# Team 18 - Mid-Term Project Report

G Karthik Balaji
cs19btech11001@iith.ac.in

Rachit Keerti Das
cs19btech11034@iith.ac.in

Gunangad Pal Singh Narula
cs19btech11035@iith.ac.in

## Abstract

*Our goal is to present a software-based algorithm for video stabilization which eliminates unintended camera wobble, while preserving the intended camera motion. We aim to perform a deep dive into the various techniques and models that already exist in the field, and attempt to construct a deep learning model that utilizes the relevant metadata from the video frames to synthesize a stable output video. In specific, we shall focus on diverse aspects such as optical flow vectors, depth information and motion trajectories and attempt to extract additional information that would enhance the performance of our algorithm.*

## 1. Introduction

Video stabilization is a problem that arises due to the jittery or wobbly motion of the camera while capturing the video, and seeks to generate a stable version of the same. With the advent of the modern smartphone era, there is an increasing amount of focus on the field due to the significant number of videos captured using hand-held devices, which are more prone to wobbling.

The resulting shaky video is agonising for both the viewer and the cameraman, and is a potential waste of resources. Video stabilization methods attempt to correct these jitters in an optimal and efficient manner.

Early stabilization methods included one-point and two-point tracking, that located the respective number of points and attempted to keep them at a constant position in the frame. However these methods fail in a setting where the subjects change over time. This necessitates the need to find more efficient algorithms.

## 2. Problem Statement

Video stabilization is a nascent field that gives rise to novel approaches on a regular basis. This paper serves as an exploratory exercise that experiments with various techniques proposed in the past and aims to find a hybrid model that builds upon these works.

We desire to introduce an efficient algorithm that stabilizes a given video, eliminating visual distortion caused by wobbling or shaking of the video apparatus. We currently aim to extract relevant and useful metadata from the video frames, such as the depth information of various objects in the frame, and the optical flow of the various image features; the metadata-enforced image data will then be fed to our model that smoothens out the respective image trajectories, while maintaining relative motion consistency between the various image components. We intend to explore different neural network architectures and analyse the explainability and results of the best performing model.

## 3. Literature Review

Traditional Methods for Video Stabilization have involved both image-based (software) and sensor-based (hardware) approaches.

Hardware approaches for tackling this problem have included reducing the camera motion through OIS and other aids. These solutions are used to this day, and technical implementations of OIS and its associated solutions have been explored in detail extensively. [4]

A very recent approach involves fusing the gyroscope sensor's data and traditional image-based metadata and passing it through neural networks before applying further processing to produce highly stabilized output videos across various environments. [6]. However, such approaches are possible only when the required hardware/metadata is available. This might not always be the case, thus necessitating the case for purely software-centered methods.

One of the early software methods include YouTube's video stabilization algorithm. This algorithm uses the tool of linear programming to minimize the first, second and third derivatives of the estimated L1-optimal camera path, and synthesizes the resulting video based on this new estimated smooth path [3]. This is achieved by breaking the path into partitions of constant, linear and parabolic paths, and the algorithm attempts to find the optimal partition of these paths. L1 optimization is preferred, as L1 optimization attempts to solve the required constraints exactly, while L2 optimization focuses on them ”on an average”. The algorithm is successfully able to remove low-frequency motion disturbances, an improvement upon the conventional algorithms that only suppress high frequency jitters.

Sometimes, it is not enough to just use 2D motion methods to estimate camera movement, as there can be a perceived global shaking even after processing, due to their inability in handling parallax. In such a case, it is useful to construct a 3D spatial mapping of the objects present in the image, and eliminate parallax. A similar approach has been tried in the past, by (Wang et al. [7]) , creating maps of the image features and extracting their trajectories. These trajectories are then fitted onto Bezier curves, and stabilization is essentially represented as a spatial-temporal optimization problem. The Bezier representation helps ensure smoothness, minimizes distortion and also helps reduce the number of variables in the optimization problem, thereby reducing the computational complexity. However, this method can be ineffective, in the case where there are very little stationary or background features in the image.

Yet another software-based approach has been to use deep learning architectures, a popular tool in the field of computer vision. One such method [8] generates an optical flow field from the input video, and feeds this to a deep neural network, which then generates the corresponding warp field. On the other hand, there are papers [1] that specifically focus on stabilizing the video using convolutional neural networks, without using optical flow. In particular, the model takes in five neighboring frames, and focuses on stabilizing the middle frame.

These diverse methods only highlight the versatility of neural networks, and the untapped potential that lies in them.

# 4. Preliminary Results

## 4.1. The Base Stabilization Algorithm

As a first step, we constructed a basic model for video stabilization using the standard concepts of optical flow, trajectory extraction and geometric transformations. Videos from the DeepStab dataset were used, as it is the most popular video stabilization dataset, with unstable and stable versions available for each video. However, for the purposes of this algorithm, any unstable video can be used to observe the video stabilization effect. This basic algorithm can be explained as follows.

The video is processed frame by frame. For each frame, initially, we extract the more important features present in the frame. This is done using corner detection. For this, the *Shi-Tomasi Corner Detection* [5] method is used, applied via the `goodFeaturesToTrack()` method in the OpenCV library. [2]

Next, the same objects are compared between neighbouring frames. The traceable object corners give us a set of good reference points for object tracking throughout the video. The *Lucas-Kanade algorithm* is used to extract the optical flow of these feature points between the neighbouring frames. Once again, the algorithm is available as a library function in OpenCV. To provide a high-level overview, the Lucas-Kanade algorithm assumes that the corners belonging to the same object satisfy the below optical flow equation.

$$\begin{aligned}
I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\
I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\
&\vdots \\
I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n)
\end{aligned}$$

In other words, it assumes that the feature points from the same object possess the same optical flow velocity. The above system of linear equations can be solved to obtain the optical flow corresponding to these points, which can then be used to find the location of the points in the current frame, based on their location in the previous frame.

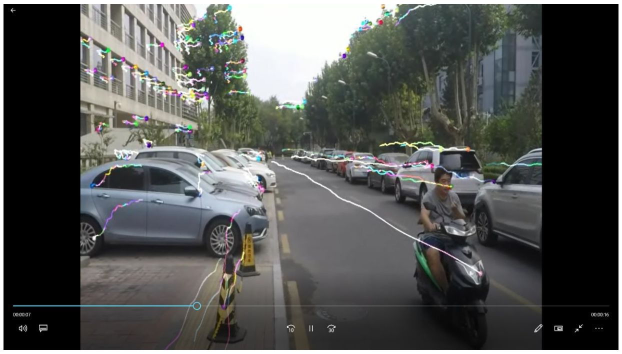

Figure 1. Depiction of optical flow

Having obtained the set of frame points, and their locations in both frames, we identify the Euclidean transformation that maps the points from the previous frame to the current frame. This transformation matrix can then be decomposed down to its translation (x and y) and rotation ($\theta$) components.

Thus, for each frame, the translation and rotation components that transform it to the next frame are obtained. These components can be cumulatively summed to obtain the trajectory vector. A trajectory vector corresponding to each component is obtained. An example trajectory for the `y` component, before and after smoothing is demonstrated in the figure below:

Once this trajectory is obtained, the actual video stabilization occurs. A moving average filter is applied on the trajectory vector, to smoothen small kinks and noise components obtained due to the unstable motion of the camera. In other words, a moving average filter (of a predefined window size) is convolved with the trajectory vector. Once the smoothened trajectory is obtained, the difference between the smoothened and original versions is computed.

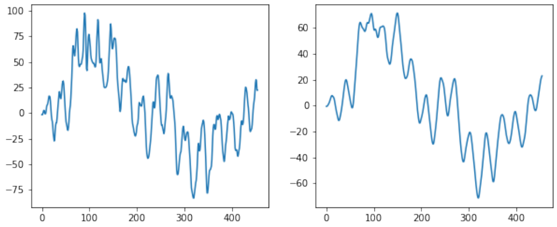

Figure 2. y Trajectories, before and after smoothing

By adding this difference to the original translational and rotational components, the new components are obtained. These components are used to construct the new transformation matrix corresponding to the current frame, which is then applied on the frame to generate the stabilized output frame. With this, the 'local' distortions in the camera motion, such as small jerks, are considerably reduced in the output video.

## 4.2. Our Improvements

To further improve on this basic model, we tried and implemented the following strategies:

### 4.2.1 Variations of smoothening used:

1. **Single Filter**: This is the direct implementation of our stabilization model. The filter size corresponds to the window range of the moving average filter used in generating the stabilized video.

2. **Progression Over Trajectory**: The stabilized videos are obtained by repeatedly applying moving average filters of increasing sizes (which in our application was in increments of 2) on the trajectory vector, to obtain a final smoothened trajectory vector. This has an added benefit of removing rather larger low frequency bumps. But has a downside of having a turbulent border which may increase as the final filter size increases. This effect can be observed on viewing the sample videos.

3. **Progressive Application on Video**: This method is slightly similar to the previous variation in terms of compounding the stability over iterations. Instead of applying the filters to the trajectory, at each step, the newly obtained trajectory is used to generate intermediate frames and thus, a video, and the next filter is then applied on the trajectory vector extracted from these new intermediate frames. Once again, the filters are applied in increments ( 2, in our demonstration).

The demonstrations for the various methods mentioned above, and the associated code, are available on our Github repository.

### 4.2.2 Experiments for removing larger perturbations:

For removing larger camera shakes which are longer in duration, and not very jittery, we then try to fit the smoothened trajectories with a sufficiently high-degree polynomial. For the purpose of our experiments, we chose the degree to be 11. We then applied a least-squares fit algorithm to obtain the polynomial coefficients approximating the trajectory.

Sample fits for the x, y and theta components respectively are depicted below:

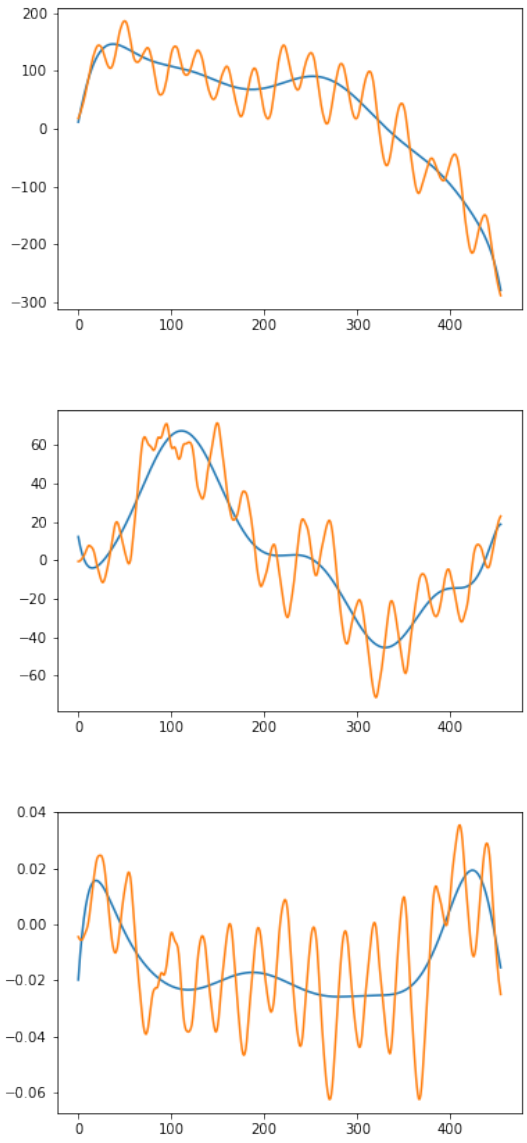

Figure 3. Polynomial Fit for polynomial of degree = 11

In the output, we noticed that while the focus scenes of the video were far more stable than just smoothening the local perturbations using a moving average filter, a visible border jarring effect was observed due to significant deviations from the observed values of theta. However, purely from a video stabilization perspective, much better stabilization was observed, due to the removal of larger perturbations in the trajectory.

The associated code can once again be found in the provided GitHub repository.

# References

[1] Muhammad Kashif Ali, Sangjoon Yu, and Tae Hyun Kim. Learning deep video stabilization without optical flow, 2020. 2

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2

[3] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR 2011*, pages 225–232, 2011. 1

[4] Fabrizio La Rosa, Maria Celvisia Virzì, Filippo Bonaccorso, and Marco Branciforte. Optical image stabilization (ois). *STMicroelectronics*, 2017. 1

[5] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 2

[6] Zhenmei Shi, Fuhao Shi, Wei-Sheng Lai, Chia-Kai Liang, and Yingyu Liang. Deep online fused video stabilization. *arXiv preprint arXiv:2102.01279*, 2021. 1

[7] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee. Spatially and temporally optimized video stabilization. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1354–1361, 2013. 2

[8] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8156–8164, 2020. 2