Analysis of Visual-Inertial Odometry Algorithms for Outdoor Drone Applications

Anand George

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 19.11.2021

Supervisor

Prof. Ville Kyrki

Advisor

D.Sc. Eija Honkavaara

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Anand George

Title Analysis of Visual-Inertial Odometry Algorithms for Outdoor Drone Applications

Degree programme Automation and Electrical Engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Ville Kyrki

Advisor D.Sc. Eija Honkavaara

Date 19.11.2021 **Number of pages** 68+8 **Language** English

Abstract

Visual-inertial odometry (VIO) and visual-inertial simultaneous localisation and mapping (VISLAM) enables mobile robots to localise without relying on global navigation satellite systems (GNSS) or heavy sensors. They enable mobile robots, especially payload critical robots, such as drones, to perform autonomous tasks with limited resources. Localisation of drones for outdoor applications using visual and inertial sensor fusion is of particular interest, since it widens the use cases and reliability of autonomous drones in different flying conditions and environments. The goal of this thesis is to identify suitable VIO/VISLAM algorithms, and to develop a platform for localising a drone for outdoor applications. A stereo camera and IMU sensor suite was developed to collect visual-inertial data, since suitable off-the-shelf systems were not available. Three state-of-the-art VIO/VISLAM algorithms, FLVIS, ORB-SLAM3 and VINS-Fusion, were evaluated with outdoor drone datasets of varying flight altitudes of 40, 60, 80 and 100 m and speeds of 2, 3 and 4 m/s. The estimation results were compared with the ground truth and were quantitatively evaluated. VINS-Fusion estimated the trajectories most accurately among the three algorithms with an absolute trajectory error of 2.186 m and a relative rotation error of 0.862° at an altitude of 60 m for a trajectory of length 800 m. System configurations, algorithm parameters, external conditions, and scene content impacted the estimation results. These factors, further developments and future scopes are discussed along with the obtained results.

Keywords vio, vislam, drone, stereo-vision

Preface

Autonomous systems, especially mobile robotics, has always been an interesting area for me since the early days of my undergraduate studies. This interest motivated me to pursue master's studies and gain more knowledge and exposure in this field. It has been a rewarding journey all this while, especially this thesis work, which has contributed to strengthen my academic, professional and personal life.

This thesis work was conducted at the department of Remote Sensing and Photogrammetry of the Finnish Geospatial Research Institute (FGI). The project is part of Academy of Finland projects, *Finnish UAV Ecosystem – FUAVE* (Decision number 337018) and *Autonomous tree health analyzer based on imaging UAV spectrometry – ASPECT* (Decision number 327861).

I would like to express my gratitude towards my advisor, D.Sc. Eija Honkvaara, for helping and guiding me throughout the thesis. Her enthusiasm for the project and her constant support had encouraged me to try various approaches and methods to learn and explore different ideas related to the thesis topic. I am also thankful to my supervisor Prof. Ville Kyrki for providing direction and guidance from the beginning of this thesis.

People of FGI have been helping and supporting me throughout the thesis. Juha Suomalainen, Teemu Hakala and Niko Koivumäki had helped me with building the system in the initial stages of this work. Their experience with sensors and drones had helped me to choose suitable hardware. Niko and Roope Näsi helped me record all the datasets for this work by flying the drone. Raquel Alves de Oliveira and Niko processed the datasets in Metashape software, which was another important part of this work. Without their help, I would not be able to make progress in this thesis. I would like to express my sincere thanks to all of them and other colleagues at FGI.

Last but not the least, I would like to thank my parents, sisters, and friends for their support throughout my studies.

Otaniemi, 19.11.2021

Anand George

Contents

Abbreviations

ATE	absolute trajectory error
BRIEF	binary robust independent elementary features
BVLOS	beyond visual line of sight
DLT	direct linear transform
FAST	features from accelerated segmented test
FPS	frames per second
GNSS	global navigation satellite system
GPIO	general-purpose input/output
GSD	ground sample distance
IMU	inertial measurement unit
LIDAR	light detection and ranging
ORB	oriented FAST and rotated BRIEF
PnP	perspective-n-point
RE	relative error
ROS	robot operating system
RPE	relative pose error
RTK	real time kinematics
SfM	structure from motion
SIFT	scale invariant feature transform
SLAM	simultaneous localisation and mapping
SSH	secure shell
SURF	speeded up robust features
VIO	visual-inertial odometry
VISLAM	visual-inertial SLAM
VO	visual odometry
VSLAM	visual SLAM

1 Introduction

Over the last decade, unmanned aircraft systems (UAS, drones) have gained much popularity and are being used in various fields. Although remote controlled flights are currently common, autonomous flights and beyond visual line of sight (BVLOS) flights are enabling new sets of autonomous applications ranging from delivery of commercial packages and medical supplies, surveying and mapping to military operations and logistic missions [1]. Availability of accurate position and orientation information is crucial for autonomous navigation of drones [2]. Drones rely on different types of sensors to obtain odometry information and for navigating through known or unknown environments. This sensor suite usually includes global navigation satellite system (GNSS) receivers and inertial measurement unit (IMU) sensors, but complementary systems, such as light detection and ranging (LIDAR) sensors, and/or cameras, are necessary in autonomous systems [3]. Additional sensors enable autonomous systems to perform secure and reliable operations in various environments [4]. Implementing autonomous drone flights is challenging since there are limitations associated with different sensors. GNSS receivers are not very reliable for accurate positioning as the accuracy is degraded or the signal may be blocked by the presence of buildings, trees, or when the signals are reflected off the wall [5, 6]. LIDARs can be heavy in comparison with the weight of the drone and might have limited performance at higher flight altitudes. Moreover, being an active sensor, LIDAR utilises more battery power, which can cause shorter flight times for the drone.

Visual odometry (VO) is an odometry technique used in various domains, such as robotics, automotive, and wearable computing. It is the process of estimating the ego-motion of a system using the input from a single or multiple cameras attached to it [7]. Earlier works have explored the idea of using images from a monocular camera or stereo cameras mounted on a drone to calculate its ego-motion using VO [8–13]. Other studies [14–16] have mapped the region along with the path and developed visual simultaneous localisation and mapping (SLAM) algorithms. SLAM is the process of simultaneously building a map of the environment and localising the system in that map [17]. Inertial data can be incorporated with visual data to increase the accuracy and robustness, resulting in visual-inertial odometry (VIO) and visual-inertial SLAM (VISLAM) solutions [18–21].

Integrating VIO/VISLAM algorithms with outdoor drones enable them to perform various tasks autonomously without relying on GNSS or heavy sensors, such as LIDARs. Open-source implementations of VIO and VISLAM algorithms are already available and they have been evaluated with the data collected using drones. State-of-the-art VIO algorithms are compared with publicly-available drone datasets in [22], and autonomous drone navigation has been implemented using the odometry output from the VINS algorithm [23–25]. In most of these works in which VIO/VISLAM algorithms are used, the drone is flown inside the buildings while collecting the data. According to the author's knowledge, there does not exist evaluations or comparative studies of these algorithms with outdoor drone datasets, which are relevant for many drone applications, especially the BVLOS tasks.

The objective of this thesis is to identify suitable VIO and/or VISLAM algorithms,

and to design the required sensor suite for localising a drone in outdoor environments for autonomous outdoor applications. The accuracy and reliability of existing solutions in different scenarios have to be studied, along with an understanding of the effects of sensor configurations on the accuracy have to be realised. To accomplish these goals, the thesis evaluates three state-of-the-art VIO algorithms with the data collected using a custom-made stereo camera and IMU sensor suite mounted on an off-the-shelf drone. Camera and IMU data collected at various flight configurations are processed by VINS-Fusion, ORB-SLAM3, and FLVIS algorithms to estimate flight trajectories, which are compared with the ground truth data. This thesis focuses only on the odometry/ego-motion of the drone; maps generated by VISLAM algorithms are not compared with the ground truth in this work.

The remainder of this thesis is structured as follows. Chapter 2 provides the required theoretical background which supports the algorithms and evaluation methods presented in this work. Chapter 3 introduces the VIO and VISLAM algorithms. The platform is presented in Chapter 4, which includes the sensor suite and the additional hardware, as well as the software. Chapter 5 describes the data collection methods, overview of recorded datasets, data processing, and the error metric calculation. Chapter 6 compares the estimation results with the ground truth. These results are analysed in Chapter 7. Chapter 8 concludes the thesis and discusses the scope for future work.

2 Background

Visual-inertial odometry/SLAM is an extension of VO/VSLAM where inertial data is coupled with visual data to estimate the position and orientation of the sensors. VO and VSLAM techniques fall under structure from motion (SfM). In SfM, 3-dimensional (3D) reconstruction and 6 degree-of-freedom (DOF) pose estimation from unordered image sets whereas VO estimates 3D motion from sequential images by computing and concatenating the relative motion between the images, and by optimising previous poses to refine the path locally. VSLAM, along with VO, incorporates loop detection, and a global graph optimisation for trajectory estimation, and mapping of the area. In VIO/VISLAM, inertial data is fused with above techniques to improve the estimation as well as to make the system robust by adding an additional sensor. This chapter provides the necessary background knowledge to understand the algorithms and frameworks used in this thesis.

2.1 Feature detection and tracking

Features are recognisable elements in the environment which can be extracted from the images of the environment and can be described mathematically. A feature, also known as a keypoint, is an image pattern that differs from its neighbourhood in terms of texture, colour and intensity. Features can be classified into different categories based on their properties. First class includes edges and blobs, which give some semantic interpretation of the feature. The features in the second category do not have a semantic interpretation, but they are used in applications such as feature tracking and 3D reconstruction as their location can be determined accurately. The third type is used to recognise an object or a scene, whose location is not relevant whereas the number of matches is important. This type of features are used in the visual-word based place recognition [26].

Edges and corners are detected in the locations where there is an abrupt intensity change in the image. The corner, which is defined as the intersection of two or more edges, has high localisation accuracy – accurately localising a feature both in image position and scale. Corners are usually detected using Harris [27], Shi-Tomasi [28], and FAST (Features from Accelerated Segment Test) detectors [29]. A blob is an image pattern which is not a corner, but differs significantly from its neighbours in terms of texture and intensity. Even though blobs have less localisation accuracy than a corner, they are better in the case of place recognition as they are more distinctive than corners. Blobs are detected by using methods such as laplacian of gaussian (LoG), difference of gaussians (DoG), speeded up robust features (SURF) [30], and scale-invariant feature transform (SIFT) [31].

The Harris detector has been identified as the most stable corner detector [32]. Corners are invariant to image rotation and affine intensity changes, but not to the image scale. The Shi-Tomasi detector is derived from the Harris detector and has similar invariance properties. FAST detector is more efficient as compared to Harris but is not robust at high levels of noise. Even though these corner detectors are not scale invariant, by analysing images at multiple scales for scale estimation, they can

be made scale invariant. Since the appearance of corners varies little at adjacent scales, the scale estimation of corners is less accurate than blobs.

In comparison with corners, the scale and shape of a blob are defined better as the blob is localised by its boundary while the corner is localised by a point. On the other hand, blobs are located less accurately than corners. SIFT is able to detect robust keypoints which are extremely distinctive as well as successfully matched in images with varying illumination, scale, rotation and viewpoint. SIFT also computes a descriptor which distinctively describes the feature. The SURF detector is inspired by SIFT. It is more efficient than SIFT, at the expense of robustness. Binary Robust Independent Elementary Features (BRIEF) descriptor [33] detects features at a high rate but is not invariant to scale or rotation. Oriented FAST and rotate BRIEF (ORB) descriptor [34], which is based on FAST features, provides rotation invariance to BRIEF features.

Large-scale feature matching and retrieval demands quick ways to narrow down the search to a few likely images on which detailed matching techniques are applied. Similar to the approaches in fast document retrieval algorithms, where the frequency of occurrence of particular words in a document is used to find matches for a particular query, high dimensional feature descriptors in an image is mapped into visual words and are stored as bag-of-words for matching in a large set of images [35]. This method of feature matching is useful for place recognition, which enables SLAM systems to recognise previously visited places.

The extracted features and corresponding descriptors from an image can be matched with features in another image. It is assumed that the Euclidean distance between feature descriptors can be used to rank potential matches. The simplest matching strategy is to set a threshold to the distance between the matches and to return all of them [35]. These matches can be verified using random sample consensus (RANSAC) method [36]. In contrast to independently searching for features in all images and matching them, feature tracking methods find a set of possible feature locations in the first image, and search for the match in corresponding locations in subsequent images. The Kanade-Lucas-Tomasi (KLT) tracker is developed based on this idea and it has been used widely in real-time applications [28].

2.2 Structure from motion

The process of simultaneously estimating both the camera pose and the 3D geometry using multiple images is known as structure from motion (SfM). The motion of the camera and the structure of the environment is estimated together with the available images. Images can be from different cameras and they do not have to be processed in the same order as they were captured.

The motion or the 3D pose of the camera is estimated from a set of 2D image points and corresponding 3D points [35]. This pose estimation problem, also known as extrinsic calibration, can be solved using linear methods or iterative methods. In linear methods, the camera pose is obtained by solving a set of linear equations which relates the 3D points with their 2D correspondences. Iterative methods optimise the pose solution by minimising the reprojection errors iteratively.

The 3D geometry of the environment can be calculated by triangulating a set of corresponding image locations with the knowledge of camera positions. This is the converse of the pose estimation problem. 3D positions can be computed by minimising the reprojection errors, in the same way the pose estimation problem is solved using the iterative method.

When there are multiple images, simultaneous adjustment of parameters for all cameras and the 3D scene points can be performed using bundle adjustment (BA). Reprojection errors in all images are minimised together during the optimisation. Since BA is an iterative process, convergence to an optimal solution from an arbitrary starting point is not guaranteed. This leads to development of solutions in which an easily computable non-optimal solution for reconstruction is implemented as an initialisation technique, and BA is performed with the obtained initial value to solve the reconstruction problem [37].

2.3 Visual odometry and visual SLAM

Visual odometry is a particular case of SfM technique in which the motion of an agent is estimated by observing images from a single or multiple cameras attached to it [7]. This method of motion estimation, similar to wheel odometry, estimates the pose incrementally by processing sequential images from the on-board camera(s). Compared to SfM, where the final structure and camera poses are refined using an offline optimization (bundle adjustment), VO performs a real-time estimation of 3D motion of the camera as a new frame is received. The local trajectory estimation can be refined using BA.

2.3.1 Formulation

The camera is moved through the area and images are captured at discrete time instants. Two camera positions at adjacent timestamps are related by a rigid body transformation $T \in \mathbb{R}^{4\times4}$ such that

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

Where $R \in SO(3)$ is a rotation matrix, and $t \in \mathbb{R}^{3\times1}$, a translation vector. The set of camera poses for any arbitrary time span contains all the transformations of the camera with respect to the initial coordinate frame. The main aim of VO is to compute all the relative transformations and subsequently concatenate them to recover the full path.

The relative motion between the frames can be calculated by two main approaches; appearance-based and feature based methods. Appearance based methods use the information of intensity changes in the corresponding pixels whereas feature based techniques extract salient features which are matched in both frames for further computation. Feature based methods are more accurate, faster and computationally cheaper than appearance based methods while they require robust feature matching or tracking techniques for better estimation. Figure 1 shows the pipeline of feature

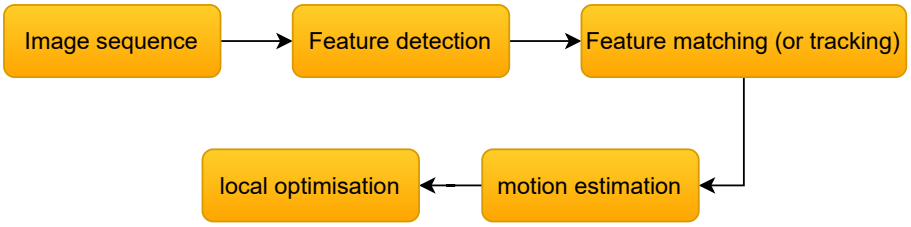


Figure 1: Pipeline of visual odometry

based VO. For each input image, features in it are detected and matched (or tracked) initially. Based on these features, relative motion between the frames are calculated. Finally, a sliding window based iterative refinement to minimise the sum of squared reprojection errors of the reconstructed 3D points is performed to obtain a more accurate estimate of the local trajectory.

2.3.2 Motion estimation

The motion estimation from two frames can be performed using different methods depending on the dimension in which the feature correspondences are specified, in 2D or 3D. In 2D-to-2D method, features in both frames are specified in 2D image coordinates whereas the same are specified in 3D coordinates in the case of 3D-to-3D method. In the 3D-to-2D method, features of the current frame are specified by 2D reprojections of the same features specified in 3D coordinates of the previous frame. 3D points are triangulated from two consecutive frames.

Motion estimation from 2D-to-2D feature correspondences are computed by estimating the essential matrix, which describes the geometric relations between two images of a calibrated camera [7]. The corresponding rotation and translation can be extracted from the essential matrix using the five point algorithm [38]. The absolute scale of the translation has to be computed before concatenating the transformations. This scale cannot be recovered from the 2D sequences alone, it is computed by triangulating the 3D points from a pair of two subsequent images. This scale is used to correct the translation between the image pairs. Transformations are concatenated to obtain the full trajectory. Above steps are repeated for every new image input.

3D-to-3D methods computes the camera motion by calculating the aligning transformation of between the two 3D feature sets which are triangulated from two sets of stereo image pairs. The transformation is computed by minimising the L_2 distance between the two 3D feature sets. These transformations are computed and concatenated for each incoming image pair.

2D-to-2D and 3D-to-2D methods are more accurate than 3D-to-3D method as the 2D correspondence methods minimise the reprojection errors whereas the 3D method minimises the feature position error [39]. In 3D-to-2D methods, direct linear transform (DLT) recovers the pose of a camera by solving a set of linear equations which relates the 3D points with their 2D correspondences. Using DLT, the intrinsic calibration of the cameras, along with its pose can be determined. Usually, the calibration matrix of the camera is available from the calibration results, which enables

to solve this linear problem with a minimum set of three point correspondences. This method is known as the perspective-3-point (P3P) algorithm. An extension of this approach, PnP [40] takes in a large number of point correspondences for pose estimation.

Iterative methods, along with linear methods, are preferred to solve the motion estimation problem since minimal PnP solutions can be quite noise sensitive. An initial guess of the pose can be estimated, for example, by using linear 6-point algorithm and the precise pose solution is optimised by iteratively minimising the reprojection error [35]. For each new frame, new features are matched with the features in the previous frame and the camera pose is computed using the PnP algorithm. Features in the new frame are triangulated to match with the new features in the upcoming frame. This process is repeated and the transformations are concatenated.

2.3.3 Camera pose optimisation

Transformations between adjacent frames are concatenated to compute camera poses. It might also be possible to compute transformations between the frames which are not subsequent. Knowledge of these transformations can be used to improve the camera pose estimation by using pose-graph optimisation [41].

The camera pose corresponding to each frame can be represented as a node in a graph where the transformations between the frames are edges between corresponding nodes. In addition to the transformations between adjacent frames, it is possible to compute the transformations between any two frames which are not of subsequent views. These extra transformations can be modelled as edges between corresponding nodes, and add additional constraints in the pose-graph. Non-linear optimisation algorithms try to minimise the cost function, which is the L_2 norm of the difference between the pose of one node and the transformed pose from another node [41].

Usually the nodes are connected to nearby nodes which correspond to nearby camera poses and this accounts for drift accumulation. Re-observing a landmark or moving the camera in a previously mapped area is called loop detection which allows the application of additional constraints in the pose graph [41]. These loop constraints connect the nodes that are spaced at larger intervals of time based on the visual similarity and optimise the graph further to reduce the accumulated drift. The visual similarity between the frames are usually detected using visual word based approaches in which a feature descriptor is represented as a single integer number. These similarity computations are coupled with a visual-word database where all the observed feature descriptors are stored for future comparison.

Windowed bundle adjustment, which is similar to the pose-graph optimisation, is applied to optimise the 3D landmark parameters along with the camera parameters [41, 42]. The optimisation is applied for a set or window of frames to minimise the image reprojection error and this window is moved across the trajectory to optimise the trajectory locally. Since the reprojection error is a nonlinear function, the optimisation is performed using the Newton-Gaussian or Levenberg-Marquardt approaches.

2.3.4 VSLAM

VO aims to realise local consistency of the local map and trajectory to obtain an accurate estimate of the local trajectory whereas VSLAM aims for a globally consistent trajectory as well as a globally consistent map. A global refinement technique is required to estimate globally consistent trajectory and map. The pose-graph maintained along with the estimation process can be optimised globally to adjust the poses to satisfy the constraints between the poses. These constraints include the rigid-body transformation between the poses and loop-closure matches. This non-linear least square optimisation problem can be solved using various open-source frameworks such as g^{2}o [43] or Ceres Solver [44].

A global BA, similar to global pose-graph optimisation, can be performed to jointly optimise the 3D structure parameters and the camera poses of the entire trajectory by minimising the reprojection error corresponding to each frame [45]. The global BA results in accurate estimations at the cost of high computation resource utilisation. A local windowed BA is preferred in real-time applications with limited computational resource capacity.

This chapter presented a short summary of the techniques of computer vision in regard to SfM as well as the basics of VO and VSLAM. An alternate solution for pose estimation problem is implemented by the fusion of visual and inertial sensor data. Next chapter provides necessary background for VIO/VISLAM techniques and introduces state-of-the-art VIO/VISLAM algorithms.

3 Visual-inertial Odometry and SLAM Algorithms

Localisation is one of the fundamental problems for autonomous systems and there have been many solutions presented over the years. Many systems rely on data from GNSS coupled with IMU, and LIDAR sensors, which are fused together for an accurate estimation of position and mapping. For payload critical systems, computationally expensive processes, which require a high processing device, are not feasible; so alternative methods are desired. Cameras and IMUs provide sufficient information to estimate the pose of the system on which the sensors are mounted, and such methods use less resources than LIDAR based solutions [4]. VIO and VISLAM algorithms provide pose and map estimation using visual and inertial data.

In order to get real-time performance with a limited resource computational system, the camera output rate is typically limited to less than 40 FPS, which does not allow fast movement of the camera since adjacent images should have sufficient overlap for estimation. Along with this limitation, vision based estimation systems experience scale ambiguity in monocular setup, and are not robust to images from low textured scenes [46]. On the other hand, an IMU is highly scene-independent, but it has poor signal-to-noise ratio at low accelerations and low angular velocities, and the motion estimated from IMU data alone suffer from quick drift accumulation. But a combination of IMU and cameras can provide a robust and accurate state estimation due to their complementary properties [46].

Visual-inertial odometry estimates the state of the sensor suite using the inertial and visual data inputs. This state usually includes the 6-DOF pose of the IMU, velocity of the IMU, and biases of the gyroscope and accelerometer. In comparison to VO, the extra quantities in the state vector are the biases and the velocity. The biases are needed for calculating the actual sensor values from raw IMU measurements whereas the velocity is needed to calculate position by integrating the acceleration [46].

Early works of visual-inertial estimation fused independent estimation results from IMU and camera inputs. Such systems are known as loosely coupled systems. In contrast, tightly coupled systems include both the visual measurements and IMU states in the estimation framework [21]. In these systems, IMU measurements are integrated to predict feature locations in the next visual frame to improve the feature tracking. Also the coupling enables correction of the drift in the vision-only estimation [46].

Two main approaches for visual-inertial fusion are filter-based methods and optimisation-based approaches. In filter-based methods, the pose of the camera and the landmarks are part of the system state vector. The IMU measurements between two visual inputs are propagated and the result acts as the prediction term for the next visual input. The states are updated with the subsequent visual input. In the optimisation-based techniques, camera poses and landmarks are considered as vertices in a graph. Adjacent vertices are connected using two edges, one corresponds to the reprojection error and the other to the IMU pre-integration. The pose estimation problem is defined as a minimisation of cost functions of the two types of edges [21].

In this thesis, three state-of-the-art VIO/VISLAM algorithms – VINS-Fusion, ORB-SLAM3 and FLVIS – are analysed quantitatively. These algorithms are chosen

based on previous integration with UAVs and the availability of their open-source implementations. The workflow and main features of these algorithms are described in this chapter.

3.1 VINS-Fusion

VINS-Fusion is an extension of VINS-Mono, which is a tightly coupled, non-linear optimisation-based state estimation method by fusing data from a minimum set of sensors [18]. In VINS-Mono, IMU measurements are preintegrated and fused with the features extracted from the monocular camera images to estimate the 6 degree-of-freedom (DOF) position and orientation of the robot or the sensor suite. It also enables relocalisation, has a robust initialisation procedure, four DOF pose graph optimisation, map merging and map reuse features. In VINS-Fusion, apart from a monocular camera and an IMU, additional sensors such as cameras and LIDARs can be integrated seamlessly to improve the estimation results. Both VINS-Fusion and VINS-Mono share a similar pose estimation pipeline, but VINS-Fusion is designed as a general framework which supports multi-sensor combination [47].

3.1.1 Data collection and preprocessing

In VINS-Fusion, each sensor is considered as a factor which provides a measurement. In this review, the type of sensors are limited to visual and inertial as this thesis focuses only on VIO algorithms. An overview of the framework is shown in Figure 2. In the case of a stereo camera and IMU suite, the inertial data is collected at a higher frequency than the image capture rate. This ensures multiple inertial measurements between two consecutive images. Shi-Tomasi corners are extracted from the input stereo images and they are matched in the left and right image. As new images are received, the detected features are tracked using the KLT sparse optical flow algorithm [28]. Sufficient corner features are detected to maintain a minimum number of features in each frame while enforcing uniform feature distribution over the frame. Keyframes are selected based on the number of tracked features, and the average parallax of tracked features in the input images.

The IMU data collected between two consecutive image frames are preintegrated to calculate relative velocity and rotation of the IMU. Preintegration of IMU reduces the need of repropagating the states during the optimisation, which is a computationally expensive process. As explained in [18], the relative velocity and rotation can be calculated using the IMU measurements and the biases of the accelerometer and gyroscope. The biases of the accelerometer and gyroscope are modelled as random walk which are estimated continuously. Repropagation of IMU values is performed only if the bias estimation changes the bias values significantly. Since IMU propagation is not needed repeatedly, the preintegration method reduces the computational resource usage in optimisation-based algorithms.

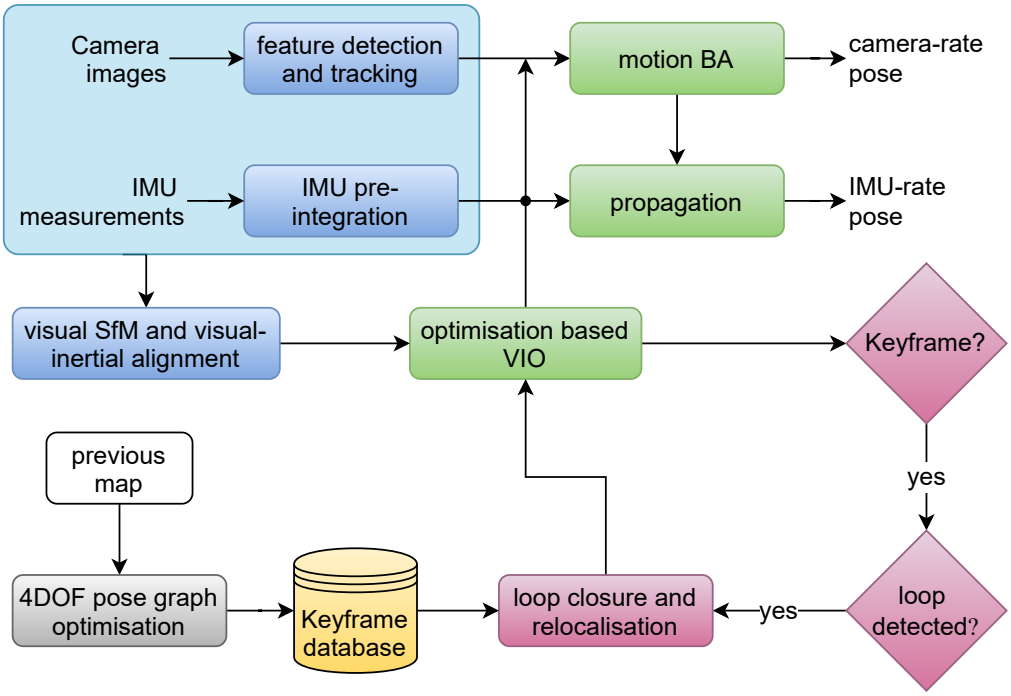


Figure 2: Pipeline of VINS-Fusion. Adapted from [18].

3.1.2 Initialisation

The highly nonlinear tightly coupled VIO have to be initialised accurately at the beginning by loosely aligning the preintegrated IMU values with the vision-only structure [18]. To accomplish this alignment, rotation and translation between the latest frame and one of the previous frames in the sliding window is computed using the five-point algorithm. There should be sufficient parallax and stable feature tracking between these two frames. An arbitrary scale is set and 3D points are triangulated. Also using PnP algorithm, poses of all the other frames in the window are determined. Finally a full global bundle adjustment is applied to minimize the total reprojection errors of all observed features. This vision-only SfM generates a pose graph where the only parameter is the arbitrarily set scale.

The IMU initialisation results in calculating the gyroscope bias, initialising velocity, gravity vector and metric scale, and gravity refinement. Gyroscope is calibrated using the relative rotation between two consecutive frames calculated by visual SfM and the results of IMU preintegration. This new gyroscope bias is used to repropagate all the IMU preintegration terms. The velocity, gravity vector and the metric scale are initialised using the information obtained from the SfM and the new IMU preintegrated terms. The gravity vector is refined by constraining its magnitude to the known value. Finally, rotation between the world frame and the camera frame are computed by rotating the gravity vector to the z axis, and all the variables from the reference frame to the world frame.

3.1.3 Estimation

A sliding window based state optimisation is performed in VINS-Fusion where the nonlinear least square problem of minimising the cost function is solved using Newton-Gaussian or Levenberg-Marquardt approaches. Ceres solver [44], an open sourse C++ library for modeling and solving large optimisation problems, is used in the implementation. The state vector includes the position and orientation of the body in the world frame, depth of each feature observed in the first frame, and the IMU measurements corresponding to each image in the sliding window which includes the position, velocity, rotation and the IMU biases. As the number of states increases with time, in order to reduce the computational complexity, marginalisation of states is incorporated without losing useful information. Marginalisation process converts the previous measurements into prior terms in the estimation process as new features and IMU data are collected. This restricts the number of states which in turn bounds the computational complexity.

3.1.4 Relocalisation

The sliding window with marginalisation method of estimation introduces drift that accumulates over time. A tightly coupled relocalisation technique is incorporated to reduce this drift. The first step for relocalisation is the loop detection. DBoW2 [48], the state-of-the-art library for converting images into a bag-of-word representation, is used for loop detection. The features are detected and described using BRIEF descriptors [33]. Upon detecting geometrical and temporal consistency between the descriptors, DBoW2 returns the loop-closure candidates. Relocalisation process aligns the sliding window to previous poses where poses of all loop-closure frames are constant. After this, all the IMU measurements, visual measurements and the feature correspondences in the sliding window are optimised jointly which reduces or eliminates the drift.

3.1.5 Global pose estimation

A global pose-graph optimisation is performed in addition to the pose-graph optimisation performed in the sliding window to ensure that the set of previous poses are registered in a globally consistent manner. The roll and pitch are fully observable in visual-inertial systems. They are absolute states in the world frame while the position and the yaw values are relative estimates in the reference frame. This results in drift accumulation only in x, y, z, and the yaw angle. So, the pose-graph optimisation is performed after fixing the roll and pitch values, which is a 4-DOF optimisation.

VINS-Fusion supports map merge, save, and load features. For every keyframe, a node is added to the graph with its pose information. Edges between the nodes are defined by either the relative transformation between two frames or if the frames are relocated with loop closure. Along with the node, feature descriptors of corresponding frames are also saved. This graph can be saved and later loaded, which can be

connected with a new graph whenever another frame is relocated with one of the nodes in the loaded graph.

3.2 ORB-SLAM3

ORB-SLAM3 is a feature-based tightly coupled visual-inertial SLAM system [20]. It is developed based on ORB-SLAM2 and ORB-SLAM-VI which are, respectively, visual and visual-inertial systems, to use short-term, mid-term and long-term data association which eliminates drift in mapped areas [15, 16]. ORB-SLAM, the initial framework upon which other variants are developed, uses the same features for mapping, tracking, and place recognition. This avoids the need to interpolate the depth of recognition features adjacent to SLAM features. Also for the place recognition capabilities, features with rotation invariant property are chosen. Oriented FAST and rotated BRIEF (ORB) features are extracted from input images and are associated with 256 bit descriptors [49] for fast computation and matching of features.

3.2.1 System overview

Figure 3 shows the main components of ORB-SLAM3 system. Each component is summarised below.

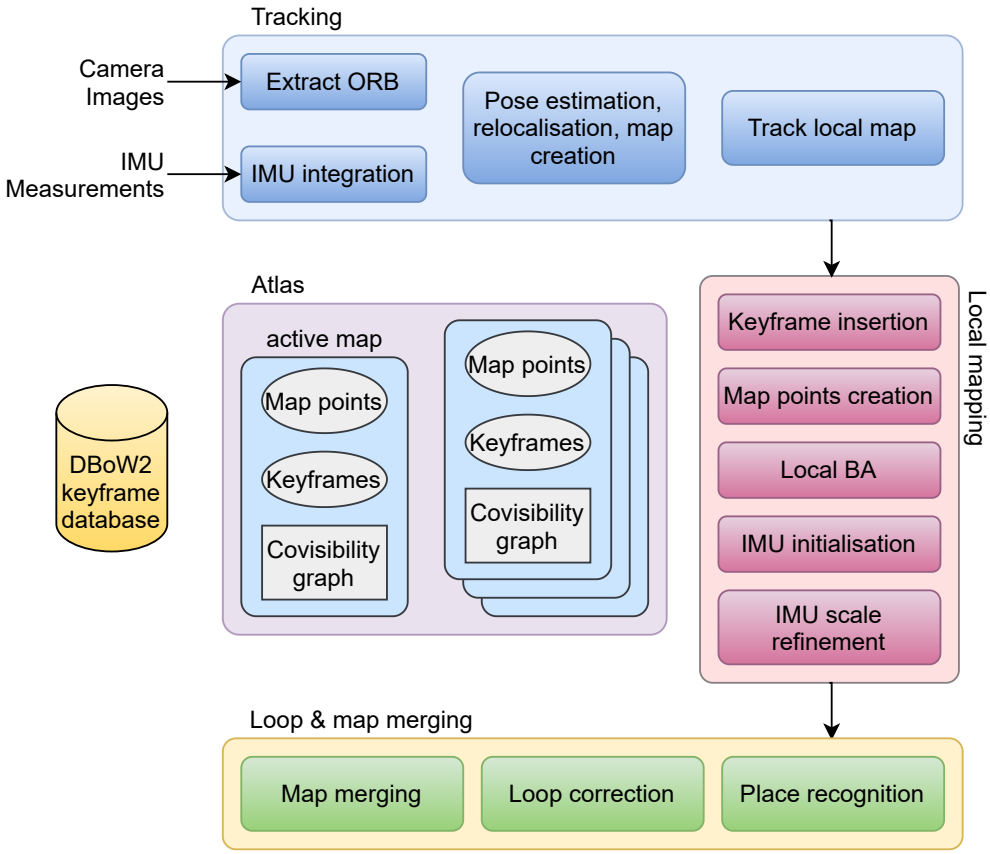


Figure 3: Overview of ORB-SLAM3. Adapted from [20].

Atlas represents/stores multiple maps created while tracking. There are non-active maps and one active map. The tracking thread localises the incoming frames in the active map, which is continuously grown and optimised with new keyframes. An active map becomes non-active when the tracking thread fails to localise the current keyframe in the active map.

Tracking thread processes information from the sensors and estimates the pose of the current frame in the active map by minimizing the reprojection error of the matched map features. The tracking thread initialises a new active map, makes an active map to non-active and switches the active map if needed.

Local mapping thread maintains the active map by adding keyframes and points to it. It also adjust the map using visual or visual-inertial bundle adjustment. If an IMU is present in the system, local mapping thread initialises IMU parameters and refines them using MAP-estimation technique.

Loop and map merging thread performs loop closure if it detects overlapping regions in the active map and other non-active maps in the Atlas. If the new frame overlaps with the active map, loop correction is performed. If the common region belongs to a non-active map, both active map and the non-active map are merged together into a single one, which becomes the active map. The map is refined using a full BA after the loop correction.

3.2.2 Camera model

ORB-SLAM3 supports various camera types and configurations. Apart from the previously supported pinhole camera, the Kannala-Brandt fisheye model is also supported [20]. The images from the cameras, regardless of the type of the camera, are not rectified as it forces to crop out outer parts of the frame to follow the assumption of uniform reprojection error, which is not the case for fisheye cameras, or for the cameras with large field of view (FOV). The same applies to the images from a stereo camera setup. Advantages of large FOV will not be utilised, especially in the case of a divergent stereo pair, or a stereo fisheye camera setup, if images from these setups are rectified. The stereo pair is considered as two monocular cameras of known constant relative transformation between them. If there are overlapping regions between their frames, corresponding landmarks are triangulated to get the true scale from the initial observation.

3.2.3 VISLAM

The state vector for the estimation problem contains the body pose and velocity in the world frame, as well as the gyroscope and accelerometer biases. The IMU measurements between two visual frames are preintegrated to obtain rotation, velocity and position of the IMU [50]. IMU residuals are calculated based on these preintegrated values and the state values corresponding to the input image frame. The visual residual terms are calculated based on the reprojection error and the known rigid transformation between the camera and the IMU. By combining both

visual and inertial residuals, a keyframe based minimisation approach is followed to solve the visual-inertial SLAM problem.

In order to obtain the scale, IMU parameters, and the gravity direction, the IMU has to be initialised. This process is split into three steps. The vision only MAP estimation processes the incoming images for two seconds to estimate camera poses which are combined to create the camera trajectory. In the next step, an optimal estimation of inertial variables are performed by using only the inertial measurements between the received keyframes. The final step is the visual-inertial MAP estimation which optimises the visual and inertial parameters jointly.

ORB features are extracted from the input frame and are matched with the previous frame. From the matched features, the camera pose is predicted and the map points in the local map are projected and matched with the keypoints in the frame. The current frame is optimised by minimising the feature reprojection error of all matched points along with the IMU error term. The optimisation is solved with the Gauss-Newton approach implemented using g^2o [43]. Local mapping manages the keyframes in the fixed local window. Each Time a new keyframe is added to the window, a local BA is applied to optimise the keyframes in the window along with the 3D points seen by the frame. The frames are connected by covisibility graphs whenever they share a common observation. The local BA cost function is a combination of IMU error terms and the reprojection error terms. In the event of short term visual tracking loss, due to camera occlusion or fast motions, the body state is estimated from the IMU measurements, along with the projection of map points in the estimated camera pose. These points are matched with images in a large window and are included in a joint visual-inertial optimisation. This recovers the visual tracking. In the case of a long-term lost, another visual-inertial map is initialised and added to the *Atlas*.

3.2.4 Map merging and loop closing

Similar to the recovery of lost visual tracking, short-term and mid-term data associations between the active and a new frame are found in an image window to expand and optimise the local map. The long-term data association enables loop-closure and relocalisation. This, in ORB-SLAM3, is realised by using the DBoW2 bag-of-words place recognition system. Whenever a new keyframe is created, the place recognition system matches it with the keyframes stored in the *Atlas*. If the keyframe belongs to the active map, loop-closure is performed. On the other hand, if it is part of another map, the corresponding map is merged with the active map. The accuracy of loop-closure and map merging is improved by searching for mid-term data associations in a window defined by the matching keyframes and their neighbours in the covisibility graph.

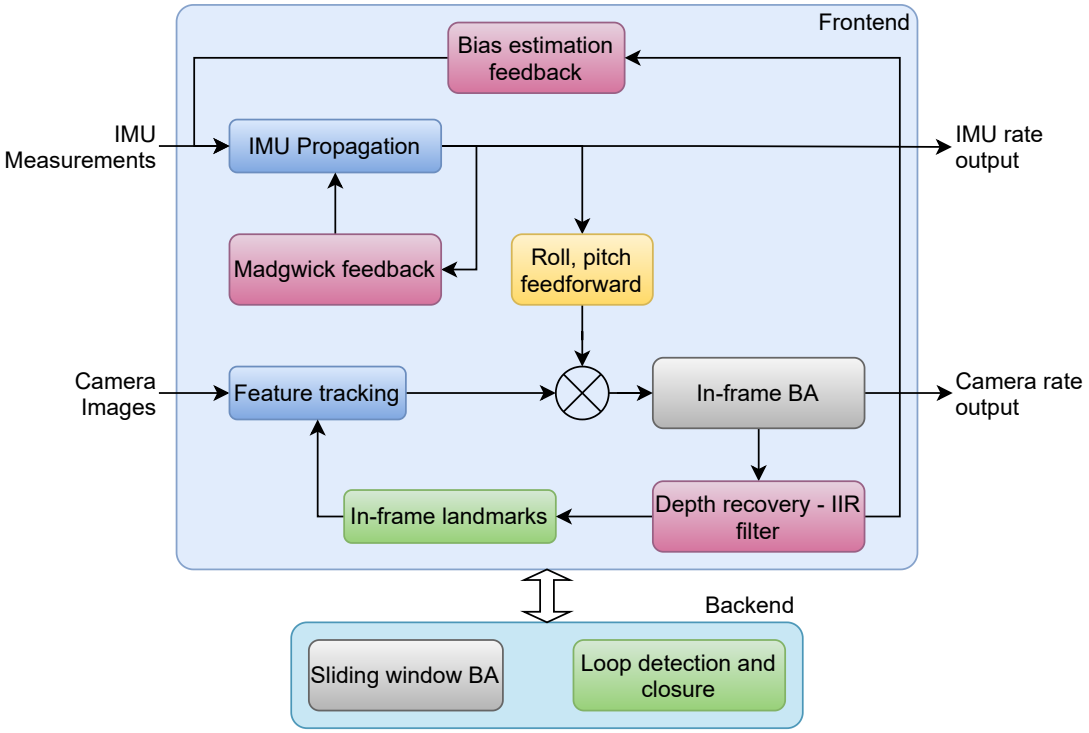


Figure 4: Simple workflow of FLVIS. Adapted from [21].

3.3 FLVIS

Feedback-feedforward loop-based visual inertial system (FLVIS) is a visual odometry technique which is neither a filter-based method nor an optimisation based method, but it leverages the advantages of both methods [21]. The main difference from other techniques are the use of additional filters for propagation of IMU values and for landmark updation, and solving the pose estimation as a control problem.

3.3.1 System overview

A Simple workflow of the system is shown in Figure 4. The measurement model assumes no errors in the system such as no calibration error. The estimation depends only on the accuracy of landmark positions and the quality of the initial guess. A robust orientation estimation with the accelerometer and gyroscope data is performed using a one-step gradient-based Madgwick filter [51] during the IMU propagation phase. The IMU states are forwarded to the vision pipeline as correction terms for the initial guess. IMU bias terms are updated after the BA process. An infinite impulse response (IIR) filter updates the positions of landmarks using the depth information extracted from each frame. The frontend creates and forwards keyframes to the backend where a sliding window based reprojection model optimises all the keyframe measurements and returns the corrections to the frontend. Loop closure is also implemented in the backend.

3.3.2 IMU Propagation

The position, orientation and velocity states are propagated for two consecutive IMU measurements. For a high accuracy IMU attitude estimation, the Madgwick feedback of attitude estimation is adopted. When the sensors provide steady measurements without any acceleration (moving steadily or standing still), the fields of gravity and acceleration measurements will be in the same direction. The Madgwick feedback is applied at this steady condition. This attitude estimation and the propagation model are fused together to obtain the estimations at IMU rate.

3.3.3 Visual estimation

ORB features are extracted from each frame for the camera pose estimation. These features are selected in such a way that they have equal distribution in the 16 predefined regions (grids) in the image pane. For each region, the features are sorted based on their Harris index score [27]. Top 15 features are added to the feature list, which is part of the state vector, after checking their proximity; too close features are not added to the list. These features are tracked by the KLT tracker and the tracked pairs are verified using ORB Hamming distance [52]. The camera pose corresponding to each frame is estimated by using the PnP 3D-2D method, for which the 3D positions of landmarks are accurately estimated by using an IIR filter. An in-frame BA, where camera poses are the only variables in the sliding window, is performed to get the camera frame rate pose estimation output.

3.3.4 Local mapping and loop closure

A new keyframe is added to the mapping thread if there is sufficient motion or visual change in the frame. Each keyframe contains a camera pose and landmark positions. A sliding window with eight continuous keyframes is maintained in the mapping thread. The optimiser refines poses of all the camera frames and landmarks based on the reprojection error except for the first pose in the window, which is set as fixed in the optimiser.

The loop closure thread takes in the new keyframe, and feature descriptors are extracted from each frame. A bag-of-visual-words approach is implemented using DBoW2 to find the loop candidate from the keyframe database. In order to verify the loop candidate matches, a geometry check is performed, which eliminates perpetual aliasing. After the geometry check, a pose-graph optimisation is performed to correct the keyframe poses of the loop. This optimisation is implemented by the Gauss-Newton method in g^{2}o.

This chapter discussed the basics of visual-inertial pose estimation and introduced VIO/VISLAM algorithms used in this thesis. Table 1 summarises the properties of these algorithms.

Table 1: Summary of algorithms

Algorithm	Properties
VINS-Fusion	• Tightly coupled, features based, optimisation-based VIO • Pinhole camera, monocular-inertial, stereo-inertial, stereo sensor configurations • Shi-Tomasi corner features, KLT tracker • Sliding window based optimisation using Ceres Solver • DBoW2 based place recognition with BRIEF features • Global optimisation based on pose-graph
ORB-SLAM3	• Tightly-coupled, feature-based, optimisation-based VISLAM • Pinhole, fisheye camera, RGB-D, Monocular, monocular-inertial, stereo, stereo-inertial sensor configurations • ORB features • Sliding window BA using g^2o • DBoW2 based place recognition with ORB features • Global optimisation with a full BA
FLVIS	• Tightly coupled, feature based, fusion of filter-based and optimisation-based estimation, VIO • Pinhole camera, stereo, stereo-inertial, RGB-D-inertial, RGBD-stereo-inertial configurations • ORB features, KLT tracker • Orientation estimation and depth information extraction using filters • Sliding window BA using g^2o • DBoW2 based place recognition with ORB features • Loop closure and global optimisation based on pose-graph

4 Platform Development

The algorithms mentioned in Chapter 3 support various sensor configurations including monocular vision with inertial data, stereo vision data, and stereo vision with inertial data. In order to leverage the maximum potential of these algorithms, this work used a stereo camera and IMU setup.

Off-the-shelf sensor suites for visual-inertial systems are readily available on the market. Intel RealSense T265 packs two fisheye cameras, an IMU and a video processing unit, which provides raw sensor data along with accurate tracking results [53]. Nerian Karmin3 stereo camera series offers stereo cameras with different baselines, which can be paired up with their SceneScan system to extract data from the cameras synchronously [54, 55]. Even though these systems allow quick testing by eliminating the need of developing a sensor suite, they are not suitable for this study. The T265 has a very short baseline of 10 cm while the Karmin3 series offers a maximum baseline of 25 cm (Nerian also provides custom baseline stereo setup). A shorter stereo baseline results in larger depth error, and the depth uncertainty increases as the height increases. To minimise these issues, a custom sensor suite with longer baseline was developed to collect the visual and inertial data.

In order to collect the data, suitable cameras and IMU were selected. Along with the sensors, a suitable computing device of a small form factor was chosen. Following factors were considered while developing the system. For an easy integration of the sensor suite with the computer, the computer should be compatible with Linux to run robot operating system (ROS) [56] in it, and the availability of ROS drivers for sensors is desired. The weight of the system should be less than the payload capacity. The camera should have a global shutter sensor with a frame rate of at least 50 FPS. Availability of manual triggering, exposure control and binning are also desired. The IMU should be able to output the data at a frequency of at least 100 Hz and it should have trigger control options for synchronisation. Availability of optional RTK corrected GNSS data is preferred as it can be used for generating ground truth data. This chapter describes the developed sensor suite, hardware and software configurations, calibration of sensors and integration of this system on a commercially available drone.

4.1 Hardware

The sensor system comprises two Basler acA2440-75uc [57] cameras and one Xsens MTi-680G RTK GNSS/INS [58] IMU which are mounted on an aluminium bar. This sensor system is connected to GIGABYTE GB-BSi5H-6200-B2-IW (rev. 1.0) mini computer [59] using universal serial bus (USB) 3.0 interface. This system is shown in Figure 5.

The Basler cameras output colour images of 2448×2048 resolution at maximum 75 FPS. It has a 2/3" global shutter sensor of size 8.4 mm $\times$ 7.1 mm, and a physical pixel size of 3.45 μm $\times$ 3.45 μm. Basler provides a ROS package along with camera application programming interfaces (API) for interfacing their cameras easily with ROS environment. A Fujinon HF6XA-5M 1:1.9/6mm lens is attached to each

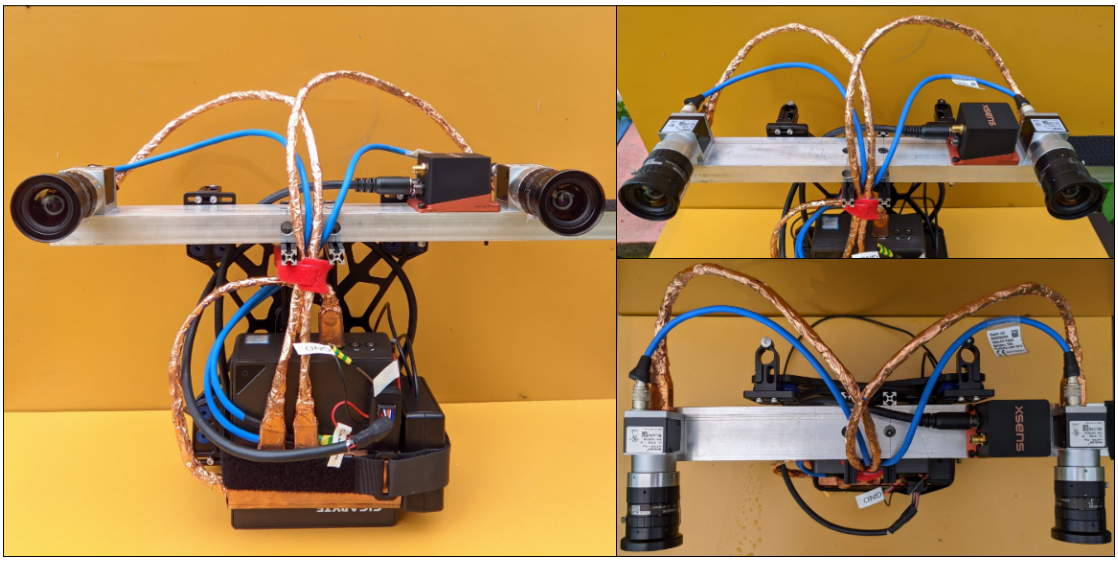


Figure 5: Stereo camera and IMU setup connected to the Intel NUC mini computer

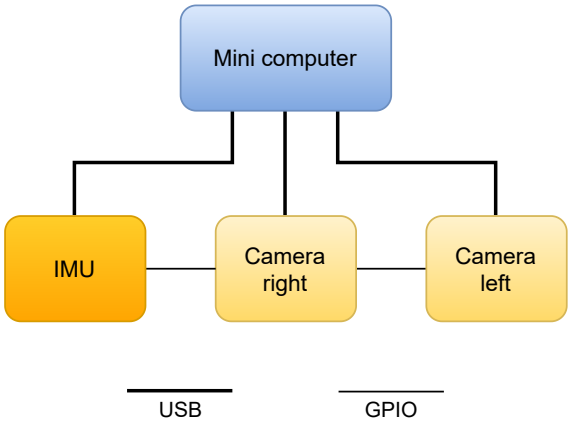


Figure 6: System diagram

camera [60]. It is a 6.23mm fixed focus length lens with maximum resolution of 3.45 μm pixel pitch. The cameras are configured to output monochrome images since the algorithms in this study use monochrome images. Also the frame rate of the camera is reduces to the range 15-25 FPS in order to get realtime performance. Positioning of cameras in the stereo setup is described later in this section.

The Xsens MTi-680G RTK IMU is capable to output inertial data at a frequency of 400 Hz. GNSS data is also available if a receiver is connected to the sensor module.

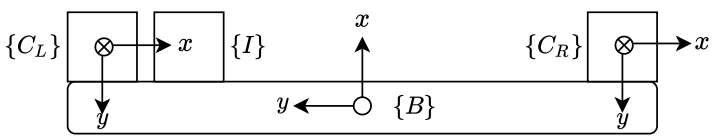
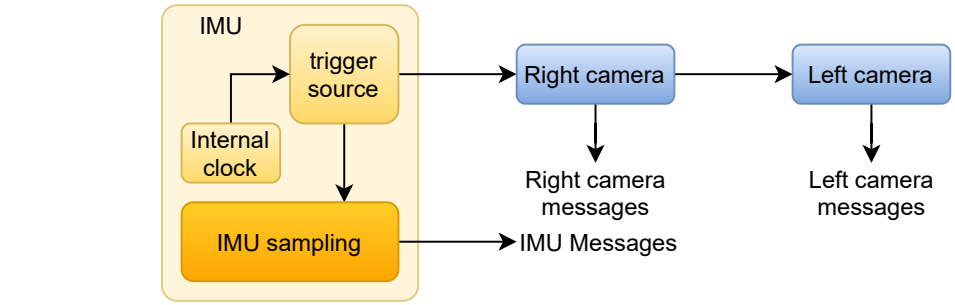
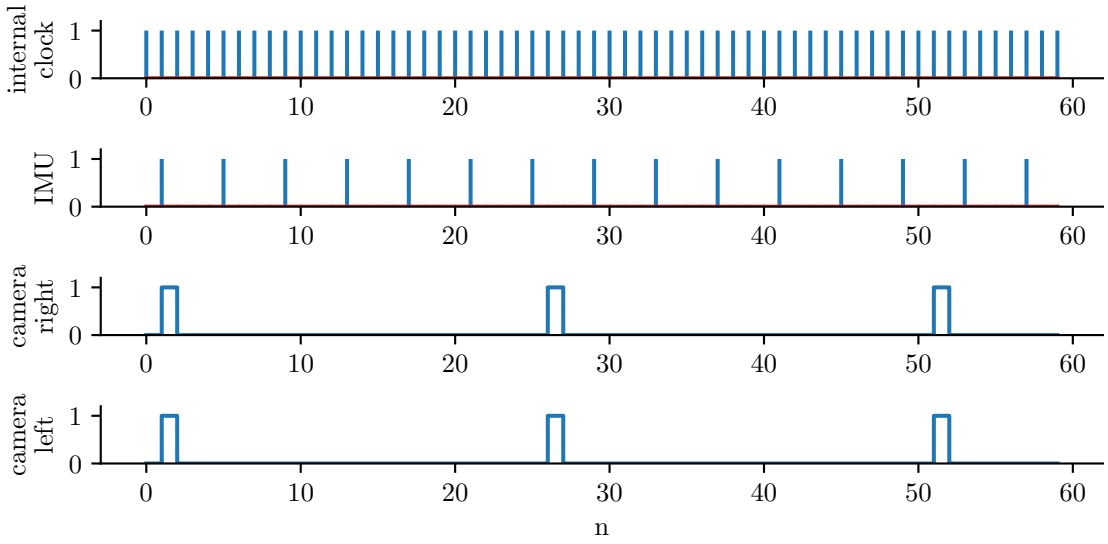


Figure 7: Top view of stereo camera and IMU setup with frames of references



(a) Trigger signal flow



(b) Trigger signals and camera exposure signals.

Figure 8: Hardware synchronisation

It is possible to send real-time kinematic (RTK) corrections to the module to get accurate GNSS measurements. RTK corrections are received from a base station using NTRIP (Networked Transport of RTCM via Internet Protocol) for which an internet connection is needed. In our system, a wireless 4G module is used to access internet. Even though the VIO or VISLAM algorithms do not need GNSS data, it is collected as a ground truth to evaluate outputs of the algorithms.

The sensors are mounted on an aluminium channel. Cameras are placed 30 cm apart from each other to form the stereo camera pair, which faces downwards when mounted on the drone. The IMU is placed in between the cameras at a distance of 5 cm from the left camera. The top view of the setup is shown in Figure 7 along with the frames of references. $\{C_R\}$ and $\{C_L\}$ are the right and left camera frames respectively, frame $\{B\}$ is the body frame. $\{I\}$ is the frame attached to IMU, which is parallel to the body frame. Approximate transformations between these frames are given below. The exact transformations were calculated during sensor calibration.

$$T_{IC_R} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -0.25 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{IC_L} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0.05 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{C_LC_R} = \begin{bmatrix} 1 & 0 & 0 & -30 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{C_RC_L} = \begin{bmatrix} 1 & 0 & 0 & 30 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The sensor system should acquire data synchronously to process it further. This system is synchronised without using any signals from an external device as shown in Figure 8a. Xsens module generates trigger signals based on its internal 400 Hz SDI sampling clock. This signal, with desired skip factor (to reduce the output frequency), triggers the right camera using through its input GPIO pin. Since there is only one output synchronisation pin present in Xsens module, the right camera send trigger signals to the left camera when it captures an image. Since there are no external triggers present, the synchronisation is not exact. On an average, a temporal difference of 200 μs is present between the corresponding stereo images. This difference is handled by software synchronisation described in subsection 4.2. The trigger signals and camera exposure signals corresponding to 100 Hz of IMU sampling rate and 16 FPS of camera frame rate are shown in Figure 8b.

The USB 3.0 devices cause interference to wireless devices with lower radio frequencies [61]. The cameras transfer data through USB 3.0 cables which interfere the GNSS signals by reducing the signal to noise ratio of GNSS signals. This would affect the GNSS data from the Xsens module and also put the drone at the risk of having inaccurate position data by interfering with its position and navigation system. In order to reduce or fully eliminate this problem, camera cables and the USB 3.0 hub are shielded using copper foil tape.

4.2 Software

The data from both cameras and the IMU are processed in the VIO algorithms. These data are collected from the sensors using the drivers provided by the manufacturers. All the algorithms and the drivers are compatible with robot operating system (ROS) [56]. ROS is an open-source middleware, a collection of software frameworks for robot software development [56]. It provides drivers for various sensors, packages for various algorithms, means to communicate between different modules of the system, along with various other features. Modules in ROS are called nodes and communication between the nodes are realised using topics, services, and actions. Nodes can publish or subscribe to a topic to send or receive messages in a one way manner. Service and action servers receive a request from a node in order to achieve a goal for which the server sends back a response based on the result. Availability of a huge number of algorithms, drivers and tools for analysing or monitoring robotic systems makes ROS the most popular middleware in the robotics field.

Basler, the camera manufacturer, provides a ROS package to interface with their cameras. The `pylon_camera`[1] package is a ROS wrapper for their C++ APIs which can be used to receive images from and to configure various parameters of the camera. One of such parameters is the exposure time. This can be adjusted by calling a ROS service provided by the `pylon_camera` package. Pylon Viewer[2], a software provided by Basler is also used to configure the GPIO pin modes and the image acquisition modes. The images from the camera are of size 2448×2048 px (5 Mpix). Since the algorithms have to process these images in realtime, the images are subsampled to reduce their size. OpenCV functions are used for subsampling since the camera does not have binning option. Images are reduced to 612×512 px (0.3 Mpix) size, which is sufficient enough for the algorithms to process.

The algorithms expect the stereo images to have the same timestamps to process these images. Since one camera triggers the other camera using GPIO pins, there is a small latency of $200\,\mu$s between corresponding stereo images. This is addressed by using `ApproximateTime` policy of `message_filters`[3] package in ROS. `ApproximateTime` policy takes messages with similar timestamps and outputs as a single message containing those messages.

IMU data is acquired from the sensor using `xsens_mti_ros_node`[4] ROS package provided by Xsens. The IMU sends electrical signal to trigger one camera. The frequency of this trigger signal, along with other configurations, is configured in the MT Manager[5] software provided by Xsens. The GNSS data is corrected using RTK corrections provided by FinnRef, a Finnish network of reference stations [62]. A ROS package based on `ntrip_ros`[6] streams the corrections through the mini computer that is connected to the networked transport of RTCM via Internet protocol (NTRIP) server of FinnRef.

4.3 Calibration

Stereo cameras and the IMU have to be calibrated to obtain their intrinsic and extrinsic parameters. A multisensor calibration toolbox, Kalibr[7], along with `imu_utils`[8] was used to calibrate the sensor suite. `imu_utils` produces IMU noise values. Stereo camera calibration was performed to calculate intrinsic and extrinsic parameters for each camera using Kalibr toolbox [63]. Finally, using the above two results, a stereo camera and IMU joint calibration was performed in Kalibr [64].

The IMU measurement model used in Kalibr for stereo camera and IMU joint calibration contains two types of errors: additive noise and sensor bias. The angular

[1] https://github.com/basler/pylon-ros-camera
[2] https://baslerweb.com/en/products/software/basler-pylon-camera-software-suite
[3] http://wiki.ros.org/message_filters
[4] https://github.com/xsens/xsens_mti_ros_node
[5] https://mtidocs.xsens.com/mt-manager
[6] https://github.com/dayjaby/ntrip_ros
[7] https://github.com/ethz-asl/kalibr
[8] https://github.com/gaowenliang/imu_utils

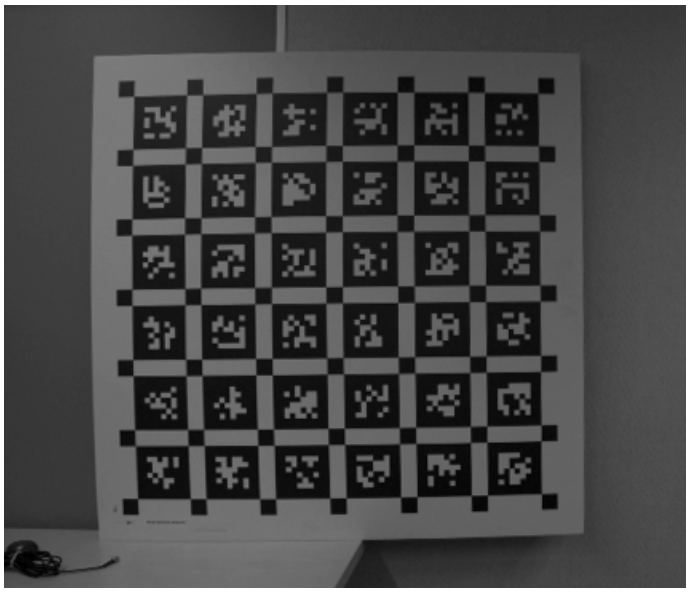


Figure 9: Aprilgrid board captured during calibration

rate measurement, $\tilde{\omega}$ is modelled as

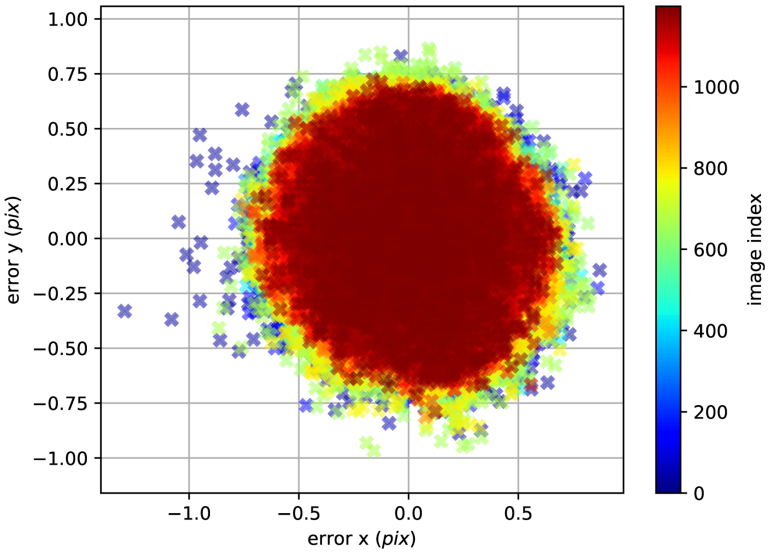
$$\tilde{\omega}(t) = \omega(t) + b(t) + n(t)$$

Here ω is the actual angular rate. This measurement model is for a single axis of gyroscope. The fluctuations in sensor signal are modelled with a zero-mean continuous-time white Gaussian noise $n(t)$ and the variations in sensor bias are modelled with a random walk $b(t)$. Both of these parameters are calculated for the accelerometer and the gyroscope of IMU. These noise model parameters are obtained by analysing Allan standard deviation of the IMU data [65]. This was performed by the `imu_utils` tool with the IMU data collected while the IMU was kept stationary for a duration of two hours.

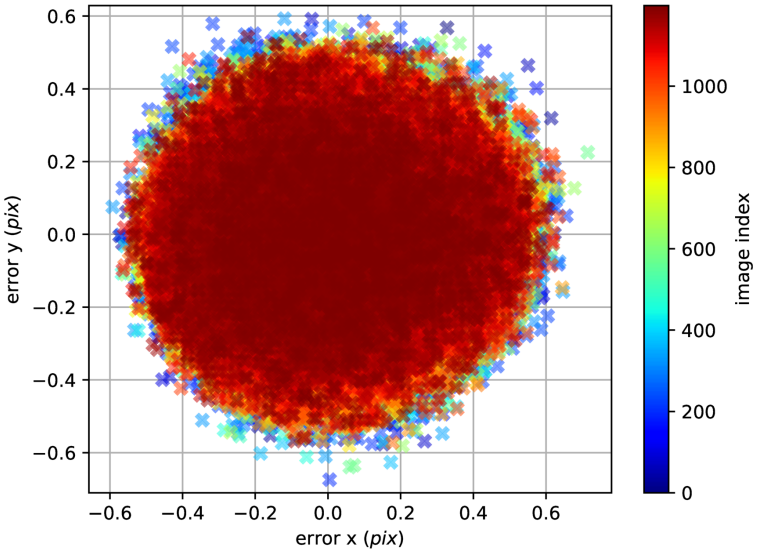
For the stereo camera calibration, stereo image pairs were recorded at a low framerate of 4 FPS as a Bag file. ROS messages can be recorded along with timestamps to a Bag file, which can be played later for further analysis. An Aprilgrid board [66], as shown in Figure 9, was used as the calibration target. The camera was kept stationary and the board is moved while recording the data. Patterns on the Aprigrid targets can be detected individually, which simplifies the data collection as the calibration can be performed even if the target is only partially visible in the image. Reprojection errors of each camera are shown in Figure 10. Kalibr does not provide the stereo rectification parameters. So, it was calculated using the intrinsic and extrinsic parameters obtained by stereo calibration by using OpenCV `StereoRectify`[9] function.

A joint IMU-stereo camera calibration was performed to calculate the extrinsic parameters, such as rotation and translation of cameras about the IMU axis. In

[9] https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereorectify



(a) left camera



(b) right camera

Figure 10: Reprojection errors

this case, the calibration target, Apriltag, was kept stationary and the sensors are moved while collecting the data. Image pairs were collected at a 20 FPS rate and the IMU data was collected at 100 Hz rate. Sensors were moved in such a way that all rotations and accelerations axes of the IMU are excited. This recorded data, along with the results of individual IMU and stereo camera calibrations were processed together in Kalibr to calculate transformation between the IMU and the cameras. Calibration results are presented in Appendix A.

Figure 11: Sensor suite and the mini computer mounted on the drone

4.4 Integration with drone

The system was mounted on a DJI Matrice 600 drone to collect data as shown in Figure 11. The data collection process is completely independent of the systems of the drone. The aluminium channel on which the sensors are mounted was attached to a frame, along with the mini computer. This frame can be easily attached to the drone. There was no gimbal used in this mounting, but the frame was screwed to the drone rigidly.

5 Experiments

Experiments were performed to calculate the accuracy of each VIO/VISLAM algorithm mentioned in Chapter 3. In order to analyse effects of flying altitude and speed, inputs to these algorithms – stereo camera images and IMU data – were collected at different flight conditions. Ground truth data was also collected. The sensor data was processed with the algorithms and the results were compared with ground truth data. Error metrics were computed to evaluate the algorithms.

The algorithms were evaluated in offline processing mode in which the collected sensor data were processed later to estimate the trajectory. There exist various parameters for each algorithm based on their implementations. Most of these parameters, except the sensor calibration details, were kept unchanged. Tuning these parameters would give better estimation results, but it was not performed as part of this work.

The data collection method and relevant parameters are described in Section 5.1. Section 5.2 details the collected datasets and their properties. Data processing and evaluation methods are explained in Section 5.3 and Section 5.4 respectively.

5.1 Data collection

The drone was flown in predefined paths marked by waypoints to collect the data. These waypoints were defined in UgCS drone mission planning and flight control software [67] which generated the mission plan. The flight altitude and flight speed were modified and uploaded to the drone before each flight using this software. These parameters are shown in the Table 2.

Table 2: Flight parameters

Altitudes (m)	Speeds (m/s)
40	2
60	3
80	4
100	

The flight altitude is the distance of the drone from the ground. Depending on the object below the drone (eg. tree, building, field etc.) the distance to the closest object is changing continuously, which has an effect on triangulation. The ground sample distance (GSD) values corresponding to each flight altitudes are listed in Appendix C.

The framerate of both cameras, the stereo baseline and the IMU output frequency were kept fixed for each dataset. Exposure time of the cameras was adjusted based on the lighting conditions before each flight. As mentioned in Chapter 4, the cameras and the IMU output data to the Intel NUC mini computer over the ROS ecosystem. In order to start the ROS nodes and to record the data, the mini computer was

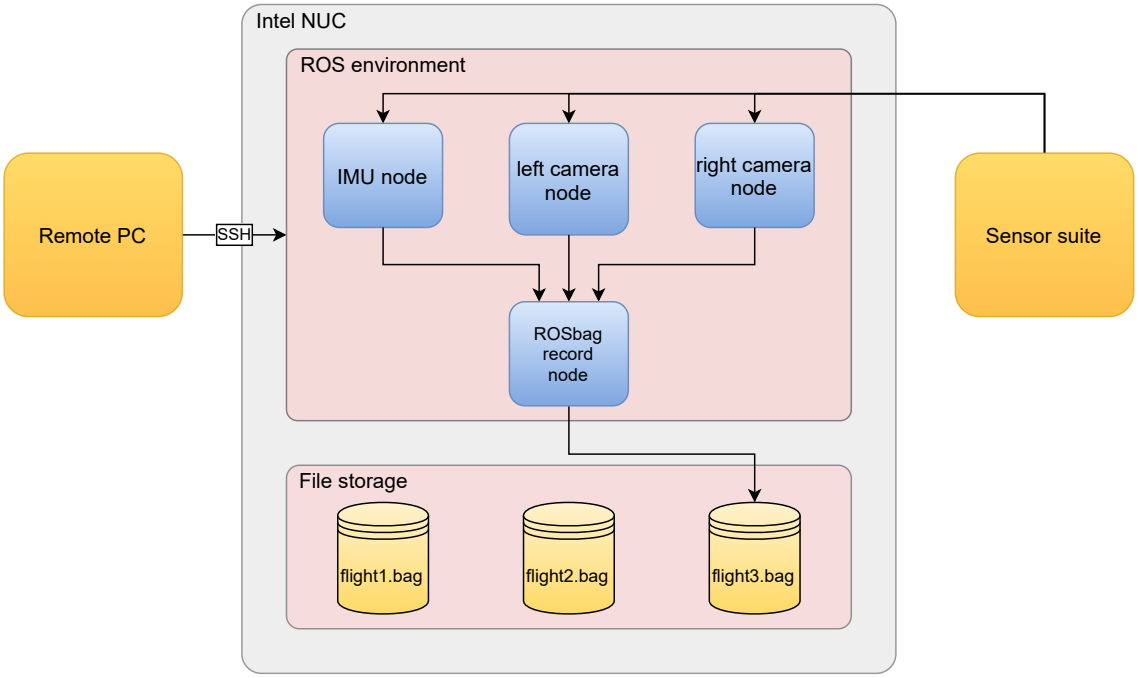


Figure 12: Data flow chart while sensor data recording

accessed over a secured shell (SSH) connection. ROS nodes of the sensors were launched which publish sensor data as ROS messages over specified ROS topics. These messages were stored as bag files[10] in the mini computer. Each bag file contains image messages from both left and right cameras, IMU messages and GNSS messages. IMU messages contain orientation, angular velocity and linear acceleration of the IMU. Table 3 contains details of messages collected in each bag file. Figure 12 depicts the data flow during the data record process.

Table 3: Data collection: details of recorded ROS messages

Sensor	Topic	Message type	Frequency
Left camera	/left/downsample_raw	sensor_msgs/Image	16 Hz
Right camera	/right/downsample_raw	sensor_msgs/Image	16 Hz
IMU	/imu/imu/data	sensor_msgs/Imu	100 Hz
	/imu/gnss	sensor_msgs/NavSatFix	4 Hz

The ground truth of the drone flight was not measured directly. Since RTK corrected GNSS measurements were not able to provide the orientation information, the ground truth was obtained by post processing the recorded images along with the ground control point information in Agisoft Metashape software [68]. Metashape software provides digital photogrammetric solutions including, but not limited to point cloud generation, photogrammetric triangulation, stereoscopic measurements and georeferenced orthomosaic export [69]. Previous studies have shown that estimation

[10] http://wiki.ros.org/Bags

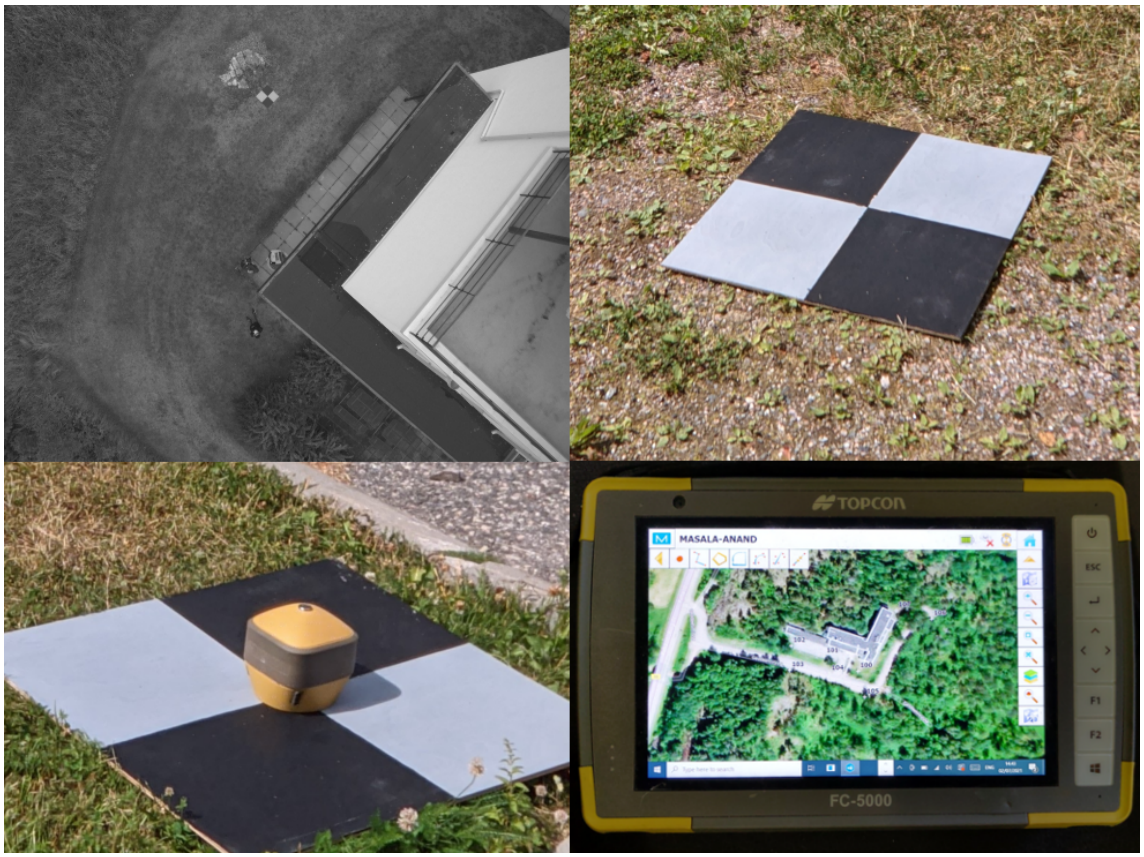


Figure 13: Ground control point measurements for Metashape processing. Top two images show the marker. Bottom left image shows the HiPer HR GNSS receiver on the marker and the last image shows the FC-5000 Field Controller which has marked the ground control points.

results from Metashape are highly accurate, given accurately measured ground control points are available for processing [70]. In this work, photogrammetric triangulation was performed on the stereo images in Metashape to obtain the position and orientation of cameras corresponding to each image. The ground control points were located with the help of markers placed in the mapping area. Positions of these markers were recorded using a HiPer HR GNSS receiver and a FC-5000 Field Controller by Topcon Positioning Systems [71, 72]. Figure 13 shows the marker and its location measurement.

5.2 Datasets

Table 4 lists the collected datasets. These datasets are based on the flight altitudes and speeds mentioned in Table 2. All the datasets except 02.08.2021 have a camera frame rate of 16 FPS and IMU output frequency of 100 Hz. Corresponding camera and IMU output rates in 02.08.2021 dataset are 15 FPS and 30 Hz, respectively. This difference is due to the changes in triggering of the sensors. Here, one of the cameras triggers the IMU. The exposure time for cameras were kept constant based on the

(a) near a building (b) above an open field

Figure 14: aerial views of the test areas

lighting conditions. Data was collected from two areas as described below.

Table 4: Recorded datasets

Altitude (m)	40			60			80			100			Exposure time (μs)
Speed (m/s)	2	3	4	2	3	4	2	3	4	2	3	4	
12.07.2021	✓			✓	✓								500
02.08.2021						✓		✓	✓		✓	✓	500
06.08.2021		✓						✓			✓		1000
17.08.2021						✓			✓			✓	1000
08.09.2021			✓				✓			✓			1200

Area 1: near FGI main building

The drone was flown in predefined paths around the FGI main building with varying flying altitudes and speeds. One of these paths is shown in Figure 15. This area contains large buildings, roads, grass and sparse forest as shown in Figure 14a. The waypoints were defined in such a way that the drone flew in a loop and this path was followed twice during the flight. Total length of the flight path, including take off and landing, varies from approximately 700 m to 1000 m, depending on the flight altitude. Most of the datasets in this work, excluding the datasets of 02.08.2021, were collected from this area, and their details are listed in Table 4.

Area 2: open field

This dataset was collected over an open field with a few trees along the boundary. As shown in Figure 14b, the frame does not contain many features as compared to the

frames in the building dataset. The drone was flown at various heights and speeds as mentioned in the Table 4 under the date 02.08.2021. Flight paths contain small patches in which the drone has already flown, which enable loop closure.

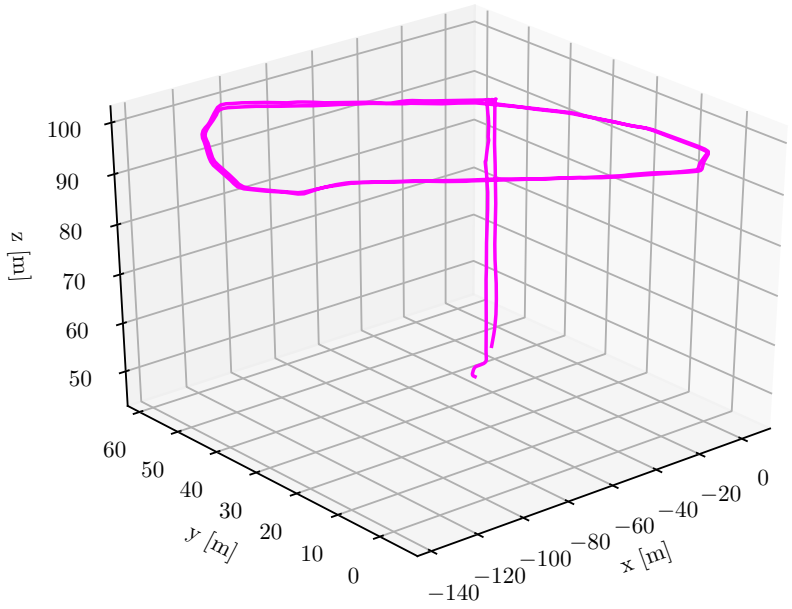


Figure 15: Ground truth data for the flight at 60 m altitude and 3 m/s speed

5.3 Data processing

The collected data was given as input to each algorithm and the estimated trajectory of the drone was obtained. All the algorithms are available as ROS packages (`ORB_SLAM3`[11], `VINS-Fusion`[12] and `FLVIS`[13]) which can be installed along with their dependencies. These packages, after configuring the sensor calibration details, read the recorded sensor data, carry out the trajectory estimation, and write the estimated trajectory to a file. The poses are in local coordinates where the origin is the starting point of the flight.

Images from the recorded bag file were extracted and processed in the Metashape software to obtain the ground truth. Since the images were recorded at 16 FPS rate, adjacent images have high overlap, especially when the drone had flown at a low speed of 2 m/s. Therefore the images were subsampled with a fixed interval, and this small sample along with the ground control points were processed. Output of this process includes the pose of the camera for each image. Poses were generated in both local and geographic coordinate systems. Since this output did not have timestamps along with the pose information, they were added separately from the bag file.

The configuration parameters of each algorithm were almost unchanged. Based on the sensor calibration, camera and IMU parameters were adjusted. Important

[11] https://github.com/UZ-SLAMLab/ORB_SLAM3
[12] https://github.com/HKUST-Aerial-Robotics/VINS-Fusion
[13] https://github.com/HKPolyU-UAV/FLVIS

Table 5: Relevant algorithm parameters

FLVIS	
max. tracked features	480
min. tracked features	240
min. distance between two features	5

VINS-Fusion	
max. features tracked	150
min. distance between two features	30
RANSAC threshold	1px

ORB-SLAM3	
features tracked	1200
scale factor between levels in scale pyramid	1.2
number of levels in scale pyramid	8

and relevant parameters specific to each algorithm are listed in Table 5. In this work, the ORB-SLAM3 estimation is based only on visual inputs, but not visual-inertial data, due to unresolved problems related to the visual-inertial mode[14].

5.4 Error metrics calculation

The quality of estimated trajectories have to be evaluated and analysed to understand and benchmark different algorithms. Studies have proposed absolute trajectory error (ATE) and relative pose error as error metrics to evaluate visual odometry/SLAM algorithms [73]. More informative results are obtained by calculating relative error (RE) for each sub-trajectory of the estimation [74]. [75] provides methods to quantitatively evaluate the quality of visual(-inertial) odometry estimation by computing ATE and RE with respect to the ground truth. It follows a two step approach in which the estimated trajectory is aligned with the ground truth initially, and the error metrics are calculated based on the aligned trajectory.

The trajectory alignment is performed based on the Umeyama method [76], which aligns both the ground truth and the estimated trajectories based on multiple estimated poses of the trajectory. The alignment process calculates a rotation and translation (and scale in case of monocular VO) which is applied on the estimated trajectory to align it with the ground truth.

The ATE of position and rotation give a single number metric corresponding to the estimation. For a single state, the error from the ground truth is parameterised as

$$\Delta\mathbf{x} = \{\Delta\mathbf{R}, \Delta\mathbf{p}, \Delta\mathbf{v}\}$$

where $\Delta\mathbf{R}$, $\Delta\mathbf{p}$ and $\Delta\mathbf{v}$ correspond to the rotation, position and velocity errors respectively. For the entire trajectory, the root mean square error (RMSE) value is

[14] https://github.com/UZ-SLAMLab/ORB_SLAM3/issues/406

calculated to obtain a single metric value.

$$\mathrm{ATE}_{\mathrm{rot}} = (\frac{1}{N}\sum_{i=0}^{N-1} \|\angle(\Delta\mathbf{R}_i)\|^2)^{1/2}$$
$$\mathrm{ATE}_{\mathrm{pos}} = (\frac{1}{N}\sum_{i=0}^{N-1} \|(\Delta\mathbf{p}_i)\|^2)^{1/2}$$

If there is a rotation error at the beginning of the trajectory, the final $\mathrm{ATE}_{\mathrm{rot}}$ value would be higher compared to that corresponding to the case where the rotation error occurs at the end of the trajectory [75]. Since ATE is sensitive to the time of error occurrence, RE, which provides more information about relative changes, is also calculated along with ATE in this work.

The quality of estimations can also be evaluated by measuring relative relations between the states at different times since there exists no global reference for VO/VIO systems [75]. RE is computed by splitting the trajectory into small sub-trajectories, aligning each of them separately, and calculating the RMSE of the errors between the ground truth and the aligned sub-trajectory, similar to the computation of ATE. Sub-trajectories are defined using a set of pairs of states, $d_k = \{\hat{\mathbf{x}}_s, \hat{\mathbf{x}}_e\}, e > s$. These pairs are selected based on some criteria (e.g., distance along the trajectory). For each sub-trajectory, the errors are parameterised as

$$\delta\mathbf{d}_k = \{\delta\phi_k, \delta\mathbf{p}_k, \delta\mathbf{v}_k\}$$

Rotation, position and velocity errors in $\delta\mathbf{d}_k$ are calculated similar to ATE, and the error calculation is performed for all the sub-trajectories. Unlike ATE, the relative error provides a number as a metric for each sub-trajectory as given below.

$$\mathrm{RE}_{\mathrm{rot}} = \{\delta\phi_k\}$$
$$\mathrm{RE}_{\mathrm{pos}} = \{\delta\mathbf{p}_k\}$$

In this thesis, the algorithms were evaluated based on error metrics $\mathrm{ATE}_{\mathrm{pos}}$ and $\mathrm{RE}_{\mathrm{rot}}$. ATE and RE of the estimated trajectories were calculated using an open-source tool, `rpg_trajectory_evaluation`[15], which is based on [75]. Inputs to this tool were the estimated trajectory, ground truth, and time range for which the evaluation has to be performed. Both trajectories were given in separate files in the below format.

```
timestamp x y z qx qy qz qw
```

Here `x`, `y` and `z` correspond to the 3D position and `qx`, `qy`, `qz` and `qw` correspond to the rotation in quaternion format. For the RE calculation, the sub-trajectory length was set to default values in the evaluation tool. By default, five sub trajectories of different lengths were chosen for the calculation. These lengths were 10, 20, 30, 40 and 50% of the total length of the trajectory.

[15] https://github.com/uzh-rpg/rpg_trajectory_evaluation

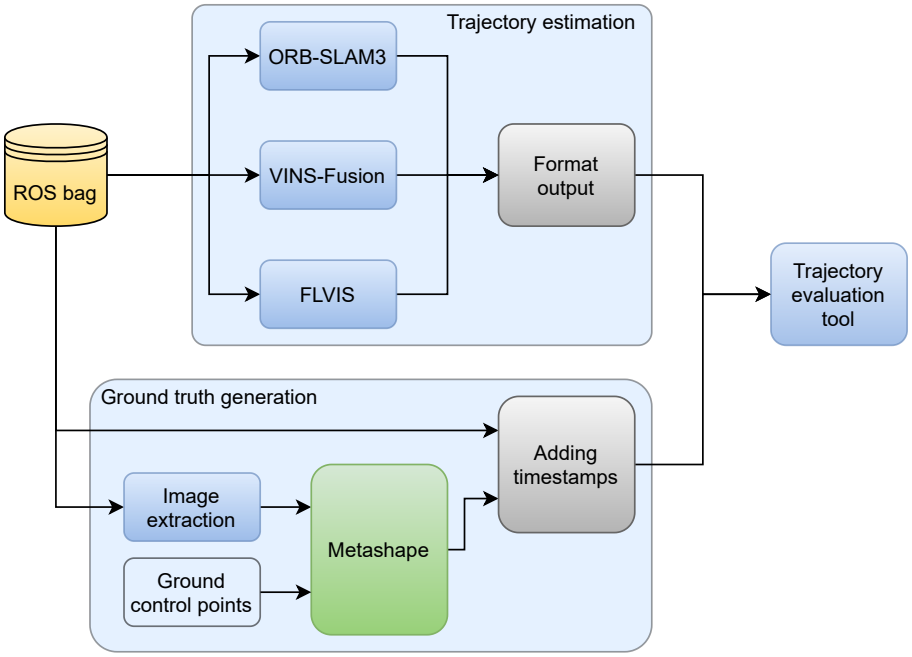


Figure 16: Data processing and trajectory evaluation data flow

Figure 16 shows the sensor data processing and evaluation as a flowchart. Outputs of this tool include ATE and RE values along with their plots. To get a single value for the relative error, a combined RMSE value from each individual RE was calculated.

This chapter presented the data collection methods, the datasets along with the data processing pipeline. The error metrics for evaluating the algorithms are also explained in this chapter.

6 Results

This chapter presents the results obtained from the trajectory evaluation tool. Even though there are multiple datasets collected from different areas and flight parameters, only some of them were fully processed and evaluated. There are some datasets for which some or all of the algorithms were not able to produce a good estimate of the path, those datasets were excluded from the evaluation. Data collected from the two areas are analysed separately as the visual features differ significantly in both cases.

6.1 Area 1: near FGI main building

Datasets were collected near the FGI main building with the flight parameters listed in the Table 2. The pose estimations from all the algorithms for all the datasets were compared against the ground truth and the results are analysed below. All the three algorithms estimate the path similar to the ground truth for the height 40 m. Since the drone was flown in the same loop twice, the estimation results are expected to resemble the ground truth as there exist many loop-closure candidates. The top view as well as the side view of the estimations along with the ground truth are plotted in Figure 17.

Table 6: ATE of position and RE of yaw for the data collected at different flight parameters. The values are coloured for each algorithm as FLVIS, ORB-SLAM3 and VINS-Fusion.

Height \ Speed	2 m/s		3 m/s		4 m/s	
	ATE(m)	RE(deg)	ATE(m)	RE(deg)	ATE(m)	RE(deg)
40 m	16.603	75.704	19.384	64.204	9.402	73.407
	7.444	85.143	13.508	74.566	11.715	72.687
	21.689	3.627	2.554	2.638	4.636	1.312
60 m	22.375	67.816	16.890	87.258	48.798	79.954
	19.800	70.683	39.517	76.803	33.695	85.022
	10.563	0.949	4.744	0.862	2.186	3.214
80 m	39.021	63.984	28.444	65.497	79.665	95.814
	31.570	88.392	59.692	82.552	60.038	90.912
	4.362	1.414	7.397	4.034	9.245	1.819
100 m	66.804	80.764	107.901	79.658	55.458	70.308
	43.557	82.014	45.541	82.375	71.405	97.672
	12.861	1.497	12.459	4.069	9.544	4.435

The error metrics corresponding to each flight condition are tabulated in Table 6. It contains the RMSE values of absolute position error and relative yaw error of the estimated trajectories compared against the ground truth trajectory after aligning both trajectories together. Each of the flight conditions have three different values

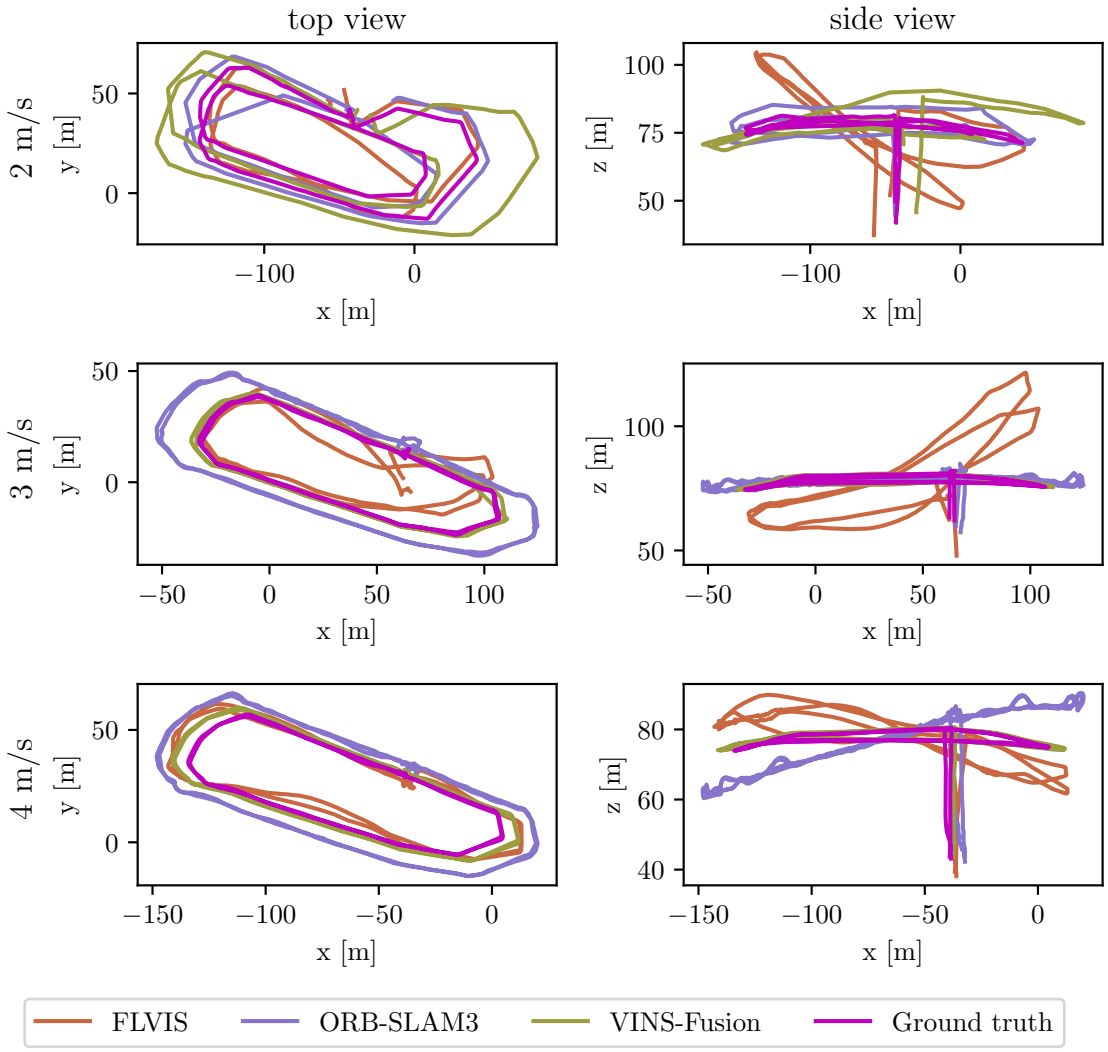
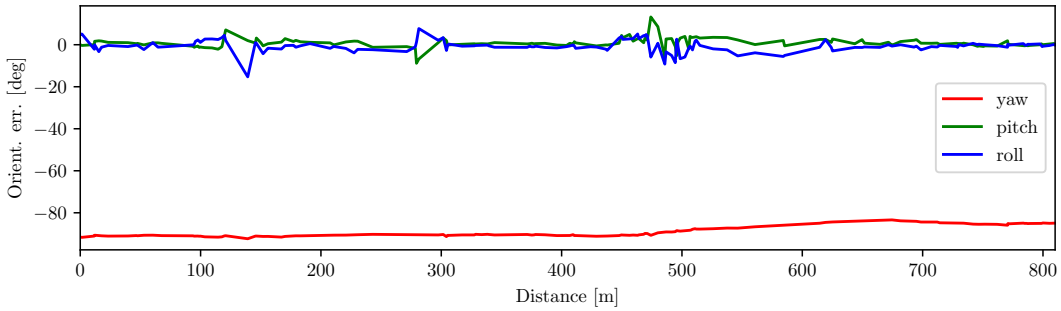


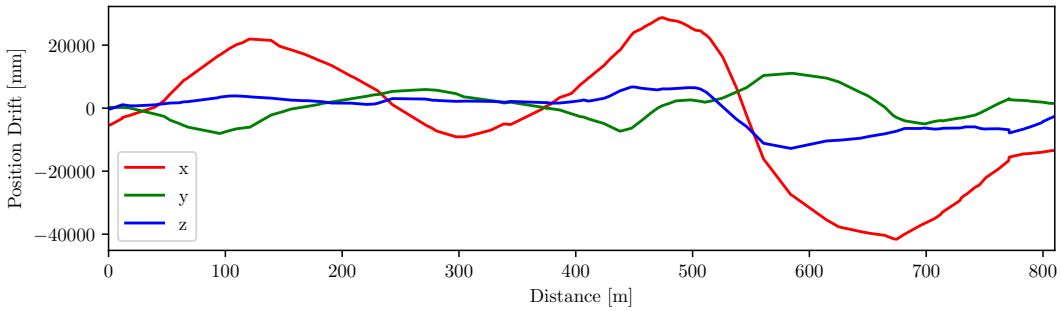
Figure 17: Estimated trajectories plotted along with the ground truth at an altitude of 40 m.

under ATE and RE, which correspond to the estimations of FLVIS, ORB-SLAM3 and VINS-Fusion in the same order.

At an altitude of 40 m, as shown in Figure 17 as well as the error metrics in Table 6, ORB-SLAM3 has the lowest trajectory error at 2 m/s speed whereas VINS-Fusion estimates more accurately at speeds 3 m/s and 4 m/s. In all three cases, VINS-Fusion estimates the yaw angle most accurately with a maximum relative RMSE of 3.63°. It is clear from the top and side views that estimation of FLVIS suffer from drift in the z axis while the x and y axes are almost similar to the ground truth. In the case of ORB-SLAM3, the drift at 4 m/s speed accumulates significantly more along the z axis than along x and y axes, while that along the x and y axes increases as the speed increases.

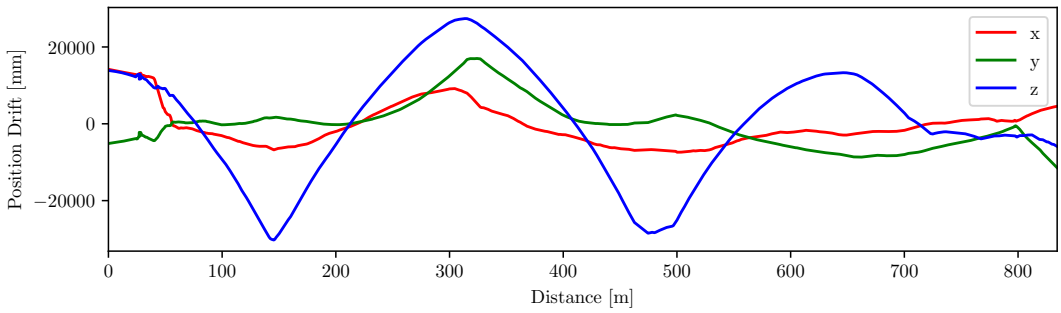


(a) rotation error

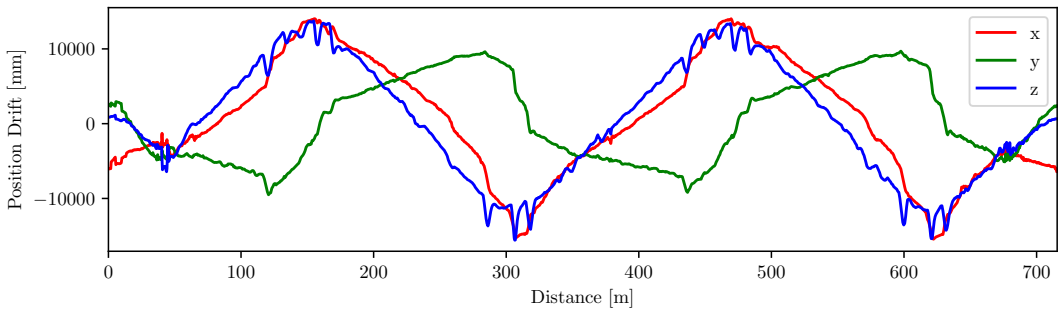


(b) translation error

Figure 18: translation and rotation error of VINS-Fusion estimation at height 40 m and speed 2 m/s

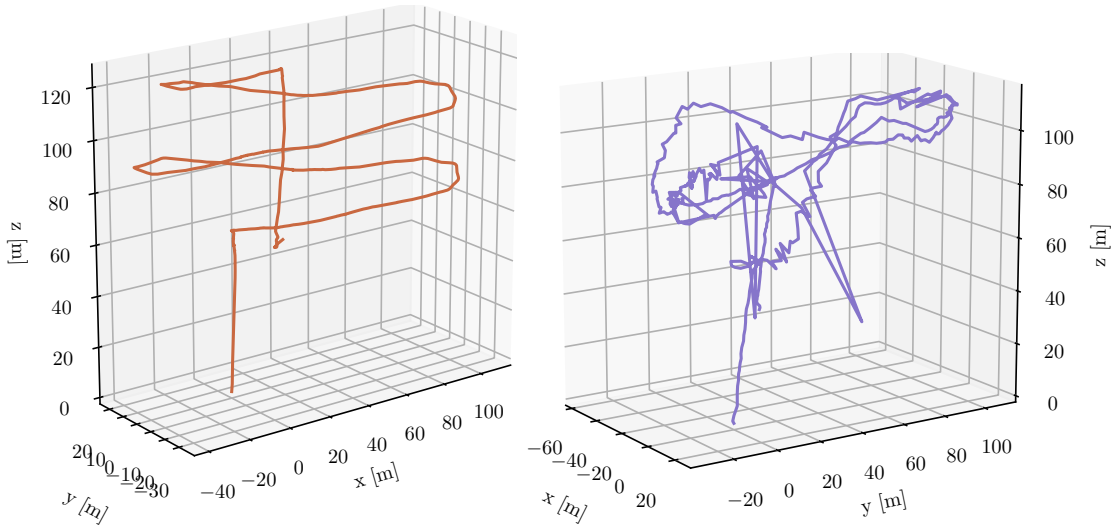


(a) FLVIS estimation at 2 m/s speed



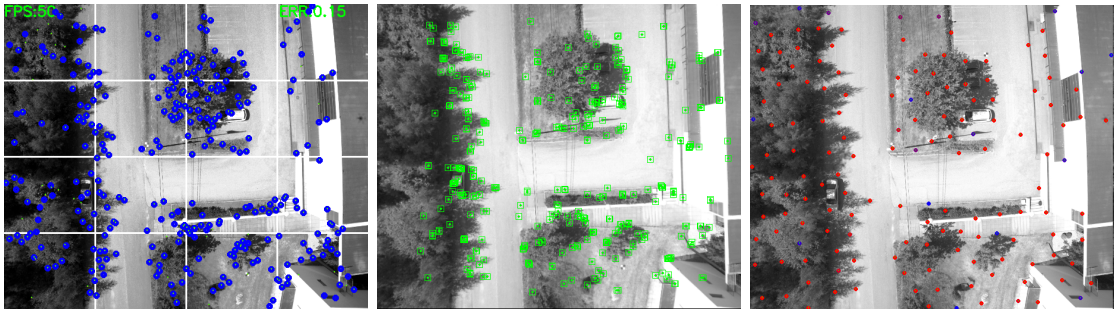
(b) ORB-SLAM3 estimation at 4 m/s speed

Figure 19: translation error at height 40 m



(a) FLVIS (b) ORB-SLAM3

Figure 20: ORB-SLAM3 and FLVIS estimations at height 60 m and speed 2 m/s



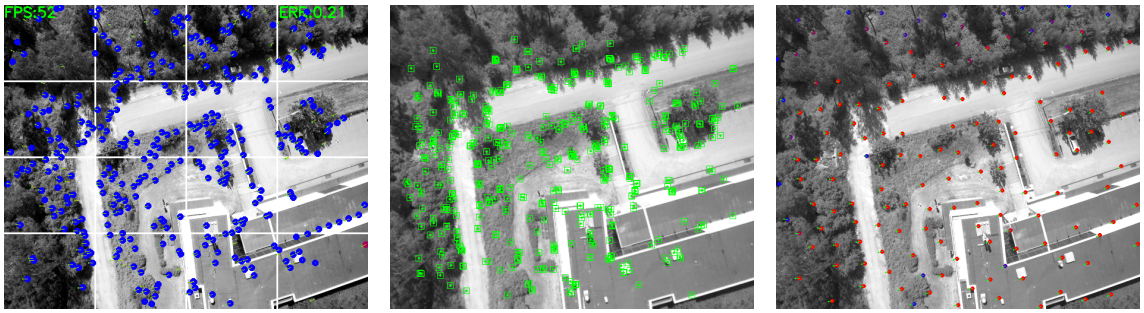
(a) FLVIS (b) ORB-SLAM3 (c) VINS-Fusion

Figure 21: Features detected for each algorithm at height 40 m.

Figure 18 shows translation and rotation errors corresponding to VINS-Fusion estimation at 40 m height at different 2 m/s speed. Sharp increases in the rotation error occur due to the changes in direction of the drone during the flight. As the drone changes its direction, especially when it flies in the opposite direction, the direction of drift accumulation also changes for x and y axes. The same trend is seen in FLVIS and ORB-SLAM3 estimations. Translation errors corresponding to their estimations are shown in Figure 19. The drift accumulation can be attributed to the weakly observed scale due to the limitations discussed in Chapter 7.

The features tracked by the algorithms for the same flight parameters are shown in Figure 21 and Figure 22. Features in VINS-Fusion are spreaded all through the frame while those of FLVIS and ORB-SLAM3 are concentrated in some areas, and they do not cover the entire frame. This distribution is based on the algorithm parameters specified in Table 5.

The remaining sets of data were collected at altitudes higher than 40m. At these altitudes, the estimations of FLVIS and ORB-SLAM3 do not match with the



(a) FLVIS (b) ORB-SLAM3 (c) VINS-Fusion

Figure 22: Features detected for each algorithm at height 60 m.

(a) frame contains only trees (b) frame with more details

Figure 23: different frames extracted during the flight at 40 m and speed 2 m/s

ground truth path. The estimations with the data collected at 60 m altitude and at 2 m/s speed are plotted in Figure 20. Since these estimations do not resemble the ground truth (similar to Figure 15), further analyses are limited to VINS-Fusion estimations. Comparison plots of ORB-SLAM3 and FLVIS estimations are included in Appendix B.

6.1.1 VINS-Fusion – detailed analysis

Flying altitude – 40m

The RMSE of ATE_{pos} of VINS-Fusion estimations at 40 m altitude do not show any particular trend with respect to the flying speed. The ATE_{pos} is highest at 2 m/s speed and lowest at 3 m/s speed. The flight at 2 m/s has significantly high ATE, and as seen in Figure 18, the drift is the highest during the last phase of the flight (from distance 500 m). Images extracted corresponding to this span of flight have only trees in the frame, as shown in Figure 23a, which have similar features (Shi-Tomasi corners) across consecutive frames and thus have caused incorrect matching and poor estimation. Images captured from other parts of the path have much more details, and distinctive features as depicted in Figure 23b. The relative yaw error decreases as the flight speed increases.

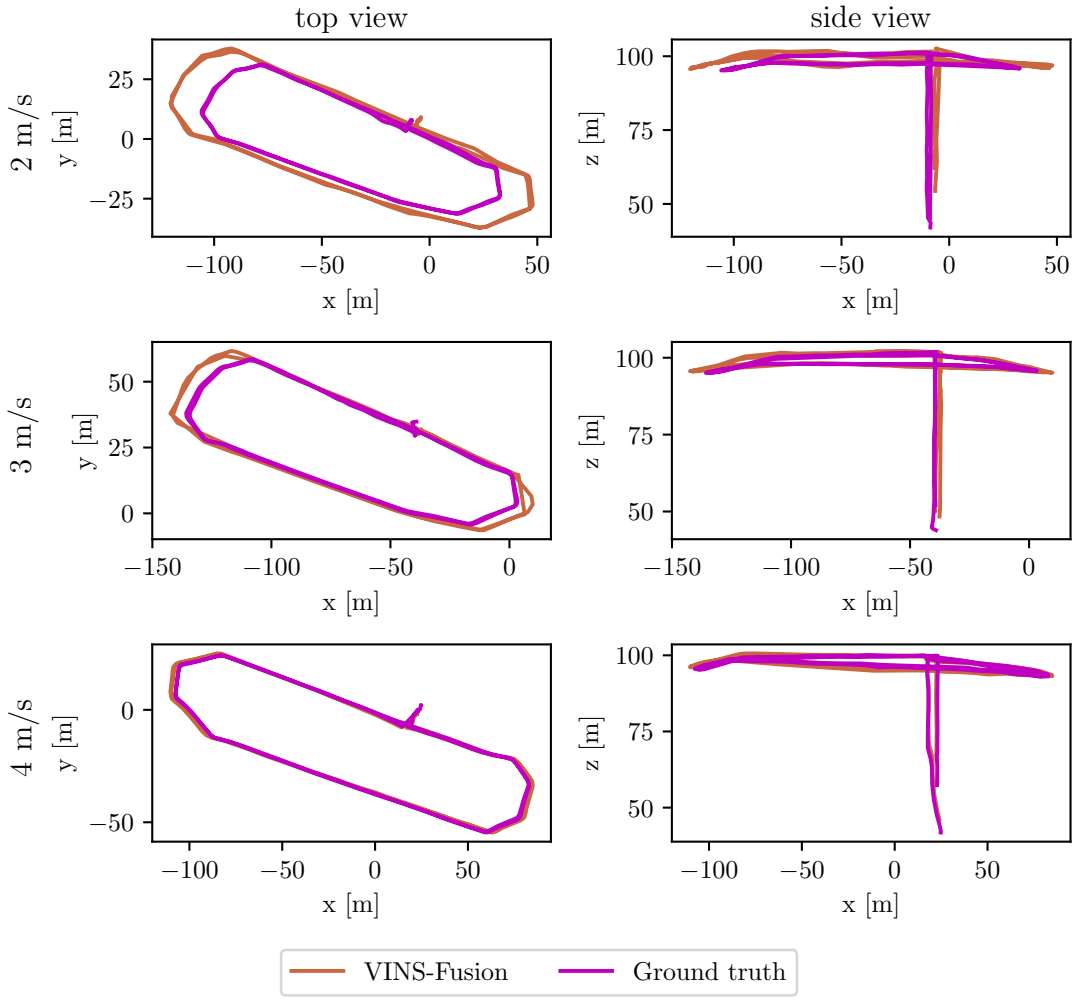
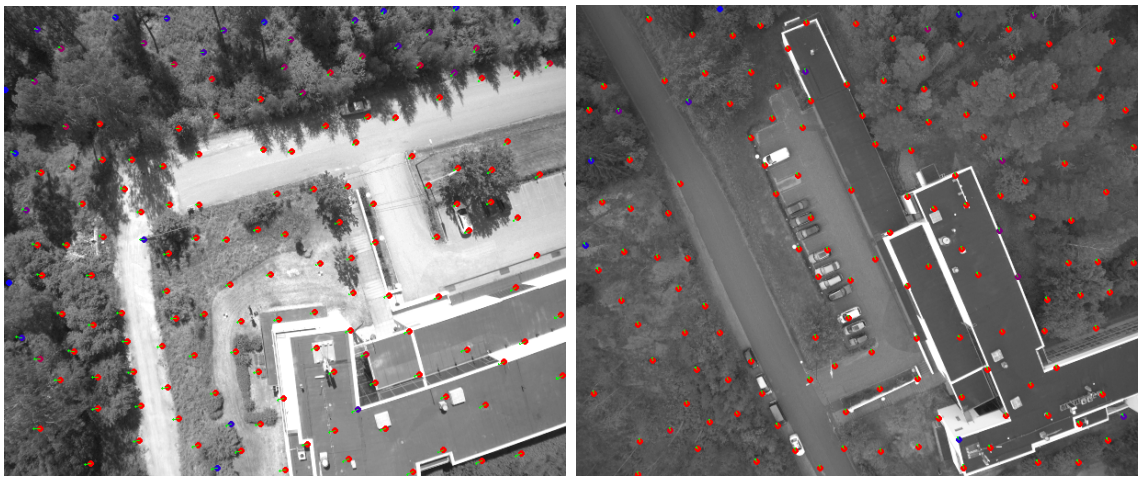


Figure 24: Trajectory estimated by VINS-Fusion plotted along with the ground truth at an altitude of 60 m.

(a) 2 m/s dataset (b) 3 m/s dataset (c) 4 m/s dataset

Figure 25: Difference in illumination of images in 60 m datasets.



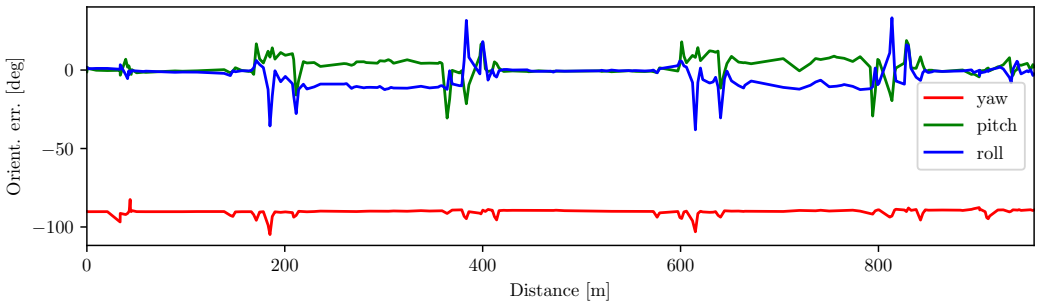
(a) features at 60 m (b) features at 80 m

Figure 26: Features detected in VINS-Fusion estimation at 60 m and 80 m altitude

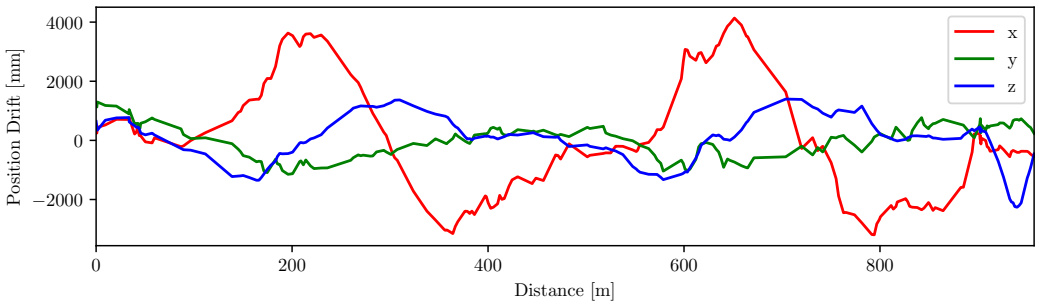
Flying altitude – 60m

Figure 24 plots the VINS-Fusion estimate along with the ground truth for the data collected at height 60 m. As given in Table 6, the VINS-Fusion estimation at 60 m height and 4 m/s speed has the lowest ATE_{pos} value. From the plots it is seen that the drift along each axis reduces as the flight speed is increased. The path taken by the drone while flying at 4 m/s speed was slightly different from the paths corresponding to other speeds due to the variations in the waypoints defined to make the path (as mentioned in Section 5.1). This accounts for the fewer sharp turns in the former path as compared to the latter ones. The features tracked at height 60 m and 2 m/s is shown in Figure 26a. Figures 27, 28 and 29 show the translation and rotation error corresponding to 4 m/s, 3 m/s and 2 m/s estimations respectively. In each case, the drift accumulates in the direction of motion of the drone. But, noticeably, it is decreasing as the speed is increased. The maximum magnitude of drift decreases from above 10 m to almost 4 m as the speed increases from 2 m/s to 4 m/s. The number of new features observed increases as the drone flies faster, which leads quicker to creation of new keyframes. For each new keyframe, a loop-closure detection was performed, which optimises the path and reduces the drift.

Meanwhile, the estimation at 4 m/s speed has the highest relative yaw error at this height. As seen in Figures 27a, 28a and 29a, the absolute value of rotation error for each axis increases when the drone changes its direction. These direction changes are usually fast rotations with minimal translation, which can cause temporary tracking failures. Also the 60 m data was collected in two separate days as mentioned in Table 4. The illumination and the exposure time were different for each case. The 2 m/s and 3 m/s speed flight data images (Figure 25a and 25b) have good exposure while those corresponding to 4 m/s speed (Figure 25c) are underexposed. This low exposure data can be one of the reasons behind the high relative yaw error of 3.21° for 4 m/s data while other speeds have the relative yaw error less than 1°.

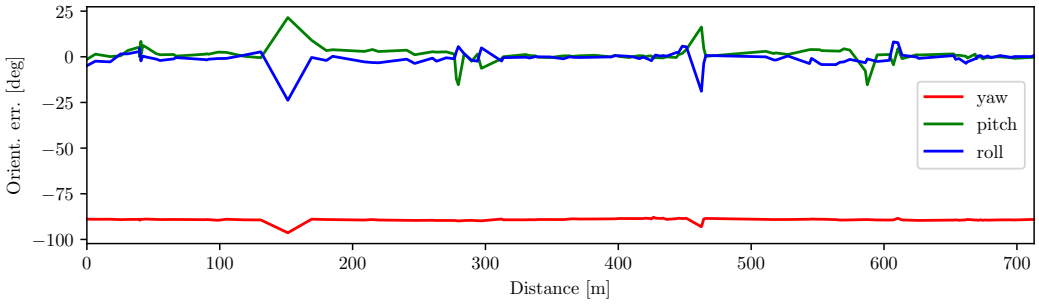


(a) rotation error

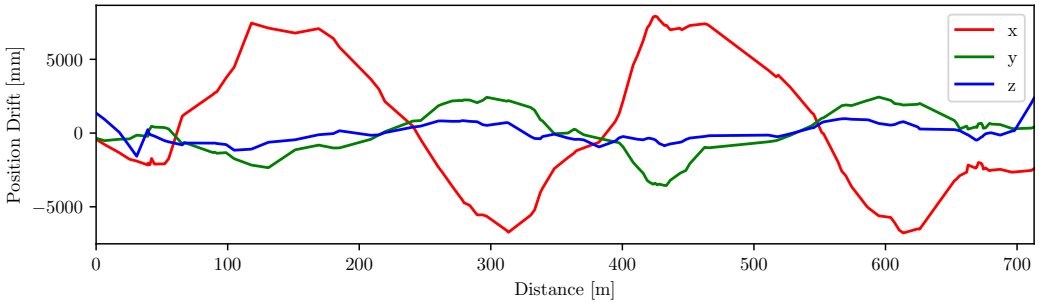


(b) translation error

Figure 27: translation and rotation error of VINS-Fusion estimation at height 60 m and speed 4 m/s

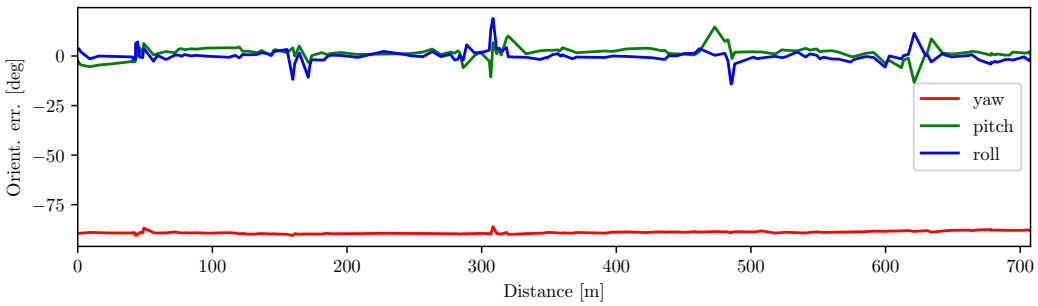


(a) rotation error

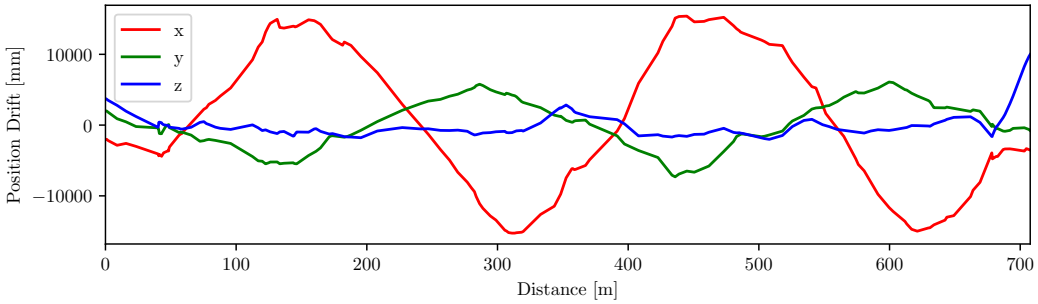


(b) translation error

Figure 28: translation and rotation error of VINS-Fusion estimation at height 60 m and speed 3 m/s



(a) rotation error



(b) translation error

Figure 29: translation and rotation error of VINS-Fusion estimation at height 60 m and speed 2 m/s

(a) 2 m/s dataset (b) 3 m/s dataset (c) 4 m/s dataset

Figure 30: Difference in illumination of images in 80 m datasets

Flying altitude – 80m

The estimation results for different speeds at height 80 m is shown in Figure 31. The RMSE of relative yaw angle error is less than 2° for 2 m/s and 4 m/s speeds, but it is more than 4° at 3 m/s. Similar to 60 m data, the illumination of images affects the orientation estimation. As shown in Figure 30, the images in 3 m/s dataset are overexposed while that of other speeds are a little underexposed. The features tracked at height 80 m and 2 m/s is shown in Figure 26b.

In contrast to the results for 60 m height, the ATE_{pos} value increases as the flight speed increases, as shown in Table 6. There is no clear evidence to make a correlation

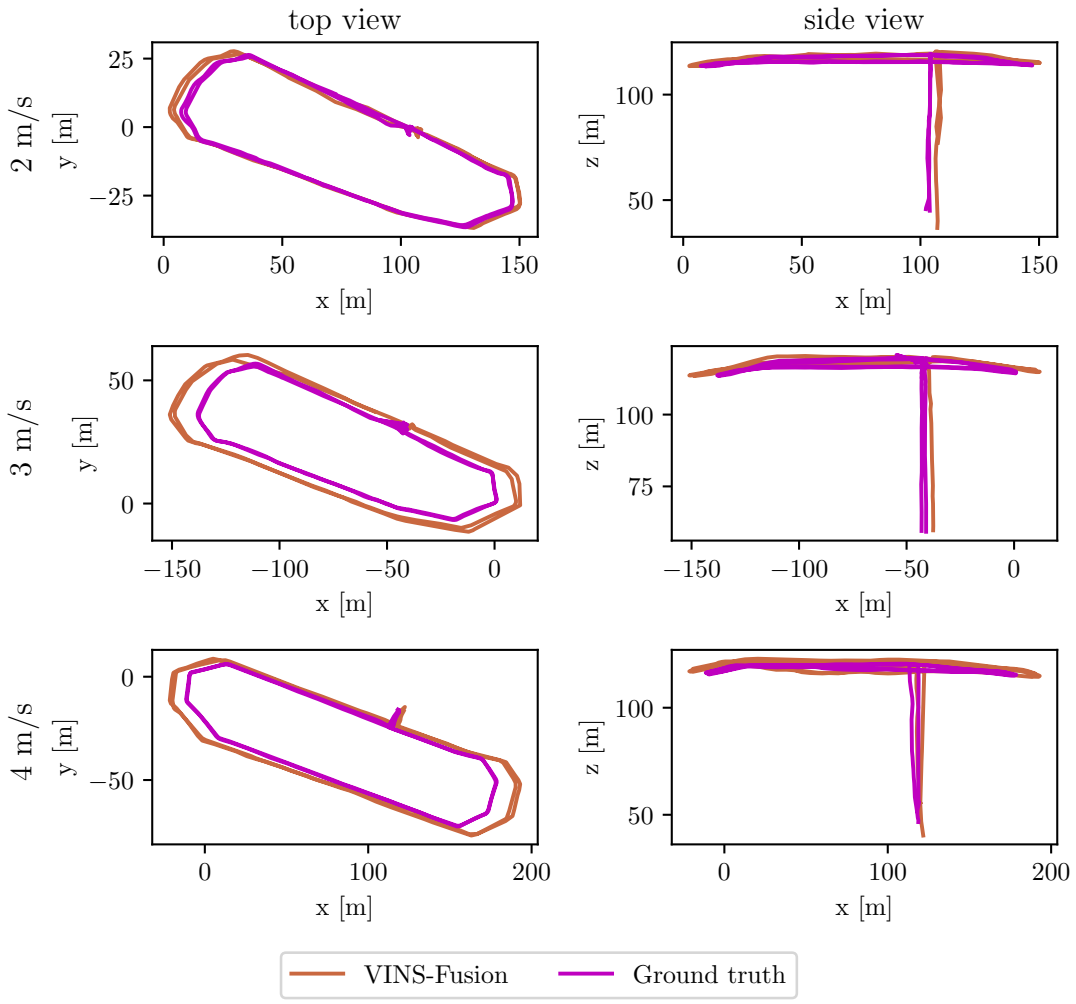
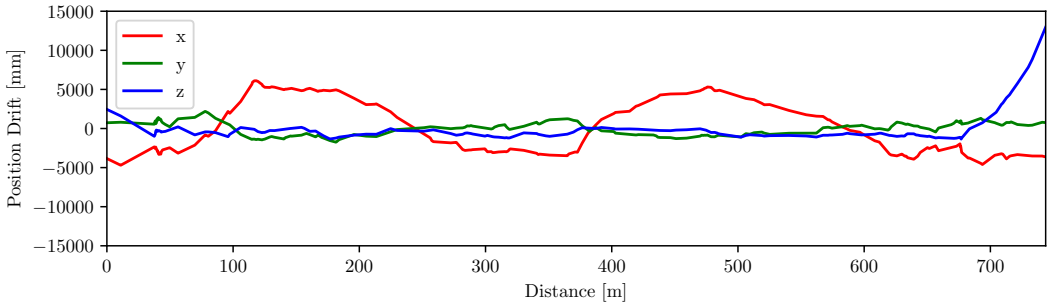
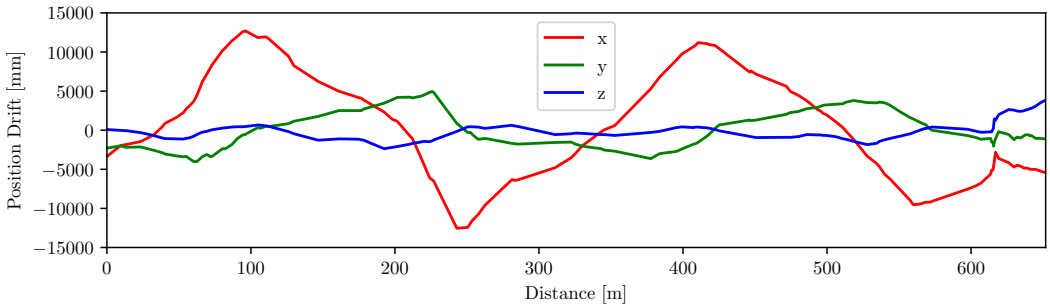


Figure 31: Trajectory estimated by VINS-Fusion plotted along with the ground truth at an altitude of 80 m.

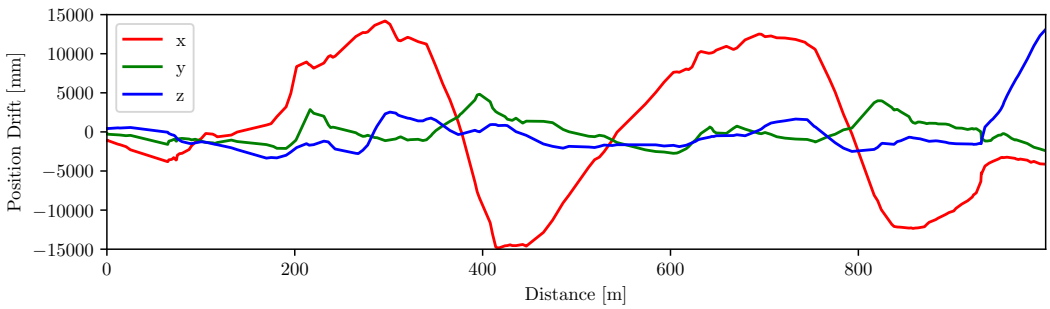
between the flight speed and the drift in estimation, which in turn affect the ATE. The position drift values of estimations for all speeds are plotted in Figure 32. The slope of drift changes the direction as the drone changes its flying direction, similar to the results of 40 m and 60 m altitudes. Interestingly the slope while the drift value varies from a local maximum to local minimum increases as the speed increases. One image per dataset corresponding to this small stretch of path is shown in Figure 30. More than half of the image contains trees which have similar features across the frame. For the speed 2 m/s, the drift is relatively low, and the rate of increase of drift in the x axis towards local minimum is less in comparison with the other speeds. This happens due to the presence of the building in the frame, which constitutes to more distinctive features, which accounts for better estimation. Even though these datasets differ only in speed, and the same waypoints were used to define the path, this change or tilt in the field-of-view is attributed to the wind speed while the drone is flown. According to Finnish Meteorological Institute, the wind speeds at the time



(a) at 2 m/s



(b) at 3 m/s



(c) at 4 m/s

Figure 32: translation error of VINS-Fusion estimation at height 80 m at different speeds

of recording the drone data were 10.4 m/s, 7.8 m/s and 5.9 m/s corresponding to flight speeds 2 m/s, 3 m/s and 4 m/s respectively [77]. The control system of the drone adjusts its orientation to minimise the wind force to keep it in the predefined path. Wind speeds during the data collection are tabulated in Appendix D.

Flying altitude – 100m

Figure 33 plots the results of comparison of estimations of VINS-Fusion at various speeds for the flight altitude of 100 m. Similar to the 80 m dataset, the illumination of images is different in all three speeds. Sample images are shown in Figure 34. As shown in Table 6, ATE values are almost similar and above 12 m for speeds 2 m/s and 3 m/s, and that of 4 m/s is around 10 m, which are large to use in localisation problems. For the speed 4 m/s, as seen in Figure 33 and 35c, the drift is increases more

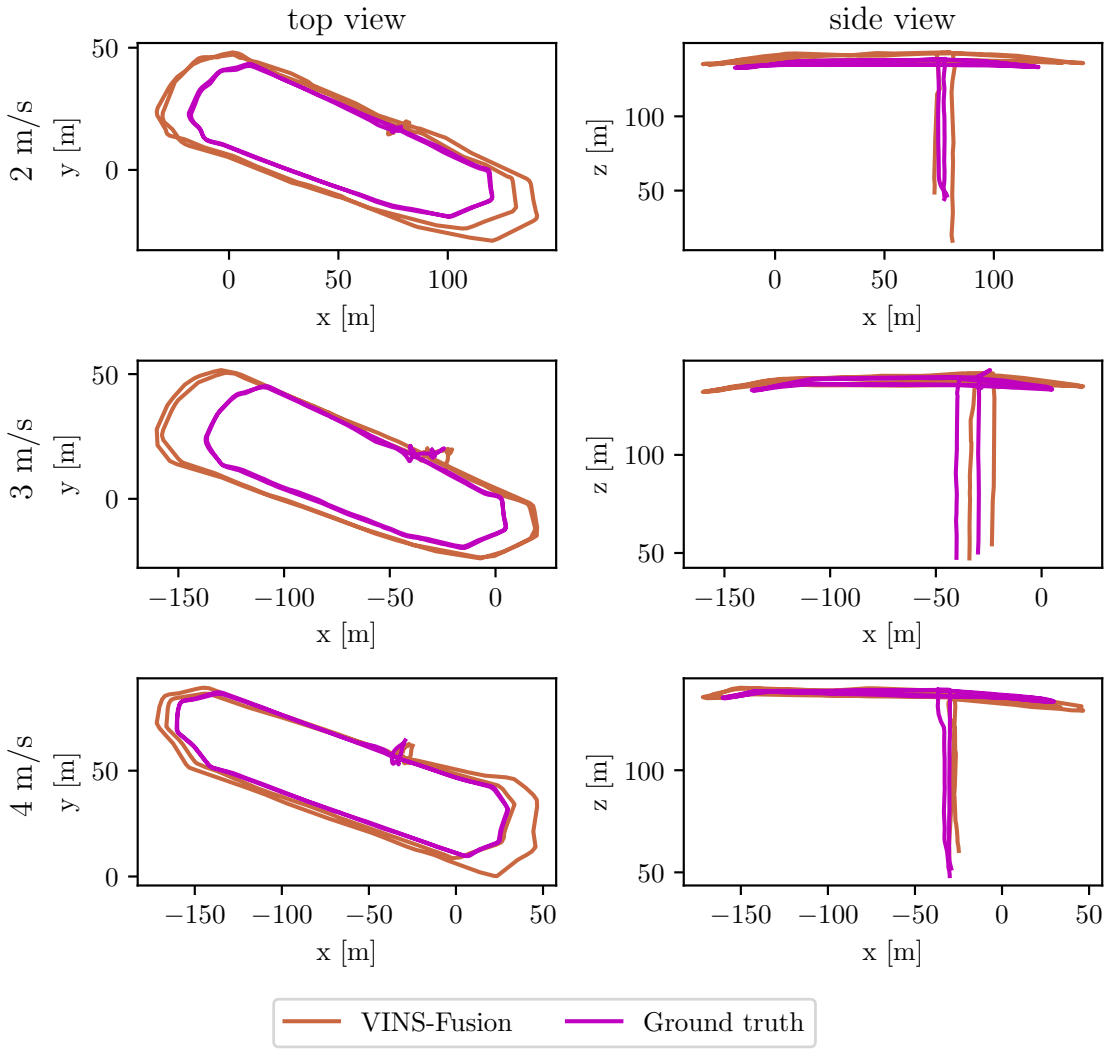
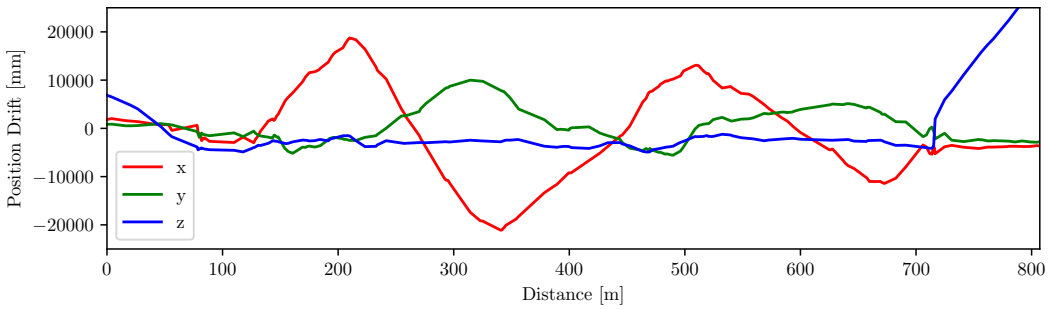


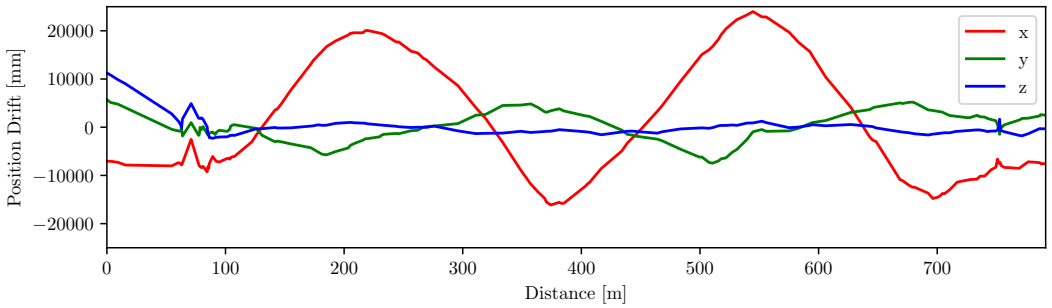
Figure 33: Trajectory estimated by VINS-Fusion plotted along with the ground truth at an altitude of 100 m.

(a) 2 m/s dataset (b) 3 m/s dataset (c) 4 m/s dataset

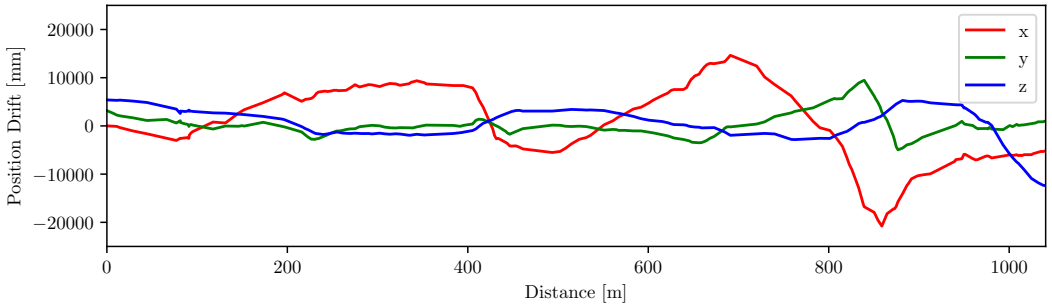
Figure 34: Difference in illumination of images in 100 m datasets



(a) at 2 m/s



(b) at 3 m/s



(c) at 4 m/s

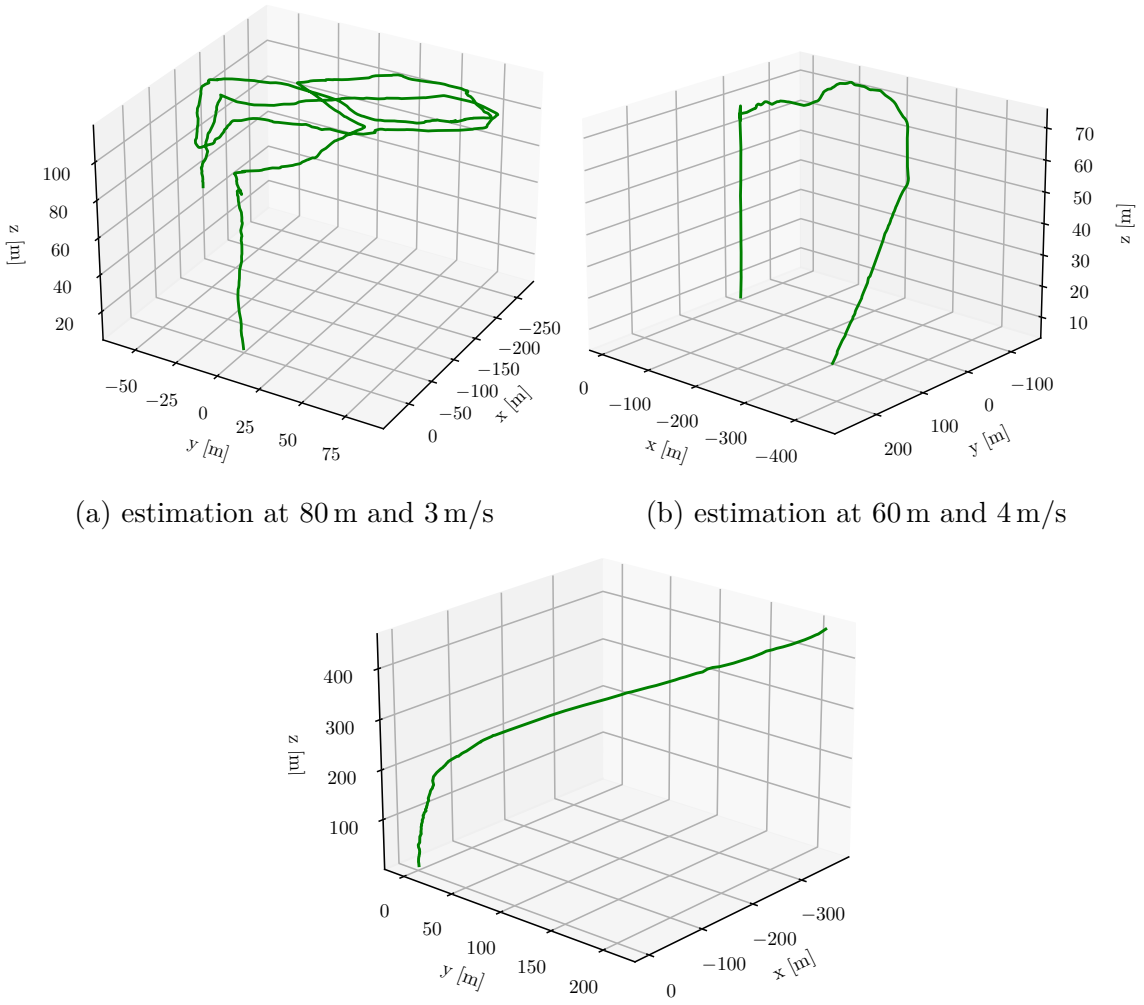
Figure 35: translation error of VINS-Fusion estimation at height 100 m at different speeds

towards the end of the trajectory in comparison with the other parts. In this part, the images captured by the cameras contain mostly trees, and due to poor illumination, there are fewer distinctive features for triangulation and estimation. Faster motion also degrades the tracking. These result in incorrect scale determination and thus increases drift. In general, the baseline-to-depth ratio is small for the developed stereo setup, with a baseline of 30 cm, which degrades the scale observation. This also can be attributed to the poor estimation results.

Relative rotation error for 2 m/s, however, is around 1.5°, which is in range with the value corresponding to the same speed in the 80 m dataset. The same metric is over 4° in the case of 3 m/s and 4 m/s, which can be attributed to poor exposure of images in these datasets.

6.2 Area 2: open field

Table 4 in Section 5.2 details the data collected over the field. Each flight path differs only in the altitude and flight speed, so the top view of these paths would be similar. Figure 36 plots the paths estimated by VINS-Fusion at different flight settings. The features tracked at height 80 m and 100 m are shown in Figure 37.



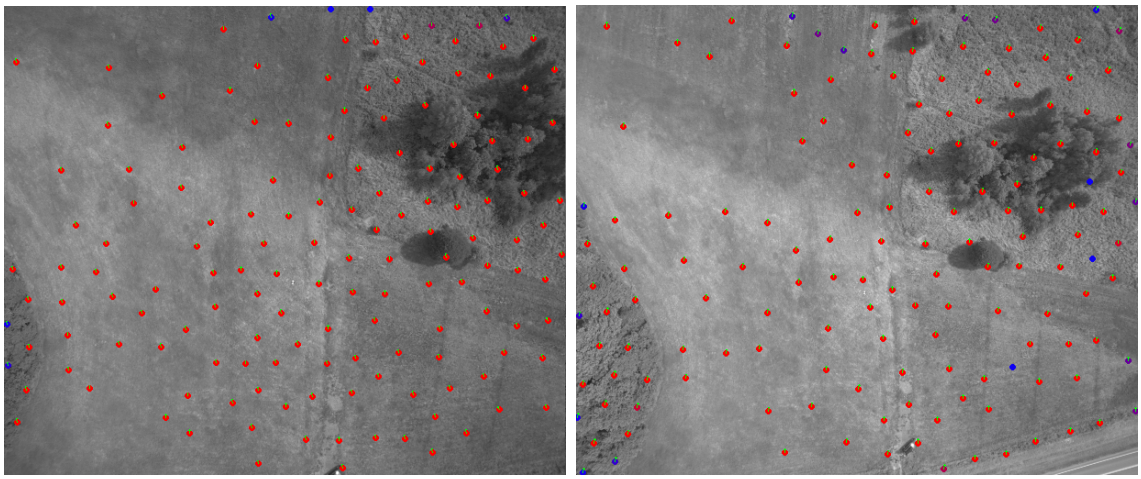
(a) estimation at 80 m and 3 m/s

(b) estimation at 60 m and 4 m/s

(c) estimation at 100 m and 3 m/s

Figure 36: VINS-Fusion path estimations at different flight settings of the field dataset

Flights with lower speeds seem to estimate the path similar to the ground truth as shown in Figure 36a. But as the height increases, the scale error increases and the path drifts quickly from the beginning of the estimation, as seen in Figure 36c for the height 100 m. At height 60 m, the path estimation follows the ground truth initially, but the visual tracking losses at a sharp turn, as the flight speed is 4 m/s, as shown in Figure 36b. Since most of the estimation results do not match with the ground truth, they are not evaluated against the ground truth.



(a) features at 80 m and 3 m/s (b) features at 100 m and 3 m/s

Figure 37: Features detected in VINS-Fusion estimation for the field dataset

This chapter presented the results obtained from the comparison tool and explained some of the reasons behind them. As FLVIS and ORB-SLAM3 were not able to provide estimations similar to the ground truth, only the estimations of VINS-Fusion were analysed in detail. There is no common trend observed from the result, and this is due to the varying external conditions during the data collection. Also there are some limitations associated with the platform which makes the estimation unreliable in some conditions. These points are discussed in the next chapter.

7 Discussion

This thesis has detailed three VIO/VISLAM algorithms, collected visual and intertial data with the developed stereo camera and IMU setup, estimated the trajectory of a drone with the VIO/VISLAM algorithms using the collected data, and compared the estimations with the ground truth to analyse these algorithms quantitatively. The data was collected from two different areas that differ in their aerial views and textures. The first area contains buildings, trees, cars and roads, and the second area is an open field with uniform grass cover. Algorithms were able to estimate trajectories with the data from the first area, while they failed for most of the datasets from the field. Due to this, comparison with the ground truth was performed only for the first area. Findings from the experiments and its results are discussed here.

Performance of VIO/VISLAM algorithms

Based on the comparison plots presented in Chapter 6, and the Table 6, VINS-Fusion estimated the path more accurately than FLVIS and ORB-SLAM3 in most of the cases. VINS-Fusion estimation at height 60 m had the lowest RMSE of absolute translation error and relative yaw error out of all the cases. The RMSE of ATE was 2.186 m at 4 m/s and the RMSE of relative yaw error was 0.862° at 3 m/s, for a trajectory of length 800 m. Even though the error values of FLVIS and ORB-SLAM3 were relatively low for the 40 m dataset, they estimated the path poorly at higher altitudes. ORB features were tracked in FLVIS and ORB-SLAM3 and they were close to each other and they did not cover the entire image, whereas corner features in VINS-Fusion were distributed more evenly across the frame. Also, as the height increased, features of FLVIS and ORB-SLAM3 were grouped together more closely where distinctive objects were present. The quality of BA is improved when the features are well-distributed in the image [78]. This might be one of the reasons for ORB-SLAM3 and FLVIS to perform poorly after 40 m height. Only the visual data was used with ORB-SLAM3 while estimating the path in this thesis due to the challenges with incorporating inertial data with ORB-SLAM3. The visual-inertial mode of ORB-SLAM3 would have resulted in better estimations as seen in [20]. A better parameter tuning of algorithms, or a different sensor configuration could improve the estimations with these algorithms.

The estimations were made offline after collecting the data. It was observed that VINS-Fusion and FLVIS processed the input data in almost real-time, while ORB-SLAM3 required more time to process the inputs, even without the inertial data. The online estimation outputs poses in real-time, and it was tested with handheld datasets during the initial stages of this thesis. Algorithms can be run along with the data collection nodes to obtain an online estimation of the pose of the drone. These estimations can also be compared with the ground truth to perform evaluations in real-world scenarios. Apart from better estimation results, VINS-Fusion has options to change the sensor configuration during the estimation process. For example, during an online estimation, one of the cameras can be removed from the estimation pipeline to perform a monocular visual-inertial estimation. This might improve results

at higher altitudes where the performance of stereo-visual-inertial configuration is decreased due to possible reduction in accuracy of estimation due to the reduced baseline-to-depth ratio. VINS-Fusion is also designed to incorporate multiple local and global sensors with the existing system. Among the three algorithms, VINS-Fusion performed better and these results are sufficient to proceed for further improvements and research for localisation and autonomous navigation.

Flying altitude and speed

From the obtained results, a consistent trend for drift, or scale error, was not observed with any of the changing flight configurations. But as the flying altitude was increased, the drift also increased. Especially when the drone flew at heights 80 m and 100 m, there was a clear distinction between the estimated path and the ground truth. This drift, or scale error, occurred due to the small baseline-to-depth ratio used in this study. The triangulation of feature points from a single pair of stereo images from a small baseline setup is highly error prone at high altitudes [79]. Thus the scale is weakly observable at high altitudes with the current setup. A monocular camera with IMU would perform better at higher altitudes [7].

A consistent behaviour in estimation was not found in relation with the flying speed. The drift reduced as the speed increased for the height 60 m whereas it increased for height 80 m, but this cannot be fully associated with the change in speed, but also to the objects visible in the scene. One constant observation was the increase in the rotation error when the drone changed its direction. Since the path and speed of the drone was predefined, the drone did not reduce its speed while it rotated. This resulted in poor estimation as rotation without translation is not desired while tracking visual features. Reducing the speed of the drone in such scenarios would improve the estimation quality.

The scene captured by the cameras also affected the estimation. The field dataset, as described in Section 6.2, showed that VINS-Fusion was able to estimate the path more or less similar to the ground truth for the height 80 m, while it failed for 60 m. This was due to the difference in the scene observed in both cases. As the drone flew above the field at a lower altitude, the images mostly had a uniform grass field. In contrast, for a higher altitude, the field-of-view was increased which resulted in having more distinctive objects, such as trees, in the frame. Similarly, in the datasets near the building, when the drone flew above the trees to have only trees in the frame, the drift increased. But when stable and well defined objects, such as cars or the building, were present in the frame, the estimations were better, as seen in the 80 m dataset near the building. Presence of strong wind in the 80 m, 2 m/s dataset caused the drone to tilt, which resulted in seeing more distinct features in the frame and to estimate the path more accurately than other speeds for the same height. Hence, the presence of distinctive objects in the field-of-view increased the estimation accuracy.

Developed platform

The data collection was performed using a custom made stereo visual-inertial sensor system. This system was developed such that it is easily customisable. Two monocular cameras and an IMU were mounted on a frame that can be easily attached to the drone. Hardware and software synchronisation were enabled to retrieve synchronous data from these sensors. The system produced best results at a flight altitude of 60 m. However, the system was not very suitable at high altitudes since the stereo baseline was only 30 cm, which resulted in higher depth error at higher altitudes. But there are provisions to increase the baseline till 50 cm and to test the algorithms with stereo images from a longer baseline setup. This setup enables further testing and developments in the future. Currently the cameras are facing downwards. A tilted or forwarding facing configuration can be adopted depending on the application. Fast maneuvering of drones through the forest, where the cameras face forward, is presented in [80].

The stereo-camera and IMU setup was calibrated as explained in the Section 4.3. The calibration target was kept at a distance of 4 m while the calibration was performed. But during the data collection, the cameras were farther away from the objects captured in the frame. Even though the reprojection errors for the present camera calibration are small, it is better to calibrate the sensors in conditions similar to the actual data collection conditions [81].

The relative rotation error values were affected by the illumination of the images. When the images in the dataset were highly underexposed or overexposed, the rotation error increased, while the estimations from normally exposed images resulted in low rotation error. The exposure time was set to a constant value before each flight as mentioned in the Chapter 4. This value was chosen intuitively based on the lighting conditions. Adapting a proper technique to obtain well illuminated images will help to improve the estimation quality.

As the altitude of the flight increases, the GSD increases. Based on the calculations and the values given in Appendix C, the GSD increased from 8.9 cm/px to 22.26 cm/px as the height increased from 40 m to 100 m. This might have resulted in not detecting some key features that corresponded to objects that were smaller than the GSD for each height. This can be one of the reasons behind the larger errors in high altitude flight estimations. Raw images from the camera were downsampled before the estimation process as mentioned in Section 4.2, which increased the GSD values. Reducing the downsampling factor or using the full image would improve the estimation at the cost of high resource usage.

Cameras and IMU were mounted on an aluminium frame which was rigidly attached to the drone. During the flight, this setup experienced strong forces due to the gravity and the motion of the drone, along with the effects of wind flow. These forces imparted vibration on the frame and made the sensor data noisy, especially the raw IMU data. The vibration will increase if the stereo baseline is further increased. Mounting the sensors on a gimbal would reduce the vibration effect.

Based on the results and observations, VINS-Fusion performed better among the

three algorithms with the data collected using the custom sensor suite. The flight parameters, such as speed and height, as well as external environment conditions, such as illumination and wind, affect the estimation results. Besides, various parameters of the algorithms also affect the estimation. As discussed previously, there exist many possible ways to further improve the system to obtain a better trajectory estimation using VIO/VISLAM algorithms. Further possibilities of interest include the dense reconstruction of the object as well as semantic segmentation using the colour information from the cameras.

8 Conclusion

In this thesis, a sensor suite to collect synchronised stereo visual and inertial data was developed and mounted on a drone to collect visual and inertial data. This data was processed to estimate the trajectory of the drone using three state-of-the-art VIO/VISLAM algorithms – VINS-Fusion, ORB-SLAM3 and FLVIS. A quantitative evaluation of these estimations are performed using a publicly available trajectory evaluation tool. The results indicated that VINS-Fusion outperformed other algorithms with the collected datasets. In a detailed analysis of VINS-Fusion estimations, it was found that the flight parameters, configuration of the sensor suite, and the environmental conditions affected the estimation quality. Further improvements and development for enabling autonomous flights can be proceeded by using the estimations from VINS-Fusion.

The results showed that the VINS-Fusion estimates the path accurately, given the drone flies close to the object, the captured image contains distinctive features, and the image is well illuminated. The least RMSE value of absolute translation and relative yaw errors for the VINS-Fusion estimations are 2.186 m and 0.862°, respectively for a 800 m long trajectory, at a flying altitude of 60 m. With the availability of an initial global reference value and the above mentioned assumptions, VINS-Fusion estimation can be used to get a global trajectory of the drone. This estimation can be used as one part in autonomous operation of a drone.

This work evaluated several factors influencing the performance of the estimation algorithms. However, it was not possible to evaluate impacts of all the factors quantitatively. Further experiments and analyses are needed to understand the effects of sensor configuration, algorithm parameters, external factors, etc. Changes in the sensor system, such as wider baseline or monocular setup at higher altitudes, improving the exposure settings of images and tuning the algorithm parameters can improve the estimation results. Also, for an effective comparison at different flight altitudes and speeds, the data has to be collected in similar environmental conditions. Online estimations can also be performed to evaluate the algorithms and to enable autonomous operation. All of these possibilities can be studied with the developed sensor suite to identify a better visual-inertial localisation system to carry out autonomous drone flights.

References

[1] InfiniDome. Enabling Drone BVLOS Operations for Defense and Critical Missions. [Online]. Available: https://uasweekly.com/2021/05/13/infinidomes-enabling-drone-bvlos-operations-for-defense-critical-missions-webinar/

[2] S. Poddar, R. Kottath, and V. Karar, *Motion Estimation Made Easy: Evolution and Trends in Visual Odometry.* Cham: Springer International Publishing, 2019, pp. 305–331. [Online]. Available: https://doi.org/10.1007/978-3-030-03000-1_13

[3] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft mav," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4974–4981.

[4] L. Reddy Cenkeramaddi, J. Bhatia, A. Jha, S. Kumar Vishkarma, and J. Soumya, "A survey on sensors for autonomous systems," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 1182–1187.

[5] V. Uzodinma and U. Nwafor, "Degradation of gnss accuracy by multipath and tree canopy distortions in a school environment," *Asian Journal of Applied Sciences*, vol. 6, 09 2018.

[6] N. L. S. of Finland. New steps in nordic collaboration against gnss interference. [Online]. Available: https://www.maanmittauslaitos.fi/en/topical_issues/new-steps-nordic-collaboration-against-gnss-interference

[7] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[8] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and slam for medium and high altitude flying uavs," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, p. 137–161, 2008.

[9] H. Romero, S. Salazar, O. Santos, and R. Lozano, "Visual odometry for autonomous outdoor flight of a quadrotor uav," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2013, pp. 678–684.

[10] M. Warren, P. Corke, and B. Upcroft, "Long-range stereo visual odometry for extended altitude flight of unmanned aerial vehicles," *The International Journal of Robotics Research*, vol. 35, no. 4, p. 381–403, 2015.

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

[12] 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, 2017.

[13] R. Wang, M. Schworer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.

[14] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, p. 27–42, 2017.

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

[16] 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.

[17] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[18] 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.

[19] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.

[20] 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*, 2021.

[21] S. Chen, C.-Y. Wen, Y. Zou, and W. Chen, "Stereo visual inertial pose estimation based on feedforward-feedback loops," *arXiv preprint arXiv:2007.02250*, 2020.

[22] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2502–2509.

[23] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, 2018.

[24] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3681–3688.

[25] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[26] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[27] C. Harris, M. Stephens *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50, 1988, pp. 147–151.

[28] C. Tomasi and T. Kanade, "Detection and tracking of point," *International journal of computer vision*, vol. 9, pp. 137–154, 1991.

[29] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[30] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[32] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of computer vision*, vol. 37, no. 2, pp. 151–172, 2000.

[33] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[34] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[35] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[37] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2004.

[38] 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.

[39] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[40] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[41] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[42] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[43] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[44] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[45] 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.

[46] D. Scaramuzza and Z. Zhang, *Aerial Robots, Visual-Inertial Odometry of*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1007/978-3-642-41610-1_71-1

[47] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," *arXiv preprint arXiv:1901.03638*, 2019.

[48] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[49] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[50] 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.

[51] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE international conference on rehabilitation robotics*. IEEE, 2011, pp. 1–7.

[52] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[53] Intel Corporation. IntelRealSense Tracking Camera. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2019/09/Intel_RealSense_Tracking_Camera_Datasheet_Rev004_release.pdf

[54] Nerian Vision GmbH. Karmin3 Stereo Camera User Manual. [Online]. Available: https://nerian.com/nerian-content/downloads/manuals/karmin3/karmin3_manual_v1_1.pdf

[55] ——. ScenScan / SceneScan Pro User Manual. [Online]. Available: https://nerian.com/nerian-content/downloads/manuals/scenescan/scenescan_manual_v1_14.pdf

[56] ROS. About ROS. [Online]. Available: https://www.ros.org/about-ros/

[57] Basler. acA2440-75uc. [Online]. Available: https://docs.baslerweb.com/aca2440-75uc

[58] Xsens. MTi 600-series User Manual. [Online]. Available: https://mtidocs.xsens.com/mti-600-series-user-manual

[59] GIGA-BYTE Technology Co. Ltd. GB-BSi5H-6200-B2-IW (rev. 1.0). [Online]. Available: https://www.gigabyte.com/Mini-PcSystem/GB-BSi5H-6200-B2-IW-rev-10#ov

[60] FUJIFILM Corporation. HF-XA-5M Series. [Online]. Available: https://www.fujifilm.com/us/en/business/optical-devices/optical-devices/machine-vision-lens/hf-xa-5m-series#HF01

[61] Intel Corporation. USB 3.0* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices. [Online]. Available: https://www.usb.org/sites/default/files/327216.pdf

[62] H. Koivula, A. Laaksonen, S. Lahtinen, J. Kuokkanen, S. Marila, F. H. Koivula, A. Laaksonen, S. Lahtinen, J. Kuokkanen, and S. Marila, "Finnish permanent GNSS network, FinnRef," *FIG Working Week 2017*, 2017.

[63] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1280–1286.

[64] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4304–4311.

[65] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-696, Aug. 2007. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

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

[67] SPH Engineering, "Ground Station Software | UgCS PC Mission Planing." [Online]. Available: https://www.ugcs.com/

[68] Agisoft, "Agisoft metashape user manual," 2021, professional Edition, Version 1.7.

[69] GeoScan, "Metashape professional." [Online]. Available: https://www.geoscan.aero/en/software/agisoft/metashape_pro

[70] I. Elkhrachy, "Accuracy assessment of low-cost unmanned aerial vehicle (uav) photogrammetry," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5579–5590, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016821002544

[71] Topcon Positioning Systems, Inc., "HiPer HR." [Online]. Available: https://www.topconpositioning.com/support/products/hiper-hr

[72] ——, "FC-5000 Field Controller." [Online]. Available: https://www.topconpositioning.com/support/products/fc-5000-field-controller

[73] 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.

[74] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[75] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.

[76] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[77] Finnish Meteorological Institute. Download observations. [Online]. Available: https://en.ilmatieteenlaitos.fi/download-observations

[78] V. Mousavi, M. Varshosaz, and F. Remondino, "Evaluating tie points distribution, multiplicity and number on the accuracy of uav photogrammetry blocks," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, pp. 39–46, 2021.

[79] M. Warren and B. Upcroft, "High altitude stereo visual odometry," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[80] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abg5810

[81] M. Pérez, F. Agüera, and F. Carvajal, "Digital camera calibration using images taken from an unmanned aerial vehicle," *International archives of the photogrammetry, remote sensing and spatial information sciences*, vol. 38, no. 1/C22, 2011.

[82] Vision Aerial, "What is Ground Sample Distance (GSD)?" [Online]. Available: https://visionaerial.com/what-is-ground-sample-distance/

A Calibration Results

As detailed in the Section 4.3, the stereo cameras and the IMU are calibrated jointly using the Kalibr tool. IMU biases are calculated using the `imu_utils` tool and the stereo camera calibration is performed using Kalibr.

Cameras are modeled as pinhole cameras and the radial-tangential distortion model is assumed. Camera parameters, reprojection errors and the stereo baseline details are tabulated in Table A1.

Table A1: Stereo camera calibration results

Left camera	Parameters	Standard deviation
Distortion [k_1, k_2, p_1 p_2]	[−0.1762089, 0.08539239, 0.00024623, 0.00032291]	± [0.00246955, 0.00616063, 0.00023633, 0.00025822]
Projection [f_x, f_y, c_x c_y]	[448.29308778, 447.99789994, 301.68106405, 242.76373277]	± [0.24373853, 0.24317746, 0.57081775, 0.57882664]
Reprojection error [x, y]	[−0.000009, 0.000002]	± [0.083676, 0.071372]

Right camera	Parameters	Standard deviation
Distortion [k_1, k_2, p_1 p_2]	[−0.17142204, 0.07427103, 0.00013522, 0.00007182]	± [0.002164, 0.00336623, 0.00019319, 0.00028766]
Projection [f_x, f_y, c_x c_y]	[446.38011636, 445.88719477, 306.68683921, 246.39725595]	± [0.24117874, 0.24583356, 0.61003624, 0.63008214]
Reprojection error [x, y]	[−0.000013, 0.000005]	± [0.099263, 0.094998]

Baseline	Parameters	Standard deviation
Rotation [q_x, q_y, q_z q_w]	[−0.001446, 0.00080468, −0.00115645, 0.99999796]	± [0.00128989, 0.00166449, 0.00014686]
Translation [t_x, t_y, t_z]	[−0.30018969, 0.00026494, −0.00454052]	± [0.00034904, 0.00030125, 0.00127581]

Table A2: IMU biases

Bias	Value
accelerometer noise density	0.00912504793531317
accelerometer random walk	0.0001305951086175891
gyroscope noise density	0.0019425812268625707
gyroscope random walk	3.9570963198826466e-05

The accelerometer and gyroscope biases obtained from `imu_utils` are tabulated in Table A2.

The joint calibration of IMU and the stereo camera gives the geometric transformation and temporal difference between each sensor. The transformation between left camera and the IMU,

$$T_{i\_lc} = \begin{bmatrix} 0.00628532 & 0.99997808 & -0.00208084 & 0.00038067 \\ 0.99992558 & -0.0062632 & 0.01046937 & 0.00038001 \\ 0.01045611 & -0.00214649 & -0.99994303 & -0.00041705 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

Transformation between right camera and the IMU is given by

$$T_{i\_rc} = \begin{bmatrix} 0.00397347 & 0.99999177 & 0.00081937 & 0.00131225 \\ 0.9999193 & -0.00398307 & 0.0120638 & 0.3006013 \\ 0.01206696 & 0.00077137 & -0.99992689 & -0.00133506 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

Temporal differences between the IMU, and left and right cameras are given below.

$$t_{i\_lc} = 0.000185936118098\text{s} \quad t_{i\_rc} = 0.000192639407371\text{s}$$

B Estimation plots

The estimation results of FLVIS and ORB-SLAM3 are not similar to the ground truth path for altitudes above 40 m. These results, along with VINS-Fusion estimation and the ground truth are plotted here. Figure B1 plots the estimations for height 60 m. Figure B2 and B3 plot the results for 80 m and 100 m altitudes respectively.

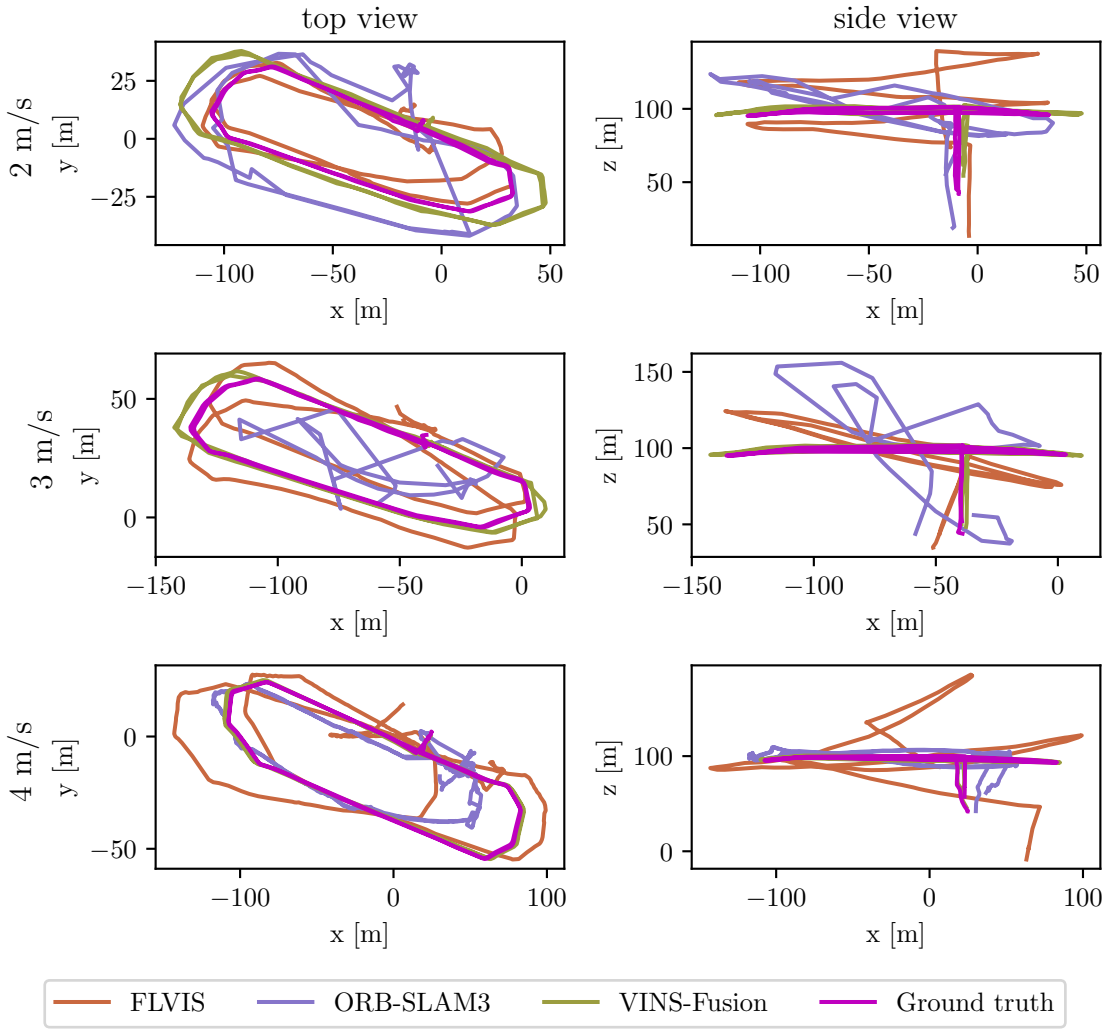


Figure B1: Estimation results along with the ground truth at an altitude of 60 m.

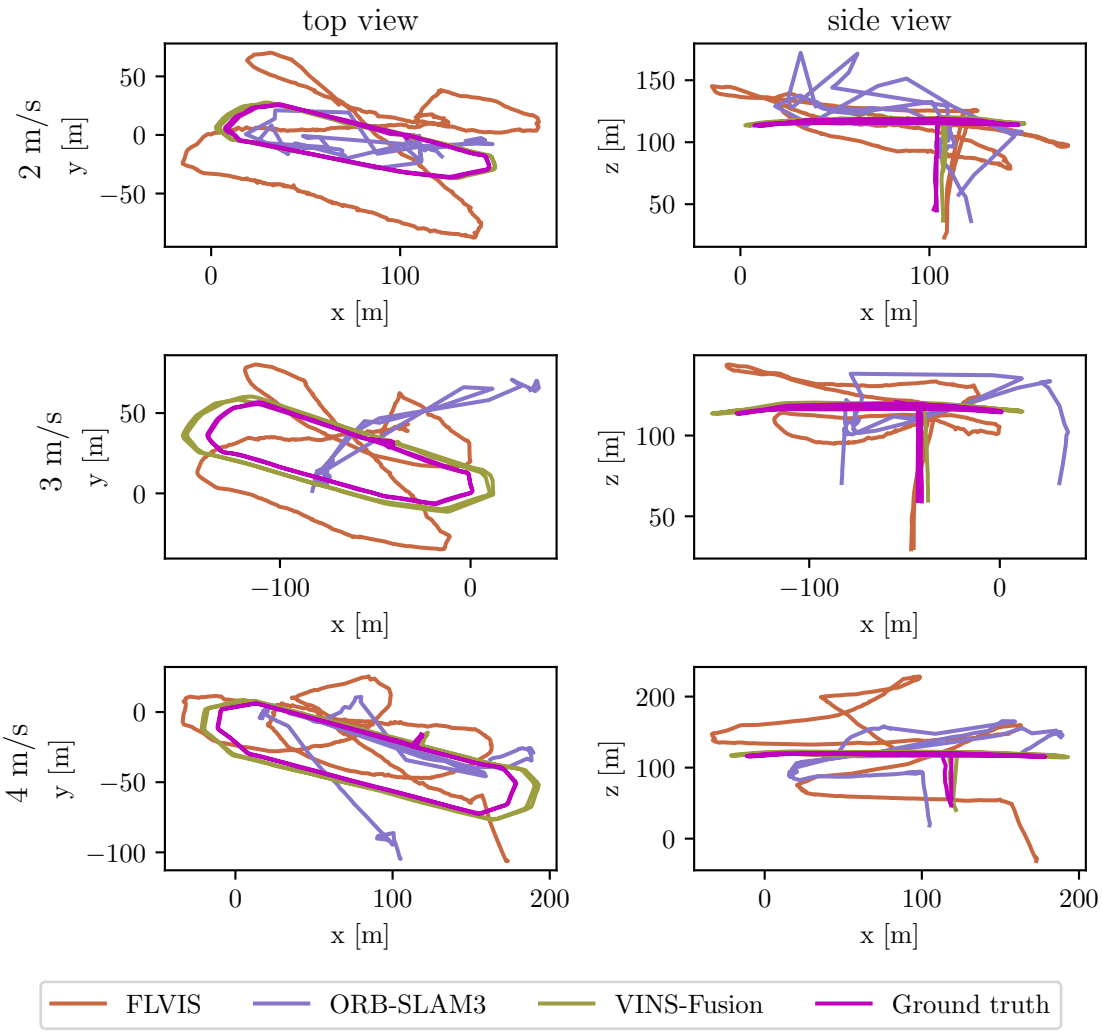


Figure B2: Estimation results along with the ground truth at an altitude of 80 m.

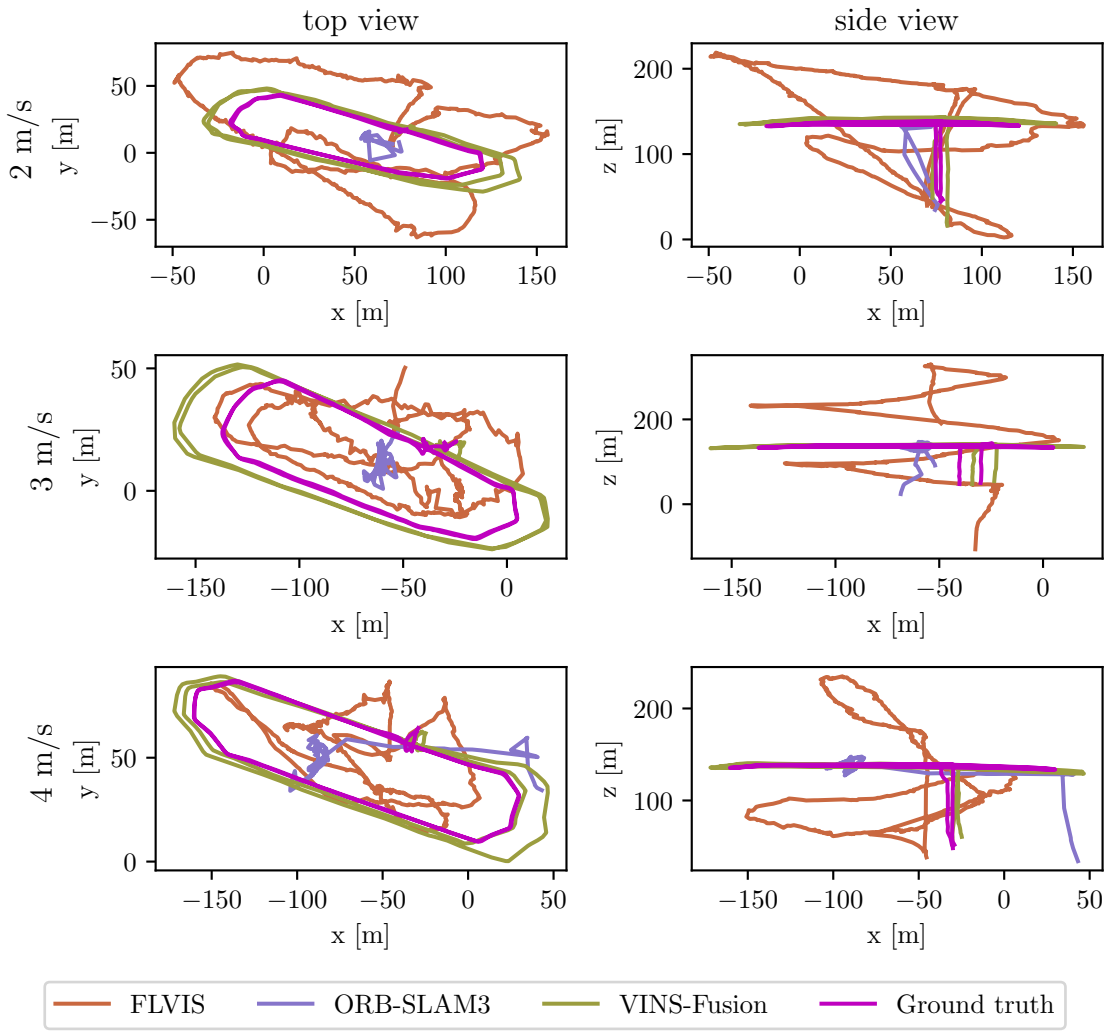


Figure B3: Estimation results along with the ground truth at an altitude of 100 m.

C Ground sample distances

The ground sample distance (GSD) is defined as the distance between the centers of two adjacent pixels measured on ground. This metric is particularly important in mapping and surveying applications [82]. The GSD is calculated based on the flying altitude and camera properties, such as sensor size, resolution and focal length. Based on the parameters listed in the Table C1, GSD is calculated for different resolution and flight altitude. The results are tabulated in Table C2.

Table C1: Parameters for GSD calculation

Parameter	Value
sensor width	8.4 mm
sensor height	7.1 mm
image size	2448 × 2048 px
focal length	6.23 mm

Raw images from the camera are subsampled to reduce their size in order to decrease the processing time. GSDs corresponding to these reduced image sizes are also given in Table C2. Images of 612 × 512 size are used in this thesis.

Table C2: GSD in cm/px for various flying altitudes and image size

Size \ Height	40 m	60 m	80 m	100 m
2448 × 2048 px	2.23	3.34	4.45	5.56
1224 × 1024 px	4.45	6.68	8.9	11.13
612 × 512 px	8.9	13.36	17.81	22.26

D Wind speeds

The wind speeds during the data collection are tabulated in Table D1. This data was collected from the publically available historical data of Finnish Meteorological Institute's website[16]. This data was collected from the Kirkkonumi Mäkiluoto observation center, which is the nearest one to the data collection location.

Table D1: Wind speed data (in m/s) from collected from FMI

Speed \ Height	40 m	60 m	80 m	100 m
2 m/s	4.2	3.1	10.4	11
3 m/s	7.9	4.4	7.8	7.8
4 m/s	10.4	5.6	5.9	5.7

[16] https://en.ilmatieteenlaitos.fi/download-observations