Leveraging Synthetic Data to Learn Video Stabilization Under Adverse Conditions (Supplementary Materials)

Abdulrahman Kerim[1,2] Washington L. S. Ramos[3] Leandro Soriano Marcolino[1]
Erickson R. Nascimento[3,4] Richard Jiang[1]
[1] Lancaster University, UK [2] University for the Creative Arts, UK [3] UFMG, Brazil [4] Microsoft
{a.kerim,l.marcolino,r.jiang2}@lancaster.ac.uk {washington.ramos,erickson}@dcc.ufmg.br

This document provides complementary results and further details to enrich the analysis and discussion provided in the main manuscript. Further statistics and details regarding the collected real videos and the generated synthetic sequences are given too. Additionally, qualitative video results are demonstrated on the paper's GitHub repository at https://github.com/A-Kerim/SyntheticData4VideoStabilization_WACV_2024.

1. Our Framework for Generating Synthetic Data for Computer Vision Tasks

Our work deploys synthetic data to train a synthetic-aware video stabilization algorithm. For that aim, we developed our novel simulator using the Unity game engine to generate our synthetic training datasets VSAC65Synth and VSNC35Synth. We employ the Procedural Content Generation (PCG) concept to create a full 3D virtual world at run-time while the system's extensibility is attained by taking advantage of the modular approach followed as we built the system from scratch. Although our simulator can provide clean, unbiased, and large-scale training and testing data for various computer vision tasks, we focus on the video stabilization task. A simplified flowchart describing the scene creation process is shown in Figure 1.

Static Elements Starting from the given parameters, our simulator initially creates the static part of the 3D virtual world. In this part, first the street length and the number of crosses are set at random. Following this, the buildings are created where buildings' locations, types, and frequency are set at random. After that, the other scene elements like benches, trash containers and bags, trees, and other elements are created. To further improve the realism and diversity of the generated scenes, we introduce a new variable called *Anomaly Rate*; higher values will cause more artifacts to appear in the scene such as more street lights being off at night, more being on at daytime, and some trash bags being on roads.

Dynamic Elements Once the static part of the scene is completed, the dynamic part of the scene is initiated. Initially, the characters generator retrieves the locations of buildings and benches, and instantiates characters based on the required characters density. The Microsoft Rocketbox Avatar Library [3] is used to define the character avatar, and the animations are selected based on character pose (standing or sitting). Character animations were adopted from Mixamo. In a similar way, the cars are created. However, number of cars and models are selected at random. Additionally, car shader attributes: Smoothness, Metallic, and BaseColour are all randomized at run-time to give different visual appearance even to the same car model. After that, the plates of the cars are selected at random from a large set collected manually from the web. The main processes are summarized in Figure 1. In parallel to that, the first-person videos are recorded using Cinemachine camera behaviour from Unity, attached to an AI navigation agent. We use Cinemachine since it gives unlimited sets of behaviours that enrich the diversity of the generated synthetic data in terms of the camera view angle and transition.

2. More Details on VSAC105Real Dataset

The available video stabilization benchmarks such as DeepStab [9], Stabfr [12], Selfie Video [10], and LiuSigg2013 [5] exclusively contain videos under normal weather condition and at a sufficient illumination. To assess the performance of the state-of-the-art video stabilization methods under foggy, rainy, snowy, and night-time conditions, we created the VSAC105Real dataset.

Our dataset is composed of videos collected from YouTube using search queries like "Fog", "Rain", "Snow", "Night", "Adverse", and "Severe". We manually inspected all the videos and selected the ones with shak-

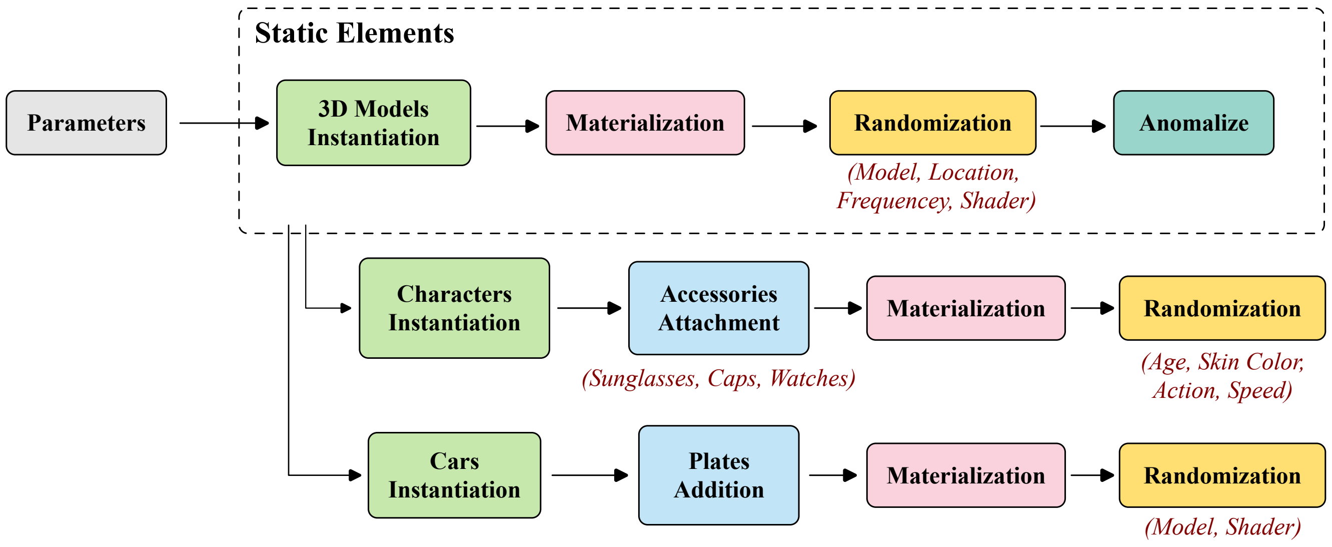

Figure 1. Flowchart describing the scene creation in our novel simulator.

Table 1. **Dataset statistics.** Comparison among the available video stabilization datasets and VSAC105Real dataset.

Dataset Name	#Videos	Average #Frames	Total #Frames
DeepStab [9]	61	714	43,585
Stabfr [12]	45	471	21,200
Selfie Video [10]	33	251	8,308
LiuSigg2013 [5]	144	578	83,257
VSAC105Real	105	737	77,477

ing camera movement. Then, we cut the videos to ensure continuous temporal criteria and the query attribute. VSAC105Real dataset comprises 105 videos spanning normal, rainy, foggy, snowy, and night-time attributes. The first four attributes were selected to study the effect of severe weather conditions on video stabilization quality. Similarly, the night-time was chosen to understand the effect of low illumination on video stabilization. Table 1 shows a comparison among different video stabilization datasets and VSAC105Real dataset. The VSAC105Real dataset has the advantage in terms of the average number of frames. Moreover, it includes a diverse set of challenging attributes where videos are evenly distributed across the classes, *i.e.*, 21 videos per class.

3. Results

3.1. More on the Evaluation Metrics

To evaluate our approach, we use three metrics commonly used to evaluate video stabilization algorithms [2, 6, 9, 11]: *i) Stability Score*. It assesses the smoothness of the stabilized video; the higher the value the better. It is computed as the average between Stability Average Translation and Stability Average Rotation Scores. To compute this score, we estimate the homography matrix between v_i and v_{i+1} to obtain the translation and rotation arrays. Following this, we calculate their Fast Fourier Transform (FFT). Finally, we obtain the score by calculating the ratio between the 2nd through 6th frequency components and all frequency components. Note that the 0th frequency component is neglected; *ii) Distortion Score.* It measures the global distortion caused by a given video stabilization method. It fits a homography matrix between the original and stabilized videos. Then, it finds the anisotropic scaling among these frames; the closer to 1, the better;
iii) Cropping Ratio. It describes the ratio of the remaining frame's area after stabilization to the original one; iv) *Success Rate.* We also measure the success rate, which computes the ratio of videos that were successfully processed and yielded a distortion score lower than or equal to one.

Table 2 shows the results of comparing our method to several video stabilization approaches. As can be seen, our method presented the best values on average in comparison to all the baselines in terms of stability average, distortion, cropping ratio, and success rate. Even though our method did not surpass the baselines at each class individually, it still achieved competitive results. Every baseline performs badly in at least one class, while our method is more robust across classes, hence holding the final best results on the VSAC105Real.

Preserving the content while compensating for camera shakiness is another important feature of our algorithm. Our method achieved the best results as compared to other

Table 2. **Comparison across different weather conditions in the VSAC105Real dataset.** Our method presents the best average values in comparison to the other competitors for all metrics. **Bold** indicates the best and <u>underline</u> second best.

Metric	Method	Weather Condition					Average
		Fog	Night	Normal	Rain	Snow	
Stability Avg. Score ↑	FuSta [6]	0.226	0.683	0.715	0.679	0.824	0.626 ±0.231
	Grundmann *et al.* [4]	0.642	0.549	0.620	0.580	0.809	<u>0.640</u> ±0.101
	StabNet [9]	0.201	0.469	0.620	0.577	0.753	0.524 ±0.207
	DIFRINT [2]	0.121	0.212	0.321	0.247	0.446	0.270 ±0.122
	Yu *et al.* [11]	0.401	0.682	0.665	0.572	0.834	0.631 ±0.159
	Ours	0.606	0.619	0.728	0.687	0.835	**0.695** ±0.093
Distortion Score *	FuSta [6]	0.202	0.692	0.725	0.712	0.798	0.626 ±0.240
	Grundmann *et al.* [4]	0.740	0.617	0.762	0.667	0.952	<u>0.748</u> ±0.128
	StabNet [9]	0.111	0.518	0.790	0.597	0.710	0.545 ±0.264
	DIFRINT [2]	0.367	0.219	0.351	0.270	0.476	0.337 ±0.099
	Yu *et al.* [11]	0.372	0.729	0.654	0.593	0.804	0.631 ±0.165
	Ours	0.719	0.746	0.952	0.809	0.997	**0.845** ±0.124
Cropping Ratio ↑	FuSta [6]	0.286	0.810	0.905	0.800	0.950	<u>0.751</u> ±0.267
	Grundmann *et al.* [4]	0.759	0.618	0.760	0.663	0.948	0.750 ±0.127
	StabNet [9]	0.278	0.579	0.875	0.667	0.850	0.650 ±0.242
	DIFRINT [2]	0.399	0.234	0.392	0.280	0.490	0.359 ±0.102
	Yu *et al.* [11]	0.476	0.810	0.842	0.650	0.947	0.745 ±0.184
	Ours	0.762	0.760	0.945	0.820	0.999	**0.857** ±0.109
Success Rate ↑	FuSta [6]	0.280	0.816	0.905	0.762	0.905	0.734 ±0.261
	Grundmann *et al.* [4]	0.762	0.619	0.762	0.667	0.952	<u>0.752</u> ±0.128
	StabNet [9]	0.238	0.524	0.667	0.571	0.810	0.562 ±0.212
	DIFRINT [2]	0.429	0.238	0.381	0.286	0.476	0.362 ±0.099
	Yu *et al.* [11]	0.480	0.815	0.762	0.619	0.857	0.707 ±0.155
	Ours	0.765	0.762	0.950	0.814	1.000	**0.858** ±0.110

↑*Higher is better* **Better closer to* 1

state-of-the-art methods. The superiority of our method can be linked to the accurate affine transformation matrix estimation and the smoothing stage. Moreover, our method achieved the highest success rate compared to the competitors as shown in Table 2. It is worth noting that all baselines failed to stabilize most of the shaky videos at foggy weather conditions. The reason for this outcome is that participating media, like fog, work as a low pass filter that removes high-quality features that most videos stabilization algorithms depend on to estimate the camera trajectory. We highlight that even though our model was not trained on any samples under the foggy weather condition, it was capable of learning useful features from both raw images and optical flow.

3.2. Computational Time Analysis

Our proposed method requires two image frames sampled closely enough to accurately estimate the camera trajectory $\hat{T}$ given a shaky video $V = \{v_1, v_2, \dots, v_N\}$. Thus, to evaluate our proposed video stabilization's performance and computation cost under a low frame rate, we performed the following set of experiments. We analysed the performance of our model under three frame rates:

High-frame rate: This is the original setup of our video stabilizer. We assume the videos are captured at 24 to 30 fps. Under this setup, we sample every 2 consecutive frames *i.e.*, v_i and v_{i+1} where $i = 1, 2, 3, \dots, N-1$, and then we generate the optical flow. Then, we estimate the affine transformation and smooth the predicted camera trajectory $\hat{T}$ as explained in the main manuscript.

Mid-frame rate: For every 4 frames from the shaky video V, we skip three frames *i.e.*, we sample v_i and v_{i+4} where $i = 1, 5, 9, 13, \dots, N-4$. Then, we calculate the optical flow for the frames v_i and v_{i+4}. After that, we estimate the affine transformation and smooth the predicted camera trajectory.

Low-frame rate: Similarly, we skip 7 frames for every 8 frames from the shaky video. We sample v_i and v_{i+8} where

Table 3. **Computational Time Analysis:** Higher sampling rate gives more stable videos but requires more computations.

	2 consecutive frames	**1 at every 4 frames**	**1 at every 8 frames**
Stability Avg. Score $\uparrow$	$\mathbf{0.695}$	0.505	0.420
Distortion Score $*$	$\mathbf{0.845}$	0.784	0.640
Avg. Time per Frame (Sec) $\downarrow$	0.022	0.010	$\mathbf{0.007}$

$\uparrow$*Higher is better* $\downarrow$*Lower is better* **Better closer to* 1

$i = 1, 9, 17, 25, \ldots, N-1$. Then, we repeat the steps mentioned earlier.

Table 3 shows the results on VSAC105Real dataset. As expected, performing video stabilization at lower frame rates requires fewer computations. The computation complexity comes from two main factors: optical flow and affine transformation estimations. For a video of 10 frames, 9, 2, and 1 estimation(s) are required for optical flow and affine transformation for high, mid, and low frame rates, respectively.

While the computation cost is reduced with lower frames, the quality of the stabilized videos degrades. Lower frame rates make the camera path estimation noisier and, thus, the final stabilized video shakier and more distorted. It should be noted that for the main problem addressed in this work, 24 to 30 fps is the standard frame rate. Considering different frame rates may not be suitable given the dynamics of the scene.

4. Discussion

Our results demonstrate the advantages of using synthetic data with a specially designed ground truth and architecture. We argue that the main factors behind achieving good results using only a small amount of synthetic data are:

1. *Accurate ground truth:* Most supervised computer vision algorithms are trained using data collected and annotated manually for this task. However, video stabilization is more challenging as collecting ideal training data is not feasible. Some approaches utilize two cameras using a mechanical stabilizer to generate the required ground truth. The main issue is that the scene is captured by two different cameras and from two different view angles. Thus, the task of video stabilization becomes harder for the model to learn. Our novel approach for ground-truth generation achieves accurate ground truth. Thus, it helps learning video stabilization to converge. Please note that corrupt and noisy labels are well-known issues in computer vision [1, 7, 8].

2. *High-quality images and plausible scene composition:* Our synthetic data is composed of high-quality images, where the scenes comprise plausible configurations. These two properties mitigate the domain gap between the synthetic and real domains. Thus, our model generalizes well on real videos.

3. *Diversity:* The key reason behind achieving good results using small size synthetic dataset can also be due to the diversity of the generated videos. Our simulator applies various domain randomization techniques, so the attributes of the generated scenes are highly diverse.

4. *Our video stabilization algorithm:* Another key factor is using two separate networks and leveraging optical flow information to help with the video stabilization task.

References

[1] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070. PMLR, 2019.

[2] 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.

[3] Mar Gonzalez-Franco, Ofek, et al. The Rocketbox Library and the Utility of Freely Available Rigged Avatars. *Frontiers in virtual reality*, 1(article 561558), 2020.

[4] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR 2011*, pages 225–232. IEEE, 2011.

[5] 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.

[6] 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.

[7] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[8] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained

dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[9] 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, 2018.

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

[11] 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.

[12] 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.