



Digital Video Stabilization of an Oscillating Camera using an Inertial Measurement Unit

MASTER THESIS

MS MECHATRONICS

Carl Zeiss AG, Jena,
Corporate Research Team (CRT)

and

Department of Mechatronics
Naturwissenschaftlich-Technische Fakultät

Universität Siegen

Ibad Rather (Matr. No. 1532894)

November 14, 2022

Supervisors:

University: Prof. Dr. Ing. Bhaskar Choube

Company: Dr. Steffen Urban

Abstract

Video stabilization is ubiquitous in today’s smartphones and action cameras. However, instead of classical “post-processing” image stabilization where image features are extracted and tracked over time [1] or modern deep learning based methods using dense optical flow [2], they need to perform the motion estimation on constrained hardware systems and with very high frame-rates of up to 100 fps or more.

Hence the motion estimation, that is necessary to correct vibrations, shaking or rolling shutter distortions is performed using Inertial Measurement Units (IMUs). Typically, the motion estimates are reduced to rotational motions [3] (using gyroscopes) as the distance between frames is small and the field-of-view of the lenses large ($\geq 50^\circ$). In addition, estimating translational motion is a difficult challenge using low-cost IMUs as the double integration of accelerometer readings is prone to strong drift and leads to large errors even over short time periods ($\leq 1s$).

This thesis should investigate and develop an image stabilization method for an oscillating camera. The camera has a small field-of-view and hence small translational movements along the viewing direction will be visible and need to be stabilized. The camera is coupled with an IMU and attached to a linear stage, that will mimic the oscillating movements. The camera records a monitor or a static target. An additional goal is the implementation, optimization and testing of the proposed algorithm on a modern embedded chip.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Challenges	2
2	Fundamental Blocks	4
2.1	Video Stabilization	4
2.1.1	Types of Video Stabilization	5
2.2	Motion Estimation for Image Stabilization	7
2.2.1	Software based motion estimation techniques	8
2.2.2	Sensor based Motion Estimation	10
2.3	Inertial Measurement Unit (IMU)	10
2.3.1	Pose Estimation with IMUs	11
2.3.2	IMU Noise Models	12
2.4	Pose Estimation using IMU	15
2.4.1	Classical Techniques	16
2.4.2	Neural Network Based	17
2.5	Neural Network Architectures	17
2.5.1	Multi Layer Perceptron (MLP)	17
2.5.2	Convolutional Neural Networks (CNNs)	18
2.5.3	Transformers	19
3	Literature Review	22
3.1	Digital Video Stabilization	22
3.1.1	5D Video Stabilization through Sensor Vision Fusion .	22
3.1.2	Deep Online Fused Video Stabilization	22
3.2	Inertial Pose Estimation	23
3.2.1	RIDI: Robust IMU Double Integration	24
3.2.2	RoNIN: Robust Neural Inertial Navigation in the Wild	24
3.2.3	TLIO: Tightly Learned Inertial Odometry	24
3.2.4	CTIN: Robust Contextual Transformer Network for Inertial Navigation	25

4 Work Methodology	26
4.1 Data Collection	28
4.1.1 Data Analysis	28
4.1.2 Simulated Data Generation	33
4.2 Structuring of Data	33
4.3 Training and Testing of Various Neural Networks	34
4.3.1 Data Augmentation	35
4.3.2 Training Setting	35
4.3.3 Convolutional Neural Network	36
4.3.4 ResNet	38
4.3.5 CNN-Transformer	41
4.3.6 Smoothing of Stabilization Trajectory	44
4.4 Model Deployment	47
4.4.1 Making Models Fast	47
4.4.2 Model Inference Latency on various Hardware	47
5 Results and Discussion	49
5.1 Future Work	50
Bibliography	53
A First Appendix Chapter	54
A.1 First Appendix Section	54
A.1.1 First Appendix Subsection	54
Declaration of Authorship	55

List of Tables

2.1	Summary of IMU error sources (Woodman, 2007)	14
4.1	Hardware technical specifications	27
4.2	IMU noise characteristics	29
4.3	Real translational-vibration data distributions	32
4.4	Real rotational-vibration data distributions	33
4.5	Inference latency of different models	48

List of Figures

1.1	GoPro Hero 10	2
2.1	Hardware Video Stabilization	5
2.2	Digital Video Stabilization	6
2.3	Camera Motion/Pose Estimation Techniques	8
2.4	Strap-down inertial navigation algorithm (Woodman, 2007) .	11
2.5	IMU noise model visualisation	12
2.6	Perceptron	18
2.7	Multi Layer Perceptron	18
2.8	CNN layers with FC layers	19
2.9	Transformer Architecture	20
2.10	Multi Headed Attention	20
3.1	Block diagram for proposed 5D video stabilization system (Zhuang et al., 2019)	23
3.2	Deep online fused video stabilization overview (Shi et al., 2022)	23
3.3	RoNIN process layout (Herath et al., 2020)	24
3.4	TLIO architecture (Liu et al., 2020)	25
3.5	CTIN Architecture (Rao et al., 2022)	25
4.1	Flowchart of a DVS pipeline using a IMU for pose estimation	26
4.2	Modified GoPro Hero 10 with large focal length lens	27
4.3	Bosch BMI260 IMU Sensor	27
4.4	Camera setup	28
4.5	Damped vibration in one axis	29
4.6	Real translational vibration data statistical analysis	30
4.7	Real rotational vibration data statistical analysis	31
4.8	Different IMU window samples	34
4.9	Stabilization trajectory (Ideal)	35
4.10	Convolutional Neural Network used	36
4.11	Model prediction vs ground truth for CNN	37
4.12	Error in each axis for CNN	38
4.13	CNN with residual skip connections (ResNet)	39
4.14	Model prediction vs ground truth for ResNet	40

4.15	Error in each DoF for ResNet	41
4.16	CNN-Transformer Network Used	42
4.17	Model prediction vs ground truth for CNN-Transformer	43
4.18	Error in each DoF for CNN-Transformer predictions	44
4.19	Rough regressed vs smooth ground-truth trajectory	45
4.20	Smoothed model output vs raw model output	45
4.21	DIS pipeline with trajectory smoothing	46

Chapter 1

Introduction

Cameras have become an integral part of our lives. We use cameras regularly, be it a smartphone camera, a hobbyist camera setup or professional cameras. Our demands for high quality media from cameras is also increasing constantly. We want sharper, color accurate, low lighting bright photographs. The videos are expected to capture scenes with as much realism as possible and we don't want too many movements in them irrespective of the way we capture them. If we are making a video capturing a beautiful sunrise or a landscape while walking, we do not want movements associated with walking in our video. We want it to be as smooth as possible. This is where *Image Stabilization* comes into play. Its goal being producing a stable video irrespective of the camera movements during capture. Stabilization is a very important quality metric for videos. A video may have all the aspects of photography correct but if it has too many sudden movements it would not look good.

1.1 Motivation

Image stabilization is a key technology to produce good quality photographs and videos. Be it a 250,000 Euros television broadcast camera or a 100 Euros hobbyist camera, stabilization of some sort is necessary. That is why image stabilization exists in every camera and smartphone these days. There even exists a series of product lines from many big companies like DJI just for this specific reason; to stabilize the image (video). These products can cost between less than 100 Euros to upwards of 1000 Euros or more for professionals. This clearly indicates that the need for stabilization is there and constant efforts are being put to make it better.

On microscopic level, the stabilization of videos becomes even more important as even small movements can cause large pixel shifts in the image plane. This can be seen by zooming in on your camera while trying to keep it stable. The effect of magnification becomes more evident and it will be-

come increasingly more difficult to keep the image smooth and stable as the movements are magnified as well. This will deteriorate the video quality and is very important to deal with.

Microscopic movements are the scope of this work. A modified GoPro Hero 10 (figure 1.1) camera is used as it allows to extract synchronized sensor and video data. The wide angle fisheye lens of the GoPro was exchanged with a narrow field-of-view lens that has a very large focal length. Switching the lens resulted in the default stabilization provided by GoPro (HyperSmooth) to not work anymore. The goal of this work is to explore different methods to obtain a stabilized video stream for large focal length lenses.



Figure 1.1: GoPro Hero 10

1.2 Challenges

- We have a camera with small FOV high focal length lens which keeps vibrating on its rig after the rig is moved around the scene.
- Vibrations are very small with a mean amplitude of 1.8mm but cause huge pixel shifts in the video recordings thus deteriorating the quality.
- Our goal is to stabilize the video so that there is no visual discomfort for the viewer and quality of the scene is preserved.
- We want to use an Inertial Measurement Unit (IMU) sensor to track these vibrations and stabilize the video based on it.
- Sensors are inherent to noise and IMUs being no different present a lot of challenges in accurate camera motion estimation.
- There exist IMU sensors with better noise characteristics, however, those are very expensive (> 1000 Euros) and are difficult to buy as they usually have a tactical grade placing them under export restrictions. Hence, in this work a low-cost consumer MEMS (Microelectromechanical systems) IMU is used.
- Using IMUs for both rotational and translational motion estimation for image stabilization present many challenges including:

-
- Accurate motion estimation over a longer period of time is not possible with classical algorithms.
 - The positional **drift** in translational motion estimation increases non-linearly with time which makes the image stabilization useless.
 - This results in to output stabilized video quality being worse
 - Working with this drift is especially not possible in this case as we have a accuracy requirement of about 0.5 mm for the stabilization to work effectively.
 - IMUs have been used for position estimation for a long time using the classic Kalman Filter and its variants. They produce acceptable results for some cases but would not work in our case as they are susceptible to strong drift without absolute position updates. IMUs are also generally coupled with cameras, GPS and Magnetometers to improve the accuracy but it is not possible for this use case.
 - Irrespective of these pitfalls, the use of IMU is necessary for estimating these small motions. To tackle these challenges the following contributions are presented in this thesis:
 - Analysis of real motion characteristics and building a simulation based on that analysis to generate huge amount of data.
 - Generating good data for data-driven approaches is very difficult, so, techniques for proper domain randomization are explored.
 - Exploring the possible use of various neural network architectures to learn motion characteristics of these vibrations and are not susceptible to the drift.

There are a lot of challenges in this work to overcome and that makes this exciting. In the next chapter , fundamental of image stabilization and pose-estimation will be discussed. It will also cover some possible solutions and their pitfalls and advantages.

Chapter 2

Fundamental Blocks

In the previous chapter I discussed the importance of Video Stabilization and the challenges associated performing it using an IMU sensor. This chapter discusses current techniques available to us and also we make a case for our chosen technique. Then we will also discuss in depth the challenges associated with IMU pose-estimation and how we can overcome those challenges. Fundamentals of neural networks and their applications will also be briefly conferred as they are a huge part of this work.

2.1 Video Stabilization

If a camera is not stationary during video capture the output video quality may degrade and not look good. Depending on the movements during capture, the video may look very shaky or even the subjects may not be properly visible. This is very undesirable and can cause visual discomfort (Jia and Evans, 2012). One solution to this is to keep the camera steady by mounting it on stable tripod or to a rigid object. But this is not practically possible all the time. There are various scenarios in which the camera may move around a lot and the movements may not be smooth at all e.g an action camera mounted on a helmet or a bicycle, a camera mounted on a car or a robot in wild or some move-able rig or a robot arm.

The motion of the camera can be high-frequency tremors due to vibration, shaking or jiggling of the mounting point (Ryu and Chung, 2012) or low frequency such as movement of handheld camera during walking (Guilley et al., 2021). Practically, in most of the video recording scenarios, the camera will have some movement. The goal of video stabilization is to create a output video with same visual content without the undesired movements.

2.1.1 Types of Video Stabilization

To deal with such movements and to make the video stable various techniques exist. These techniques can be grouped in two categories:

- Hardware Video Stabilization
 - Digital Video Stabilization

There are techniques which combine both these methods and are called hybrid video stabilization techniques.

The goal of each of these stabilization techniques is to estimate the motion or pose and then use the estimated motion to stabilize the video. Motion/pose estimation will be discussed in the section 2.2 in detail.

Hardware Video Stabilization (HIS)

Hardware video stabilization at its core means moving the lens or the image sensor or the camera setup itself to counteract these movements while recording the video. The movements of camera can be minimised by the use of mechanical stabilizers like tripods or dollies (Grundmann et al., 2011). The use of electronic gimbals for video recording has also increased recently for both professionals and hobbyists. Video can also be stabilized by varying the optical path between lens and the image sensor. This is generally achieved by using electronic motors and IMU sensors to move around the lens or the image sensor or both to cancel out the motions. After the motion has been counter-acted upon, images are captured which results in a stabilized video output as seen in figure 2.1. These methods although mostly effective are not perfect and have both advantages and disadvantages.

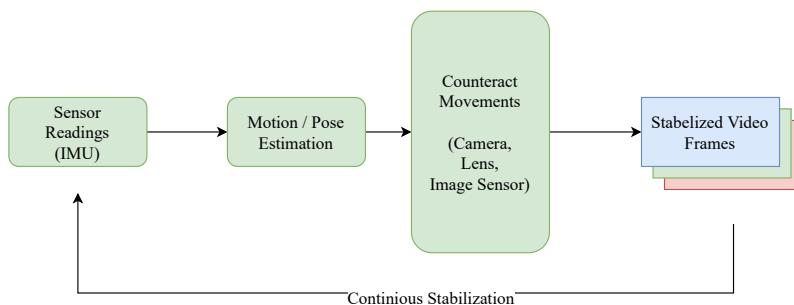


Figure 2.1: Hardware Video Stabilization

Advantages of HVS:

- No post-processing for stabilization needed.
 - Works irrespective of the nature of object being captured.

-
- Full image without cropping can be used.
 - Shutter Speed of the camera can be reduced

Disadvantages of HVS:

- Camera setups are very bulky.
- These solutions can be very expensive.
- More moving parts means higher maintenance.
- Cannot be improved once hardware is implemented.
- Not possible on all camera setups.

Digital Video Stabilization (DVS)

Digital Video Stabilization (DVS) is a software based video stabilization technique (Guilluy et al., 2021). The technique here is similar what is done is HVS i.e. first the motion of the camera is estimated (pose-estimation) and based on this, the image is stabilized. The difference here is that after pose estimation is done, instead of moving around the hardware a suitable transformation (warping) is applied to the image (Censi et al., 1999) as shown in figure 2.2. The resulting stabilized image (video) is motionless irrespective of the camera motions. DVS is a very powerful technique but like other methods and techniques, it has its advantages and disadvantages.

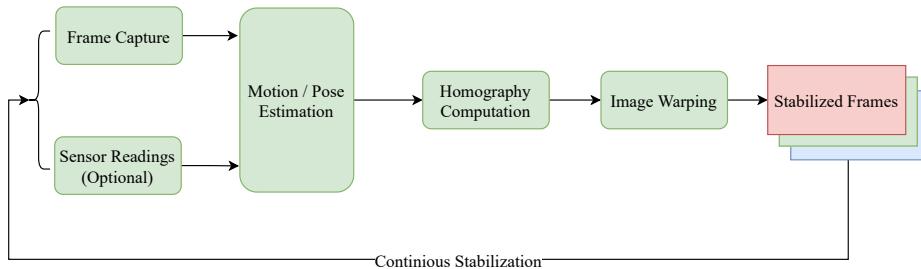


Figure 2.2: Digital Video Stabilization

Advantages of DVS:

- Cost effective.
- Smaller camera setups can be used.
- No changes need to be made to the camera setup for DIS.
- Can be achieved using IMU sensors for pose estimation which are not very expensive.

-
- Can be applied to any video; even the historic ones.
 - Different DIS algorithms can be used on a single video in post processing.
 - Relatively easy to modify DIS software.

Disadvantages of DVS:

- The resolution of the output images (video) is reduced (generally by 5 to 10 percent).
- May require some manual labour (like in Adobe Premiere Pro: manual feature tracking).
- Some videos are very difficult (even impossible) to stabilize if there is too much vibration or the features are not visible.
- May require a lot of computational power (optical flow)

Regardless of the type of video stabilization technique motion estimation is a fundamental task and will be discussed in the next section.

2.2 Motion Estimation for Image Stabilization

Camera motion (pose) estimation is very important and one of the most challenging parts of image stabilization. Before hardware and digital stabilization techniques/algorithms can be implemented, we need to perform pose estimation to figure out the camera motions. Then based on the actual, shaky camera motions, a smoothed trajectory can be calculated that will create a stabilized video. The accuracy of motion estimation ultimately determines the effectiveness of image stabilization (Ryu and Chung, 2012). To estimate the camera motion many sensor and software based techniques exist as shown in figure 2.3.

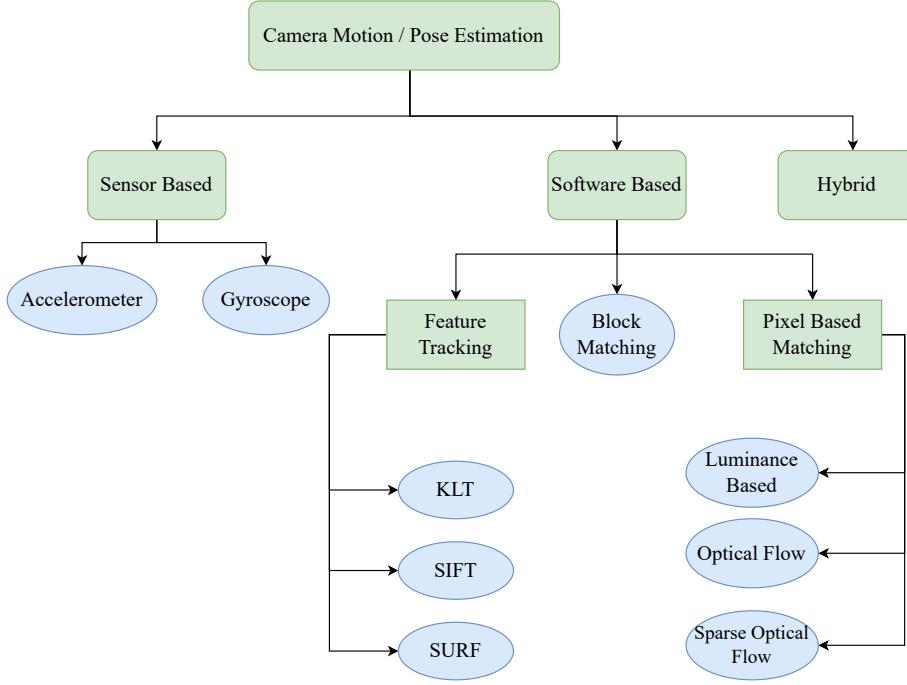


Figure 2.3: Camera Motion/Pose Estimation Techniques

2.2.1 Software based motion estimation techniques

Software based motion estimation techniques/algorithms use the images (frames) coming from the camera for motion estimation and subsequently for stabilization. Some of these software based techniques are listed below (Guilluy et al., 2021):

- Pixel-based matching
- Block-Matching
- Feature-Matching

Each of these approaches/algorithms used in motion estimation are briefly discussed in the subsequent sections. This will also lay a foundation for why we are not using any of these techniques for this work.

Pixel-Based Matching

This method estimates pose by determining the motion of pixels between two frames (Guilluy et al., 2021). For correspondence determination, luminance of two consecutive frames is assumed to be constant throughout. This poses a problem and there can be many pixels having same luminance, so, other constraints are also needed to obtain a unique solution (Guilluy et al., 2021).

Finding ***Optical Flow*** between two consecutive frame is another method for motion estimation. Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image (Horn and Schunck, 1981). This gives us important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement (Gibson, 1977). Using this information, relative motions can be estimated between consecutive image frames. These relative motions can be used for camera motion estimation. Many modern neural network based optical flow algorithms have also been developed recently and are used in image stabilization tasks (Shi et al., 2022). This method achieves very good results but has very high computational requirements.

Block-Matching

Block matching techniques use blocks of pixels for motion estimation instead of finding correspondence between pixels of consecutive frames. This allows to remove some ambiguities that occur in pixel-based matching methods (Guilluy et al., 2021). Block-matching approach offers a flexible trade-off between complexity, computational efficiency and accuracy (Guilluy et al., 2021) but relies heavily on unambiguous features in the image.

Feature-Matching (Tracking)

In these methods, the algorithm first selects the features that are easily recognisable and distinguishable. The motion is estimated by tracking only these features making them computationally less expensive than dense optical flow methods. There are several feature tracking algorithms that are widely used in many computer vision applications.

Kanade Lucas Tomasi (KLT) feature tracker (Tomasi and Kanade, 1991) is a widely used feature tracking algorithm in computer vision. Features are selected and their position is initialized using this technique and then optical flow is used to track their position (Guilluy et al., 2021).

Scale-invariant feature transform (SIFT) is also used for motion estimation in digital image stabilization algorithms. It has been designed for extracting highly distinctive invariant features from images, which can be used to perform reliable matching of the same object or scene between different images (Battiato et al., 2007). It contains the orientation of the feature in order to be rotation-invariant (Guilluy et al., 2021).

Speeded up robust features (SURF) is inspired by SIFT but is faster (Zheng et al., 2015). This technique is also widely used in feature selection and tracking for motion estimation.

A common theme here in software based motion estimation techniques is to detect features (be it a pixel, or a group of pixels or any other particular image feature) and then tracking it for motion estimation. But, there are certain use cases like ours where feature tracking from frame to frame is very difficult or unreliable. The features in an image sequence can be moving in different directions, which makes reliable motion estimation using these techniques impossible. This might happen if the observed scene is not static and contains many moving objects. This is the case for this work and the goal is to estimate the motion of the camera independent of image observations. To this end inertial sensor based pose-estimation will be used.

2.2.2 Sensor based Motion Estimation

Pose estimation is a huge area of research especially for the field of robotics and autonomous driving. Various sensors can be used for motion estimation, including accelerometers, gyroscopes, magnetometers or satellite navigation systems. To help camera pose estimation generally a combination of gyroscope and accelerometer is used. In combination, these two sensors are called Inertial Measurement Unit (IMU). With these two sensors the 6 degree of freedom (DoF) motion of a camera in space is fully observable. For this work, we will be using IMU for pose estimation as using it has many advantages and will make our image stabilization algorithm invariant to moving objects.

2.3 Inertial Measurement Unit (IMU)

IMU is one of the most commonly used sensors in the world and present all around us in mobile devices, smartwatches, virtual reality gear and cameras. IMU sensors are used in a wide variety of applications like navigation, robotics, drones, smart watches, sports learning, augmented reality systems, industrial quality control (Ahmad et al., 2013) and also for image stabilization in cameras. In all these applications, IMU are used for pose estimation in some form; either for absolute pose estimation or change in pose.

An IMU measures acceleration(m/s^2) and angular velocity (rad/s). They can make these measurements in multiple degrees of freedom e.g., a 6-DoF IMU sensor includes a 3-axis accelerometer (x, y, z) and 3 axis gyroscope (x, y, z) (Constant et al., 2021). The accelerometer will make independent acceleration measurements in these axis. Similarly, the gyroscope makes independent angular velocity measurements about these three axis.

There are a lot of reasons to use IMUs and these reasons are surely a very huge contributor to them being one of the most commonly used sensors in the world. There are different types of IMUs e.g., optical, laser and Micro-electro-mechanical systems (MEMS) IMUs. All these different

types of IMUs record similar measurements but have different characteristics, sizes, accuracy and prices. MEMS IMUs are the scope of this work as they have various qualities that make them the best fit for DVS. Some of their qualities are (Woodman, 2007):

- They have small size
- low weight
- rugged construction
- low power consumption
- cheap
- high reliability
- low maintenance
- can be used in hostile environments

2.3.1 Pose Estimation with IMUs

In this thesis, a pose has six degrees of freedom and is defined by a position and orientation with respect to some coordinate system. Continuous IMU readings are taken and using some algorithm pose is estimated. Figure 2.4 below shows the strap-down inertial navigation algorithm (Woodman, 2007). For orientation estimation gyroscope readings are used and for position estimation both gyroscope and accelerometer readings are used. The use of this algorithm directly for pose-estimation is not possible as the IMU readings are far from being ideal, they contain noise and have a bias (Woodman, 2007) which needs to be accounted for. In addition, integration of the accelerometer signal needs prior information like the initial velocity or the initial position that are, dependent on the application, not always easily obtainable.

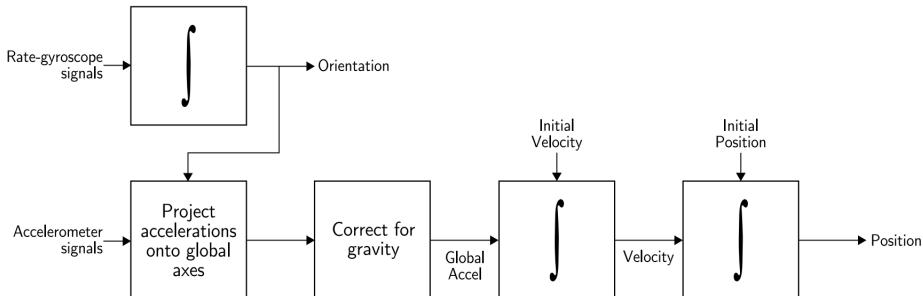


Figure 2.4: Strap-down inertial navigation algorithm (Woodman, 2007)

2.3.2 IMU Noise Models

MEMS sensor outputs are inherent to noise and there is different types of noise with which we have to deal. For IMUs, the readings coming from a sensor could be modelled as a combination of three entities: true signal, bias and white noise.

Gyroscope

$$\omega_n(t) = \omega(t) + b_g(t) + n_g \quad (2.1)$$

Where $\omega_n(t)$ is the noisy gyroscope signal, $\omega(t)$ is the true gyroscope signal, $b_g(t)$ is the gyroscope bias and n_g is the gyroscope white noise.

Accelerometer

$$a_n(t) = a(t) + b_a(t) + n_a \quad (2.2)$$

Where $a_n(t)$ is the noisy accelerometer signal, $a(t)$ is the true accelerometer signal, $b_a(t)$ is the accelerometer bias that is time varying and n_a is the accelerometer white noise.

To visualise the *Noise Model*, let's take a true unit signal as show in figure 2.5. **Bias** can be defined as constant offset from true signal value. White noise can be thought of as random fluctuations of equal intensity and different frequency from the true value of the signal (2.5).

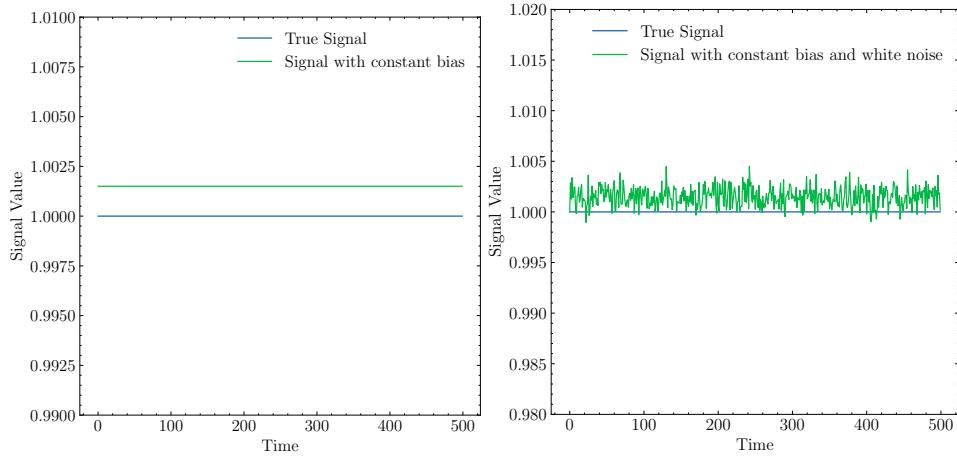


Figure 2.5: IMU noise model visualisation

For both gyroscope and accelerometer the signal can be independently modeled this way in each of its axis (x, y and z) (Trawny and Roumeliotis, 2005). As you can see from the plots, the noisy signal looks very different from the true signal. Using the noisy signal directly for pose estimation can lead to very inaccurate results. Although the modelling equation looks the same for both gyroscope and accelerometer, but they have different effects on pose estimation as the strap-down inertial algorithm suggests. So, there

are different challenges associated with position and orientation estimation using this sensor with absolute position estimation being especially difficult.

Gyroscope Error Characteristics

To get orientation from the gyroscope, the readings need to be integrated once and the global orientation needs to be updated. Doing this continuously with the noisy signal will result in erroneous orientation tracking and ultimately in a drift of the absolute rotation even if the device is stationary. Hence methods are needed to estimate the sensor errors, especially the bias. In table 2.1 you can see all the errors and effect of those on orientation measurement.

Accelerometer Error Characteristics

Generating position information from accelerometer is not a single step like in case of Gyroscope. Based on the strap-down inertial navigation algorithm shown in figure 2.4 there are multiple steps involved. First, the accelerometer readings need to be projected onto the global navigation frame using the orientation calculated from gyroscope. Then, the accelerometer reading needs to be corrected for gravity. After that we double integrate the signal based on initial velocity and initial position to finally get the position.

So, there is double integration present in the algorithm. We also have a noisy signal, so, the noise also gets integrated two times. This can result into very inaccurate position estimations and is one of the most challenging parts of working with an IMU. In table 2.1 you can see all errors in an MEMS accelerometer and effect of those on position measurement.

Error Type	Description	Result of Single/Double Integration
Gyroscope bias	A constant bias ϵ	Steadily growing angular error. $\theta(t) = \epsilon t$
Gyroscope white noise	White noise with some standard deviation σ	An angular random walk, whose standard deviation grows with the square root of time. $\sigma_\theta(t) = \sigma \sqrt{t}$
Accelerometer bias	A constant bias ϵ in the accelerometer's output signal.	A quadratically growing position error. $s(t) = \frac{\epsilon}{2} t^2$
Accelerometer white noise	White noise with some standard deviation σ	A second order random walk. The standard deviation of the position error grows as $\sigma_s(t) = \sigma \sqrt{\frac{t}{3}}$

Table 2.1: Summary of IMU error sources (Woodman, 2007)

Challenges in Working with IMUs

For the reasons mentioned in the previous sections, it is not straightforward to work with IMUs, especially for absolute position estimation as the errors introduced are squared with time. This results in a **drift** from actual position, which for any image stabilization algorithm is very detrimental. Orientation estimation is still relatively reliable over short periods of time and is the main reason why gyroscope based hardware as well as digital video stabilization is more common.

2.4 Pose Estimation using IMU

Based on the strap-down inertial navigation algorithm (Section 2.4) and Newtonian mechanics, pose can be estimated from the IMU readings using equation 2.3 and equation 2.4. We need a history of pose for the iterative pose estimation to work. ω and a are the gyroscope and accelerometer readings respectively as defined in section 2.3.

$$O(t) = O(t - dt) \exp\left(\int_{t-dt}^t \Omega(t) dt\right) \quad (2.3)$$

Orientation $O(t)$ is the updated orientation in global frame at current timestamp using orientation at previous timestamp $O(t - dt)$ and the sensor readings $\Omega(t)$ as shown in equation 2.3.

$$\begin{aligned} a_g(t) &= O(t)a(t) \\ v(t) &= v(t - dt) + (a_g(t) - g_g)dt \\ P(t) &= P(t - dt) + v(t)dt \end{aligned} \quad (2.4)$$

Position estimation can be done using a series of mathematical equations as shown in equation 2.4. The accelerometer readings are first projected onto the global axis based on the orientation $O(t)$. Then the current velocity $v(t)$ is calculated by integrating the acceleration $a_g(t)$ and adding it to velocity at previous time-step $v(t - dt)$. Finally, to get the current position $P(t)$ we integrate the velocity $v(t)$ add it to the previous position.

These equations are the backbone for classical pose estimation algorithms using IMU sensors. Bias and white noise present in the sensor readings also integrates causing a **drift** in estimation. In case of Gyroscopes, white noise causes an *angular random walk* whose standard deviation grows proportionally with square root of time (Woodman, 2007). And leaving the bias unaccounted for causes an orientation error growing linearly with time as shown in table 2.1. In case of accelerometer the drift is aggravated because the orientation error is also contributing to it. The subtraction of gravity g from the readings is not perfect either and causes further drift. The bias in the readings also increases the error (hence the drift) quadratically with

time. On top of all these errors the effect of white noise is a *second order random walk* whose standard deviation increases quadratically with time as shown in table 2.1.

We can very comfortably conclude that using the inertial navigation system in its pure form is not possible for accurate pose estimation with MEMS IMUs. There are several techniques which can be used to diminish the effect of the noise and the bias. Bias can be estimated for any IMU by keeping it stationary for a few seconds (for the accelerometer this is only true if any axis is perfectly aligned with the gravity axis). The readings that we get for a stationary IMU sensor is the bias for that sensor. Then this bias can be adjusted in the signal recorded during operation.

But modern learning based methods do not use these equations. Instead they are more data driven and let the neural network infer these relations based on the training data.

2.4.1 Classical Techniques

IMUs have been used extensively for pose estimation for the better half of the century and were used to track the Apollo 11 rocket that took man to the mars. Over the years many techniques and algorithms have been developed to improve pose-estimation accuracy using IMUs. These techniques range from simple signal filtering to more advanced state-estimation algorithms.

Signal Filtering

There are also various signal-processing techniques that can be used to reduce the white noise. Noise can be characterised as a high-frequency content in any signal. Therefore, we can use simple filters like a low pass filter or a moving average filter with big horizon to eliminate the noise up to some extent. These techniques are in no way perfect and may even cause loss of true signal if not configured properly.

Kalman Filter

Kalman filters are very useful for estimation purposes. They can be used to estimate pose using an IMU sensor by filtering noise based on a covariance matrix (Ferdinando et al., 2012). The values of this covariance matrix can be varied but there is a trade-off between noise sensitivity and accuracy. Kalman filter is used to fuse multiple sensors to improve the accuracy of estimation. But in our case the accuracy it provides over time is not enough ((Kok et al., 2017), (Alatise and Hancke, 2017), (Bangera et al., 2020)) which has been evaluated already internally and hence cannot be used.

These techniques may be good enough for some other applications, but in our case their use is not enough. The precision requirements are very

high. Even with the use of these algorithms, there is still a drift present which is unacceptable for this use case.

2.4.2 Neural Network Based

With the advancement of neural networks and their ability to outperform classical algorithms in many domains, over the years many techniques have been developed to solve the challenge of double integrating accelerometer readings. Based on the fact that the vibrations in our camera setup have similar motion characteristics and patterns, a neural network can learn these pattern and make accurate predictions. Some of the techniques which laid the foundation of the methodology we used are discussed in chapter 3 of this report.

2.5 Neural Network Architectures

The evolution of neural networks and deep learning techniques over the past decade has made the use data driven approaches very common. Data driven approaches have surpassed classical techniques in many applications. There are various neural network architectures, their variations and combinations present today. Each of these architectures have properties that aid in achieving solutions for various challenges.

2.5.1 Multi Layer Perceptron (MLP)

MLPs are the simplest form of Artificial Neural Networks (ANNs) or Deep Neural Networks (DNNs) and are made of a combination of perceptrons/neurons (figure 2.6). An MLP has at-least three layers; an input-layer, a hidden layer and an output-layer as shown in figure 2.7. During the training phase the weights of the network are adapted using back-propagation ((Rumelhart et al., 1986)) to minimize a target loss function (gradient descent). These networks act as universal function approximators (Cybenko, 1989) and therefore are used to create mathematical models by regression analysis. We will be using MLPs as Fully Connected (FC) layers in all of our networks for this purpose.

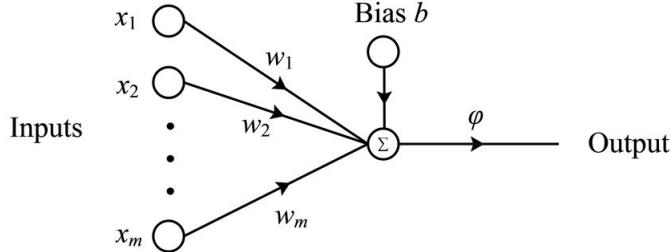


Figure 2.6: Perceptron

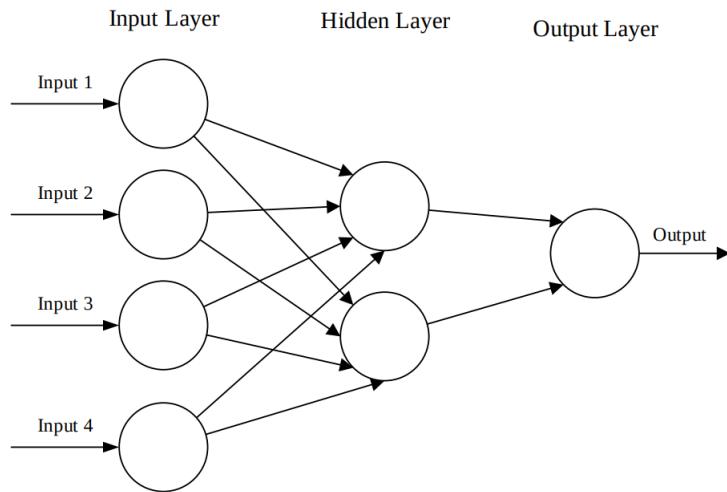


Figure 2.7: Multi Layer Perceptron

2.5.2 Convolutional Neural Networks (CNNs)

CNNs are another class of ANNs and are responsible for revolutionising the field computer vision and has applications in Natural Language Processing, Recommender Engines and Time Series as well. These networks also have neurons that optimise through learning (O'Shea and Nash, 2015). These networks are capable of learning multiple features from the data by abstracting the data into feature maps. In our case we input a 6 channel (3-accelerometer and 3-gyroscope) data and then the CNNs take the data from 6 channels to 64 channels (tensor maps). Then these features are fed into the Fully Connected Layers (FC / MLP) at the end to regress the pose as shown in figure 2.8. This also results in reducing dimensionality of the data while regressing the output.

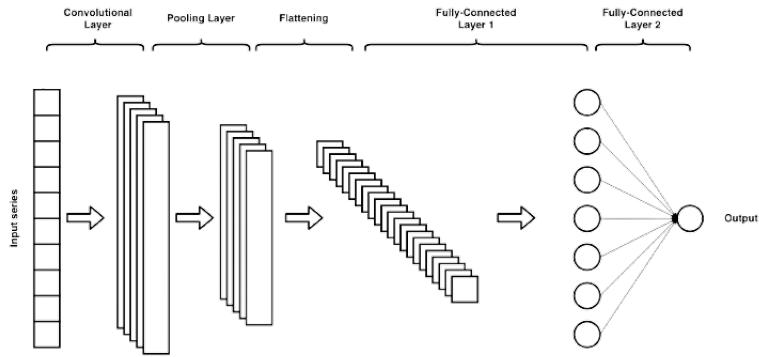


Figure 2.8: CNN layers with FC layers

2.5.3 Transformers

Transformer architecture as shown in figure 2.9 is one of the latest proposed neural network architectures that has outperformed its predecessors. Its use mainly started for natural language processing (Vaswani et al., 2017) but was also adopted for computer vision (Dosovitskiy et al., 2020). The use of transformers is also extended for Inertial based Human Activity Recognition (HAR) Shavit and Klein (2021) and Inertial Navigation (Rao et al., 2022). This architecture adopts Multi Headed Attention Mechanism (MHA) (Vaswani et al., 2017) which results in weighing the significance of each part of input data. This significantly improves performance of the network as it learns the importance of each feature and can look at the bigger picture from the data.

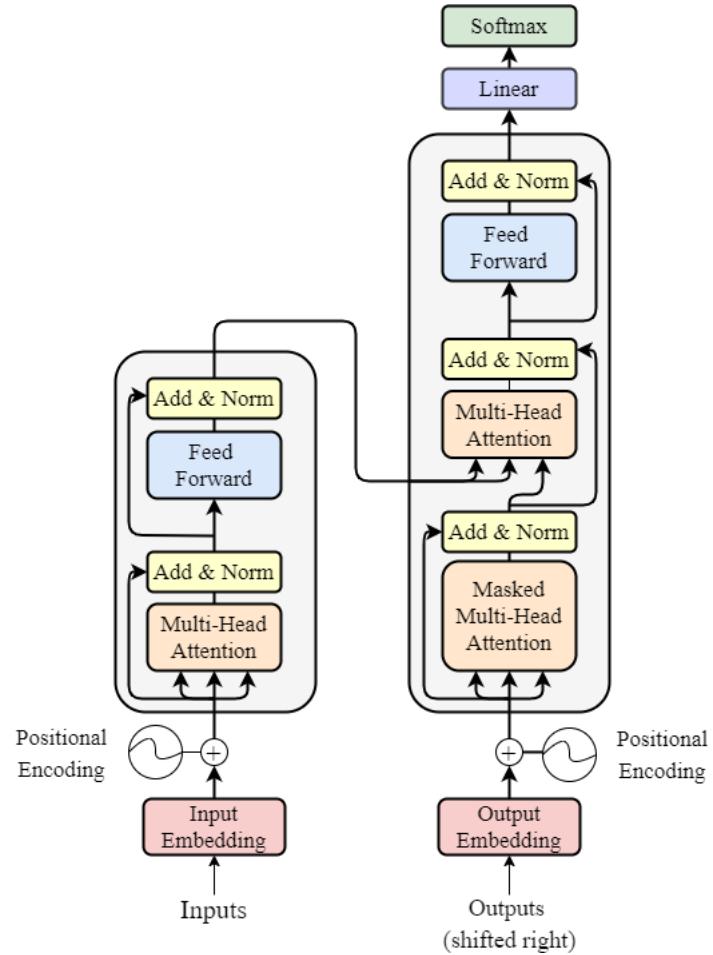


Figure 2.9: Transformer Architecture

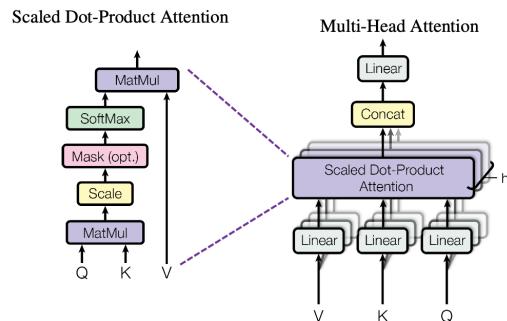


Figure 2.10: Multi Headed Attention

In this chapter we discussed different methods for image stabilization, challenges of pose-estimation and which methods are available to estimate

pose from MEMS IMUs. The next chapter summarizes some state of the art techniques used to solve the challenges we are facing.

Chapter 3

Literature Review

Previous chapter laid out the foundations for video stabilization and motion estimation. We also made the case for the techniques and algorithms chosen in this work. Now, this chapter includes some of the state-of-the-art methods for digital video stabilization and pose estimation.

3.1 Digital Video Stabilization

Various sensor based digital image stabilization techniques have been developed over the years using both classical and data-driven approaches. These techniques are limited to rotational motion compensation or using image data.

3.1.1 5D Video Stabilization through Sensor Vision Fusion

This research aims to overcome the challenge of going beyond 3D stabilization (rotation only) by also including translational motion in their image stabilization algorithm for smartphones. The proposed 5D stabilization is achieved by carefully combining sensor based 3D rotation stabilization and vision based 2D translation stabilization (Zhuang et al., 2019). Figure 3.1 shows the block diagram for this approach. This method is promising as it deals with translational motion as well but does so using image features and hence cannot be used for our use case.

3.1.2 Deep Online Fused Video Stabilization

This work aims at stabilizing videos using *Gyroscope* data as well as *Optical Flow* through unsupervised learned (Shi et al., 2022). Instead of pose estimation, warping grid calculation and image warping the aim here is to directly warp the image based input data (images and gyroscope data) as shown in figure 3.2. The results seem to outperform some of the previous state-of-the-art methods present before it. Although very promising this

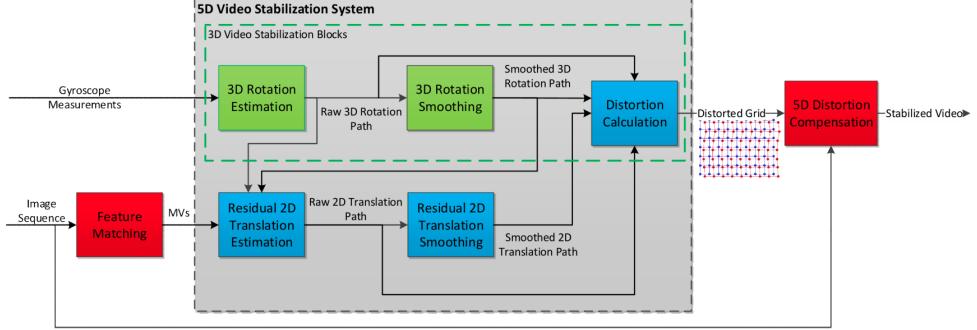


Figure 3.1: Block diagram for proposed 5D video stabilization system (Zhuang et al., 2019)

method cannot be used in our case as we want our algorithm to be image invariant based on reasons discussed in section 2.2.1. It also has a component of finding optical flow for each frame using a different neural network which makes this algorithm very computationally expensive and hence not usable all computers/hardware.

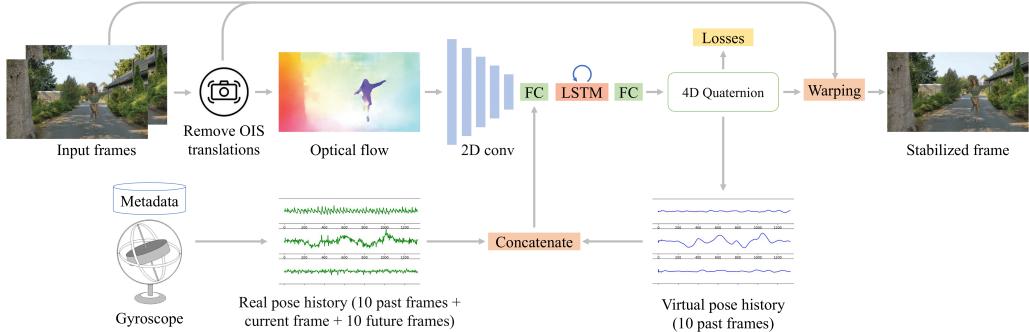


Figure 3.2: Deep online fused video stabilization overview (Shi et al., 2022)

3.2 Inertial Pose Estimation

Classical algorithms fail to accurately estimate the pose using MEMS IMU over a long period of time. With the advancement of artificial neural networks (ANNs) and their architectures new data driven techniques have been developed for this purpose. These data-driven techniques have outperformed their classical predecessors and are becoming a standard for this purpose. The drawback is that they are computationally very expensive and have very high edge hardware requirements.

3.2.1 RIDI: Robust IMU Double Integration

RIDI was the first data-driven neural network approach which tackles the challenges of double integrating accelerometer readings. The algorithm regresses a velocity vector from the history of linear accelerations and angular velocities (IMU windows - discussed in section 4.2), then corrects low-frequency bias in the linear accelerations, which are integrated twice to estimate position (Yan et al., 2018). The analysis of results manifested that this algorithm outperformed the heuristic-based methods and even came closer to visual inertial navigation (Yan et al., 2018). The issue with this is it can only regress position and orientation estimation needs to be achieved using classical algorithms.

3.2.2 RoNIN: Robust Neural Inertial Navigation in the Wild

Developed by the same team as RIDI, RoNIN uses more advanced neural network architectures like ResNet, LSTM and TCN as its backbone to regress a velocity vector given a window of IMU samples (Herath et al., 2020). Input to the network in this case are the angular rates (Gyroscope readings), accelerations (accelerometer readings) and device orientations as shown in figure 3.3 and the output is motion trajectory (poses). Using these architectures and increasing the quality and size of data RoNIN outperforms RIDI and sets a new benchmark for inertial navigation (Herath et al., 2020). Our methodology is influenced by this technique and is discussed in chapter 4 of this report.

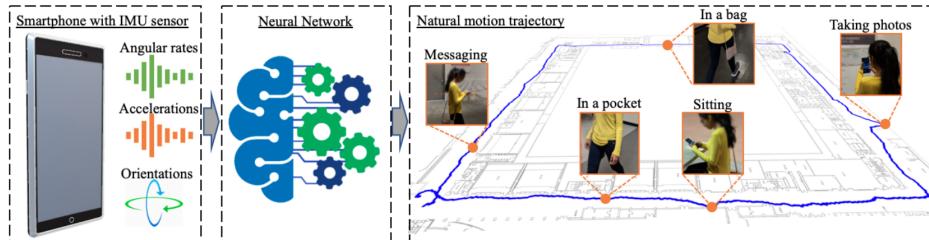


Figure 3.3: RoNIN process layout (Herath et al., 2020)

3.2.3 TLIO: Tightly Learned Inertial Odometry

This combines both classical and learn-based approaches by using an Extended Kalman Filter (EKF) and a deep neural network (DNN). The EKF is responsible for the prediction step while the outputs (displacement and uncertainty) from the neural network are used as a measurement update. The filter estimates rotation, velocity, position and IMU biases at IMU rate (Liu et al., 2020) as shown in figure 3.4. This approach is promising, but we were not able to reproduce the good results on our datasets, probably

due to the difficult hyper-parameter settings that an EKF needs to perform well.

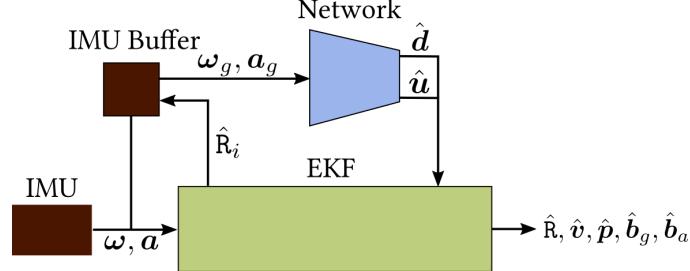


Figure 3.4: TLIO architecture (Liu et al., 2020)

3.2.4 CTIN: Robust Contextual Transformer Network for Inertial Navigation

This algorithm takes advantage of the recent advances in deep learning by using Multi-Headed-Attention (MHA) mechanism introduced in (Vaswani et al., 2017). It uses a ResNet based encoder enhanced by local and global MHA to capture spatial contextual information from IMU measurements (Rao et al., 2022) as shown in figure 3.5. CTIN is claimed to be robust and to outperform state-of-the-art models but cannot be evaluated as it is not available publicly.

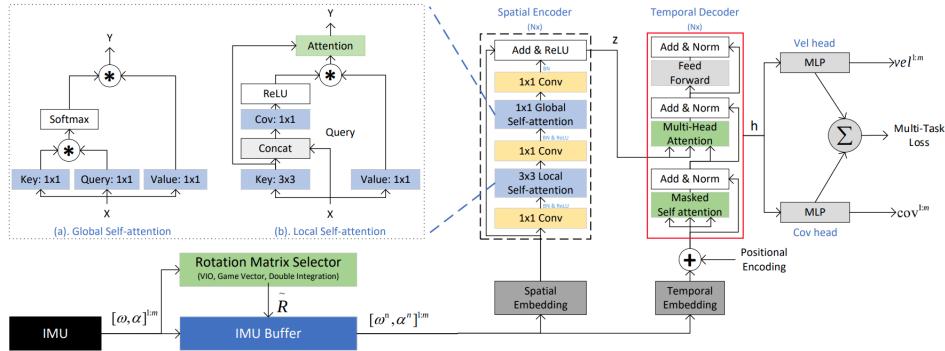


Figure 3.5: CTIN Architecture (Rao et al., 2022)

Chapter 4

Work Methodology

In the past chapters, the challenges associated and methods available were discussed. This chapter contains the process we used to digitally stabilize video for our use-case. This chapter now proposed methods to digitally stabilize videos with large focal lengths using inertial data for pose estimation alone. Figure 4.1 depicts a general flowchart of the developed pipeline.

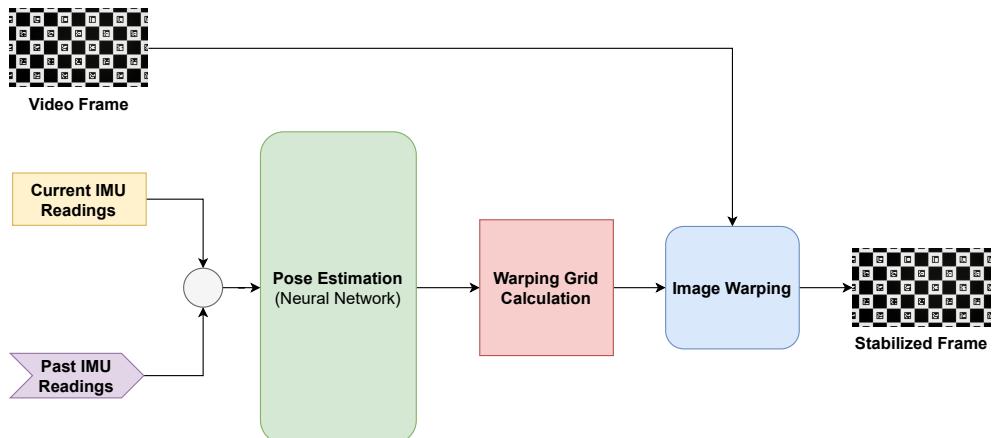


Figure 4.1: Flowchart of a DVS pipeline using a IMU for pose estimation

In this work, an IMU sensor will be used for camera pose estimation that is a prerequisite for video stabilization. Based on the accuracy requirements for the camera pose estimation algorithm a data-driven neural network approach will be used, as classical signal processing combined with filtering failed to provide the required accuracy over longer periods of time due to severe drift. To be able to train the models to the required accuracy, a combination of simulated and real data is used. To capture the video we are using a GoPro Hero 10 which provides synchronised image and sensor data capture. The factory wide FoV lens was swapped to a low field of view lens having a large focal length (depicted in figure 4.2). The simulation was

build using Unreal Engine 4 and Microsoft AirSim. Table 4.1 shows the technical specifications of the camera setup. For simulation similar camera parameters were used.

Image Sensor	Sony	Diagonal 7.85 mm (Type 1/2.3) 23.91 M CMOS Image Sensor with Square Pixel.
IMU Sensor	Bosch	6-axis IMU Sensor Accelerometer (A): 16-bit or 0.06 mg/LSB Gyroscope (G): 16-bit or 0.004 dps/LSB
Lens	Zeiss	15° FOV and 200 mm focal length

Table 4.1: Hardware technical specifications



Figure 4.2: Modified GoPro Hero 10 with large focal length lens



Figure 4.3: Bosch BMI260 IMU Sensor

4.1 Data Collection

Collecting enough data for any data driven learning approach is cumbersome and sometimes an iterative process. The data or training data in this case are the readings coming from the IMU sensor and the target (ground truth) being the camera pose (position and orientation). Real data was collected by mounting the camera on a rig which had vibrations associated with it on movement (figure 4.4). We made sure that the data depicts real life scenarios by moving the rig around in many different ways and let it vibrate and collect the data. The ground truth camera and IMU poses as well as all calibration parameters like camera intrinsics, IMU parameters and transformations between the sensor coordinate frames were generated using proprietary Zeiss software and is beyond the scope of this work.

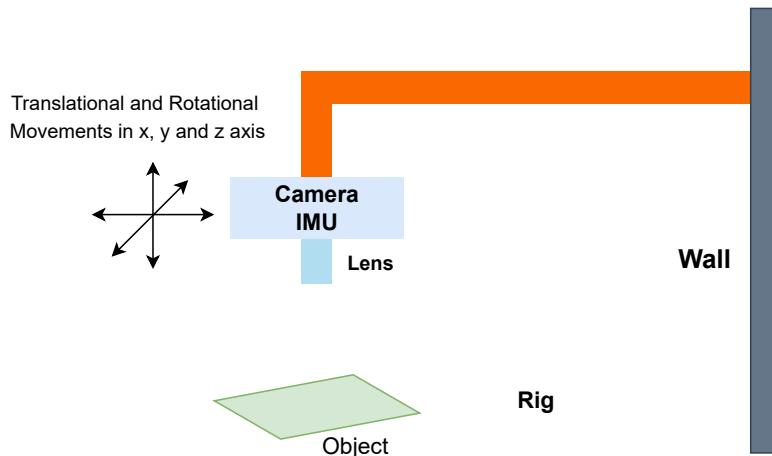


Figure 4.4: Camera setup

As generating enough real data is time consuming, the motion patterns were analyzed and a simulation was created to generate more training data. Generating simulated data with accurate ground truth is a fast processes not requiring much post-processing. This also allowed us to vary the noise characteristics of the simulated IMU data to make the neural network robust and generalize well.

4.1.1 Data Analysis

The data recorded from the real world camera setup was analysed to determine the vibration and IMU noise characteristics. Table 4.2 summarises the white noise and bias characteristics determined from the raw IMU Sensor Readings.

Parameter	Value
Accelerometer White Noise	$\mu = 0$ and $\sigma^2 = 0.3 \times 10^{-3} m/s^2$
Gyro White Noise	$\mu = 0$ and $\sigma^2 = 2.8 \times 10^{-6} rad/s$

Table 4.2: IMU noise characteristics

Apart from the IMU noise characteristics the vibrations from the real ground truth data were analyzed. These vibration characteristics are important to understand their and to set the Simulation parameters. Vibrations are characterised by **Amplitude** (A or θ), **Time Constant** τ and **Frequency** ω . We have damped oscillations for both translational and rotational vibrations as shown in figure 4.5 and can be characterised by the equations 4.1 and 4.2.

$$A(t) = Ae^{-t/\tau} \sin(2\pi\omega t) \quad (4.1)$$

$$\theta(t) = \theta e^{-t/\tau} \sin(2\pi\omega t) \quad (4.2)$$

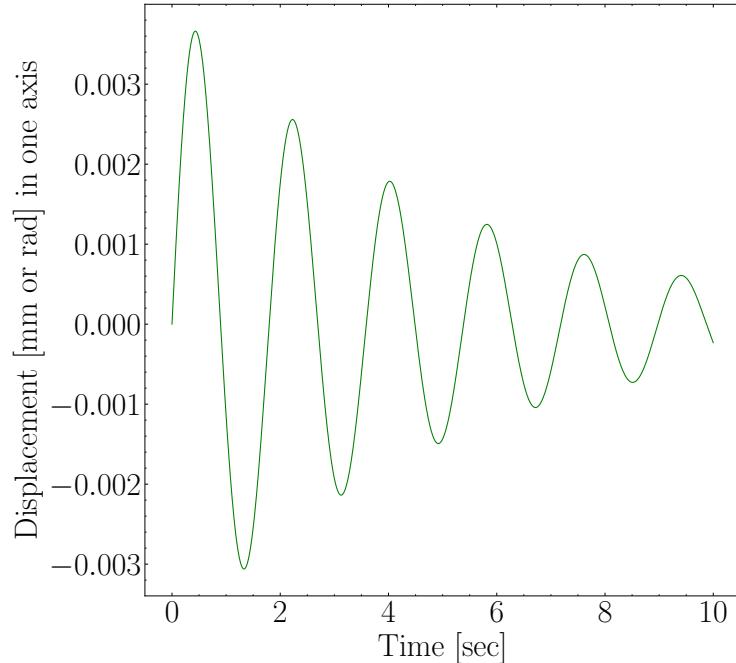
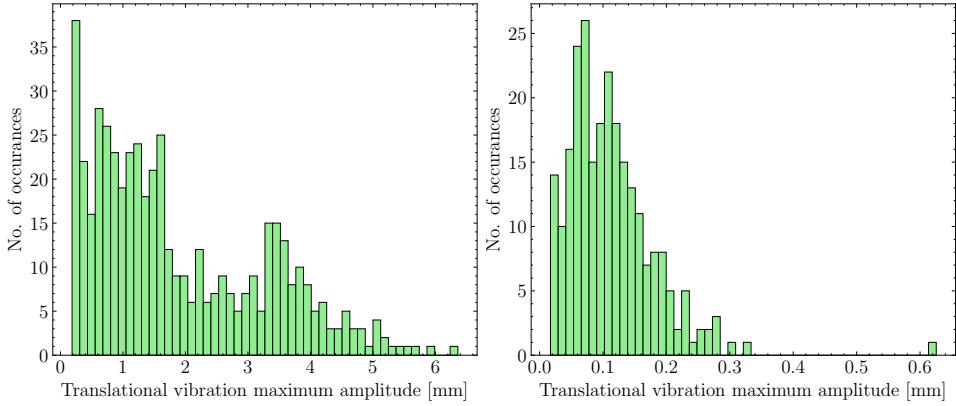
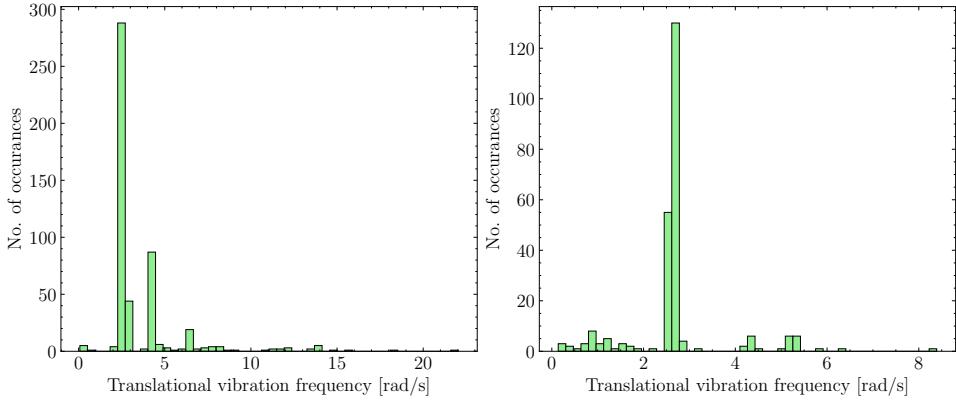


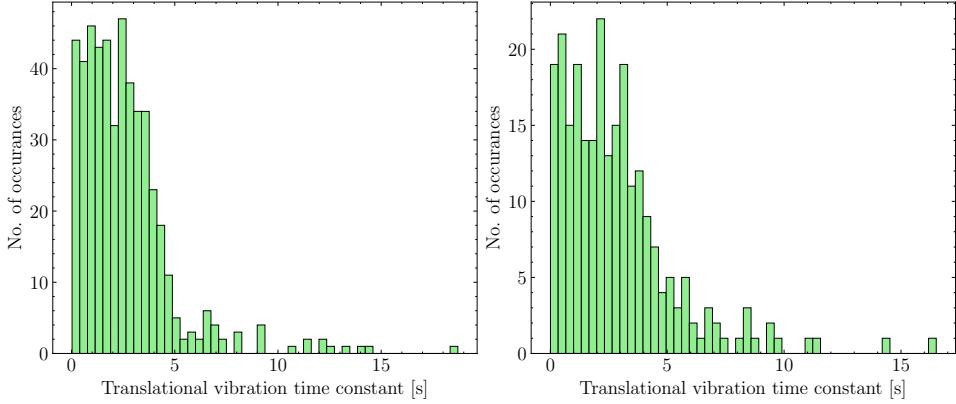
Figure 4.5: Damped vibration in one axis



(a) Distribution of maximum translational vibration displacement amplitudes A in X, Y(left) and Z(right) DoF [mm]

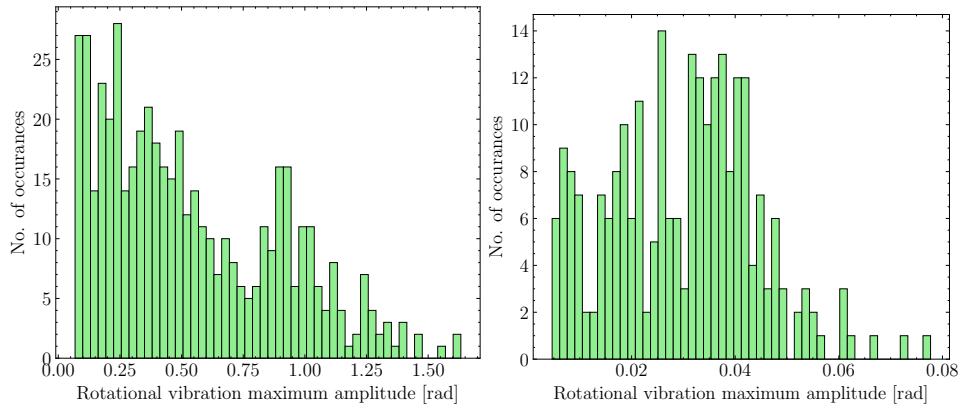


(b) Distribution of translational vibration displacement frequency ω in X, Y(left) and Z(right) DoF [mm]

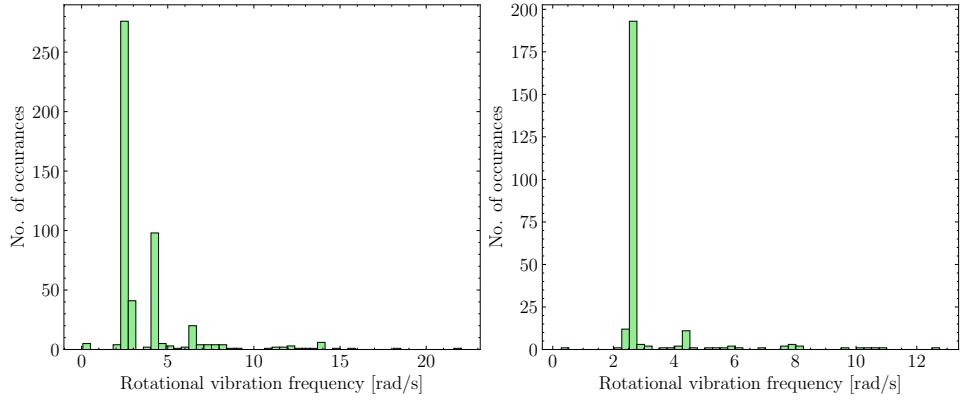


(c) Distribution of translational vibration displacement time-constant τ in X, Y(left) and Z(right) DoF [s]

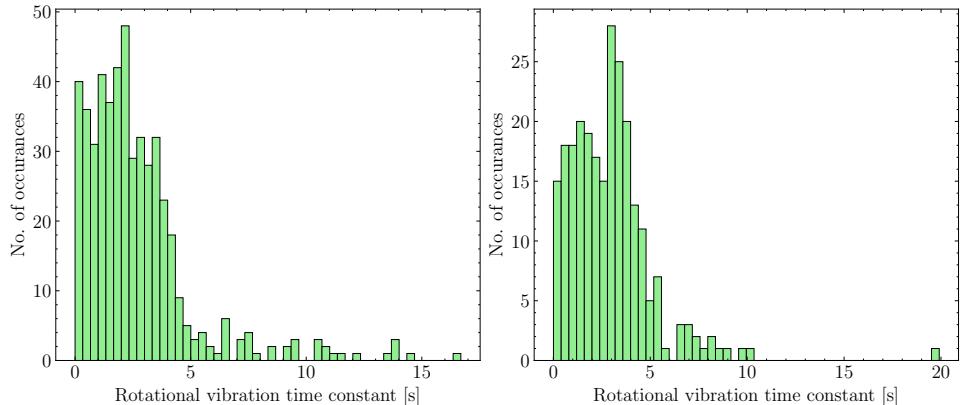
Figure 4.6: Real translational vibration data statistical analysis



(a) Distribution of rotational vibration displacement amplitude θ in X, Y(left) and Z(right) DoF [rad]



(b) Distribution of rotational vibration displacement frequency ω in X, Y(left) and Z(right) DoF [rad/sec]



(c) Distribution of rotational vibration displacement time-constant τ in X, Y(left) and Z(right) DoF [s]

Figure 4.7: Real rotational vibration data statistical analysis

The statistical analysis shown in figure 4.6 and figure 4.7 was done on 248 real world camera setup vibration sequences having an average length of 10 seconds. The results for translational and rotational vibrations are summarised in table 4.3 and table 4.4 respectively. The rotational vibrations have very small amplitude θ and is less than 1 degrees for more than 75 percent of the samples in XY plane. For Z axis the vibrations are even smaller with a mean of 0.0005 degrees. These vibrations are very small and do not have a noticeable visual effect on the video stabilization. But we have decided to account for them to develop a more general solution with multiple applications. The time constant τ for displacement and rotation is similar for all DoF with a mean value of around 2.9 seconds. The frequency ω for all DoF is also similar in the range $2.5rad/sec$ to $3.5rad/sec$. Frequency of $2.5rad/sec$ is the most dominant frequency and seems to be the natural frequency ω_n of the mechanical system (camera setup).

The amplitude A of the translational displacements is the most important characteristic as it is main source of video destabilization. The effect of motion on the quality of video is more pronounced in XY plane as the most pixel shift happens here. In Z axis the mean μ of translational vibrations is 0.113 mm with a standard deviation (σ^2) of 0.07 mm. Even the large displacements in Z seem to have little effect on visual quality of the video. Transnational vibration displacements in XY plane have the mean of 1.9 mm with a standard deviation of 1.36 mm. Vibrations with higher magnitude are also present and need to be accounted for when generating simulated data.

Parameter	Mean (μ)	Variance (σ^2)	Standard Deviation (σ)
A_{xy}	1.89mm	$1.86mm^2$	1.36mm
A_z	0.11mm	$0.0047mm^2$	0.06mm
ω_{xy}	$3.68rad/s$	$5.98rad^2/s^2$	$2.44rad/s$
ω_z	$2.68rad/s$	$2.07rad^2/s^2$	$1.03rad/s$
τ_{xy}	2.59s	$5.18s^2$	2.27s
τ_z	2.83s	$5.87s^2$	2.42s

Table 4.3: Real translational-vibration data distributions

Parameter	Mean (μ)	Variance (σ^2)	Standard Deviation (σ)
θ_{xy}	$0.0094rad$	$3.87e^{-8}rad^2$	$0.0062rad$
θ_z	$0.0005rad$	$6.07e^{-8}rad^2$	$0.0002rad$
ω_{xy}	$3.80rad/s$	$6.36rad^2/s^2$	$2.52rad/s$
ω_z	$3.17rad/s$	$2.62rad^2/s^2$	$1.61rad/s$
τ_{xy}	$2.65s$	$5.83s^2$	$2.41s$
τ_z	$2.93s$	$4.73s^2$	$2.17s$

Table 4.4: Real rotational-vibration data distributions

4.1.2 Simulated Data Generation

The simulated data plays a very important role in our approach as we use the ideal data to validate our stabilization approach. It also empowers us to simulate IMU sensors with different noise levels and characteristics. We also made sure that the data generated from the simulation is fairly domain randomized to make it realistic.

Simulated data is based on the analysed real data with vibration characteristics taken from table 4.3 and table 4.4. The data is then generated using the equations 4.1 and 4.2. A total of 400 vibration sequences with an average vibration duration of 10 seconds were generated with characteristics shown in tables 4.3 and 4.4. Generating a large number of samples ensures the sampling of diverse data from the distribution. This is where we benefit from our simulation the most as the diversity and the size of data ensures our neural network generalizes well.

4.2 Structuring of Data

In classical pose-estimation techniques, every single IMU sample is integrated to the pose estimation individually. Data driven approaches differ in this case as discussed in section 3.2. For data driven approaches instead of taking an individual sample a *window* of samples (figure 4.8) is used as a single sample does not contain enough information to extract meaningful information.

In a supervised learning setting, neural networks require a sufficient amount of data to prevent them from over-fitting to a few samples and to be able to learn the underlying statistics and patterns in the data. Initially we experimented with small window samples (3 - 20 samples) as shown in figure 4.8 a. This did not yield very good results and the pose estimation

error was too high for accurate stabilization. Good results were achieved with window a window size between 70 to 150 samples (example of windows of IMU data depicted in figure 4.8). Larger window sizes did not have an influence on the accuracy but led to increased inference times of the neural network and latency became too high. Hence in the following a window size of 143 was chosen.

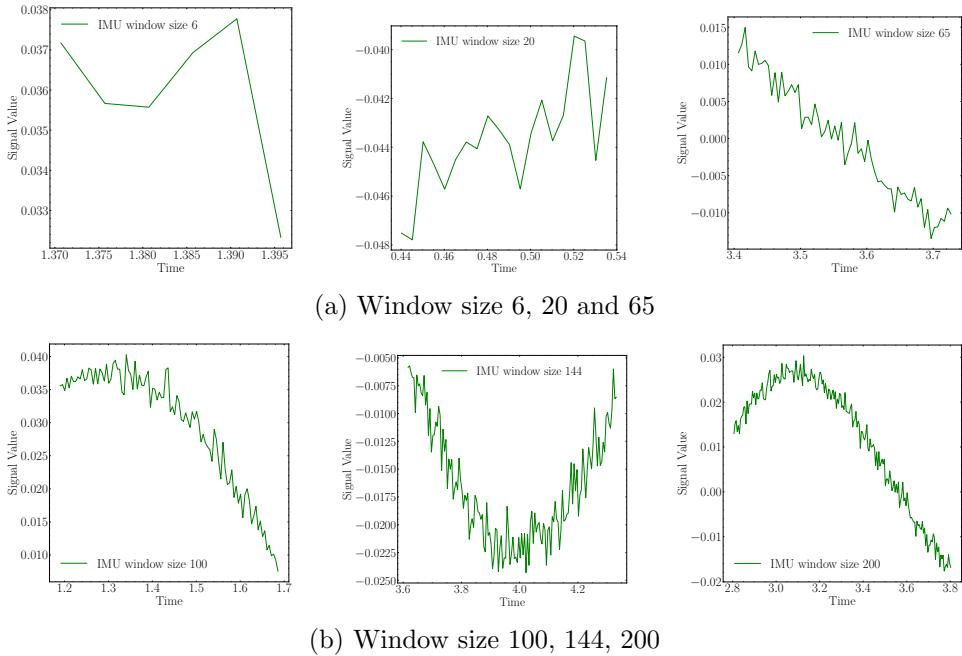


Figure 4.8: Different IMU window samples

4.3 Training and Testing of Various Neural Networks

Various neural network architectures were trained and tested to accurately estimate the camera pose with respect to a stabilized trajectory. The input to the neural network is a window of sensor readings coming from an IMU and the network is trained against the deviation with respect to the stabilization trajectory. The stabilization trajectory can be realised as a mean of the vibration displacement (linear/angular) for an ideal case as shown in figure 4.9.

Let the variable window size be \mathbf{W} . Then based on the employed sensor data (either only accelerometer or accelerometer + gyroscope) the input size to the network is either $(3, \mathbf{W})$ or $(6, \mathbf{W})$ respectively. The use of accelerometer readings is a must because destabilization caused in our use case is

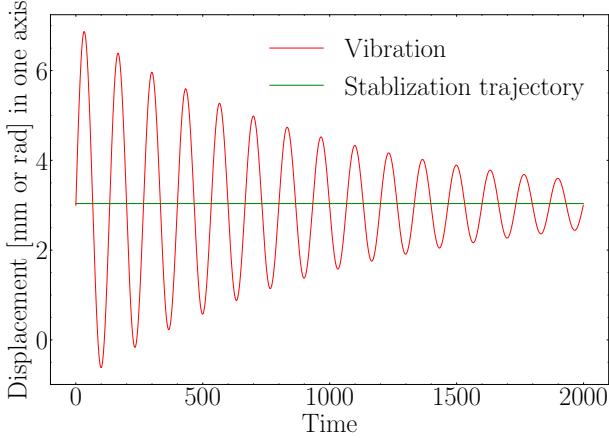


Figure 4.9: Stabilization trajectory (Ideal)

mostly due to translational displacements as discussed in chapter section 4.1.1.

Various neural network architectures like CNN, MLP, LSTM, ResNet, Transformers and their variations were trained and tested for stabilization trajectory regression. Out of these CNN, ResNet and Transformer architectures had acceptable performance.

4.3.1 Data Augmentation

The simulated data is augmented with modified noise and bias characteristics from table 4.2. On each training step randomly generated bias and white noise is added to the true IMU window samples. This results in the network learning to compensate for white noise and bias for a wide variety of noise characteristics as shown by the decrease of average trajectory error by 5 percent.

4.3.2 Training Setting

The training dataset consists of 400 samples of length 10 seconds each. Which means a total of more than 5000 IMU windows W of size $(6, 143)$ are available for training, validation and testing the network. The training, validation and testing had a split of 80:15:5. The network was trained using the Adam optimizer with an initial learning rate of 1×10^{-5} and a learning rate scheduler. Initial learning rate was estimated with PyTorch-Lighting and optimized using grid search. Mean Squared Error (MSE) proved to be the best loss function for this use case and is also used by other data driven pose estimation algorithms (Yan et al., 2018), (Herath et al., 2020).

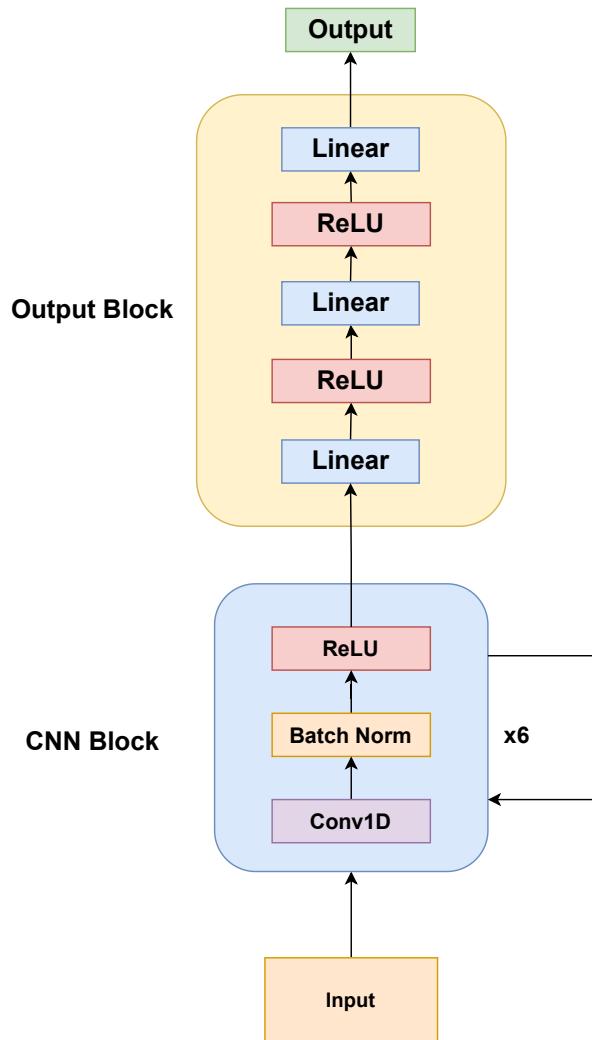


Figure 4.10: Convolutional Neural Network used

Note: In the next sections, only output plots for translational displacement predictions will be shown as the rotation is very small and all the networks correctly predict it with average error in micro-radians range.

4.3.3 Convolutional Neural Network

A convolutional neural network with six stacked Conv1D layers each with batch normalization and ReLU activations is used to extract features from the input data as shown in figure 4.10. Then the output of these convolution layers is regularized during training using **dropout** and down-sampled using **Max-pooling**. Finally, pose is regressed using a series of two fully connected **MLP** layers.

Model Output Analysis

Figure 4.11 shows the model output vs ground truth plot for a 27 seconds long video with translational vibrations in 3 axis (x,y,z). The model can regress stabilization trajectory based on input IMU readings with good precision but there are zones where the prediction too far off and not smooth enough which will case jitter in the stabilization. The error in all axis is depicted in figure 4.12.

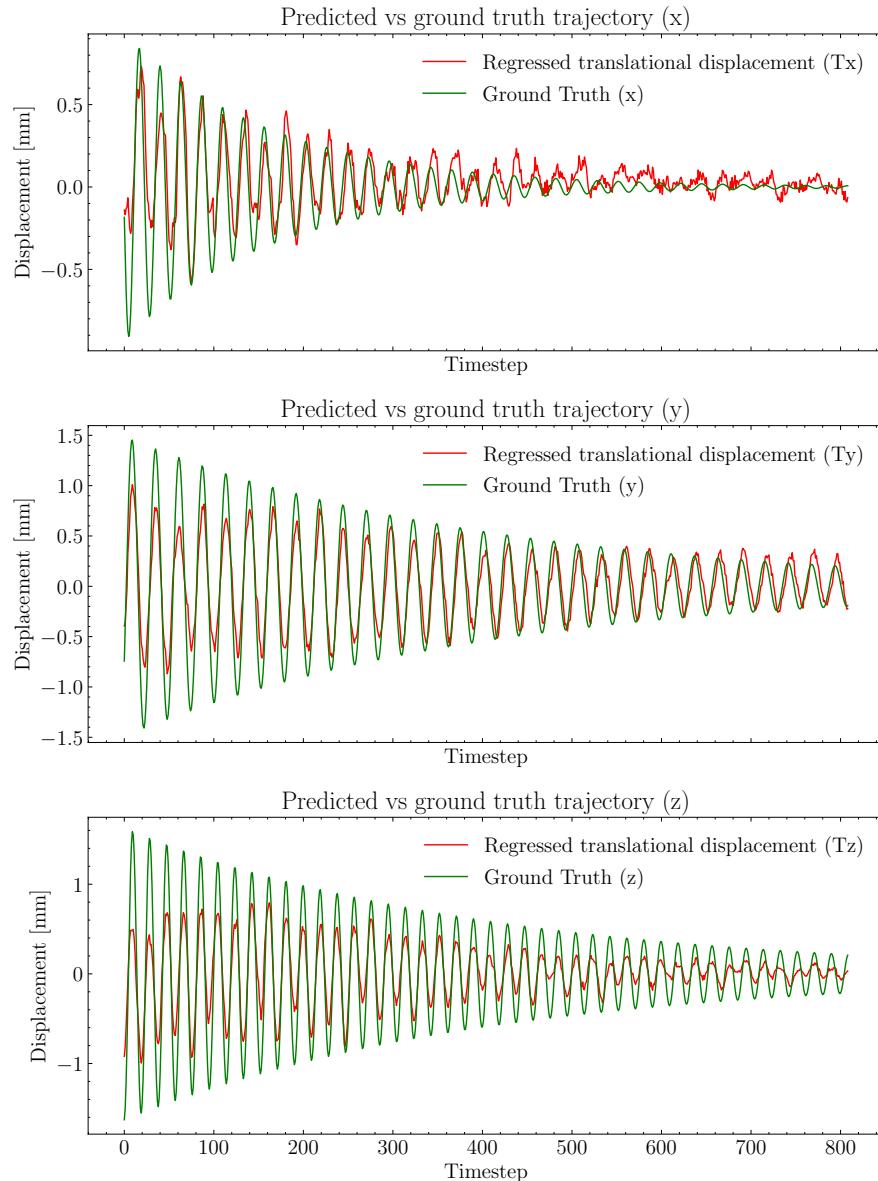


Figure 4.11: Model prediction vs ground truth for CNN

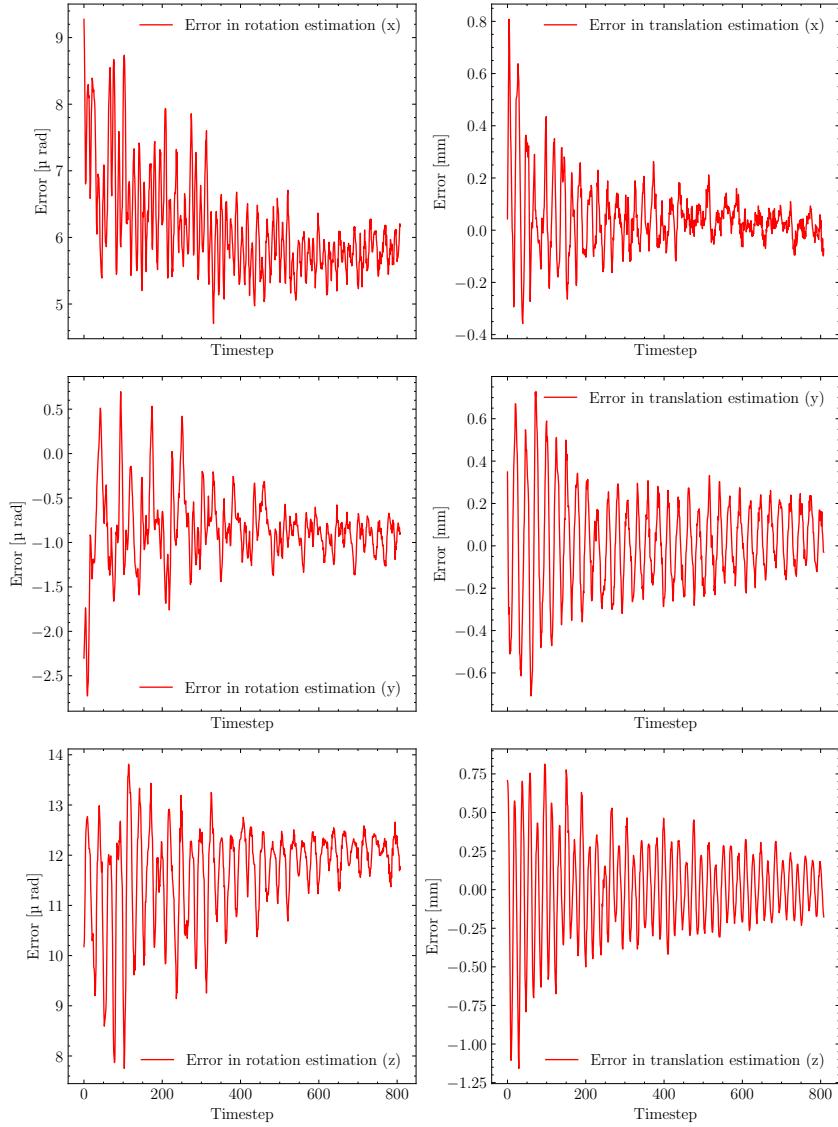


Figure 4.12: Error in each axis for CNN

4.3.4 ResNet

Convolution neural network having **Residual Skip Connections** with six **1D-Convolutional Layers** in sequence with **Batch Normalisation** and **ReLU** activation is used to extract features from the input data. Then the output of these convolution layers is regularized using **dropout** and down-sampled using **Max-pooling**. Finally, pose is regressed using a series of two fully connected **Linear** layers.

Model Output Analysis

Figure 4.14 shows the model output vs ground truth plot for a 27 seconds video three linear DoF (x, y, z). The model can regress stabilization trajectory based on input IMU readings with good precision but there are zones where the prediction is off and thus causes jitter in stabilization. The error in all DoF can be seen in figure 4.15.

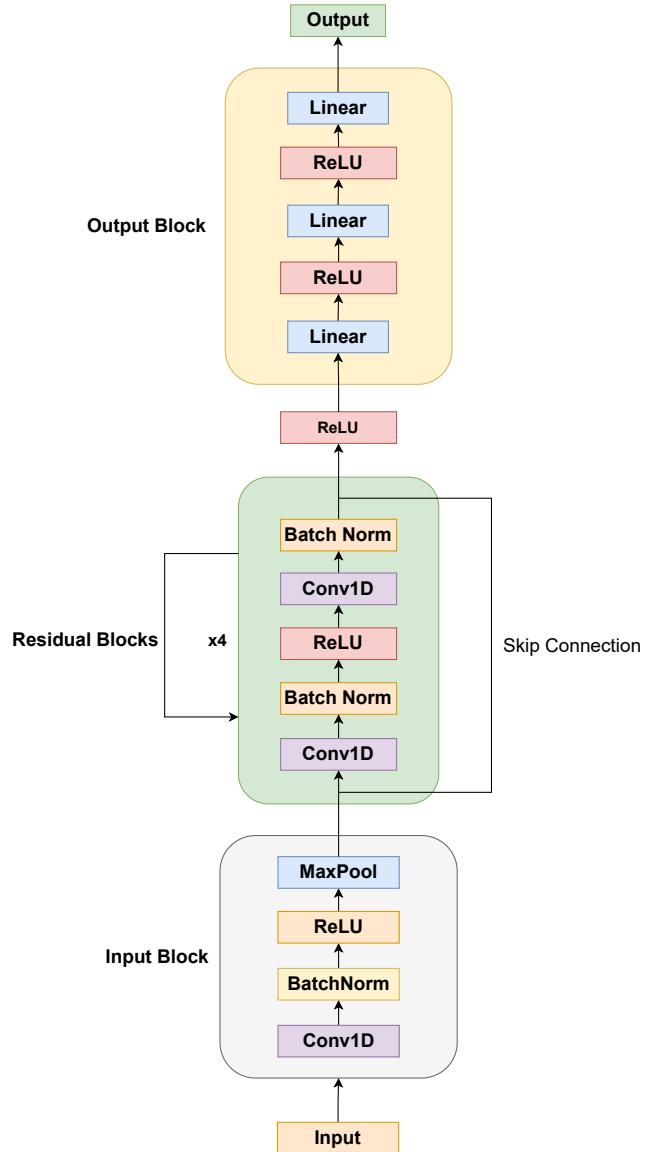


Figure 4.13: CNN with residual skip connections (ResNet)

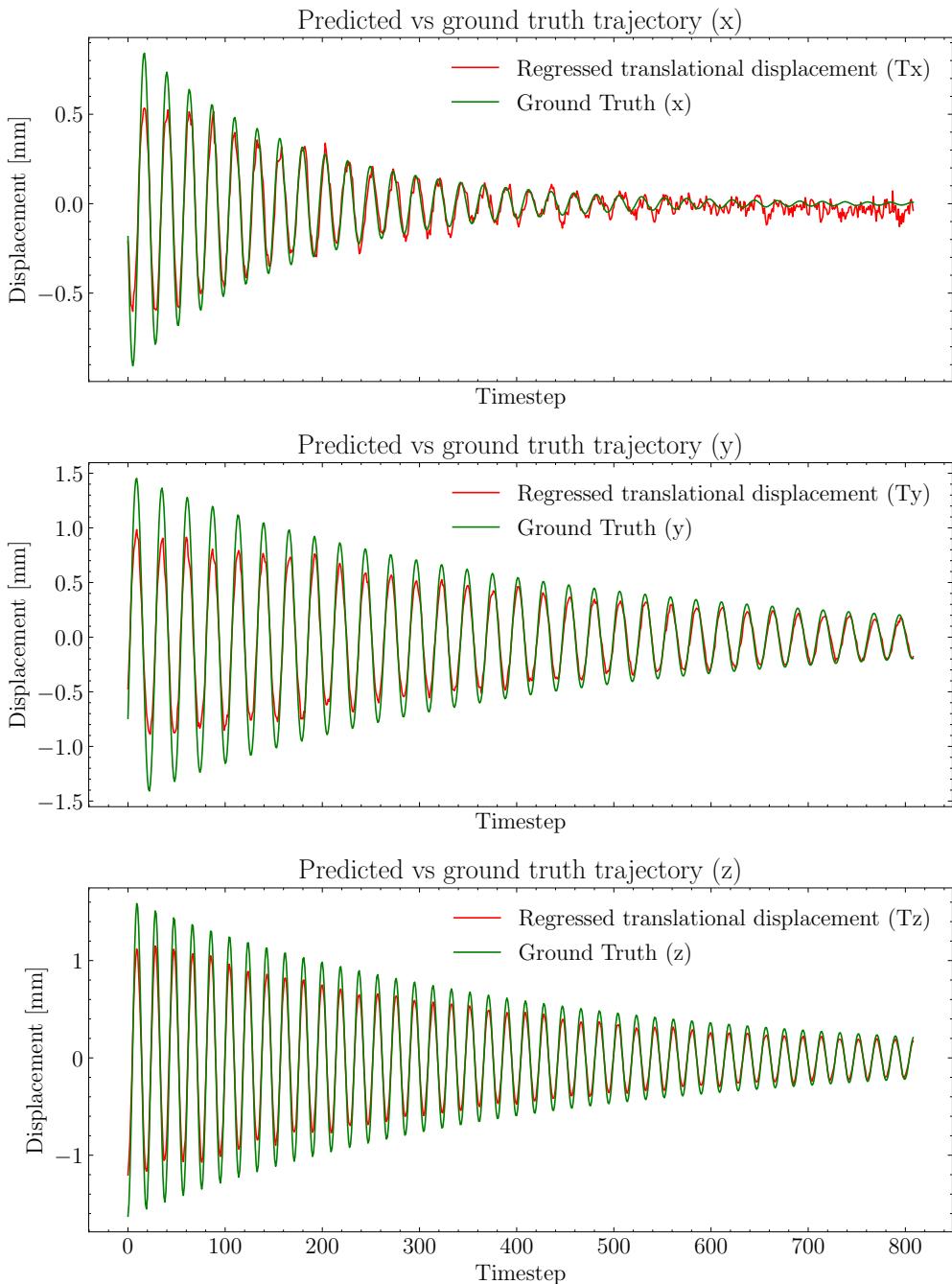


Figure 4.14: Model prediction vs ground truth for ResNet

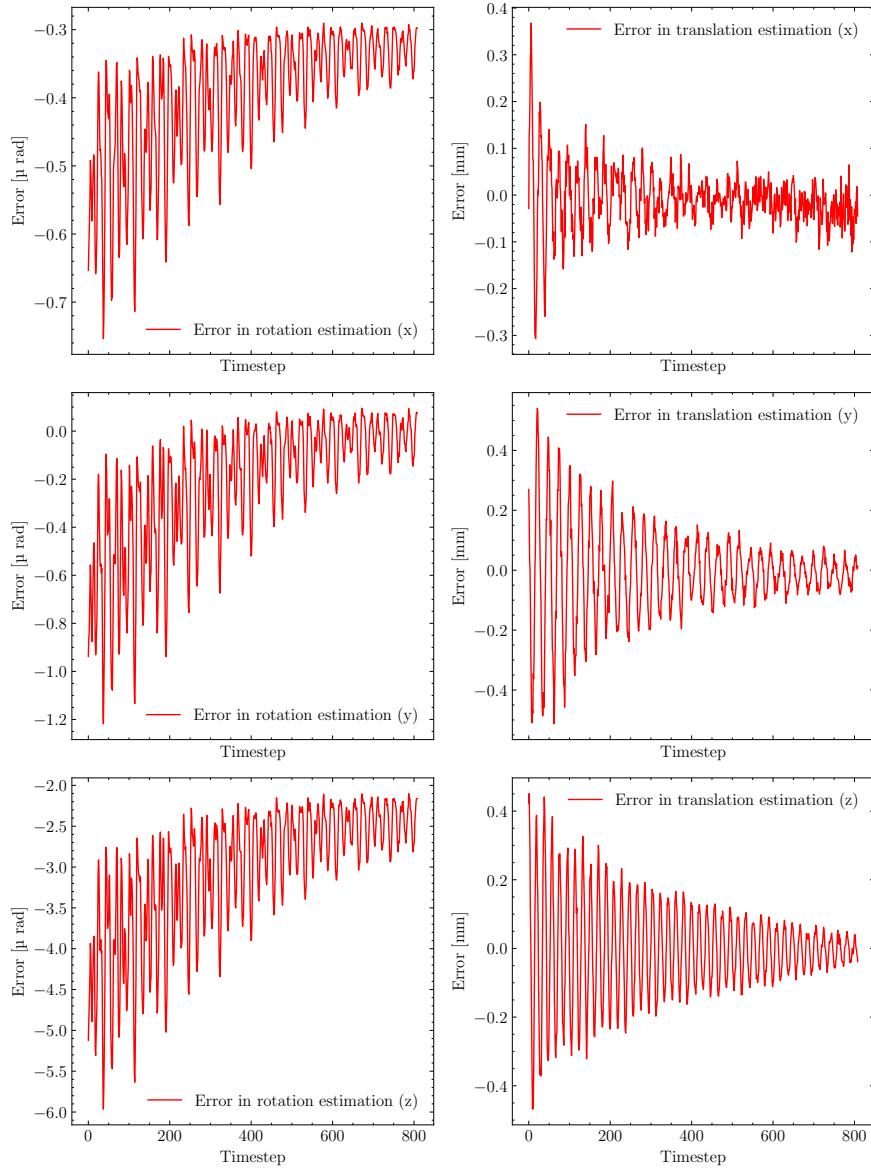


Figure 4.15: Error in each DoF for ResNet

4.3.5 CNN-Transformer

The network is structured in a way that the input goes through four **1D-Convolutional Layers** in sequence with **GELU** activation (non-linear). The output from these convolutional layers is fed to the *Transformer Encoder*. Transformer Encoder has six layers with each layer having **Multi Headed Attention**, **Dropout**, **GELU** and **Layer Norm**. The output from the encoder layer goes into the output layers consisting of **Layer Norm**.

malization, **Linear** layer, **GELU** non-linearity, **Dropout** and finally a **Fully Connected** layer to regress the output pose.

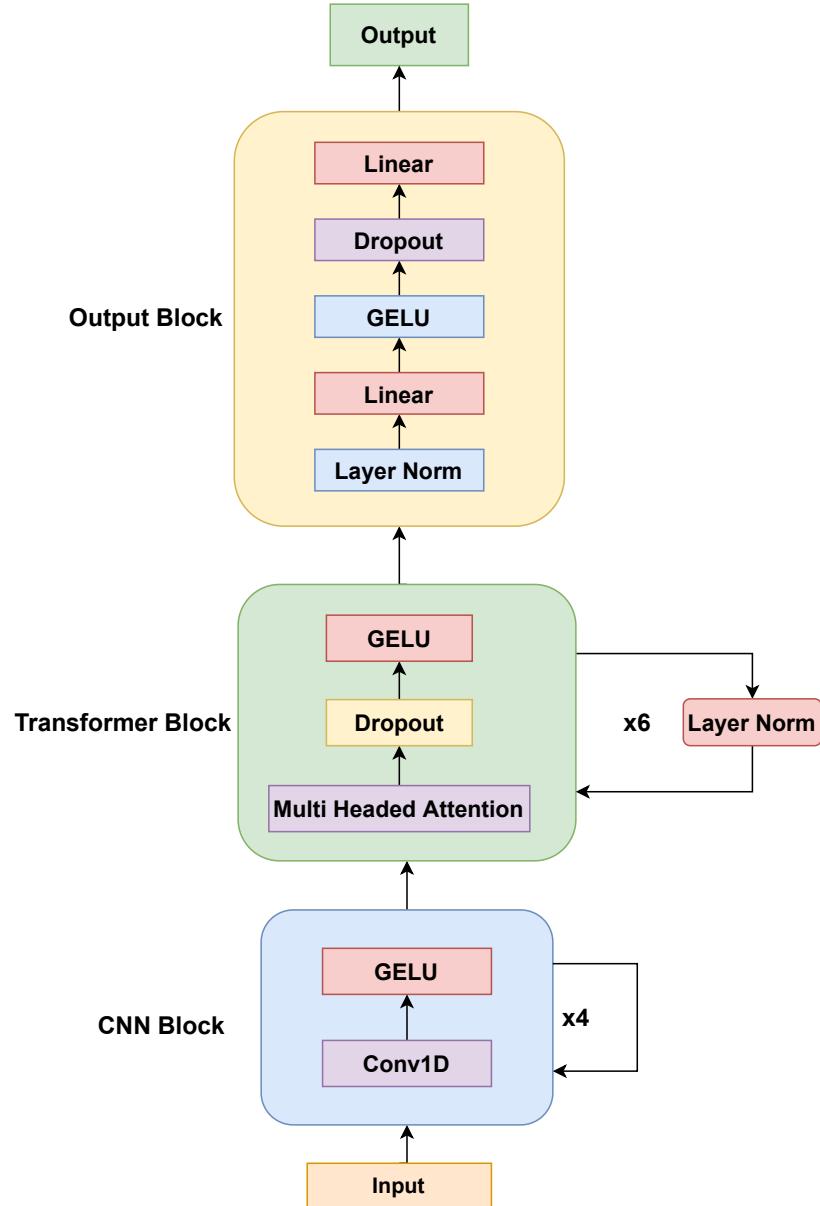


Figure 4.16: CNN-Transformer Network Used

Model Output Analysis

Figure 4.17 shows the model output vs ground truth plot for a 27 seconds video three linear DoF (x, y, z). The model can regress stabilization trajec-

tory based on input IMU readings with very precision and the error is in a very low range of the order of micrometers (μm) and micro radians ($\mu\theta$).

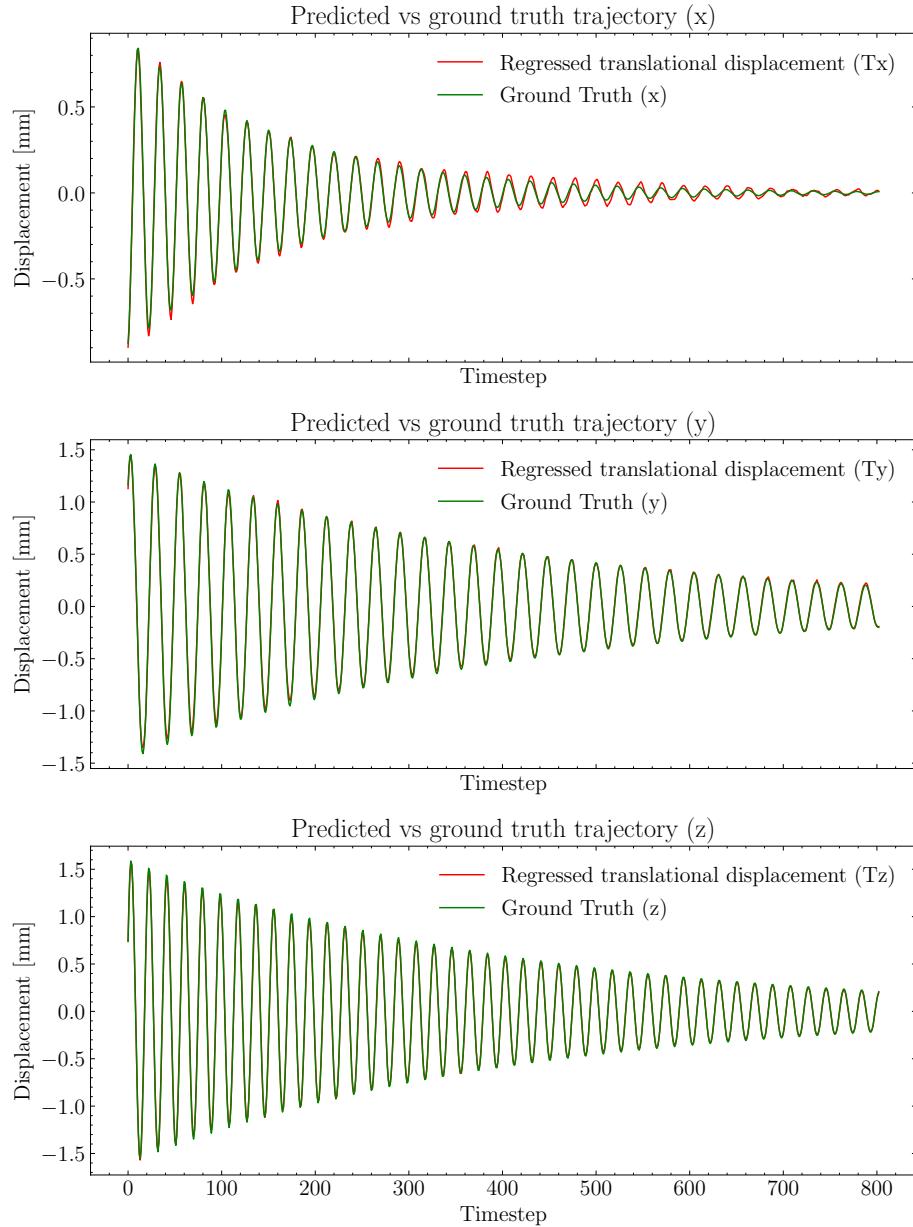


Figure 4.17: Model prediction vs ground truth for CNN-Transformer

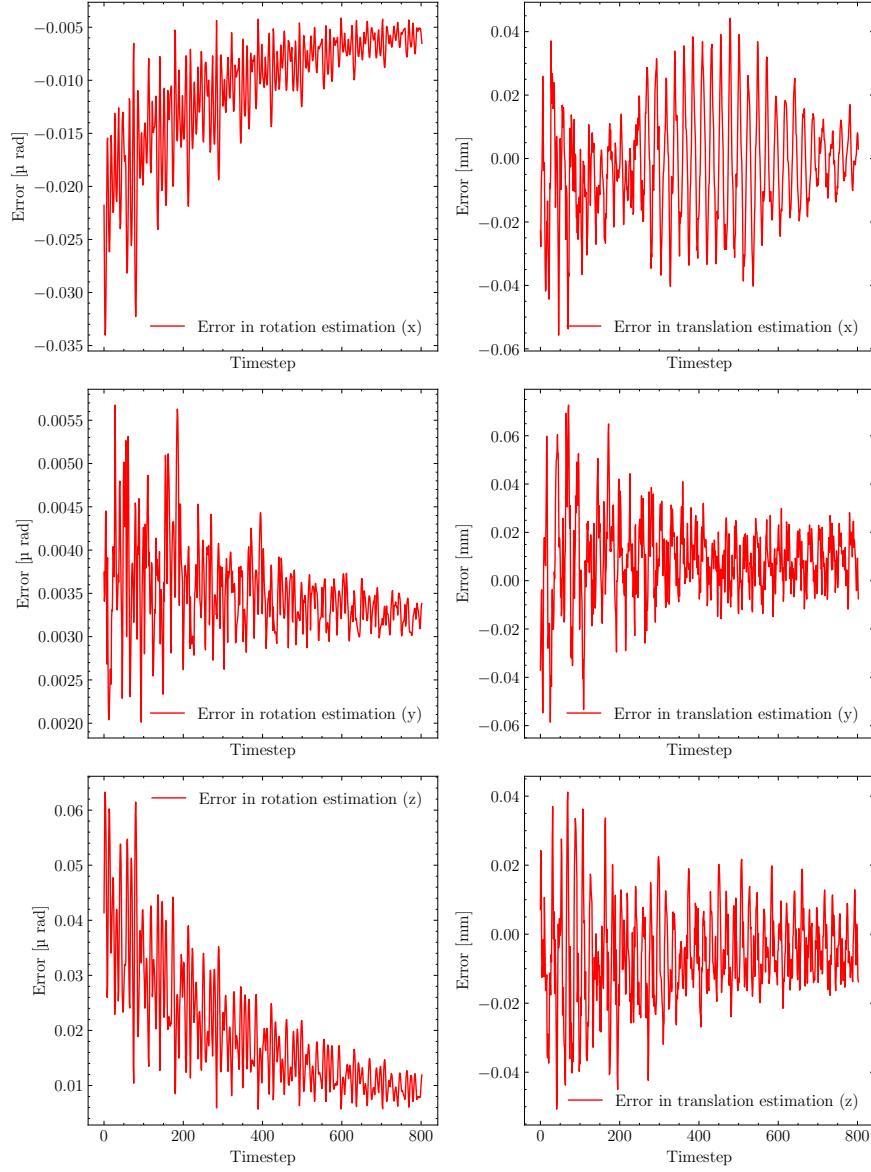


Figure 4.18: Error in each DoF for CNN-Transformer predictions

4.3.6 Smoothing of Stabilization Trajectory

The neural network is able to regress the stabilization trajectory accurately with the required precision. But the regressed trajectory is not smooth as shown in figure 4.19 and has sharp peaks and sudden changes. This is not acceptable as these rough peaks cause jitter in the stabilized video. To account for this, we implemented *Trajectory-Smoothening* in our digital image stabilization pipeline as shown in figure 4.21. An **Exponential Moving Average Filter** (Equation 4.3) is implemented to smooth the trajectory

and minimise jitter. Figure 4.20 shows the effect of this filter on the model output.

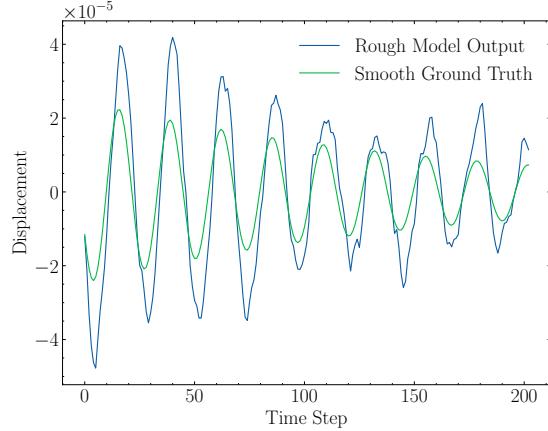


Figure 4.19: Rough regressed vs smooth ground-truth trajectory

$$y_t = x_t \frac{S}{1 + H_l} + y_{t-1} \left(1 - \frac{S}{1 + H_l}\right) \quad (4.3)$$

Where y_t is the smoothed signal and y_{t-1} is the smoothed signal at previous time-step. x_t is the signal value at current time-step. S is the smoothing factor and H is the horizon length. We need to be careful with smoothing the stabilization trajectory because the more we smooth the higher will be the error introduced in the stabilization trajectory.

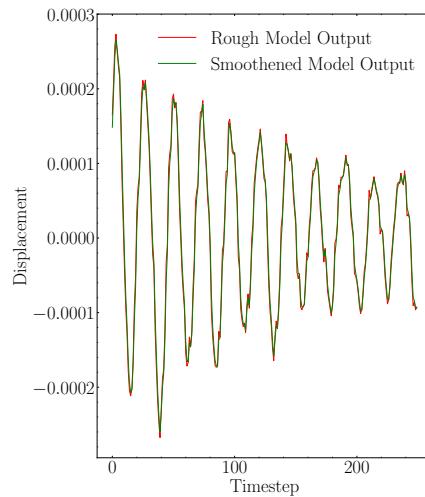


Figure 4.20: Smoothed model output vs raw model output

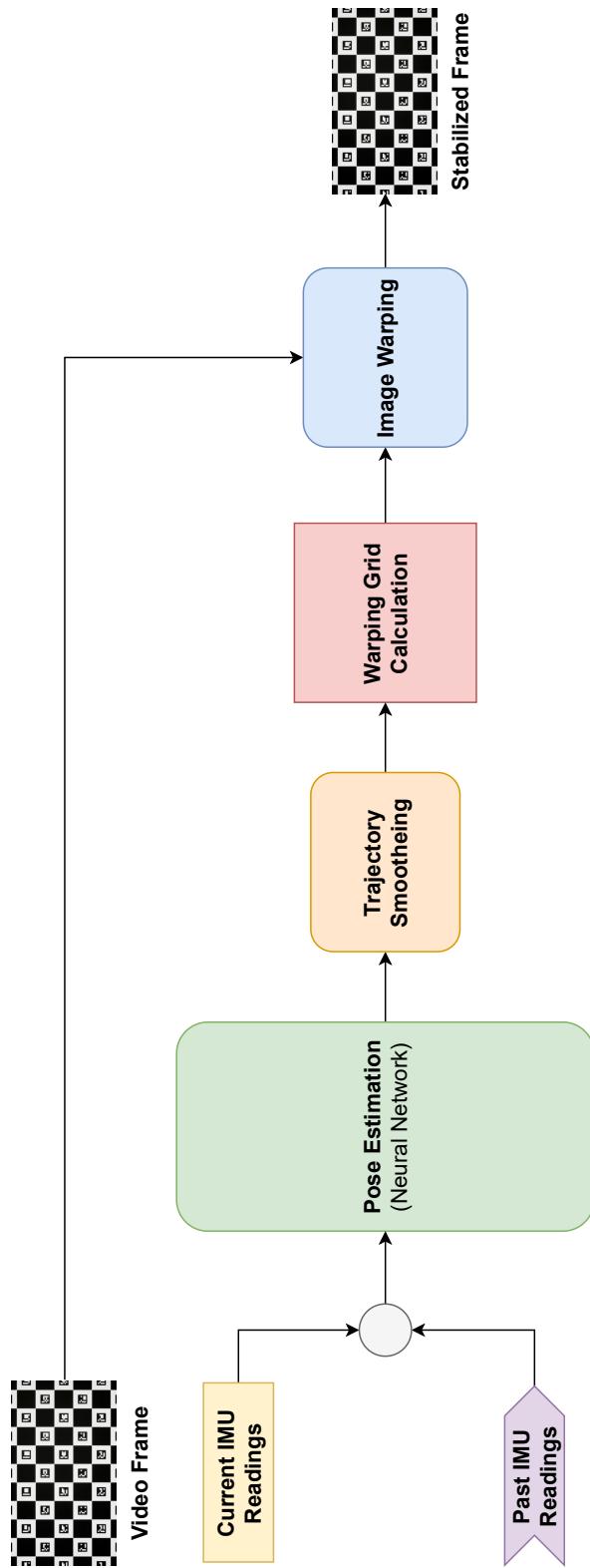


Figure 4.21: DIS pipeline with trajectory smoothing

4.4 Model Deployment

Figure 4.21 shows the digital image stabilization pipeline and our goal is to minimise the time taken at each step. The video (images) are coming at 60 FPS from the camera that has 4K resolution (3840x2160 pixels). We ideally have 1/60 seconds (16.67 ms) to complete all the steps of the DIS pipeline. In this pipeline, model inference and stabilization grid calculation take most of the given time. Stabilization grid calculation is a series of matrix multiplications and has been already optimized. Thus, low inference latency is highly desirable.

4.4.1 Making Models Fast

There are many techniques to reduce the inference latency of neural network models. Using these methods may reduce the performance of a neural network but are essential for deployment as models in their original form may not be usable at all. Some of the popular methods are discussed below.

Quantization

This is a very common approach used to tackle high latency of deep learning models especially in case of edge hardware deployment. We can **quantize** the model by converting the weights from floating point (32-bits) to integers (8 bits). This decreases the memory requirements significantly while also improving CPU and hardware accelerator latency (Gambone, 2020).

Another recent approach is to convert the model to **half-precision** (16-bit floating point). It acts as a middle ground between FP32 and INT8 and the performance trade-off is reduced while decreasing the model size. Another very important thing to note is that not all layers have high latency and thus selective weight quantization for layers like convolution layers can also be done making the performance trade-off even smaller.

Making Models Lean

The field of deep-learning is growing at a fast pace and we have a large selection of various neural network architectures some even with more than 100 Billion trainable parameters. But while developing deep-learning models, our goal should be to use models with the lowest number of parameters and complex layers while keeping the performance acceptable.

4.4.2 Model Inference Latency on various Hardware

Various trained models (CNN, ResNet and Transformer) are checked for inference latency and the results are summarised in table 4.5. Models were

also quantized or converted to ONNX for deployment. The models were inferred 1000 times and the result in the table is an average.

Architecture	Format	No. of Parameters	Inference Latency (ms)
CNN	PyTorch Checkpoint (.ckpt)	745,375	-
CNN	ONNX (.onnx)	00	-
ResNet	PyTorch Checkpoint (.ckpt)	1.85 Million	-
ResNet	ONNX (.onnx)	1.85 million	-
CNN-Transformer	PyTorch Checkpoint (.ckpt)	225,356	-
CNN-Transformer	ONNX (.onnx)	225,356	-

Table 4.5: Inference latency of different models

This chapter contained all the steps we took and the techniques we used to successfully realise digital image stabilization without incorporating any image observations into the algorithm but rather fully relying on inertial measurement data. Various neural network architectures for pose estimation were examined. Throughout the experiments transformer networks outperformed standard CNN architectures for this task. The importance of post-processing the neural network output with a moving average filter to smooth out the jitter was also discussed. In the end a run-time analysis of each architecture was performed that showed that the latency requirements were met.

Chapter 5

Results and Discussion

We had a camera with small FOV high focal length lens which keeps vibrating on its rig after the rig is moved around the scene. Vibrations are very small with a mean translational displacement amplitude of 1.8mm but cause huge pixel shifts in the video recordings thus deteriorating the quality. Our goal was to stabilize the video so that there is no visual discomfort for the viewer and quality of the scene is preserved.

Digital Image stabilization was chosen to stabilize the video for our use case as hardware stabilization is expensive and bulky with a lot of moving parts. A MEMS IMU was chosen for camera motion estimation due to its low price and high reliability. The use of MEMS IMUs presents a huge challenge in accurate pose estimation as the noise characteristics of IMU along with the use of classical inertial navigation algorithm makes it impossible to accurately estimate pose. Even the use of advanced state estimation algorithms is futile.

Data-driven approaches like RoNIN(CNN and ResNet) were tested to estimate camera pose accurately and showed promising results over classical approaches. More advanced self-attention based neural network architectures were used (CNN-Transformer) which improved the accuracy of stabilization trajectory estimation to a very high level. Both real world and simulated data was collected to train the neural networks and test the image stabilization algorithm.

Taking all these steps resulted into very good IMU based digital video stabilization. The networks were able to accurately estimate the stabilization trajectory (camera pose) and did not suffer from drift. Outputs from the neural networks were not smooth causing some jitter in the stabilized video. Exponential moving average filter was used on neural network outputs to smooth the predicted trajectory and improve the quality of video stabilization even more.

5.1 Future Work

- Adapt this stabilization technique to high field of view and low focal length camera setups.
- Explore lean neural network architectures to obtain similar results with low latency networks that are deployable on cheaper hardware.
- Figure out way to use semi-supervised or unsupervised learning techniques to eliminate the hassle of ground truth generation.

Bibliography

- Ahmad, N., Ghazilla, R. A. R., Khairi, N. M., and Kasi, V. (2013). Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262.
- Alatise, M. B. and Hancke, G. P. (2017). Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter. *Sensors*, 17(10):2164.
- Bangera, S. S., Shiyana, T., Srinidhi, G., Vasani, Y. R., and Sukesh Rao, M. (2020). Mems-based imu for pose estimation. In *Advances in Communication, Signal Processing, VLSI, and Embedded Systems*, pages 1–14. Springer.
- Battiato, S., Gallo, G., Puglisi, G., and Scellato, S. (2007). Sift features tracking for video stabilization. In *14th international conference on image analysis and processing (ICIAP 2007)*, pages 825–830. IEEE.
- Censi, A., Fusiello, A., and Roberto, V. (1999). Image stabilization by features tracking. In *Proceedings 10th International Conference on Image Analysis and Processing*, pages 665–667. IEEE.
- Constant, N., Cay, G., Ravichandran, V., Diouf, R., Akbar, U., and Mankodiya, K. (2021). Data analytics for wearable iot-based telemedicine. In *Wearable Sensors*, pages 357–378. Elsevier.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- Ferdinando, H., Khoswanto, H., and Purwanto, D. (2012). Embedded kalman filter for inertial measurement unit (imu) on the atmega8535. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pages 1–5. IEEE.

-
- Gambone, R. (2020). Practical Ways to Speed Up your Deep Learning Model. *Xmart Labs Blog*.
- Gibson, J. J. (1977). On the analysis of change in the optic array. *Scandinavian Journal of Psychology*.
- Grundmann, M., Kwatra, V., and Essa, I. (2011). Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR 2011*, pages 225–232.
- Guilluy, W., Oudre, L., and Beghdadi, A. (2021). Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication*, 90:116015.
- Herath, S., Yan, H., and Furukawa, Y. (2020). Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3146–3152. IEEE.
- Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.
- Jia, C. and Evans, B. L. (2012). Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, pages 203–208. IEEE.
- Kok, M., Hol, J. D., and Schön, T. B. (2017). Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*.
- Liu, W., Caruso, D., Ilg, E., Dong, J., Mourikis, A. I., Daniilidis, K., Kumar, V., and Engel, J. (2020). Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660.
- O’Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- Rao, B., Kazemi, E., Ding, Y., Shila, D. M., Tucker, F. M., and Wang, L. (2022). Ctin: Robust contextual transformer network for inertial navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5413–5421.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Ryu, Y. G. and Chung, M. J. (2012). Robust online digital image stabilization based on point-feature trajectory without accumulative global motion estimation. *IEEE Signal Processing Letters*, 19(4):223–226.

-
- Shavit, Y. and Klein, I. (2021). Boosting inertial-based human activity recognition with transformers. *IEEE Access*, 9:53540–53547.
- Shi, Z., Shi, F., Lai, W.-S., Liang, C.-K., and Liang, Y. (2022). Deep online fused video stabilization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1250–1258.
- Tomasi, C. and Kanade, T. (1991). Detection and tracking of point. *Int J Comput Vis*, 9:137–154.
- Trawny, N. and Roumeliotis, S. I. (2005). Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Woodman, O. J. (2007). An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory.
- Yan, H., Shan, Q., and Furukawa, Y. (2018). Ridi: Robust imu double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 621–636.
- Zheng, X., Cui, S., Wang, G., and Li, J. (2015). Video stabilization system based on speeded-up robust features. In *2015 International Industrial Informatics and Computer Engineering Conference*, pages 1995–1998. Atlantis Press.
- Zhuang, B., Bai, D., and Lee, J. (2019). 5d video stabilization through sensor vision fusion. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4340–4344. IEEE.

Appendix A

First Appendix Chapter

A.1 First Appendix Section

A.1.1 First Appendix Subsection

First Appendix Subsubsection

Declaration of Authorship

I hereby declare that I have written the above Master thesis report independently and that I have not used any sources or aids other than those indicated.

Jena, Germany

Ibad Rather