Leveraging Synthetic Data to Learn Video Stabilization Under Adverse Conditions

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

Abstract

Stabilization plays a central role in improving the quality of videos. However, current methods perform poorly under adverse conditions. In this paper, we propose a synthetic-aware adverse weather video stabilization algorithm that dispenses real data for training, relying solely on synthetic data. Our approach leverages specially generated synthetic data to avoid the feature extraction issues faced by current methods. To achieve this, we present a novel data generator to produce the required training data with an automatic ground-truth extraction procedure. We also propose a new dataset, VSAC105Real, and compare our method to five recent video stabilization algorithms using two benchmarks. Our method generalizes well on real-world videos across all weather conditions and does not require large-scale synthetic training data. Implementations for our proposed video stabilization algorithm, generator, and datasets are available at https://github.com/A-Kerim/SyntheticData4VideoStabilization_WACV_2024.

1. Introduction

Over recent years, we have witnessed an explosion of videos being recorded and shared on the Internet. However, most shared videos are unedited and shaky, which makes them unpleasant to watch. Therefore, video stabilization techniques became essential in the video processing pipeline, gaining momentum as more unedited videos are being created and shared. Recent video stabilization approaches perform well under standard conditions but struggle under adverse ones. Moreover, collecting training videos in these adverse conditions is hard, dangerous, and time-consuming. Furthermore, creating photo-realistic synthetic videos simulating these conditions is complex, expensive, and poses certain limitations because of the domain gap problem between synthetic and real domains.

The training data bottleneck mentioned above causes many video stabilization methods to be essentially non-learning-based, commonly adopting affine or homography matrix estimation in the camera motion estimation step to extract the camera trajectory. Usually, feature extraction, description, and matching are involved in this process. Feature extractors like SIFT [19] and the learning-based ones, like R2D2 [24] and ASLFeat [20], perform well under standard conditions, but they may fail under challenging conditions, such as foggy, rainy, and snowy weather, as well as nighttime scenes. For instance, raindrop and snowflake particles along with texture-depleted scenes in fog or darkness pose a clear challenge to extract robust features. Failing to estimate the camera trajectory accurately, in the motion estimation stage, propagates the error to later steps of the process, decreasing the quality of the stabilized video.

Synthetic data have shown great progress in the field of computer vision, captivating numerous researchers who seek to apply it to diverse computer vision problems [12, 17, 21, 28, 29]. Most importantly, synthetic data holds the promise of addressing the lack of suitable data for training supervised learning models. However, we argue that the potential of synthetic data lies not only in the amount of generated data for training but also in how we design and use this data jointly with our methods.

In this paper, we propose a novel synthetic-aware video stabilization method that leverages synthetic data and achieves state-of-the-art results using only a small-scale synthetic dataset. Using specially designed synthetic videos for training, our algorithm can bypass the feature extraction step, commonly adopted by video stabilization methods, and thus be more robust to adverse weather conditions. We leverage the Unity engine to build a simulator that creates three-dimensional photo-realistic virtual worlds procedurally at run-time. The system automatically diversifies essential scene attributes like weather conditions, time of the day, and crowdedness. Most importantly, our simulator generates the required ground-truth training data for our learning-based video stabilization method. We also introduce VSAC105Real, a new evaluation dataset

with real adverse weather videos, since available benchmarks lack these weather conditions.

Our proposed method does not require any real data for training and is more robust than the state-of-the-art methods across different weather conditions. To the best of our knowledge, this is the pioneering work addressing video stabilization in adverse weather, utilizing synthetic videos. Despite supervised learning-based approaches [1, 18,30,34] being able to learn parameters like cropping window, sensitivity, and even to extract discriminative features, there is no sufficient labeled data for obtaining high-quality results in any condition. Sourcing, collecting, and annotating relevant data is cumbersome, time-consuming, error-prone, costly, and subject to privacy issues.

Hence, our main contributions are three-fold: *i)* a novel synthetic-aware video stabilization method, achieving state-of-the-art results on real videos while trained only on synthetic videos; *ii)* a new synthetic data generator capable of producing specially designed training videos; and *iii)* a new video stabilization dataset, VSAC105Real, composed of real videos spanning foggy, rainy, snowy weather, and nighttime attributes.

2. Related Work

Video Stabilization. Video stabilization methods are categorized into non-learning-based and learning-based. Non-learning-based video stabilization methods do not perform training. For instance, Grundmann *et al*. [9] stabilize the shaky camera trajectory using L1-norm optimization under constraints, and Bradley *et al*. [3] address the stabilization task as a constrained convex optimization problem. Although these methods do not require training data for tuning the model's parameters, they work only under predefined conditions, their parameters must be tuned manually, and their results tend to be less pleasant.

The learning-based approaches are classified into unsupervised and supervised approaches. Non-supervised methods require training videos but do not demand shaky-stable video pairs. DIFRINT [5], for example, is trained end-to-end and utilizes frame interpolation to synthesize middle frames for stabilization. However, supervised learning approaches require labeled data, which is the main limitation of applying them to video stabilization. StabNet [30] uses a mechanical stabilizer to generate ground-truth stable videos to train Convolutional Neural Networks for video stabilization. The network learns a warping transformation of multi-grids given the shaky and previously stabilized frames. While Liu *et al*. [18] apply a learning-based hybrid-space fusion to compensate for optical flow inaccuracy, Yu *et al*. [34] stabilize videos by computing the per-pixel warp field from the shaky video optical flow.

The previous methods present a partial solution to the video stabilization problem since they are assumed to work under normal weather conditions and sufficient illumination. However, finding resilient features in adverse conditions is rather challenging. For example, rain particles, foggy weather conditions, and low illumination pose clear challenges to finding robust features. Thus, it leads to inaccurate motion estimation and low-quality video stabilization. Our video stabilization method belongs to the supervised learning-based category. However, unlike other methods, we use only synthetic data for training. No pre-training or fine-tuning on real data is required by our method, and by using only a small-scale training dataset, it is more robust than state-of-the-art methods.

Affine and Homography Transformation. Estimating affine and homography transformations between two images is common to aligning one image with another. There are different ways to find these matrices, like applying a feature extractor (*e.g*., SIFT [19] and OAN [35]) and an outlier rejection algorithm (*e.g*., RANSAC [8] and MAGSAC [2]) or via learning approaches [6, 36].

Although the traditional feature extraction approach does well under standard conditions, it performs poorly under challenging scenarios. Supervised approaches cannot reflect scene parallax [32], and generating suitable training data is hard. Unsupervised approaches may solve these problems, but they fail under large baseline alignment, which makes them impractical for video stabilization under sharp camera movements [36].

In contrast to prior methods, our model learns the affine transformation through supervised learning on synthetic training data exclusively. Although it is possible to decompose the homography matrix to extract camera translation, rotation, and scale, it is inaccurate. Moreover, training a model to estimate the affine transformation is easier than estimating the homography. While homography can accurately model camera motion for a few frames, it introduces artifacts such as skew and perspective distortions as the number of frames increases [9, 15]. Additionally, it overfits even with some regularization. Thus, utilizing homography transformation is harder to train (more parameters and easier to overfit) and more subject to artifacts.

Synthetic Data Generation. Synthetic data is typically used to overcome training data scarcity for supervised learning models [4, 12, 17, 25, 28, 29]. Richter *et al*. [25] modified the GTA-V game to generate synthetic data and the corresponding ground truth for semantic segmentation. Shafaei *et al*. [28] used photo-realistic games to generate data for image segmentation and depth estimation. Dosovitskiy *et al*. [7] presented CARLA, an autonomous driving simulator that provides ground-truth data for semantic segmentation and depth estimation tasks. Recently, the UrbanScene3D simulator was proposed by Liu *et al*. [17] for

autonomous driving and flying support.

While Sim2RealVS [22] shares some similarities with our work, it uses GTA-V, a game not originally designed for synthetic data generation. In contrast, our simulator was purpose-built for this task, offering greater diversity and control. Sim2RealVS is limited to GTA-V assets, while our simulator allows us to procedurally create new cities for each synthetic video and easily modify or add scene elements, which is not possible with Sim2RealVS relying on GTA-V. Our work also complements [14] by not requiring user manipulations. In contrast to [11], which uses a complex teacher-student network to learn the affine transformation, we achieve this with a simpler, easier-to-train architecture. Moreover, unlike [31], our approach does not require tuning numerous hyper-parameters like smoothing iterations and multi-planar thresholds.

These previous methods partially solve the data generation issue because of the lack of control over the generation process. They fail to randomize scene elements, leading to less diverse datasets. Our simulator utilizes procedural content generation to create 3D virtual worlds and generates special training data for video stabilization. Our experiments show that generating appropriate training data and creating a synthetic data pipeline achieve superior results. Our algorithm provides better results in real videos even though our simulator does not generate state-of-the-art photo-realistic videos. Additionally, our synthetic-aware algorithm and specially designed synthetic data teach the model to accurately estimate the affine transformation while not being overfitted to the synthetic data distribution. Thus, it mitigates the domain gap and achieves satisfactory results on real data.

3. Methodology

Let $V = \{v_1, \dots, v_N\}$ be a shaky video composed of N frames. Our approach aims to generate a stabilized version $V' = \{v'_1, \dots, v'_N\}$ while preserving the original camera movement made by the recorder.

Our method has two stages: *i)* Motion Estimation; and *ii)* Trajectory Smoothing. In the former, we train a motion estimation network using the ground-truth data from generated synthetic videos to estimate an affine transformation matrix $\mathbf{A}_i$ for every consecutive frames v_i and v_{i+1}. Then, in the latter, we calculate the camera trajectory $\hat{T} = \{\hat{\tau}_1, \dots, \hat{\tau}_N\}$ from the estimated parameters $\hat{\mathbf{x}}_j$ for the pair of frames (v_j, v_{j+1}), and after smoothing $\hat{T}$, we warp and crop frames using transformations retrieved from the smoothed trajectory $\tilde{T} = \{\tilde{\tau}_1, \dots, \tilde{\tau}_N\}$. Figure 1 shows the pipeline.

3.1. Motion Estimation

The first stage of our pipeline estimates the camera motion throughout the video. Most existing 2D-based stabilization approaches apply key-point feature extraction and tracking to solve this task [9, 13, 16]. However, both steps may fail under adverse weather conditions due to repetitive textures and partial occlusions caused by rain and snow particles or textureless objects under foggy weather or at night. To overcome this issue and properly recover the camera motion in V, we propose estimating parameters t_x, t_y, θ, and s of an affine transformation matrix $\mathbf{A} = [s\cos\theta, -s\sin\theta, t_x; s\sin\theta, s\cos\theta, t_y]$ for every consecutive pair of frames using synthetic data for deep estimation. Thus, we abdicate the feature extraction procedure entirely since, using our proposed engine, we can generate the ground-truth affine transformation needed for training as described in Section 4.

We use two identical networks for estimating the parameters: $f_{tr} : \mathbb{R}^{4\times W\times H} \rightarrow \mathbb{R}^2$, which estimates the x and y translations $\mathbf{x}^{tr} = [t_x, t_y]$; and $f_{rs} : \mathbb{R}^{4\times W\times H} \rightarrow \mathbb{R}^2$, which predicts the rotation angle and scale $\mathbf{x}^{rs} = [\theta, s]$, where $W = H = 256$ is the center-cropped image width and height. Note that f_{tr} and f_{rs} share the same architecture but not the same weights. Both networks consist of a feature extractor implemented as four convolutional layers, a pooling and a dropout layer, and a regressor, which is a fully connected network composed of three linear layers that process the extracted features to estimate the parameters. Figure 1-a shows the number of output channels of each layer. For each training step, we feed the networks with an input $I = [v_i; v_{i+1}; O_i] \in \mathbb{R}^{4\times W\times H}$, where $v_i, v_{i+1} \in \mathbb{R}^{W\times H}$ are two consecutive grayscale frames from the input video V and $O_i \in \mathbb{R}^{2\times W\times H}$ is the dense Optical Flow (OF) map for the pair (v_i, v_{i+1}). Then, we estimate the parameters for $\mathbf{A}_i$ as $\hat{\mathbf{x}}^{tr} = f_{tr}(I)$ and $\hat{\mathbf{x}}^{rs} = f_{rs}(I)$. To optimize the networks f_{tr} and f_{rs}, we train separately each one using the MSE loss.

A key contribution of this paper is the usage of specially designed synthetic data to learn an affine transformation matrix. Let $\mathcal{P}_i = \{\mathbf{p}_1, \dots, \mathbf{p}_K\}$ denote the 2D coordinates of K marked points at the frame v_i from a generated synthetic video. Since we can control the camera motion during the synthetic video generation, we can analytically determine the new positions of the marked points in frame v_i as they transition to frame v_{i+1}. With these $2K$ points in frames v_i and v_{i+1}, we can compute an affine transformation $\mathbf{A}_i$ with 4 degrees of freedom using $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$, then use it as the ground truth for training f_{tr} and f_{rs}. We detail the process of ground truth generation in Section 4.

Finally, with $\hat{\mathbf{x}} = [\hat{\mathbf{x}}^{tr}, \hat{\mathbf{x}}^{rs}] = [\hat{t}_x, \hat{t}_y, \hat{\theta}, \hat{s}]$, the estimated parameters for each video frame pair, we compute the estimated camera trajectory $\hat{T} = \{\hat{\tau}_1, \dots, \hat{\tau}_{N-1}\}$, where $\hat{\tau}_i = \sum_{j=1}^{i} \hat{\mathbf{x}}_j$, and $\hat{\mathbf{x}}_j$ represents the estimated parameters for the pair of frames (v_j, v_{j+1}). It is important to note that, similar to Grundmann *et al*. [9], we do not directly use $\hat{t}_i$ to warp the shaky images. We warp the frames

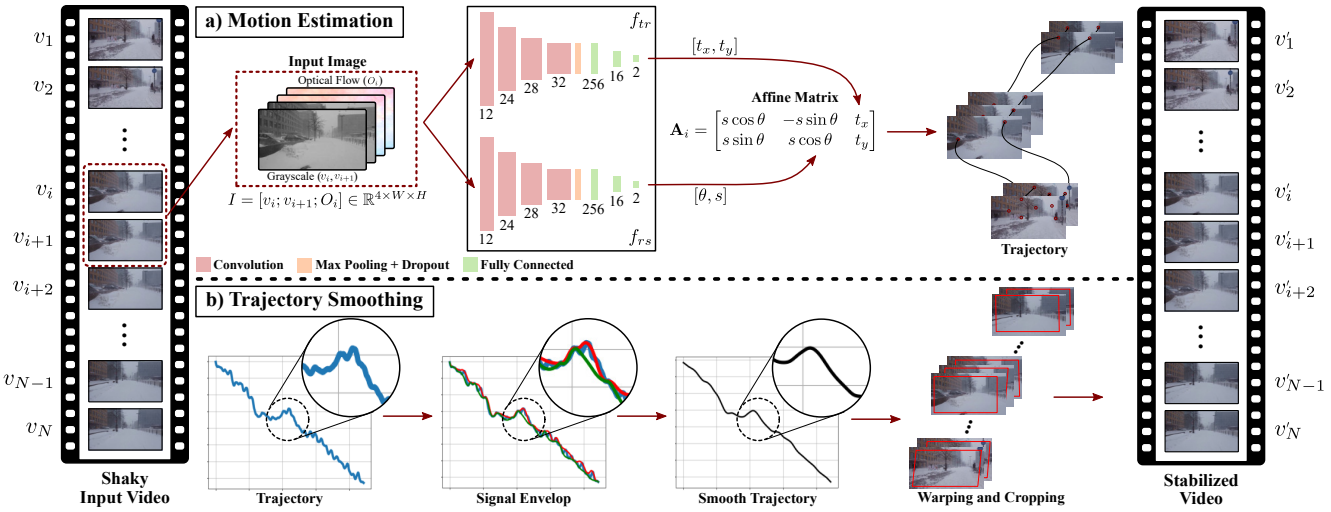

Figure 1. **Video stabilization pipeline.** Our method estimates the translation, rotation, and scale for each pair of frames of the shaky video. After computing the camera trajectory, upper (red) and lower (green) bounds are found and averaged, and the Savitzky-Golay filter is applied to smooth the trajectory. Finally, warping and cropping are performed.

by applying the smoothed affine transformations composed of the smoothed translation, rotation, and scale parameters.

3.2. Trajectory Smoothing

The next step after estimating the shaky camera trajectory is smoothing it. Unlike other methods, which tackle the camera trajectory smoothing as an optimization problem [9], we deploy the Savitzky-Golay filter [27] on the averaged envelop of the shaky camera trajectory to smooth it, as described in the sequel.

Given the camera trajectory $\hat{T}$, we first calculate the extremes of $\hat{T}$ by applying the first-order discrete derivative. Then, we interpolate the trajectory maxima ($\hat{T}_{max}$) and minima ($\hat{T}_{min}$) values to extract the upper and lower envelopes, respectively. Quadratic interpolation presented the best results in our experiments since it makes smooth interpolations and tends to stay within the ranges of the interpolation points. The final upper and lower signal envelopes are represented as $E_{up} = \{e_1^{up}, \ldots, e_{N-1}^{up}\}$ and $E_{low} = \{e_1^{low}, \ldots, e_{N-1}^{low}\}$, respectively.

After obtaining the envelopes, we apply the Savitzky-Golay filter on the average envelop $\bar{E} = (E_{up} + E_{low})/2$ to remove the unwanted sudden camera shakiness and create the smooth camera trajectory $\tilde{T} = \{\tilde{t}_1, \ldots, \tilde{t}_{N-1}\}$ as shown in Figure 1-b. The Savitzky-Golay filter smooths the digital signal by fitting a low-degree polynomial to consecutive signal points using linear least squares. This strategy has an advantage over other techniques as it preserves the signal tendency. Thus, $\tilde{T}$ still maintains the properties of $\hat{T}$ while ensuring a smooth camera transition over time. After that, we calculate the difference between both trajectories $\delta_T = \hat{T} - \tilde{T}$. Then, the smoothed affine transformation parameters $\tilde{X}$ are calculated as $\tilde{X} = \hat{X} - \delta_T$, where $\tilde{X} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{N-1}\}$ and $\hat{X} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{N-1}\}$, with $\tilde{\mathbf{x}}_i$ being the smoothed parameters $\tilde{t}_x$, $\tilde{t}_y$, $\tilde{\theta}$, and $\tilde{s}$.

At last, we warp and crop the video frames to compose the final video. For each video frame v_i, we compute its warped version $\tilde{v}_i$ by applying a transformation matrix to every pixel. Formally, we retrieve $\tilde{\mathbf{x}}_i$ from the smoothed transformations $\tilde{\mathbf{X}}$ and use the smoothed parameters $\tilde{t}_x$, $\tilde{t}_y$, $\tilde{\theta}$, and $\tilde{s}$ to compose the smoothed affine matrix $\tilde{\mathbf{A}}_i$. Then, we crop the warped frames using a predefined virtual cropping window similar to [9] to generate the stabilized video.

4. Synthetic Data and Ground truth

Synthetic Data Generation. There are many synthetic data generators like CARLA [7] and UrbanScene3D [17] that simulate photo-realistic and diverse 3D worlds in the literature. However, generating special data in such engines is cumbersome, and they do not support video stabilization. Therefore, in this paper, we introduce a new synthetic data generator capable of filling this gap and creating the required training data for this task. Our generator supports other computer vision tasks like semantic and instance segmentation, depth, and pose estimation. However, this paper focuses on its usability for the video stabilization task. We show that more vital than photo-realistic and diverse 3D scenes is designing computer vision models targeted at using synthetic data and generating the appropriate data for these models. We control scene aspects in virtual worlds and generate more suitable training data for supervised learning algorithms. A shaky synthetic video is recorded after procedurally creating a 3D virtual world sampled from a predefined set of 3D models, materials, and animations. Note that for each video, a new virtual world

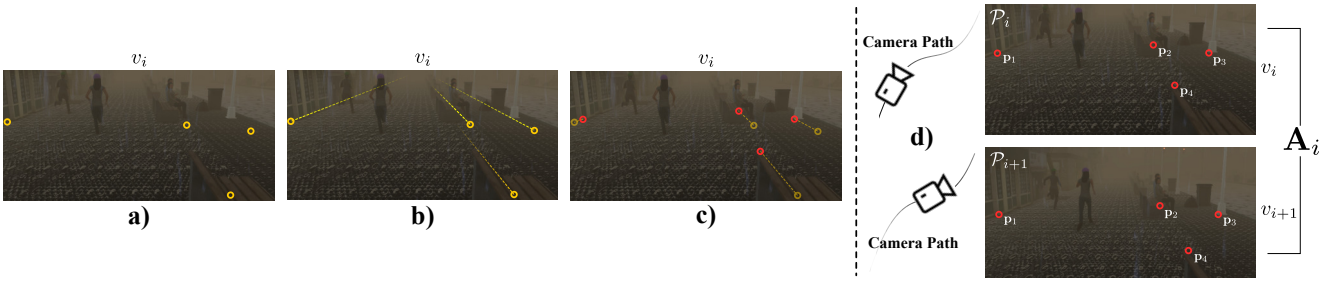

Figure 2. **Ground-truth Generation.** K points are randomly sampled from the screen space (yellow circles in a). From each of these points, we cast rays to infinity in the 3D scene space (dashed yellow lines in b) and create hypothetical objects at the intersection of these rays with the scene (red circles in c). We obtain the affine transformation matrix A_i using the coordinates of the hypothetical objects in screen space $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ since they remain stationary in the scene from the frame v_i to v_{i+1}.

Figure 3. Samples from the procedurally generated scenes using the proposed synthetic data generator.

is created to diversify the training data. Figure 3 demonstrates examples of the generated scenes.

To introduce shakiness to the recording camera, we create noise using a predefined noise profile asset. The amplitude and frequency of the noise are randomly sampled from a uniform distribution. The noise is applied to change the translation and rotation of the recording agent camera.

Ground-truth Generation. The goal of our networks proposed in Section 3.1 is to infer the affine transformation matrix, $\mathbf{A}$, given two consecutive frames. Since finding, collecting, and annotating data is a complex and expensive process, we generate and use synthetic videos to obtain the ground truth affine transformation matrices to supervise the training process.

Our idea works as follows: We first create stationary hypothetical labeled objects in the 3D world scene; then, we record their coordinates in the screen space of the recording camera. In that way, we can guarantee that the coordinates of these objects in frames (v_i, v_{i+1}) correspond to exactly the same static elements in the 3D world seen by the recording camera at frames v_i and v_{i+1}.

In other words, we create a number of invisible objects and save their coordinates in the camera space for each frame. To do so, for each frame, a number of random points are sampled from the screen camera space. Then, a ray is cast from each of these points to infinity. At each ray's intersection point with the scene, a hypothetical invisible object is created. The object remains stationary for a number of seconds before being destroyed. For each frame, the object's position in world space is transferred to the camera space. Figure 2 demonstrates how these hypothetical objects are created. If the hypothetical object is not in the camera view, or if it exceeds its time limit duration β, it is removed. Each object is given a Unique Identifier (UID) over its lifetime. Later in the post-processing stage, for every two consecutive frames (v_i, v_{i+1}) using the UIDs of these hypothetical objects and their screen locations, we calculate the ground-truth affine transformation for each pair. The process is further clarified as described in Algorithm 1. While generating numerous hypothetical objects is feasible, our algorithm creates objects within the recording agent's field of view, enhancing overall performance.

5. Experiments

5.1. Experimental Setup

Synthetic Datasets. Using our generator, we create two synthetic training datasets: VSNC35Synth and VSAC65Synth. VSNC35Synth includes 35 videos at 24 fps with an average of 400 frames per video; it covers only videos in normal weather conditions. The average number of frames was set to 400 to match the number of frames in other datasets. VSAC65Synth consists of 65 videos, including normal and adverse weather condition videos. The classes span normal, rainy, foggy, and snowy weather conditions at daytime and night-time.

Real Dataset. To evaluate video stabilization methods under adverse conditions, we created the VSAC105Real dataset since available benchmarks are missing these attributes. 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 4. **VSAC105Real versus other datasets.** On top, each dashed box shows frames from other datasets. Our proposed dataset, VSAC105Real, is at the bottom. It includes more diverse and challenging attributes as compared to the other datasets.

Algorithm 1: Affine Transformation Ground-truth Generation

Require: N_{frames}, T
Ensure : Per frame file containing camera screen locations of stationary hypothetical objects.
N_{frames}; ▷ Total number of frames to generate
T; ▷ Sampling period
while *Recording* **do**
  for $ID_{frame} = 0;\ ID_{frame} < N_{frames};\ ID_{frame}{+}{+}$ **do**
    if $ID_{frame} \% T == 0$ **then**
      $Points \leftarrow samplePoints(\text{K})$; ▷ Sampling K points on camera screen space
      foreach *point* $\in Points$ **do**
        $M \leftarrow castRayToInfinity(\text{point})$; ▷ Cast a ray from *point* to infinity
        $point_{inters} \leftarrow findIntersPoint(M)$; ▷ Intersection point between the ray with scene objects
        $O \leftarrow createHypotheticalObject(point_{inters})$;
        $O.UID \leftarrow assignObjectUID(O)$; ▷ Assign O a unique identifier
        $O.ScreenPos \leftarrow Cam.WorldToScreen(O.WorldPos)$; ▷ Transfer coordinates from world to screen space
        while *O is visible and within its lifetime* **do**
          $Save(O.ScreenPos)$;
          $WaitFewFrames()$;
        $Destroy(O)$;
    else
      $WaitFewFrames()$;

ing camera movement. Then, we cut the videos to ensure continuous temporal and query attributes. VSAC105Real dataset comprises 105 videos spanning normal, rainy, foggy, snowy, and night-time attributes. Compared to other datasets, VSAC105Real is better regarding the average number of frames, the diverse set of challenging attributes, and the even distribution of videos across the classes, *i.e.*, 21 videos per class. A visual comparison among VSAC105Real and other video stabilization datasets is depicted in Figure 4. See supplementary material for statistical comparison among them.

Evaluation Metrics. We use four metrics commonly used to evaluate video stabilization algorithms [5, 18, 30, 34]: *i)* the Stability Score, which assesses the smoothness of the stabilized video; the higher the value, the better; *ii)* the Distortion Score, which measures the global distortion caused by a given video stabilization method; *iii)* the Cropping Ratio, which describes the ratio of the remaining frame's area after stabilization to the original one; and *iv)* the Success Rate, which computes the ratio of videos that were successfully processed and yielded a distortion score lower than or equal to one.

Baselines. We evaluate five state-of-the-art video stabilizers on two real datasets: VSAC105Real and Selfie Video [33]. The baselines span non-learning based (Grundmann *et al*. [9]), supervised (StabNet [30]), and unsupervised (DIFRINT [5]) video stabilization methods. Furthermore, we compare our method to Yu *et al*. [34], which heavily relies on optical flow since we use optical flow, and FuSta [18] because it also uses CNNs for video stabilization similarly to our approach.

Implementation Details. We trained our method using only the synthetic data provided by our simulator, *i.e.*, VSNC35Synth which contains only normal weather conditions videos. The hypothetical object's time limit duration was empirically set to $\beta = 1$ second. We trained the t_x and t_y translation prediction model (f_{tr}) and rotation θ and scale s prediction model (f_{rs}) for 65 and 2 epochs, respectively, using batches of size $M = 40$. After 10 epochs,

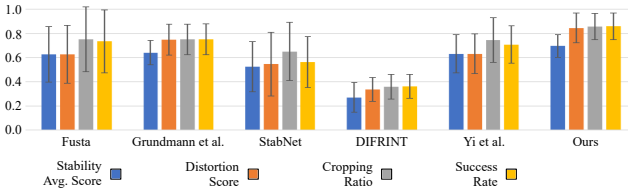

Figure 5. Comparison across different weather conditions in the VSAC105Real dataset.

we decrease the learning rate of f_{tr} to $1e{-}5$. Our architecture is fully implemented in PyTorch, and the training procedure takes 33 hours on a Tesla V100 GPU. For the smoothing step, a window length, *i.e.*, number of coefficients, equal to 51 with 1^{st} order polynomial were used as parameters to the Savitzky-Golay filter as they give better results. We used FlowNet2 [10, 23] as the OF estimator.

5.2. Results

Figure 5 shows the comparison among the competitors across different weather conditions in the VSAC105Real dataset. Our method presented the best values, on average, in terms of stability average and distortion scores, cropping ratio, and success rate. We argue that its superiority relates to the accurate affine transformation matrix estimation and the smoothing stage. Our algorithm preserves the frame content while compensating for camera shakiness.

We highlight that our method achieved the highest success rate compared to the competitors. All baselines failed to stabilize most of the shaky videos in foggy weather conditions due to the nature of participating media. Fog works as a low-pass filter that removes high-quality features. Most video stabilizers depend on resilient features to estimate the camera trajectory. Even though our model was not trained on foggy weather conditions, it learned useful features from both raw images and optical flow.

We also evaluate our model on the Selfie Video dataset [33], which contains videos under normal weather conditions and standard illumination. Results are presented in Table 1. Even though our model was trained from scratch solely on our synthetic data, it achieved the best distortion score. For a detailed analysis of computational time, please see the supplementary material.

Ablation Study. We analyzed the design options and showed the effectiveness of each component of our pipeline. The results are reported in Table 2.

First, we assess a variant that uses a single CNN instead of two, as we proposed in our final model. As a result, the network was unable to converge well. One problem could be that the translation losses were larger than others. However, even with loss weighting applied, the network could not learn well (*Single Network* row). We hypothesize that the reason is the large value range between translation (*i.e.*, 0 to image height/width) and scale (*i.e.*, 0 to 1). Even with normalization, a single network struggled to back-propagate a meaningful error.

Table 1. **Comparison in the Selfie Video dataset [33]**. **Bold** indicates the best, underline second best, and *italic* the third.

Method	Stability Avg. Score↑	Distortion Score∗	Cropping Ratio↑	Success Rate↑
FuSta [18]	0.818	*0.777*	**0.970**	**0.970**
Grundmann *et al.* [9]	0.727	0.828	0.848	0.848
StabNet [30]	0.763	0.680	*0.917*	0.667
DIFRINT [5]	**0.827**	0.691	0.912	*0.915*
Yu *et al.* [34]	0.770	0.739	0.909	0.909
Ours	*0.787*	**0.933**	0.939	0.939

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

Table 2. **Ablation study.** Performance for different design choices (best in **bold**).

Variant	Stability Avg. Score↑	Distortion Score∗	Cropping Ratio↑	Success Rate↑
Single Network	0.443	0.540	0.557	0.543
SIFT	0.576	0.650	0.670	0.667
No Optical Flow	0.678	0.781	0.829	0.800
Directed Smoothing	0.675	0.828	0.844	0.838
More Data	0.690	0.793	0.850	0.829
Complete Model	**0.695**	**0.845**	**0.857**	**0.840**

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

To emphasize the advantages of using our learning-based model for affine transformation matrix estimation over applying SIFT, we apply SIFT to find the affine transformation matrix while keeping the smoothing part of our model intact. As expected, standard feature extractors like SIFT struggle to extract robust features under adverse conditions. Rain and snow particles, low illumination at night, and foggy weather lead to inaccurate affine transformations and, thus, low-quality stabilized videos.

To evaluate our smoothing algorithm's advantages over l_1 directed smoothing, as done in [9], we apply l_1 directed smoothing on the predicted camera trajectory while keeping our learning-based model for the affine transformation matrix estimation. As expected, the model does not perform very well as compared to using our proposed smoothing algorithm (*Directed Smoothing* row) because ours considers more sophisticated camera paths and is not limited to constant, linear, and parabolic motions like [9].

To highlight the importance of optical flow in affine transformation learning, we train a variant using only grayscale images, i.e., $I = [v_i; v_{i+1}]$. As expected, excluding optical flow reduced video stabilization quality (see *No Optical Flow* row).

To assess the impact of our synthetic data on video stabilization quality, we trained our model from scratch on more data using VSAC65Synth dataset, encompassing nor-

Table 3. **Affine matrix estimation**. Comparison among different methods for affine matrix estimation on CA-Unsupervised dataset [36] using l_2 distance (best in **bold**).

Method	RT↓	LT↓	LL↓	SF↓	LF↓	Average↓
ORB [26] + RANSAC [8]	9.24	14.63	12.27	11.36	7.20	10.94
ORB [26] + MAGSAC [2]	10.11	19.79	12.48	11.86	7.85	12.42
ORB [26] + LMEDS	9.78	40.11	12.02	10.84	7.01	15.95
SIFT [19] + RANSAC [8]	10.63	11.70	13.37	11.75	6.44	10.78
SIFT [19] + MAGSAC [2]	10.75	10.97	12.99	11.09	6.35	10.43
SIFT [19] + LMEDS	10.47	9.72	13.04	10.14	5.88	9.85
Supervised [6]	8.39	9.33	8.63	10.29	5.92	8.51
Unsupervised [36]	7.05	7.60	6.84	7.42	**3.84**	6.55
Ours	**4.55**	**5.58**	**5.68**	**5.17**	9.73	**6.14**

↓ Lower is better

mal and adverse weather conditions, day and night videos. The results indicated no significant improvement over the method trained on VSNC35Synth (*Complete Model* row). Therefore, a few synthetic videos with accurate ground truth are sufficient to train the model.

To investigate the accuracy of our estimated affine transformation matrix, we compare our learning-based affine transformation estimation with several estimation approaches, including the traditional ones like ORB and SIFT with RANSAC, MAGSAC, and LMEDS for outliers rejection, the supervised, and the unsupervised ones. Traditional approaches estimate the affine transformation directly, but supervised and unsupervised methods are designed to estimate the homography matrix. Thus, we extract the affine transformation from their estimated homography for a fair comparison. We utilize the dataset of [36], which contains $4{,}200$ pairs of images where each image-pair includes six matching human-annotated pairs of points. It covers regular texture (RT), low texture (LT), low light (LL), small foregrounds (SF), and large foregrounds (LF). We use l_2 distance to measure the error between the warped and ground-truth points similar to [32, 36].

Table 3 demonstrates the generalizability of our affine estimation model, which is better on both standard and challenging conditions. The dataset used in this experiment includes challenging images, such as low-texture images similar to images under foggy or snowy weather conditions and images at low illumination similar to the ones at nighttime. These results demonstrate the ability of our affine estimator of learning to extract robust features under challenging conditions.

Figure 6 shows a qualitative comparison among our model, traditional (*e.g.*, SIFT and ORB), supervised [6], and unsupervised [36] methods, demonstrating examples of low texture pair of images. While other methods fail under such challenging conditions because they were manually tuned for standard settings or not trained with data under adverse conditions, our method performs well because it learned how to extract resilient features using our specially designed synthetic data.

Figure 6. Qualitative comparison for affine transformation estimation.

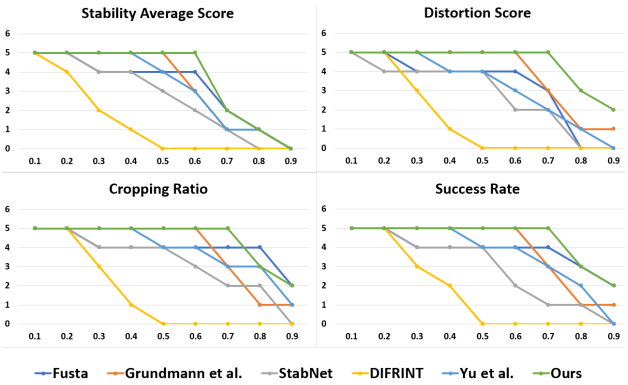

Figure 7. **Classes coverage by threshold.** Number of times the metrics values for classes in Figure 5 are above each threshold. Our method presents the highest metrics coverage, achieving the best results compared to other video stabilization methods.

Additionally, we present in Figure 7 a new metric that computes the coverage of classes. The figure shows the number of classes (*i.e.*, fog, night, normal, rain, and snow) whose output values achieved a result above different thresholds (*i.e.*, 0.1 to 0.9 with a 0.1 step). As expected, our method presents the highest coverage as the threshold values increase, achieving the best results compared to other video stabilization methods.

6. Conclusion

Recent video stabilization methods struggle under adverse conditions. In this paper, we proposed a synthetic-aware video stabilization method that requires only synthetic data for training, surpassing all other baselines. We also provided one real dataset for video stabilization under adverse conditions and two synthetic datasets for training produced by our novel synthetic data generator. Currently, we are randomly and manually diversifying the synthetic data generation parameters. In the future, we intend to use active learning to guide this generation process.

Acknowledgement. This work was funded by the Faculty of Science and Technology of Lancaster University. We thank the High End Computing facility of Lancaster University for the computing resources.

References

[1] Muhammad Kashif Ali, Sangjoon Yu, and Tae Hyun Kim. Learning deep video stabilization without optical flow. *arXiv preprint arXiv:2011.09697*, 2020.

[2] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: marginalizing sample consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10197–10205, 2019.

[3] Arwen Bradley, Jason Klivington, Joseph Triscari, and Rudolph van der Merwe. Cinematic-L1 Video Stabilization with a Log-Homography Model. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1041–1049, 2021.

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *European conference on computer vision*, 2012.

[5] 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.

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[8] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] 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.

[10] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[11] Jerin Geo James, Devansh Jain, and Ajit Rajwade. Globalflownet: Video stabilization using deep distilled global motion estimates. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5078–5087, 2023.

[12] Abdulrahman Kerim, Ufuk Celikcan, Erkut Erdem, and Aykut Erdem. Using synthetic data for person tracking under adverse weather conditions. *Image and Vision Computing*, 111:104187, 2021.

[13] Ken-Yi Lee, Yung-Yu Chuang, Bing-Yu Chen, and Ming Ouhyoung. Video stabilization using robust feature trajectories. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1397–1404. IEEE, 2009.

[14] 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*, pages 10621–10630, 2021.

[15] Shiwei Li, Lu Yuan, Jian Sun, and Long Quan. Dual-feature warping-based motion model estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4283–4291, 2015.

[16] 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.

[17] Yilin Liu, Fuyou Xue, and Hui Huang. Urbanscene3d: A large scale urban scene dataset and simulator. *arXiv preprint arXiv:2107.04286*, 2021.

[18] 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.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[20] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] Luca Piano, Filippo Gabriele Pratticò, Alessandro Sebastian Russo, Lorenzo Lanari, Lia Morra, and Fabrizio Lamberti. Bent & broken bicycles: Leveraging synthetic data for damaged object re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4881–4891, January 2023.

[22] Qi Rao, Xin Yu, Shant Navasardyan, and Humphrey Shi. Sim2realvs: A new benchmark for video stabilization with a strong baseline. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5406–5415, 2023.

[23] Fitsum Reda, Robert Pottorff, Jon Barker, and Bryan Catanzaro. Flownet2-pytorch: Pytorch implementation of FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. `https://github.com/NVIDIA/flownet2-pytorch`, 2017.

[24] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[25] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In *European conference on computer vision*, 2016.

[26] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[27] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[28] Alireza Shafaei, James J Little, and Mark Schmidt. Play and Learn: Using Video Games to Train Computer Vision Models. *arXiv:1608.01745*, 2016.

[29] Apostolia Tsirikoglou. *Synthetic data for visual machine learning: A data-centric approach*. PhD thesis, Linköping University Electronic Press, 2022.

[30] 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.

[31] Yufei Xu, Jing Zhang, Stephen J Maybank, and Dacheng Tao. Dut: Learning video stabilization by simply watching unstable videos. *IEEE Transactions on Image Processing*, 31:4306–4320, 2022.

[32] Nianjin Ye, Chuan Wang, Haoqiang Fan, and Shuaicheng Liu. Motion basis learning for unsupervised deep homography estimation with subspace projection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13117–13125, 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. Learning video stabilization using optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8159–8167, 2020.

[35] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5845–5854, 2019.

[36] Jirong Zhang, Chuan Wang, Shuaicheng Liu, Lanpeng Jia, Nianjin Ye, Jue Wang, Ji Zhou, and Jian Sun. Content-aware unsupervised deep homography estimation. In *European Conference on Computer Vision*, pages 653–669. Springer, 2020.