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

Abdulrahman Kerim\textsuperscript{1,2}, Washington L. S. Ramos\textsuperscript{3}, Leandro Soriano Marcolino\textsuperscript{1}, Erickson R. Nascimento\textsuperscript{3,4}, Richard Jiang\textsuperscript{1}

\textsuperscript{1} Lancaster University, UK \textsuperscript{2} University for the Creative Arts, UK \textsuperscript{3} UFMG, Brazil \textsuperscript{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.

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 \cite{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 \cite{9}, Stabfr \cite{12}, Selfie Video \cite{10}, and LiuSigg2013 \cite{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-
Table 1. **Dataset statistics.** Comparison among the available video stabilization datasets and VSAC105Real dataset.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>#Videos</th>
<th>Average #Frames</th>
<th>Total #Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stabfr [12]</td>
<td>45</td>
<td>471</td>
<td>21,200</td>
</tr>
<tr>
<td>Selfie Video [10]</td>
<td>33</td>
<td>251</td>
<td>8,308</td>
</tr>
<tr>
<td>VSAC105Real</td>
<td>105</td>
<td>737</td>
<td>77,477</td>
</tr>
</tbody>
</table>

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

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
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 \( i = 1, 2, 3, \ldots, N - 1 \), 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 \( i.e., v_i \), \( v_{i+1} \) and \( v_{i+2} \) where \( i = 1, 5, 9, 13, \ldots, N - 4 \). 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.

<table>
<thead>
<tr>
<th></th>
<th>2 consecutive frames</th>
<th>1 at every 4 frames</th>
<th>1 at every 8 frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability Avg. Score↑</td>
<td>0.695</td>
<td>0.505</td>
<td>0.420</td>
</tr>
<tr>
<td>Distortion Score *</td>
<td>0.845</td>
<td>0.784</td>
<td>0.640</td>
</tr>
<tr>
<td>Avg. Time per Frame (Sec) ↓</td>
<td>0.022</td>
<td>0.010</td>
<td>0.007</td>
</tr>
</tbody>
</table>

*Higher is better \* Lower is better \* Better closer to 1

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

Table[5] 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


[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...


