Focal-Plane Sensor-Processor-Based Visual Inertial Odometry

by

Matthew Lisondra
Honours Bachelor of Science – Physics and Mathematics, University of Toronto, 2021

A thesis
presented to Toronto Metropolitan University
in partial fulfillment of
the requirements for the degree of
Master of Applied Science (MASc)
in the program of
Mechanical Engineering with focus in Mechatronics, MEMS and Robotics Engineering

Toronto, Ontario, Canada, 2024

Author's Declaration For Electronic Submission Of A Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Toronto Metropolitan University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Toronto Metropolitan University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

FOCAL-PLANE SENSOR-PROCESSOR-BASED VISUAL INERTIAL ODOMETRY

Matthew Lisondra

Master of Applied Science (MASc)

Mechanical Engineering with focus in Mechatronics, MEMS and Robotics Engineering

Toronto Metropolitan University, 2024

Abstract

Focal-Plane Sensor-Processors (FPSPs) are an emerging technology that can execute vision algorithms directly on the image sensor. Unlike conventional cameras, FPSPs perform computation on the image plane – at individual pixels – enabling high frame rate image processing while consuming low power, making them ideal for mobile robotics. FPSPs, such as the SIMD Current-Mode Analog Matrix Processor Version 5.0 (SCAMP-5), use parallel processing and are based on the Single Instruction Multiple Data (SIMD) paradigm. In this thesis, Binary Feature Visual Inertial Odometry (BIT-VIO), the first Visual Inertial Odometry (VIO) which utilises SCAMP-5 is presented. BIT-VIO is a loosely-coupled iterated Extended Kalman Filter (iEKF) which fuses together the visual odometry running fast at 300 Frames Per Second (FPS) with predictions from 400 Hz (Hertz) Inertial Measurement Unit (IMU) measurements to provide accurate and smooth trajectories.

Acknowledgments

I would first like to give my sincerest thanks to my research supervisors, Dr. Sajad Saeedi and Dr. Kourosh Zareinia. None of this would have been possible without the incredible guidance and support of both of them. Dr. Sajad Saeedi's research specifically was the first I was exposed to in this exciting and rapidly evolving new field, namely the field of Robotics.

I would also like to thank Dr. Guanghui (Richard) Wang, who taught me the fundamentals of Computer Vision, Dr. Aliaa Alnaggar and the rest of the review committee for reviewing this thesis.

Next, I would like to thank everyone whose research intersected my own. Specifically, I would like to thank Junseo Kim for our research collaboration and fruitful discussions on visual-inertial sensor fusion and visual-inertial SLAM. To Riku Murai especially, for his inspiration and support, guidance and feedback on Focal-Plane Sensor-Processors and the SCAMP-5 technology. To both of them for our long late nights either over calls, across seas and at the university, and to Ali Babaei and Abrar Ahsan as well for their early help in setting up the SCAMP-5.

I would like next to thank everyone whose time I spent with at the the Robotics and Computer Vision Lab (RCVL) including Ishaan Mehta, Mahboubeh Asadi, Nikolaos Kourtzanidis, Hussein Ali Jaafar, Glenn Takashi Shimoda, Robel Efrem, Christopher Kolios, Messiah Esfahani, Navid Zarrabi, Parham Jafary and Haniyeh Altafi for their inspiration and motivation throughout. To Kitty Choi, who was one of the first I contacted at the university, who helped me meet so many of all these special people. For the memories, and the welcoming nature of my research studies. They have all been imperative in my studies and people whom I will always look up to in the highest regard.

I owe my most sincerest to my family. To my mother and my father who worked immensely hard to bring me to this country, who exhausted themselves in the hopes of providing me the equal opportunity to study as intensely and deeply in my interests. To my sister Kristine and my brother James who have believed in me since the beginning, which has allowed me to grow and foster my curiosity since I was young.

Most importantly, I would like to sincerely thank my Trisha, who supported me so much that she would often wait late until two in the morning to pick me up from the lab after research. To her, for her unwavering support throughout it all, which of whom I could not have done any of this without.

Dedication

To my mother, my father, Kristine, James and Trisha

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The need for a lower latent, more efficient, and lower power usage camera technology grows increasingly important onboard mobile robotic systems.

Currently, conventional camera technology typically operates at 30-60 FPS and transfers a non-trivial amount of data from the sensor to the host device (e.g. a desktop Personal Computer (PC)). Such data transfer is not free – in terms of both power and latency –, and additionally, all these pixels must be then later processed on the host device. High computational power is often necessary, and processing resources on the PC host are used extensively. As an alternative, FPSPs, such as SCAMP-5, is a new technology that enables computation to occur on the imager's focal plane before transferring the data to a host-device [1]. By performing early-stage computer vision algorithms on the focal plane such as feature detections, FPSPs compress the image data down to the size of the features. By transferring only the detected features, redundant pixel information is not transferred or even not digitized as FPSPs such as SCAMP-5 can perform analog computation.

The SCAMP-5 has been utilized across many robotic systems, beneficial due to its high frame-rate nature. The high frame-rate nature of SCAMP-5 also opens up the possibility for many interesting applications. For example, in-sensor Convolutional Neural Network (CNN) inference can perform hand gesture recognition for a rock, paper, scissors game at 8000 FPS, and can always make a robot play a winning hand [2]. While it is a simple setup, the game requires the system to have low end-to-end latency and is challenging to replicate using a conventional camera.

Fully utilising the different computational capabilities available on the SCAMP-5 device, Castillo-Elizalde et al. [3] proposed an all-on-sensor mapping and localization framework. It performed visual route mapping and localization on the SCAMP-5 and ran more than 300 FPS on large-scale datasets.

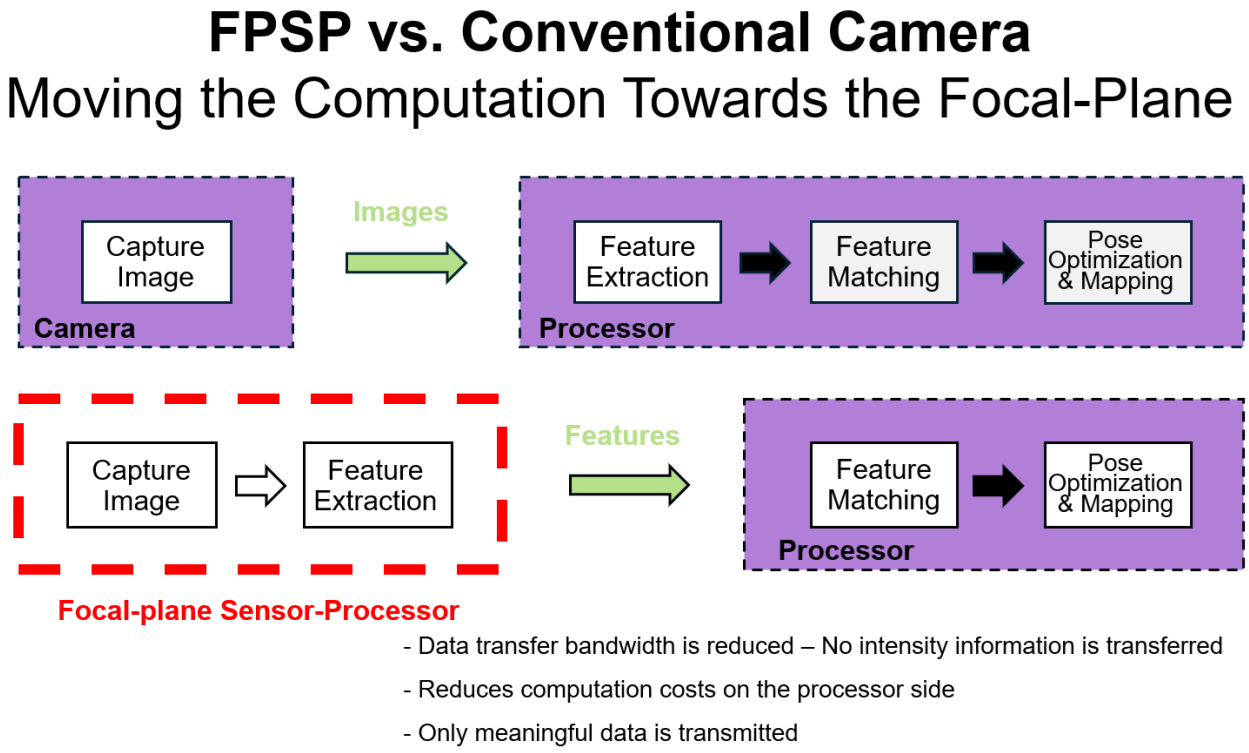


Figure 1.1: Conventional cameras output to the host PC or external processor, often transmitting information it does not need (top). FPSPs alleviate processing on the vision chip itself, ultimately reducing computation costs, and transmitting only meaningful data (bottom).

SCAMP-5 has also been applied for controls, for example, using focal-plane processing, a ground target was detected and tracked to guide a small quadrotor Unmanned Aerial Vehicle (UAV) [4]. In Greatwood et al. [5], they performed drone racing, using the SCAMP-5 to detect the gates. The gate size and location are the only data that was transferred, with minimal data transfer resulting in 500 FPS. In Chen et al. [6], different visual features such as corner points, blobs, and edges were extracted on the SCAMP-5 and fed into a Recurrent Neural Network (RNN) for obstacle avoidance. In-sensor analog convolutions have been proposed in [7] and [8]. AnalogNavNet [9] utilized Cain [10] to implement a CNN that operated on the analog registers on SCAMP-5 for robotic navigation inside a corridor and racetrack environment. Most interestingly, is SCAMP-5's use-case in fast, hostile, more complex state estimation algorithms, such as in dealing with Visual Odometry (VO), which is the process of estimating a robotic system's camera pose by visual information from the camera alone. The high framerate provided by the FPSP would greatly benefit VO systems.

The odometry system by Greatwood et al. performed High Dynamic Range (HDR) edge-based odometry on the SCAMP-5 [11] and achieved lower power consumption than a conventional

camera system. Additionally, the high frame rate nature of the system meant that it suffered less from motion blurs under agile motions. In [12], SCAMP-5 was used to track the 4-Degrees of Freedom (DoF) camera motions using direct image alignment, and all computation was performed on the sensor itself. In [13], another algorithm was proposed, performing optical flow to estimate 4- DoF camera motion. In [14], SCAMP-5 was utilized for visual odometry on UAVs. Compared to using a conventional camera, they demonstrated that SCAMP-5-based systems have a clear practical advantage, for example, by computing HDR on the SCAMP-5, UAVs can transition from outdoor to indoor environments whilst successfully tracking, despite the changes in the lighting conditions. PixelRNN [15] developed an efficient RNN on the vision sensor, reducing that data transfer 256 times, whilst maintaining competitive accuracy. PixRo [16] did frame-to-frame rotational estimation, using local, directly on the chip, pixel information.

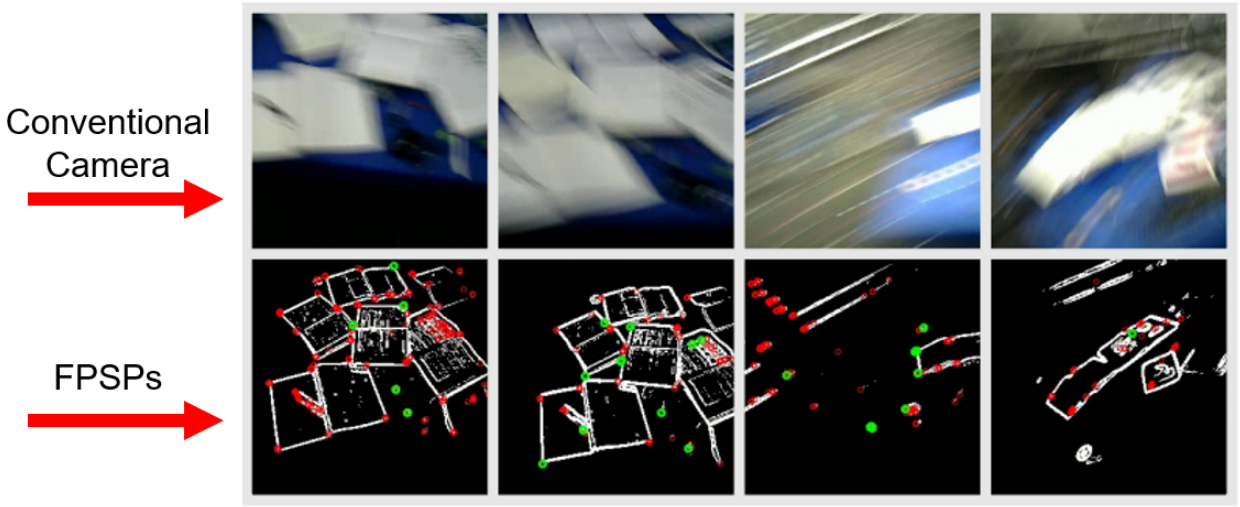


Figure 1.2: A conventional camera suffers from motion-blur, causing poor camera pose tracking (top), while an FPSP-based camera tracks robustly in how it tracks accurately edges and corners in high framerate motions (bottom). Figure from Binary Feature Visual Odometry (BIT-VO) [17] by Riku Murai.

There was a work titled BIT-VO [17], which succeeded 4- DoF visual odometry, able to track a mobile device at the full 6- DoF, operating at 300 FPS using a SCAMP-5 camera. In BIT-VO, the VO system is clearly separated into a frontend, which performs feature extraction, and a backend, which performs the matching of the features and the camera pose optimization. Following this separation, BIT-VO performs the frontend feature detection on the SCAMP-5 camera itself, where corners and binary edges are detected and transferred at 300 FPS utilising the SIMD processing capability of SCAMP-5 and the event readouts.

1.1 Problem Statement

By operating at 300 FPS, BIT-VO [17] is robust against rapid, agile camera motions. However, the estimated trajectory contains a high-frequency noise, which is due to the noisy feature detection on the focal plane. This thesis aims to address this problem. Succeeding state estimation from VO, there is VIO, which is sensor fusion of both visual, as well as inertial measurements from an onboard sensor known as an IMU for even more accurate estimation. The SCAMP-5 has not yet been incorporated on such a system. Extending on the previous work BIT-VO [17], this thesis presents BIT-VIO, the first 6- DoF VIO algorithm to utilize the advantages of the FPSP for vision-IMU-fused state estimation.

1.2 Objectives

The objectives of this thesis include:

1. **Study the usability and advantages of FPSPs to leverage a more accurate state estimation framework**: Investigate the use-case advantages of using a FPSP-camera over conventional cameras, especially in the state-of-the-art VO and VIO context.

2. **Design an algorithm for VIO using Focal-Plane Binary Features**: Succeeding off of the state-of-the-art 6- DoF FPSP VO, design an algorithm that allows for IMU fusing for more accurate estimation.

3. **Implement the FPSP vision- IMU-fused estimation algorithm on a mobile device for offline and online real-world testing**: Implement the designed algorithm in a offline, as in post-processing the data via algorithm feeding after raw data taking, and online, as in real-time, real-world experiment context.

4. **Evaluate the performance, benchmarking against FPSP vision-alone and ground-truth data**: Evaluate the performance of the algorithm by benchmarking it against FPSP VO and ground-truth data acquired from a motion capture system.

1.3 Significance of Work

The work presented in this thesis is significant as it presents a novel design and implementation of the first 6- DoF VIO algorithm which utilizes the advantages of the FPSP for vision- IMU estimation. This thesis compares the use-case advantages of FPSPs, bottlenecks of conventional cameras, overall showcasing the superiority over the latter. This thesis also presents an evaluation of the algorithm's performance with state-of-the-art 6- DoF FPSP VO and ground-truth data.

1.4 Contributions

The contributions of this thesis include:

1. **Efficient VIO operating and correcting by loosely-coupled sensor-fusion iEKF at 300 FPS using predictions from IMU measurements obtained at 400 Hz**: In Sec. 3.3 to Sec. 3.7 a detailed derivation and formulation of the FPSP visual-inertial algorithm is established.

2. **Uncertainty propagation for BIT-VO's pose as it is based on binary-edge-based descriptor extraction, 2-Dimensional (2D) to 3-Dimensional (3D) re-projection**: Sec. 3.5 goes over the camera pose measurement used in the update of the iEKF, while Sec. 3.6 goes over the uncertainty propagated on this 6- DoF pose.

3. **Extensive real-world comparison against BIT-VO, with ground-truth obtained using a motion capture system**: Sec. 3.10 to Sec. 3.11 presents the experimental results, the visual-inertial-sensor calibration done, the overall accuracy and robustness, as well as benchmark comparisons.

4. **Extensive study on the algorithmic execution timing/frame, accuracy, memory usage and power consumption of the visual front-end processing performance on the FPSP**: Sec. 3.12 showcases the on-sensor processing performance of the FPSP, comparing with the processing of algorithms using conventional cameras, ultimately showcasing how FPSPs outperform them.

Chapter 2

Literature Review and Background

In this chapter, the background of this work will be covered. Sec. 2.1 begins with the SCAMP-5 FPSP and Sec. 2.2 goes over applications to broader robotics and mobile systems. Sec. 2.3 then goes over 6- DoF FPSP visual odometry, BIT-VO [17]. Sec. 2.4 then covers the different types of state estimation algorithms with special attention in going over odometry. Sec. 2.5 then goes through VIO, as well as explaining loosely-coupled and tightly-coupled visual-inertial frameworks. As well, the three different paradigms of VIO are covered: filtering, fixed-lag smoothing and full-smoothing. The chapter ends with a note on the ideal framework for FPSP-vision- IMU-fusion, as well as other competitor unconventional cameras for estimation.

2.1 SCAMP-5 Focal-Plane Sensor-Processor

Conventional camera technology generally takes in information of an environment, transferring data over to a digital processing hardware such as PC host or external computer where it is then further processed and used for vision-based applications [1]. High computational power is often necessary and processing resources on the PC are used extensively. FPSPs are a new technology where much of the computational load can be off-put by allowing image and signal processing to be done on the chip, before transferring to a PC host or other external device to be further processed, if still necessary [1]. Pixel data processing is done directly on the sensor chip, providing lower latency (high FPS) and power consumption that would have incurred had more communication been needed to be done between the sensor and processor if the processor was on some PC host or external computer. The FPSP chip is capable at its maximal potential to track a moving object at $100,000$ FPS [1]. What is communicated is only meaningful data at the transmission stage,

relevant to the vision algorithm. The sensor device's pixels, also known as Processing Elements (PEs) are based on the differing idea of building "artificial retinas" operating in a biological and continuous way, and instead building them via the parallel architecture of SIMD [1].

A particular FPSP technology of great interest is the SCAMP-5 camera technology [18], which contains a 256×256 processor array, doing parallel-processing on a SIMD processor array into the sensor device's pixels. The parallelism architecture of the SCAMP-5 FPSP camera technology helps to provide large computing power and high efficiency.

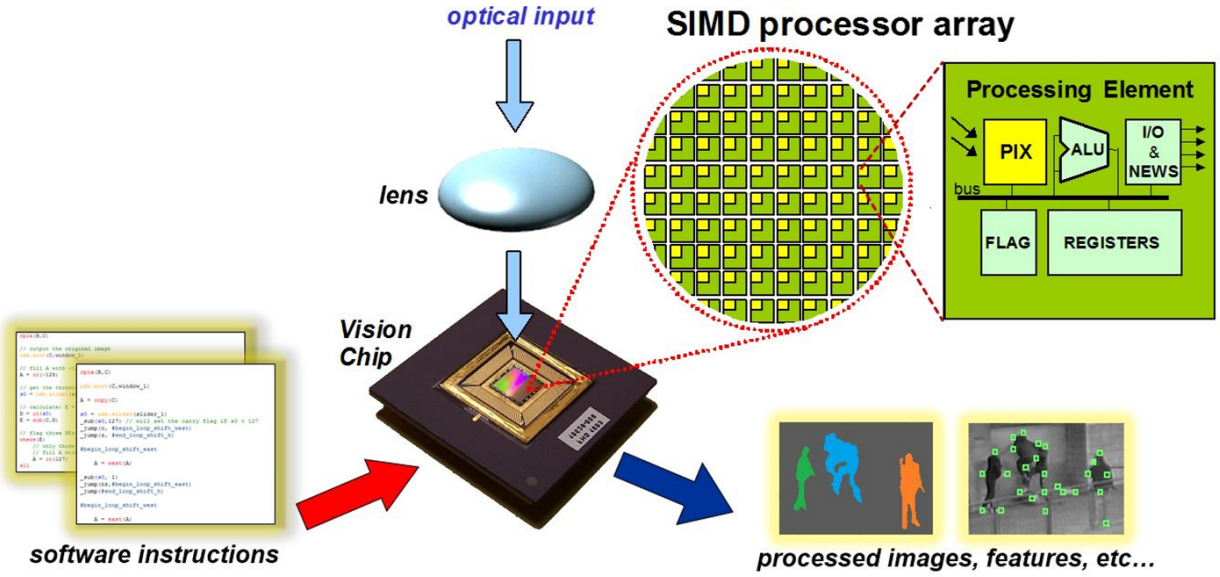


Figure 2.1: SIMD architecture on the FPSP vision-chip which allows for low latency due to parallel computation capabilities. Each PE doing SIMD is complete, capable of acting as each mini computer. Software instructions are burned on each PE seamlessly. Figure from Piotr Dudek [1].

Each PE contains 6-7 analog registers and 13 Dynamic Random-Access Memory by Digital Registers (DRAM) and an ALU which means they are capable of performing logical and arithmetic operations [18]. The analog registers are capable of storing a real-valued variable, able to store around 250 different quantity values, like of the 8-bit nature. The analog registers can do operations such as add, negate, split and compare-against-0. The DRAM are capable of storing 13 binary values, with 1 being special in having influence over the analog registers [18]. The DRAM can do operations such as MOV, OR, NOR and NOT. Each PE can communicate EAST, WEST, NORTH, SOUTH with its neighbouring PEs analog registers and DRAM. The SCAMP-5 FPSP camera technology is designed to have additional benefits on top of its parallel architecture of SIMD [1].

As each PE contains 13 DRAM, events can be read-out and the coordinates of the events can be scanned so that the time cost is only proportional to the number of events [18]. In a conventional camera, the time cost would have been a lot more, being proportional to the scanning area taken. Next, flooding is digital register propagation of 1s used for hardware acceleration [18]. Through the SCAMP-5 FPSP camera technology's asynchronous propagation network, the flooding speed is much faster than in a conventional camera, almost $62\times$ more [18]. In essence, the SCAMP-5 FPSP camera technology has the advantage of providing lower latency with lower power usage on its PC host or external computer [18].

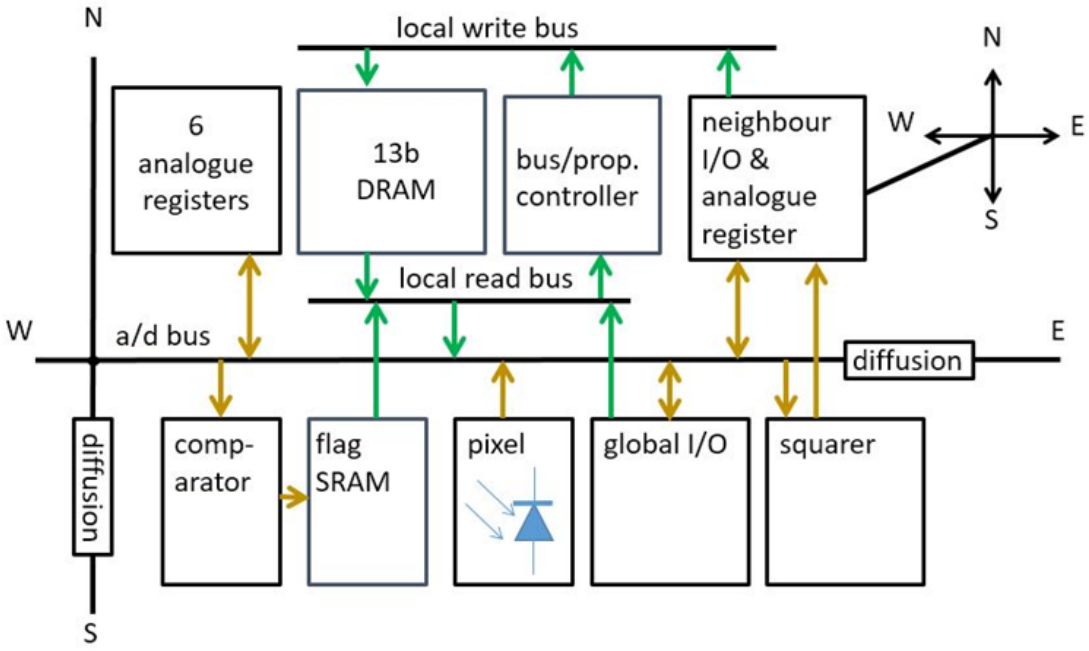


Figure 2.2: Low-level breakdown of the design of the SCAMP-5 FPSP PEs. Each PE consists of 13 DRAM and 6 analogue registers. Each PE can communicate with its 4-neighbours for information passing. Figure from Piotr Dudek [1].

2.2 Application of SCAMP-5 to Robotics

The SCAMP-5 has been utilized across many robotic systems. Greatwood et al. proposed an odometry that performed HDR edge-based odometry on the SCAMP-5 [11] with achieved lower than conventional camera technology-level power processing. Motion blurring was alleviated by the high frame rate tracking capability of the FPSP. 4- DoF camera tracking was done on the FPSP via image-alignment, directly, all on-sensor [12]. There was another competitor 4- DoF

odometry [13] and this used base-vectors from the FPSP to determine the optical-flow motions. An application of FPSPs to high frame rate tracking of UAVs was done in [14]. SCAMP-5 systems are shown to be advantageous, especially for UAVs in how the high framerate HDR tracking allows tracking despite hostile changes in lighting.

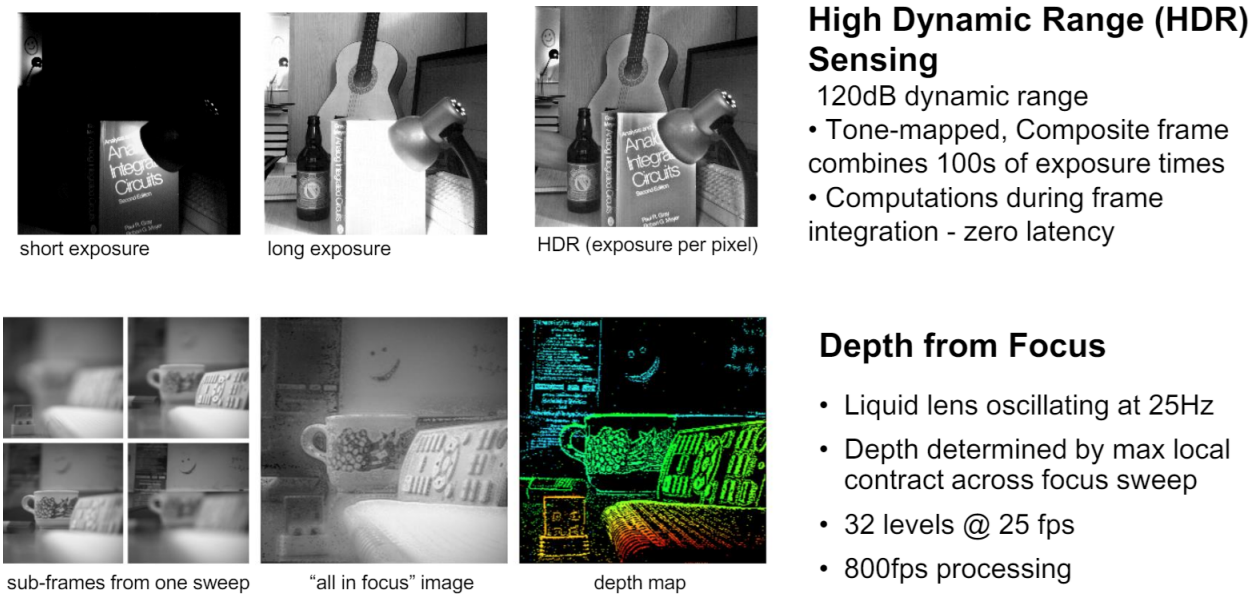


Figure 2.3: HDR done with zero latency on the FPSP, as well as depth map extraction at 800 FPS. Figure from CVPR 2019 Workshop on Event-based Vision and Smart Cameras by Pitor Dudek [19].

The high frame-rate nature of SCAMP-5 also opened up the possibility for many interesting applications. For example, in-sensor CNN inference could perform hand gesture recognition for a rock, paper, scissors game at 8000 FPS, and could always make a robot play a winning hand [2]. While it is a simple setup, the game required the system to have low end-to-end latency and was challenging to replicate using a conventional camera. Fully utilising the different computational capabilities available on the SCAMP-5 device, Castillo et al. [3] performed all-on-sensor mapping and localization. It performed visual route mapping and localization on the SCAMP-5 and runs at more than 300 FPS on large-scale datasets. SCAMP-5 has also been applied for controls, for example, using focal-plane processing, a ground target was detected and tracked to guide a small, agile quadrotor UAV [4]. In [5], they performed drone racing, using SCAMP-5 to detect the gates. The gate size and location are the only data transferred, with minimal data transfer resulting in 500 FPS. In [6], different visual features such as corner points, blobs, and edges were extracted on the SCAMP-5 and fed into a RNN for obstacle avoidance. In-sensor analog convolutions have

been proposed in [7] and [8]. AnalogNavNet [9] utilized CAIN [10] to implement a CNN that operated on the analog registers on SCAMP-5 for robotic navigation inside a corridor and racetrack environment. PixelRNN [15] developed an efficient RNN on the vision sensor, reducing that data transfer 256 times, whilst maintaining competitive accuracy. PixRo [16] did frame-to-frame rotational estimation, using local, directly on the chip, pixel information.

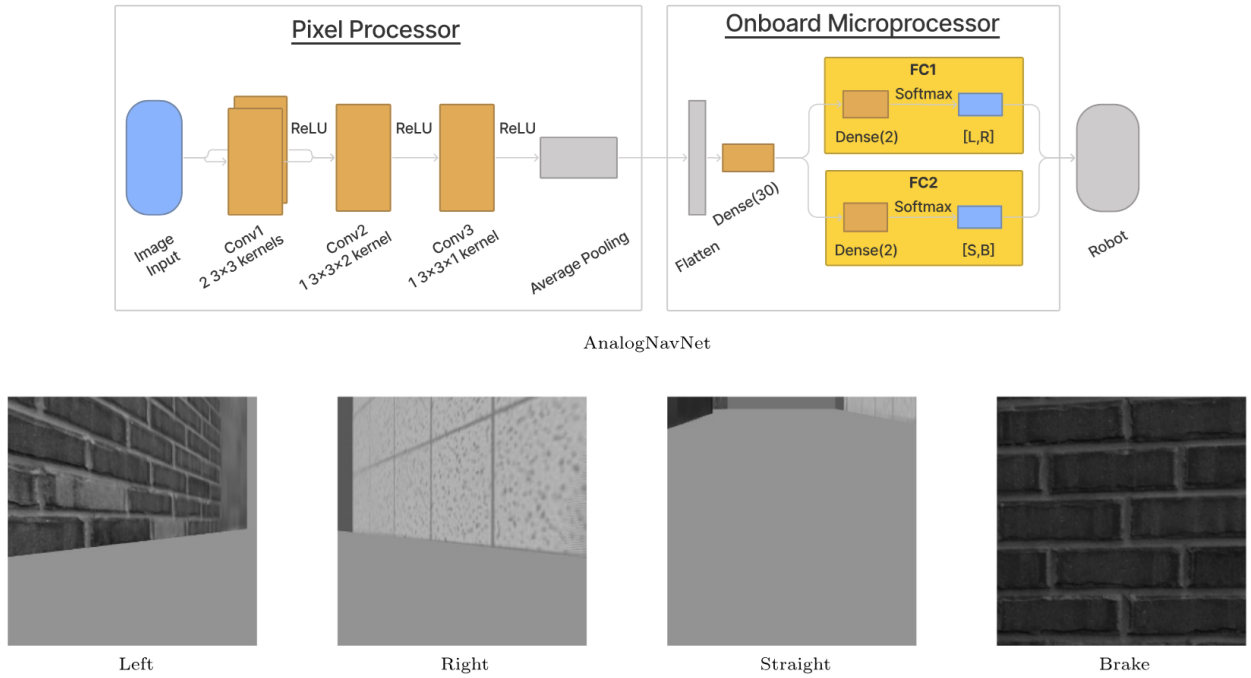


Figure 2.4: AnalogNavNet: CNN on FPSP vision chip itself for robot navigation. A simple 3-layer RNN is on the vision-chip with dense layers, softmax for controls to robot on microprocessor. Figure from CAIN [10].

2.3 BIT-VO

The algorithm BIT-VIO builds on the previous work BIT-VO [17], which performed 6- DoF visual odometry at 300 FPS using a SCAMP-5 camera. In BIT-VO, the VO system is clearly separated into a frontend, which performs feature extraction, and a backend, which performs the matching of the features and the camera pose optimization. Following this separation, BIT-VO performs the frontend feature detection on the SCAMP-5 camera itself, where corners and binary edges are detected and transferred at 300 FPS utilising the SIMD processing capability of SCAMP-5 and the event readouts. The detected features are then transferred to a host device, which performs the backend processing. Here, for each corner feature, a descriptor is formed using the binary edges.

Using brute form matching, corners across frames are matched using the descriptors, similarly to the Oriented Rotated BRIEF (ORB) descriptors in ORB-SLAM [20]. Once the correspondences are established, the system is initialised using a 5-Point RANSAC Homography for Determining 6 DoF Camera Pose [21] algorithm and after the initialization, the camera pose is optimized by minimising the map-to-frame reprojection error.

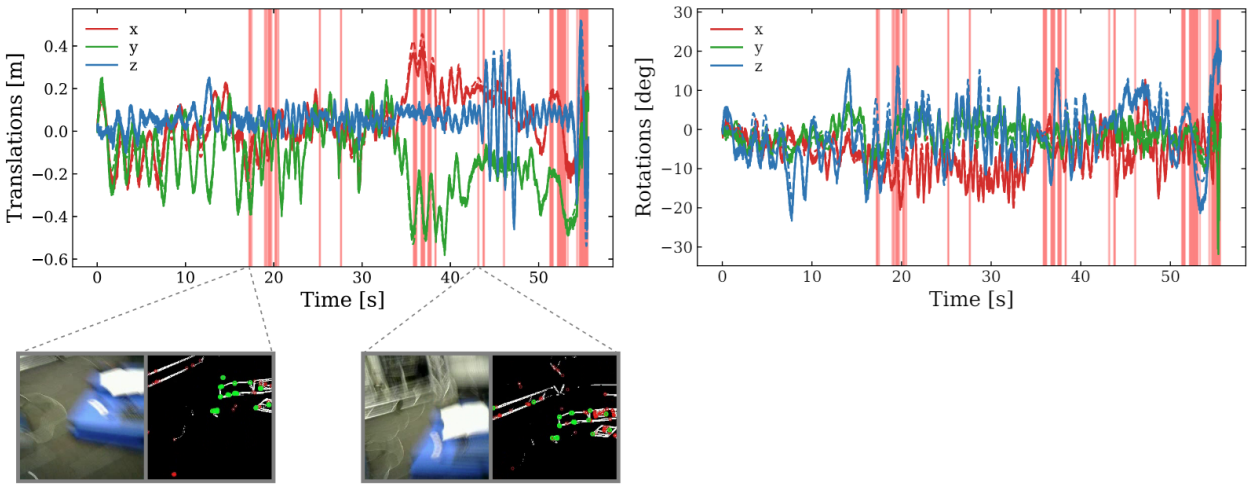


Figure 2.5: BIT-VO vs ORB-SLAM. The translational (left) and rotational (right) tracking of a BIT-VO trajectory is given, with red vertical lines representing where ORB-SLAM fails due to the fast, hostile motions of the trajectory. Figure from BIT-VO [17].

By operating at 300 FPS, BIT-VO is robust against rapid, agile camera motions. However, the estimated trajectory contains a high-frequency noise, which is due to the noisy feature detection on the focal plane. This thesis aims to address this problem by incorporating IMU measurements.

There have been many implementations of the SCAMP-5 FPSP on robotic systems. Next, will be going through a deep literature review on the main robot state localization estimation methods, leading to VIO, which will lead to the aim of this research to utilize the advantages of the FPSP.

2.4 Related Works

Robot state estimation is the process of estimating the internal state variables of a dynamic system based on available measurements. The state represents a set of variables that describe the system's behavior and current condition [22].

There are three major types of methods for robot state localization estimation and they are as follows: firstly, estimation by 'Apriori' Mapping. Next, Simultaneous Localization and Mapping (SLAM). Last, Odometry [23]- [24].

2.4.1 'Apriori' Mapping

The 'A Priori' method relies on an existing and given map, one of which the robot does not need to explore in order to generate. This existing map provides ready information which the agent can use to localize its position. A map consisting of such information is called a feature map, can consist of landmarks, typically visual markers. These are just a few ways in which to represent the map, and often a higher-level representation is needed, such as discretizing the map into a grid [25]. In scenarious where the accuracy of the existing map is essential, say in applications such as robotic maneuvering in delicate spaces where obstacles must be avoided, this method prevails over the others. The type of positional localization used by this method is typically using statistical or probabilistic representations of the state; this is known as Markov Localization [26]. Being based on such a probabilistic framework, the 'A Priori' method has some drawbacks, one of which being that there is can be a sacrifice in accuracy of the robot localization or a sacrifice in bearing more computational efficiency on the system.

2.4.2 Simultaneous Localization and Mapping (SLAM)

SLAM does not rely on an existing map, and instead creates by itself a map of the environment, while simultaneously optimizing parameters to self-localize within it. SLAM systems can use a range of different extroceptive sensors in creating a map, include laser scanners [27]- [28], depth cameras [29], [30], [31], and regular mono- or stereo-cameras [31]- [32]. As there is a wide range of sensors that can be used, SLAM is essential, especially in unexplored, dynamically changing scenes in the use-case of autonomous vehicles, for ground-robots, UAVs and marine robotics. Most interesting of the sensors with SLAM, are the uses of visual systems. This subpart of SLAM is known as Visual SLAM, comprising of three major types: firstly, Only Visual SLAM; second, Visual-Inertial; lastly, Red-Green-Blue-Depth (RGB-D) [33]. As this thesis intends to outline the advantages of FPSP cameras with conventional cameras, more attention will be on Visual SLAM, focusing on Only Visual SLAM and RGB-D SLAM; Visual-Inertial next section.

Only Visual SLAM is the form of SLAM that depends only on vision from camera sensors alone for map estimation and robot localization. That said, it can consist of monocular, Red-Green-Blue-Depth SLAM (RGB-D SLAM) and stereo cameras in its sensor setup. It started with Parallel Tracking and Mapping (PTAM) is monocular- SLAM, working in small workspaces [34] localizing efficiently and effectively, especially in harsh lighting conditions.

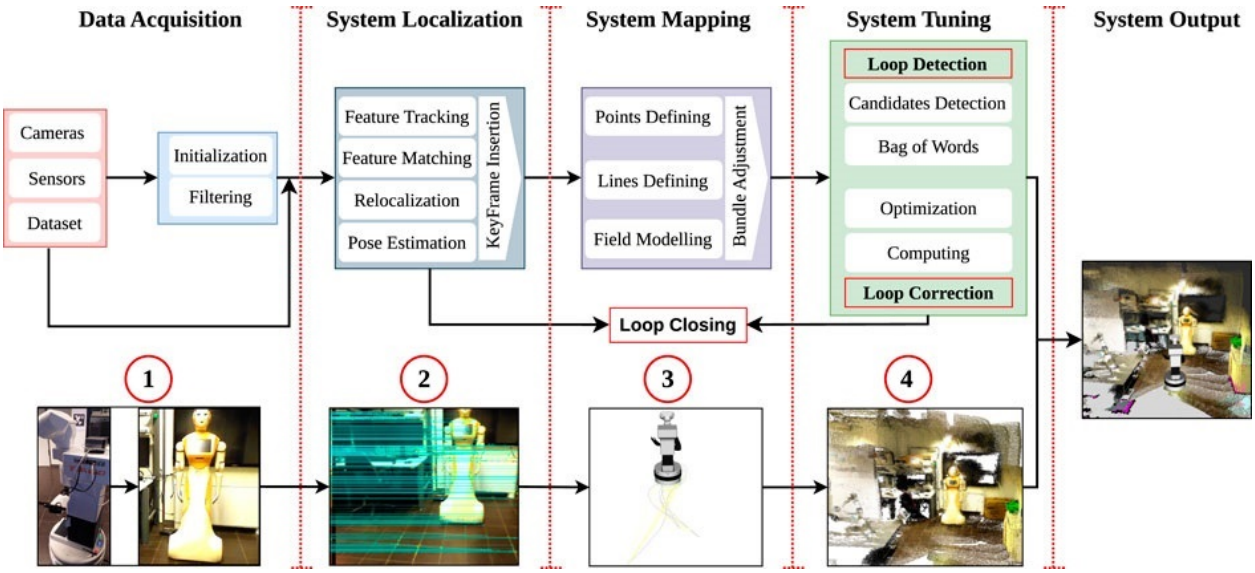


Figure 2.6: Visual SLAM architecture, comprised of 4 main components: first, data acquisition from extroceptive sensors of environment; second, system localization which involves processing data; next, system mapping, initial optimization guess in mapping; lastly, fine-tuning map by loop detection or correction. Figure from [33].

ORB-SLAM, built on the oriented Features from Accelerated Segment Test (FAST) and rotated Binary Robust Independent Elementary Features (BRIEF) named ORB feature-based descriptor, allowed real-time and more detailed map generation than its counterparts [35]. ORB-SLAM [20], [36] and ORB-SLAM2 [37], [31] are Only Visual SLAM. There is then Large-Scale Direct Monocular SLAM (LSD-SLAM) [32], [38] which was a more advanced framework for tracking and mapping, which generated an even more high-level accurate detailed map than the existing ORB-SLAM methods. Dense Visual Odometry and SLAM (DVO-SLAM) leveraged robot localization and mapping using depth-sensing from a monocular or stereo camera setup, using first principles from point-to-plane metrics in Photo Metric Bundle Adjustment (PBA) [39].

Next, RGB-D SLAM is the form of SLAM that depends on both vision using red, green, blue color ranges from camera sensors as well as depth data from a depth sensor. The RGB-D SLAM method has the advantage of generating maps with far higher quality detail than some of the best

vision-alone SLAM algorithms, able to track robustly in highly dynamic environments, not just static ones. Some RGB-D SLAM include Real-Time Appearance-Based Mapping (RTAB-Map) [37], Spatial Coordinate Error SLAM (SCE-SLAM) [40] as well as use-case with ORB-SLAM2 [31] and ORB-SLAM3 [41], which build off of existing ORB-SLAM to leverage with other sensors such as IMU, especially depth.

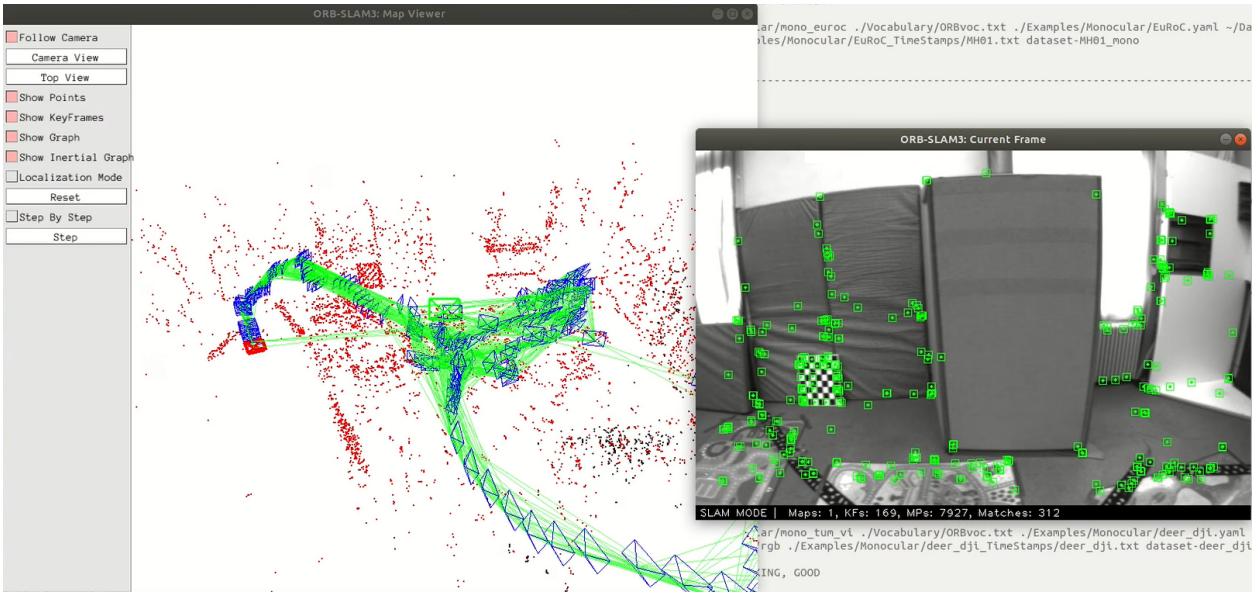


Figure 2.7: ORB-SLAM3 on the EuRoC Vicon Room 101 Dataset, doing both mapping and robot state localization in map itself.

Though SLAM methods provide a very detailed understanding of an environment's map, as well as allowing for robot localization to be done robustly within it, SLAM algorithms can be very computationally demanding.

2.4.3 Odometry

Odometry does not deal with generating a map at all, relying solely on estimating incremental changes in its position from some reference initial position [42], [43]. Constraining the robot in 2D, a wheel encoder can be used which gives a motion model of the robot state, which can then be integrated to give an estimate on the position. In 3D, an IMU can be used to track the full 6- DoF in time, but with no global reference information that is allocated, saved and stored, loop closure in Odometry systems is not as easily attainable when compared to the 'Apriori' mapping and SLAM methods. Odometry is not concerned with mapping unlike 'Apriori' mapping and SLAM, with this drawback in not readily providing loop closure, the benefit is that Odometry systems are more

computational inexpensive, more efficient, less algorithm intensive, especially important for mobile robots who need accuracy whilst maintaining far less computation cost in computing for it.

Various Odometry strategies exist, including IMU-based integration [44], and notably, VIO, which will be discussed further in the next section. There are several methods that use Odometry, either directly, or in the prediction step of their algorithms. One very important robot state localization framework based in Odometry is estimation by the Kalman Filter (KF). As well, odometry is used in the prediction motion model when doing estimation by the Particle Filter (PF).

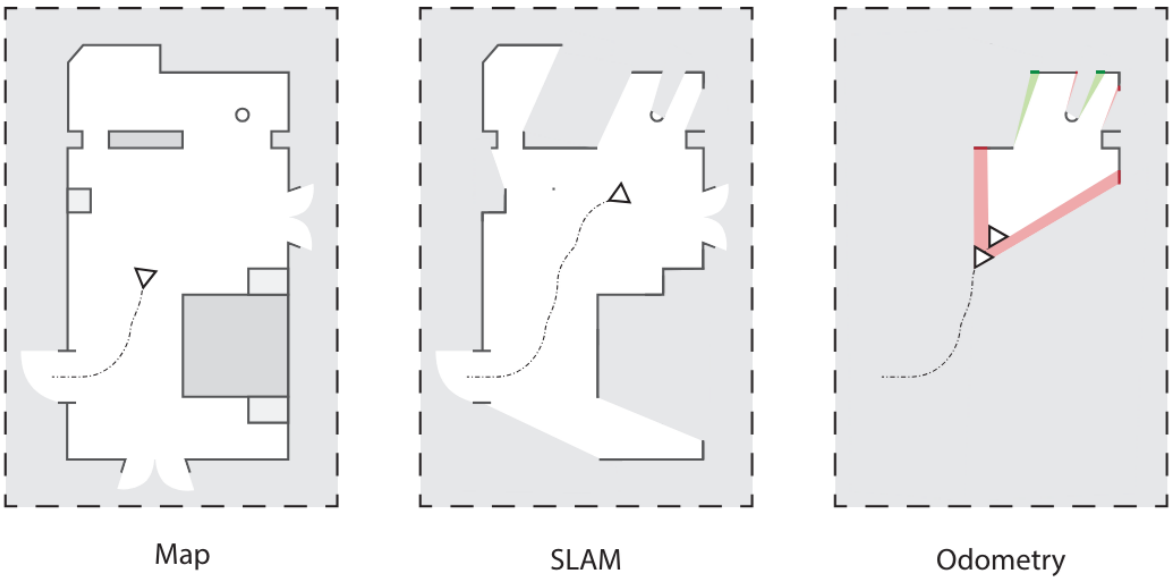


Figure 2.8: 'Apriori' Mapping vs SLAM vs Odometry. 'Apriori' is based on an existing map, SLAM is based on generating a map and self-localizing within it, Odometry is not focused on mapping but on incremental change in positioning. Figure from Raphael Maenle Master's thesis [45].

2.4.4 Kalman Filtering

The KF robot localization estimation approach, which is a Linear-Gaussian Estimation method, efficiently combines sensor measurements and system dynamics to estimate the state. The state, is the pose, which is the position and orientation of the robot. In very simple circumstances, the KF provides accurate and robust pose estimates even in the presence of noise and uncertainties [42].

The key idea is to represent the pose using a mathematical model that captures the inherent uncertainty in the system. Most estimation methods do this. By incorporating this uncertainty into the KF framework, the pose estimates are refined over time, taking into account both the measurements from sensors and the robot's motion dynamics. The KF is based on two parts: the

prediction step, and the update step. This is the same for the Extended Kalman Filter (EKF) and PF methods. The prediction step is based on the kinematic model of the robot. The update is based in part by some measurement, such as a range-bearing measurement. The prediction propagates the robot's state and the update step improves the accuracy and stability of this robot's state [42], [22].

Algorithm 1 Kalman Filter algorithm from [42] (more detail)

1: **Input:** μ_{t-1}, Σ_{t-1}, u_t, z_t ▷ Previous state mean and covariance, control input, observation
2: $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$ ▷ Predict mean
3: $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$ ▷ Predict covariance
4: $K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$ ▷ Compute Kalman gain
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$ ▷ Update mean with observation
6: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$ ▷ Update covariance
7: **return** μ_t, Σ_t ▷ Return updated mean and covariance

EKF, which is a Nonlinear Non-Gaussian Estimation method and an extension of the traditional KF, generally estimates the robot's pose more accurately. The EKF is particularly useful in scenarios where the robot's dynamics or measurement models are nonlinear, as it linearizes these models to enable efficient estimation. Nonlinear motion and measurement models more accurately capture the complexities of robotic systems.

Algorithm 2 Extended Kalman Filter algorithm from [42] (refer to for more detail)

1: **Input:** μ_{t-1}, Σ_{t-1}, u_t, z_t ▷ Previous state mean and covariance, control input, observation
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$ ▷ Predict mean using non-linear function
3: $\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + R_t$ ▷ Predict covariance using Jacobian of g
4: $K_t = \bar{\Sigma}_t H_t^T (H_t\bar{\Sigma}_t H_t^T + Q_t)^{-1}$ ▷ Compute Kalman gain using Jacobian of h
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$ ▷ Update mean with observation using non-linear function
6: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$ ▷ Update covariance
7: **return** μ_t, Σ_t ▷ Return updated mean and covariance

The framework has three main differences, all based on approximating the next state function as nonlinear, linear by first order Taylor expansion to go from non-Gaussian to approximate Gaussian: firstly, the state prediction is now based on the robot's Jacobian, linearizing the nonlinear function expressing the robot's motion model dynamics. Next, the measurement prediction is nonlinear, linearized, and approximated Gaussian through its Jacobian. This is based on the robot's measurement model. Lastly, the EKF's propagation of uncertainties is influenced by the accuracy of the linearization process. The frequency of the prediction and update and all other parts remain the same as the KF [42].

There are other variants of the KF, such as the Unscented Kalman Filter (UKF) [42]. There is also the iEKF which are equivalent to the Gauss-Newton algorithm [46]. Similar to the iEKF, the UKF is a variant of the KF for nonlinear systems. It avoids the linearization step used in the standard KF by employing the Unscented Transform, a deterministic sampling technique. The UKF generates representative points called sigma points, which are then propagated through the nonlinear system models to capture the system's behavior more accurately. By avoiding explicit linearization, the UKF provides improved estimation performance for nonlinear systems compared to the standard KF. The KF state estimation approach is imperative in VIO sensor fusion [47], [48], [49].

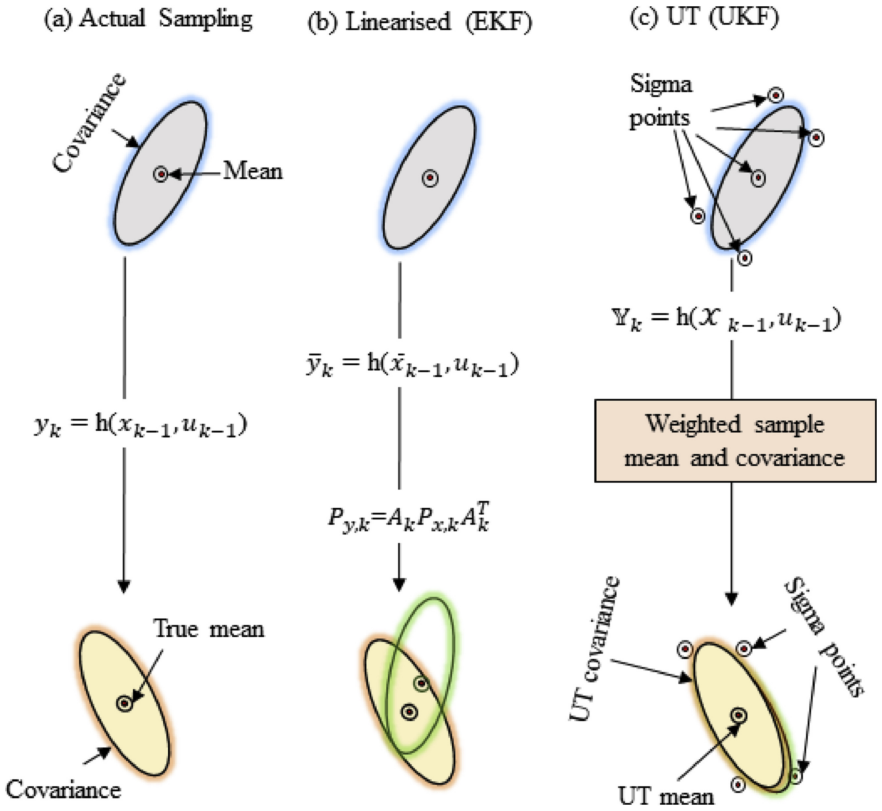


Figure 2.9: Process of (a) Actual Sample. Linearization Process in (b) EKF and (c) UKF. Figure from [50].

2.4.5 Particle Filtering

The PF, which is a Nonlinear Non-Gaussian Estimation method, represents the state estimate using a set of particles, each carrying a hypothesis of the robot's state. The PF framework under nonlinear and Non-Gaussian assumptions, gives benefit to the pose estimates by allowing for the flexibility to handle complex and nonlinear dynamics.

Like the KF and EKF approach, the PF operates through a two-step process: prediction and update. In the prediction step, the particles are propagated using the motion model, accounting for the robot's expected motion. In the update step, the particles are re-weighted based on the likelihood of measurements obtained from sensors, allowing for refinement and adjustment of the particle set. See, just the details of the prediction and update are different [42], [22].

Algorithm 3 Particle Filter algorithm from [42] (refer to for more detail)

Input: X_{t-1}, u_t, z_t ▷ Previous set of particles, control input, observation
$\bar{X}_t = X_t = \emptyset$ ▷ Initialize temporary and new set of particles
for $m = 1$ to M **do** ▷ For each particle
    sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ▷ Sample new state based on control input and previous state
    $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ▷ Compute weight based on observation likelihood
    $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$ ▷ Add weighted particle to temporary set
end for
for $m = 1$ to M **do** ▷ Resample particles
    draw i with probability $\propto w_t^{[i]}$ ▷ Select particle index based on weights
    add $x_t^{[i]}$ to X_t ▷ Add selected particle to new set
end for
return X_t ▷ Return the new set of particles

A major disadvantage of the PF is particle depletion. Sample depletion happens when the particles no longer effectively represent the true distribution, while particle degeneracy occurs when most particles have negligible weights, making them ineffective for accurate estimation.

As an overview of the three major types of methods for robot state localization estimation, it is worth mentioning that Odometry has many benefits when wanting to directly focus on the estimation of the robot state, as opposed to the details of the map it resides in, especially when computation costs need to be reduced on a robotic system. As mentioned, this thesis will now go over estimation by Visual-Inertial sensor fusion by Odometry means. In doing so, this will lead to utilizing the advantages of the high framerate and low power usage of FPSP cameras with the advantages provided by Odometry for vision- IMU-fused estimation.

2.5 Visual Inertial Odometry

VO is the process of a robotic system's ability to understand its state (position and velocity) with use of cameras. VIO is the process of estimating camera pose by combining visual information from

a camera and inertial measurements from IMUs. VIO provides more accurate and robust pose estimates. The sensors complement each other and are used in many applications and products such as AR headsets. VIO can be categorised into loosely-coupled and tightly-coupled methods. In a loosely-coupled method, the visual and inertial measurements are independently processed to estimate the motion and then are fused together for correction. On the other hand, the tightly-coupled method directly estimates the motion from the visual and inertial measurements [51].

Filtering VIO	**Fixed-Lag-Smoothing VIO**	**Full-Smoothing VIO**
Only updates the most recent states (e.g., EKF)	Optimizes window of states: • Marginalization • Nonlinear Least squares optimization	Optimizes all states: • Nonlinear Least squares optimization
X Linearization	✓Re-Linearize	✓Re-Linearize
X Accumulation of linearization errors	X Accumulation of linearization errors	✓Sparse Matrices
X Gaussian approximation of marginalized states	X Gaussian approximation of marginalized states	✓Highest Accuracy
✓Fastest	✓Fast	✗Slow (faster with Georgia Tech Smoothing and Mapping (GTSAM))

Table 2.1: Advantages and Disadvantages of Different VIO. Table from Davide Scaramuzza Lecture 13 Visual Inertial Fusion [52]. Note: the ✓highlights an advantage to this particular VIO type, while ✗highlights a disadvantage over the other types.

There are many different ways one can implement a VIO system. For example, one can use filtering [53]- [54] approaches, or using non-linear optimization, perform fixed-lag smoothing [55]- [56] or even full-smoothing [57]. For the full-smoothing, solving the entire system at every observation quickly becomes infeasible, hence they rely on Incremental Smoothing and Mapping Version 2.0 (iSAM2) [58] for incremental factor graph optimization.

Now, it is time to go over the current various VIO paradigms, implementations and methods.

2.5.1 Loosely-Coupled and Tightly-Coupled Visual Inertial Odometry

VIO has had two major paradigms since it started: loosely-coupled VIO and tightly-coupled VIO approaches [59]. The loosely-coupled approach is based on separate data acquisition, where the conventional camera takes in images to do feature tracking while the IMU sensor takes in IMU

sensor measurement data; in essence, this approach is based on two independent motion estimates which are fused together without feedback or modification to output the state of the robotic system [51]. Now, the tightly-coupled approach is based on outputting the state directly from the raw conventional camera data and IMU sensor measurements. The main difference is that in the tightly-coupled approach the output is only one motion estimate. The tightly-coupled approach is more robust and hence outputs a far more accurate output in comparison to the loosely-coupled approach.

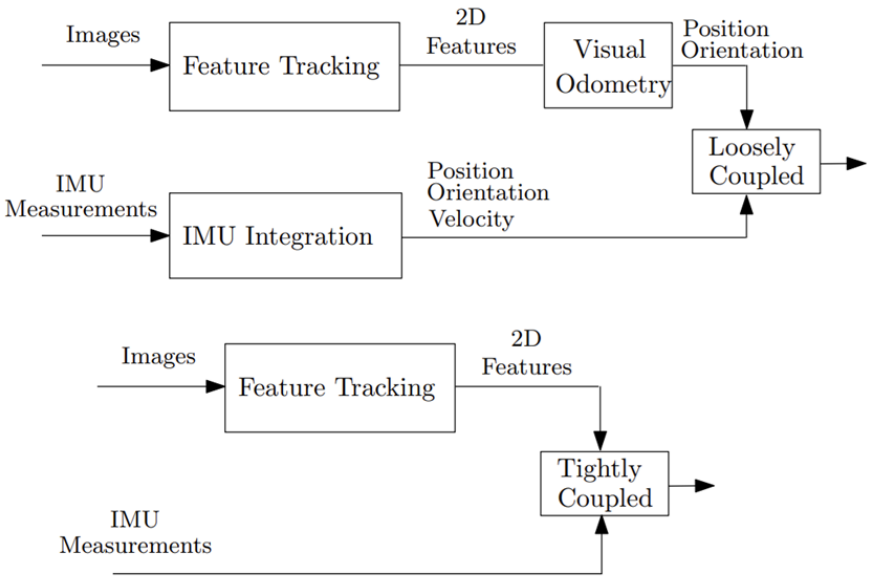


Figure 2.10: Loosely-coupled vs tightly-coupled VIO. Loosely-coupled is on seperate fusing of two poses, one from vision and one from IMU-integration. Tightly-coupled is on direct integration with image data and IMU measurement simultaneously. Figure from [51].

Further, there have been so many different VIO approaches under these coupling paradigms. They can be summarized in three different major categories: filtering based VIO, fixed-lag smoothing VIO, and full-smoothing VIO [51].

2.5.2 Filtering-Based Visual Inertial Odometry

To start with filtering based VIO, these VIO algorithms are entirely based on efficient state estimation by updating only the last state. By filters, this means to say various types of KF on the state estimation, such as the EKF [51] and UKF [42]. Classical approaches estimate both the pose and landmarks and the complexity grows quadratically based on the number of these estimated landmarks. Work from Davison et al. [60], Bloesch et al. [54] optimized and a small number

of landmarks were tracked to allow real-time operation instead. There is then a structureless approach, known as Multi-State Constraint Kalman Filter (MSCKF) [53], where landmark positions are marginalized out of the state-vector [53]. The disadvantage to this is that processing needed to be delayed until all measurements of a single landmark are obtained [53].

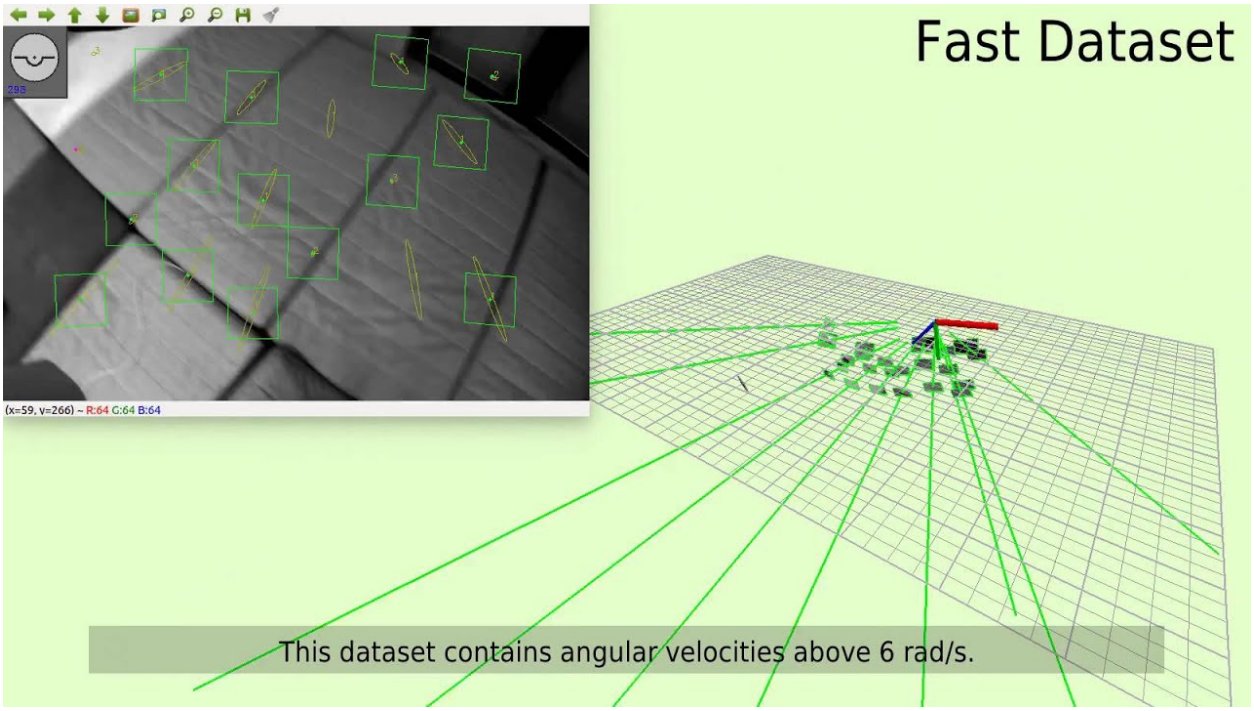


Figure 2.11: Robust Visual Inertial Odometry (ROVIO): Filtering VIO able to track robustly in fast, hostile motions. Here, the 'Fast' dataset deals with angular velocities above 6 rad/s. Figure from ROVIO Video Demo [61]- [54].

There is then filtering-based ROVIO [61]- [54]. Often it is mistaken that ROVIO falls under a smoothing VIO method. ROVIO uses a filtering approach. ROVIO minimizes photometric multi-level patches of monocular image frames to estimate the camera pose and velocity. Newly detected visual features are incorporated into the state vector. If a feature that is expected to be visible in the current frame already exists in the state, an intensity error is calculated by comparing the expected position of the feature in the frame with its actual observed position. This intensity error serves as an innovation term, which helps correct any accumulated errors from the propagation process. The state correction step is performed whenever a new image is captured by the camera. Another popular filtering-based VIO is the Multi-Sensor Fusion Extended Kalman Filter (MSF-EKF) [62], which of whom will be discussed in more detail at the end of this section. Discussing some additional frameworks that are leveraged atop MSF-EKF, adding atop this framework, would

be a VIO method known as Fast Semi-Direct Monocular Visual Odometry (SVO) [63], [62]. It combines direct image alignment with a filtering algorithm, specifically a KF, to estimate the camera's motion and position. It minimizes the photometric error of patches around features found. The direct image alignment is then optimized by minimizing reprojection error of the features via Least Squares Regression (LS). The MSF-EKF loosely fuses from SVO to give a VIO pose estimate. This is known as Fast Semi-Direct Monocular Visual Odometry + Multi-Sensor Fusion Extended Kalman Filter (SVO+MSF) [63], [62].

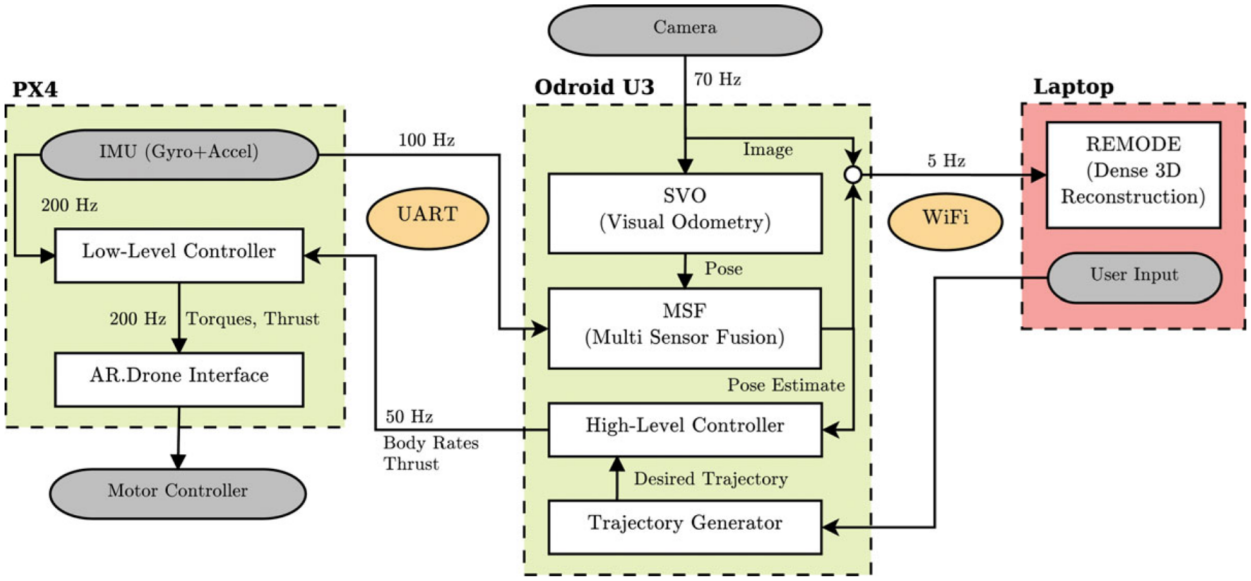


Figure 2.12: SVO+MSF Pipeline Utilized on Quadrotor. Note: Loosely-coupled, filtering VIO. Figure from [64].

Despite how filtering based is advantageous in being more efficient than fixed-lag and full-smoothing, filtering generally linearized propagation and measurement, which accumulates linearization errors, giving too much bias towards Gaussian approximations of marginalized states [51]. Generally the VIO problem has four unobservable directions: the global position and orientation around the gravity direction [65]. In Kottas et al. [66], linearization at the wrong estimate adds incorrect information in those unobservable directions. Huang et al. proposed a solution which better accounted for real-world nonlinear systems, being more consistent and better estimation than the linear approximated case [67]. PF is an instance of alleviating the Gaussian noise assumption, modelling non-linearity well. Non-filtering VIO too does not inherently assume Gaussian noise, but this all depends on the algorithm's associated loss function. The Huber loss function can handle non-Gaussian noise. Such non-filtering include iSAM2 [58], GTSAM [68].

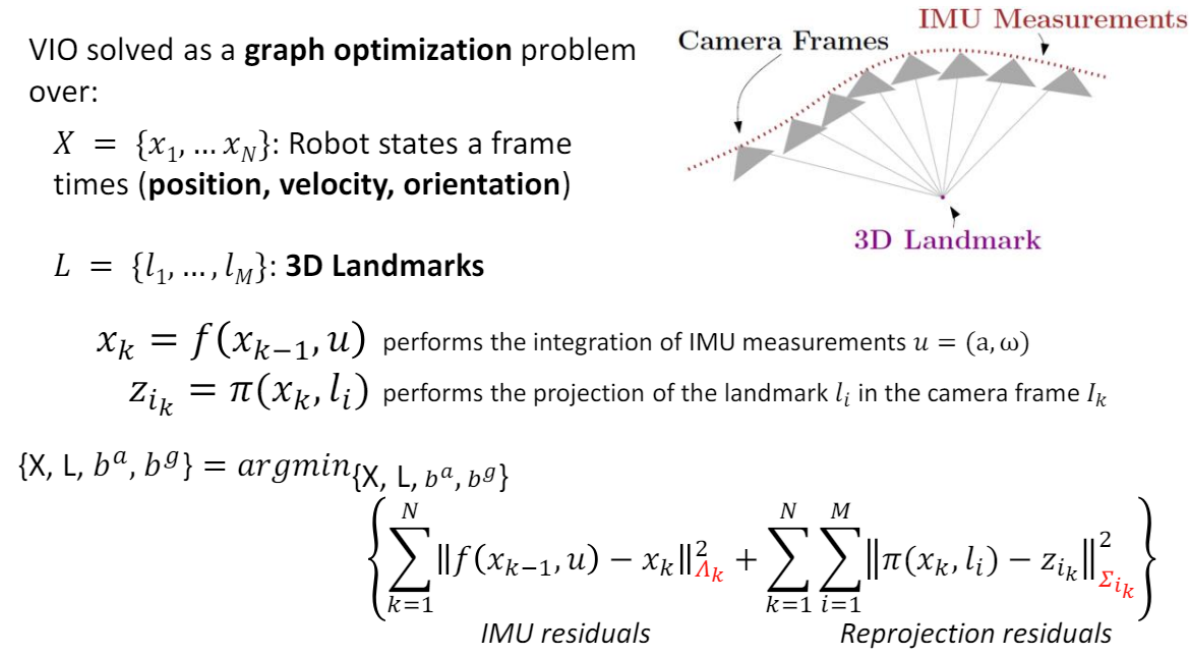


Figure 2.13: Fixed-Lag-Smoothing-Based VIO, showcasing how smoothing methods involve assuming VIO as a graph optimization problem. Figure from Davide Scaramuzza Lecture 13 Visual Inertial Fusion [52].

2.5.3 Fixed-Lag-Smoothing-Based Visual Inertial Odometry

There is then fixed-lag smoothing VIO, these VIO algorithms are based on optimization via a window of states and often require a nonlinear optimization approach to output the state [51].

Their advantage over filtering based VIO is they allow re-linearization and are still fast [51]. They also are generally more accurate than the filtering VIO method. Use of robust cost functions or by more explicit outlier rejection after optimization make these approaches more robust to outliers. With fixed-lag-smoothing-based VIO still resorting to marginalization of the state, like with filter-based VIO, they still suffer from some inconsistency and linearization errors [69], [70], [71]. Marginalization of states outside the estimation window also can lead to dense Gaussian priors which is bad on the computational front when matrix operations are done. To alleviate this, some and certain measurements are dropped, as was proposed by Leutenegger et al. [55].

A popular fixed-lag-smoothing-based VIO is known as Open Keyframe-based Visual-Inertial SLAM (OKVIS) [72], [55]. It combines visual and inertial measurements for SLAM applications. It employs a sliding window optimization framework and utilizes bundle adjustment to jointly

optimize the camera poses, landmarks, and sensor biases. has gained popularity in the robotics field for its robust performance in challenging scenarios, making it a notable method against other VIO methods.

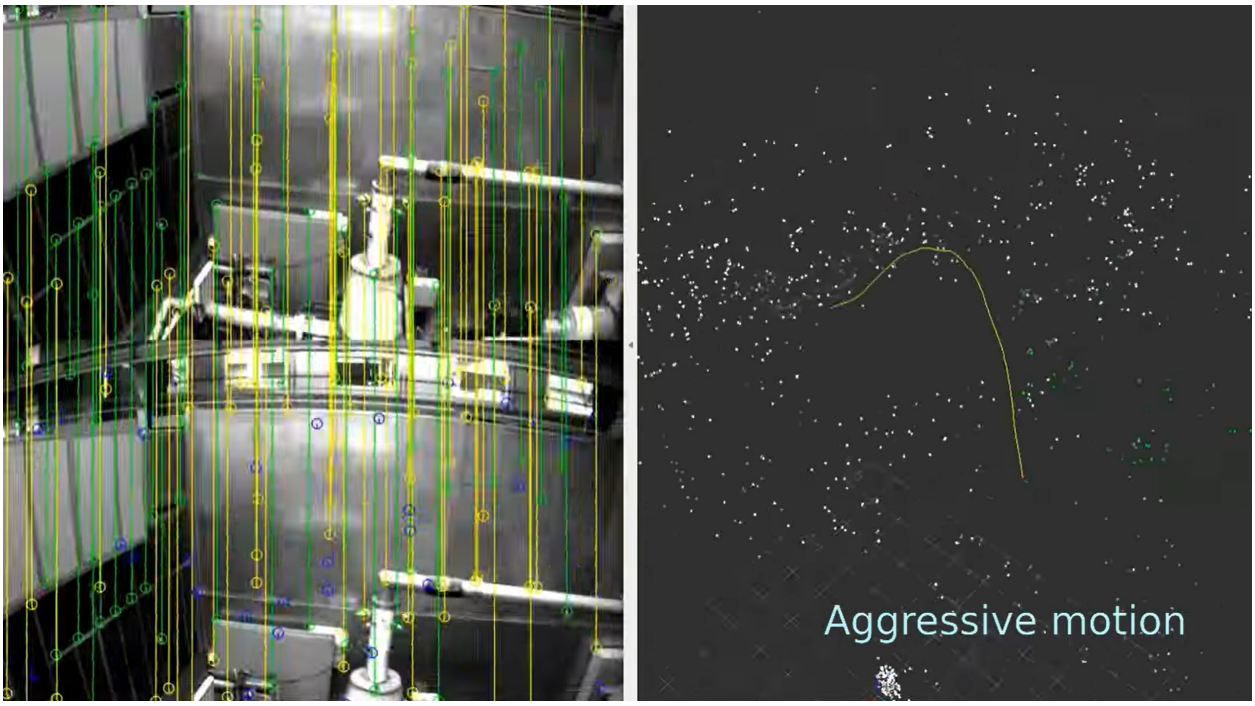


Figure 2.14: OKVIS VIO in aggressive motions. Figure from Stefan Leutenegger Video Demo [72], [55].

There is also, VINS-Mono [56] is a sliding window estimator that employs non-linear optimization techniques and robust corner feature tracking. Its design is similar to fixed-lag-smoothing-based but differs in being fixed-lag in how it introduces a loosely-coupled sensor fusion initialization procedure that enables the estimator to start from arbitrary initial states. Additionally, IMU measurements are pre-integrated before being incorporated into the optimization process, and a tightly-coupled relocalization procedure is proposed [56].

Next, there are semi-direct methods such as SVO integrates direct visual odometry with IMU data using a keyframe-based method to benefit from both direct and feature-based approach [63]. The improved version SVO 2.0 [73] incorporates more robustness and accuracy with IMU integration. Kimera VIO uses the GTSAM [68] for IMU pre-integration and fixed-lag smoothing, although this can be switched to full-lag smoothing for comprehensive trajectory optimization [74] [75]. This approach estimates the 3D position of observed features and eliminates these 3D points from the VIO state, which the states that fall out are marginalized out using GTSAM [68].

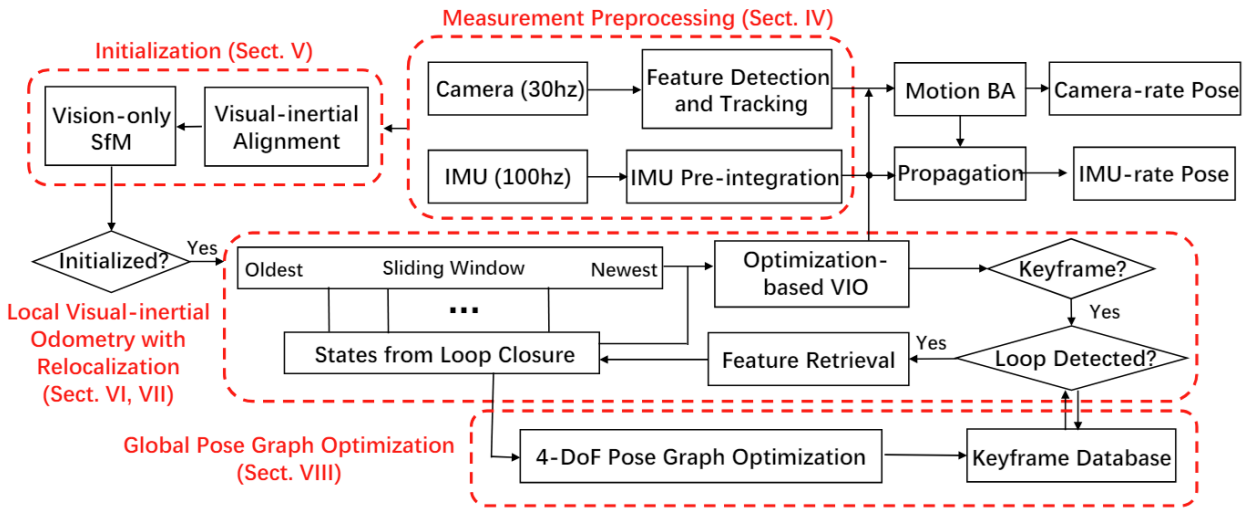


Figure 2.15: Monocular Visual-Inertial State Estimator (VINS-Mono) pipeline for monocular-inertial sensor fusion. Shows initialization (left), measurement preprocessing (middle top) and global pose graph optimization (middle bottom). Figure from VINS-Mono [56].

2.5.4 Full-Smoothing-Based Visual Inertial Odometry

The last VIO approach are full-smoothing algorithms which triumph over the last two [51]. These VIO algorithms optimize all states of the full trajectory of the robotic system and do so, similar to the last approach, on nonlinear optimization [51]. They not only have the added benefit of re-linearization but deal with sparse matrices. They have the highest accuracy of the three VIO approaches, but the only major disadvantage with them is they are generally slow and inefficient and take more computational power and resources. The complexity of the optimization problem is approximately cubic with respect to the dimension of the states.

This makes its use in real-time robotic systems limited as the map grows over time. What can be done is to only keep selected keyframes, as was proposed by Leutenegger et al. [55], Qin et al. (VINS-Mono [56]), etc. and/or run the optimization in a PTAM architecture [76]. A major leap has been the development of incremental smoothing methods such as Incremental Smoothing and Mapping (iSAM) [77] and iSAM2 [58]. They are based on factor graphs to identify and update only typically small subset of variables affected by a new measurement. Forster et al. have used this incremental framework in VIO.

Another disadvantage with this VIO is with their use of IMUs. And this is a disadvantage too with fixed-lag-smoothing-based VIO. With filtering-based VIO, the IMU is used for the process/prediction model and cameras for the measurement model and thus IMUs, by standard, are

dealt with at different rates. Filtering-based VIO can handle different rates of IMUs and cameras easily. In smoothing VIO, however, it is not good in real-time robotic systems to add a state at every IMU measurement so it was proposed that IMU measurements be integrated between frames, through reparameterization, and form relative motion constraints. This is what is known as IMU preintegration [57]- [73] and is to be done after each optimization iteration (repeated when the state prediction changes), as proposed in Lupton and Sukkarieh [78]. Full-smoothing-based VIO is fast though with GTSAM [51]. SVO was discussed before with SVO+MSF. Here, it is worth noting it can also can be added onto a tightly-coupled VIO design as Fast Semi-Direct Monocular Visual Odometry + GTSAM (SVO+GTSAM) [57]. This has the same front-end as the SVO+MSF, but now full-smoothing is done, generally using iSAM2 [58]. Preintegrated IMU factors in the pose graph are utilized.

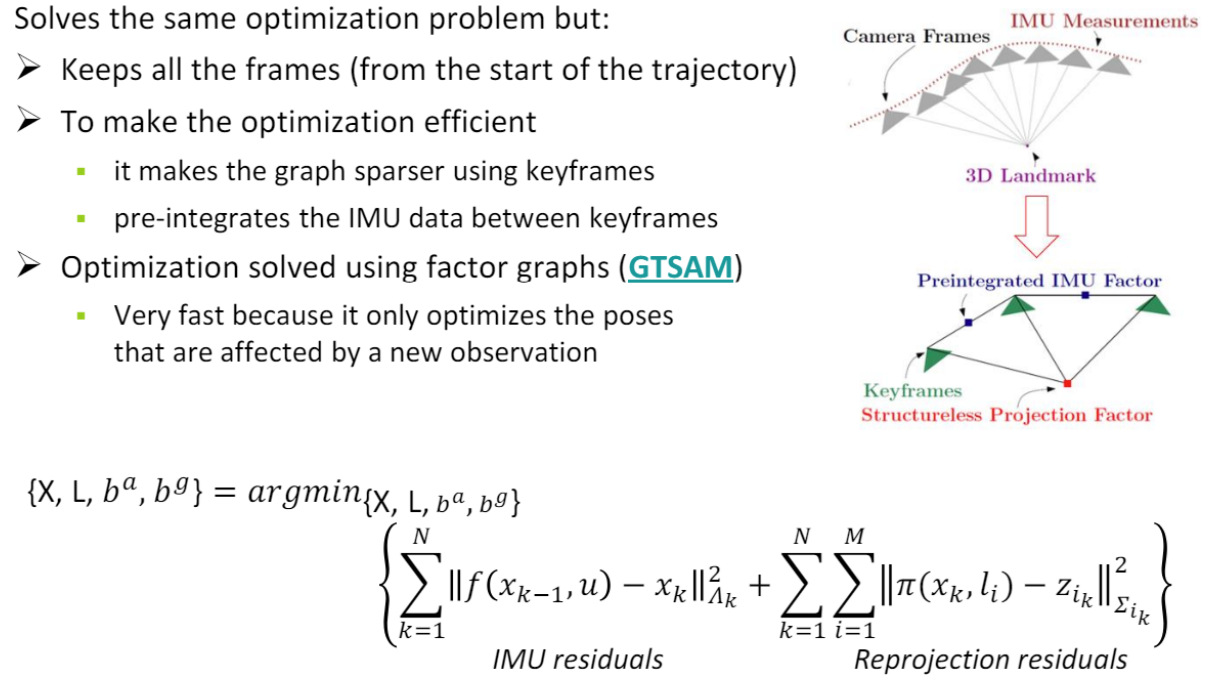


Figure 2.16: Full-Smoothing-Based VIO: SVO+GTSAM [57]. Here, solves the same optimization problem as fixed-lag smoothing but keeps all the frames, and typically uses factor graphs [51]. Figure from Davide Scaramuzza Lecture 13 Visual Inertial Fusion [52].

Further, Kimera VIO [75] also uses full-smoothing in cases where more accurate state estimation is required over a larger temporal window. This is particularly used for mesh generation where a larger VIO is used to perform full-smoothing to obtain accurate 3D mesh [75]. Famously, Campos et al. proposed ORB-SLAM3 [41] which is full-smoothing VIO. This approach optimizes entire set of key frames and points for consistency, effective in environments with many loop closures.

2.5.5 Visual Inertial Odometry on Unconventional Cameras

While a conventional camera is typically used in VO and visual SLAM, other camera technologies such as event-based cameras are used in many state-of-the-art VIO algorithms. Event cameras provide low power usage and low latency benefits over conventional cameras and are also robust against illumination changes [79] as they are based in tracking changes in pixel-wise intensity.

Zihao Zhu et al. [80] presented the first event-based VIO framework that utilized the high frame rate capability of tracking events at 1 MHz using a EKF. Next, [81] focused on the mere front-end and proposed a novel spatio-temporal event-feature tracker that was then aligned using current camera motion and scene structure, leveraged on a keyframe-based VIO pipeline for 850 deg/s tracking. The nature of the low latency of event streaming was then further used in a SLAM setting, titled Ultimate SLAM [82], where events are multi-sensor-fused along with images, IMU measurements, especially robust in high-speed scenarios. Inspired by fixed-lag smoothing and full-smoothing VIO, [83] then used event cameras for the first work in continuous-time representation for visual-inertial estimation with micro-second accuracy.

There are several downsides though with event-cameras which FPSPs do better:

- First, while event cameras compress visual information into a continuous stream of events, they don't offer the flexibility of the FPSP, which supports user-programmed algorithms like the FAST-corner or edge-coordinates detector [84]. This implies that event-based cameras only consist of one temporal feature output, capability, but what about more complex information extraction [19]? FPSPs provide this allowing for capabilities such as full on-sensor convolutions, filters, not just corner or edge feature outputs.

- Further, the data volume transferred by an event camera is proportional to camera motion. This characteristic is not optimal; for instance, a robot has a shorter window during rapid movements to determine its subsequent action. FPSPs consist of digital and analog memory directly on-sensor which allows for windowing and memory allocation of the image-plane.

In this regard, FPSPs provide a promising alternative to event-based cameras, making note that, really, a FPSP-based camera is also an event camera in how it can track and store feature-events at high framerate, but can do much more than them, given its on-sensor advantages and capabilities.

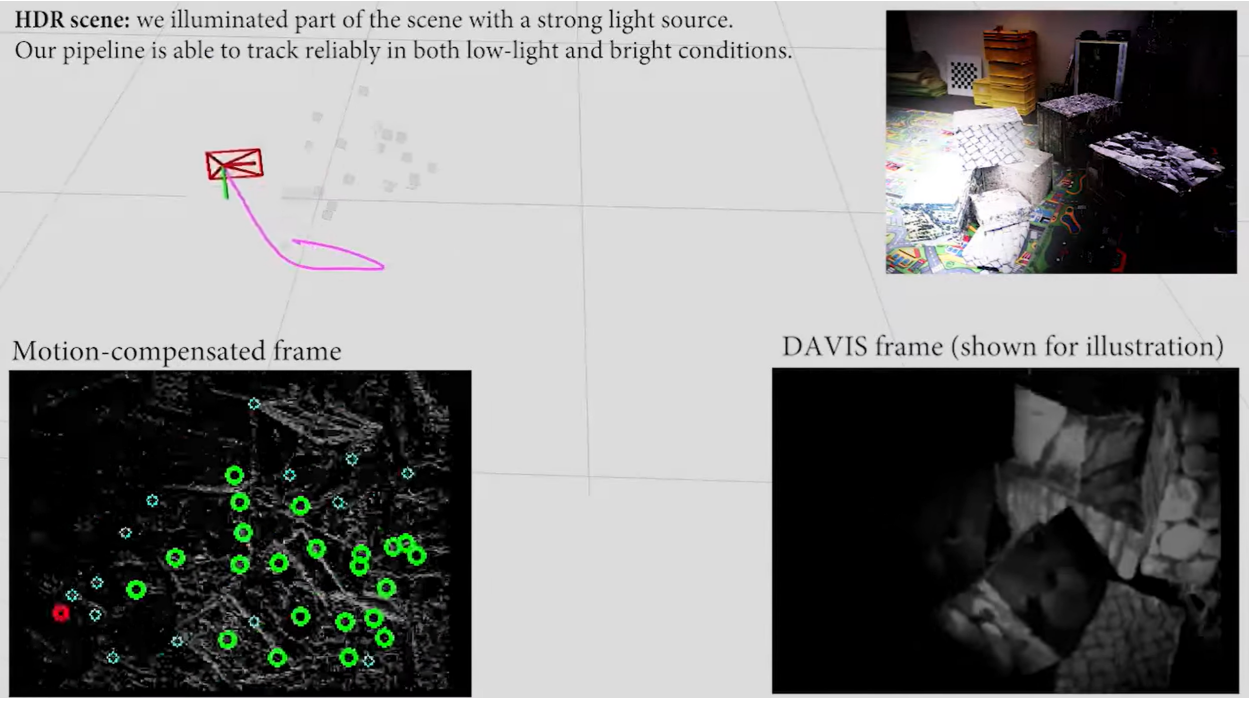
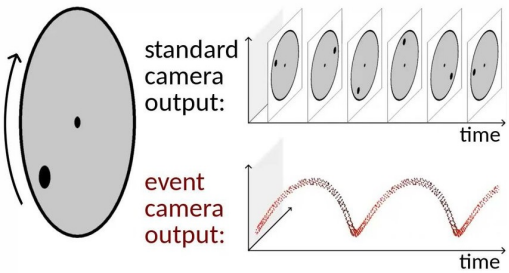


Figure 2.17: Event cameras and how they can track robustly, especially in unilluminated rooms. An event is merely a temporal feature on the illumination of light changes. Figure from Video Demo [82].



- Sparse, high-temporal resolution data extracted from images
- log-intensity changes (DVS, DAVIS, ATIS, Celex, etc.)
  - are only **one type** of temporal feature
  - and perhaps not always the most useful one…
  - NB. Biology does a lot more on the retina!
- How about **other spatial & temporal features?**
  e.g. events corresponding to moving edges,
  corners, general convolution filters, etc.
- How about **more complex information extraction**?
  e.g. object centroids, compressive sensing,
  optical flow, tracking, visual odometry, neural net, etc…
- How can we construct a vision sensor that can **do this all**?

Figure 2.18: The disadvantage of event cameras: only single temporal feature output. FPSPs can do any meaningful feature output. Figure from CVPR 2019 Workshop on Event-based Vision and Smart Cameras by Pitor Dudek [19].

2.5.6 Multi-Sensor Fusion Extended Kalman Filter Visual-Inertial Odometry

Out of all the different approaches, of great interest, is the MSF-EKF [62], which is a loosely-coupled VIO, entirely filtering-based approach [62]. As was mentioned, there are several other filtering based VIO algorithms such as MSCKF, SVO+MSF [63], [62] and ROVIO [61]- [54], but MSF-EKF is modular, allowing for easy integration, especially with high frame rate outputting unconventional vision sensors. This VIO essentially allows for easy implementation via its loose-framework where the front-end can be swapped, especially allowing for front-end benefit with unconventional cameras of varying outputs and frequencies. Though the high frame rate capabilities are the benefit of generally most filtering-based VIO, specifically the MSF-EKF framework provides one of the highest frequency estimation outputs of all other filtering frameworks, perfect for utilizing the capability of the FPSP. In having a VIO algorithm which deals with high frame rate, would allow for the spatial drift to be addressed, and help be alleviated alongside with the temporal drift provided by the IMU. In the next section, this thesis will now discuss the design of the BIT-VIO algorithm necessary to incorporate the SCAMP-5 FPSP for visual-inertial estimation, and present experimental results of the proposed method.

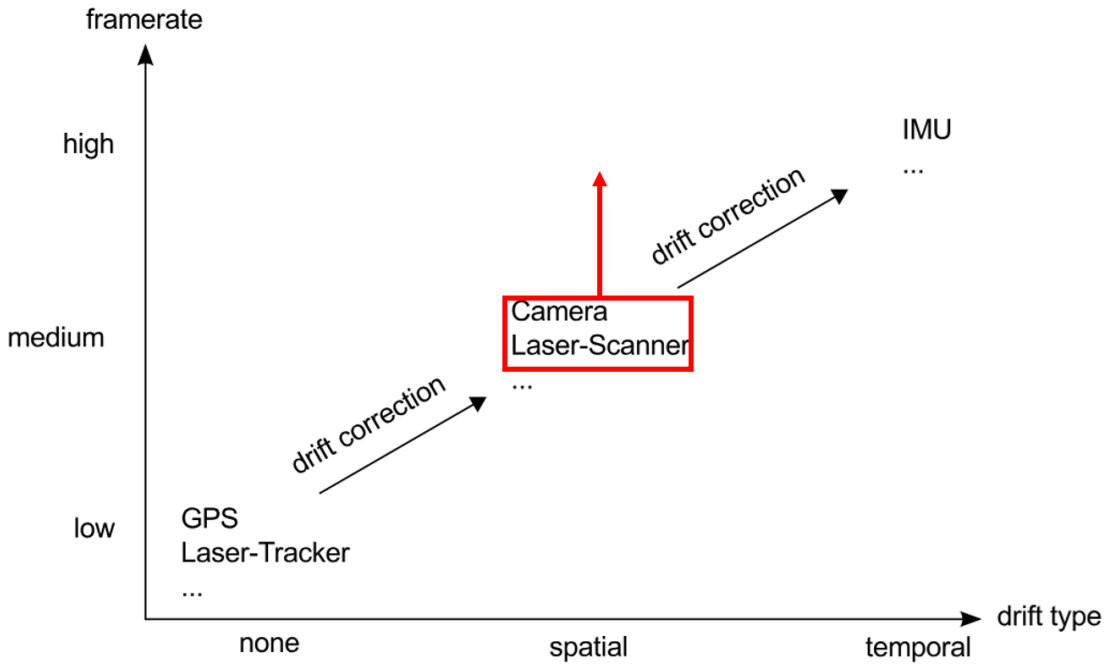


Figure 2.19: MSF-EKF for FPSP- IMU-fused state estimation as it is filtering VIO which provides the highest frequency estimation output. MSF-EKF over all other VIO as it allows for easy integration, especially with high frame rate outputting unconventional vision sensors. In having a VIO algorithm which deals with high frame rate, would allow for the spatial drift to be addressed, and help be alleviated alongside with the temporal drift provided by the IMU. Figure from Stephan Weiss PhD Thesis [85].

Chapter 3

Methodology and Implementation

In this chapter, the overarching algorithm established in this thesis, BIT-VIO, is presented, and is summarized briefly in Sec. 3.1. In Sec. 3.2, the notations used are introduced. In Sec. 3.3, a broader overview of the system design is presented and the coordinate frames used are stated, followed by the IMU model and state prediction in Sec 3.4. BIT-VO, namely the camera pose measurement in the BIT-VIO algorithm is discussed in more detail in Sec. 3.5, along with the uncertainty propagation done on it in Sec. 3.6. In Sec. 3.7, the correction step via the iEKF of MSF-EKF is presented. After the BIT-VIO algorithm formulation, performance evaluation metrics in Sec. 3.8 discussed, the first initial attempts at testing the framework with different IMU cases is explored, attempting with approximated calibration intrinsics and extrinsics and lower IMU frequency rates are in Sec. 3.9. Addressing the first issues in the first attempts, the experimental setup outlining the visual-inertial calibration process is defined in Sec. 3.10 with IMU-alone calibration, SCAMP-5 FPSP-alone calibration and the full visual-inertial calibration. Next there is Sec. 3.11 with the experimental results, going over accuracy and robustness of the BIT-VIO algorithm, evaluations by Root-Mean-Square Error (RMSE), and error mapping and noise variation benchmarking against prior BIT-VO and ground-truth. This chapter ends in Sec. 3.12 with an extensive study on the algorithmic execution timing/frame, accuracy, memory usage and power consumption of the visual front-end processing performance on the FPSP.

3.1 Visual Inertial Odometry Using Focal Plane Binary Features

BIT-VIO is presented, the first VIO which utilises SCAMP-5. BIT-VIO is a loosely-coupled iEKF which fuses together the VO running fast at 300 FPS with predictions from 400 Hz IMU

measurements to provide accurate and smooth trajectories. This thesis presents cases too when the IMU is running prediction at a lower 200 Hz with the UM7 IMU and when it runs at an even lower 100 Hz with the MPU-9250. In both cases, BIT-VO is running at 100 FPS. Though these IMUs do not fully utilize the FPSP VO at its full high framerate capability, it is necessary in the context of making sure any black box IMU works with the proposed BIT-VIO framework.

3.2 Notations

The following notation conventions are used in this work, adopted from [86], [87]:

- Units of a variable A as $[A]$ (e.g. $[a_x] = m/s^2$).
- Skew-symmetric matrix of A is $\lfloor A \rfloor$.
- p_A^B represents the translation from frames $A \to B$.
- q_A^B represents the Hamiltonian quaternion rotation $(q_{A\,w}^B, q_{A\,x}^B, q_{A\,y}^B, q_{A\,z}^B)$ from frames $A \to B$.
- $\hat{p}, \hat{q}$ are the expected translation and rotation.
- $\tilde{p}, \tilde{q}$ are the error in translation and rotation.
- $C_{(q)}$ is the rotational matrix to the quaternion q.
- $\Omega(\omega)$ is quaternion-multiplication matrix of ω.
- $\delta q = q \otimes \hat{q} \approx \left[\frac{1}{2}\delta\theta^T, 1\right]^T$ approx. for quaternion δq.
- $\vec{q} \otimes \vec{p} = (q_4 + q_1 i + q_2 j + q_3 k)(p_4 + p_1 i + p_2 j + p_3 k)$ where the quaternion multiplication is defined by operation $\otimes$.

3.3 System Overview

Fig. 3.1 demonstrates an overview of the system. The VO pipeline is shown on the top, and the inertial pipeline is shown on the bottom. The algorithmic components of the VO are mostly done on a remote host, except the corner/edge detection, which is done on the SCAMP-5 FPSP.

Fig. 3.2 shows the coordinate frames used, between World, IMU and SCAMP-5 FPSP.

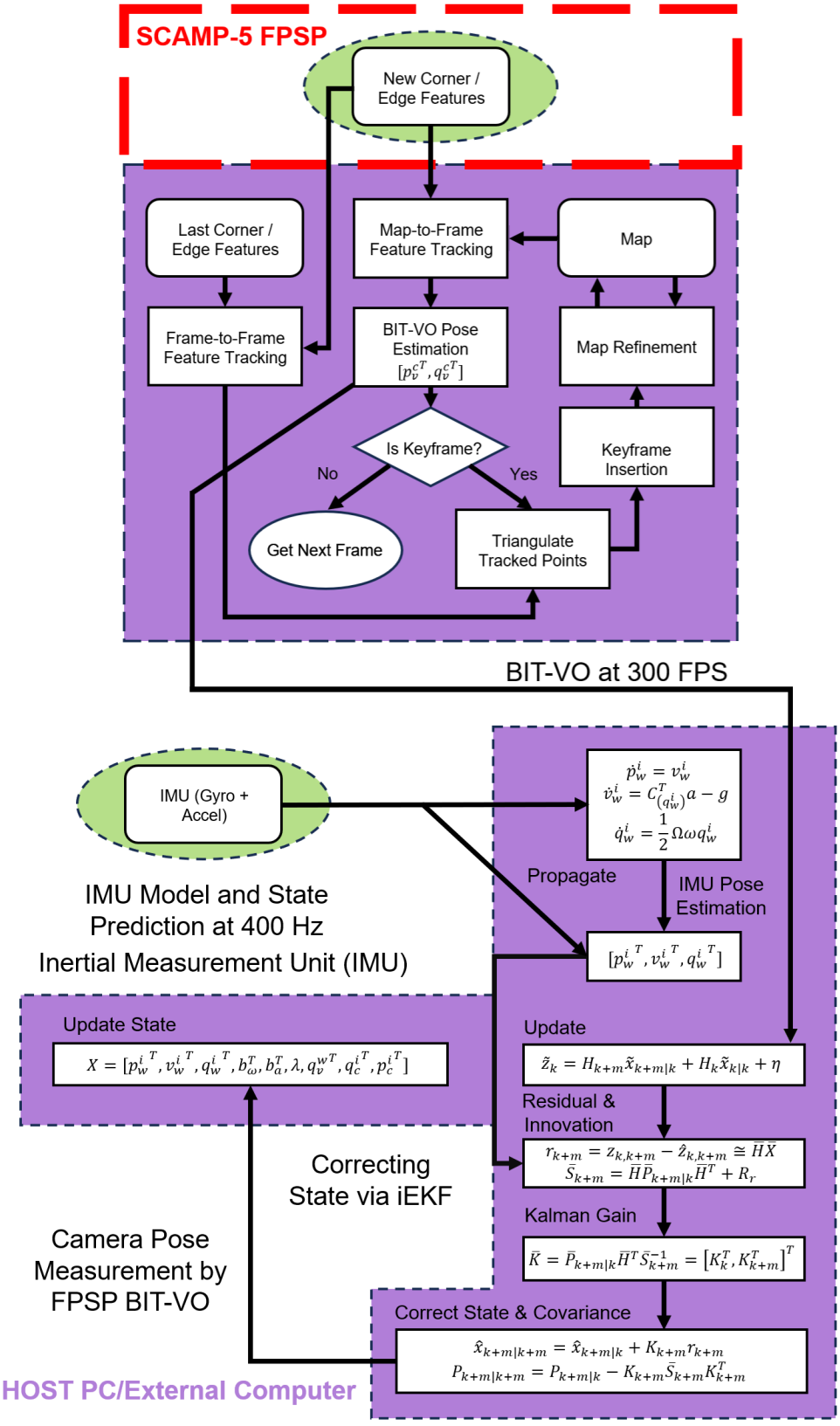


Figure 3.1: Pipeline of BIT-VIO. The multi-sensor fusion is to the bottom. BIT-VO is on the top. From the BIT-VO algorithm [17], the vision sensor utilizes the SCAMP-5 FPSP, highlighted in red. New corner/edge features are detected via the FPSP, off-putting computational load by allowing some image and signal processing to be done on the chip before transferring to a PC host or other external device to be further processed.

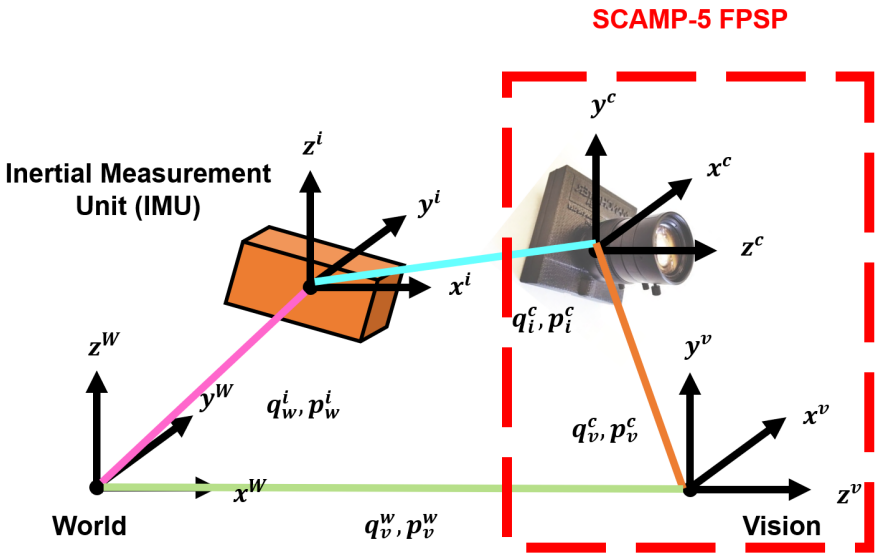


Figure 3.2: Coordinate frame definition of the IMU and the SCAMP-5 FPSP. In total, define four coordinate frames. Notation p_A^B and q_A^B are used to represent transformation from A to B. Highlighted in red is the SCAMP-5 FPSP camera and vision cooedinate frames.

3.4 IMU Model and State Prediction

Using MSF-EKF [62], assume absolute IMU measurements have bias b_ω, b_a with Gaussian noise n_ω, n_a. IMU measures and outputs angular velocities ω and linear accelerations a in the IMU-frame [86]:

$$\omega = \omega_{\text{meas}} - b_\omega - n_\omega \,, \quad \dot{b}_\omega = n_{b_\omega} \,, \tag{3.1}$$

$$a = a_{\text{meas}} - b_a - n_a \,, \quad \dot{b}_a = n_{b_a} \,. \tag{3.2}$$

In Eq. (3.1) and (3.2), the subscript "meas" means the measured value. Terms $\dot{b}_\omega, \dot{b}_a$ are the dynamic models of the IMU biases. The MSF-EKF states x are represented in two parts: the x_{IMU}^T and x_{BIT-VO}^T. The x_{IMU}^T, which is a 16-element state, is formed by the IMU measurements and dynamic models, as follows [86]:

$$x_{IMU}^T = [{p_w^i}^T, {v_w^i}^T, {q_w^i}^T, b_\omega^T, b_a^T] \,, \tag{3.3}$$

$$\dot{p}_w^i = v_w^i \,, \tag{3.4}$$

$$\dot{v}_w^i = C_{(q_w^i)}^T a - g \,, \tag{3.5}$$

$$\dot{q}_w^i = \frac{1}{2} \Omega \omega q_w^i \,. \tag{3.6}$$

In Eq. (3.3)-(3.6), ${p_w^i}^T, {v_w^i}^T, {q_w^i}^T$ represents the translation, velocity, and quaternion rotation of the IMU with respect to world (or inertial frame). The dynamic models $\dot{p}_w^i, \dot{v}_w^i, \dot{q}_w^i$ propagate the state and do so at the rate of the IMU.

3.5 Camera Pose Measurement by FPSP BIT-VO

In the BIT-VO [17] part of the algorithm, the front-end visual processing occurs on the SCAMP-5 FPSP. FAST corner and binary edge features are detected on the chip before it is transferred to the PC. The proposed BIT-VIO algorithm used a modified version of FAST for the front-end corner feature-part (from BIT-VO [17]), where the processing for each pixel is done in parallel, simultaneously and not row-by-row, column-by-column indexed [18], [88].

Algorithm 4 FAST-corner Feature Extraction (on FPSP) algorithm from [89] (refer to for more detail)

DREG registers used: R1 - R8 as ring counter, R9 for storing corner coordinates of image
$C \leftarrow \emptyset$
Parallel for each pixel $p = (x, y)$ in I where I is a 256×256 image **do**
  $I_p \leftarrow I(x, y)$ ▷ Intensity of the center pixel
  Define the 16-pixel ring $p_{i\,i=1}^{16}$ around p
  Initialize $N_b \leftarrow 0$, $N_d \leftarrow 0$, $N_s \leftarrow 0$
  for each pixel p_i in the 16-pixel ring **do**
    if $I(p_i) > I_p + t$ **then**
      $N_b \leftarrow N_b + 1$ ▷ Count bright pixels in the ring
    elseif $I(p_i) < I_p - t$ **then**
      $N_d \leftarrow N_d + 1$ ▷ Count dark pixels in the ring
    else
      $N_s \leftarrow N_s + 1$ ▷ Count non-similar neighbor pixels
    end if
  end for
  if $(N_b \geq 10)$ or $(N_d \geq 10)$ or $(N_s \geq 10)$ **then**
    $C \leftarrow C \cup p$
  end if
end parallel for
return C in DREG register R9
Reset DREG registers R1 - R8 for next image

Sobel Edge Detection [90] was used but on the FPSP, where analog registers are used, computing in parallel, like with the corner-features above. The first step is to do, simultaneously across all pixels on the entire image-plane, vertical edge detection, then horizontal. The corner and edge feature intensity thresholds used were $t_f = 35, t_e = 60$, respectively.

Algorithm 5 Sobel Edge Detection (on FPSP) algorithm from [90], [18] (refer to for more detail)

Analog registers used: A, B, C, D for computations, E for edge detected binary image
Parallel for each pixel $p = (x, y)$ in I where I is a 256×256 image **do**
  Vertical Edge Detection:
    $A \leftarrow$ Move C NORTH ▷ move all values in register C, 1-pixel NORTH (repeat logic below)
    $B \leftarrow$ Move C SOUTH
    $A \leftarrow A + B + 2C$
    $B \leftarrow$ Move A EAST
    $A \leftarrow$ Move A WEST
    $D \leftarrow \text{abs}(B - A)$ ▷ D Analog register stores the vertical edge
  Horizontal Edge Detection:
    $A \leftarrow$ Move C EAST
    $B \leftarrow$ Move C WEST
    $A \leftarrow A + B + 2C$
    $B \leftarrow$ Move A SOUTH
    $A \leftarrow$ Move A NORTH
    $A \leftarrow \text{abs}(B - A)$ ▷ A Analog register stores the horizontal edge
  $E \leftarrow A + D$ ▷ Combine vertical and horizontal edges into E
end parallel for

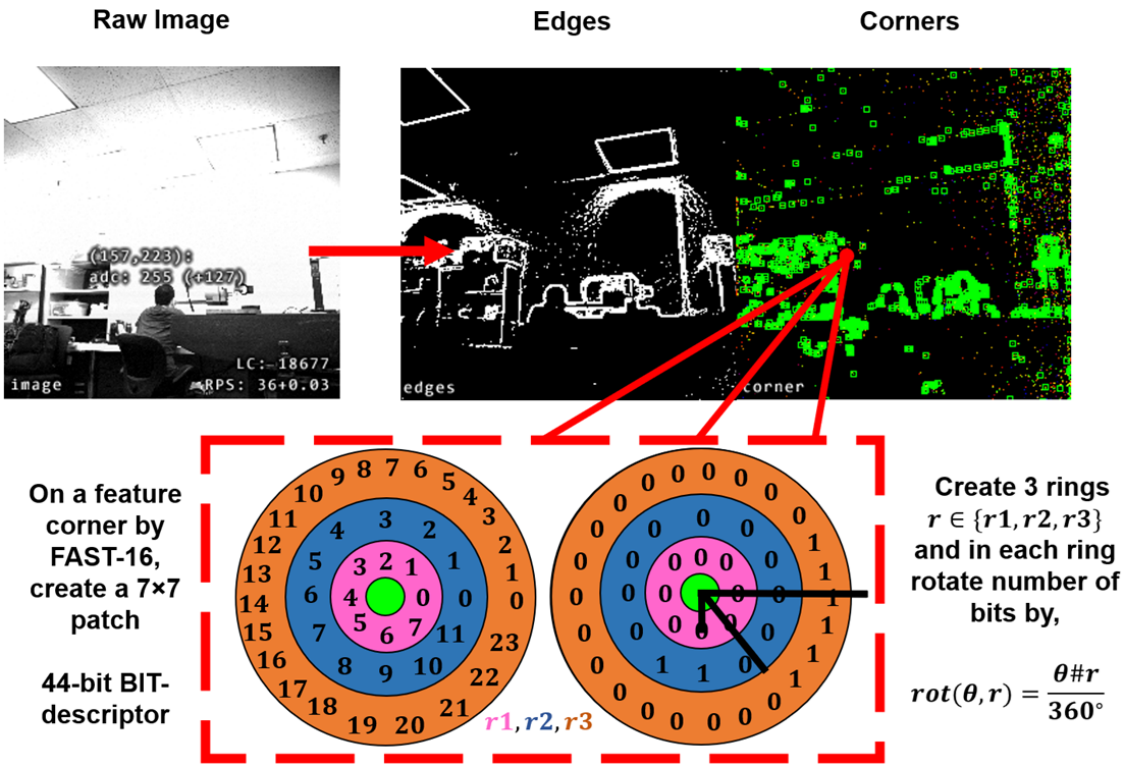


Figure 3.3: The front-end is from BIT-VO [17] where every incoming frame, the SCAMP-5 FPSP does edge computation and corner feature extraction, sends these computations to PC where back-end part of the algorithm computes BIT-descriptor on each corner feature, accounting for rotation invariance by $rot(\theta, r)$.

The visual features are further processed to obtain camera pose estimation (unscaled as the system is monocular), x^T_{BIT-VO}, which is composed of ${p^v_w}^T$, ${q^v_w}^T$, i.e. position and orientation. BIT-VO uses a BIT-descriptor (44-bit long feature), which is created from local binary edge information around the corner features and is used to establish feature correspondences between frames.

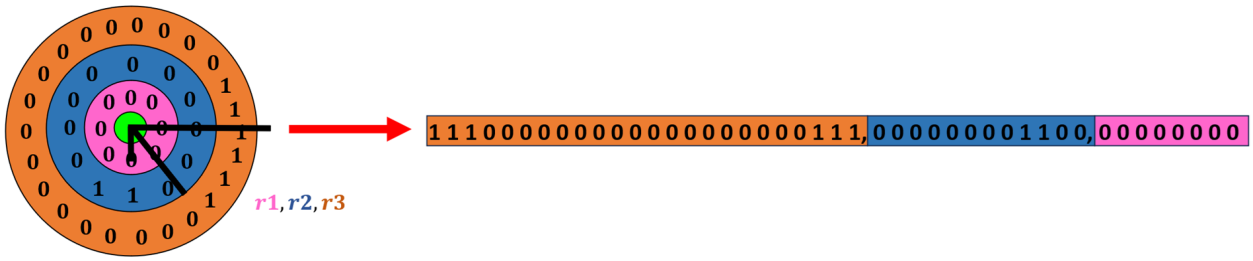


Figure 3.4: After all feature corners' BIT-descriptors are computed, 44-bit long feature is unrolled for space-efficient, fast bit-wise manipulation and computing in back-end algorithm for frame-to-frame feature tracking and map-to-frame feature mapping.

It differs from other binary descriptors because BIT-VO does not have access to the image intensity information. To create a BIT-descriptor, around a corner feature, BIT-VO creates a 7×7 patch and rotates the patch to be rotationally invariant. In the 7×7 patch, BIT-VO creates 3 rings $r \in \{r_1, r_2, r_3\}$. To establish a correspondence, Hamming Distance between two features (as the feature is binary) is taken to measure how similar two descriptors are. Though the BIT descriptor is rotation invariant, it is not scale invariant.

Like the standard VO, initialization is done by the 5-point algorithm with RANSAC. In pose estimation, RANSAC helps determine the optimal pose transformation that aligns a model (such as a 3D object or keypoints) with a scene (such as a point cloud or image). The objective is to find the pose that minimizes the differences between the model and the scene, considering the presence of outliers or incorrect correspondences [21]. RANSAC used for pose estimation [21] uses a five-step process: randomly select a sample, generate a hypothesis, evaluate the number of inliers, refine the hypothesis and repeat until fixed number of iterations or until criterion complete, which in this case is five points. RANSAC is similar in dealing with uncertainty, but as RANSAC can be applied to different forms of data (features, re-projections, etc.), the uncertainty is flexible and dependent on the feature data.

The high frame rate feature detection using SCAMP-5 FPSP simplifies the frame-to-frame and map-to-frame matching processes, as the inter-frame motion between frames is small. This allows the feature matching to be based upon simple brute-force search-and-match around a small radius.

Algorithm 6 5-Point Algorithm with RANSAC for Initialization from [21] (refer to for more detail)

Input: Point correspondences $\{(p_i, p_i')\}$, iterations N, threshold ϵ
Output: Essential matrix $\mathbf{E}_{\text{best}}$
for $i = 1$ to N **do**
    Randomly select 5 point correspondences $\{(p_j, p_j')\}_{j=1}^{5}$ ▷ Select random sample
    Formulate the constraint $\tilde{\mathbf{q}}^T \mathbf{E} = 0$: ▷ Set up constraint equation

$$\tilde{\mathbf{q}}_j = [x_j x_j', y_j y_j', 1] \otimes [x_j', y_j', 1] \tag{3.7}$$

    Construct matrix $\mathbf{Q}$ from $\tilde{\mathbf{q}}_j$: ▷ Construct matrix from constraints

$$\mathbf{Q}\mathbf{e} = 0 \tag{3.8}$$

    Solve for $\mathbf{e}$ (vector form of $\mathbf{E}$) ▷ Solve the homogeneous equation
    Decompose $\mathbf{E}$: ▷ Perform SVD to decompose $\mathbf{E}$

$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \quad \mathbf{D} = \text{diag}(1, 1, 0) \tag{3.9}$$

    Count inliers: ▷ Evaluate inliers based on current $\mathbf{E}$
    $inliers = 0$
    for each (p_k, p_k') **do** ▷ Iterate over all correspondences
        if $\left| {p_k'}^T \mathbf{E} p_k \right| < \epsilon$ **then** ▷ Check if correspondence is an inlier
            $inliers \leftarrow inliers + 1$ ▷ Increment inlier count
        end if
    end for
    if $inliers > best\_inliers$ **then** ▷ Update best model if current one is better
        $best\_inliers \leftarrow inliers$
        $\mathbf{E}_{\text{best}} \leftarrow \mathbf{E}$
    end if
end for
return $\mathbf{E}_{\text{best}}$ ▷ Return the best essential matrix

The map refinement and keyframe selection of the BIT-VO algorithm are similar to PTAM [34] and SVO [73].

Once the 3D map points and their corresponding k-projected points on the image plane are found, the pose is estimated by minimizing the reprojection error:

$$[p_v^{cT}, q_v^{cT}] = \underset{[p_v^{cT}, q_v^{cT}]}{\text{argmin}} \frac{1}{2} \sum_{i=0}^{k} \rho \left(\| u_i - \pi(T_v^c \cdot {}_v p_i) \|^2 \right), \tag{3.10}$$

where $\pi(T_v^c \cdot {}_v p_i)$ is the function projecting 3D points on the vision image plane and $\rho(\cdot)$ is the Huber loss function, reducing the effect of outlying data.

The x^T_{BIT-VO} (scaled with scale λ) part of the 10-element state is defined as:

$$x^T_{BIT-VO} = [\Delta\lambda, \delta\theta_i^{cT}, \Delta p_w^{v\,T}, \delta\theta_w^{v\,T}] \tag{3.11}$$

Assuming BIT-VO vision sensor measurement z_{BIT-VO} has Gaussian noise in position and rotational n_p, n_q. The measurement model is given by,

$$z_{BIT-VO} = \begin{bmatrix} p_v^c \\ \\ q_v^c \end{bmatrix} \tag{3.12}$$

$$= \begin{bmatrix} C_{(q_w^v)}(p_w^i + C^T_{(q_w^i)} p_i^c)\lambda + p_w^v + n_{p_v} \\ \\ q_i^c \otimes q_w^i \otimes q_w^{v-1} + n_{q_v} \end{bmatrix}, \tag{3.13}$$

p_v^c, q_v^c propagate the state and do so at the BIT-VO vision sensor rate, which is the rate of 300 FPS [87] [86].

3.6 Uncertainty Propagation of FPSP BIT-VO Pose

BIT-VO itself does not propagate an uncertainty or consist of covariance for its vision 6- DoF pose. 3D map points and correspondences are computed on the PC, where the pose is optimized by minimizing the reprojection error. Once the optimal pose $[p_v^{cT}, q_v^{cT}]$ is found from the set, take the pose and, using Ceres [91], generate a 6×6 covariance block for the optimized parameters based on the optimal pose.

It starts with forming the Jacobian of the residual blocks with respect to $[p_v^{cT}, q_v^{cT}]$, then the Hessian H is approximated as,

$$H \approx J^T J. \tag{3.14}$$

Last, with the covariance being computed as the inverse of the approximated Hessian,

$$\Sigma = H^{-1} = (J^T J)^{-1} \tag{3.15}$$

Note, here the covariance matrix is a 6×6 positive definite matrix, correctly matching the system's DoF rather than the state's dimensionality (which is 7 because have 3 parameters for the translation and 4 parameters for the quaternion).

3.7 Correcting State via iEKF

Build full covariance matrix $\bar{P}_{k+m|k}$. Then, the update is computed as:

$$\tilde{z}_k = H_{k+m}\tilde{X}_{k+m|k} + H_k\tilde{X}_{k|k} + \eta, \tag{3.16}$$

where H is the measurement Jacobian found via $z_{\text{BIT-VO}}$. Then compute residual as,

$$r_{k+m} = z_{k,k+m} - \hat{z}_{k,k+m} \overset{\sim}{=} \bar{H}\bar{X}. \tag{3.17}$$

Compute innovation as,

$$\bar{S}_{k+m} = \bar{H}\bar{P}_{k+m|k}\bar{H}^T + R_r, \tag{3.18}$$

where R_r is covariance of measurement and,

$$\bar{H} = [H_{k|k}, H_{k+m|k}]. \tag{3.19}$$

Then compute Kalman gain and correct state,

$$\bar{K} = \bar{P}_{k+m|k}\bar{H}^T\bar{S}_{k+m}^{-1} = [K_k^T, K_{k+m}^T]^T, \tag{3.20}$$

$$\hat{x}_{k+m|k+m} = \hat{x}_{k+m|k} + K_{k+m}r_{k+m}. \tag{3.21}$$

Then, correct the covariance as,

$$P_{k+m|k+m} = P_{k+m|k} - K_{k+m}\bar{S}_{k+m}K_{k+m}^T. \tag{3.22}$$

Note, can either assume BIT-VO vision sensor measurements as relative (as in depending between time-instants k and $k+m$) or absolute (e.g. IMU or GPS measurements).

3.8 Trajectory Performance Evaluation Metrics

Evaluating the accuracy and robustness of several visual-alone, visual-inertial algorithms to ground-truth is done by comparing their Absolute Trajectory Error (ATE) and Relative Trajectory Error (RTE) metrics. These metrics either give a whole-trajectory, or local-trajectory error comparison, respectively, and are the standard for VIO performance evaluations.

3.8.1 Absolute Trajectory Error (ATE)

ATE is a measure of the difference between the estimate trajectory with the ground-truth trajectory [92] along the whole of the trajectory. It can be taken as the absolute relative pose of estimate and ground-truth, taken at time t_i,

$$ATE(t_i) = P_{estimate,i} - P_{ground-truth,i} = P^{-1}_{estimate,i} P_{ground-truth,i} \in \text{SE}(3), \tag{3.23}$$

where $P_{estimate,i}, P_{ground-truth,i}$ are the relative pose of estimate and ground-truth at time t_i. It can be further broken into translational dealing with xyz and rotational Roll, Pitch and Yaw (RPY) ATE,

$$ATE_{trans} = (||\Delta T||^2)^{1/2} = ((\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2)^{1/2}, \tag{3.24}$$

$$ATE_{rot} = (||\Delta \theta||^2)^{1/2} = ((\Delta R)^2 + (\Delta P)^2 + (\Delta Y)^2)^{1/2}. \tag{3.25}$$

Generally in evaluation for SLAM comparisons [20], [36], the translational ATE is stated while the rotational accuracy of the estimate is expressed via trajectory RMSE [93].

RMSE shows the variance in deviation of the ATE and is taken as over the entire set of time t_i,

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ATE(t_i)^2} \tag{3.26}$$

The accuracy of BIT-VO compared with the BIT-VIO algorithm will be evaluated by measuring the ATE by reporting the RMSE and the median.

3.8.2 Relative Trajectory Error (RTE)

RTE is a measure of the difference between the estimate trajectory with the ground-truth trajectory, measuring not the overall trajectory path which ATE does, but instead measuring the local consistency of the path. It can be taken as comparing the relative poses of estimate and ground-truth, based on the delta pose difference, taken at times t_i, t_j,

$$RTE(t_i, t_j) = \delta_{estimate,i,j} - \delta_{ground-truth,i,j} \tag{3.27}$$

$$= (P^{-1}_{ground-truth,i} P_{ground-truth,j})^{-1} (P^{-1}_{estimate,i} P_{estimate,j}) \in \text{SE}(3), \tag{3.28}$$

where $P_{estimate,i,j}, P_{ground-truth,i,j}$ are the relative pose of estimate and ground-truth at time t_i, t_j respectively. It can be further broken into translational dealing with xyz and rotational RPY RTE with same form as in case of ATE.

Again, like with ATE, the RMSE, which is the variance in deviation of the RTE over the relative poses along the trajectory path can be taken as over the entire sets of time t_i, t_j,

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N_i}\sum_{j=1}^{N_j} RTE(t_i, t_j)^2} \tag{3.29}$$

It must be noted that when dealing with RTE, the parameter Δ is the constraint in which the relative pose frequency is based upon. This gives the constraint on the difference between relative poses and when to make the RTE measurement upon. Smaller Δ gives better local accuracy of an algorithm, as the relative poses should be measured as instantaneously against one after the other.

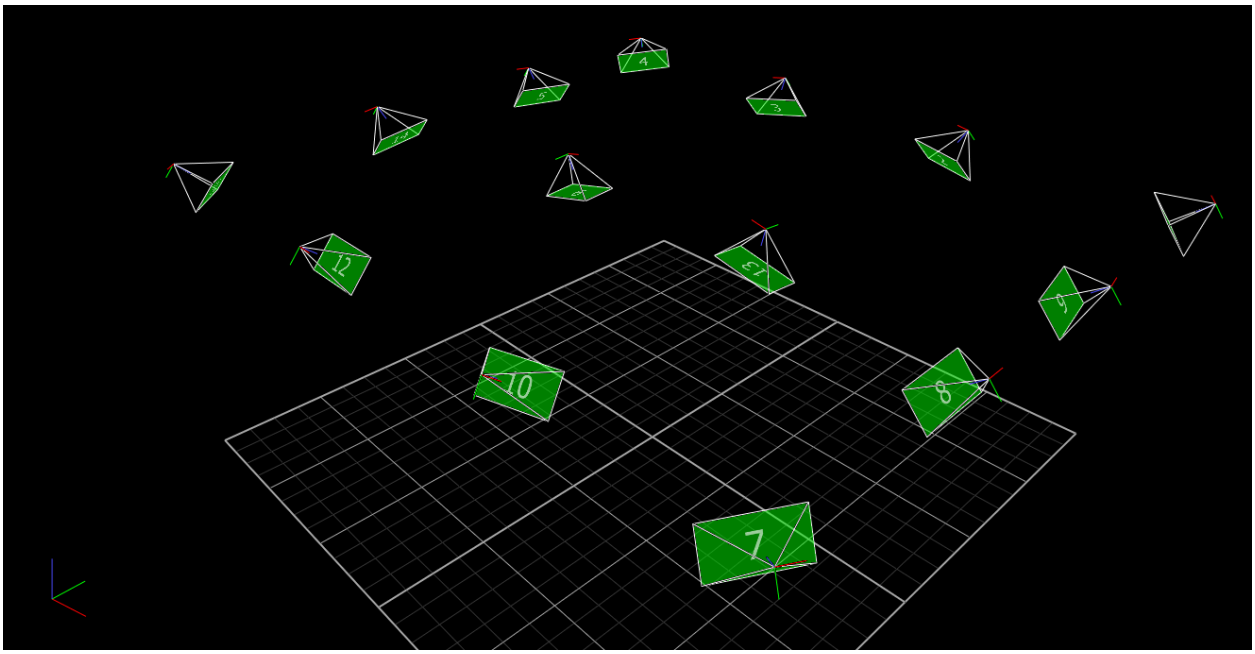


Figure 3.5: Vicon motion capture system for ground-truth tracking. 14 cameras are calibrated and time-synced.

3.9 Experimental Results: Initial Attempts

The proposed BIT-VIO algorithm was first tested on a moving robotic system with several IMU cases. It was initially tested with approximated visual-inertial sensor calibration, approximating $p_i^c = (0.035\ m, 0\ m, 0.125\ m)$ with respect to the IMU-frame. The IMU intrinsics, being the acceleration, gyroscopic, and bias noises, $n_a, n_\omega, b_a, b_\omega$, are initially estimated as well by directly using the provided data sheets for the different IMUs. This was done in hopes to evaluate the performance and accuracy of the algorithm using approximated calibration intrinsics and extrinsics, as well as see how robust the algorithm is in varying IMU cases. As mentioned, dealing with two IMU cases: UM7 at 200 Hz and MPU-9250 at 110 Hz, to achieve fully the BIT-VO vision sensor. The former two instances loosely sensor-fuse with BIT-VO at 100 FPS, as the update step in the iEKF model requires the rate to be less than or equal to it. To alleviate this, an IMU at the BIT-VO rate is needed.

The robotic system is evaluated against ground-truth data from a Vicon motion capture system, which consisted of 14 cameras calibrated and time-synced. Given the inherent limitations of the SCAMP-5 FPSP, which precludes its compatibility with video datasets for direct comparison, evaluation is extended to encompass four real-world trajectory scenarios. These scenarios, designed

to mimic practical applications, included Circular, Straight, Curve, and Zigzag trajectories. Note, what is necessary is to align and scale-adjust the recorded trajectories along ground-truth as this is monocular SLAM. The host computations were made on a PC host or external computer, with 12th Gen Intel Core i7-12700 CPU.

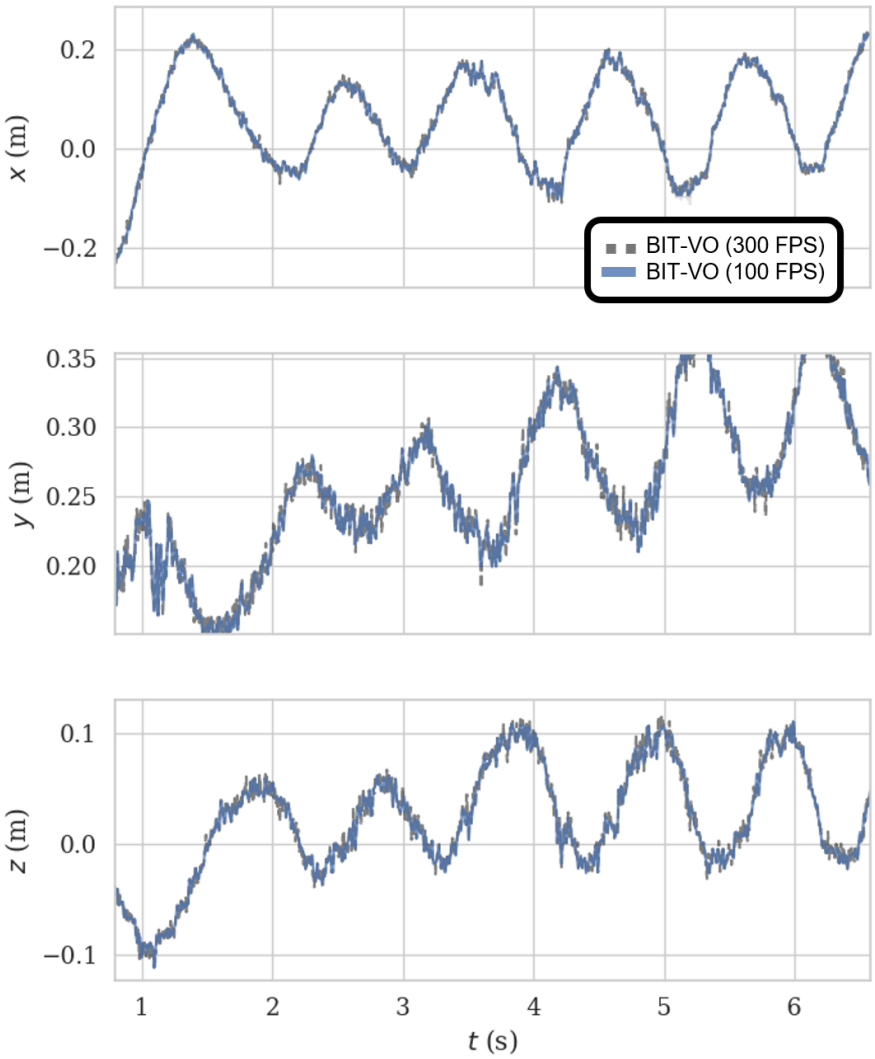


Figure 3.6: BIT-VO at 300 FPS in dashed-grey is taken as the reference, with BIT-VO at 100 FPS taken to be the estimate. BIT-VO maintains itself well between different FPS, and this makes sense as 100 FPS is a sampling on the 300 FPS. For lower rate IMU cases.

3.9.1 BIT-VO from 300 FPS to 100 FPS

With UM7 at 200 Hz and MPU-9250 at 110 Hz, what is required is to have BIT-VIO to operate at a lower rate, e.g., 100 FPS in these IMU cases. To ensure that the accuracy and performance of BIT-VO is maintained at a lower rate, BIT-VO at 300 FPS and at 100 FPS are compared on the

same trajectory. Taking 100 FPS to be just a sampling of the 300 FPS BIT-VO at a lower rate, it is clear that the difference between these cases is negligible, and so choosing to bottleneck BIT-VO at this lower rate so that the update step in the iEKF model meets its requirement of having its IMU-based prediction rate to be less than or equal to it, is fair in evaluation.

3.9.2 UM7 IMU at 200 Hz with BIT-VO at 100 FPS Results

First, the performance of BIT-VIO is evaluated using the UM7. As the translational ATE shows in Table 3.1, incorporating an IMU generally enhances estimation, yielding a more accurate trajectory. Trajectory 1, 4, and 6 performed with lower RMSE and median, showing that the predictions are close to the ground truths. However, in Trajectory 2 and 3, purposefully simulate a fast, hostile motion. Under these conditions, the IMU, operating at 200 Hz, failed to contribute valuable data to the system, and consequently, the VIO's performance did not surpass that of the VO system.

Translational ATE with UM7 at 200 Hz, BIT-VO at 100 FPS			
Traj.	Type	BIT-VO ATE (m)	BIT-VIO ATE (m)
1	RMSE:	0.019694	**0.017769**
	median:	0.019483	**0.010923**
2	RMSE:	**0.015603**	0.020269
	median:	**0.01383**	0.016463
3	RMSE:	**0.015848**	0.020321
	median:	**0.014144**	0.020184
4	RMSE:	0.009329	**0.003849**
	median:	0.008104	**0.003564**
5	RMSE:	**0.013718**	0.014460
	median:	0.009872	**0.008942**
6	RMSE:	0.025920	**0.026486**
	median:	0.019431	**0.012808**

Table 3.1: UM7 at 200 Hz and BIT-VO at 100 FPS has good translational ATE for BIT-VIO in most cases.

Such trajectories are typically challenging for state estimation algorithms, especially when the vision system is running at a low frame rate (e.g., 30 FPS on conventional camera technology or 100 FPS limitation, have to impose on BIT-VO due to limited IMU readout frequency). A low frame rate means slower feature tracking and matching, thus increasing the overall latency of the state estimation. As demonstrated in BIT-VO [17, 94], running the vision system with a high frame

rate increases the robustness against such hostile motions. Hence, with confidence, it can be said that utilizing an IMU with a higher frequency and, thus, running BIT-VO with a higher frame rate would lead to different outcomes.

In evaluating the translational RTE shown in Table 3.2, what is shown is a similar trend as with the ATE case, but emphasizing more that prior BIT-VO has higher RTE, implying that it is less locally consistent along its relative trajectory points than the BIT-VIO algorithm. In investigating the instantaneous local relativity between successive points along the trajectory path, here, RTE is set to take a measurement of RTE every $\Delta = 0.0001$ m. Traj. 1, 4 to 6 across have lower median RTE between successive points. As was noted, Traj. 2 and 3 simulate fast hostile motions and the trend that VO update failed to contribute and correct the system is evident here in the RTE statistics too where VO has nearly the same local consistency as with the FPSP-based estimate.

Translational RTE with UM7 at 200 Hz, BIT-VO at 100 FPS with $\Delta = 0.0001$ m			
Traj.	Type	BIT-VO RTE (m)	BIT-VIO RTE (m)
1	RMSE:	0.010753	**0.012644**
	median:	0.003906	**0.003419**
2	RMSE:	**0.010753**	0.012686
	median:	**0.003906**	0.003910
3	RMSE:	**0.006195**	0.013241
	median:	0.004802	**0.004365**
4	RMSE:	**0.002821**	0.004280
	median:	0.002554	**0.000908**
5	RMSE:	**0.002895**	0.003178
	median:	0.002527	**0.000799**
6	RMSE:	**0.003173**	0.004405
	median:	0.002733	**0.000913**

Table 3.2: UM7 at 200 Hz and BIT-VO at 100 FPS has good translational RTE for BIT-VIO in only some cases.

As an initial attempt in working with real-world sensor data, the acquired translational ATE and RTE show that BIT-VIO is accurate and robust showing working consistency with a abstract IMU case with approximated calibration intrinsics and extrinsics, but, at this stage, seemingly only as accurate as prior BIT-VO, not better. There seem to be as many cases where BIT-VO is to the same degree of accuracy, sometimes even better than BIT-VIO. It is clear that utilizing an IMU with a higher frequency and a higher frame rate VO system for more robust feature tracking is needed.

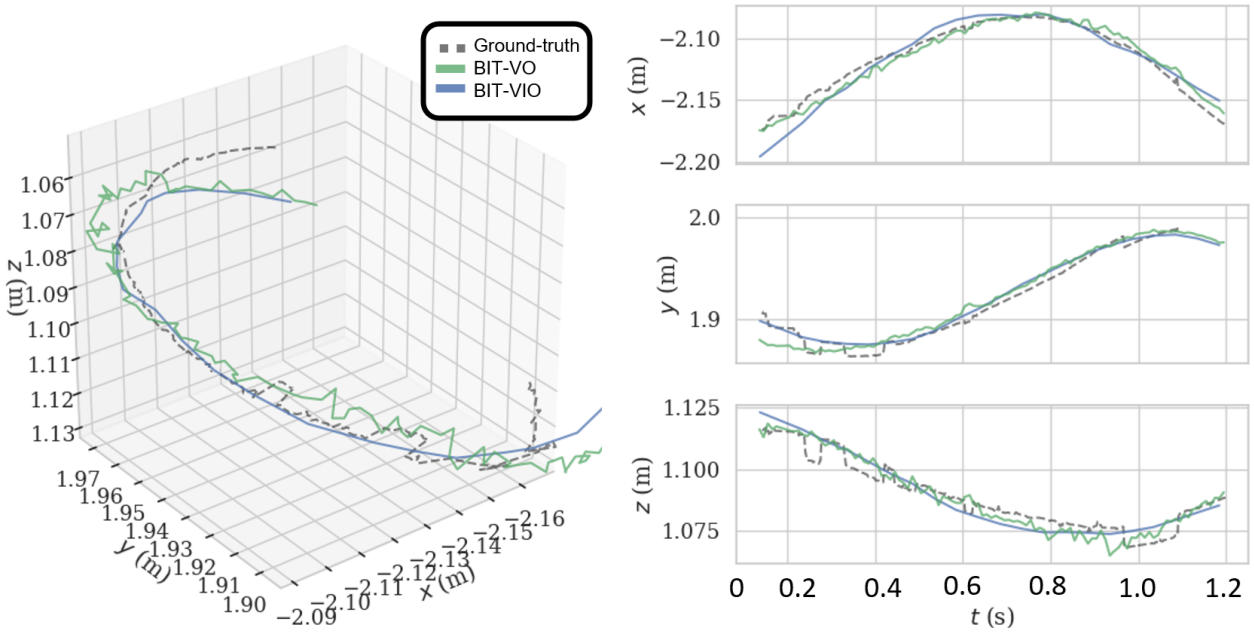


Figure 3.7: BIT-VIO has less frequency noise than BIT-VO. Estimates are close but BIT-VIO is smoother. This is Traj. 5.

3.9.3 MPU-9250 IMU at 110 Hz with BIT-VO at 100 FPS Results

Next, in evaluating with the MPU-9250 IMU at 110 Hz, keeping the same FPS for BIT-VO, it can be assessed that IMU offers minimal to no supplementary data as shown in the translational ATE in Table 3.3. In Traj. 1 and 3, BIT-VIO does not perform better than the BIT-VO. As anticipated, when the IMU publishes at a lower frequency compared to Table 3.1, the ATE of the BIT-VO and BIT-VIO becomes similar and shows minimal difference in RMSE [93] and median values for each trajectory. This is because the IMU propagation is at or near the vision rate.

Translational ATE with MPU-9250 at 110 Hz, BIT-VO at 100 FPS			
Traj.	Type	BIT-VO ATE (m)	BIT-VIO ATE (m)
1	RMSE:	**0.005498**	0.007680
	median:	**0.00519**	0.006496
2	RMSE:	0.002743	**0.002637**
	median:	**0.002066**	0.002174
3	RMSE:	**0.005652**	0.006625
	median:	**0.005068**	0.005845
4	RMSE:	0.005504	**0.004457**
	median:	0.005067	**0.003966**

Table 3.3: MPU-9250 at 110 Hz and BIT-VO at 100 FPS has poor translational ATE.

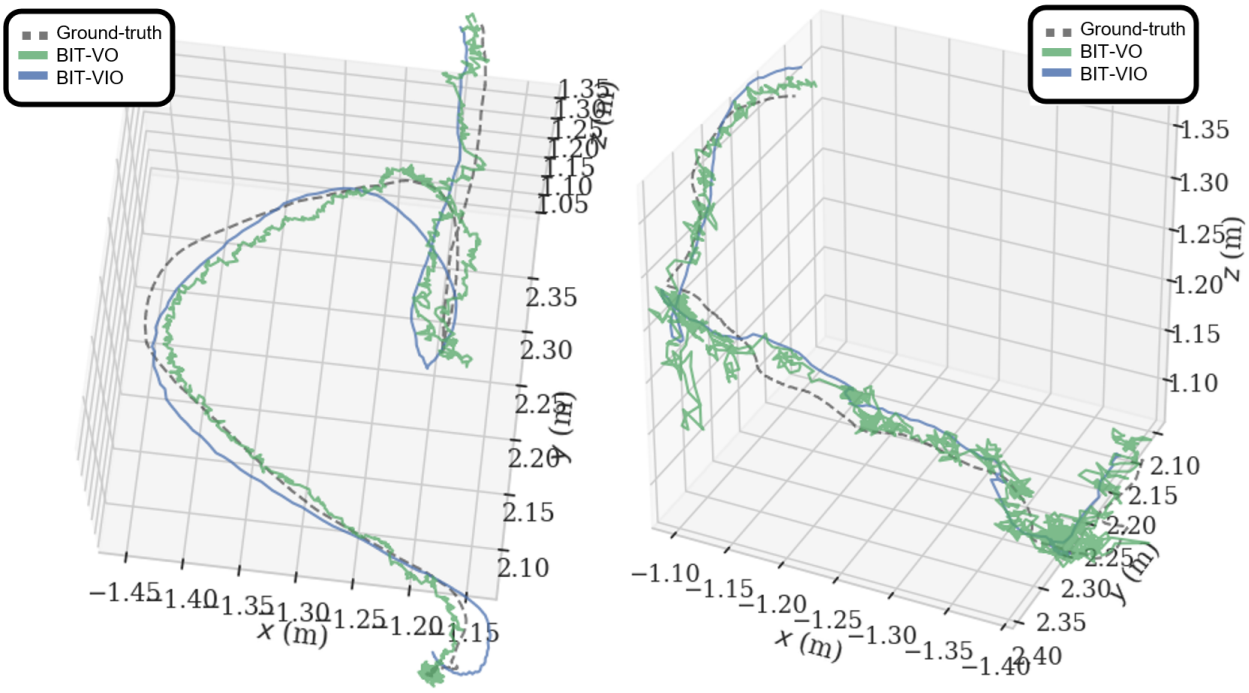


Figure 3.8: Much longer trajectories with the MPU-9250 where BIT-VIO still has lower ATE. Traj. 1 (left) and 4 (right).

The difference in rates between prediction and update of an iEKF model does not have to be strictly different or have a large difference, but with larger differences in the rate, this allows the dynamic models to wait some time, propagate itself with update correcting after some propagation, resulting in a more stabler state from its prior. This is also good because the IMU is taken with more preference and influence over the robotic system's state. IMUs are more reliable in pose estimation than vision and should be given more control of the filtered trajectory. If the difference was not large, the stability of the state would deter, and this would now be allowing for vision to propagate itself and update the state at the same rate, and as known, the vision is not as reliable in its pose estimate. Just finding the correct difference between the IMU rate and the VO rate is imperative for accurate FPSP- IMU-fused estimation.

Further, as was apparent in the last IMU case, one of the main limitations of BIT-VO was the high-frequency noise in the predicted trajectory. Though the VIO and VO performances are comparable with MPU-9250 IMU at 110 Hz with BIT-VO at 100 FPS, the filtered estimates as shown in Table 3.3 shows how the BIT-VIO algorithm predicts significantly smoother trajectory than trajectories predicted from BIT-VO. The ATE of the BIT-VIO algorithm has lower error than prior BIT-VO, but in doing these experiments, it is clear that proper visual-inertial sensor calibration

is necessary. Further, the full capabilities of BIT-VO are silenced by the cases of sampling down due to the limitations of low frequency IMUs. So, not only does it seem that the low rates affect the performance of the fused system, but also the differences between the two need to be accounted for, where the rates not only need to be operating with high frequency but also with large difference.

Translational RTE with MPU-9250 at 110 Hz, BIT-VO at 100 FPS with $\Delta = 0.0001$ m			
Traj.	Type	BIT-VO RTE (m)	BIT-VIO RTE (m)
1	RMSE:	**0.002850**	0.009099
	median:	**0.002709**	0.006564
2	RMSE:	**0.004855**	0.006613
	median:	**0.003863**	0.006096
3	RMSE:	0.004793	**0.003274**
	median:	0.003242	**0.002625**
4	RMSE:	0.013372	**0.007863**
	median:	0.010326	**0.006580**

Table 3.4: MPU-9250 at 110 Hz and BIT-VO at 100 FPS has poor translational RTE.

In evaluating the translational RTE shown in Table 3.4, there, again, is a similar trend like with the last IMU case, specifically between the ATE and RTE statistics. Taking RTE $\Delta = 0.0001$ m, the general trend that BIT-VO has larger RTE is maintained compared to the lower error provided by the BIT-VIO algorithm, though with some instances not. Unlike the ATE trend, Traj 2. and 3 show the opposite trend for the RTE, which implies that that the small difference in the rates between the IMU and VO system makes it so the BIT-VIO algorithm is consistent in showing significant improvement over prior BIT-VO, especially in the relative trajectory error context, where it is expected that the high frequency noise of the VO system, and hence the relative error is alleviated.

Like with the last IMU case, as another initial attempt in working with real-world sensor data, the acquired translational ATE and RTE show that BIT-VIO is accurate and robust showing working consistency with another abstract IMU case with approximated calibration intrinsics and extrinsics, but, it is further clear that utilizing a FPSP- IMU-fused system where the difference in the rates is large is also imperative. Now, the full 300 FPS capability of BIT-VO will now be studied with a faster IMU along with proper sensor calibration to address the two problems associated with these IMU cases.

3.10 Experimental Setup with Visual-Inertial Sensor Calibration

In the last section, it was indeed verified that any black box IMU works with the proposed BIT-VIO framework, in how several IMU cases were presented sensor fusing with the BIT-VO algorithm from the SCAMP-5 FPSP. Though, these IMU case do not fully utilize the FPSP VO at its full high framerate capability, as they bottleneck with the UM7 at 200 Hz and the MPU-9250 at 100 Hz. It was clear also that using approximated calibration intrinsics and extrinsics in the initial attempts section can greatly hinder the BIT-VIO algorithm's performance and accuracy, and a proper calibration then is shown to be necessary. It is then next in this section to properly calibrate the FPSP camera and IMU and to measure the expected improvement in the performance of the robotic system. To start, to address resolving the bottleneck of the IMU frequency rate, so that BIT-VO can fully utilize the FPSP VO at its full high framerate capability for vision- IMU-fused state estimation, the proposed BIT-VIO algorithm is tested at now the full 300 FPS, running BIT-VO on a SCAMP-5 FPSP device. Next is to do the proper camera- IMU calibration. The UM7 and MPU-9250 are swapped out, and now the robotic system has its SCAMP-5 FPSP attached to an Intel D435i RealSense Camera to provide the IMU measurements at 400 Hz. This setup is using an IMU that does not limit the high framerate achievable by FPSP BIT-VO.

Evaluations are again done against ground-truth data from a Vicon motion capture system, which consisted of 14 cameras calibrated and time-synced. As both BIT-VO and BIT-VIO assume a fast frame rate, hence small inter-frame motion, again the method of using a standard benchmark dataset for direct comparison with other methods cannot be directly used. Hence, BIT-VIO is evaluated now on eight real-world trajectories against BIT-VO. These trajectories, like in last section with the UM7 and MPU-9250 IMU are designed to mimic practical applications and are a compilation of Circular, Straight, Curved, and Zigzag trajectories.

The recorded trajectories are aligned and scaled to the ground-truth trajectory as the setup is again monocular. As was done in last section, the ATE [92] will again be measured by reporting the RMSE [93] together with the median to evaluate the accuracy against the ground-truth. A different PC is used as the high framerate processing is expected to require more computational capability. BIT-VO and BIT-VIO use a host device to perform the VO backend, and for the host device, use an new external laptop with 13th Gen Intel Core i7-12700 CPU.

3.10.1 IMU-Alone Sensor Calibration

The IMU measurements inherently suffer from noise and must be calibrated prior to experiment evaluations. In the last section, there was an attempt to approximate the intrinsic values using the datasheet, but that proved to affect greatly the accuracy and consistency of the BIT-VIO algorithm's performance.

To calibrate the IMU intrinsics: acceleration, gyroscopic, and bias noises, $n_a, n_\omega, b_a, b_\omega$, what is needed is to conduct a calibration using the Allan variance method [95]- [96], which involves fine-tuning the estimation parameters for the IMU's acceleration and gyroscopic biases and noises. Allan variance is a time domain analysis technique used in signal analysis for defining noise and stability of a system in time [97]. The Allan deviation helps to visualize characteristics of the noises inherent in the system. This is done by discretizing the long sequence IMU data in bins $(b_1(t), b_2(t), ..., b_n(t))$ and then calculating the Allan variance [97] as,

$$\text{AVAR}(t) = \frac{1}{2 \cdot (n-1)} \sum_i \left(b_{i+1}(t) - b_i(t)\right)^2 . \tag{3.30}$$

From this, the Allan deviation can then be found by,

$$\text{ADEV}(t) = \sqrt{\text{AVAR}(t)}. \tag{3.31}$$

From the Allan deviation plots directly, this approach allows analysis of the sensor readings across various observation intervals, providing the white noise and bias inherent intrinsic measures in the system. With this information, the noise can be directly accounted for, expectedly enhancing the accuracy and reliability of the measurements.

The IMU-alone calibration process is done by leaving the IMU on a flat surface, its data to be recorded for a duration of time, with outputting noise. Extra caution should be taken to make sure the IMU was stationary as the noise processes are delicate and the Allan deviation is sensitive to outlier data. Did $t = 18$ hr IMU data stationary recording, but for the sake of consistency and accuracy, used IMU data with duration of time $t = 3$ hr with timestamps already arranged, at 400 Hz stationary provided by Patrick Geneva [98].

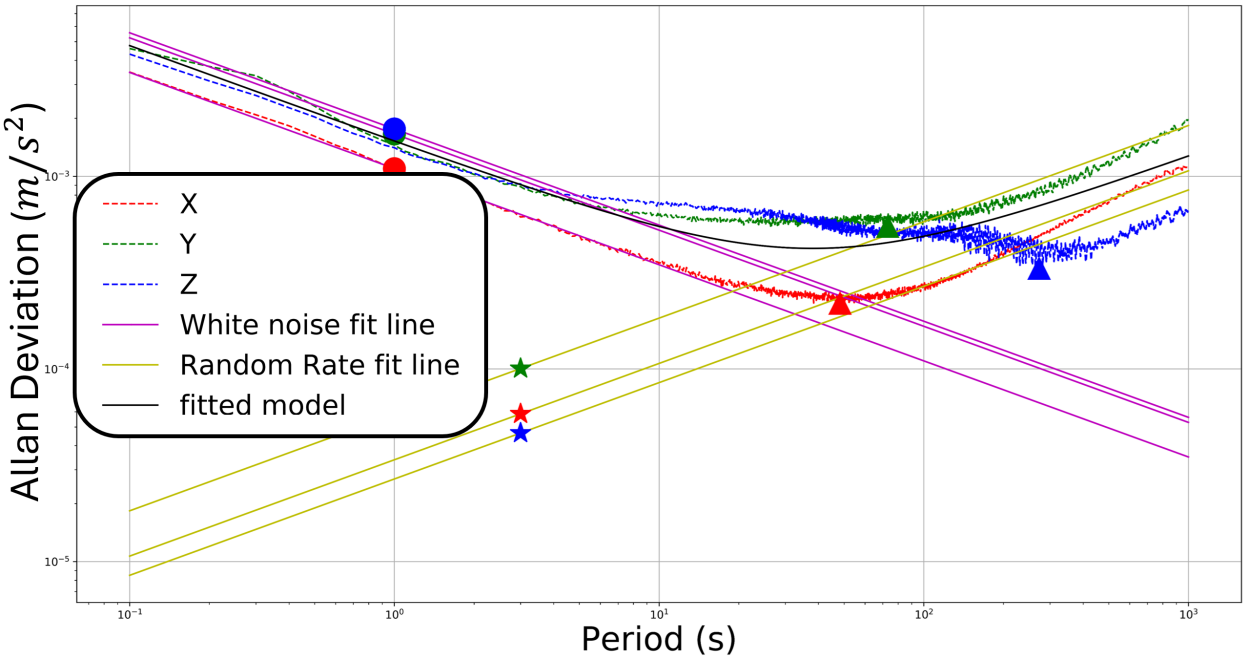


Figure 3.9: Acceleration Allan Deviation plotting. These plots give the intrinsics related to the accelerometer of the IMU.

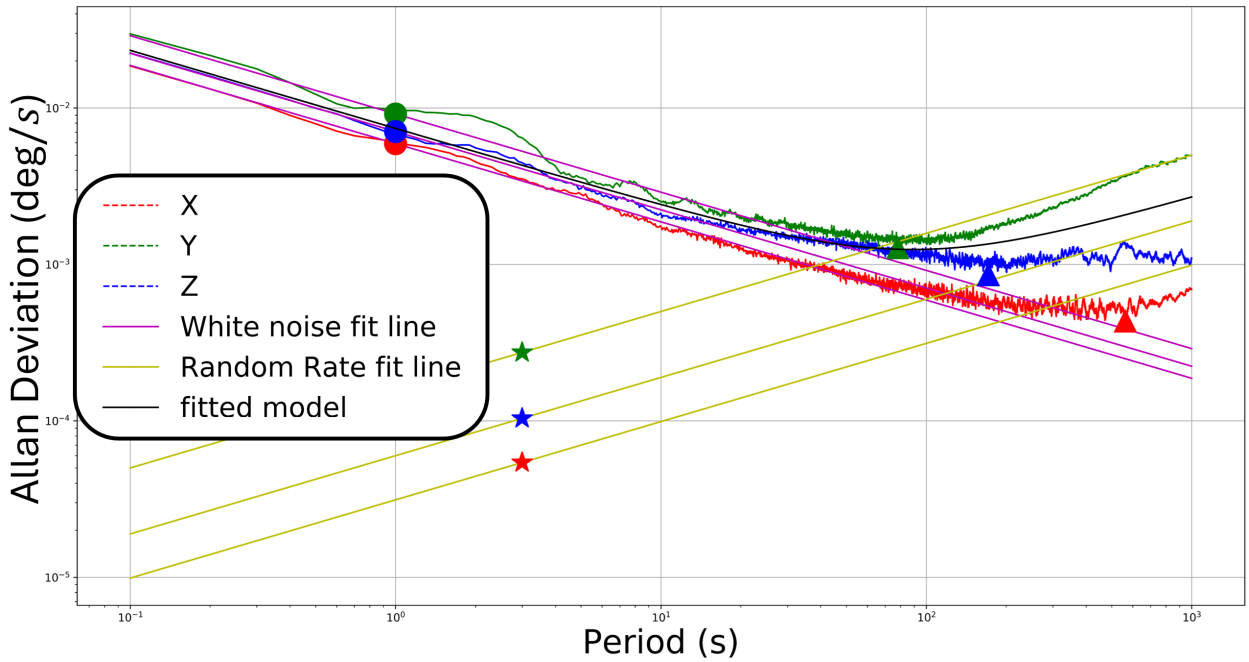


Figure 3.10: Gyroscope Allan Deviation plotting. These plots give the intrinsics related to the gyroscope of the IMU.

The Allan deviation ADEV(t) plot gives the following raw IMU intrinsics. Typically, these raw IMU intrinsic values by the Allan deviation are not directly used without first some correcting. The white noise and bias estimates are inflated several orders magnitudes to give less weighting on the IMU measurements if dealing with a standard or lower-cost IMU, as used here. Inflate the white noise by $\times 5$ and random walk by $\times 10$ to get corrected accelerometer noise density and random walk, and gyroscope noise density and random walk are in Table 3.5.

	Accelerometer		Gyroscope	
	Noise Density (n_a)	Random Walk (r_a)	Noise Density (n_ω)	Random Walk (r_ω)
Raw	0.00176 m/s$^2/\sqrt{\text{Hz}}$	1.0053×10^{-4} m/s$^3/\sqrt{\text{Hz}}$	0.0001598 rad/s$/\sqrt{\text{Hz}}$	4.712×10^{-6} rad/s$^2/\sqrt{\text{Hz}}$
Corrected	0.0088 m/s$^2/\sqrt{\text{Hz}}$	1.0053×10^{-3} m/s$^3/\sqrt{\text{Hz}}$	0.000799 rad/s$/\sqrt{\text{Hz}}$	4.712×10^{-5} rad/s$^2/\sqrt{\text{Hz}}$

Table 3.5: IMU-alone calibration: accelerometer and gyroscope intrinsics. The raw and inflated measures are provided.

3.10.2 Camera-Alone Sensor Calibration

Next, is to calibrate and acquire the intrinsics of the SCAMP-5 FPSP camera. Some modifications must be made for proper camera-calibration: the front-end system needs to be swapped from outputting corner and edge coordinates for the bit-descriptor, to instead just output plain grayscale 256×256 images for the sake of calibration optimization simplicity. Further, the high frame rate of the 300 FPS stream needs to be bottle-necked at 30 FPS, as it would mean less data to be post-processed by the optimizer. Note, there is no issue calibrating with 300 FPS but that would greatly slow the processing runtime.

Kalibr [99] is used in the camera-alone calibration, where a calibration target on the outputting images is needed. What was used was the supported 6×6 Aprilgrid [100], [101]. Over alternative calibration targets, the Aprilgrid is operable even in occluded or partially occluded scenes, as well, that the checkerboard tiles are unique and provide reprojection such that the camera pose is fully resolvable (i.e. with no flips or non-uniqueness).

Data consisted of $t = 60$ s IMU at 400 Hz and images at 30 FPS. It was made sure that all IMU axes were excited, that there was no abrupt jitter in the start or end of the trajectory path, that the shutter was low as well that the scene was illuminated properly [99], [95].

The camera projection model used in data was pinhole-configuration with a lens distortion model assumed to be radial-tangential (radtan) [102], [103] as the lens is spherical-type and the optimizer provided the following estimates shown in Table 3.6.

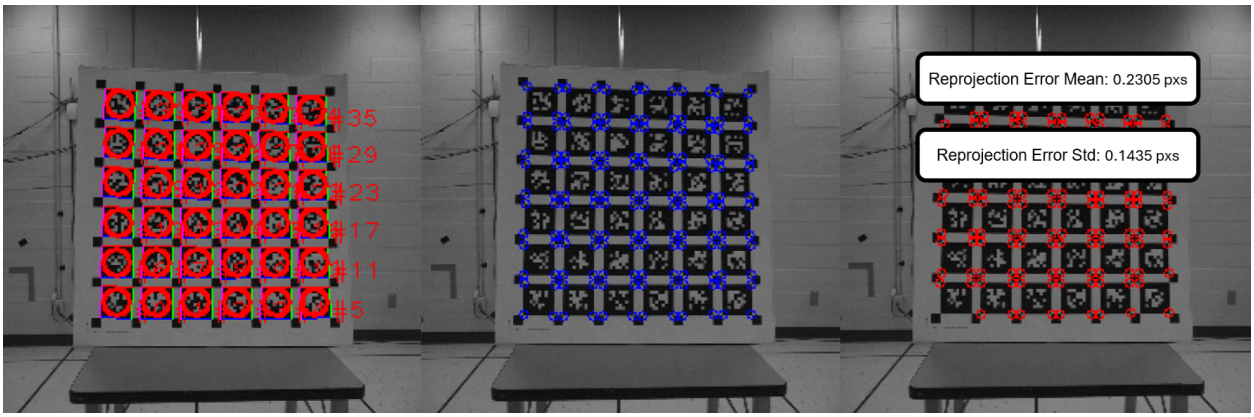


Figure 3.11: Camera-Alone calibration: Aprilgrid is used to track tags (far left), corners (middle), reprojection error (far right).

Camera-Alone SCAMP-5 FPSP Calibration Intrinsics	
Camera Model	Pinhole
Intrinsics Vector	$[\mathrm{f}_u, \mathrm{f}_v, \mathrm{p}_u, \mathrm{p}_v]$
Focal Length $(\mathrm{f}_u, \mathrm{f}_v)$	$[257.2735, 258.0083]$
Principal Point $(\mathrm{p}_u, \mathrm{p}_v)$	$[127.4410, 128.1666]$
Distortion Model	Radial-Tangential (Radtan)
Distortion Coefficients $(\mathrm{k}_1, \mathrm{k}_2, \mathrm{r}_1, \mathrm{r}_2)$	$[-0.0845, 0.2569, -0.00049, 0.0016]$

Table 3.6: Camera-Alone calibration: camera model and distortion model intrinsics. This is for SCAMP-5 FPSP.

Seeing as the optimizer estimated $(f_u, f_v) = (257.27 \text{ pxs}, 258.00 \text{ pxs})$ as known and expected utilizing of 256×256 resolution of SCAMP-5 FPSP, estimates are within reason. Further, the reprojection error of the polar angle and azimuthal angle are less than one-pixel with a major count in traversing a large range of angle values over the trajectory path.

The camera-alone calibration is also verified in the robustness in tracking of the camera pose reprojections of the Aprilgrid checkerboard labels. Notice how the spacing between grid element labels is consistent both horizontally and vertically between neighbours, with no loss in tracking the whole 6×6 over time, meaning that the estimates are optimized and set. Further,the reprojection error is also within a less than one-pixel bound, emphasizing the accuracy of the derived estimates.

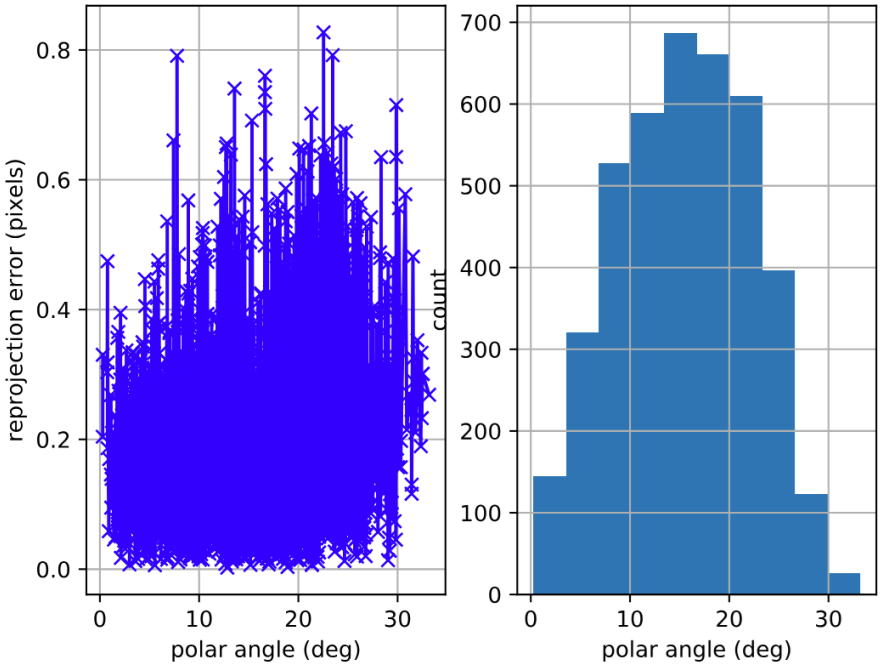


Figure 3.12: SCAMP-5 FPSP polar error. See within less than one-pixel (left). Histogram of frequency of polar angle (right).

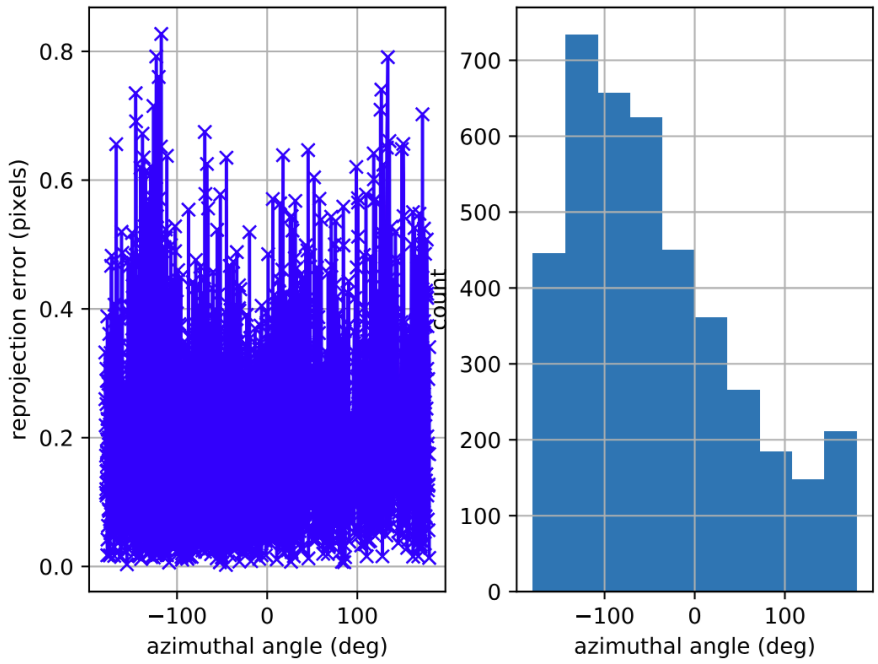


Figure 3.13: SCAMP-5 FPSP azimuthal error. See within less than one-pixel (left). Histogram of frequency of azimuthal angle (right).

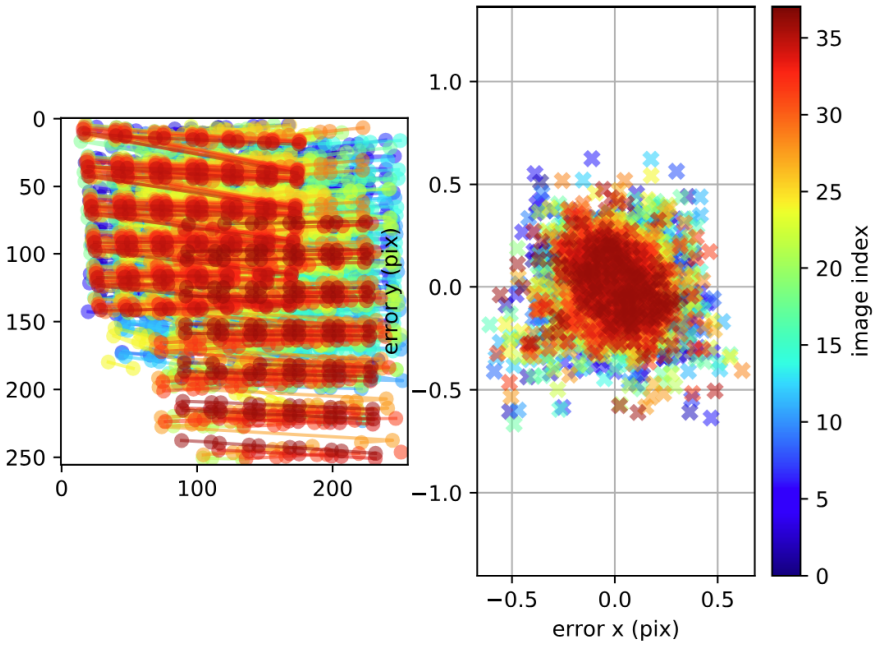


Figure 3.14: SCAMP-5 FPSP reprojection tracking robust (left) and error is within less than 0.5-pixel (right).

With both IMU-alone calibration and SCAMP-5 FPSP camera-alone calibration intrinsic estimates optimized and determined, the extrinsic calibration between the joint-two is now necessary.

3.10.3 Camera-IMU Sensor Calibration

Finally it is time to calibrate the extrinsic calibration between the SCAMP-5 FPSP and IMU. This looks to find the spatial transformation between the camera and IMU frames whilst finding the temporal time offset between both. The optimization then does this simultaneously.

Here, the calibration process uses Kalibr [99] again where the estimates are optimized by a full batch optimization using spline interpolation to fit a curve which models the pose trajectory of the system. More information on the optimization process is provided in [99], [95]. For this particular problem, the optimization is addressed as having spline order 6 where the pose spline is initialized with 4862 knots to 5062 knots with 10143 design variables and 168722 error terms. The camera- IMU process optimizes on a Jacobian matrix of size 418526×45627 using back-to-back Block-Cholesky as the linear system solver and Levenberg-Marquardt [104] as the trust region policy.

The camera- IMU calibration provides the following extrinsic estimates are shown in Table 3.7.

The last column top 3×1 of both T_{ci}, T_{ic} represents the position p_{ci}, p_{ic} respectively between the frames. The spatial estimates are within reason as a simple measure with the ruler gives

approximately these values. As for the temporal calibration, as the SCAMP-5 FPSP was done with recorded images outputting at 30 FPS the time offset is as expected, to be -0.0264 s which translates to -37.87 Hz difference. Outputting at 300 FPS the time offset is then as expected -0.0033 s which is a -303.03 Hz difference. With both the spatial and temporal calibration done, the parameters must now be verified.

Camera-IMU SCAMP-5 FPSP Calibration Extrinsics (Transformations and Time Offsets)	
T_{ci} (IMU to SCAMP-5 FPSP)	T_{ic} (SCAMP-5 FPSP to IMU)
$\begin{bmatrix} -0.9991 & 0.0074 & 0.0410 & 0.0038 \\ -0.0065 & -0.9997 & 0.0236 & 0.0639 \\ 0.0412 & 0.0233 & 0.9989 & -0.0234 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.9991 & -0.0065 & 0.0412 & 0.0052 \\ 0.0074 & -0.9997 & 0.0233 & 0.0644 \\ 0.0410 & 0.0236 & 0.9989 & 0.0217 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Time Offset from SCAMP-5 FPSP to IMU (with Camera at 30 FPS): -0.0264 s	
Time Offset from SCAMP-5 FPSP to IMU (with Camera at 300 FPS): -0.0033 s	

Table 3.7: Camera- IMU calibration: transformation, time offset extrinsics between the SCAMP-5 FPSP and IMU. Last 3×1 are p_{ci}, p_{ic}.

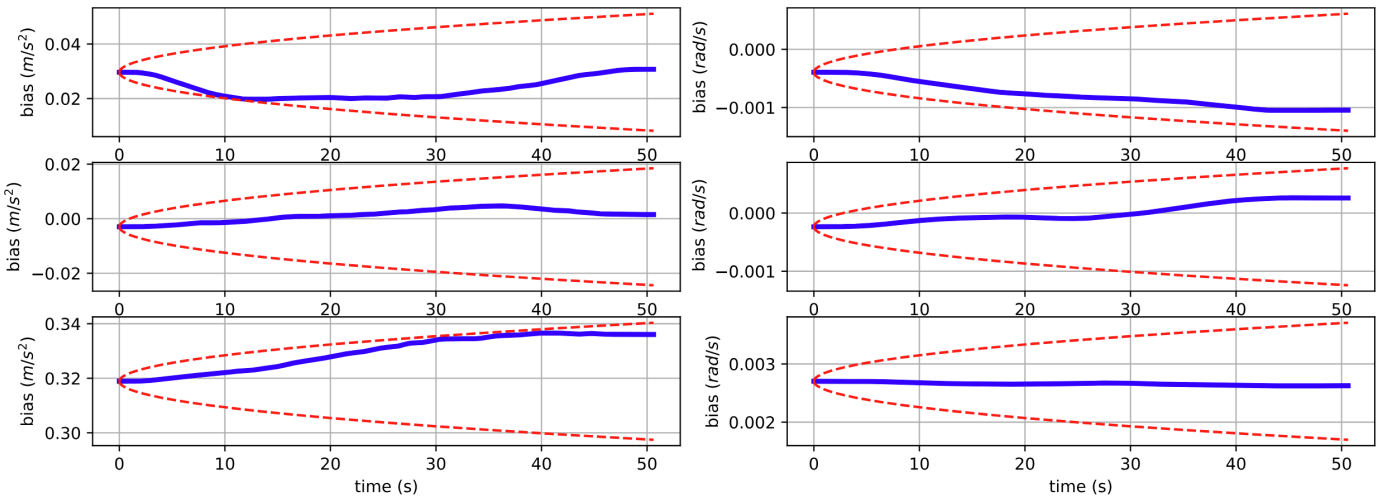


Figure 3.15: Accelerometer (left) bias and gyroscropic (right) bias estimates optimized in camera- IMU calibration.

To verify the accuracy and robustness of the camera- IMU extrinsic calibration, in taking the IMU intrinsic estimates from the IMU-alone calibration process, take them as the first estimate and in allowing the camera- IMU calibration process to optimize further, see that the further estimated and joint-optimized accelerometer and gyroscopic biases reside well within the σ-error bound in dashed red over the trajectory path. As the estimates oscillate, that they reside within the error

margin means that the intrinsics were done correctly so that the camera- IMU calibration estimates remain smoothly in the bound, not jerking outside along the trajectory. Further, see that the the white noise error of the IMU accelerometer and gyroscope after the camera- IMU estimates are within the 3σ-error bound.

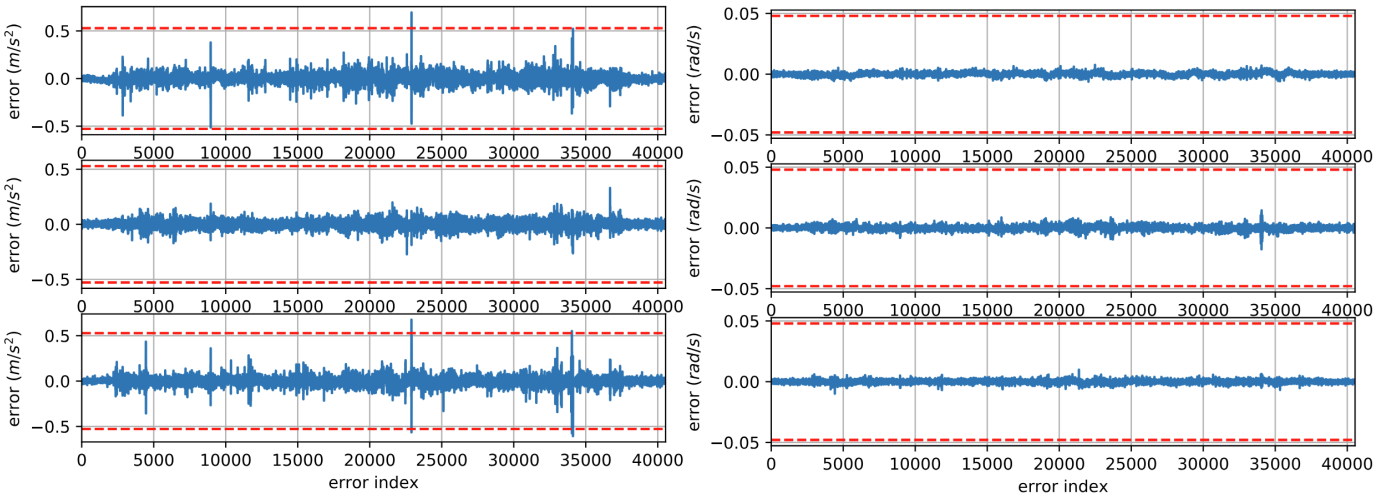


Figure 3.16: Camera- IMU calibration: accelerometer (left) and gyroscopic (right) bias intrinsics error are within the 3σ-error bound.

The camera- IMU calibration now provides error margins on the estimates in the prior calibration steps, having jointly optimized both. The reprojection error maintains less than one-pixel error shown in Table 3.8.

	Normalized Residuals			Residuals		
	Mean	Median	Std	Mean	Median	Std
Reprojection error (SCAMP-5 FPSP)	0.2249 px	0.2045 px	0.1342 px	0.2249 px	0.2045 px	0.1342 px
Gyroscope error (IMU)	0.1405 rad/s	0.1275 rad/s	0.0766 rad/s	0.0022 rad/s	0.0020 rad/s	0.0012 rad/s
Accelerometer error (IMU)	0.3197 m/s^2	0.2716 m/s^2	0.2251 m/s^2	0.0563 m/s^2	0.0478 m/s^2	0.0397 m/s^2

Table 3.8: Camera- IMU calibration: normalized and raw residuals for reprojection error and gyroscopic and accelerometer errors.

Having verified the accuracy of the estimates of the camera- IMU calibration, see further that the intrinsics along with the extrinsics output poses that have high stability in both the SCAMP-5 FPSP camera and IMU frames. This is a trajectory path that excites all IMU axes whilst moving a camera along tracking features from the calibration Aprilgrid target.

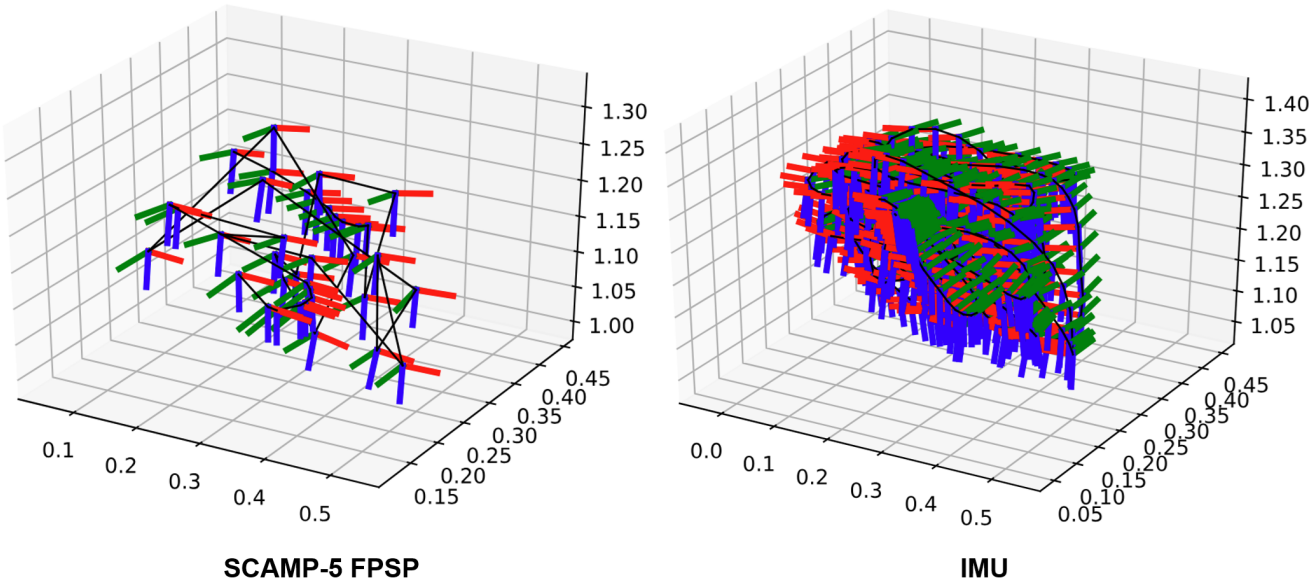


Figure 3.17: Estimated poses by SCAMP-5 FPSP (left) and IMU (right) by camera- IMU calibration algorithm.

3.11 Experimental Results: BIT-VO at 300 FPS, IMU at 400 Hz

The proposed BIT-VIO algorithm is now tested on a moving robotic system where the VO is running fast at the full capable 300 FPS with predictions from 400 Hz IMU measurements. With no bottleneck on BIT-VO high framerate now in swapping out the low frequency outputting UM7 and MPU-9250 IMUs, as well coupled now with proper camera- IMU calibration, what is needed now in this section is to measure the expected improvement in the performance of the robotic system. Evaluations are done again against ground-truth data from the Vicon motion capture system. As stated prior in the initial attempts in working the framework, both BIT-VO and BIT-VIO assume a fast frame rate, hence small inter-frame motion, the method cannot be evaluated using a standard benchmark dataset for direct comparison with other methods. Like before, BIT-VIO is evaluated on eight real-world trajectories against BIT-VO. These trajectories are, again, designed to mimic practical applications and are a compilation of Circular, Straight, Curved, and Zigzag trajectories. Again, like before, the recorded trajectories are aligned and scaled to the ground-truth trajectory as the setup is monocular. BIT-VO and BIT-VIO use a host device to perform the visual odometry backend, and for the host device, an external laptop with 13th Gen Intel Core i7-12700 CPU is used, different from the prior approaches with the other IMU cases, with a more computationally efficient processor needed for the processing power for high frame rate estimation.

3.11.1 Accuracy and Robustness Results

Finally, the performance of BIT-VIO with an IMU at 400 Hz is evaluated. As the translational ATE shows in Table 3.9, it is evident that the accuracy and robustness of the algorithm improved over the prior lower rate IMU cases, in looking over just the translational context.

Translational ATE with IMU at 400 Hz, BIT-VO at 300 FPS			
Traj.	Type	BIT-VO ATE (m)	BIT-VIO ATE (m)
1	RMSE:	0.215732	**0.167631**
	median:	0.170214	**0.152106**
2	RMSE:	0.134617	**0.12071**
	median:	0.119079	**0.111856**
3	RMSE:	0.094479	**0.086911**
	median:	0.07561	**0.068756**
4	RMSE:	0.175323	**0.153335**
	median:	0.174444	**0.140952**
5	RMSE:	0.206866	**0.195263**
	median:	0.15714	**0.149103**
6	RMSE:	**0.134361**	0.134328
	median:	**0.116587**	0.124618
7	RMSE:	0.132624	**0.10535**
	median:	0.125924	**0.095664**
8	RMSE:	0.10864	**0.104366**
	median:	0.089689	**0.08788**

Table 3.9: Translational ATE comparison of BIT-VIO and BIT-VO. The lower ATE is emphasized in bold.

As shown in Table 3.9, when incorporating an IMU, the state generally enhances its estimation with a more accurate trajectory, showcasing lower RMSE and median closer to the ground-truth values. Traj. 1 and 2 are Circular and Curved, Traj. 3 is Straight, and the rest are combinations of all with Zigzag. Focusing specifically on Traj. 1 and 2, the ATE difference between BIT-VO and BIT-VIO is nearly $0.02 - 0.03$ m, compared with the lower ranged difference of the prior lower rate IMU cases that had a difference of only about 0.009 m. This error margin is greater, and is more frequently and consistently closer to ground-truth over the prior cases.

Next, in evaluating the rotational ATE in Table 3.10, the trend is maintained the same as the translational. As RTE is an allocation of the deviation of the estimate from ground-truth, see how

in Traj. 1 how both BIT-VO and BIT-VIO have large accumulated rotational error over the fast, hostile motions done along the trajectory path, yet there is nearly a 52° difference with BIT-VIO rotational error consistently lower than prior BIT-VO. Traj. 5 and 7 are the same, where in the rotational context, tracking is more accurate and robust in the BIT-VIO algorithm. In all Traj. 1-8, the difference is distinguishable even in Traj. 2 and 6 where BIT-VO tracks better with lower rotational RTE than BIT-VIO, and this is the instances where these trajectories have a poorer ATE for BIT-VIO.

Rotational ATE with IMU at 400 Hz, BIT-VO at 300 FPS			
Traj.	Type	BIT-VO ATE (deg)	BIT-VIO ATE (deg)
1	RMSE:	130.88853	**80.37104**
	median:	134.07423	**82.10670**
2	RMSE:	24.24767	**21.49025**
	median:	23.16826	**20.83190**
3	RMSE:	**8.96942**	14.21910
	median:	**8.29516**	13.43212
4	RMSE:	69.37594	**54.37606**
	median:	68.96894	**54.31473**
5	RMSE:	122.33199	**81.82867**
	median:	125.33160	**82.55529**
6	RMSE:	**16.21884**	22.28823
	median:	**15.26092**	21.42191
7	RMSE:	42.62543	**12.80740**
	median:	41.03315	**11.04883**
8	RMSE:	20.58872	**18.12338**
	median:	20.53628	**18.17582**

Table 3.10: Rotational ATE comparison of BIT-VIO and BIT-VO in degrees. The lower ATE is emphasized in bold.

For both the translational and rotational ATE for this higher frequency IMU case with FPSP-vision BIT-VO at its full 300 FPS, more consistently shows the BIT-VIO algorithm as outperforming with less absolute error over the prior method, closer to ground-truth in tracking. The other IMU cases had bottlenecks in performance, as well as greatly limited the full advantages of the FPSP. With proper calibration and better sensor modalities, BIT-VIO is operating as intended.

Now, looking at the local relative trajectory error, and in evaluating the translational RTE shown in Table 3.11, it is clear that the accuracy and robustness of the BIT-VIO algorithm dominates over

prior BIT-VO. Here, taking $\Delta = 0.0001$ m, the relative trajectory error measures the local error between nearly successive points along the trajectory path. As expected, for all Traj. 1-8, the local drift error is larger for BIT-VO and smaller for the corrected framework by BIT-VIO. Specifically for Traj. 7, the larger RTE of BIT-VO over the small BIT-VIO is shown visually in the error mapping and noise variation subsections next.

Translational RTE with IMU at 400 Hz, BIT-VO at 300 FPS with $\Delta = 0.0001$ m			
Traj.	Type	BIT-VO RTE (m)	BIT-VIO RTE (m)
1	RMSE:	0.015884	**0.001138**
	median:	0.009612	**0.000920**
2	RMSE:	0.013175	**0.001607**
	median:	0.007837	**0.000523**
3	RMSE:	0.007075	**0.001443**
	median:	0.004895	**0.000462**
4	RMSE:	0.007320	**0.001058**
	median:	0.005190	**0.000337**
5	RMSE:	0.014288	**0.001264**
	median:	0.008916	**0.000526**
6	RMSE:	0.007876	**0.001275**
	median:	0.005932	**0.000408**
7	RMSE:	0.007931	**0.000800**
	median:	0.005917	**0.000532**
8	RMSE:	0.012996	**0.000932**
	median:	0.007453	**0.000472**

Table 3.11: Translational RTE comparison of BIT-VIO and BIT-VO in degrees. The lower RTE is emphasized in bold.

Next, in evaluating the rotational RTE in Table 3.12, like in the rotational ATE, the trend is maintained the same as the translational. The local rotational error for $\Delta = 0.0001$ m between nearly successive points along the trajectory path shows how vision BIT-VO estimate, as expected, is greater than the corrected estimate, even more so in the rotational context as rotational tracking is robust even soley using IMU measurements alone. The larger RTE of BIT-VO has to do with its higher frequency noise, hence local drift. Traj. 8 noise variation is discussed.

Like with translational and rotational ATE, the RTE shows how BIT-VIO has lower local drift than the FPSP-vision BIT-VO to ground-truth. This is expected, as sensor fusion with the high frequency IMU measurements correct the spatial drift provided by the vision measurements.

This method uses high frequency IMU along with the high frame rate capability of BIT-VO over the other IMU cases. So, both ATE and RTE are shown to be less for the BIT-VIO algorithm.

Rotational RTE with IMU at 400 Hz, BIT-VO at 300 FPS with $\Delta = 0.0001$ m			
Traj.	Type	BIT-VO RTE (deg)	BIT-VIO RTE (deg)
1	RMSE:	0.417609	**0.065281**
	median:	0.267496	**0.021067**
2	RMSE:	0.464772	**0.119347**
	median:	0.313887	**0.024954**
3	RMSE:	0.007075	**0.001443**
	median:	0.004895	**0.000462**
4	RMSE:	0.221387	**0.085660**
	median:	0.155030	**0.015796**
5	RMSE:	0.402258	**0.069330**
	median:	0.265461	**0.016976**
6	RMSE:	0.251571	**0.068593**
	median:	0.202932	**0.014952**
7	RMSE:	0.333434	**0.060890**
	median:	0.263394	**0.017005**
8	RMSE:	0.297526	**0.091247**
	median:	0.195866	**0.014956**

Table 3.12: Rotational RTE comparison of BIT-VIO and BIT-VO in degrees. The lower RTE is emphasized in bold.

3.11.2 Root-Mean-Square Error Results

VIO benchmarking to ground-truth is not only done by evaluation metrics on the ATE and RTE but also by RMSE plots. See that in the case of Traj. 7 in Fig. 3.18, it is shown, as expected, that IMU-alone accumulates error and drifts away from ground-truth data, as shown in the large translational (left three) and rotational RMSE (right three). Comparing the translational with the rotational RMSE, it is evident that the differences in the error between BIT-VO, IMU-alone and BIT-VIO can be small at instances along the trajectory path, and so that is why this thesis emphasizes more on showcasing the translational ATE to study further the performance of BIT-VIO over the other instances. In the rotational RMSE plots, the differences in the error is not as great and it clearly shows how BIT-VIO has lowest error, indicative of being closer to ground-truth than the latter two. The rotational ATE trend is as expected, showcased in the rotational RMSE plots.

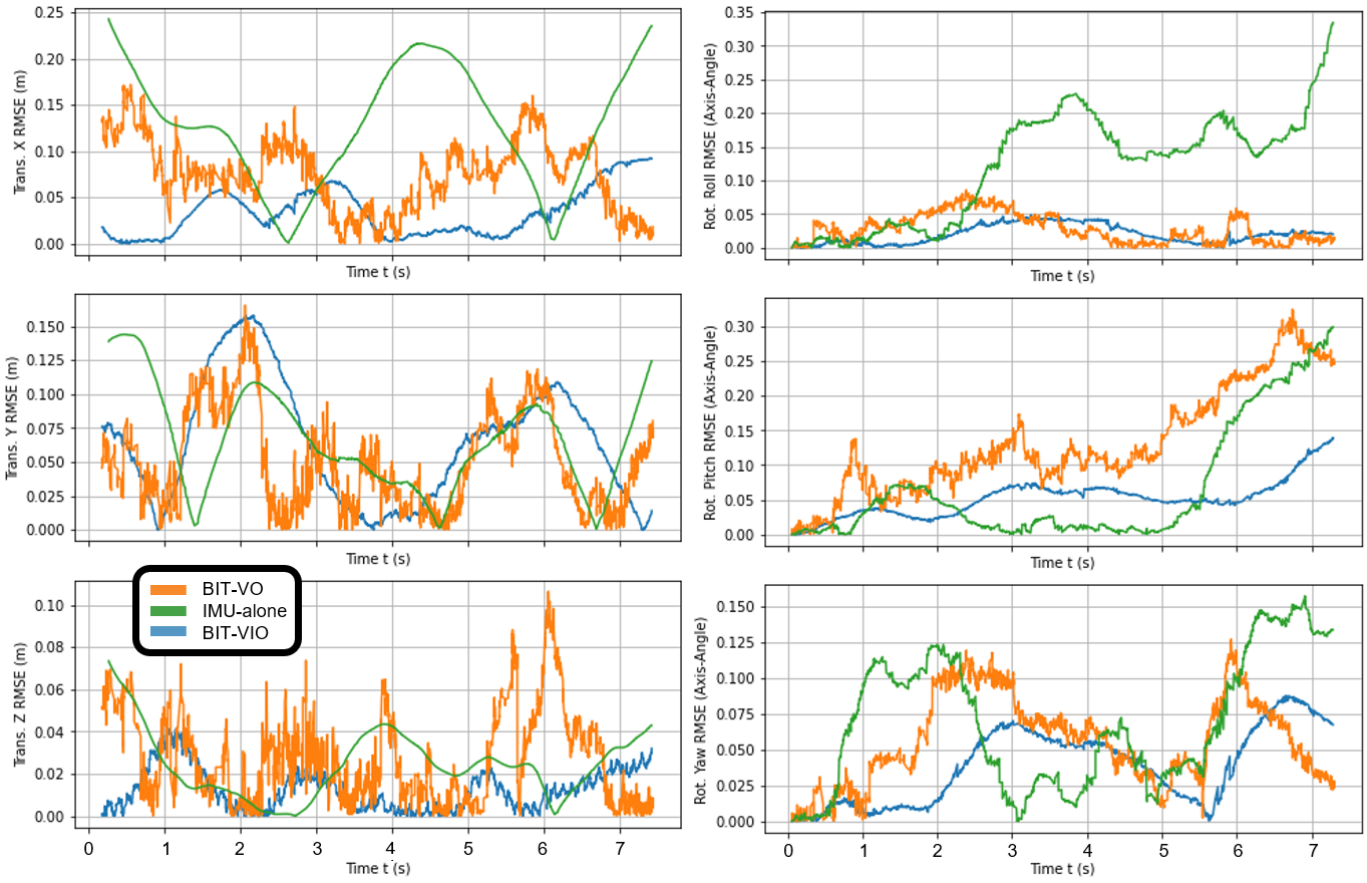


Figure 3.18: Plots of the estimated translational RMSE (left) and rotational RMSE (right) for Traj. 7 from Table 3.11 and Table 3.12. For both translation and rotation, BIT-VIO is much closer and smoother to ground-truth data than IMU-alone and BIT-VO. The drift of the IMU-Alone is very evident, as well as the high-frequency noise of BIT-VO.

These translational and rotational RMSE plots can be further expressed as total RMSE plots, which further showcase the absolute accuracy and robustness of BIT-VIO over BIT-VO and IMU-alone. In doing so, it is clear that BIT-VIO has the largest RMSE compared to BIT-VO and BIT-VIO. The BIT-VIO algorithm fixes this IMU error drift, using the BIT-VO update to align it closer to ground-truth data, hence why its RMSE is the lowest of the three. This is true in both the total translational and rotational context, where it can be seen that in the plots of Fig. 3.18, BIT-VIO RMSE generally resides much more below both IMU and BIT-VO. The total RMSE plots imply that BIT-VIO is generally more accurate and robust over FPSP visual odometry and IMU-alone.

To add, the BIT-VIO algorithm deals well with fast, hostile motions, covering the main limitation of the prior work BIT-VO with the high-frequency noise on its predicted trajectories. BIT-VIO not only maintains itself closer to ground-truth trajectory but also tracks smoothly with less noise, especially in more violent, quick, hostile motions. This is true for all RMSE plots for all Traj. 1-8.

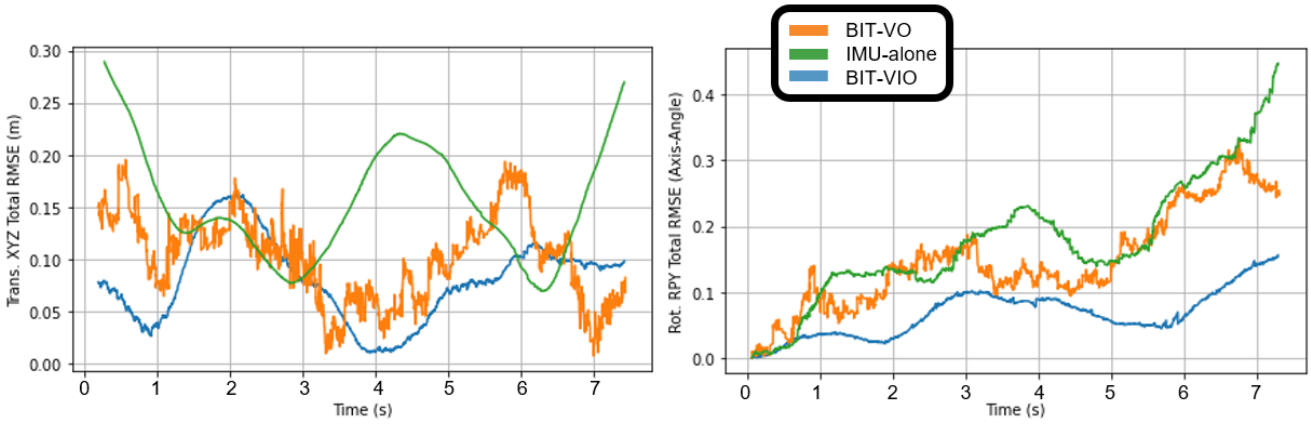


Figure 3.19: Plots of the total estimated translational RMSE (left) and rotational RMSE (right) for Traj. 7. For both the total translation and rotation too, BIT-VIO consistently maintains total less error than prior BIT-VO and IMU-alone, being always maintained less in error along the trajecory path.

3.11.3 Error Mapping and Noise Variation Results

Lastly, a study on the error projected onto the estimate trajectories provided by BIT-VO and BIT-VIO with respect to ground-truth, as well as their associated noise variation in-time is done. See that doing error mapping qualitatively gives further insights into the magnitude of the high-frequency error present in BIT-VO and how much is removed by the BIT-VIO algorithm.

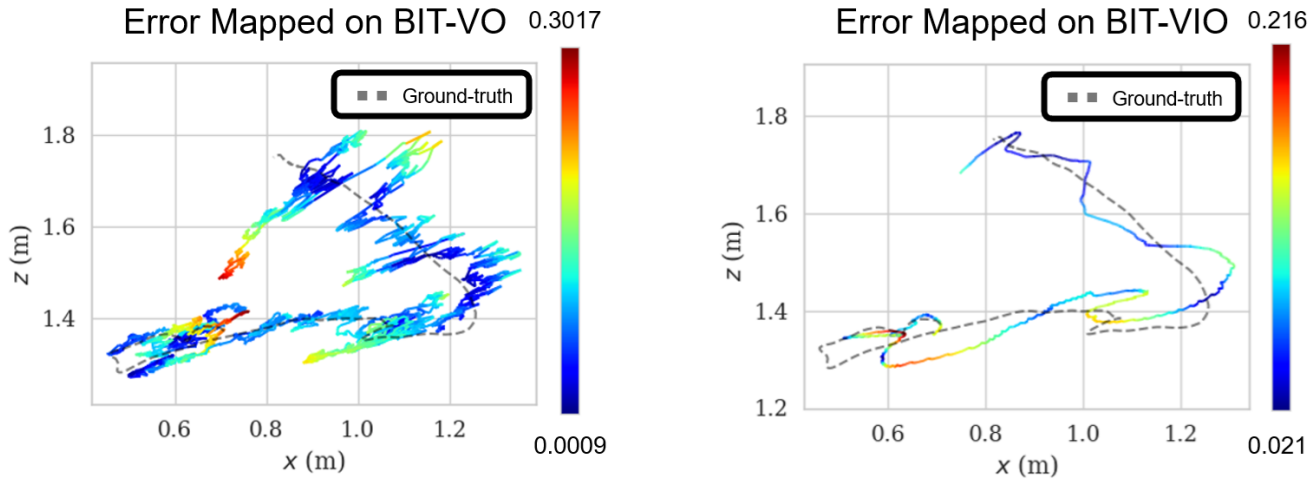


Figure 3.20: Projection of Traj. 8 from Table 3.11 onto xz-plane for BIT-VO (left) and BIT-VIO (right). Observe that BIT-VO suffers from high-frequency noise when compared to BIT-VIO's estimates, although the overall RMSE ATE are similar.

Take, for instance, Traj. 8, projecting the trajectory error first for BIT-VO on ground-truth as shown in Fig. 3.20 (left). The high frequency noise is evident especially in the red tip regions near the ends of the trajectory, peaking at an ATE of 0.3017 m, violently oscillating about ground-truth.

In projecting for BIT-VIO on ground-truth shown in in Fig. 3.20 (right), the high-frequency error is subsided and removed, estimating closer to ground-truth whilst maintaining tracking stability. The ATE peaks only at 0.216 m which is comparably much lower in error than BIT-VO. In the same tail-end region where BIT-VO had that high ATE, for the BIT-VIO algorithm, the ATE error is within the median of a small bound around 0.118 m. Generally, it can be said that in the fast hostile motion of on Traj. 8, see fewer regions of large ATE error, indicated in red, as is shown on prior trajectory of BIT-VO.

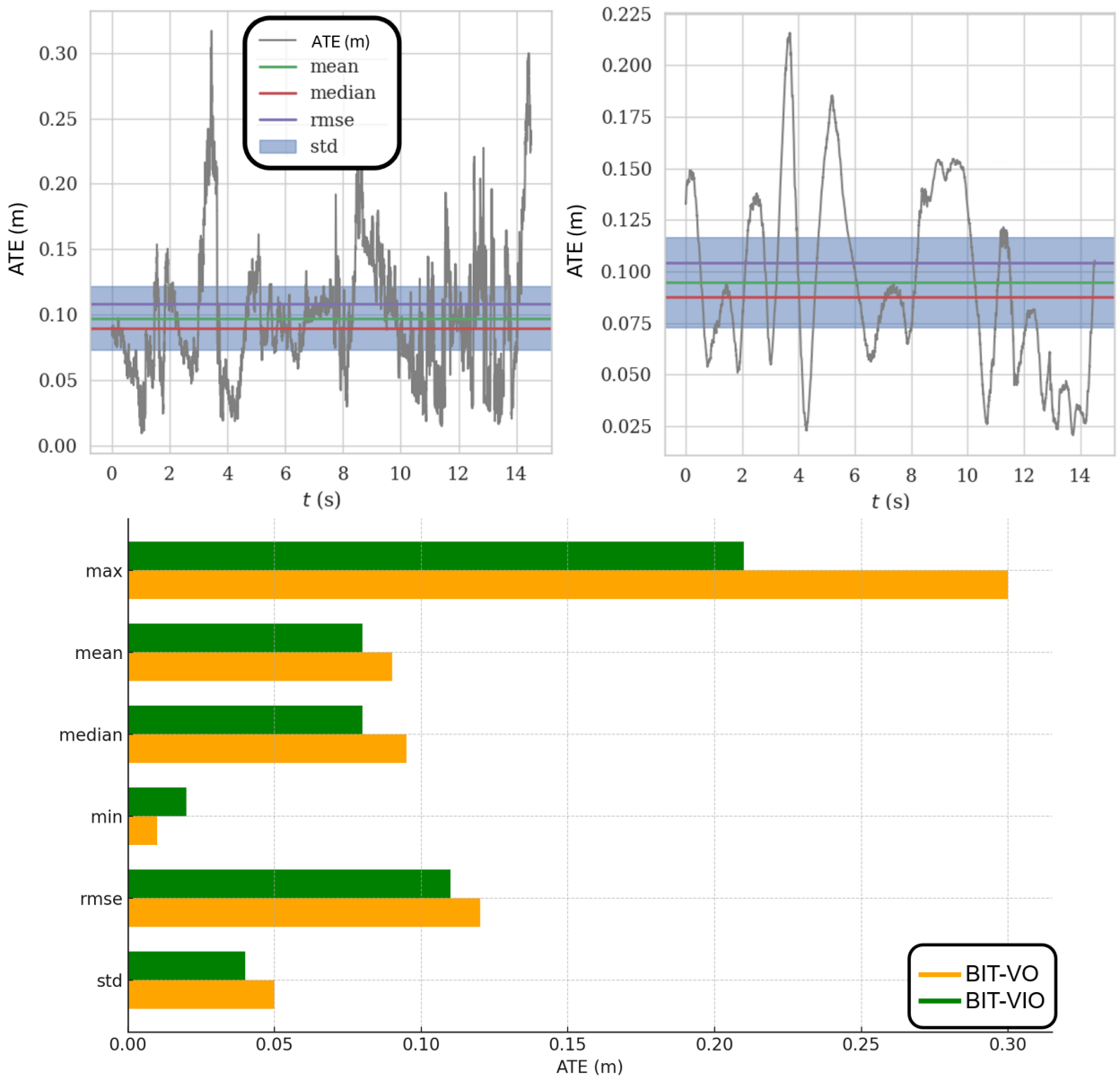


Figure 3.21: ATE allocation and noise variation over time for BIT-VO and BIT-VIO (top left, right). More detailed metrics (bottom).

Showing now the mapped noise variation in time on Traj. 8, the ATE of BIT-VO in Fig. 3.21 (top left) is not stable, protruding with high frequency noise, varying greatly from a median of 0.08968 m. In doing so with the ATE in time of BIT-VIO in Fig. 3.21 (top right), the median error is a lower 0.0878 m, with far greater stability. The noise variation of Traj. 8 between BIT-VO and BIT-VIO shows how the BIT-VIO algorithm error upper-bound is smaller at 0.2157 m than BIT-VO which is a larger value at 0.3170 m. Better shown in Fig. 3.21 (bottom), it is clear of the difference between the two. In other aspects too, namely the mean, RMSE and Standard Deviation (STD), BIT-VIO maintains smaller error over its trajectory when comparing to the BIT-VO estimate.

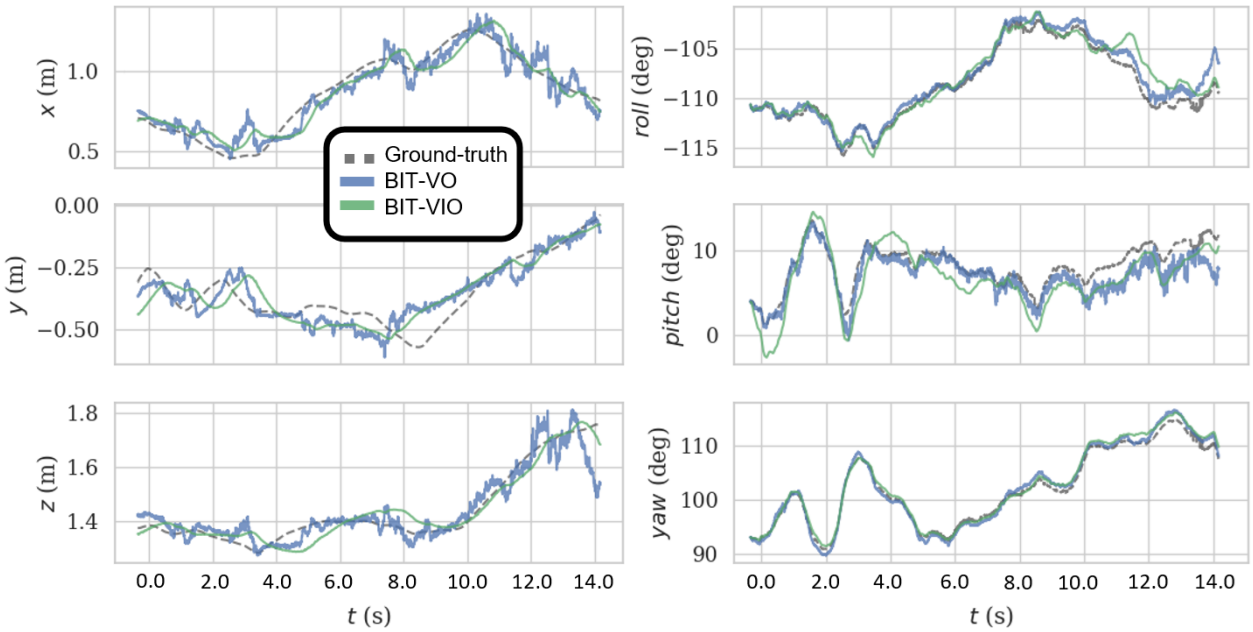


Figure 3.22: Full 6- DoF Tracking Comparison: BIT-VIO and BIT-VO with more noise variation for BIT-VO. This is Traj. 8.

As a final note on the noise variation, what is shown now is Traj. 8 expressed in terms of its full 6- DoF, shown in Fig. 3.22. The rotational noise variation high frequency trend propagated by BIT-VO when compared to BIT-VIO is the same as the translational. Though, emphasizing that with its translational parts, there is noticeably more noise and further shown from ground-truth than its rotational counterpart, hence why the translational part was focused more throughout this thesis, especially in the other IMU cases.

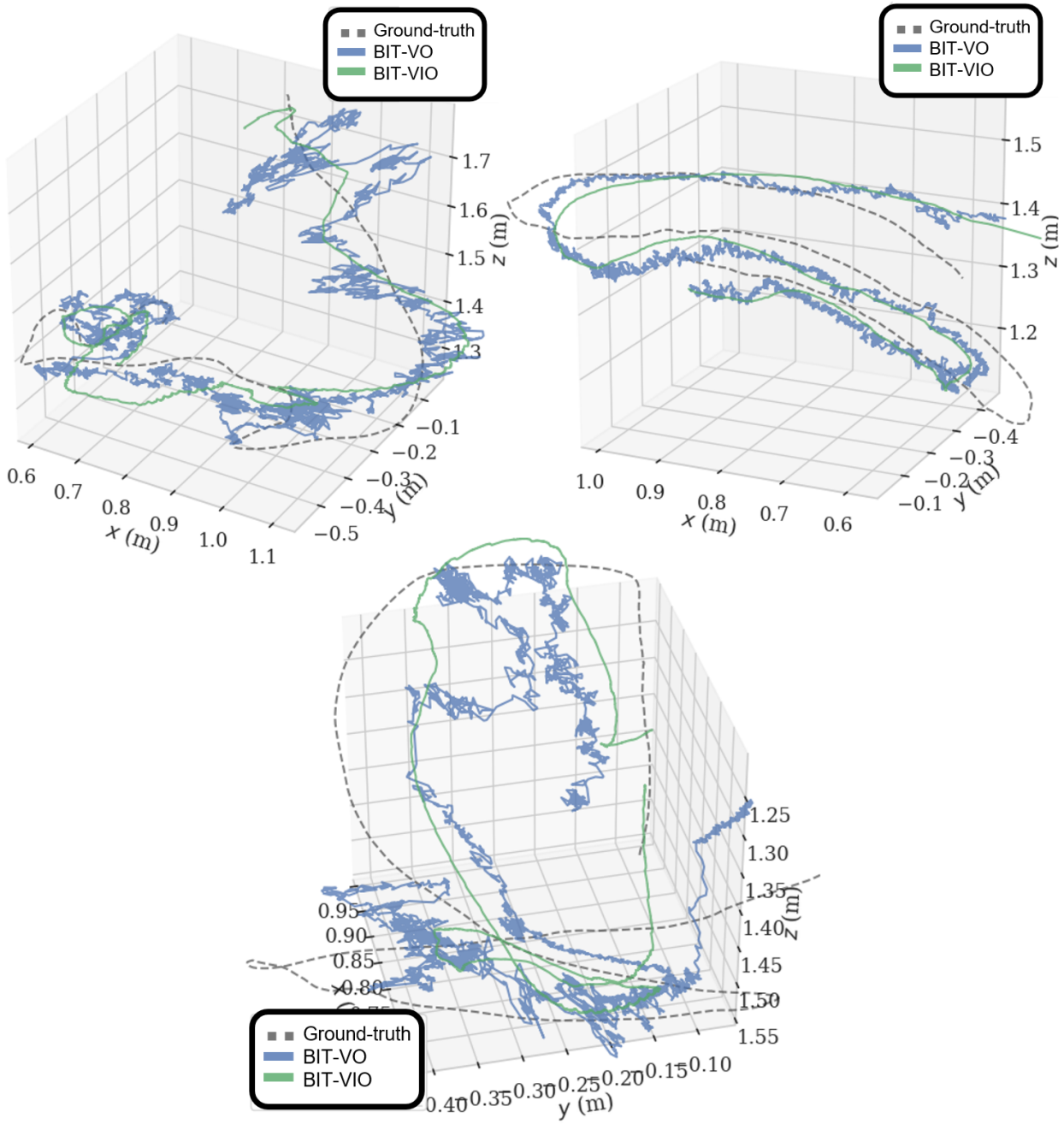


Figure 3.23: BIT-VIO and BIT-VO trajectories in 3D space, where the robustness and accuracy of the estimates is very clear, with BIT-VO having large noise variation compared with BIT-VIO to ground-truth. Traj. 8 (top left) other Traj.

In mapping the trajectories in their full 3D space, the noise variation coming from BIT-VO is even more visibly silenced by BIT-VIO shown in Fig. 3.23. The error mapping and noise variation trend of BIT-VIO having less overall error over BIT-VO generally and in time, is the same for all other trajectories Traj. 1-8. BIT-VIO stabilizes the high frequency noise of prior BIT-VO.

3.12 Visual Front-End Processing Performance on FPSP

Having now presented the experimental results of the full 6- DoF BIT-VIO algorithm at the full frame rate capability, an extensive study on the algorithmic execution timing/frame, accuracy, memory usage and power consumption of the visual front-end processing performance on the FPSP is done. Comparisons are done between the SCAMP-5 FPSP and the Intel D435i RealSense Camera. Note, the IMU is not used here like before and what is focused strictly on is using only the visual front-end processing parts associated with each camera.

Evaluated is executions in real-time of three associated algorithms to the BIT-VIO algorithm – these include Sobel Edge Detection, Laplacian Edge Detection and FAST-16 Corner Feature Extraction. As well, Motion Detection by background subtraction [89], [84], [105].

Note, the BIT-VIO algorithm used primarily the Sobel Edge Detection, but the Laplacian Edge Detection can also be swapped out and utilized. Results below will show that using either or makes little or no discernible difference, so Sobel is used for the sake of simplicity.

3.12.1 Parameterizing on Each Visual Front-End Algorithm

For Sobel and Laplacian edge detection, the kernel size $k \times k$ is used as the parameter metric when measuring the performance of each edge detection algorithm on both the FPSP and Intel RealSense. For FAST-16 Corner Feature Extraction, the gray-level pixel value threshold t (which is used to detect if a corner candidate p is a corner if there exists a set of $n = 10 - 12$ contiguous pixels in the 16-pixel circle which are all brighter than the intensity of the candidate pixel I_p plus a threshold, so $I_p + t$ [89], [84], [105]) is used as the parameter metric when measuring the performance of the FAST algorithm on both the FPSP and Intel RealSense. Further, a study on 3 different profiles based on different readout modes of FAST will be done in measuring the performance of the FPSP. For Motion Detection, gray-level pixel value threshold t (by input of a global value to an analog-register image (each PE has six analog registers capable of storing a real-valued variable)) is used as the parameter metric when measuring the performance of the Motion Detection algorithm on both the FPSP and Intel RealSense. On the Intel RealSense, what was used was the Gaussian Mixture-based MOG2 Background Subtractor [106], [107], and the K-Nearest Neighbors (KNN) Background Subtractor [108].

3.12.2 Execution Timing/Frame, Accuracy, Memory Usage, Power Consumption

As a fair benchmark, these cameras are mounted on each other, seeing essentially the same scene. Both cameras compile the algorithms in C++ code for optimal program execution and runtime (in comparison to other programming languages such as Python). All core code is in C++ with code being transferred to the SCAMP-5 and performing separate to the Lenovo P43s host PC or external computer.

Execution time/frame is measured using the FMT library and the $Timer.h$ function. This is really processing time per frame while running the system. On both the SCAMP-5 FPSP camera and the Intel RealSense take the image frames in a loop, so each iteration the processing time is measured.

Accuracy is measured by ground-truth comparison with AprilTags (of varying sizes but specifically 6×6 as the larger is better for use in detection and tracking in these algorithms). AprilTags are not usually used here, but chose them as they have easily discernible edges (for Sobel and Laplacian) and corner features (for FAST-16 [89], [84], [105]).

Memory usage is measured by the $getrusage()$ function based on maximum resident set size. "Maximum RSS" means the maximum of the RSS since the process's birth, i.e., the largest it has ever been. So, this number tells you the largest amount of physical memory the processing algorithm has ever been using at any one instant.

Power consumption is measured by the KEWEISI USB detector in-out in which connect the SCAMP-5 FPSP and Intel RealSense through before connection to the host PC or external computer (that is Lenovo P43s host or external computer for the SCAMP-5 FPSP camera technology and Intel NUC for the Intel RealSense). Also measured by the $powertop$ and $htop$ libraries (specifically for the Intel RealSense).

3.12.3 Sobel and Laplacian Edge Detection FPSP Performance

Note, $k = 3$ is fixed on SCAMP-5 FPSP as the image-plane is 256×256, constraining the size output of the kernel $k \times k$. $k = 3$ best showcases the nearest-neighbour PE- SIMD architecture of the FPSP and should be used as the basis when comparing with conventional cameras.

For Sobel, the execution time/frame on the SCAMP-5 FPSP goes from 17628.61 μs/frame at 30 FPS to 416.77 μs/frame at 300 FPS. For increasing Sobel kernel sizes $k \times k$ on the Intel RealSense, go from 5826.37 μs/frame at $k = 3$ to 14379.6 μs/frame at $k = 7$ at fixed 30 FPS. As for Laplacian, the execution time/frame on the SCAMP-5 FPSP goes from around the same 17671.83 μs/frame at 30 FPS to 440.125 μs/frame at 300 FPS. For increasing Laplacian kernel sizes $k \times k$ on the Intel RealSense, start smaller going from 3434.63 μs/frame at $k = 3$ to a smaller value of 8138.19 μs/frame at $k = 7$ at fixed 30 FPS. There is a drastic improvement on the SCAMP-5 FPSP when increasing the FPS. On it, Sobel and Laplacian kernel computation is optimizing execution time/frame using the SIMD architecture with only its few set instructions [1]- [18] (for instance, a horizontal Sobel edge detection kernel requires 30 machine-level instructions [1]). It should be noted that after a certain point on the SCAMP-5 FPSP (around the 90 FPS cutoff), the execution time/frame does not improve any more so, though.

		Ext. time/frame	**Memory usage (bytes)**	**Power Consumption /CPU Usage**
SCAMP-5	**30 FPS**	17628.61 μs	51/1024 @ 0x100809ec	0.0256 W
	60 FPS	1027.265 μs	51/1024 @ 0x100809ec	0.0374 W
	90 FPS	399.7965 μs	51/1024 @ 0x100809ec	0.0292 W
	180 FPS	435.25 μs	51/1024 @ 0x100809ec	0.0287 W
	300 FPS	416.7788 μs	51/1024 @ 0x100809ec	0.0293 W
IRS D435i	**k=3**	5826.37 μs	95964 (start with) + 87888 (increase) bytes	210.32 ms/s (powertop), 236.18 % (htop)
	k=5	11231.67 μs	95964 (start with) + 88388 (increase) bytes	457.52 ms/s (powertop), 246.62 % (htop)
	k=7	14379.6 μs	95964 (start with) + 88388 (increase) bytes	194.46 ms/s (powertop), 254.94 % (htop)

Table 3.13: Sobel done on the SCAMP-5 FPSP by varying FPS - showcasing execution time, memory usage and power consumption. Also shown is the Intel RealSense with parameterization by Sobel kernel sizes $k \times k$ ($k = 3$ is fixed on SCAMP-5 FPSP).

The accuracy of ground-truth comparison particularly of the Laplacian on the SCAMP-5 FPSP is affected by the analog operations done on the registers of each PE. The analog registers bring the problems of decaying over time and accumulate errors when copying and adding set instructions

are done (which Laplacian needs). Despite this, the SCAMP-5 FPSP is able to obtain sufficient/satisfactory results. It is much more cleaner and well-defined on the Intel RealSense. Note, the comparison made was assuming the 30 FPS and Sobel and Laplacian kernel size $k = 3$ case.

For Sobel, the memory usage on the SCAMP-5 FPSP is 51 bytes on all FPS. For Laplacian, on it is a lot lower in usage being 29 bytes on all FPS. It is using a ARM Cortex M4 + M0 dual core [18], 204 MHz chip set where Sobel and Laplacian are allocated on the M0. The SCAMP-5 FPSP limits itself in memory allocation based on its design. For Sobel, as for the Intel RealSense, it is with the Intel NUC and Sobel allocates 95964 bytes (at start) with 87888 bytes (for maximal increase). For Laplacian, it allocates with 96128 bytes (at start) with 82212 bytes (for maximal increase). Memory allocation is not a problem with this because it makes use of the entire host PC or external computer.

		Ext. time/frame	**Memory usage (bytes)**	**Power Consumption /CPU Usage**
SCAMP-5	**30 FPS**	17671.83 μs	29/1024 @ 0x100809ec	0.0256 W
	60 FPS	945.4 μs	29/1024 @ 0x100809ec	0.0292 W
	90 FPS	434.8125 μs	29/1024 @ 0x100809ec	0.0287 W
	180 FPS	493.8125 μs	29/1024 @ 0x100809ec	0.0293 W
	300 FPS	440.125 μs	29/1024 @ 0x100809ec	0.0293 W
IRS D435i	**k=3**	3434.63 μs	96128 (start with) + 80336 (increase) bytes	383.76 ms/s (powertop), 224.58 % (htop)
	k=5	4663.37 μs	96128 (start with) + 81128 (increase) bytes	202.64 ms/s (powertop), 228.16 % (htop)
	k=7	8138.19 μs	96128 (start with) + 82212 (increase) bytes	306.40 ms/s (powertop), 240.12 % (htop)

Table 3.14: Laplacian done on the SCAMP-5 FPSP by varying FPS - showcasing execution time, memory usage and power consumption. Also shown is the Intel RealSense with parameterization by Laplacian kernel sizes $k \times k$ ($k = 3$ is fixed on SCAMP-5 FPSP).

For Sobel and Laplacian, the power consumption on the SCAMP-5 FPSP increases as FPS increases (for Sobel, maximal at 0.0374 at 60 FPS). Despite this, comparably this is still low power operation with the advantage of higher FPS. The Intel RealSense the CPU usage performs worse. Note, it does not improve or worsen when varying the Sobel and Laplacian kernel sizes $k \times k$ even.

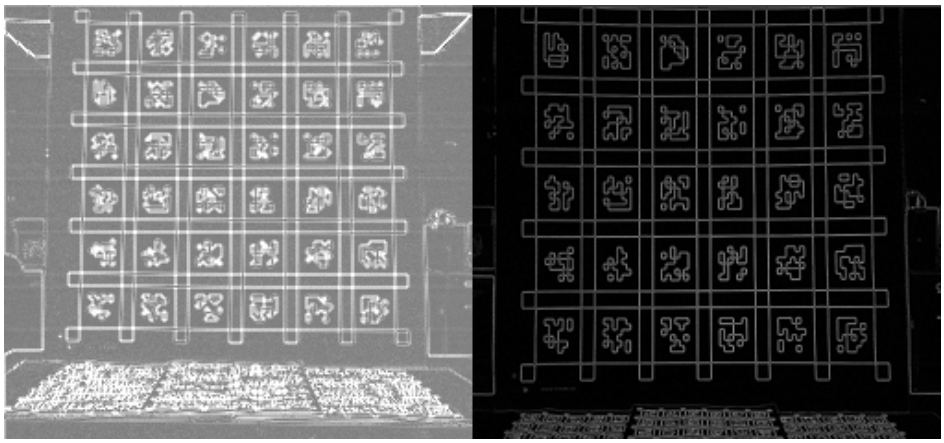


Figure 3.24: Accuracy of ground-truth comparison of Laplacian on the SCAMP-5 FPSP (left) compared with Intel RealSense (right).

3.12.4 FAST-Corner Feature Extraction FPSP Performance

For FAST [89], the execution time/frame on the SCAMP-5 FPSP does not change as much as it is 16791.13 μs/frame at 30 FPS to 16609.8 μs/frame at 300 FPS at fixed 1000 keypoints. There seems to be no effect of the FPS on the execution time. Despite this, on the Intel RealSense, average at a much higher 33634.67 μs/frame but this is at a much higher 17464.94 keypoints. Merely varying the FPS on the SCAMP-5 FPSP is not enough to improve the execution time. It should be noted that for FAST on the SCAMP-5 FPSP, its event-readout infrastructure is used to scan the coordinates of 1s in the 256×256 image [89]. These are in the digital registers. The time cost is generally less because only the coordinates are outputted not the entire image as the Intel RealSense does.

The accuracy of ground-truth comparison is that both the SCAMP-5 FPSP and Intel RealSense are able to keep hold and track the 1000 and 17464.94 keypoints respectively on the AprilTags, but the advantage of the SCAMP-5 FPSP is that in incredibly high FPS there is no motion blur and it is able to still track with high degree.

For FAST [89], the memory usage on the SCAMP-5 FPSP is 282 bytes on all FPS which is significantly larger because each pixel in the 256×256 image is checked to be seen as feature corner and each requires thresholding, saving and counting of 10 pixels in a circle to the candidate corner [89], [84], [105]. Despite this, FAST on the SCAMP-5 FPSP uses the parallel SIMD architecture and each PE does it in parallel [1]- [18]. Different than the others, the feature map of FAST is post-processed meaning that not all the processing is done on the SCAMP-5 FPSP, but most of it just to alleviate on the system.

		Ext. time/frame	Memory usage (bytes)	Power Consumption /CPU Usage	Number of Events /Keypoints
SCAMP-5	**30 FPS**	16791.13 μs	282/1024 @ 0x10080ba0	0.0212 W	1000 keypoints
	60 FPS	16653.5 μs	282/1024 @ 0x10080ba0	0.0297 W	1000 keypoints
	90 FPS	16656.63 μs	282/1024 @ 0x10080ba0	0.0286 W	1000 keypoints
	180 FPS	16668.19 μs	282/1024 @ 0x10080ba0	0.0272 W	1000 keypoints
	300 FPS	16609.88 μs	282/1024 @ 0x10080ba0	0.0287 W	1000 keypoints
	Profile 1	2501.188 μs	355/1024 @ 0x10080ba0	0.0340 W	1000 keypoints
	Profile 2	804.4375 μs	355/1024 @ 0x10080ba0	0.0367 W	1000 keypoints
	Profile 3	2392.875 μs	355/1024 @ 0x10080ba0	0.0251 W	1000 keypoints
IRS D435i		33634.67 μs	96008 (start with) + 77160 (increase) bytes	509.3 ms/s (powertop), 291.62 % (htop)	17464.94 keypoints

Table 3.15: FAST done on the SCAMP-5 FPSP (fixed at 1000 keypoints) by varying FPS - showcasing execution time, memory usage and power consumption. Also shown is the Intel RealSense conventional camera technology. Show SCAMP-5 FPSP profiles $1-3$. (profile 1 is 480 FPS with dithering grayscale image, profile 2 is 1000 FPS with no image transmission and profile 3 is 120 FPS with 1 500th exposure of a low-resolution image).

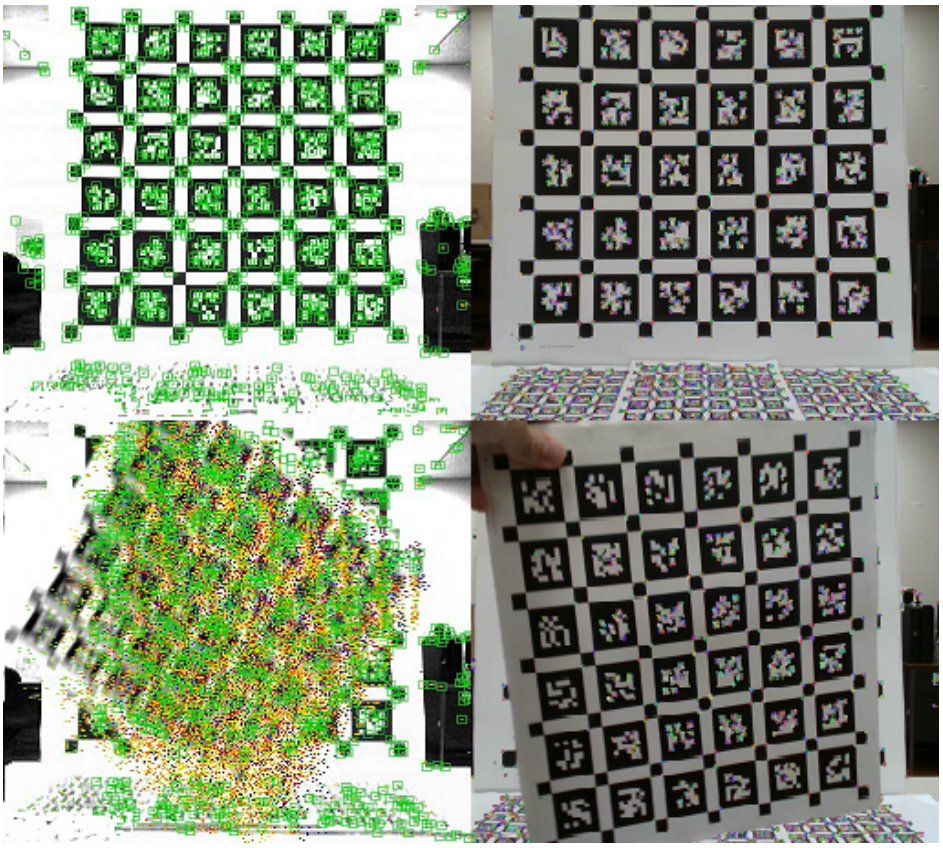


Figure 3.25: Accuracy of ground-truth comparison of FAST on the SCAMP-5 FPSP (left) compared with Intel RealSense (right).

For FAST [89], [84], [105], the memory usage on the Intel RealSense, it is with the Intel NUC and it allocates 96008 bytes (at start) with a smaller 77160 bytes (for maximal increase). This is comparably not much, memory is allocated just enough for 17464.94 keypoints and because it does not vary too much from this.

The power consumption on the SCAMP-5 FPSP maximizes at 0.0297 W at 60 FPS. See that it does so and brings the minimum execution time/frame too compared to all FPS. There seems to be a relation between power consumption and execution time. On the Intel RealSense the CPU usage is a lot 509.3 ms/s (on $powertop$) and 291.62% percentage of CPU usage (on $htop$).

The SCAMP-5 FPSP advantages can be exploited better via the special profiles 1-3. See that in profile 1 (very left) dealing with 480 FPS with dithering grayscale image transmission (meaning the grayscale 256×256 image is processed and made binary). The execution time/frame on the SCAMP-5 FPSP drastically improves going to 2501.188 μs/frame and this is because dithering binarizes the 256×256 image so that thresholding, saving and counting of 10 pixels in a circle to the candidate

corner [89], [84], [105] is done more easily for the system. For profile 2 (center) dealing with 1000 FPS without image transmission at all (meaning that only the coordinates of are outputted, no image at all). The execution time/frame on the SCAMP-5 FPSP is most optimal at 804.4375 μs/frame and this is because the event-readout infrastructure is fully exploited to only output the coordinates of 1s in the 256×256 image. The time cost is optimized, no image transmission is done and take only the information needed to track the corner features. For profile 3 (very right) dealing with 120 FPS with $1/500$th the exposure of a low-resolution image. The execution time/frame on the SCAMP-5 FPSP is maintained and better but is slightly better than profile 1.

The accuracy ground-truth comparison of profiles 1-3 on the SCAMP-5 FPSP is maintained on the AprilTags for all profiles, there is no difference.

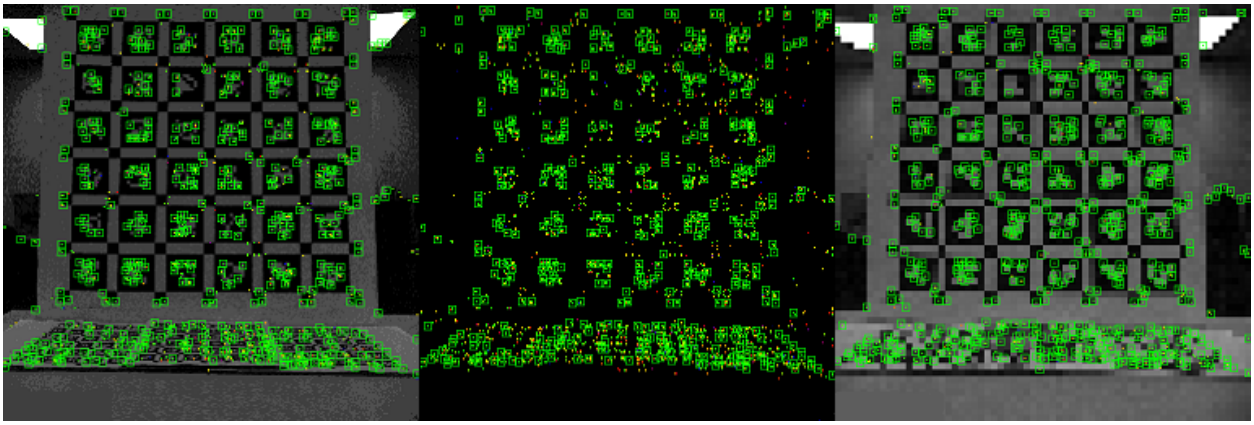


Figure 3.26: Profile 1 is 480 FPS with dithering grayscale image, profile 2 is 1000 FPS with no image transmission and profile 3 is 120 FPS with 1 500th exposure of a low-resolution image. All from SCAMP-5 FPSP.

The memory usage on the SCAMP-5 FPSP is greatest at 355 bytes on all profiles which is significantly largest. The power consumption on the SCAMP-5 FPSP maximizes at 0.0340 W for profile 1 and 0.0367 W for profile 2. For both, this is because for profile 1, dithering the grayscale image and for profile 3, scaling the exposure. For profile 2 not taking in the image transmission requires flagging on the PE registers to not take in which takes memory and more power in order to do.

3.12.5 Motion Detection FPSP Performance

For Motion Detection, the execution time/frame on the SCAMP-5 FPSP goes from 33232.21 μs/frame at 30 FPS to 3258.458 μs/frame at 300 FPS. For KNN-Background Subtraction [108] on the Intel RealSense, average at 12036.7 μs/frame and for MGO2-Background Subtraction [106], [107] get a higher 18725.5 μs/frame at fixed 30 FPS. Again, improvement on the SCAMP-5 FPSP when increasing the FPS. Making full use of the parallel SIMD architecture, simple subtraction of two frames, where the last being stored in each PE is easily done. On the Intel RealSense, chose to use the more sophisticated and efficient KNN [108] and MGO2-Background Subtraction [106], [107] (KNN-Background Subtraction [108] being optimal between the two). Despite this, SCAMP-5 FPSP is still advantageous with its effect of increasing FPS.

The accuracy of ground-truth comparison is that the SCAMP-5 FPSP is able to track motion of a hand and AprilTags with high degree. It does not obstruct or take in the background unlike the on the Intel RealSense, as the AprilTags are moved in the foreground (most apparent in the error of motion detected with MGO2-Background Subtraction [106], [107]).

		Ext. time/frame	**Memory usage (bytes)**	**Power Consumption /CPU Usage**
SCAMP-5	**30 FPS**	33232.21 μs	23/1024 @ 0x100809f4	0.0237 W
	60 FPS	16558.17 μs	23/1024 @ 0x100809f4	0.0245 W
	90 FPS	11006.38 μs	23/1024 @ 0x100809f4	0.0240 W
	180 FPS	5462.042 μs	23/1024 @ 0x100809f4	0.0238 W
	300 FPS	3258.458 μs	23/1024 @ 0x100809f4	0.0255 W
IRS D435i	**KNN**	12036.7 μs	95376 (start with) + 249156 (increase) bytes	148.86 ms/s (powertop), 473.32 % (htop)
	MOG2	18725.5 μs	95876 (start with) + 274452 (increase) bytes	579.16 ms/s (powertop), 680.24 % (htop)

Table 3.16: Motion Detection done on the SCAMP-5 FPSP by varying FPS - showcasing execution time, memory usage and power consumption. Also shown is the Intel RealSense conventional camera technology with parameterization by Gaussian Mixture-based MOG2 Background Subtractor [106], [107], and the KNN Background Subtractor [108]. KNN is an imperative and popular approach in learning and segregating patterns present in sensor data [109].

For Motion Detection, the memory usage on the SCAMP-5 FPSP is 23 bytes on all FPS. For KNN-Background Subtraction [108] on the Intel RealSense, it is with the Intel NUC and it allocates 95376 bytes (at start) with a large 249156 bytes (for maximal increase). With MGO2-Background Subtraction [106], [107], it allocates 95876 bytes (at start) with a much larger 274452 bytes (for maximal increase). It should be noted that the host or external PC allocates more to the program after run for KNN [108] and MGO2-Background Subtraction [106], [107], more than starting. It is required, whereas the SCAMP-5 FPSP maintains 23 bytes despite its limited memory.

The power consumption on the SCAMP-5 FPSP does not change as FPS increases and is maintained. On the Intel RealSense the CPU usage is much more on the host or external computer with MGO2-Background Subtraction [106], [107] being worst at 579.16 ms/s (on $powertop$) and 680.24% percentage of CPU usage (on $htop$).

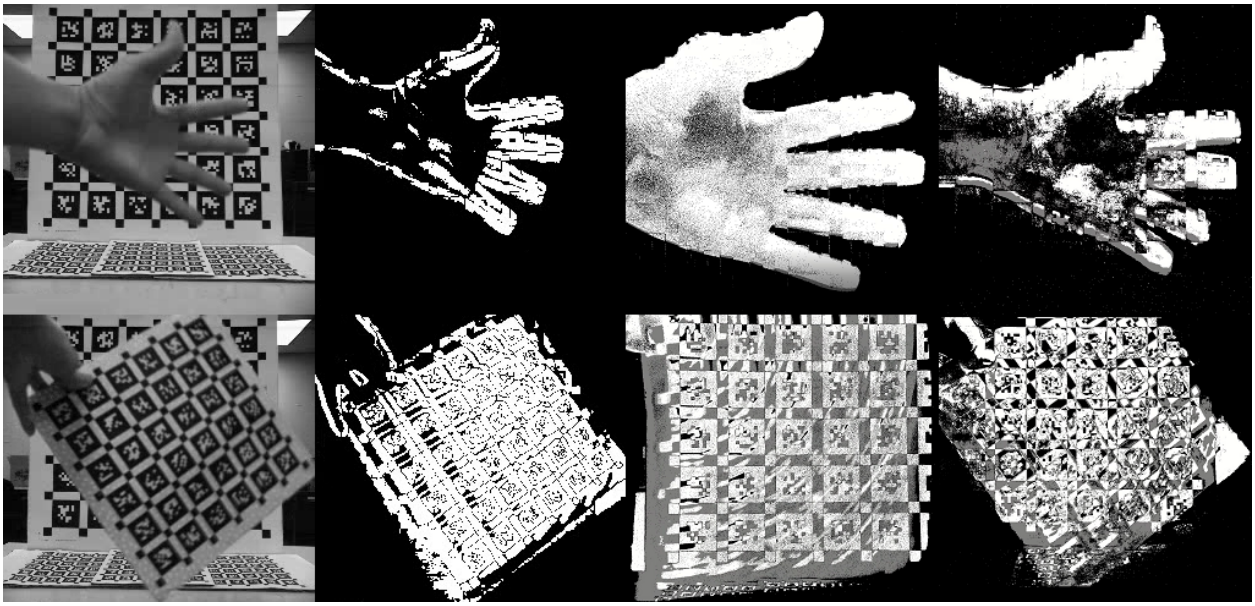


Figure 3.27: Accuracy of ground-truth comparison of Motion Detection on the SCAMP-5 FPSP (first 2×2) compared with Intel RealSense D435i (column to the right of center-line is KNN-Background Subtraction [108] and to the far right is MGO2-Background Subtraction [106], [107]).

Chapter 4

Conclusions and Future Works

In this chapter, the work presented in this thesis is summarized, along with its contributions and significance, as well as its potential avenues for future works. The thesis is summarized in Sec. 4.1, providing an overview of the entire work. Then, significance of the work is restated in Sec. 4.2. The thesis ends in Sec. 4.3 with a detailed discussion of extensions that can be made to this work and its importance to the future.

4.1 Summary and Main Contributions

The high frame-rate nature and selective feature-outputting aspect of SCAMP-5 opens up the possibility for many interesting applications. Most interestingly, is SCAMP-5's use-case in fast, hostile, more complex state estimation algorithms. The reduced power consumption and latency associated with VO and VIO are becoming increasingly important as future mobile devices are anticipated to require rich and accurate spatial understanding capabilities [110]. The problem which SCAMP-5 FPSP addresses is how, currently, conventional camera technology typically operates at 30-60 FPS and transfers a non-trivial amount of data from the sensor to the host device. This thesis has done an extensive literature review on FPSP technology, specifically SCAMP-5, and gone over its applications to broader robotics and mobile systems. It presented a design and implementation of a 6- DoF VIO algorithm which utilizes the advantages of the FPSP for vision- IMU estimation. As an initial attempt, the framework with different IMU cases is explored, attempting with approximated calibration intrinsics and extrinsics and lower IMU frequency rates. Then, a framework with proper visual-inertial calibration process is discussed ending with experimental results, and then a study on the visual front-end processing performance.

The main contributions of this thesis can be summarized as follows:

1. **Efficient VIO operating and correcting by loosely-coupled sensor-fusion iEKF at 300 FPS using predictions from IMU measurements obtained at 400 Hz**: In Sec. 3.3 to Sec. 3.7 a detailed derivation and formulation of the FPSP visual-inertial algorithm is established.

2. **Uncertainty propagation for BIT-VO's pose as it is based on binary-edge-based descriptor extraction, 2D to 3D re-projection**: Sec. 3.5 goes over the camera pose measurement used in the update of the iEKF, while Sec. 3.6 goes over the uncertainty propagated on this 6- DoF pose.

3. **Extensive real-world comparison against BIT-VO, with ground-truth obtained using a motion capture system**: Sec. 3.10 to Sec. 3.11 presented the experimental results, the visual-inertial-sensor calibration done, the overall accuracy and robustness, as well as benchmark comparisons.

4. **Extensive study on the algorithmic execution timing/frame, accuracy, memory usage and power consumption of the visual front-end processing performance on the FPSP:** Sec. 3.12 showcases the on-sensor processing performance of the FPSP, comparing with the processing of algorithms using conventional cameras, ultimately showcasing how FPSPs outperform them.

4.2 Significance

The work presented in this thesis is significant as it has presented a novel design and implementation of the first 6- DoF VIO algorithm which utilized the advantages of the FPSP for vision- IMU estimation. This thesis compared the use-case advantages of FPSPs, bottlenecks of conventional cameras, overall showcasing the superiority over the latter. This thesis also presented an evaluation of the algorithm's performance with state-of-the-art 6- DoF FPSP VO and ground-truth data.

4.3 Future Work

There are many possible extensions for future works to build off the work presented in this thesis.

As a future work, it is recommended next to work towards a tightly-coupled VIO approach with the FPSP. Loop-closure would be better achievable. A tightly-coupled VIO approach would be far more robust in estimation tracking than the loosely-coupled approach. Recommended specifically is the MSCKF, which leverages a EKF framework analogous to BIT-VIO. The use of the high frame rate BIT-descriptor as well as the high frequency input capability of the MSCKF framework would allow better feature integrated tracking.

The front-end tracking performance can be better improved by incorporating scale invariance, along with the rotation invariance of the FPSP-based BIT-descriptor. This is leveraged in incorporating a scale factor between levels in the image scale pyramid, as is done so with ORB-features in ORB-SLAM [20]. In making the BIT-descriptor also scale invariant, this loosely-coupled VIO framework can also be improved by then traversing larger scenes and workspaces where depth awareness would be better improved on the front-end. Note, the methods in this thesis worked with trajectory lengths that were as small as $1-2$ m workspaces, with the full high frequency 6- DoF framework operating in workspaces as large as $7-10$ m. Floorplan or building-wide workspaces would be more challenging, but with the correct setup, it is worth exploring for BIT-VIO.

Another avenue that can be explored on the front-end part of the FPSP- IMU problem, is to displace more of the back-end processing onto the front-end as the feature-extraction was done. The SCAMP-5 FPSP with more digital and analog registers would allow for more memory allocation on parts of the algorithm such as frame-to-frame tracking directly on the sensor-chip, hence alleviating more computational burden off of the external host PC.

With so many potential direction, the importance of this work is in how it has laid the foundation as the first 6- DoF VIO algorithm which utilizes the advantages of the FPSP for vision- IMU estimation.

Appendices

Bibliography

[1] P. Dudek, "SCAMP-3: A Vision Chip with SIMD Current-Mode Analogue Processor Array," *Focal-Plane Sensor-Processor Chips*, pp. 17–43, 2011.

[2] Y. Liu, J. Chen, L. Bose, P. Dudek, and W. Mayol-Cuevas, "Direct servo control from in-sensor CNN inference with a pixel processor array," *arXiv preprint arXiv:2106.07561*, 2021.

[3] H. Castillo-Elizalde, Y. Liu, L. Bose, and W. Mayol-Cuevas, "Weighted node mapping and localisation on a pixel processor array," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6702–6708.

[4] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a UAV with a parallel visual processor," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4248–4254.

[5] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, R. Clarke, J. Chen, S. J. Carey, and P. Dudek, "Towards drone racing with a pixel processor array," in *Proceeding of 11th International Micro Air Vehicle Competition and Conference, IMAV 2019*, 2019, pp. 76–82.

[6] J. Chen, Y. Liu, S. J. Carey, and P. Dudek, "Proximity estimation using vision features computed on sensor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2689–2695.

[7] M. Z. Wong, B. Guillard, R. Murai, S. Saeedi, and P. H. J. Kelly, "AnalogNet: Convolutional neural network inference on analog focal plane sensor processors," *ArXiv*, vol. abs/2006.01765, 2020.

[8] T. Debrunner, S. Saeedi, and P. H. J. Kelly, "AUKE: Automatic Kernel Code Generation for an Analogue SIMD Focal-Plane Sensor-Processor Array," *ACM Transactions on Architecture and Code Optimization*, vol. 15, pp. 1–26, 01 2019.

[9] E. Stow, A. Ahsan, Y. Li, A. Babaei, R. Murai, S. Saeedi, and P. H. J. Kelly, "Compiling CNNs with Cain: Focal-Plane Processing for Robot Navigation," *Autonomous Robots*, vol. 46, no. 8, pp. 893–910, 2022.

[10] E. Stow, R. Murai, S. Saeedi, and P. H. J. Kelly, "Cain: Automatic code generation for simultaneous convolutional kernels on focal-plane sensor-processors," in *International Workshop on Languages and Compilers for Parallel Computing*. Springer, 2020, pp. 181–197.

[11] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Perspective correcting visual odometry for agile MAVs using a pixel processor array," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 987–994.

[12] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "Visual odometry for pixel processor arrays," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017, pp. 4614–4622.

[13] T. Debrunner, S. Saeedi, L. Bose, A. J. Davison, and P. H. J. Kelly, "Camera tracking on focal-plane sensor-processor arrays," in *High Performance and Embedded Architecture and Compilation (HiPEAC), Workshop on Programmability and Architectures for Heterogeneous Multicores (MULTIPROG)*, 2019.

[14] A. McConville, L. Bose, R. Clarke, W. Mayol-Cuevas, J. Chen, C. Greatwood, S. Carey, P. Dudek, and T. Richardson, "Visual odometry using pixel processor arrays for unmanned aerial systems in GPS denied environments," *Frontiers in Robotics and AI*, vol. 7, p. 126, 2020.

[15] H. M. So, L. Bose, P. Dudek, and G. Wetzstein, "Pixelrnn: In-pixel recurrent neural networks for end-to-end-optimized perception with neural sensors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 25 233–25 244.

[16] I. Alzugaray, R. Murai, and A. Davison, "PixRO: Pixel-distributed rotational odometry with gaussian belief propagation," *arXiv preprint arXiv:2406.09726*, 2024.

[17] R. Murai, S. Saeedi, and P. H. Kelly, "BIT-VO: Visual odometry at 300 FPS using binary features from the focal plane," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8579–8586.

[18] J. Chen, S. J. Carey, and P. Dudek, "Scamp5d vision system and development framework," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*, 2018, pp. 1–2.

[19] P. Dudek, "SCAMP-5: Vision sensor with pixel parallel SIMD processor array," in *CVPR 2019 Workshop on Event-Based Vision and Smart Cameras*. Long Beach, CA, USA: IEEE, June 2019, school of Electronic & Electrical Engineering, The University of Manchester.

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[21] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[22] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[23] M. Li, *Visual-Inertial Odometry on Resource-Constrained Systems*. University of California, Riverside, 2014.

[24] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[25] A. Alhashimi, G. Nikolakopoulos, and T. Gustafsson, "Observation model for monte carlo localization," in *Reglermöte 2014: 03/06/2014-04/06/2014*, 2014.

[26] W. Burgard, A. Derr, D. Fox, and A. B. Cremers, "Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, vol. 2. IEEE, 1998, pp. 730–735.

[27] X. Sun, F. Sun, B. Wang, J. Yin, X. Sheng, and Q. Xiao, "Robotic autonomous exploration SLAM using combination of kinect and laser scanner," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2017, pp. 632–637.

[28] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[29] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[30] R. Jamiruddin, A. O. Sari, J. Shabbir, and T. Anwer, "RGB-Depth SLAM Review," *arXiv preprint arXiv:1805.07696*, 2018.

[31] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[32] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1935–1942.

[33] B. Al-Tawil, T. Hempel, A. Abdelrahman, and A. Al-Hamadi, "A review of visual SLAM for robotics: Evolution, properties, and future applications," *Frontiers in Robotics and AI*, vol. 11, p. 1347985, 2024.

[34] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2007, pp. 225–234.

[35] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "Visual SLAM: What are the current trends and what to expect?" *Sensors*, vol. 22, no. 23, p. 9297, 2022.

[36] R. Mur-Artal and J. D. Tardós, "Orb-slam: Tracking and mapping recognizable," in *Proceedings of the Workshop on Multi View Geometry in Robotics (MVIGRO)-RSS*, 2014.

[37] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, "Benchmark of visual SLAM algorithms: ORB-SLAM2 vs RTAB-Map," in *2019 Eighth International Conference on Emerging Security Technologies (EST)*. IEEE, 2019, pp. 1–6.

[38] E. Fernández-Moral, J. G. Jiménez, and V. Arévalo, "Creating metric-topological maps for large-scale monocular SLAM," in *ICINCO (2)*, 2013, pp. 39–47.

[39] B. W. Babu, S. Kim, Z. Yan, and L. Ren, "σ-DVO: Sensor noise model meets dense visual odometry," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2016, pp. 18–26.

[40] S. Song, J. Chen, Y. Zhong, W. Zhang, W. Hou, and L. Zhang, "SCE-SLAM: A real-time semantic RGBD SLAM system in dynamic scenes based on spatial coordinate error," *Measurement Science and Technology*, vol. 34, no. 12, p. 125006, 2023.

[41] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[42] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[43] M. Ben-Ari, F. Mondada, M. Ben-Ari, and F. Mondada, "Robotic motion and odometry," *Elements of Robotics*, pp. 63–93, 2018.

[44] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[45] R. Maenle, "Pose estimation using a stereo-photometric multi-state constraint kalman filter," Master's Thesis, University of Applied Sciences Technikum Wien, Vienna, September 2019, degree Program Mechatronics/Robotics.

[46] B. M. Bell and F. W. Cathey, "The iterated kalman filter update as a gauss-newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[47] A. E. Bondoc, M. Tayefeh, and A. Barari, "Employing live digital twin in prognostic and health management: Identifying location of the sensors," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 138–143, 2022.

[48] ——, "Live digital twin: Developing a sensor network to monitor the health of belt conveyor system," *IFAC-PapersOnLine*, vol. 55, no. 19, pp. 49–54, 2022.

[49] A. E. Bondoc, M. Lopez, I. Shilbayeh, A. Abdulkadir, M. Tayefeh, M. Hosseini, and A. Barari, "Implementation of live digital twin enabled smart maintenance using smart structural sensors," in *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*. IEEE, 2023, pp. 1205–1212.

[50] P. Shrivastava, T. K. Soon, M. Y. I. B. Idris, and S. Mekhilef, "Overview of model-based online state-of-charge estimation using kalman filter family for lithium-ion batteries," *Renewable and Sustainable Energy Reviews*, vol. 113, p. 109233, 2019.

[51] D. Scaramuzza, Z. Zhang, M. H. Ang, O. Khatib, and B. Siciliano, "Aerial robots, visual-inertial odometry of," 2020.

[52] D. Scaramuzza, "Lecture 13: Visual inertial fusion," Lecture, Robotics & Perception Group, University of Zurich and ETH Zurich, Institute of Informatics – Institute of Neuroinformatics, Zurich, Switzerland, 2019, http://rpg.ifi.uzh.ch/.

[53] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[54] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 298–304.

[55] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[56] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[57] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[58] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[59] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," pp. 519–535, 2007.

[60] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[61] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[62] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3923–3929.

[63] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 15–22.

[64] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.

[65] A. Martinelli *et al.*, "Observability properties and deterministic algorithms in visual-inertial structure from motion," *Foundations and Trends® in Robotics*, vol. 3, no. 3, pp. 139–209, 2013.

[66] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis, "On the consistency of vision-aided inertial navigation," in *Experimental Robotics: The 13th International Symposium on Experimental Robotics*. Springer, 2013, pp. 303–317.

[67] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A first-estimates jacobian EKF for improving SLAM consistency," in *Experimental Robotics: The Eleventh International Symposium*. Springer, 2009, pp. 373–382.

[68] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," *Georgia Institute of Technology, Tech. Rep*, vol. 2, p. 4, 2012.

[69] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Camera-IMU-based localization: Observability analysis and consistency improvement," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 182–201, 2014.

[70] T.-C. Dong-Si and A. I. Mourikis, "Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5655–5662.

[71] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "An observability-constrained sliding window filter for SLAM," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 65–72.

[72] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.

[73] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[74] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI*, 2015.

[75] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1689–1696.

[76] A. I. Mourikis and S. I. Roumeliotis, "A dual-layer estimator architecture for long-term localization," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008, pp. 1–8.

[77] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[78] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2011.

[79] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[80] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5391–5399.

[81] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," 2017.

[82] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[83] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1425–1440, 2018.

[84] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 430–443.

[85] S. M. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, ETH Zurich, 2012.

[86] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4531–4537.

[87] N. Trawny and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[88] R. Murai, "Visual odometry using a focal-plane sensor-processor," MEng Individual Project, Imperial College London, London, UK, June 2019, supervisors: Prof. Paul Kelly, Dr. Sajad Saeedi; Second Marker: Prof. Andrew Davison.

[89] J. Chen, S. J. Carey, and P. Dudek, "Feature extraction using a portable vision system," in *IEEE/RSJ Int. Conf. Intell. Robots Syst., Workshop Vis.-Based Agile Auton. Navigation UAVs*, vol. 2, 2017, p. 3.

[90] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[91] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 2023, https://github.com/ceres-solver/ceres-solver.

[92] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997.

[93] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

[94] R. Murai, S. Saeedi, and P. H. J. Kelly, "High-frame rate homography and visual odometry by tracking binary features from the focal plane," *Autonomous Robots*, Jul 2023. [Online]. Available: https://doi.org/10.1007/s10514-023-10122-8

[95] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, 2012, pp. 2088–2095.

[96] L. Zhang, Z. Liu, and C. H. Xia, "Clock synchronization algorithms for network measurements," in *Proceedings of the IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[97] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep., 2007.

[98] Ori-drs, "Allan variance ros," 2021, bSD-3-Clause license. [Online]. Available: https://github.com/ori-drs/allan_variance_ros

[99] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

[100] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3400–3407.

[101] M. Kaess, "Apriltags," http://people.csail.mit.edu/kaess/apriltags/, Nov. 2013, accessed: 2024-06-25.

[102] J. Kannala and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[103] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *Proc. of the IEEE Intelligent Vehicles Symposium (IVS)*, 2013.

[104] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[105] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, 2008.

[106] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, pp. 135–144, 2002.

[107] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2. IEEE, 2004, pp. 28–31.

[108] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[109] A. E. Bondoc, M. Tayefeh, and A. Barari, "Learning phase in a live digital twin for predictive maintenance," *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 13, 2022.

[110] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, C. Gorgovan, A. Webb, J. Clarkson, E. Tomusk, T. Debrunner, K. Kaszyk, P. Gonzalez-De-Aledo, A. Rodchenko, G. Riley, C. Kotselidis, B. Franke, M. F. O'Boyle, A. J. Davison, P. H. J. Kelly, M. Luján, and S. Furber, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proceedings of the IEEE*, vol. 106, no. 11, pp. 2020–2039, 2018.

Acronyms

2D 2-Dimensional. 5, 14, 79, *Glossary:* 2D

3D 3-Dimensional. 5, 14, 24, 26, 36–38, 67, 79, *Glossary:* 3D

ATE Absolute Trajectory Error. 40, 41, 44–49, 59–62, 64–66, *Glossary:* ATE

BIT-VIO Binary Feature Visual Inertial Odometry. iii, 4, 10, 29–32, 34, 41–50, 58–68, 80, *Glossary:* BIT-VIO

BIT-VO Binary Feature Visual Odometry. 3–6, 10, 11, 30–32, 34–38, 41–49, 58–67, 79, *Glossary:* BIT-VO

BRIEF Binary Robust Independent Elementary Features. 13, *Glossary:* BRIEF

CAIN Automatic Code Generation for Simultaneous Convolutional Kernels on FPSPs. 10, *Glossary:* CAIN

CNN Convolutional Neural Network. 1, 2, 9, 10, *Glossary:* CNN

DoF Degrees of Freedom. 3–6, 8, 10, 14, 38, 39, 66, 68, 78–80, *Glossary:* DoF

DRAM Dynamic Random-Access Memory by Digital Registers. 7, 8, *Glossary:* DRAM

DVO-SLAM Dense Visual Odometry and SLAM. 13, *Glossary:* DVO-SLAM

EKF Extended Kalman Filter. 16, 17, 19, 20, 27, 80, *Glossary:* EKF

FAST Features from Accelerated Segment Test. 13, 27, 34, 68, 69, 72–74, *Glossary:* FAST

FPS Frames-Per-Second. iii, 1–6, 9–11, 30, 31, 38, 42–49, 52, 56, 58, 60, 70–79, *Glossary:* FPS

FPSP Focal-Plane Sensor-Processor. iii, 1–12, 18, 27–36, 42, 45, 47–49, 52–58, 60, 61, 63, 68–80, *Glossary:* FPSP

GTSAM Georgia Tech Smoothing and Mapping. 19, 22, 24, 26, *Glossary:* GTSAM

HDR High Dynamic Range. 2, 3, 8, 9, *Glossary:* High Dynamic Range

Hz Hertz. iii, 5, 30, 31, 42–50, 52, 56, 58, 59, 79, *Glossary:* Hz

iEKF iterated Extended Kalman Filter. iii, 5, 17, 30, 42, 44, 47, 79, *Glossary:* iEKF

IMU Inertial Measurement Unit. iii, 4–6, 11, 14, 15, 18–20, 24–27, 29–31, 33, 34, 42–52, 55–64, 66, 68, 78–80, *Glossary:* IMU

iSAM Incremental Smoothing and Mapping. 25, *Glossary:* iSAM

iSAM2 Incremental Smoothing and Mapping 2. 19, 25, 26, *Glossary:* iSAM2

KF Kalman Filter. 15–18, 20, 22, *Glossary:* KF

KNN K-Nearest Neighbors. 68, 76, 77, *Glossary:* KNN

LS Least Squares. 22, *Glossary:* LS

LSD-SLAM Large-Scale Direct Monocular SLAM. 13, *Glossary:* LSD-SLAM

MSCKF Multi-State Constraint Kalman Filter. 21, 29, 80, *Glossary:* MSCKF

MSF-EKF Multi-Sensor Fusion Extended Kalman Filter. 21, 22, 29, 30, 33, *Glossary:* MSF-EKF

OKVIS Open Keyframe-based Visual-Inertial SLAM. 23, 24, *Glossary:* OKVIS

ORB Oriented Rotated BRIEF. 11, 13, 80, *Glossary:* ORB

ORB-SLAM ORB-Simultaneous Localization and Mapping. 11, 13, 14, 80, *Glossary:* ORB-SLAM

ORB-SLAM2 ORB-Simultaneous Localization and Mapping Version 2.0. 13, 14, *Glossary:* ORB-SLAM2

ORB-SLAM3 ORB-Simultaneous Localization and Mapping Version 3.0. 14, 26, *Glossary:* ORB-SLAM3

PBA Photo Metric Bundle Adjustment. 13, *Glossary:* PBA

PC Personal Computer. 1, 2, 6, 8, 32, 34, 35, 38, 43, 49, 69, 71, 77, 80, *Glossary:* PC

PE Processing Element. 7, 8, 68–70, 72, 75, 76, *Glossary:* PE

PF Particle Filter. 15–18, 22, *Glossary:* PF

PTAM Parallel Tracking and Mapping. 13, 25, 37, *Glossary:* PTAM

RANSAC Random Sample Consensus. 11, 36, *Glossary:* RANSAC

RGB-D Red-Green-Blue-Depth. 13, *Glossary:* RGB-D

RGB-D SLAM Red-Green-Blue-Depth SLAM. 12–14, *Glossary:* RGB-D SLAM

RMSE Root-Mean-Square Error. 30, 40, 41, 44, 46, 49, 59, 62–64, 66, *Glossary:* RMSE

RNN Recurrent Neural Network. 2, 3, 9, 10, *Glossary:* RNN

ROVIO Robust Visual Inertial Odometry. 21, 29, *Glossary:* ROVIO

RPY Roll, Pitch and Yaw. 40, 41, *Glossary:* RPY

RTAB-Map Real-Time Appearance-Based Mapping. 14, *Glossary:* RTAB-Map

RTE Relative Trajectory Error. 40, 41, 45, 48, 59–62, *Glossary:* RTE

SCAMP-5 SIMD Current-Mode Analog Matrix Processor Version 5.0. iii, 1–4, 6–11, 29–36, 42, 49, 52–58, 68–78, 80, *Glossary:* SCAMP-5

SCE-SLAM Spatial Coordinate Error SLAM. 14, *Glossary:* SCE-SLAM

SIMD Single Instruction Multiple Data. iii, 3, 7, 10, 69, 70, 72, 76, *Glossary:* SIMD

SLAM Simultaneous Localization and Mapping. 12–15, 23, 27, 40, 43, *Glossary:* SLAM

STD Standard Deviation. 66, *Glossary:* STD

SVO Fast Semi-Direct Monocular Visual Odometry. 22, 24, 26, 37, *Glossary:* SVO

SVO+GTSAM Fast Semi-Direct Monocular Visual Odometry + GTSAM. 26, *Glossary:* SVO+GTSAM

SVO+MSF Fast Semi-Direct Monocular Visual Odometry + Multi-Sensor Fusion Extended Kalman Filter. 22, 26, 29, *Glossary:* SVO+MSF

UAV Unmanned Aerial Vehicle. 2, 3, 9, 12, *Glossary:* UAV

UKF Unscented Kalman Filter. 17, 20, *Glossary:* UKF

VINS-Mono Monocular Visual-Inertial State Estimator. 24, 25, *Glossary:* VINS-Mono

VIO Visual Inertial Odometry. iii, 4–6, 11, 15, 17–27, 29, 30, 40, 44, 47, 62, 78–80, *Glossary:* VIO

VO Visual Odometry. 2–5, 10, 18, 27, 30, 31, 36, 44, 45, 47–49, 58, 78, 79, *Glossary:* VO

Glossary

2D 2-Dimensional: Confined to two-dimensional plane-space. 5, 104

3D 3-Dimensional: Confined to full three-dimensional space. 5, 104

ATE Absolute Trajectory Error: A measure of trajectory error over the entire path. 40, 104

BIT-VIO Binary Feature Visual Inertial Odometry: proposed algorithm in this thesis operating at 300 FPS with IMU measurements at 400 Hz. iii, 104

BIT-VO Binary Feature Visual Odometry: First-ever 6-DoF visual odometry algorithm whose front-end feature extractor is on the FPSP. 3, 104

BRIEF Binary Robust Independent Elementary Features: Feature descriptor framework that is used for distinguishing uniquely different features across many frames. BRIEF is highly efficient using simple intensity tests. 13, 104

CAIN Automatic Code Generation for Simultaneous Convolutional Kernels on Focal-plane Sensor-processors: compiler for convolutional filters that can be compressed onto the FPSP. 10, 104

DoF Degrees of Freedom: In statics this refers to the three translational motions and three rotational motions (i.e. angular velocities) of a robot system body. 3, 104

DRAM Dynamic Random-Access Memory by Digital Registers: Volatile memory that is stores digital data (of 1 or 0) in transistors. 7, 104

DVO-SLAM Dense Visual Odometry and SLAM: SLAM framework that is based on minimizing both the photometric and depth error in all pixels in the camera-plane. This framework presents an entropy-based keyframe selection rule, and has loop-closure. 13, 104

EKF Extended Kalman Filter: algorithm that loosely-couples a pose from prediction and a pose from update by Kalman filtering means, able to work with nonlinear systems. Note: state is computed as an approximate conditional mean. 16, 104

FAST Features from Accelerated Segment Test: Corner detection method that determines whether a pixel on image-plane is corner if the pixels within a fixed ring of prospect corner are above a threshold intensity value. 13, 104

FPS Frames-Per-Second: A measure of the frequency of frames an algorithm can process at. iii, 104

FPSP Focal-Plane Sensor-Processor: Next-generation vision-sensor with low latency and low power usage capabilities. iii

GTSAM Georgia Tech Smoothing and Mapping: library or framework that uses factor-graph optimization to address smoothing and mapping (SAM) problems. 19, 104

High Dynamic Range High Dynamic Range: Refers to the Computer Vision processing method of creating a highly-contrasted image or frame. 2, 104

Hz Hertz: Oscillations-Per-Second. iii, 104

iEKF iterated Extended Kalman Filter: algorithm that loosely-couples a pose from prediction and a pose from update by Kalman filtering means, able to work with nonlinear systems. Note: state is computed by "Maximum a Posteriori" (MAP) estimate. iii, 1, 104

IMU Inertial Measurement Unit: body sensor that tracks the angular velocities and linear accelerations of a system in motion. iii, 104

iSAM Incremental Smoothing and Mapping: Novel SLAM framework that is based on fast incremental matrix factorization. iSAM is efficient and provides an exact solution to the information matrix by QR factorization. 25, 104

iSAM2 Incremental Smoothing and Mapping 2: Extended novel SLAM framework that is based on fast incremental matrix factorization using Bayes Tree which is a graphical model inference

algorithm. iSAM2 is more efficient than prior iSAM and presents a relationship between graphical model inference and matrix factorization. 19, 104

KF Kalman Filter: Linear discrete-time algorithm that consists of a prediction step and update step, from extroceptive sensors for measurements. 15, 104

KNN K-Nearest Neighbors: non-parametric supervised learning classifer where a sample is distinguised by being associated by its neighbouring samples, based on density, as well as other parameters. 68, 104

LS Least Squares Regression: Statistical method to find the best curve fit to a set of points. 22, 104

LSD-SLAM Large-Scale Direct Monocular SLAM: novel, direct monocular SLAM framework which instead of using keypoints uses image intensities for tracking and mapping. 13, 104

MSCKF Multi-State Constraint Kalman Filter: Tightly-coupled filtering-based VIO approach where landmark positions are marginalized out of the state-vector. 21, 105

MSF-EKF Multi-Sensor Fusion Extended Kalman Filter: Loosely-coupled filtering VIO framework that is modular in allowing easy integration of many sensors. Advantage of using MSF-EKF is use-case of high framerate camera with high frequency IMU. 21, 105

OKVIS Open Keyframe-based Visual-Inertial SLAM: Tightly-coupled fixed-lag-smoothing keyframe-based VIO using nonlinear optimization, where IMU error is integrated with landmark reprojection error via a proposed probabilistic approach. 23, 105

ORB Oriented Rotated BRIEF: Feature descriptor framework that is used for distinguishing uniquely different features across many frames, building off of BRIEF by being scale and rotational invariant. BRIEF is highly efficient using simple intensity tests. 11, 105

ORB-SLAM ORB-SLAM: Versatile and monocular visual odometry algorithm that can track robustly the 6 DoF camera pose, trajectory and local sparse map of the environment. 11, 105

ORB-SLAM2 ORB-SLAM2: Versatile monocular, stereo and RGB-D visual odometry algorithm that can track robustly the 6 DoF camera pose, trajectory and local sparse map of the environment. 13, 105

ORB-SLAM3 ORB-SLAM3: Versatile monocular, stereo and RGB-D visual odometry and visual-inertial (i.e. with IMU) algorithm that can track robustly the 6 DoF robot pose, trajectory and local sparse map of the environment. 14, 105

PBA Photo Metric Bundle Adjustment: novel framework for Vision-Based SLAM that is based on maximizing the photometric consistency to optimize the structure and motion parameters, unlike standard bundle adjustment that seeks to minimize the reprojection error of features. 13, 105

PC Personal Computer: External device in which the algorithm is being processed on. 1, 105

PE Processing Element: Programmable entity discretized in a grid on the FPSP vision chip, each acting like a mini-computer for processing and computation. 7

PF Particle Filter: algorithm that uses set of particles to represent the state, each assumed to carry a hypothesis of the robot's state. This uses a "Posterior" distribution for the state. 15, 105

PTAM Parallel Tracking and Mapping: algorithm that works well in small AR workspaces, where two base processes are split: tracking and mapping. Both are processed in parallel on the processor for efficiency and faster frame rate. 13, 105

RANSAC 5-Point RANSAC Homography for Determining 6 DoF Camera Pose: Random Sample Consensus (RANSAC) algorithm is used in optimizing a camera trajectory from a selection of 5 random points. 11, 104

RGB-D Camera that outputs red-green-blue colored image, as well as pixel-point correspondence with depth from a reference camera center. 13, 105

RGB-D SLAM Red-Green-Blue-Depth SLAM: Versatile framework for 6 DoF camera tracking and mapping with use of a monocular or stereo vision-setup with other extroceptive sensors such as IMUs, as well as using depth sensors. Camera that outputs red-green-blue colored image, as well as pixel-point correspondence with depth from a reference camera center. 12, 105

RMSE Root-Mean-Square Error: A measure of trajectory error over time. 30, 105

RNN Recurrent Neural Network: Different to CNN which is based in solving problems involving spatial data, RNN deal with sequential or temporal data such as text or videos. 2, 105

ROVIO Robust Visual Inertial Odometry: Monocular visual-inertial odometry that directly uses image patch pixel intensity errors to achieve accurate and robust estimation and tracking. 21, 105

RPY Roll, Pitch and Yaw: Rotational directions with respect to the x, y and z-axes. 40, 105

RTAB-Map Real-Time Appearance-Based Mapping: A RGB-D, Stereo and Lidar Graph-Based SLAM framework that presents an incremental appearance-based loop-closure detector. 14, 105

RTE Relative Trajectory Error: A measure of local trajectory error. 40, 105

SCAMP-5 SIMD Current-Mode Analog Matrix Processor Version 5.0: Particular FPSP vision-sensor model, being the 5th version in development. iii, 105

SCE-SLAM Spatial Coordinate Error SLAM: Real-time semantic learning RGB-D SLAM framework based on tightly-coupling semantic and geometric information for camera tracking and mapping. 14, 105

SIMD Single Instruction Multiple Data: Computer paradigm that processes multiple data in parallel at same time for processing. iii, 105

SLAM SLAM: Versatile framework for 6 DoF camera tracking and mapping with use of a monocular or stereo vision-setup with other extroceptive sensors such as IMUs and depth sensors. 12, 105

STD Standard Deviation: Measure of the spread of data, which is taken to be the square of the variance. 66, 105

SVO Fast Semi-Direct Monocular Visual Odometry: Semi-direct monocular visual odometry algorithm that eliminates use of costly feature extraction and matching, using a probabilistic mapping method to instead directly operate on pixel intensities, allowing fro high frame rate tracking. 22, 104

SVO+GTSAM SVO+GTSAM: Full-smoothing-based VIO approach that uses SVO as visual odometry measurement with factor-graph optimization framework GTSAM. 26, 104

SVO+MSF SVO+MSF: Loosely-coupled filtering VIO approach that uses SVO as visual odometry measurement with high frequency IMU used in prediction step of MSF. 22, 104

UAV Unmanned Aerial Vehicle: Guided robotic aircraft, either autonomously or by remote control by use of extroceptive sensors. 2, 105

UKF Unscented Kalman Filter: Discrete-time algorithm that consists of a prediction step and update step, which is based on using several points known as sigma points (instead of using mean and approximate) for estimation, generally enhancing accuracy. 17, 105

VINS-Mono Monocular Visual-Inertial State Estimator: Full-smoothing-based optimization-based sliding-window VIO framework which leverages loop closure, and global pose optimization. This VIO supports both monocular and stereo camera setups. 25, 105

VIO Visual Inertial Odometry: robot state estimation by sensor fusion of visual and inertial measurements. iii, 105

VO Visual Odometry: robot state estimation by visual measurements alone. 2, 105

Index