

Video Stabilisation for Teleoperated System

A Project Report

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

In

Programme

Electronics and Communication Engineering

by

Shivangi Srivastava | 16BEC0111

Uday Mukhija | 16BEC0001

Kshitij Tiwari | 16BEC0314

Under the guidance of

Dr. Rajesh Kumar Muthu

SENSE

VIT, Vellore.



May,2020

DECLARATION

I hereby declare that the thesis entitled “**Video Stabilisation for Teleoperated System** ” submitted by me, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Rajesh Kumar Muthu. We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “ **Video Stabilisation for Teleoperated Systems**” submitted by Shivangi Srivastava | 16BEC0111 , Uday Mukhija | 16BEC0001 and Kshitij Tiwari | 16BEC0314 ,School of Electronics Engineering (SENSE) , VIT University, for the award of the degree of *Bachelor of Technology in Programme*, is a record of bonafide work carried out by them under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

Programme

ACKNOWLEDGEMENT

This report on “Video Stabilization for Teleoperated Systems” has been brought to fruition by the efforts of many people. We would like to articulate our deep sense of gratitude to our project guide **Dr, Rajesh Kumar Muthu** who has always been a source of motivation and firm support for carrying out the project. This thesis work would have not been complete without his help and guidance.

We would also like to thank **The Head of the Department, ECE, Prof Thanikaiselavan, and The Dean, SENSE, VIT, Prof Kittur Harish Mallikarjun.**

We wish to express our sincere thanks to **Honourable Chancellor, Dr.G.Viswanathan; esteemed VicePresidents, Shri. Sankar Viswanathan; Shri. Sekar Viswanathan and Shri. G. V. Selvam; respected Vice Chancellor, Dr. Anand A. Samuel; respected ProVice Chancellor Dr. S.Narayanan** of this prestigious Institute for providing us an excellent world class academic environment and facilities for pursuing our B. Tech Program. Our special thanks to our friends for their timely help and suggestions rendered for the Successful completion of this project. This acknowledgement would be incomplete without expressing our whole hearted thanks to our parents for their continuous support and guidance in all walks of our lives.

Kshitij Tiwari | 16BEC0314

Uday Mukhija | 16BEC0001

Shivangi Srivastava | 16BEC0111

EXECUTIVE SUMMARY

Video stabilisation is an essential part of image processing. It's application in teleoperated systems has been an area of research which has been prominent recently. Our project shows an easy to understand program which stabilises the video feed frame by frame. Unlike other algorithms our project works frame by frame. Due to this, it is ideal for a live feed. Our program is divided into three stages:

- 1) Feature extraction
- 2) Inter frame motion detection and feature tracking
- 3) Correction phase

We have used Shi-Tomasi for feature extraction. This algorithm is proven by theoretical research to be ideal for feature tracking. Many other alternatives are known to be more system heavy. We have chosen it for its simplicity and ease of use.

For the second phase used the pyramidal implementation of the Lucas and Kanade model. This implementation is better than the original Lucas and Kanade model since it is more robust than its predecessor. In combination with the first phase it can efficiently track movement of features through the feed. It's theoretically proven to be efficient due to its simplicity. It also gives good results in spite of its simplicity.

For the correction phase we have used affine transformation and Kalman filter.

Affine transformations are mathematical transforms which match features in between consecutive frames and measure their movements. Using these values, the frame can be rotated and morphed based on its predecessor. These transformations give frame average values are given by affine transforms for each feature.

The values provided by affine transforms are then used by Kalman filter to rotate and morph the frame based on its prediction. Kalman filter is very effective as it does not need previous system values to operate and make predictions. It does that as the system progresses. It corrects the x and y axis frame average movements along with the frame average angular movements. This leads to morphing of the current frame hence smoothening the frame. This process is simple and effective for systems with live feeds. We have chosen methods which are theoretically proven to be fast and simple for all our steps.

CONTENTS

	PAGE
Acknowledgement	i
Executive Summary	ii
Table of Contents	Iii
List of Figures	ix
List of Tables	xiv
Abbreviations	xvi
Symbols and Notations	xix
1 INTRODUCTION	1
1.1 Objective	1
1.2 Motivation	2
1.3 Background	3
2 PROPOSED FRAMWORK	
2.1 Video-Stabilization	
2.2 Tele-Operated Systems	
2.3 Integrated model of tele-operated systems	
3 TECHNICAL SPECIFICATION	3
3.1 PYTHON	
3.2 OPENCV	
4 EXPERIMENTAL ANALYSIS AND DESCRIPTION	
4.1 Stage 1: Feature Extraction	
4.1.1 About feature extraction	
4.1.2 The Shi -Tomasi Corner Detector	
4.2 Inter-Frame motion Estimation	

4.2.1 optical flow

4.2.2 optical flow detection

4.2.3 pyramidal implementation of Lucas Kanade method

4.3 Stage 3: Motion Intention Estimation And Correction

4.3.1 Affine Transformation

4.3.2 Kalman filter

5 FLOW CHART DEMONSTRATING THE FULL PROJECT WORKING

6 RESULTS

7 CONCLUSION & SUMMARY

8 FUTURE SCOPE

9 REFERENCES

APPENDIX A

List of Figures

Figure No.	Title	Page No.
1	Example Image showing video stabilisation	18
2	Integrated teleoperated model	19
3	Shi Tomasi model demonstration	26

List of Tables

Table No.	Title	Page No.
1	Table showing algorithm used in the small window to detect corners	24
2	Movement of features being tracked in video 1	37
3	Movement of features being tracked in video 2	38
4	Movement of features being tracked in video 3	39
5	Output for video 1	40
6	Output for video 2	41
7	Output for video 3	42
8	Output for video 4	43

List of Abbreviations

MAV

Micro Air Vehicles

AWGN

Additive White Gaussian Noise

DoG

Difference of Gaussian

1. INTRODUCTION

Shooting videos can be exhilarating and challenging all at the same time. We're focused on capturing a special moment, a family event or an extreme sport, and in the midst of all the movement and excitement, there is often shaking or vibration in your final recorded footage. However, video feeds suffer from undesired motion generated from multiple reasons such as mechanical vibration, or atmospheric turbulence which leads to like annoying rotations and translations due to aerodynamic characteristics. The shaky motion in the video hinders the fundamental intention of recording, and noise appears in the sequence of images. The degeneration increases computational cost and bit rate, leading to overloading video storage and transmission. Video stabilization essentially takes the unwanted jitters and blurriness that are captured in an action-packed video and provides a cleaner, smoother end result. In addition, we can choose from several different video stabilization options to get the job done.

The emerging branch of aerial robotics, that uses tele operated systems for their indoor navigation capabilities is gaining ground fast. A common problem of on-board video quality is the effect of undesired movements, so different approaches solve it with both mechanical stabilizers or video stabilizer software. Even though video stabilisation has come a long way with modern technology, stabilising video feeds in tele operated systems is still an issue. These issues arise from the very nature of these systems. These systems are used for their flight manoeuvrability in closed spaces, time and cost of manufacturing and maintenance, and safety for application in human bot interaction. Their performance is dependent on the input data of on-board sensors and cameras. Standard stabilization and tracking methods may be less effective on video clips from such systems. The aim is to aligning video frames which have undergone misalignment due to annoying hand motions and/or platform vibrations. Optimizing video stabilization is of paramount importance since it can give rise to issues like memory management and computational cost. The proposed method is robust to the frame content and handles multiple moving objects in the scene. It is to discriminate between intentional and undesired movements, and correct the unintentional motion.

1.1 OBJECTIVE

Many current video stabilisation algorithms are programmed to post process after the entire feed is given to them. There are many robust and effective algorithms for such cases but in the case of

live systems a more frame by frame stabilisation approach is required. We have developed an easy to understand video stabilisation which will provide good results for live systems and smoothen the feed frame by frame.

Our program is written in Python3. We have used the OpenCV library primarily for image processing.

1.2 Literature Review: (Background)

Until now we have selected a total of seventeen research papers as our references. These papers try to tackle the same issues we are trying to solve. The aforementioned papers are listed below:

- 1.) In 2015, Tiezheng Ma et al⁽¹⁾ demonstrated a technique for video stabilization, which is composed of three stages: feature trajectory extraction, trajectory smoothing, and frame warping. Most previous approaches view them as three separate stages. This paper proposes a method combining the last two stages, namely the trajectory smoothing and frame warping stages, into a single optimization framework. The novelty exists in the way of how we combine them: the trajectory smoothing part plays a major role while the frame warping part plays an auxiliary role. This kind of design conveniently increases the strength of the trajectory smoothing part by a robust first-order derivative term, which makes it possible to produce very aggressive stabilization effects. On the other hand, adaptive weighting mechanisms in the frame warping part, to follow the smoothed trajectories as much as possible while regularizing other places as similar as possible. The method is robust to utilize both foreground and background features, and very short trajectories. The utilization of all these information in turn increases the accuracy of the proposed method. There is also a simplified implementation of our method, which is less accurate but more efficient. Experiments on various kinds of videos demonstrate the effectiveness of our method.
- 2.) In 2018, Cong Liu et al. ⁽²⁾ a new method based on Kalman filter and homography transformation is proposed to stabilize the unstable video. Firstly, the SURF (Speed-Up Robust Feature) point-feature matching algorithm is employed to find the corresponding matching points between two consecutive frames, and the bidirectional nearest neighbour

distance ratio method is used to clear false matches. Secondly, motion estimation is computed by homography model and least square method. Then, Kalman filter are applied to separate the global and local motion. Finally, the unstable video frames is compensated by global motion vector. The experiment result shows that proposed method can effectively eliminate the random jitter.

3.) In 2017, Anli Lim et. al ⁽³⁾ describe the development of a novel algorithm to tackle the problem of real-time video stabilization for unmanned aerial vehicles (UAVs). There are two main components in the algorithm: (1) By designing a suitable model for the global motion of UAV, the proposed algorithm avoids the necessity of estimating the most general motion model, projective transformation, and considers simpler motion models, such as rigid transformation and similarity transformation; (2) to achieve a high processing speed, optical flow-based tracking is employed in lieu of conventional tracking and matching methods used by state-of-the-art algorithms. These two new ideas resulted in a real-time stabilization algorithm, developed over two phases. Stage I considers processing the whole sequence of frames in the video while achieving an average processing speed of 50 fps on several publicly available benchmark videos. Next, Stage II undertakes the task of real-time video stabilization using a multi-threading implementation of the algorithm designed in Stage I.

4.) In 2009, Ken-Yi et. al ⁽⁴⁾ propose a new approach for video stabilization. Most existing video stabilization methods adopt a framework of three steps, motion estimation, motion compensation and image composition. Camera motion is often estimated based on pairwise registration between frames. Thus, these methods often assume static scenes or distant backgrounds. Furthermore, for scenes with moving objects, robust methods are required for finding the dominant motion. Such assumptions and judgements could lead to errors in motion parameters. Errors are compounded by motion compensation which smoothes motion parameters. This paper proposes a method to directly stabilize a video without explicitly estimating camera motion, thus assuming neither motion models nor dominant motion. The method first extracts robust feature trajectories from the input video. Optimization is then performed to find a set of transformations to smooth out these

trajectories and stabilize the video. In addition, the optimization also considers quality of the stabilized video and selects a video with not only smooth camera motion but also less unfilled area after stabilization. Experiments show that our method can deal with complicated videos containing near, large and multiple moving objects.

- 5.) In 2016, Wilbert Aguilar et. al discuss the emerging branch of Micro Aerial Vehicles (MAVs) has attracted a great interest for their indoor navigation capabilities, but they require a high quality video for tele-operated or autonomous tasks. A common problem of on-board video quality is the effect of undesired movements, so different approaches solve it with both mechanical stabilizers or video stabilizer software. Very few video stabilizer algorithms in the literature can be applied in real-time but they do not discriminate at all between intentional movements of the tele-operator and undesired ones. In this paper, a novel technique is introduced for real-time video stabilization with low computational cost, without generating false movements or decreasing the performance of the stabilized video sequence. proposal uses a combination of geometric transformations and outliers rejection to obtain a robust inter-frame motion estimation, and a Kalman filter based on an ANN learned model of the MAV that includes the control action for motion intention estimation.
- 6.) In 2011, Matthias Grundmann present a novel algorithm for automatically applying constrainable, L1-optimal camera paths to generate stabilized videos by removing undesired motions. Our goal is to compute camera paths that are composed of constant, linear and parabolic segments mimicking the camera motions employed by professional cinematographers. To this end, our algorithm is based on a linear programming framework to minimize the first, second, and third derivatives of the resulting camera path. Our method allows for video stabilization beyond the conventional filtering of camera paths that only suppresses high frequency jitter. We incorporate additional constraints on the path of the camera directly in our algorithm, allowing for stabilized and retargeted videos. Our approach accomplishes this without the need of user interaction or costly 3D reconstruction of the scene, and works as a post-process for videos from any camera or from an online source.

- 7.) In 2018, Mu and Li ⁽²²⁾ demonstrated how Shi Tomasi implementation is used in image processing. It includes filtering, conversion to grayscale and edge detection. Then, PPHT is used and combined with Shi Tomasi algorithm for more refined corner detection. The traditional Shi Tomasi algorithm is affected by conditions such as illumination change. This method is shown to have decreased misdetection which is better than the traditional Shi Tomasi algorithm. It shows how Shi Tomasi is a stable algorithm, which along with Hough transform will improve the tracking of corner points. This paper proposes the use of Hough computation for auxiliary detection and assists Shi Tomasi algorithm to detect the strongest corners.
- 8.) In 2009, Siong et al. ⁽²³⁾ showed the fundamental role of optical flow in estimating motion of objects from a sequence of images is described. The authors focused on difference between standalone Lucas Kanade algorithm and the effect when it is combined with number of iterations and smoothing from Hora Schmuck algorithm and filtering is shown. Comparison is made based on optical flow pattern, segmentation of the motion of the images and processing time. With filtering, processing time reduces and results in a better optical flow. Filtering removes noise and unwanted optical flow around the interested significant motion. In Lucas Kanade algorithm, optimized window size is 5 while for enhanced algorithm, the smoothing will be 0.001 with iteration 10.
- 9.) In 2018, Wang et al.'s ⁽²⁴⁾ study to better understand target detection and tracking, pyramidal Lucas-Kanade optical flow method is used to detect and track moving targets. First, corners are detected, which is easy to track. Then tracking accuracy is improved as sub-pixel corner is calculated and then the video in each frame of the image layered in the image pyramid to calculate the optical flow at the top corner, use the next pyramid as the starting point of the pyramid and this process is repeated. In this method, the optical flow method does not make assumptions even if the background of the car changes. Unlike traditional optical flow methods, it is shown how this method overcomes shortcomings and better detects higher speed targets. However, the computation is relatively large since the algorithm needs to compute the sub-pixel corner and calculate the pyramid level one more time.

- 10.)** In 2011, Okade and Biswas⁽²⁵⁾ this paper proposes the idea of deblurring for efficient feature matching in the context of video stabilization. It explores the effect of motion blur on feature extraction as opposed to deblurring as a post processing step. It shows how pre-processing deblurring helps in increasing feature matching accuracy, hence improving the video stabilizing performance. The approach shown isn't computationally expensive as treatment of blur is done only on those frames which show significant amount of motion blur.
- 11.)** In 2018, Huang et al. ⁽²⁶⁾ address the problem of shakiness induced by camera movement, which not only incurs a terrible experience when viewing the video content, but also increases the computational cost and bit rate in video coding. This degeneration overloads video storage and transmission. By reusing the stabilized motion of feature points and geometric transformations, a better predictor can be established to replace the original motion vectors in the motion estimation stage of video coding. These motion vectors are optimized based on statistics of the residuals between the stabilization predictor and the standard one to improve the efficiency of motion search.
- 12.)** In 2018, Cohen et al. ⁽²⁷⁾ demonstrate how standard video stabilization and tracking methods are less effective on video clips, which are captured on body worn cameras. Use-case categorization has been applied enabling different requirements of scenarios. The method performs foreground-background separation using joint estimation and filtering of the camera path, can handle multiple moving objects in the scene. Instead of a one size fits all approach, this approach groups video clips into a number of categories and treats each one separately. The author distinguishes between four typical scenarios of law enforcers that use body worn camera.
- 13.)** In 2018, Juranek et al. ⁽²⁸⁾ work on finding a way of using a low-pass spatial filter instead of window function of Shi–Tomasi corner detector, which is an enhancement of the Harris corner detector. The paper thus includes verification of the validity of this method on a reference image, and a comparison of different low-pass filters on test images.

Particular spatial filters are linear smoothing filters mean filtering and Gaussian blur, and non-linear median filtering. The reason for this is to get an easy-to-implement algorithm for different architectures, such as a graphics card using shaders, which will allow fast processing of the input image even in real-time applications on less powerful devices, for example smartphones.

14.) In 2012, Zhao et al.⁽²⁹⁾ demonstrated how in order to locate and track eyes accurately, in this paper, we used the Adaboost algorithm to detect face, then located the eyes in the first frame using the Template Matching algorithm, finally combined the Pyramid algorithm and Lucas-Kanade optical flow algorithm to track eyes. The combination of the two algorithms results in dramatically improving the computing speed. It can be clearly understood that in normal lighting condition with a relatively small head movement, the algorithm tracks eyes well, and the recognition rate is also satisfying. It solves the delay problem of template matching method and improves the accuracy. However, the tracking performance is not so good in low light or rapid head movement.

15.) In 2017, Prawira et al.⁽³⁰⁾ his research proposes the use of Harris Corner Detector and Lucas-Kanade Tracker methods for the detection of 3D objects based on stereo image. This research is the early step in the development of the ability of a computer vision to be able to mimic the performance of eye organs in humans in detecting an object. The Effectiveness of the detection result of the proposed method is measured using the recall and precision parameter values obtained in the merged of the image. The proposed method gives a Recall value above 90% and a precision value above 50% for a distance of the cameras 10cm and 20cm for ball and tube objects. In the box object, the Recall value is 60% for a distance between the cameras 50cm and below 25% for a distance of the cameras 10cm and 20cm. The precision value for detection of the box object is very low, i.e. less than 25%.

16.) In 2012, Ejaz et al.⁽³¹⁾ published a paper which presented a technique for removing the unwanted jitter from the videos. The experimental results indicated that the proposed methodology significantly improved the quality of the videos. A particular requirement was to stabilize those videos which had unwanted jitter effects because of the hand motion

of the person making the video. The recorded video were then expected to have a lot of jitters because of the hurdles in the path. The most important issue in the design of video stabilization systems was differentiating between intentional and unintentional motion of the camera. Moreover, it is important to reconstruct the effected frames to compensate the loss of details because of jitter. In this paper, they proposed a practical approach for video stabilization which removed jitter from the videos. The scheme is based on the detection of curvature points in a curve of collective motion estimates.

- 17.) In 2016, Okade et al.⁽³²⁾ published a paper which investigated a novel learning based camera motion characterization scheme along with its application to the video stabilization problem. The proposed scheme worked at the frame level by classifying the inter-frame camera motion patterns which was extended to classify the video segments and a novel application to video stabilization was investigated. Experimental analysis on a number of test sequences captured using a hand held video camera showed that by characterizing the smooth and jittery motions, selective video stabilization could be carried out only on those video segments which had been degraded. This approach of selective video stabilization saved considerable amount of computational time as compared to running the stabilization algorithm on the entire video sequence as done currently in literature. The proposed strategy of using a classification scheme prior to applying the video stabilization routine offered a new paradigm to the conventional video stabilization . A novel camera motion characterization scheme was proposed by transforming the block motion vectors into a representation scheme using the polar angle and magnitude histograms. The proposed scheme works at the frame level by classifying the inter-frame camera motion patterns which was extended to classify the smooth and jittery video segments and a novel application to video stabilization was explored.

2. PROPOSED FRAMEWORK

2.1 Video Stabilization

Video stabilization refers to algorithms used to improve video quality by removing unwanted camera shakes and jitters due to hand jiggling and unintentional camera panning. ^[17]

Video stabilization technology is used to avoid visual quality loss by reducing unwanted shakes and jitters of an image/video capturing device without influencing moving objects or intentional camera panning . This is particularly essential in handheld imaging devices, which are more affected by shakes due to their smaller size. Unstable images are typically caused by undesired hand jiggling and intentional camera panning, whereas unwanted position fluctuations of camera result in unstable image sequences. Using video stabilization techniques ensures high visual quality and stable video footages even in nonoptimal conditions. Ideally, imaging devices are equipped with mechanical means, which physically avoid camera shakes etc. ^[17]



BEFORE STABILIZATION

AFTER STABILIZATION

Fig. 1. An image showing example of video stabilization ^[18]

2.2 TELEOPERATED SYSTEMS

Teleoperation (or remote operation) indicates operation of a system or machine at a distance. It is similar in meaning to the phrase "remote control" but is usually encountered in research, academic and technical environments. It is most commonly associated with robotics and mobile robots but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance. ^[19]

Similarly, in teleoperator systems, the human operator is connected by means of such displays and controls to a telerobot that can sense, travel through, and manipulate the real world. ^[21]

The term teleoperation is in use in research and technical communities as a standard term for referring to operation at a distance. This is as opposed to telepresence which is a less standard term and might refer to a whole range of existence or interaction that include a remote connotation. ^[19]

Teleoperation had become one of the most prolific areas of research due to the great number of applications in which the remote operation paradigm can be applied. Teleoperation is made from several disciplines, areas or techniques, ranging from communications to systems control and from virtual reality to digital signal processing. Hence, individual advancements in these areas could be applied to developments in teleoperation systems. ^[20]

2.3 INTEGRATED MODEL OF TELE-OPERATED SYSTEMS

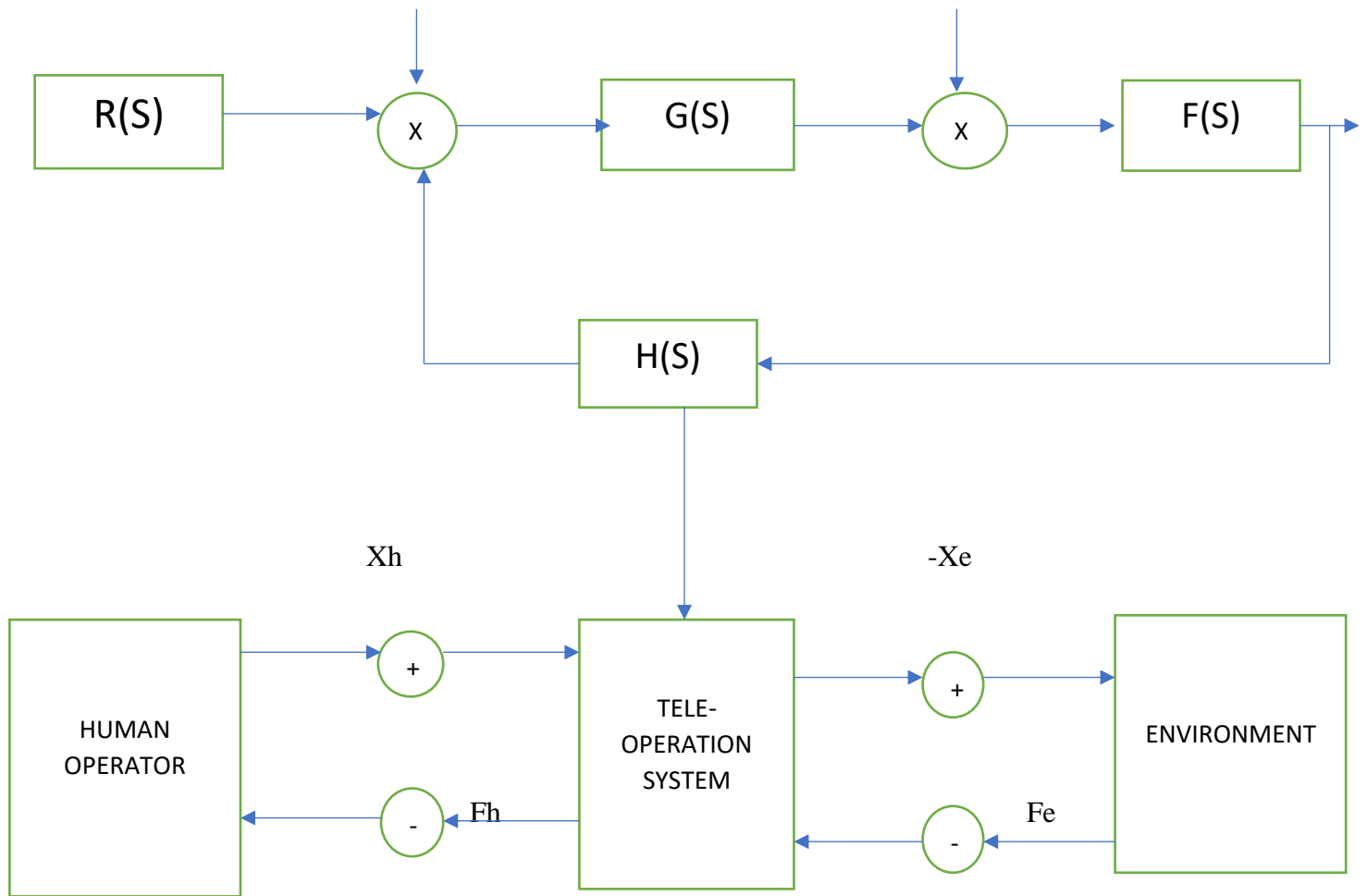


Fig. 2. A figure showing the integrated model of tele-operated system

Even though video stabilisation has come a long way with modern technology stabilising video feeds in tele operated systems is still an issue. These issues arise from the very nature of these systems.

Issues like memory management, computational cost, camera quality etc cause major issues when stabilising these feeds.

In our project we have used algorithms that try to minimise computational cost at each phase of stabilisation. Our main objective is to get the best quality possible at a low computational cost as tele operated systems by design have less computational power and memory.

Our program is written in Python3. We have used the OpenCV library primarily for image processing.

We have attempted to keep computational cost as low as possible in our project.

Video stabilisation is used to improve video quality in many fields. As our advances in technology have improved, a lot of sophisticated algorithms and techniques have been devised to stabilise video feeds.

A lot of these algorithms are designed to work on high end systems which have immense computational power. This is something that can be overlooked in many applications but when it comes to some fields this becomes a major limitation.

In case of tele operated systems like MAVs and mini-drones computer vision plays a major role. Since the memory and computational power on board is limited due to the size it is very hard for sophisticated algorithms to work.

Keeping such systems in mind we have used algorithms in our project that try to minimise computational cost at each phase of stabilisation. All video stabilisation follows similar frameworks in general.

Our goal is to get the best quality possible at a low computational cost as tele operated systems by design have less computational power and memory.

3. TECHNICAL SPECIFICATIONS

3.1 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. ^[9]

Advantages of using Python over other languages

- Easy to read and easy to learn. It is easier to write a program in Python than in C or C++. We gain the possibility to think clearly while coding, which also makes the code easier to sustain.
- Which reduces the cost for maintenance of the program and seen as one of Python programming advantages.

- No bug can originate a segmentation fault.
- Wide applicability, and is extensively used by scientists, engineers, and mathematicians.
- The language includes a large library with memory management which is another one of the advantages of Python programming.
- Python is better for automating build systems, collecting test data, server-side applications.

[10]

3.2 OPEN CV

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection^[11]

Features of OpenCV Library-

Using OpenCV library, you can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it. ^[12]

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSX, FreeBSD, Net BSD, Open BSD, etc^[11]

4. EXPERIMENTAL ANALYSIS AND DESCRIPTION

Image stabilisation algorithms are divided into 3 stages.

4.1 Stage 1: Feature Extraction

4.1.1 About Feature extraction

In machine learning, pattern recognition and in image processing, **feature extraction** starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction. When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. Before we begin feature extraction, we have to convert our image into a grayscale image. ^[13]

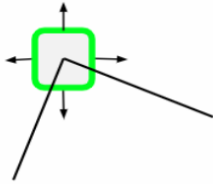
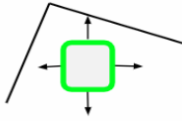
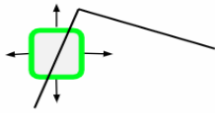
4.1.2 The Shi-Tomasi Corner Detector

Shi-Tomasi Corner Detection was published by J.Shi and C.Tomasi in their paper '*Good Features to Track*' in 1994. This was an improvement on the original Harris corner detection algorithm. Here the basic intuition is that corners can be detected by looking for significant change in all direction.

For a simple explanation, we consider a small window on the image then scan the whole image, looking for corners.

Shifting this small window in any direction would result in a large change in appearance, if that particular window happens to be located on a corner.

Table 1: The following table shows how the algorithm uses this small window to detect corners [8]

<i>S.No.</i>	<i>Case</i>	<i>Representation</i>	<i>Interpretation made by the algorithm</i>	<i>Action taken by the algorithm</i>
1.	Corner		A corner will have major changes in all directions	A corner is detected
2.	Flat region		Flat regions will have no change in any direction	No output as no corner detected
3.	Edge		In this case there will be no major change along the edge direction	No output as corner is detected

We know that the Shi-Tomasi algorithm is based on **Harris** corner detector. The change Shi and Tomasi made was in the scoring function of the algorithm.

The score for this Harris corner detector is calculated like this (R is the score):

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

λ_1 and λ_2 are the eigen values of the matrix we are observing at the moment while scanning the image for corners. Also, k is a constant.

Instead of the above-mentioned method Shi and Tomasi proposed;

$$R = \min(\lambda_1, \lambda_2)$$

Here λ_1 and λ_2 are the eigen values of the matrix we are observing at the moment while scanning the image for corners.

If it is greater than a threshold value, it is considered as a corner. ^[14]

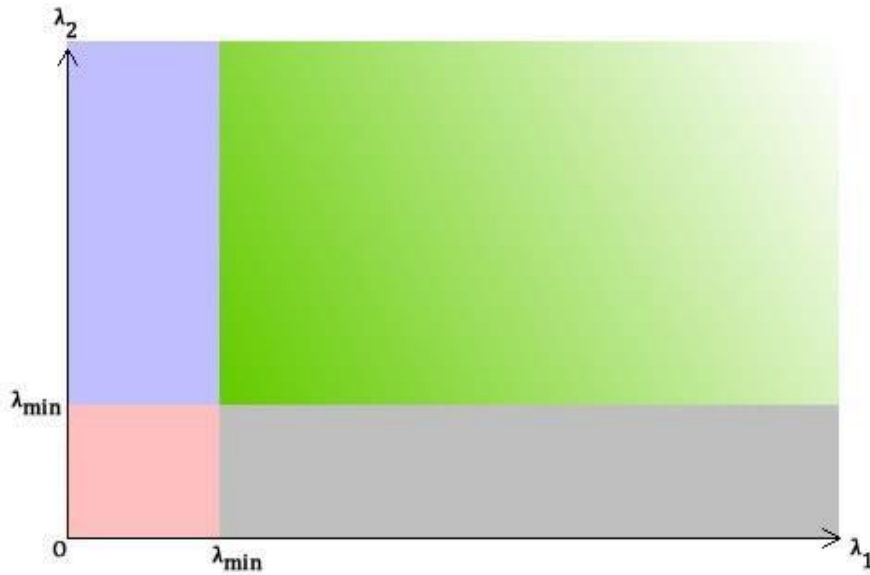


Fig. 4. Plot of R in $(\lambda_1 - \lambda_2)$ space [6]

In figure 4 we have represented the plot of R in $(\lambda_1 - \lambda_2)$ space;

- Green: both λ_1 and λ_2 are greater than a certain value. Thus, this region is for pixels "accepted" as corners.
- In the blue and gray regions, either λ_1 or λ_2 is less than the required minimum.
- In the red region, both λ_1 and λ_2 are less than the required minimum. Compare the above with a similar graph for Harris corner detector. We will see the blue and gray areas are equivalent to the "edge" areas. The red region is for "flat" areas. The green is for corners. [6]

OpenCV implements the Shi-Tomasi corner detector by `goodFeaturesToTrack()` function. We have used shi tomasi because it very convenient for our purpose. It makes feature tracking in between frames really easy. Another advantage is that this method is much more efficient than other open source methods. We tried using different methods which are listed below.

4.2 Stage 2: Inter-Frame Motion Estimation

4.2.1 OPTICAL Flow

Optical flow or optic flow is the pattern of apparent_motion_of objects, surfaces, and edges in a visual scene caused by the_relative motion_between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image. The concept of optical flow was introduced by the American psychologist_James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world. Gibson stressed the importance of optic flow for affordance perception, the ability to discern possibilities for action within the environment. Followers of Gibson and his_ecological approach to psychology_have further demonstrated the role of the optical flow stimulus for the perception of movement by the observer in the world; perception of the shape, distance and movement of objects in the world; and the control of locomotion.

The term optical flow is also used by roboticists, encompassing related techniques from image processing and control of navigation including motion detection,_object segmentation, time-to contact information, focus of expansion calculations, luminance, motion compensated encoding, and stereo disparity measurement. [15]

4.2.2 Optical Flow Detection

There are different approaches used to estimate the parameters relating two consecutive frames, we have used optical flow model.

In this step feature points are detected and described for being used instead of all pixels from each image. We have used the Lucas and Kanade method in our program. This step just calculates the movement of the features in between consecutive features.

4.2.3. Pyramidal Implementation Of Lucas Kanade Method

In computer vision, the **Lucas–Kanade method** is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade in 1984. Since the time it was proposed it has been used and implemented in many different ways in modern image processing. [2]

The pyramidal implementation was proposed by J. Y. Bouguet in 2001.

This was a major improvement on the original model as the standard Lucas-Kanade method can only deal with small pixel displacement. To work around this major limitation pyramidal implementation is used. [2]

Let I and J be the 1st and 2nd grayscale images respectively. Now let I(x,y) be gray value of I at

$[x \ y]^T$. Now, we can represent a point in the first image as $u = [u_x \ u_y]^T$.

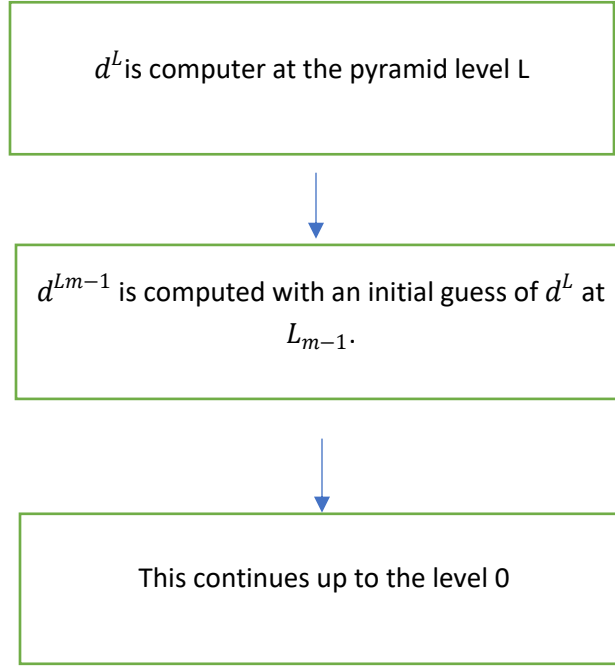
The goal is to find a similar point v on J, where I(u) and J(v) are similar. In mathematical terms $v = u + d = [u_x + d_x \ u_y + d_y]^T$. Here ‘d’ is the image velocity or optical flow at ‘u’. [2]

Let $I^0 = I$ be the 0th level image. From here the pyramid representation is built recursively in the following manner,

$$I^0 \rightarrow I^1 \rightarrow I^2 \rightarrow I^3 \rightarrow I^{Lm} \dots (L: 2 \sim 4)$$

This will continue till L levels. So, I^{L-1} is the image at level L-1 and I^L is the image at level L.

The overall working of the pyramidal tracking algorithm is shown below ^[5]



To elaborate and explain the above flowchart, let $g^L = [g^L_x g^L_y]^T$ be an initial guess at level L. This initial guess is available from level L_m to level L+1. The residual pixel displacement vector that is used to minimise the image matching error function at each level is mathematically written as $d^L = [d^L_x d^L_y]^T$. ^[2]

The error function ε^L is mathematically written as;

$$\varepsilon^L(d^L) = \varepsilon^L([d^L_x, d^L_y]) = \sum_{y=u^L_x-w_x}^{u^L_x+w_x} \sum_{y=u^L_y-w_y}^{u^L_y+w_y} (I^L(x, y) - J^L(x + g^L_x + d^L_x, y + g^L_y + d^L_y))^2$$

The initial guess g^L is used to pre-translate the image patch in 2nd image J. The residual pixel displacement is small and therefore very easy to compute using Lukas-Kanade algorithm.

Once d^L is computed the new initial guess at level L-1 is,

$$g^{L-1} = 2(g^L + d^L)$$

The initial guess for the deepest level L_m in this iteration is

$$g^{L_m} = [0 \ 0]^T$$

The final optical flow computed after this in this iteration is given by

$$d = \sum_{L=0}^{Lm} 2^L d^L$$

This process is repeated iteratively until level 0. ^[2]

4.3 Stage 3: Correction Phase

In this step the program has to smoothen the feed. Jitters and disturbances can be caused by external factors and the program must be capable of correcting this.

We have used affine transformations to analyse the position of features in between frames.

4.3.1 Affine Transformation

It is any transformation that can be expressed in the form of a *matrix multiplication* (linear transformation) followed by a *vector addition* (translation). ^[1]

Affine Transformation are used to express:

- a. Rotations (linear transformation)
- b. Translations (vector addition)
- c. Scale operations (linear transformation) ^[1]

In essence, an Affine Transformation represents a **relation** between two images.

The usual way to represent an Affine Transform is by using a 2 X 3 matrix.

Consider two matrices A and B,

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}_{2 \times 2} \text{ and } \begin{bmatrix} b_{00} \\ b_{10} \end{bmatrix}_{2 \times 1}$$

Now,

$$M = [A \ B] = \begin{bmatrix} a_{00} & a_{01} & b_{00} \\ a_{10} & a_{11} & b_{10} \end{bmatrix}_{2 \times 3}$$

Considering that we want to transform a 2D vector $X = \begin{bmatrix} x \\ y \end{bmatrix}$ by using A and B, we can do it as follows;

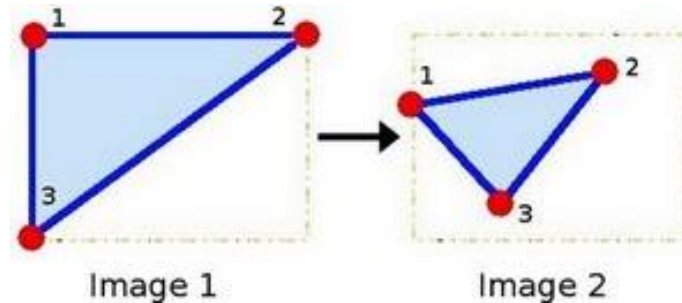
$$T = A \cdot \begin{bmatrix} x \\ y \end{bmatrix} + B \text{ or } T = M \cdot [x, y, 1]^T$$

$$T = \begin{bmatrix} a_{00}x & a_{01}y & b_{00} \\ a_{10}x & a_{11}y & b_{10} \end{bmatrix} [1]$$

The information about this relation can come, roughly, in two ways:

- We know both X and T and we also know that they are related. Then our job is to find M.
- We know M and X. To obtain T we only need to apply $T = M \cdot X$. Our information for M may be explicit (i.e. have the 2-by-3 matrix) or it can come as a geometric relation between points. ^[1]

To explain this using a figure we consider two images, image 1 and image 2. We can see the point 1, 2 and 3 are forming a triangle in image 1. These three points are mapped into image 2 still forming a triangle but their positions have changed significantly. If we find the affine transformation with these three points, we can apply this new found relation to all pixels in the image. ^[1]



In our project we have used affine transformations to form matrices which contain values that show the relation between consecutive frames. In order to obtain a stabilized video, it is assumed that the first frame is stable which is used as a reference frame to stabilize the next frame. The second frame is subsequently used as a reference frame to stabilize the next frame and the process continues until the last frame.

This is implemented by creating a new affine transform for the current frame with respect to the previous one which is done by the motion filtering model. This new matrix is the used to warp the current frame into the stabilised frame. Our motion filtering model works on the principles of Kalman Filter.

4.3.2 Kalman Filter

Kalman Filter is a linear recurrence filter that predicts the next state of the system based only on the previous state of the system, but not dependent on sequence of past states, so it can also be used for real time processing. In statistics and control theory, Kalman filtering, also called linear quadratic estimation (LQE), uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and generates estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe. The filter is named after Rudolf E. Kálmán, one of the primary developers of its theory. ^[16]

Implementation

Let X_K be the state of system at time k , A is the state transfer matrix which describes the transition relationship of the system between time, k and $k+1$ and W_K is the system noise.

$$X_{K+1} = AX_K + W_K$$

Z_K is the observation state of system state X_K at time k , H is the observation matrix, V_K is the observation noise at time k . The final observation state Z_K is described as below:

$$Z_K = HX_K + V_K$$

The specific procedure of Kalman Filter consists of two steps: prediction and correction state and the overall aim is to regain the value of X_K . Here Q is the covariance matrix of process noise, R is the covariance matrix of observation noise whose value is set to 10^{-5} , x_K is the estimated value of X_K and P_K is the estimated value of covariance of X_K .

1. The aim of prediction state is to predict the values of X_K and P_K .

$$x'_k = Ax_{K-1}$$

$$P'_k = AP_{K-1}A' + Q$$

Where x'_k and P'_k are the estimated values of X_K and P_K respectively. ^[3]

2. The aim of correction state is to use the value of current measurement, z_K to update the value of x'_k .

$$K_g = P_{K-1}H'(HP_{K-1}H' + R)^{-1}$$

$$x_k = x'_k + K_g(Z_K - Hx'_k)$$

$$P_k = (I - K_gH)P'_k$$

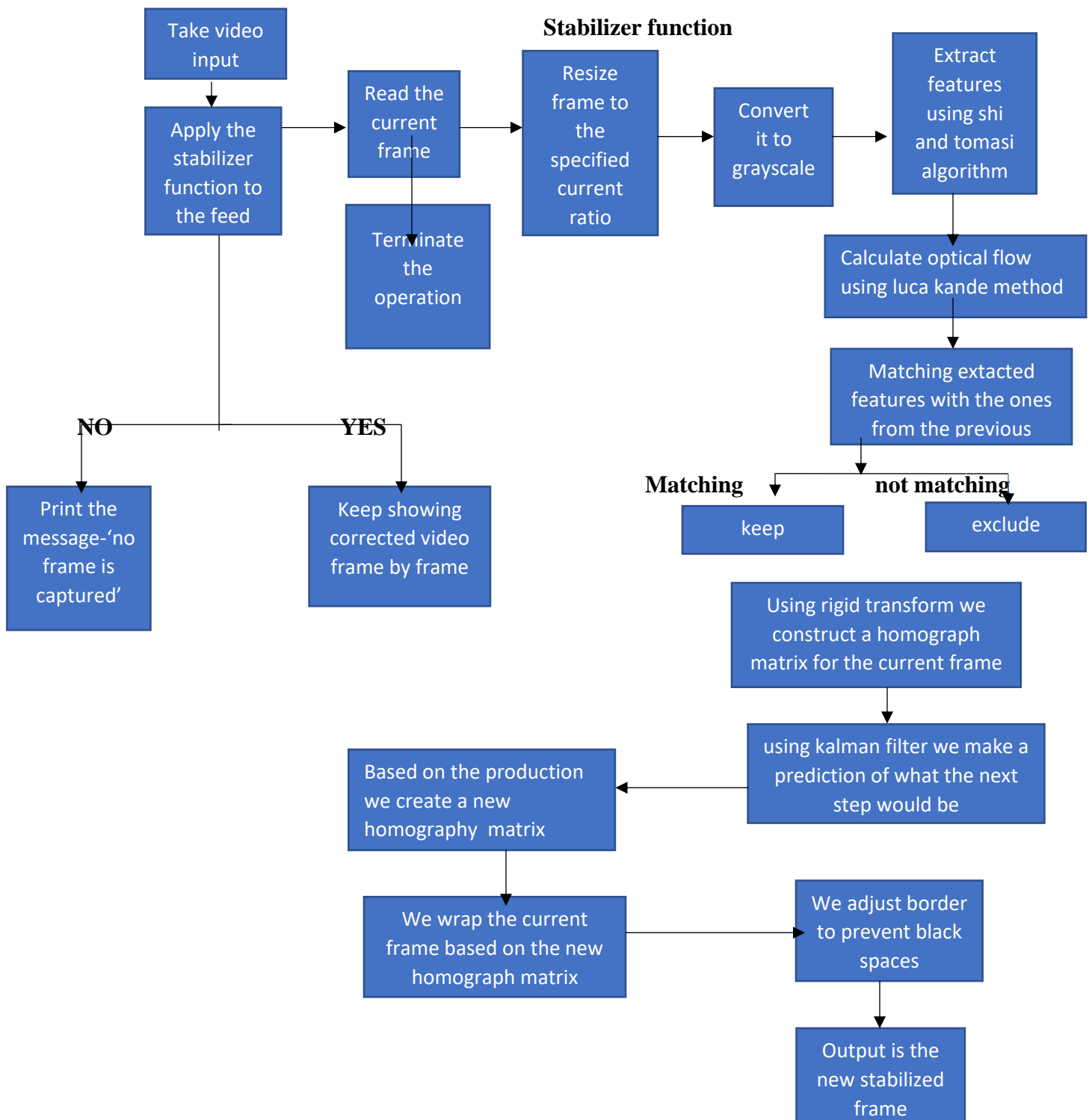
Where K_g is the Kalman Gain and I is the identity matrix. ^[3]

he initial value of x_0 and P_0 is selected at random because we can't get the initial information of the video sequence before we process it. The values are adjusted in the first few frames after which the Kalman Filter can operate properly. The output is a smoothened frame. ^[3]

5. FLOW CHART

Main script

Stabilizer function



6. RESULTS

6.1 Motion tracking output


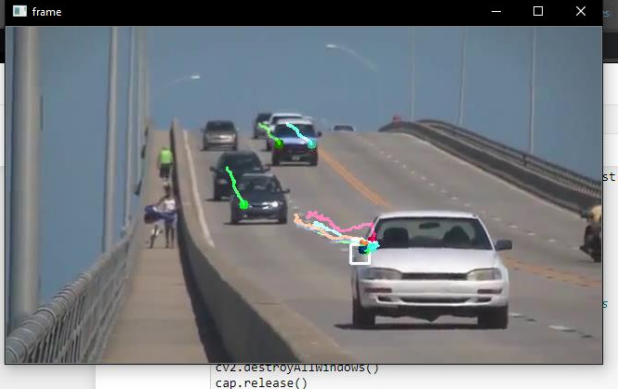

S.no	Output at different time stamps
1.	
2.	
3.	

Table 2: movement of features being tracked in video 1



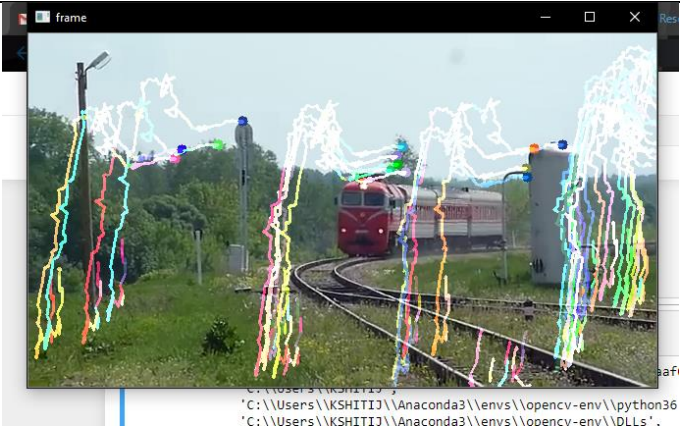
S.No	Output at different time stamps
1.	 <p>A screenshot of a video frame labeled 'frame' showing a train on a curved track. The scene is overlaid with numerous colorful lines (red, green, blue, yellow, magenta) representing tracked features. The lines are mostly vertical, indicating stationary or slowly moving features. The train is visible in the background on the right side of the track.</p>
2.	 <p>A screenshot of a video frame labeled 'frame' showing a train on a curved track. The scene is overlaid with numerous colorful lines representing tracked features. The lines are more varied in orientation compared to frame 1, showing some movement. The train is visible in the background on the right side of the track.</p>
3.	 <p>A screenshot of a video frame labeled 'frame' showing a train on a curved track. The scene is overlaid with numerous colorful lines representing tracked features. The lines are highly varied in orientation and length, indicating significant movement of the tracked features. The train is visible in the background on the right side of the track.</p>

Table 3: movement of features being tracked in video 2

S.No.	Output at different time stamps
1.	 <p>A video frame showing a concrete path between a grey wall on the left and a stone railing on the right. Numerous colorful lines (red, green, blue, yellow) are overlaid on the image, representing the movement of tracked features. The lines are mostly concentrated on the walls and railing, with some extending into the path.</p>
2.	 <p>A second video frame from the same sequence, showing the same concrete path. The feature tracking lines are more numerous and appear more chaotic, indicating more movement or a different set of features being tracked compared to the first frame.</p>
3.	 <p>A third video frame showing the same scene. The density of the colorful tracking lines is significantly higher than in the previous frames, covering most of the visible surfaces (walls, railing, and path), suggesting a more complex or dense set of tracked features.</p>

Table 4: movement of features being tracked in video 3

Note: Video 1 is an already stable video used to show smooth tracking of features as the feed itself is very stable. The other two videos however are very unstable which can be observed by the erratic movement of features. It is much easier to see how the tracking is being done on a stable feed.

6.1.1 Inference from Feature Tracking Outputs

For better understanding of the inner workings of the program we have shown how feature tracking is done below. The screenshots given below in the tables are of the algorithm working at different time stamps in increasing order. Hence, we can see features being track through frames.

In these stills of the outputs, the lines are depicting the movement through frames. We have used different colours to depict different features. Since we are using affine transformations, we can rotate entire frames just by observing the movement of some features. We don't have to detect each and every corner for our program to work, we just have to observe the movement of some features and based on these we can morph the entire frame based on Kalman filter.

Another thing to notice is that video 1 is very stable by itself and hence the tracking appears to be very smooth. It is strictly used to depict smooth tracking. Again, as mentioned above we don't have to detect all of the features in each frame. We are just tracking any 50 features (this can be changed by changing the argument in the program).

Videos 2 and 3 are also used to show the stabilized output and hence are very unstable and the movement of features is erratic. After this we use affine transform to find a relation between features in consecutive frames and use Kalman filter to smoothen the feed. That is shown below.

6.2. Output for stabilisation

Table 5: Output for video 1


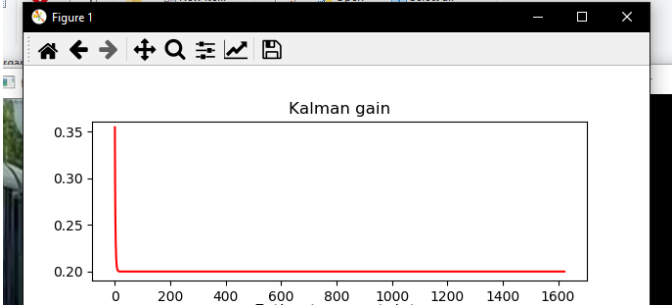
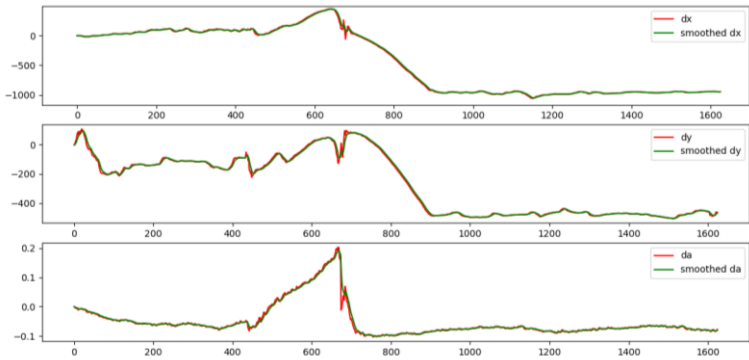
Snapshot of the stabilised output	
Graph for Kalman gain	
Trajectory of dx , dy and da	

Table 6: Output for video 2


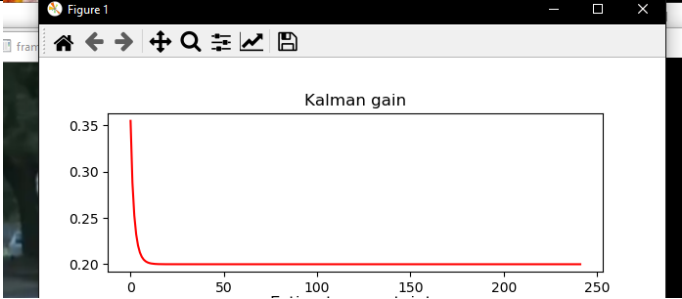
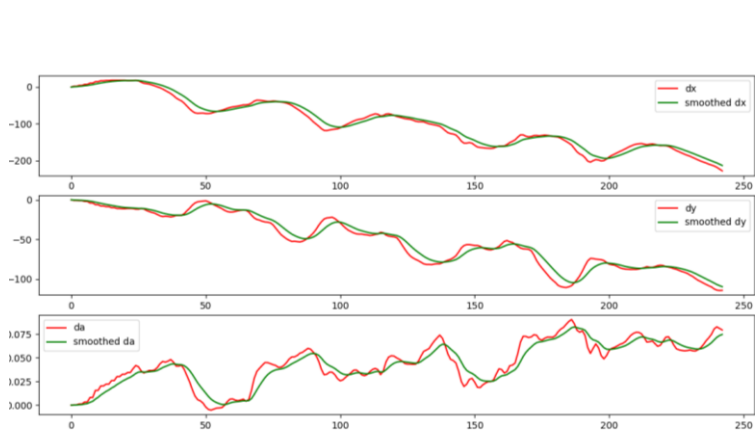
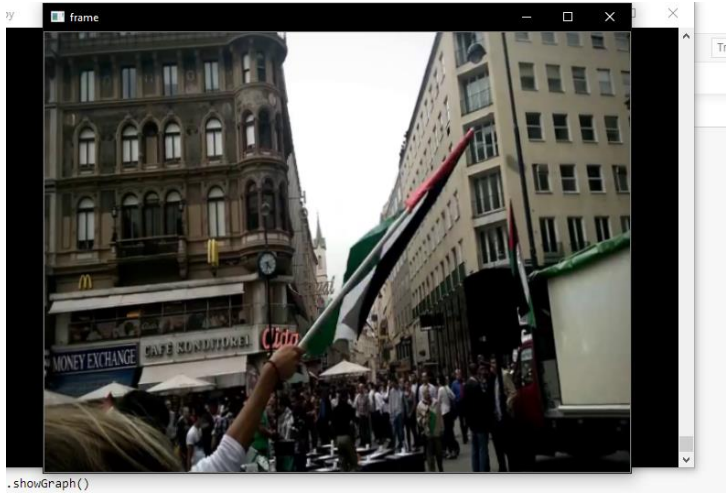
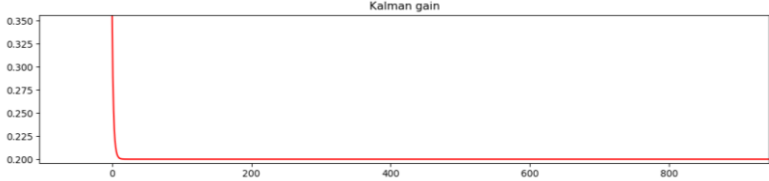
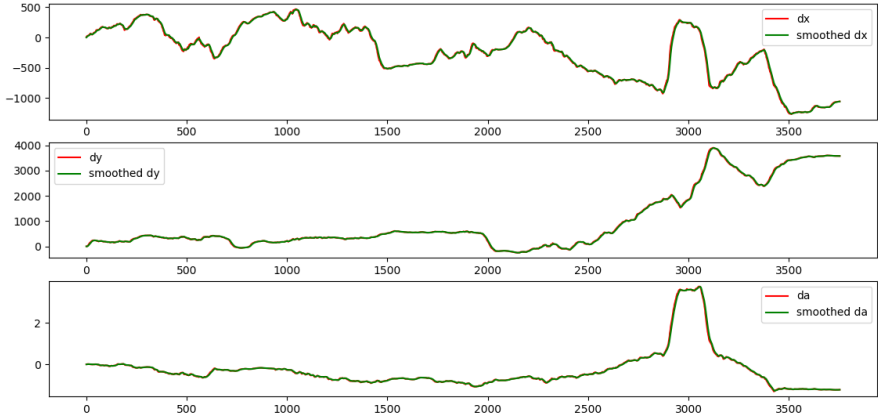
Snapshot of the stabilised output	
Graph for Kalman gain	
Trajectory of dx, dy and da	

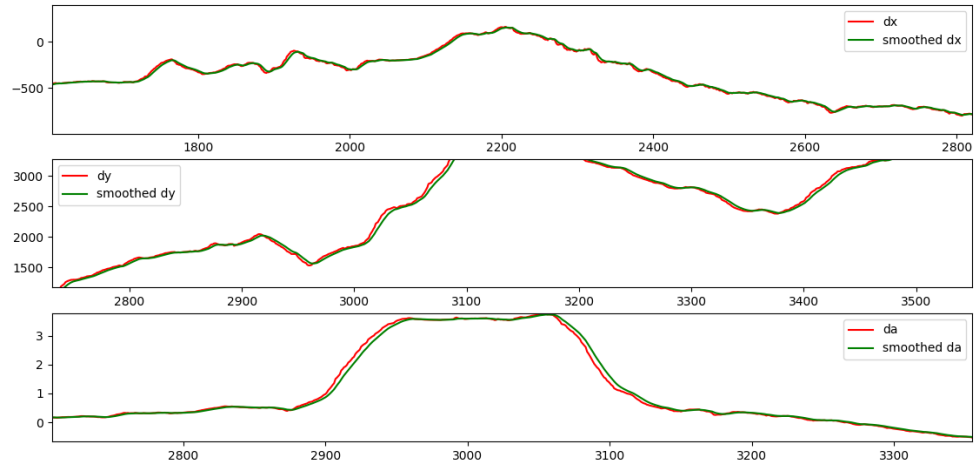
Table 7: Output for video 3

<p>Snapshot of the stabilised output</p>	
<p>Graph for Kalman gain</p>	
<p>Trajectory of dx, dy and da</p>	

Table 8: Output for video 4

<p>Snapshot of the stabilised output</p>	
<p>Graph for Kalman gain</p>	
<p>Trajectory of dx, dy and da (since the video is longer the graphs are zoomed in for better observation below)</p>	

Zoomed
in
trajectori
es of dx,
dy and da



Note: The output feed has a different resolution from the input video as we have scaled it by 10% in order to prevent black spaces appearing on the border due to rotation of frames. This rotation is caused by the morphing.

Note: All the output shown are done on videos. The program works on live feeds as well but it harder to show the results for a live feed. The graphs for these feeds show the correction much more clearly.

6.2.1 Scales for All Graphs

6.2.1.1 Kalman gain plot

Kalman gain vs frame no.: - x axis = 200, y axis = 0.05 (0.25 for the 4th case)

6.2.1.2 dx, dy and da plots for all cases

Video 1:

dx vs frame no. plot: - x axis = 200, y axis = 500

dy vs frame no. plot: - x axis = 200, y axis = 200

da vs frame no. plot: - x axis = 200, y axis = 0.1

Video 2:

dx vs frame no. plot: - x axis = 50, y axis = 100

dy vs frame no. plot: - x axis = 50, y axis = 50

da vs frame no. plot: - x axis = 50, y axis = 0.025

Video 3:

dx vs frame no. plot: - x axis = 100, y axis = 500

dy vs frame no. plot: - x axis = 100, y axis = 200

da vs frame no. plot: - x axis = 100, y axis = 0.1

Video 4:

dx vs frame no. plot: - x axis = 500, y axis = 500

dy vs frame no. plot: - x axis = 500, y axis = 200

da vs frame no. plot: - x axis = 500, y axis = 1

For the zoomed in plot in column 4;

dx vs frame no. plot: - x axis = 200, y axis = 500

dy vs frame no. plot: - x axis = 100, y axis = 500

da vs frame no. plot: - x axis = 100, y axis = 1

6.2.2 Inferences from Stabilization Outputs

6.2.2.1 Table 1

First row shows a screenshot of the stabilised output. The output has a different resolution of 640X480. The feed itself is taken at this resolution to make the scaling process easier.

The second column shows the graph for Kalman gain. The x axis here the number of frames and the y axis is the value of Kalman gain. This shows how the magnitude of Kalman gain changes as the feed progresses. A similar pattern can be observed in all the cases. Kalman gain changes a lot at the beginning and then reaches an equilibrium point as time progresses. This happens since we are using a nonlinear system.

The third row shows the graph for dx, dy and da. dx and dy are frame average movements in x and y axes respectively. da depicts the frame average angular movement. The x axis is the number of frames and y axis is dx for the first graph, dy for the second graph and da for the third graph.

The red line shows the trajectory of dx in the original feed and the green line shows the trajectory for the smoothened dx.

Similarly, the red line shows the trajectory of dy in the original feed and the green line shows the trajectory for the smoothened dy.

In the third graph again, the red line shows the trajectory of da in the original feed and the green line shows the trajectory for the smoothened da.

Very high degree of movement can be noticed between 600 to 800 frames in the original feed. The stabilised output shows a smoothened trajectory even in this case.

From observation we can see that in all the graphs the red lines are very erratic and unstable. These lines depict the nature of movement of the features in the original feed.

On the other hand, the green lines are stable and smooth. This shows that the movement of features in the stabilised frames is very smooth.

6.2.2.2 Table 2

Similar to table 1, the first row shows a screenshot of the stabilised output. The output has a different resolution of 640X480(the standard for our program).

Similar to the first case the second column here shows the graph for Kalman gain. The x axis here the number of frames and the y axis is the value of Kalman gain. This shows how the magnitude of Kalman gain changes as the feed progresses. A similar pattern can be observed in all the cases. Kalman gain changes a lot at the beginning and then reaches an equilibrium point as time progresses. This happens since we are using a nonlinear system.

The third row shows the graph for dx , dy and da . dx and dy are frame average movements in x and y axes respectively. da depicts the frame average angular movement. The x axis is the number of frames and y axis is dx for the first graph, dy for the second graph and da for the third graph.

The red line shows the trajectory of dx in the original feed and the green line shows the trajectory for the smoothened dx .

Similarly, the red line shows the trajectory of dy in the original feed and the green line shows the trajectory for the smoothened dy .

In the third graph again, the red line shows the trajectory of da in the original feed and the green line shows the trajectory for the smoothened da .

In the case of this video the trajectories of all three variables are very erratic but da specifically shows a lot of variation.

From observation we can see that again, in all the graphs the red lines are very erratic and unstable. These lines depict the nature of movement of the features in the original feed.

On the other hand, the green lines are stable and smooth. This shows that the movement of features in the stabilised frames is very smooth.

6.2.2.3 Table 3

Again, the first row shows a screenshot of the stabilised output. The output has a different resolution of 640X480.

Again, the second column shows the graph for Kalman gain. The x axis here the number of frames and the y axis is the value of Kalman gain. This shows how the magnitude of Kalman gain changes as the feed progresses. Again similar pattern is observed in this case. Kalman gain changes a lot at the beginning and then reaches an equilibrium point as time progresses. This happens since we are using a nonlinear system.

The third row shows the graph for dx , dy and da . Again, dx and dy are frame average movements in x and y axes respectively. da depicts the frame average angular movement. The x axis is the number of frames and y axis is dx for the first graph, dy for the second graph and da for the third graph.

The red line shows the trajectory of dx in the original feed and the green line shows the trajectory for the smoothened dx .

Similarly, the red line shows the trajectory of dy in the original feed and the green line shows the trajectory for the smoothened dy .

In the third graph, the red line shows the trajectory of da in the original feed and the green line shows the trajectory for the smoothened da .

In this case dy and da show much higher variation in comparison to dx .

From observation we can see that again, in all the cases our program can make the changes necessary to smoothen the feed. The red lines depict the nature of movement of the features in the original feed.

Again, the green lines are stable and smooth. This shows that the movement of features in the stabilised frames is very smooth.

6.2.2.4 Table 4

First row shows a screenshot of the stabilised output. The output has a resolution of 640X480 similar to other cases. This resolution makes it easier to scale the output to prevent black spaces caused by the continuous morphing and rotation of the frames done by the program.

Similar to all the other cases, the second column shows the graph for Kalman gain. The x axis here the number of frames and the y axis is the value of Kalman gain. This shows how the magnitude of Kalman gain changes as the feed progresses. Again, a similar pattern can be observed in this case. Kalman gain changes a lot at the beginning and then reaches an equilibrium point as time progresses. This happens since we are using a nonlinear system.

The third row shows the graph for dx , dy and da . similar to other cases, dx and dy are frame average movements in x and y axes respectively. da depicts the frame average angular movement. The x axis is the number of frames and y axis is dx for the first graph, dy for the second graph and da for the third graph.

The red line shows the trajectory of dx in the original feed and the green line shows the trajectory for the smoothened dx .

Similarly, the red line shows the trajectory of dy in the original feed and the green line shows the trajectory for the smoothened dy .

In the third graph, the red line shows the trajectory of da in the original feed and the green line shows the trajectory for the smoothened da .

This video is a bit longer in length and hence to really hard to observe as there's are a higher number of frames the graph has to depict in the same scale used to show other outputs.

To make it easier to observe the process we have included zoomed in the graphs in the 4th column of the table.

From observation we can see that again, in all the cases our program can make the changes necessary to smoothen the feed. The red lines depict the nature of movement of the features in the original feed.

Again, the green lines are stable and smooth. This shows that the movement of features in the stabilised frames is very smooth.

7. CONCLUSION

Our main objective in this project was to develop a video stabilisation program which was easy to understand and would provide good results.

Unlike many video stabilisation programs record the whole video feed and then process it to provide the stabilised output, our algorithm stabilises the feed frame by frame which means it can work in live feeds. While other algorithms are very robust and effective, they cannot be used for a live feed. Such live feeds are major parts of modern day teleoperated systems.

In our model we have used a combination of different methods which allow us to implement our algorithm with simplicity and efficiency. We have used the Shi-Tomasi model for extracting features from a frame. To detect and calculate optical flow we have used Lucas Kanade method.

This method is a simple and effective way to track the movement of features through consecutive frames. We have displayed exactly how Shi-Tomasi algorithm and Lucas-Kanade algorithm work together and allow us to track features in section 6.1 of this report. Using affine transformations, we are also able to display the frame average movement in x-axis (dx) and y-axis (dy). We are also able to show the angular frame average movement (da). We have used Kalman filter To smoothen our feed. Kalman filter corrects the dx, dy and da values according to it's predictions. We have compared the values of dx, dy and da for the input feeds and stabilised outputs in section 6.2. The graphs in the third column show this comparison in all cases.

Our algorithm is very simple to understand implement. One of the key aspect for us is that we can use this in teleoperated systems like MAVs (micro aerial vehicles) and UAVs (unmanned aerial

vehicles) as stabilisation is done frame by frame. The program works really well for rectifying small jitters and irregular camera movements but one of the issues that we can into is when the program encounters very high degrees of movement of features it can cause the program to fail.

8. FUTURE SCOPE

In our program we have used Kalman filter to smoothen the feed. Our algorithm is very effective in dealing with jitters and stutters but even when the feed is stable the algorithm keeps on smoothening the already smooth feed. This problem can be corrected by using an algorithm which can predict the nature of the motion. Once the algorithm has predicted the nature of motion the program can decide whether the feed needs to be smoothened. Such prediction algorithms have been a significant research topic in recent years as this step can minimize the load on the system specifically in the case of tele-operated systems. ^[22]

Another thing that we noticed from experimental results was that very drastic changes can cause problems. When the changes from frame to frame are very drastic there's a chance the program can run into an exception due to the nature of methods used. Research to rectify such drastic changes is being done right now. Solutions like using sigmoid function and inverse exponential functions are being developed as these allow the program to deal with very erratic changes in feed. The ability to handle such changes is essential for live systems as the code running into such exceptions can cause major problems. ^[23]

Although our results were really good with Kalman filter, we have used the basic Kalman filter operation for our project. From our own results we can see that the Kalman gain changes very quickly at the beginning but quickly saturates as time progresses. The gain would reach an equilibrium point even though the position of features keeps on changing. This can be insignificant in most cases but the smoothening can be improved by using extended Kalman filter which is shown to have better results for nonlinear change position. Another approach which is being used in many recent papers is using Kalman filter in conjunction with other filter such low pass filters to improve the output. ^[24]

9.REFERENCES

- [1] https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/warp_affine/warp_affine.html
- [2] <https://web.yonsei.ac.kr/jksuhr/articles/Kanade-Lucas-Tomasi%20Tracker.pdf>
- [3] Moving camera video stabilization based on Kalman filter and least squares fitting, Information Engineering School, Communication University of China, CUC, Beijing, China, 2011
- [4] Video stabilization using Kalman filter and phase correlation matching, Ohyun Kwon, Jeongho Shin, Joonki Paik, Chung-Anf University, 2005.
- [5] Pyramidal implementation of the lucas kanade feature tracker, J.-Y. Bouguet, Intel Corporation, Microprocessor Research Labs, 1999
- [6] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html#shi-tomasi
- [7] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html
- [8] <https://www.geeksforgeeks.org/python-corner-detection-with-shi-tomasi-corner-detection-method-using-opencv/>
- [9] <https://www.python.org/doc/essays/blurb/>
- [10] <https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/>
- [11] <https://opencv.org/about/>
- [12] https://www.tutorialspoint.com/opencv/opencv_overview.htm
- [13] https://en.wikipedia.org/wiki/Feature_extraction
- [14] Good features to track, Jianbo Shi, Carlo Tomasi, computer science department, Stanford University.

- [15] https://en.wikipedia.org/wiki/Optical_flow
- [16] https://en.wikipedia.org/wiki/Kalman_filter
- [17] https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-78414-4_76
- [18] <https://www.videostudiopro.com/en/pages/video-stabilization/>
- [19] <https://en.wikipedia.org/wiki/Teleoperation>
- [20] Jimenez Moreno, Róbinson; Espinoza Valcarcel, Fabio Andrés and Amaya Hurtado, Darío. Teleoperated systems: A Perspective on Telesurgery Applications. Rev. ing. biomed. [online]. 2013, vol.7, n.14, pp.30-41. ISSN 1909-9762
- [21] https://itlaw.wikia.org/wiki/Teleoperator_system
- [22] Real-Time Model-Based Video Stabilization for Microaerial Vehicles, Wilbert G. Aguilar, Cecilio Angulo, 2015.
- [23] Deep Online Video Stabilization Miao Wang, Member, IEEE, Guo-Ye Yang, Member, IEEE, Jin-Kun Lin, Member, IEEE, Ariel Shamir, Member, IEEE Computer Society, Shao-Ping Lu, Member, IEEE, and Shi-Min Hu, Member, IEEE, 2018.
- [24] A New Extension of the Kalman Filter to Nonlinear Systems, Simon J. Julier and Jeffrey K. Uhlmann, University of Oxford, 1997.
- [25] Mu Z, Li Z 2018. A Novel Shi-Tomasi Corner Detection Algorithm Based on Progressive Probabilistic Hough Transform. *Paper presented in 2018 Chinese Automation Congress (CAC)*, Xi'an, China, pp. 2918-2922. doi: 10.1109/CAC.2018.8623648.
- [26] Siong LY, Mokri SS, Hussain A, Ibrahim N, Mustafa MM 2009. Motion Detection Using Lucas Kanade Algorithm and Application Enhancement. *Paper presented in 2009 International Conference on Electrical Engineering and Informatics*, Selangor, pp. 537-542. doi: 10.1109/ICEEI.2009.5254757.
- [27] Wang Z, Yang X 2018. Moving Target Detection and Tracking Based on Pyramid Lucas-Kanade Optical Flow. *Paper presented in 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Chongqing, pp. 66-69. doi: 10.1109/ICIVC.2018.8492786.

- [28] Okade M, Biswas PK 2011. Improving Video Stabilization in the Presence of Motion Blur. *Paper presented in 2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, Hubli, Karnataka, pp. 78-81. doi: 10.1109/NCVPRIPG.2011.25.
- [29] Huang H, Wei X, Zhang L 2019. Encoding Shaky Videos by Integrating Efficient Video Stabilization. *Paper presented in IEEE Transactions on Circuits and Systems for Video Technology*, 29(5): 1503-1514. May 2019. doi: 10.1109/TCSVT.2018.2833476.
- [30] Cohen O, Apartsin A, Alon J, Katz E 2018. Robust Motion Compensation for Forensic Analysis of Egocentric Video using Joint Stabilization and Tracking. *Paper presented in 2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, Eilat, Israel, pp. 1-5. doi: 10.1109/ICSEE.2018.8646211.
- [31] Juranek L, Stastny J, Skorpil V 2018. Effect of Low-Pass Filters as a Shi-Tomasi Corner Detector's Window Functions. *Paper presented in 2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, Athens, pp. 1-5. doi: 10.1109/TSP.2018.8441178.
- [32] Zhao Z, Fu S, Wang Y 2012. Eye Tracking Based on the Template Matching and the Pyramidal Lucas-Kanade Algorithm. *Paper presented in 2012 International Conference on Computer Science and Service System*, Nanjing, pp. 2277-2280. doi: 10.1109/CSSS.2012.565.
- [33] Prawira W, Nasrullah E, Sulistiyanti SR, Setyawan FXA 2017. The Detection of 3D Object Using a Method of a Harris Corner Detector and Lucas-Kanade Tracker Based on Stereo Image. *Paper presented in 2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*, Palembang, pp. 163-166. doi: 10.1109/ICECOS.2017.8167126.
- [34] Ejaz N, Kim W, Kwon SI, Baik SW 2012. Video Stabilization by Detecting Intentional and Unintentional Camera Motions. *Paper presented in 2012 Third International Conference on Intelligent Systems Modelling and Simulation*, Kota Kinabalu, pp. 312-316. doi: 10.1109/ISMS.2012.73.
- [35] Okade M, Patel G, Biswas PK 2016. Robust Learning-Based Camera Motion Characterization Scheme With Applications to Video Stabilization. *Paper presented in IEEE Transactions on Circuits and Systems for Video Technology*, 26(3): 453-466, March 2016. doi: 10.1109/TCSVT.2015.2412772.