - m^t,\end{gathered} \tag{5}$$

where $\mathcal{B}(\cdot, (M,N))$ represents an averaging operation, downsampling an image to the size (M,N) with bilinear interpolation, $\overline{o}^t, \Delta\overline{o}^t \in \mathbb{R}^{2\times M\times N}, \overline{c}^t \in \mathbb{R}^{1\times M\times N}$.The offset $\Delta\overline{o}^t$ is used as supervision for Eq. (3).

The averaged confidence map indicates in which local areas the estimated motions are more reliable. Therefore, we aim to use the reliable motions to supervise the offset flow learning in Eqs. (3) and (4). Thus, we employ a high threshold λ to remove unconfident areas from $\{\overline{o}^t\}$, and the motion correction loss is formulated as:

$$\begin{gathered}L_{\text{motion}} = \sum_{t=1}^{T-1}\sum_{v=1}^{MN}\sum\nolimits_{\overline{c}^t(v)>\lambda} \|\Delta\overline{o}^t(v)\ - \Delta r^t(v)\|_2^2, \\ (\overline{o}^t, \overline{c}^t) \in \{(\overline{o}_u^t, \overline{c}_u^t)\} \cup \{(\overline{o}_s^t, \overline{c}_s^t)\}, \\ \Delta r^t \in \{\Delta r_u^t\} \cup \{\Delta r_s^t\}.\end{gathered} \tag{6}$$

Although the motion correction loss is only applied to vertices with high confidence, MCC is agnostic to high confident vertices. It leverages neighboring motion information to correct or complete motions of vertices. In this way, accurate motions can be propagated to inaccurate ones. Note that MCC is trained in a self-supervised manner, and it can be applied to rectifying motions of stable videos.

We also apply a shape-preserving loss [29, 31, 15] to constrain the shapes of transformed grids:

$$\begin{gathered}L_{\text{shape}} = \sum_{t=1}^{T-1}\sum_{v=1}^{(M-1)(N-1)} \|p_{v+1} + A(p_{v+1} - p_v) - p_{v+2}\|_2^2, \\ A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},\end{gathered} \tag{7}$$

where p_v, p_{v+1}, p_{v+2} denote the triangle of deformed vertices clockwise. The matrix A conducts a 90-degree rotation. The final objective of MCC is defined as:

$$L_{\text{MCC}} = \beta L_{\text{motion}} + L_{\text{shape}}, \tag{8}$$

where β is a weight to balance the two terms.

4.2. Global Trajectory Smoothing

After obtaining the vertex motions, we aggregate the temporal trajectory for each grid vertex. As illustrated in Figure 2, an unstable trajectory usually contains unintentional motions while a stable one is much smoother. Here, we consider the trajectory smoothing task as a sequence to sequence learning problem whose input is an unstable trajectory, and output is a stabilized version, and thus propose a Trajectory Smoothing Transformer (TST).

4.2.1 Trajectory Smoothing Transformer (TST)

Denote the unstable trajectory at a specific vertex as $J_u^v = \{\hat{x}_u^{t,v}\}_{t=1}^T$, where $\hat{x}_u^{t,v}$ is the the offset of vertex v at timestamp t, calculated as the accumulated sum of motions: $\hat{x}_u^{t,v} = \sum_{i=1}^t \hat{m}_u^i(v)$. Similarly, a ground-truth stable trajectory is expressed by:

$$\begin{gathered}J_s^v = \{\hat{x}_s^{t,v}\}_{t=1}^T = \{\sum_{i=1}^t \hat{m}_s^i(v)\}_{t=1}^T, \\ \{J_s^v\}_{v=1}^{MN}, \{J_u^v\}_{v=1}^{MN} \in \mathbb{R}^{T\times MN\times 2}.\end{gathered} \tag{9}$$

We apply a linear embedding $E \in \mathbb{R}^{2\times E}$ to input trajectories $J_u = \{J_u^v\}_{v=1}^{MN}$ and obtain the trajectory features ξ. Then, ξ are fed into h parallel self-attention layers [28]:

$$\begin{gathered}\text{head}_i = \text{softmax}(\frac{\xi\xi^T}{\sqrt{d_k}})\xi, \\ \text{MultiHead}(\xi) = \text{Concat}(\text{head}_1, ..., \text{head}_h),\end{gathered} \tag{10}$$

where $\frac{1}{\sqrt{d_k}}$ is a scaling factor. Then another linear transformation is applied to the encoded features to obtain the stabilized trajectories, denoted as $J_{est} \in \mathbb{R}^{T\times MN\times 2}$.

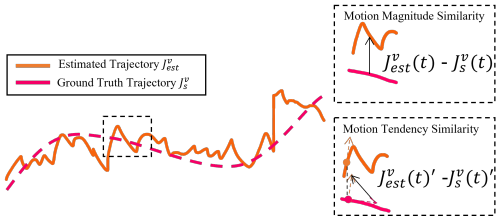

Figure 3. We design two losses to constrain the estimated trajectory by leveraging stable trajectory reference: (i) Motion Magnitude Similarity and (ii) Motion Tendency Similarity.

4.2.2 Training Loss

Similar to the works [31, 21, 20, 18], we constrain the smoothed trajectory by minimizing motion differences in a temporal neighborhood:

$$L_{\text{neighbor}} = \sum_{v=1}^{MN} \sum_{t=1}^{T} \sum_{q\in[t-3,t+3]} \|J^v_{est}(t) - J^v_{est}(q)\|_2^2. \tag{11}$$

However, this neighborhood smoothness loss can effectively remove large jittering artifacts, but is ineffective to remove medium and small shaky effects.

Benefiting from our Sim2RealVS benchmark, stable trajectories are available and we can use them to supervise TST learning, which cannot be conducted in prior arts. Therefore, we propose to constrain smoothed trajectories in two aspects with respect to the stable counterparts: motion magnitude similarity and motion tendency similarity.

The motion magnitude similarity loss is designed to minimize the geometric distance between the estimated trajectory and the stable reference trajectory for all grid vertices:

$$L_{\text{magnitude}} = \sum_{v=1}^{MN} \sum_{t=1}^{T} \|J^v_{est}(t) - J^v_s(t)\|_2^2. \tag{12}$$

We observe that unintentional motions usually occur when there is an undesired direction change in a trajectory, resulting in the motion tendency disagreement with the stable reference, as shown in Figure 3. To alleviate this issue, we design a motion tendency loss by enforcing the first derivatives of unstable and stable trajectories to be similar:

$$L_{\text{tendency}} = \sum_{v=1}^{MN} \sum_{t=1}^{T-1} \mathcal{L}[J^v_{est}(t+1) - J^v_{est}(t), J^v_s(t+1) - J^v_s(t)],$$
$$\mathcal{L}(x,y) = \begin{cases} 0.5(x-y)^2/\sigma, & |x-y| < \sigma, \\ |x-y| - 0.5\sigma, & |x-y| \geq \sigma, \end{cases} \tag{13}$$

where $\mathcal{L}$ denotes the smooth L1 loss. σ is set to 1.0 empirically. The final objective of TST is expressed as:

$$L_{\text{TST}} = \gamma_1 L_{\text{neighbor}} + \gamma_2 L_{\text{magnitude}} + \gamma_3 L_{\text{tendency}}, \tag{14}$$

where $\gamma_1, \gamma_2, \gamma_3$ balance the magnitudes of these three losses at the same level.

4.3. Frame Warping and Implementation Details

Frame warping. Once we obtain the smoothed trajectories for all grid vertices, frame warping is applied to the unstable video to generate a stabilized version. Specifically, for each unstable frame, we first project all grid vertices to the desired positions according to smoothed trajectories, and then conduct different homographies to all grid-cells independently to generate the stabilized frame.

Implementation details. For our synthetic benchmark, we use 1180 unstable and stable video pairs from Sim2RealVS for training, and 120 pairs of synthesized videos for testing. For the real-world benchmark DeepStab [29], we use 50 pairs of videos for training, and 10 video pairs for testing. The entire NUS dataset [20] with 174 unstable videos is used for testing.

During training, we randomly sample 80 consecutive frames in a video as input. Adam optimizer with the beta (0.5, 0.999) and an initial learning rate (0.0002) is applied to optimize MCC and TST, respectively. The spatial dimension numbers W, H, M, N are set to $768, 512, 48, 32$, respectively. We empirically set the loss weights $\beta, \gamma_1, \gamma_2, \gamma_3$ to 4, 10, 10, 20 respectively for satisfying performance. We train our MTSNet for 25 epochs. In the first 5 epochs, we only optimize the MCC network, and then we optimize both MCC and TST for the rest 20 epochs. The whole training process takes about 24 hours on a single NVIDIA Tesla V100 GPU.

5. Experiments

5.1. Datasets and Metrics

Apart from our proposed Sim2RealVS benchmark, existing real-world benchmarks, *i.e.*, NUS dataset [20] and the DeepStab dataset [29], are utilized in our experiments. The NUS dataset contains 174 unstable videos without stable references. The DeepStab dataset includes 60 unstable and stable video pairs.

Following [22, 31, 29, 20, 22, 32], we employ four metrics for evaluation: **(i) Cropping ratio** evaluates the remaining area ratio after stabilization. **(ii) Distortion value** measures the anisotropic scaling of the homography between unstable and stabilized video pairs. **(iii) Stability score** measures the stability and smoothness of the stabilized results. **(iv) Accumulated optical flow** accumulates the optical flow (*i.e.* RAFT [26]) over the entire stabilized video.

5.2. Experimental Details

We evaluate our MTSNet and Sim2RealVS benchmark in three steps: (i) We directly evaluate existing state-of-the-art methods, commercial software and our MTSNet on Sim2RealVS testing set. (ii) We fine-tune the pre-trained model of DUT [31], one of the representative deep learning

Table 2. Quantitative evaluations on our Sim2RealVS benchmark and existing real-world benchmarks. **S** denotes training on the Sim2RealVS benchmark and **D** denotes training on the DeepStab benchmark.The **bold** number indicates the best and the underlined number indicates the second best.

Model	NUS benchmark [20]				DeepStab benchmark [29]				Sim2RealVS benchmark			
	C↑	D↑	S↑	A↓	C↑	D↑	S↑	A↓	C↑	D↑	S↑	A↓
Bundle [20]	0.84	0.93	0.84	0.78	0.76	0.91	0.84	**0.56**	-	-	-	-
L1Stabilizer [8]	0.74	0.92	0.82	0.88	0.74	0.91	0.82	0.70	-	-	-	-
DIFRINT [1]	**1.00**	**0.97**	0.82	0.87	**1.00**	**0.96**	0.83	0.78	**1.00**	0.82	0.84	0.88
StabNet [29]	0.68	0.82	0.81	1.02	0.64	0.86	0.83	0.80	0.84	0.74	0.74	1.04
DUT [31]	0.70	0.95	0.83	0.87	0.73	0.92	0.83	0.77	0.87	0.77	0.80	0.91
Yu and Ramamoorthi [35]	0.86	0.91	0.83	0.88	0.83	0.92	0.83	0.72	0.90	0.85	0.81	0.87
FuSta [22]	**1.00**	0.96	0.84	**0.77**	**1.00**	0.95	0.83	0.64	**1.00**	0.88	0.84	0.84
DUT+OVS [32]	**1.00**	0.94	0.85	-	-	-	-	-	-	-	-	-
Adobe Premiere 2020 Stabilizer	0.74	0.83	**0.86**	0.84	0.75	0.87	0.83	0.78	0.89	0.71	0.82	0.81
DUT [31] (**S+D**)	0.72	0.94	0.85	0.86	0.73	0.93	0.83	0.74	0.88	0.81	0.85	0.86
MTSNet (**D**)	0.81	0.92	0.83	0.84	0.75	0.92	0.83	0.76	0.88	0.80	0.79	0.87
MTSNet (**S**)	0.89	0.95	0.84	0.82	0.79	0.92	0.83	0.77	0.90	0.92	0.87	0.83
MTSNet (**S + D**)	0.89	**0.97**	0.85	0.78	0.78	0.95	**0.85**	0.60	0.90	**0.93**	**0.89**	**0.79**

Figure 4. Visual comparisons between MTSNet and existing state-of-the-art methods on Sim2RealVS benchmark, NUS benchmark [20] and DeepStab [29] benchmark. We highly encourage readers to watch the videos in supplementary materials.

Table 3. Ablation study on the impacts of MCC module on Sim2RealVS benchmark.

Motion Estimation	C↑	D↑	S↑	A↓
simple sample	0.91	0.72	0.74	0.97
bilinear interpolation	0.91	0.74	0.77	0.95
adaptive average-pooling	0.90	0.77	0.78	0.93
MCC (w/o SA)	0.90	0.85	**0.86**	**0.83**
MCC (w/ SA)	**0.93**	**0.87**	**0.86**	**0.83**

based methods, on Sim2RealVS to demonstrate the contribution of Sim2RealVS benchmark. (iii) We train MTSNet on both Sim2RealVS and DeepStab benchmarks, and then evaluate it on the real-world testing videos.

5.3. Quantitative and Qualitative Results

We compare our MTSNet with existing state-of-the-art methods. The quantitative results in Table 2 indicate our MTSNet achieves state-of-the-art performance on both syn-

thetic and real-world data. Note that Fusta [22] achieves very good performance, but its stabilization speed is slow, *i.e*., around 8 seconds per frame. In contrast, our method runs 80 ms per frame. The performance of using either real or synthetic data is inferior to that of our final model, implying the importance of our Sim2RealVS benchmark. For qualitative analysis, we demonstrate visual comparisons to state-of-the-art methods in Figure 4. It can be observed that existing methods suffer from distortions (as highlighted by the orange boxes) or severe shakes at distant objects (DUT). In contrast, our MTSNet can generate more stable results with fewer distortions on both synthetic and real-world data. More visual comparisons on both synthetic and real-world data are provided in supplementary materials.

5.4. Ablation Study

We dissect our proposed components (i.e., MCC and GTS) in MTSNet to demonstrate their effectiveness. All ablation studies are conducted on Sim2RealVS benchmark.

Impacts of MCC: As the first step in the MTSNet, MCC aims to provide accurate vertex motions by leveraging optical flow. We analyze the effectiveness of MCC by comparing with several learning-free approaches that can also provide vertex motion estimation based on optical flow. We replace the second smoothing step by a Jacob Solver, which is learning-free and widely used in video stabilization methods [20, 18, 31], as the trajectory smoothing module, for fair comparison. First, we provide the motions at grid vertices by directly sampling optical flow without MCC (as illustrated in the first row of Table 3). Next, we further explore the options of estimating average local motions with different averaging operations, *i.e*. bilinear interpolation and average-pooling. Specifically, for the bilinear interpolation approach, we downsample the optical flow to the resolution of $M \times N$, as the estimated vertex motions. For the average-pooling approach, we apply an adaptive average pooling kernel to calculate the mean motion of a local patch with the stride of $\alpha = \frac{H}{M}$. The results are illustrated in the second and third rows of Table 3.

For the internal design of MCC, we study the impacts of the self-attention of MCC. The comparison between the last two rows in Table 3 indicates that MCC with self-attention achieves better performance. This indicates that MCC is able to explore the global motions of frames to correct inconsistent ones. As shown in the last row of Table 3, our approach surpasses all the other configurations, verifying that the MCC corrects and completes vertex motions accurately.

Impacts of the network architecture and losses in TST: In GTS, we propose a TST with three different losses to remove unintentional motions. We first compare the TST with a traditional Jacob Solver baseline (as shown in the first row of Table 4) to demonstrate the superiority of a learnable trajectory smoother. Here, the motion estimation module is fixed, where the same MCC pretrained model is employed for fair comparison. Then, we study the impacts of different losses as shown in the 3rd-8th rows of Table 4. For the internal design of TST, we study the impacts of the different network architectures, *i.e*. MLP with four hidden layers (the second row of Table 4), as a comparison to Transformer. As illustrated in the last row of Table 4, TST trained with the three losses surpasses the others, indicating the effectiveness of TST and our proposed trajectory smoothing losses. In our experiments, we notice that without explicit supervision, TST fails to converge. This also implies the necessity of our proposed Sim2RealVS benchmark.

Table 4. Ablation study on the impacts of GTS module on Sim2RealVS benchmark. **N, M, T** denote the neighborhood smoothness loss, the motion magnitude similarity loss and the motion tendency similarity loss, respectively. * means the training does not converge.

Trajectory Smoother	C↑	D↑	S↑	A↓
Jacob Solver	0.93	0.87	0.86	**0.83**
MLP (**N+M+T**)	0.92	0.85	0.86	0.87
TST (**N**)*	-	-	-	-
TST (**M**)*	-	-	-	-
TST (**T**)	0.93	0.81	0.80	0.87
TST (**N+M**)	0.91	0.84	0.84	0.90
TST (**N+T**)	0.91	0.87	**0.89**	**0.83**
TST (**N+M+T**)	**0.94**	**0.92**	0.87	**0.83**

6. Conclusion

In this paper, we presented the first large-scale synthetic video stabilization benchmark, dubbed Sim2RealVS, as well as a strong baseline, namely Motion-Trajectory Smoothing Network (MTSNet). Benefiting from our synthetic dataset, MTSNet can be easily trained in a fully-supervised fashion and achieves state-of-the-art performance. Moreover, we show that our benchmark also facilitates the development of other methods, further improving their performance and mitigating the data hungry issue of deep learning based methods. We believe that our Sim2RealVS benchmark will profoundly contribute to the research field and propel the development of video stabilization.

Acknowledgements. This research is funded in part by ARC-Discovery grant (DP220100800 to XY) and ARC-DECRA grant (DE230100477 to XY). We thank all anonymous reviewers and ACs for their constructive suggestions.

References

[1] 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.

[2] Anh-Dzung Doan, Abdul Mohsi Jawaid, Thanh-Toan Do, and Tat-Jun Chin. G2d: from gta to data. *arXiv preprint arXiv:1806.07381*, 2018.

[3] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[4] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017.

[6] Amit Goldstein and Raanan Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 32(5), 2012.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[8] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[11] Maria Silvia Ito and Ebroul Izquierdo. A dataset and evaluation framework for deep learning based video stabilization systems. In *2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, 2019.

[12] Yeong Jun Koh, Chulwoo Lee, and Chang-Su Kim. Video stabilization based on feature trajectory augmentation and selection and robust mesh grid warping. *IEEE Transactions on Image Processing*, 24(12):5260–5273, 2015.

[13] Yao-Chih Lee, Kuan-Wei Tseng, Yu-Ta Chen, Chien-Cheng Chen, Chu-Song Chen, and Yi-Ping Hung. 3d video stabilization with depth estimation by cnn-based optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10621–10630, June 2021.

[14] Kaimo Lin, Nianjuan Jiang, Shuaicheng Liu, Loong-Fah Cheong, Minh Do, and Jiangbo Lu. Direct photometric alignment by mesh deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2405–2413, 2017.

[15] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transcations on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 28(3), 2009.

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

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

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

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

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

[21] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4209–4216, 2014.

[22] Yu-Lun Liu, Wei-Sheng Lai, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Hybrid neural fusion for full-frame video stabilization. *arXiv preprint arXiv:2102.06205*, 2021.

[23] Xuelun Shen, Cheng Wang, Xin Li, Zenglei Yu, Jonathan Li, Chenglu Wen, Ming Cheng, and Zijian He. Rf-net: An end-to-end image matching net-

work based on receptive field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8132–8140, 2019.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[26] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[27] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning accurate dense correspondences and when to trust them. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5714–5724, 2021.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[29] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 28(5):2283–2292, 2019.

[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. Wiley Online Library, 2018.

[31] Yufei Xu, Jing Zhang, Stephen J Maybank, and Dacheng Tao. Dut: Learning video stabilization by simply watching unstable videos. *arXiv preprint arXiv:2011.14574*, 2020.

[32] Yufei Xu, Jing Zhang, and Dacheng Tao. Out-of-boundary view synthesis towards full-frame video stabilization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4842–4851, 2021.

[33] Jiyang Yu and Ravi Ramamoorthi. Selfie video stabilization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 551–566, 2018.

[34] Jiyang Yu and Ravi Ramamoorthi. Robust video stabilization by optimization in cnn weight space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3800–3808, 2019.

[35] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8159–8167, 2020.

[36] Lei Zhang, Qing-Zhuo Zheng, Hong-Kang Liu, and Hua Huang. Full-reference stability assessment of digital video stabilization based on riemannian metric. *IEEE Transactions on Image Processing*, 27(12):6051–6063, 2018.

[37] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE Transactions on Image Processing*, 29:3582–3595, 2020.