3-26-2020

Comparison of Visual Simultaneous Localization and Mapping Methods for Fixed-Wing Aircraft Using SLAMBench2

Patrick R. Latcham

COMPARISON OF VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING METHODS FOR FIXED-WING AIRCRAFT USING SLAMBENCH2

THESIS

Patrick R. Latcham, 2d Lt, USAF

AFIT-ENG-MS-20-M-034

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-034

COMPARISON OF VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING METHODS FOR FIXED-WING AIRCRAFT USING SLAMBENCH2

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Patrick R. Latcham, B.S.E.E.

2d Lt, USAF

March 19, 2020

AFIT-ENG-MS-20-M-034

COMPARISON OF VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING METHODS FOR FIXED-WING AIRCRAFT USING SLAMBENCH2

THESIS

Patrick R. Latcham, B.S.E.E.
2d Lt, USAF

Committee Membership:

Clark N. Taylor, Ph.D
Chair

Maj Aaron J. Canciani, Ph.D
Member

Robert C. Leishman, Ph.D
Member

Abstract

Visual Simultaneous Localization and Mapping (VSLAM) algorithms have evolved rapidly in the last few years, however there has been little research evaluating current algorithm's effectiveness and limitations when applied to tracking the position of a fixed-wing aerial vehicle. This research looks to evaluate current monocular VSLAM algorithms' performance on aerial vehicle datasets using the SLAMBench2 benchmarking suite. The algorithms tested are MonoSLAM, PTAM, OKVIS, LSDSLAM, ORB-SLAM2, and SVO, all of which are built into the SLAMBench2 software. The algorithms' performance is evaluated using simulated datasets generated in the AftrBurner Engine. The datasets were designed to test the quality of each algorithm's tracking solution, as well as finding any dependence on camera field of view (FOV), aircraft altitude, bank angle, and bank rate.

Through these tests, it was found that LSDSLAM, ORB-SLAM2, and SVO are good candidates for further research, with MonoSLAM, PTAM, and OKVIS failing to track any datasets. All algorithms were found to fail when the capturing camera had a horizontal FOV of less than 60 degrees, with peak performance occurring at a FOV of 75 degrees or above. LSDSLAM was found to fail when the aircraft bank angle exceeded half of the camera's FOV, and SVO was found to fail below 450 meters altitude. The simulations were also tested against a comparable real world dataset, with agreeable results, although the FOV of the real world dataset was too small to be a particularly useful test. Further research is required to determine the applicability of these results to the real world, as well as fuse VSLAM algorithms with other sensors and solutions to form a more robust navigation solution.

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Clark Taylor, for his guidance, mentorship, and unending patience through this process. Without his input, this work would not have been possible.

I would also like to thank Dr. Robert Leishman and Maj Aaron Canciani, as well as the rest of the AFIT faculty, for their excellent instruction and professional support.

A special thanks to Capt Sarantsev for being available whenever I needed him to make, remake, or re-remake a dataset, even at the latest of hours.

Finally, I am grateful for my family, friends, and especially my wife for supporting me throughout this process.

Patrick R. Latcham

Table of Contents

Page

List of Figures

List of Tables

COMPARISON OF VISUAL SIMULTANEOUS LOCALIZATION AND MAPPING METHODS FOR FIXED-WING AIRCRAFT USING SLAMBENCH2

I. Introduction

1.1 Background

Current aerial vehicle navigation is heavily reliant on the Global Positioning System (GPS), which can be degraded or denied. As such, reliable alternatives to GPS are desirable. Visual Simultaneous Localization and Mapping (VSLAM) algorithms have matured rapidly in the last few years, potentially becoming a viable solution for aerial vehicle navigation. VSLAM algorithms use the images from a camera to measure the camera's change in pose, building an internal map which can be used for localization [1]. VSLAM has been successfully used for ground and quadrotor based navigation [2][3][4][5], but has only recently started to be applied to fixed wing aerial vehicles [6] [7].

1.2 Problem Statement

While researching the viability of using existing Visual Odometry (VO) algorithms on aerial vehicles, it was found that the existing algorithms did not work at all with the datasets used. It is theorized that the algorithms failed due to the small field of view (FOV) of the capturing camera combined with the altitude of the aircraft. This research aims to determine the effect, if any, that the FOV of the capturing camera has on the quality of the navigation solution given by VSLAM algorithms, as well as investigate how this effect compounds with the aircraft's bank angle, bank rate, and

altitude.

1.3 Research Objectives

This work attempts to rigorously determine the effect of the following variables on the quality and robustness of solutions from several state of the art VSLAM algorithms.

- The capturing camera's FOV
- The aircraft's altitude
- The aircraft's bank angle
- The aircraft's bank rate

It also looks to determine any interdependencies between the above variables.

1.4 Approach

Seven implementations of VSLAM algorithms are tested using simulated flight datasets: MonoSLAM [8], PTAM [9], OKVIS [10], LSD-SLAM and its c++ implementation [3], ORB-SLAM2 [2], and SVO [11]. These algorithms are tested with datasets created using the AftrBurner Engine [12]. Several datasets are created, with varying FOV of the capturing camera, altitude of the aircraft, and bank rate. The flight path is designed to incrementally test higher bank angles to determine the quality of the tracking solution. The VSLAM algorithms are applied to the datasets using the SLAMBench2 software suite [13], which gives a consistent environment for testing all algorithms. The estimated trajectories are then aligned and analyzed using the rpg_trajectory_evaluation software from [14].

A simulated flight is also made to match a real-world flight test, with the only difference being the increased FOV of the camera, to determine applicability of the simulations to real-world situations.

1.5 Assumptions/Limitations

The following assumptions/limitations are taken during this research and analysis:

- The camera always points down in the aircraft body reference frame.
- The simulated terrain accurately represents real-world terrain, as parallax effects are minimized at a sufficient altitude.
- The camera is modeled as an ideal pinhole camera with zero distortion.
- The VSLAM algorithms are initialized when the aircraft is flying straight and level.
- This research only looks at the translational solution of the VSLAM algorithms, not the rotational accuracy.
- The algorithms are used as-is with default tuning parameters.

1.6 Contributions

This thesis contributes to the field of fixed-wing aerial vehicle VSLAM a more thorough understanding of the effect that camera FOV and flight characteristics (namely, roll rate and roll angle) have on the quality of VSLAM solutions, including the minimum FOV that can be expected to give accurate results.

1.7 Thesis Overview

This thesis is organized into five chapters. Chapter II gives a brief summary of VSLAM algorithms, introduces the algorithms and programs used in this research, and presents other related research in this field. Chapter III gives a more detailed explanation of the simulated datasets used and describes the methodology with which the VSLAM algorithms are tested against those datasets. The results of the research and subsequent analysis are given in Chapter IV. Chapter V presents a summary of this work as well as suggestions for future research.

II. Background and Literature Review

This chapter provides a brief introduction to the concepts, tools, and algorithms used in this research. It begins with a background on Visual Simultaneous Localization and Mapping (VSLAM) and the model used to describe a camera in section 2.1. Section 2.2 gives backgrounds on each of the monocular Simultaneous Localization and Mapping (SLAM) algorithms used. Section 2.3 follows with a discussion on Visual SLAM testing datasets, including the datasets used in this research. The chapter then gives background on SLAM benchmarking suites in section 2.4, specifically the KITTI Benchmark and SLAMBench 1 and 2. Finally, the chapter ends with a discussion of related research in section 2.5.

2.1 Visual SLAM

VSLAM is a method of using a sequence of images to build a map of the surroundings, while also solving for the location and trajectory of the vehicle within the map. Typically, VSLAM algorithms consist of a front-end Visual Odometry (VO) system which incrementally updates the pose of the camera, and a back-end framework which uses the odometry data to build a map, localize itself within that map, and handles loop closures and additional sensor inputs [1].

VO is the method by which odometry data can be obtained from a sequence of images. It can use monocular, stereo, or monocular-plus-depth data to calculate the camera's current pose relative to its prior pose. Stereo and depth based methods offer the advantage of being able to judge absolute scale, while monocular methods can only give relative scale. In the situation of aerial vehicle navigation, however, the camera is often too far above the ground for stereo or depth based methods to accurately give depth measurements, thus only monocular methods are explored in

this research.

2.1.1 Camera Calibration, Field of View, and Image Width

Core to VSLAM algorithms is the camera itself. VO algorithms most commonly use the pinhole model to calibrate the camera and model the camera's intrinsic parameters [15]. Fig. 1 shows the pinhole model, which assumes that all the light rays go through a single point in the camera.

Using this model, the projection of a point in space p^k onto the image plane I_k is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} K \boldsymbol{p}^k = \frac{1}{z} \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1}$$

where K is the camera calibration matrix representing the camera intrinsics. Specifically, these intrinsics are f_x, f_y, u_0, and v_0, with u_0 and v_0 representing the coordinates of the center of the projection, and f_x and f_y representing the focal length (in pixels) in each direction. The pinhole model also provides intrinsics for distortion, but in this research all distortion is assumed to be zero. Practically, f_x and f_y are taken to be equal, as the camera ideally has the same focal length in each direction, while u_0 and v_0 are taken to be half of the camera resolution width and height, respectively.

The field of view (FOV) of the camera is directly related to the focal lengths by the following equations:

$$fov_x = 2\arctan\frac{w}{2f_x} \qquad fov_y = 2\arctan\frac{h}{2f_y} \tag{2}$$

where w and h are the width and height of the image in pixels. In this research, as the focal lengths are taken to be equal, the vertical FOV of the image is taken

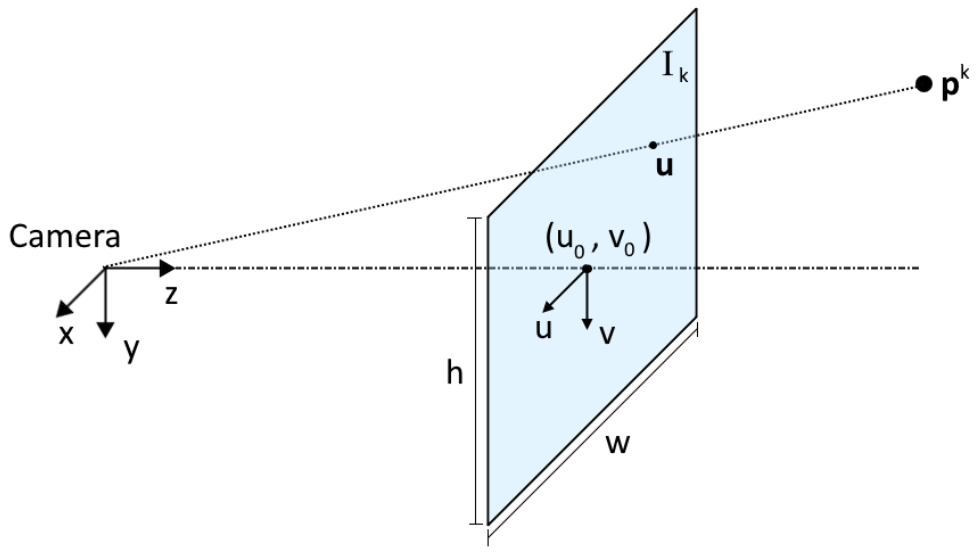


Figure 1: The Pinhole Model

to be dependent on the horizontal FOV. Further discussion about camera's FOV is understood to be referencing the horizontal FOV.

The optical flow of each image, or how much each pixel moves from image to image, is actually controlled by a combination of the horizontal FOV, aircraft altitude, and aircraft speed. To evaluate this combination of variables, we look at a factor called Image Width, measured in meters. This is a measure of how much ground the camera can see during straight and level flight. Image Width can be calculated using the equation:

$$IW = 2(Alt)\tan\frac{fov_x}{2} \tag{3}$$

where IW is the image width in meters, Alt is the altitude of the aircraft in meters, and fov_x is the horizontal FOV of the camera. This is shown graphically in Fig. 2.

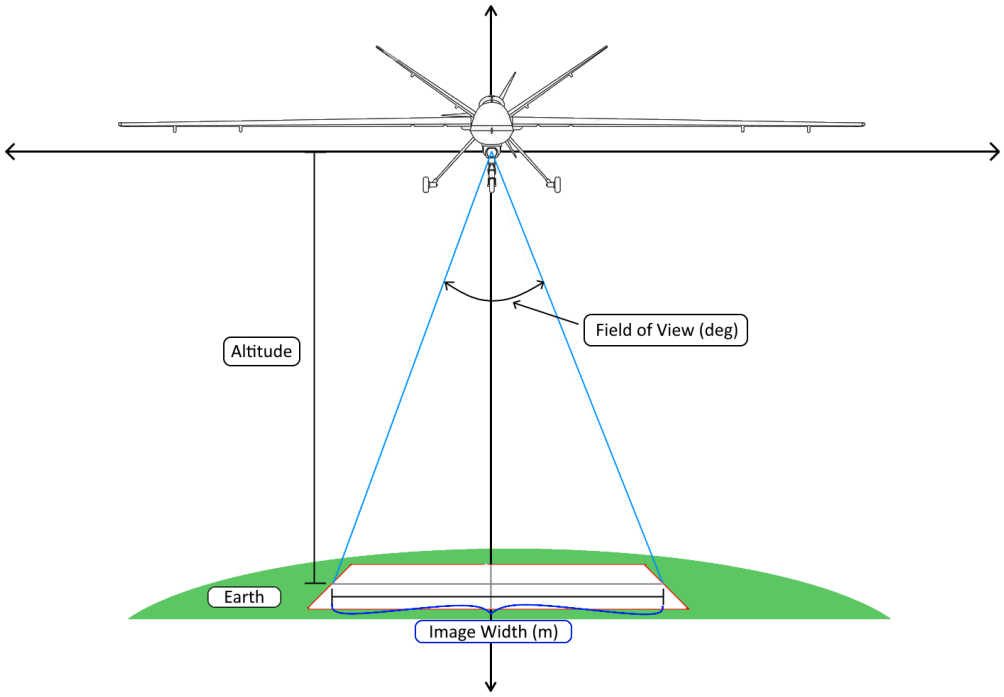


Figure 2: Image Width Illustration

2.2 VSLAM Algorithms

This research only focuses on monocular VSLAM methods, as stereo images and depth sensors are ineffective at high altitudes. VO methods can also be categorized based on how they track changes in the images captured. Feature based algorithms use a variety of methods to find unique features in each image, such as sharp corners. This allows for matching of features between images with large movement in between, at the cost of relying on a large number of tuning parameters which can drastically effect performance [11]. Direct algorithms estimate motion from the intensity values in the image, using every pixel available. This allows direct algorithms to use all of the information in an image, leading to more robust results. [3]. VO algorithms may also use a hybrid of both methods, typically called semi-direct VO [11]. In this research, all three types of VO algorithms will be used. The feature based methods include MonoSLAM [8], PTAM [9], OKVIS [10], and ORB-SLAM2 [2]. SVO [11]

is the only semi-direct method used. LSD-SLAM [3] is a direct method. Below, a brief description is given on each of the monocular VSLAM algorithms used in this research.

2.2.1 MonoSLAM

MonoSLAM was the first real time VSLAM algorithm, published by Davison et al. in 2007 [8]. It creates a probabilistic 3D map representing all current estimates of the states of the camera, image features, and all uncertainties associated with these measurements. This map is then updated using an Extended Kalman Filter as new images are received. This map is then used to locate the camera approximately 30 times a second. It is a sparse SLAM implementation, relying on features extracted from the images and tracked within the internal map. As a symptom of being monocular, MonoSLAM has no way to determine the scale of the map it maintains. Davison et al. deal with this by initializing the algorithm with known features to give an absolute scale.

2.2.2 PTAM

Also published in 2007, Klein and Murray developed the Parallel Tracking and Mapping (PTAM) algorithm [9]. PTAM had the novel idea of separating the tracking and mapping processes into separate threads to increase performance and accuracy. This allows the mapping process to be performed on keyframes instead of every frame. Whenever the algorithms recognizes a new video frame that is significantly different from the last keyframe, it can add it as a new keyframe. Bundle adjustment is then performed on the map, allowing for high accuracy estimates of the map. The tracking loop can continue in real time, separate from the mapping thread, allowing for incremental pose estimates to be calculated. PTAM also suffers from scale ambiguity,

which is resolved by translating a set distance of 10 cm between the first two images in the dataset.

2.2.3 LSD-SLAM

Engel et al. developed Large-Scale Direct SLAM (LSD-SLAM) in 2014 [3]. LSD-SLAM is the only fully direct SLAM algorithm tested in this research. It differs by using all of the information in an image, instead of extracting features from the image to help in building a map. LSD-SLAM is split into three main components: tracking, depth map estimation, and map optimization. The tracking component tracks the current camera pose with respect to the last keyframe. The depth map estimation component updates the keyframes using small-baseline stereo comparisons to create a depth map. The map optimization component incorporates these keyframe maps into the global map, and also detects loop closures and scale drifting. LSD-SLAM is also scale ambiguous, and needs to be paired with other systems to resolve absolute scales.

2.2.4 ORB-SLAM2

ORB-SLAM2 is one of the current state-of-the-art SLAM systems, introduced by Mur-Artal and Tardós in October 2017 [2]. It builds on the original ORB-SLAM system [16], which only worked on RGB-D systems. ORB-SLAM2 can work with monocular, stereo, or RGB-D datasets, incorporating keyframe tracking, local and global bundle adjustment, loop closure, place tracking, and pose-graph estimation. By combining all of these techniques into three parallel threads, ORB-SLAM2 has become one of the most robust and accurate VSLAM algorithms. Scale is still ambiguous in the monocular tracking case, but is absolute in the stereo and RGB-D cases.

2.2.5 SVO

SVO, introduced by Forster, Pizzoli, and Scaramuzza in 2014, is a hybrid semi-direct VSLAM algorithm [11]. Developed specifically for monocular VO on Micro Aerial Vehicles (MAVs), this method combines the robustness and speed of direct methods with the benefits of feature based tracking. SVO differs from other direct methods by extracting features during keyframes which are used to initialize depth filters, which are updated between keyframes to build the map. Motion estimation is accomplished by obtaining an initial guess of the new pose using direct model-based image alignment, then refining that guess using feature-based alignment. The motion estimation and mapping tasks are carried out in two separate threads for a significant performance gain. As a monocular algorithm, SVO has no way of extracting scale from its image sets, and thus the solution must be aligned to ground truth for accurate comparison.

2.2.6 Tuning Parameters

It should be noted that many of these algorithms have built in tuning parameters that can tweak performance to fit certain use cases. Common parameters include the changing the frequency of keyframes, turning loop closure on or off, as well as parameters associated with feature extraction for feature based algorithms. This research uses all of the algorithms as provided, with default parameters, although an effort is made to determine if a change in parameters will significantly impact results. This is further discussed in the Analysis section of Chapter 4.

2.3 Visual SLAM Datasets

Several common datasets have been used to compare the effectiveness of different VSLAM algorithms. While a custom simulated dataset is being used in this research,

these common datasets are used to help ensure that each VSLAM algorithm is working as intended. A brief overview of each dataset is given below:

2.3.1 ICL-NUIM

The Imperial College London and National University of Ireleand Maynooth (ICL-NUIM) dataset was introduced by Handa et al. in 2014 [17]. It is a series of handheld RGB-D sequences within synthetically generated environments. This allowed for the creation of extremely precise datasets with exact truth trajectories, as well as accurate surface truth models. The synthetic nature of the dataset also allows for different versions of each trajectory: one a noiseless trajectory with perfect depth measurements, and a more real-world scenario where the quality of the image is degraded, and the depth and image measurements suffer from noise. These various indoor scenes provide repeatable, accurate datasets for the comparison of VSLAM algorithms.

2.3.2 TUM-RGBD

The TUM-RGBD dataset is a widely used dataset consisting of 39 real-world sequences recorded using a Microsoft Kinect Sensor. It was presented by Sturm et al. in 2012 [18]. The sequences cover a wide variety of scenes and camera movements, including sequences for debugging, small scale desk scenes, and larger scale sequences with and without loop closures. Sequences were captured with the Kinect being both handheld and mounted to a robot. The large variety of real-world scenes in this dataset makes it useful for debugging and testing various VSLAM algorithms.

2.3.3 EuRoC MAV

The European Robotics Challenge Micro Aerial Vehicle (EuRoC MAV) dataset contains real-world data collected by a small aerial vehicle operating in an indoors environment. It was presented by Burri et al. in 2016 [19]. This dataset includes stereo images, inertial measurement unit (IMU) measurements, and highly accurate ground truth data. The dataset is split into two batches, with the first batch facilitating the evaluation of visual-inertial algorithms, and the second batch evaluating precise 3D environment reconstruction.

2.3.4 SUSEX

The Small Unmanned Systems Exploitation (SUSEX) dataset is a product of the Air Force Research Laboratory (AFRL) Sensors Directorate SUSEX team. It includes several real world flight datasets from a fixed-wing small unmanned aerial vehicle (UAV). The dataset was taken over Avon Park, FL in December 2016. It includes a large array of sensor data, including IMU, Global Positioning System (GPS), and monocular imagery. It was collected to facilitate a wide array of testing using many sensors. For this research, the GPS/IMU combined (INS) solution is being used as a truth trajectory, with the IMU and monocular imagery data being given to the VSLAM algorithms.

2.4 SLAMBench2 Benchmarking Suite

With so many different datasets and algorithms, the need to benchmark and compare algorithms within the same framework has been recognized. Several benchmarking suites exist, often tailored to their specific usecases. For example, the KITTI Vision Benchmark Suite is a software suite designed to compare VSLAM algorithms for use in self driving cars [20]. It includes its own datasets, including stereo cam-

eras, GPS, and a 360 degree laserscanner. Its utility in evaluating algorithms for applications besides autonomous driving, however, is limited.

To create a more general benchmarking software suite, SLAMBench, published by Nardi et al. in 2015, was designed to run different implementations of the KinectFusion algorithm [21] on various hardware. It was, however, limited to using KinectFusion and the ICL-NUIM dataset.

In an effort to release a truly general benchmark suite, Bodin et al. developed SLAMBench2 in 2018 [13]. This updated version of the software implements eight different SLAM algorithms natively: ORB-SLAM2 [2], MonoSLAM [8], OKVIS [10], PTAM [9], ElasticFusion [22], InfiniTAM [23], KinectFusion [21], and LSD-SLAM [3]. It also supports the ICL-NUIM dataset [17], the TUM-RGBD dataset [18], and the EuRoC MAV dataset [19] natively. More importantly, however, it provides an easy to use interface for implementing new SLAM algorithms and datasets natively within the software. Modifying SLAMBench2 to support testing of navigation algorithms for fixed-wing aircraft was the goal of this research.

2.5 Related Works

While most new VSLAM algorithms make a point to compare themselves against the current state of the art [2, 3, 22], these are all compared using indoor or ground-level trajectories. Due to the large distances from the camera to the objects being observed inherent in fixed-wing flight, these comparisons did not address our need for a fixed-wing VSLAM system.

Some work has been done in evaluating and developing VO algorithms for aerial applications. Carson investigated the feasibility of implementing visual-inertial odometry on a fixed-wing aircraft using a Kalman Filter [5]. In this work, Carson implemented four VO variants, and compared them against the Semi-Direct Visual Odom-

etry (SVO) algorithm [11]. Carson's work was limited to visual odometry, however, and did not expand into a full VSLAM system.

Ellingson et al. also published a work on visual-inertial SLAM for fixed wing aircraft in 2018 [24]. This work presented using a relative front end for state estimation while using a global back end for optimizations and loop closures. This was tested using custom simulated data, but was found to not be able to run in real time.

More recently, Kim evaluated several VO algorithms on aerial fixed-wing datasets, both simulated and real-world [6]. In this work, SVO was tested against DSO and ORB-SLAM2 with loop closures disabled. Each VO algorithm was tested on multiple simulated and real-world datasets. It was found that ORB-SLAM2 was the most robust solution, with all of the algorithms struggling to initialize in real-world conditions. Another limitation of this research was the disabling of ORB-SLAM2's loop closures to more accurately compare it to the other algorithms. Again, none of the systems tested were full VSLAM systems.

In this thesis, we attempt to overcome the shortcomings of this prior work by (1) using common datasets across all algorithms that are representative of fixed-wing flight and (2) running full VSLAM algorithms that are available in the literature.

2.6 Background Summary

This chapter introduced core concepts behind VSLAM methods. It also gave backgrounds on the VSLAM algorithms and datasets used in this research. It discussed the SLAMBench2 benchmarking suite that is core to this research. Finally, it summarized related research in aerial visual odometry and visual SLAM. This thesis looks to enable the testing of VSLAM algorithms on data collected from fixed-wing aircraft.

III. Methodology

The primary purpose of this thesis is to evaluate the effectiveness of existing VSLAM algorithms on data from fixed-wing aircraft. To perform this analysis, we first tried several different algorithms on a preexisting dataset (the SUSEX dataset) representing mid-level UAV flight. Unfortunately, none of the algorithms had acceptable performance with this dataset. Therefore, we shifted to using primarily simulated data to understand the system parameters that could be prohibiting any of the algorithms from performing appropriately. Specifically, we used simulated datasets to test the effect of the camera's field of view (FOV), the aircraft's bank angle, and the aircraft's bank rate on the performance of several Visual Simultaneous Localization and Mapping (VSLAM) algorithms.

This chapter goes over the experimental design of this research, starting with a discussion of the variables of interest and following with an overview on the simulated datasets that were created. It then overviews which VSLAM algorithms were used in testing, including any changes made to those algorithms, if any. It outlines the platform on which this testing was performed, and finally discusses the performance metrics used to determine the quality of the tracking solutions. It concludes with a summary of the chapter.

3.1 Variables of Interest

This research examines the effect of the camera's FOV, the aircraft's bank angle, bank rate, and altitude on the quality of the VSLAM solution. We chose to test FOV first, as it seemed to be a reasonable explanation for why the algorithms failed on the original dataset. When the initial simulations supported this, we then decided to test the altitude, bank angle, and bank rate to see how they affected the solutions.

Here, the bank angle is defined as the roll angle of the aircraft during a coordinated turn, while the bank rate is the rate at which the aircraft transitions from level flight to that bank angle. While these are related (reaching a higher bank angle in the same amount of time will also mean a higher bank rate), it is one of the goals of this research to isolate the effects of the bank rate vs bank angle, and to find out which, if any, is the limiting factor in the performance of the VSLAM tracking solution. Note that for the rest of this thesis, the word tracking is used to refer to when an algorithm successfully initializes, produces a position solution that is non-zero, and does not throw a runtime error. We also look at Image Width to try and isolate the effect of FOV from the effect of Altitude.

3.2 Simulation

The majority of the datasets used in this research were simulated in order to isolate the effects of the variables of interest, as well as to provide repeatable results over various trial runs. In addition to isolating changes in specific variables, simulation also increases the amount of data that can be "collected" in a given time frame, as far less time has to be spent capturing simulated data than real world flight tests. This section includes an overview on the simulation software used to create these datasets, a discussion of how these simulations compare to real flight paths, and a discussion of the variables these simulations help isolate in the datasets.

The simulated datasets in this research are generated using the AftrBurner Engine, an educational game engine created by Nykl et al. as the successor to the STEAMiE game engine [12]. This engine allows for the use of satellite imagery from Google Maps, as well as the ability to script camera movement in accordance with a ground truth file. An example of the imagery produces by this software is shown in Fig. 3. It also allows for the defining of the camera in terms of its resolution, distortion

coefficients, and FOV, or by using the camera calibration matrix directly, whichever is preferred. The AftrBurner engine can also map the satellite images to 3D ground geometry, as done by Kim in [6], however this research uses the 2D imagery directly.

A total of 59 flight datasets were used to perform this research. The first set of 40 datasets served to test each VSLAM algorithm for dependencies on altitude, camera FOV, and bank angle. To further test the dependency on bank angle and bank rate, a second set of datasets was created, consisting of 16 further datasets. This second set tested dependence on bank rate and bank angle independently, as well as further testing dependence on camera FOV. (We discuss how these datasets were created to test the desired parameters in the following sub-sections.) Finally, a single real world flight was used from the SUSEX flight dataset, paired with two equivalent simulated datasets. This served as a parity check between the simulated datasets and a real world dataset. All datasets used a camera resolution of 1600x1200 pixels.

3.2.1 Dataset Group 1

The first group of datasets consisted of a flight path that was created starting with 30 seconds of straight and level flight at 15 m/s. The flight path then took a series of 90 degree coordinated turns at increasing bank angles, starting with a 5 degree

Figure 3: Left shows the real world imagery taken from the SUSEX Flight 12 dataset. Right shows the equivalent imagery created in the AftrBurner Engine.

bank angle and continuing until reaching a 30 degree bank angle. Bank angles were reached by linearly rolling for 1.5 seconds before and after each turn. Each 90 degree turn was connected with another 30 second straight and level section. Finally, after the last 90 degree turn at a 30 degree bank angle, another 30 second straight and level section was added. This flight path can be seen in Fig. 4.

This flight path was created with 5 different altitudes, 100, 450, 800, 1150, and 1500 meters to create 5 different ground truths for testing. These 5 ground truths were then simulated with 8 different camera FOVs: 30, 45, 60, 75, 90, 105, 120, and 135 degrees for a total of 40 datasets.

3.2.2 Dataset Group 2

The second group of datasets was created to test if sharper bank angles would have an effect on tracking, as well as test if the bank rate was the limiting factor instead of the bank angle. This dataset group was constructed similarly to the first, with an aircraft traveling at 15 m/s, this time at a fixed altitude of 800m. Straight sections were limited to 15 seconds (225m) to limit the length of the dataset. Bank angles again started at 5 degrees and increased in 5 degree increments, this time up to a 60 degree bank. Finally, two different ground truths were made from this, one taking 1.5 seconds to reach the bank angle, similar to before, and the other taking 3 seconds to reach the bank angle. This way it could be determined if the algorithms were failing at a certain bank angle, or if they were instead failing at a specific bank rate. The ground truth for this flight path can be seen in Fig. 5.

3.2.3 Simulated Flight 12

A small section of the SUSEX 12 Avon Park Flight 12 dataset was chosen to help test against the simulation. The section chosen had a loop with bank angles less than

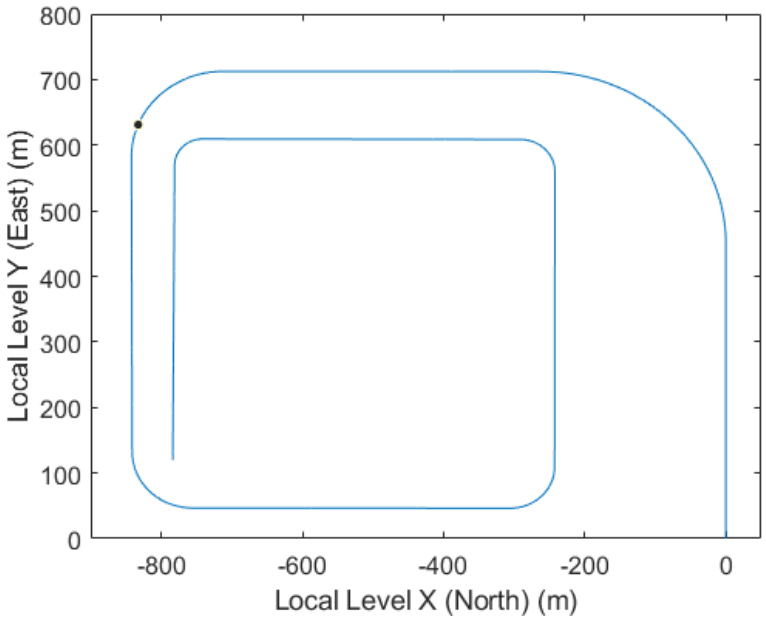


Figure 4: Ground truth used for Dataset Group 1. Aircraft moves at 15 m/s, with each straight section being 30 seconds long (450m). Each 90 degree turn has a successively higher bank angle, starting at 5 degrees and working up to 30 degrees in 5 degree increments.

30 degrees at an average altitude of 483m. The SUSEX dataset did not have the camera pointing straight down in the aircraft body frame, instead it was oriented 50 degrees below horizontal, and pointed towards the aircraft's right wing, inside of the loop taken. Additionally, the camera used had a horizontal FOV of only 25 degrees. The ground truth for this dataset is shown in Fig. 6.

Two simulated datasets were made based on this flight. One was made exactly according to the original flight. The only change made for the second simulation was the camera FOV was increased to 90 degrees. These simulated and real world flights allowed testing the parity of the simulations to real world flight tests.

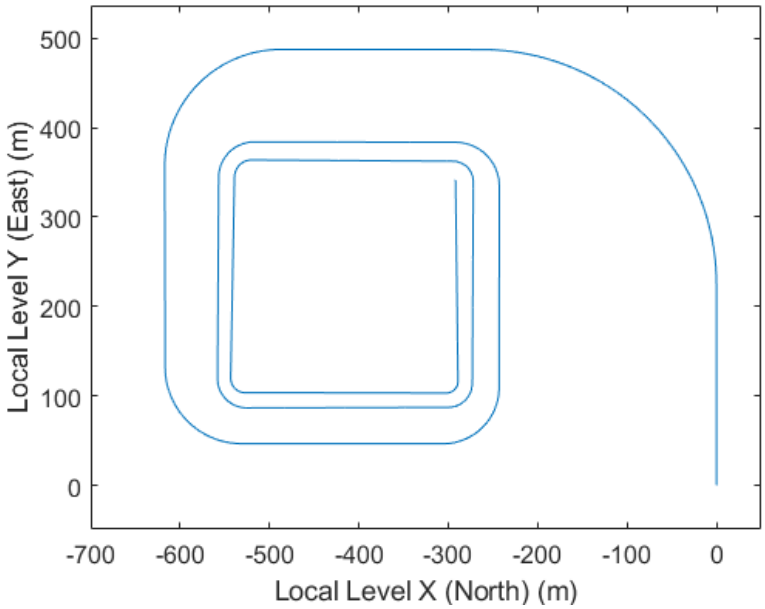


Figure 5: Ground truth used for Dataset Group 2. Aircraft moves at 15 m/s, with each straight section being 15 seconds long (225m). Each 90 degree turn has a successively higher bank angle, starting at 5 degrees and working up to 60 degrees in 5 degree increments.

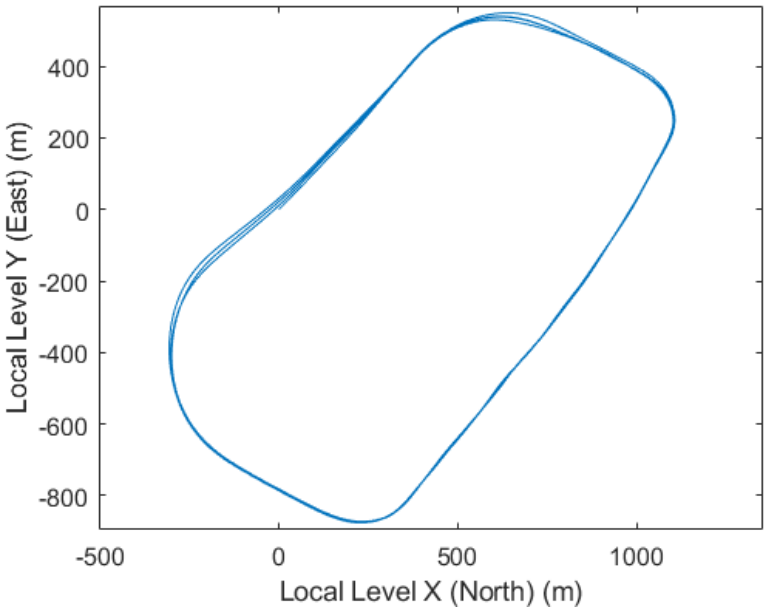


Figure 6: Ground truth taken from Flight 12 of the 12 Avon park dataset from the SUSEX group. The aircraft here flies in a loop, taking less than 30 degree bank angle turns, at an average altitude of 1587m.

3.3 VSLAM Algorithms

This research focuses on the monocular VSLAM algorithms that were implemented in the SLAMBench2 benchmarking suite [13]. These include: ORB-SLAM2 [2], LSD-SLAM [3], PTAM [9], MonoSLAM [8], OKVIS [10], and SVO [11]. Of these, the implementations of ORB-SLAM2 and OKVIS had to be modified within the SLAMBench2 framework to accept monocular datasets, as the original implementation only provided support for stereo imagery. It should be noted that the algorithms themselves were not modified, only the SLAMBench library file that interfaced the algorithms to SLAMBench2. LSDSLAM was also run using two different implementations available in SLAMBench2, noted as LSDSLAM and LSDSLAM_cpp, which has the LSDSLAM library entirely implemented in the c++ programing language. For completeness, both are analyzed in this research.

These algorithms were first tested with dataset group 1 to determine whether or not the algorithms would even function on aerial vehicle datasets. If so, they were tested with all of the datasets to determine performance dependencies as outlined above.

3.4 Processing Platform

All processing was done on a dedicated platform, the specifications of which are outlined in Table 1. SLAMBench2 does provide metrics for processing time and memory used, but as this will change depending on the platform used, we do not report it in this thesis.

Operating System	Ubuntu 18.04.3 LTS - 64 bit
Processor	Intel Core i5-3570k @ 3.40 GHz
Graphics	Intel Ivybridge Integrated Graphics
Memory	24 GB Patriot Viper DDR3-2133

Table 1: Processing Platform Specifications

3.5 Performance Metrics

This research will primarily look at the accuracy of the position solution given by the VSLAM algorithms. As SLAMBench2 does not output rotational information when outputting solutions, only the accuracy of the positional solution will be considered. As all of the VSLAM algorithms tested are monocular algorithms, none of the solutions will be in the correct scale, and thus need to be aligned. This alignment was done using the trajectory alignment tool developed by Zhang and Scaramuzza [14]. This tool gives us the Root Mean Squared Error (RMSE) of each trajectory compared to the ground truth. It should be noted that SLAMBench2 includes an alignment step as well as outputting performance metrics, however Zhang and Scaramuzza's tool was found to be more robust for the datasets used in this research.

The RMSE compared for each solution was computed only for the part of the solution that successfully tracked the ground truth without producing an error. If an algorithm completely lost tracking during the dataset, the solution was trimmed there and the RMSE only computed up to that point. Thus where the algorithm failed in the dataset is also an important metric to determining the performance of the solution.

3.6 Summary

This chapter described the simulated datasets relied on in this research, detailing the software used to create them as well as the content of the datasets themselves. It then laid out the experimental design used to test these datasets, including all assumptions and limitations, the software and hardware used, and the algorithms being tested. It finished with a discussion on the performance metrics used to evaluate the solutions gained.

IV. Results and Analysis

This chapter describes the results obtained from the datasets detailed in Chapter III. Each algorithm is analyzed with respect to its viability for use with aerial vehicles, as well as its dependency on aircraft altitude, bank angle, bank rate, and camera field of view (FOV).

4.1 Simulation Results

This section describes the results gained from testing using the simulated datasets. These results are analyzed separately in section 4.2.

4.1.1 Altitude and FOV Results

Dataset group 1 was tested with each of the algorithms. The trajectory alignments for LSDSLAM, LSDSLAM_cpp, ORB-SLAM2, and SVO are shown in Fig. 7 - Fig. 10, respectively. These figures show the aligned trajectory solution given by each algorithm, up to the point where the algorithms failed. Empty boxes denote where algorithms failed to initialize altogether. Fig. 11 through Fig. 18 show the accumulated error for each algorithm. The scales are kept constant to allow for easier comparison. PTAM, MonoSLAM, and OKVIS are notably missing from the results, as they were unable to initialize on any of the test datasets used, and thus are considered poor candidates for aerial vehicle navigation. It should also be noted that SVO consistently failed early, possible due to a memory leak. This is discussed further in the Analysis section.

The numerical results are shown in Table 2, which shows the Root Mean Squared Error (RMSE) for each algorithm at each altitude and FOV, as well as the maximum bank angle turn completed by each algorithm, corresponding to Fig. 7 - Fig. 10. An X

in the table represents where the algorithm failed to initialize. The table only shows statistics for the portion of the trajectory successfully tracked by the algorithm (e.g. if an algorithm failed part way through the dataset, only the error statistics up to that point are reported). Fig. 19 shows the RMSE for each FOV at each altitude and algorithm tested. Fig. 20 and Fig. 21 show the RMSE for each altitude for each FOV and algorithm tested. The results were also converted to use Image Width and graphed in Fig. 22. These results are discussed in the Analysis section.

4.1.2 Bank Angle vs Bank Rate Results

Fig. 23 through Fig. 26 show the results of each of the algorithms tested against dataset group 2. Blank boxes denote where algorithms failed to initialize. Again, each trajectory is only shown to the point where the algorithm failed. Table 3 shows the RMSE and maximum bank angle turn for each of these runs as well, with each "X" denoting the algorithm failing to initialize. Fig. 27 shows the RMSE vs FOV, comparing the time to bank for each algorithm.

4.1.3 SUSEX Flight 12 - Real and Simulated

None of the algorithms were able to successfully track the real world SUSEX Flight 12 dataset, or the simulated Flight 12 dataset with a 25 degree FOV. The algorithms' position solutions to the simulated Flight 12 dataset with a 90 degree FOV are shown in Fig. 28.

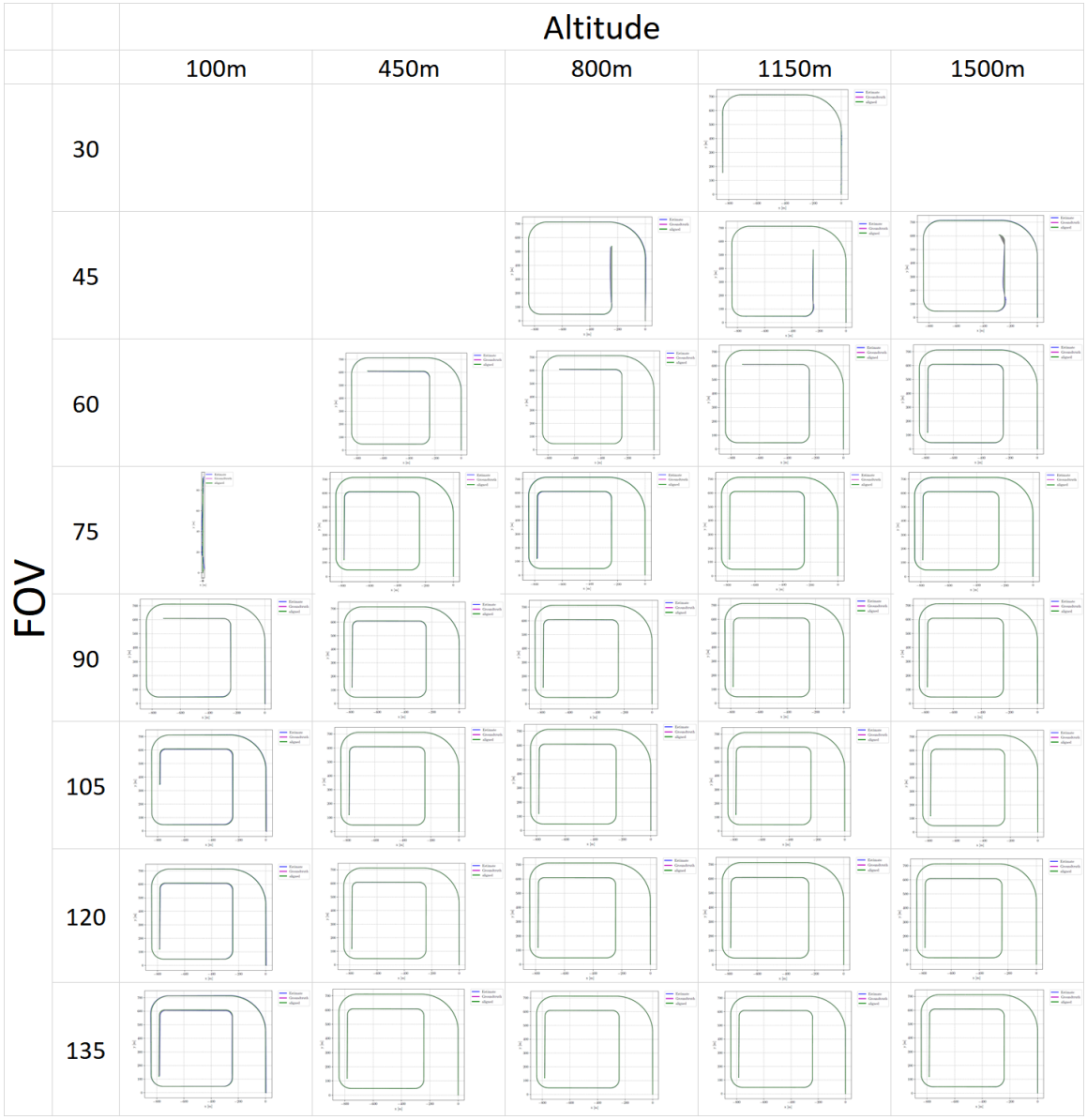


Figure 7: LSDSLAM trajectory alignment for dataset group 1, comparing solution performance at various altitudes and camera FOVs.

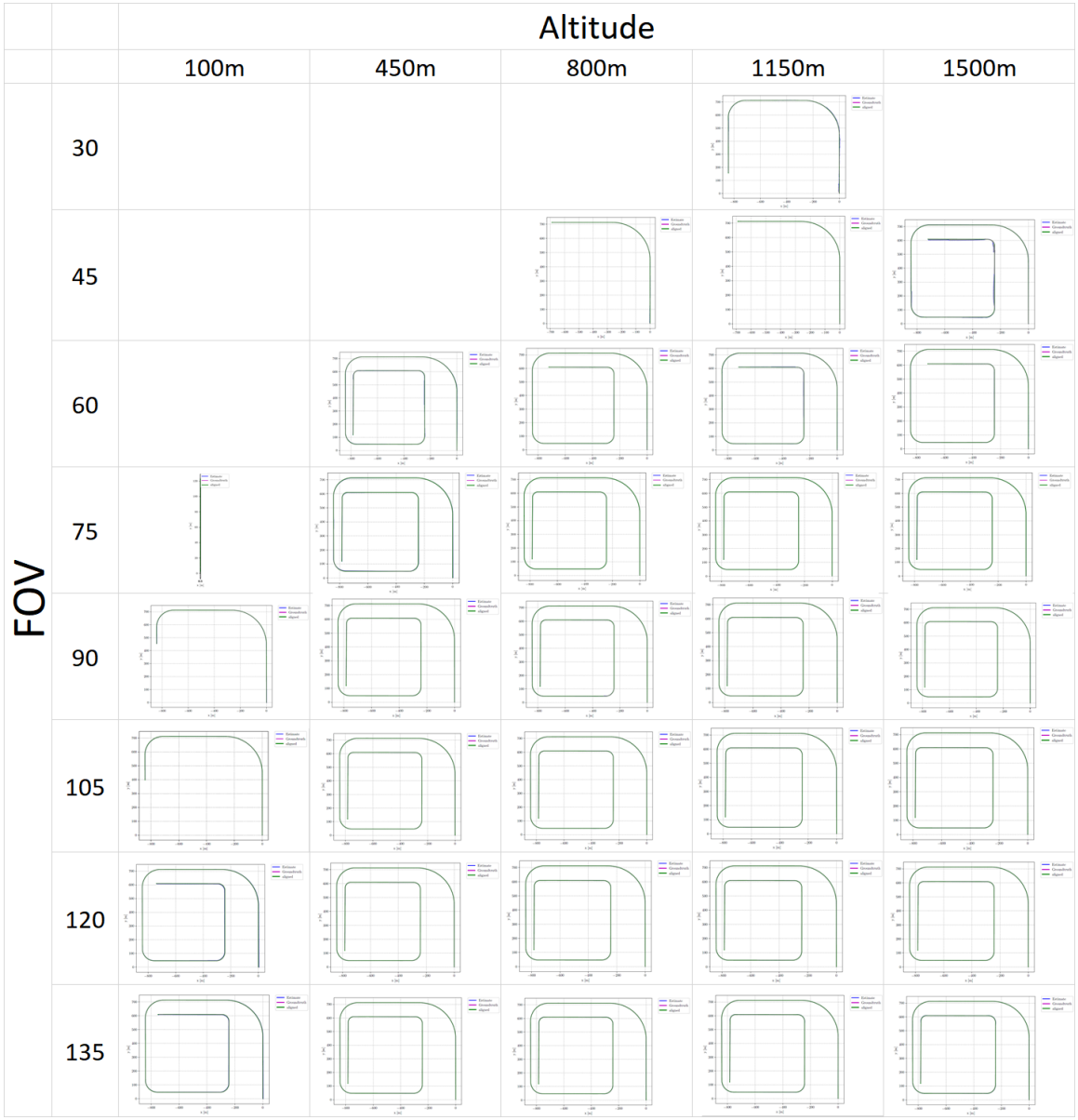


Figure 8: LSDSLAM_cpp trajectory alignment for dataset group 1, comparing solution performance at various altitudes and camera FOVs.

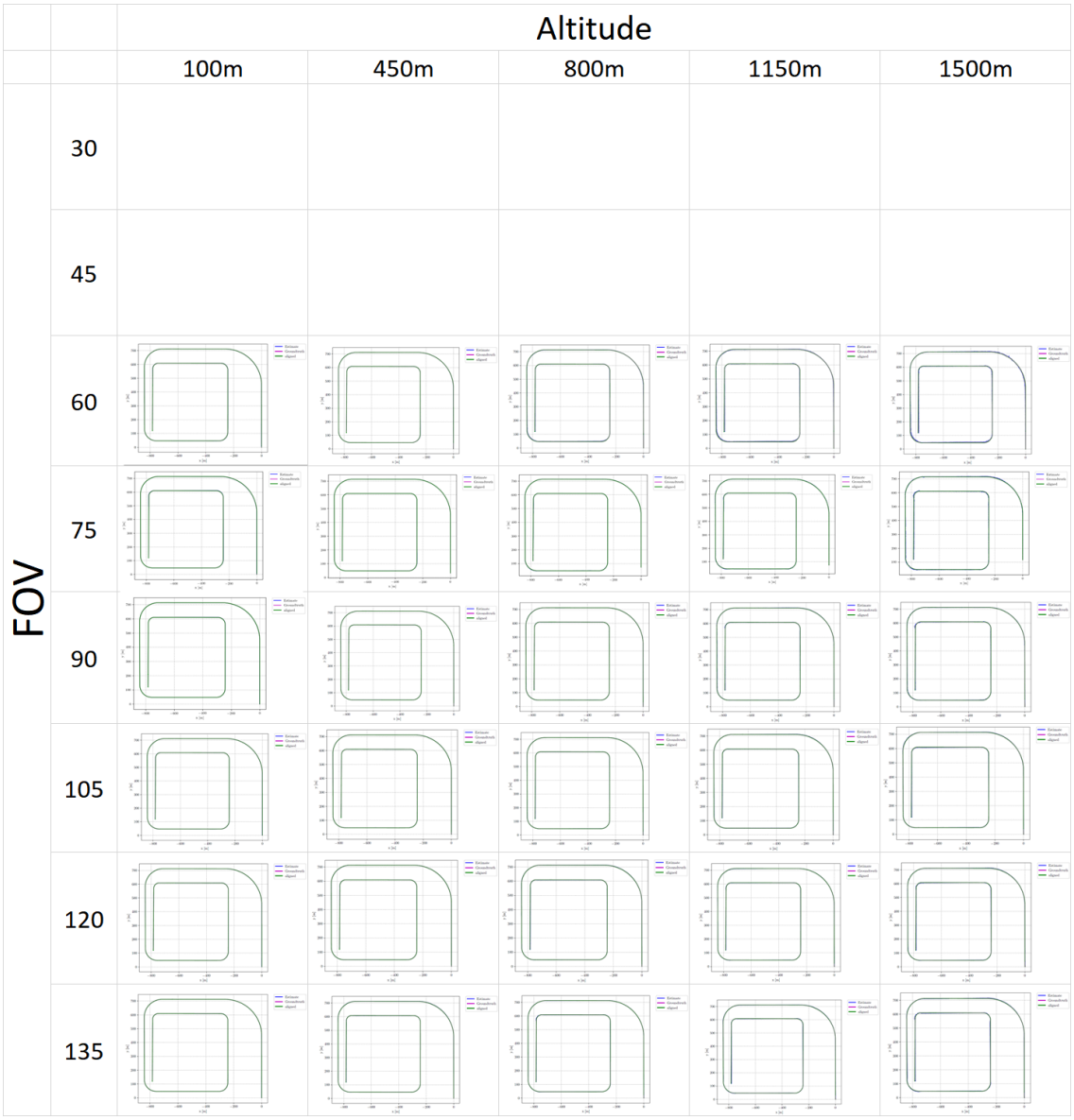


Figure 9: ORB-SLAM2 trajectory alignment for dataset group 1, comparing solution performance at various altitudes and camera FOVs.

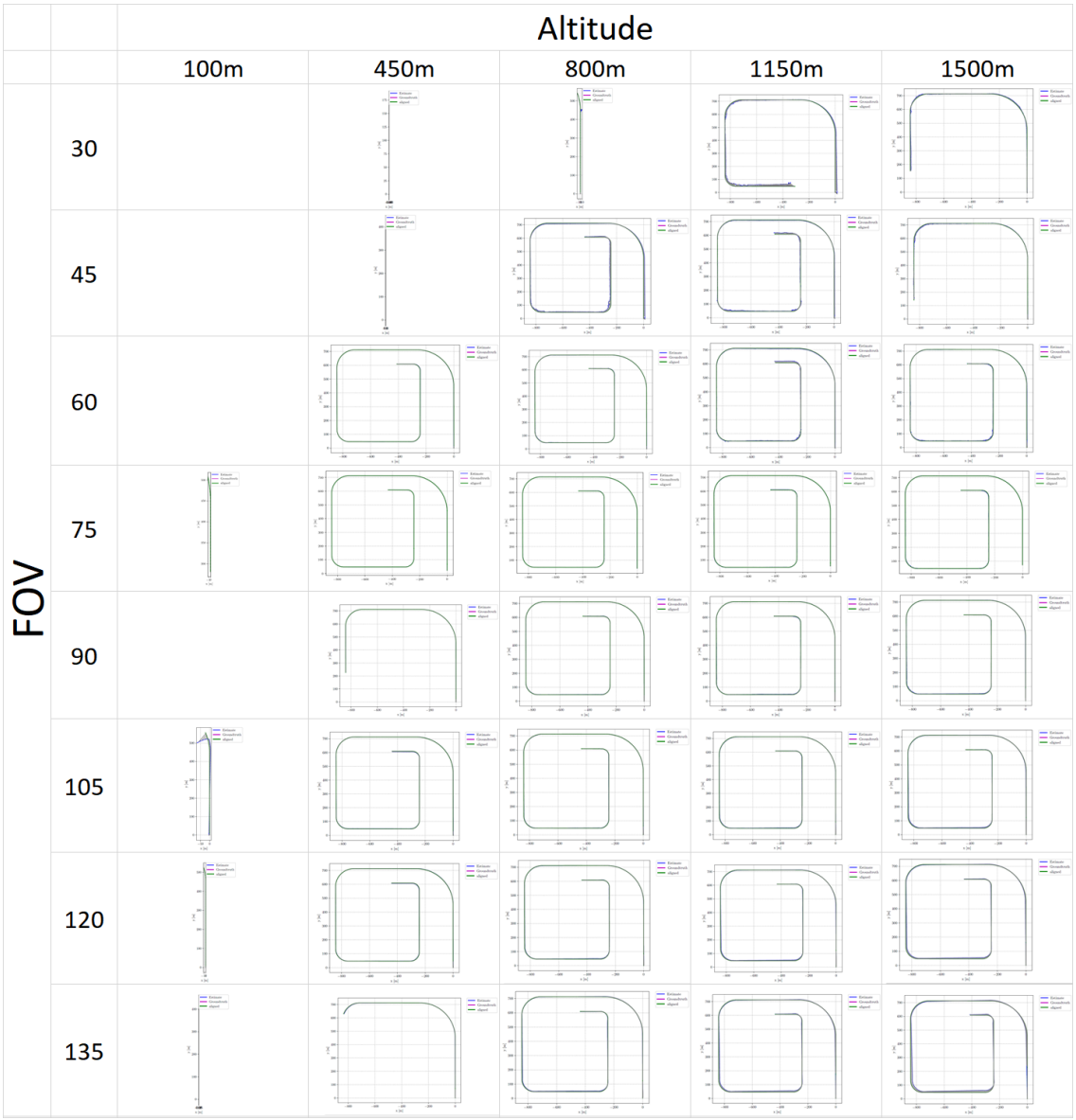


Figure 10: SVO trajectory alignment for dataset group 1, comparing solution performance at various altitudes and camera FOVs.

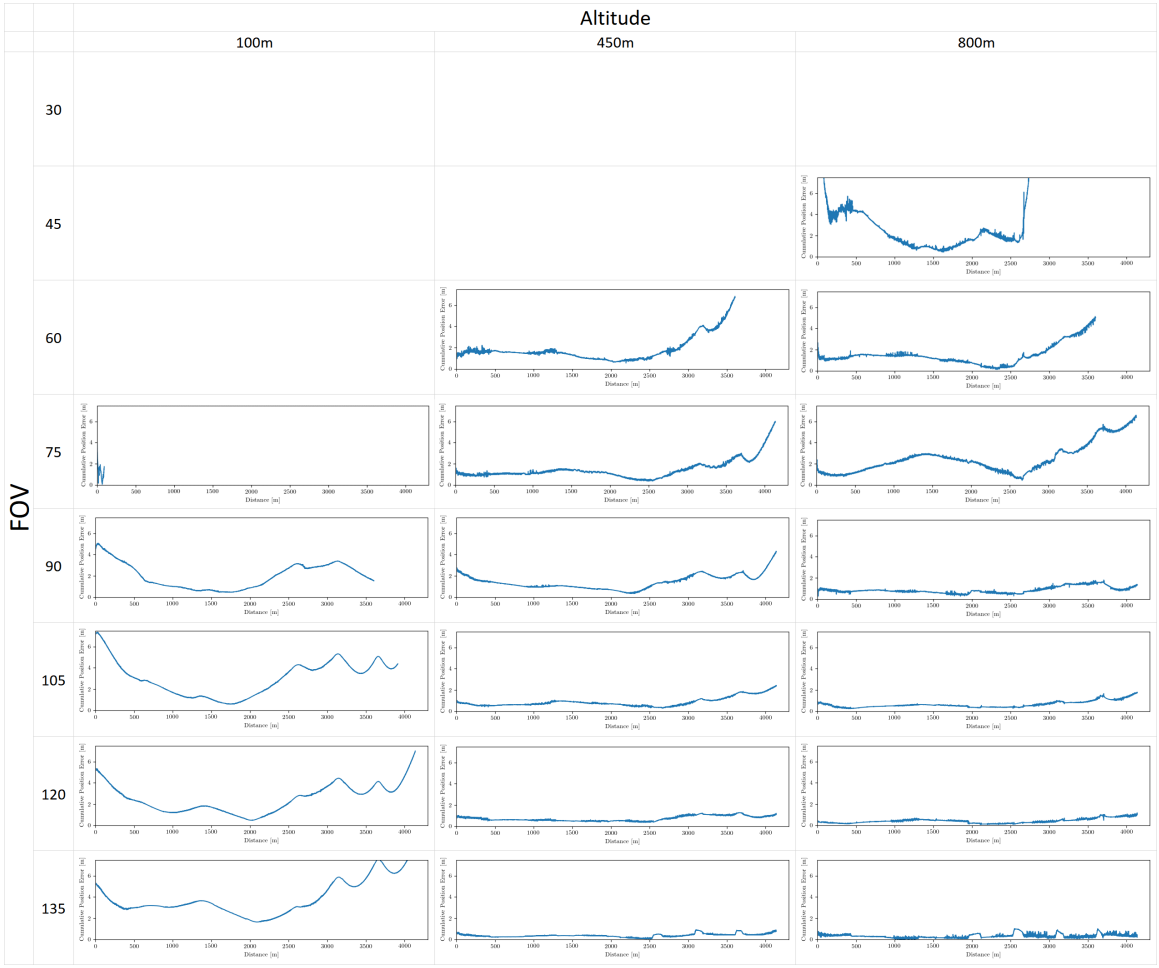


Figure 11: LSDSLAM error accumulation graphs for dataset group 1 altitudes 100-800m, showing total error accumulated at various altitudes and camera FOVs.

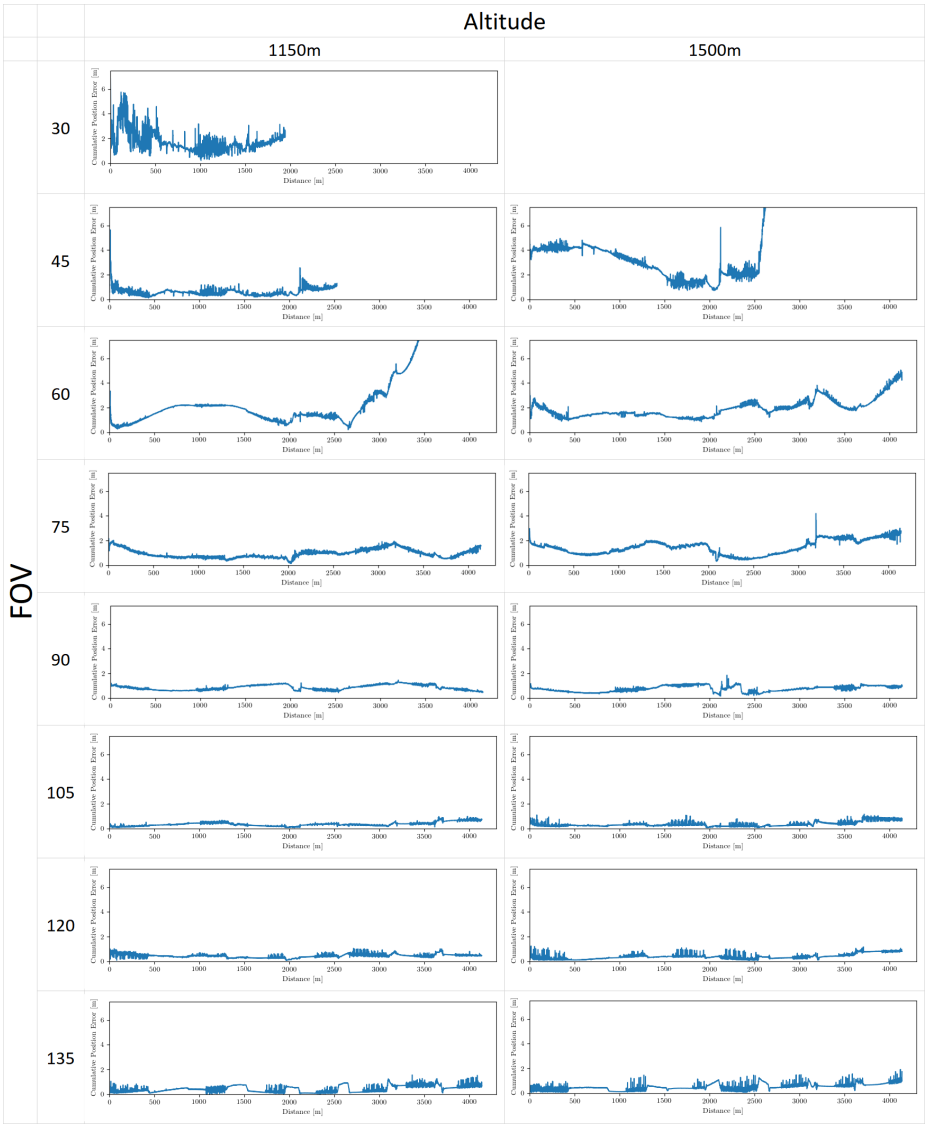


Figure 12: LSDSLAM error accumulation graphs for dataset group 1 altitudes 1150m and 1500m, showing total error accumulated at various altitudes and camera FOVs.

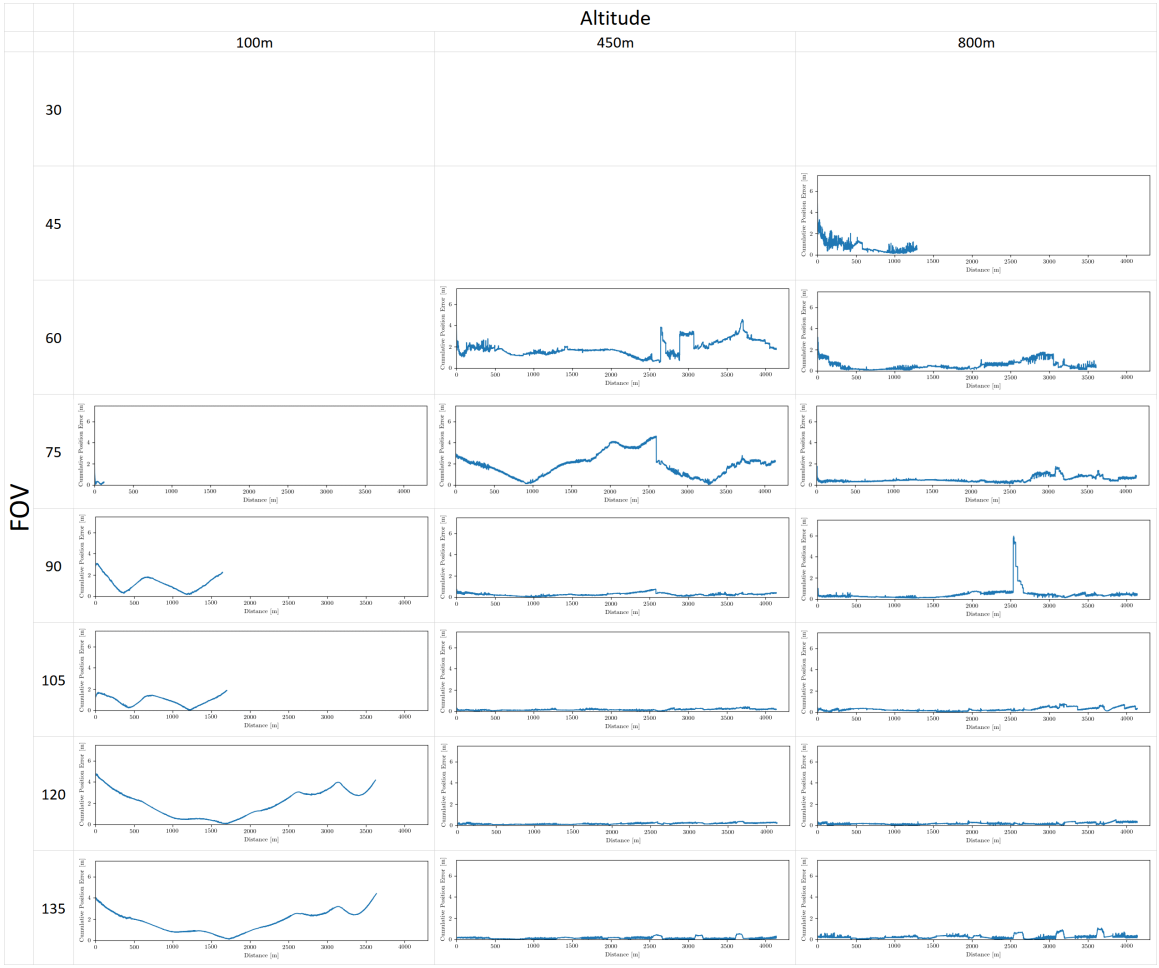


Figure 13: LSDSLAM_cpp error accumulation graphs for dataset group 1 altitudes 100-800m, showing total error accumulated at various altitudes and camera FOVs.

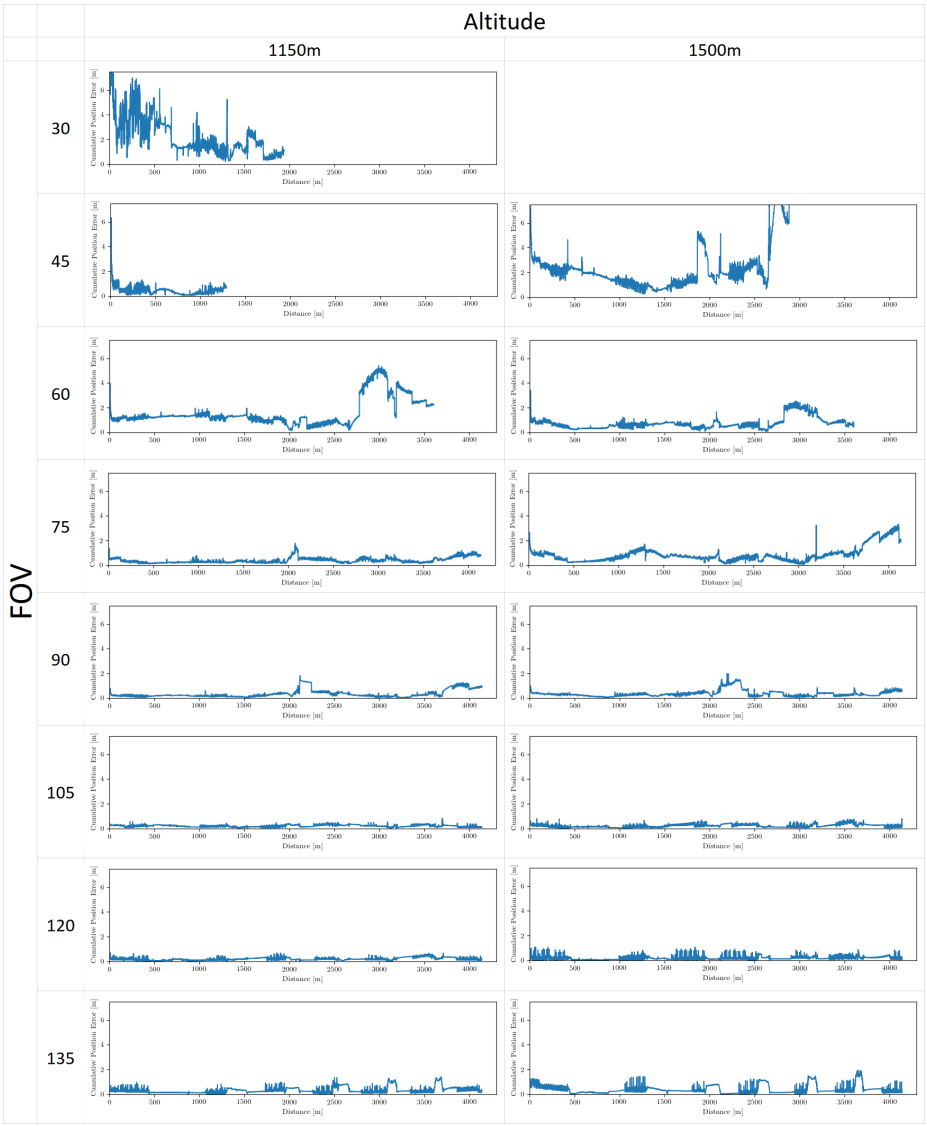


Figure 14: LSDSLAM_cpp error accumulation graphs for dataset group 1 altitudes 1150m and 1500m, showing total error accumulated at various altitudes and camera FOVs

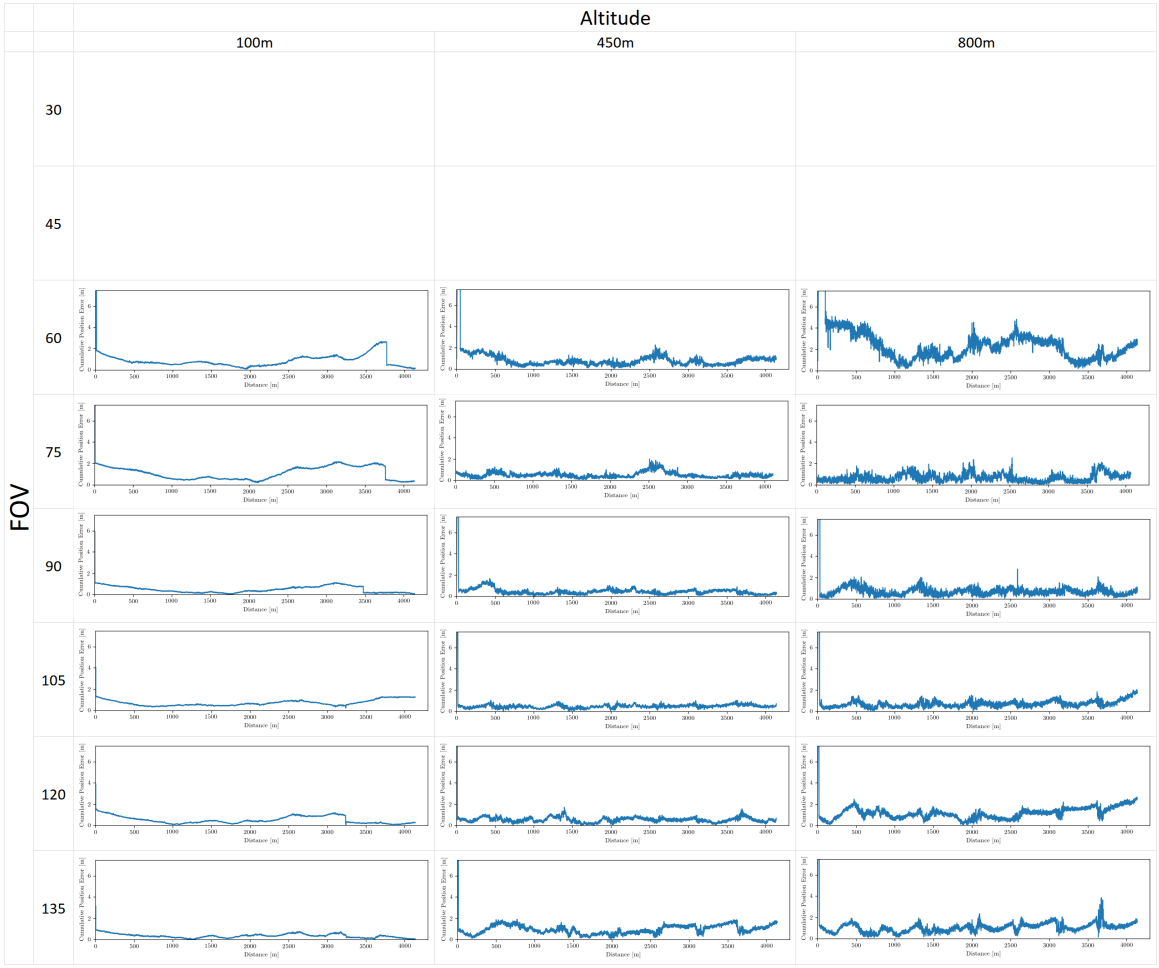


Figure 15: ORB-SLAM2 error accumulation graphs for dataset group 1 altitudes 100-800m, showing total error accumulated at various altitudes and camera FOVs.

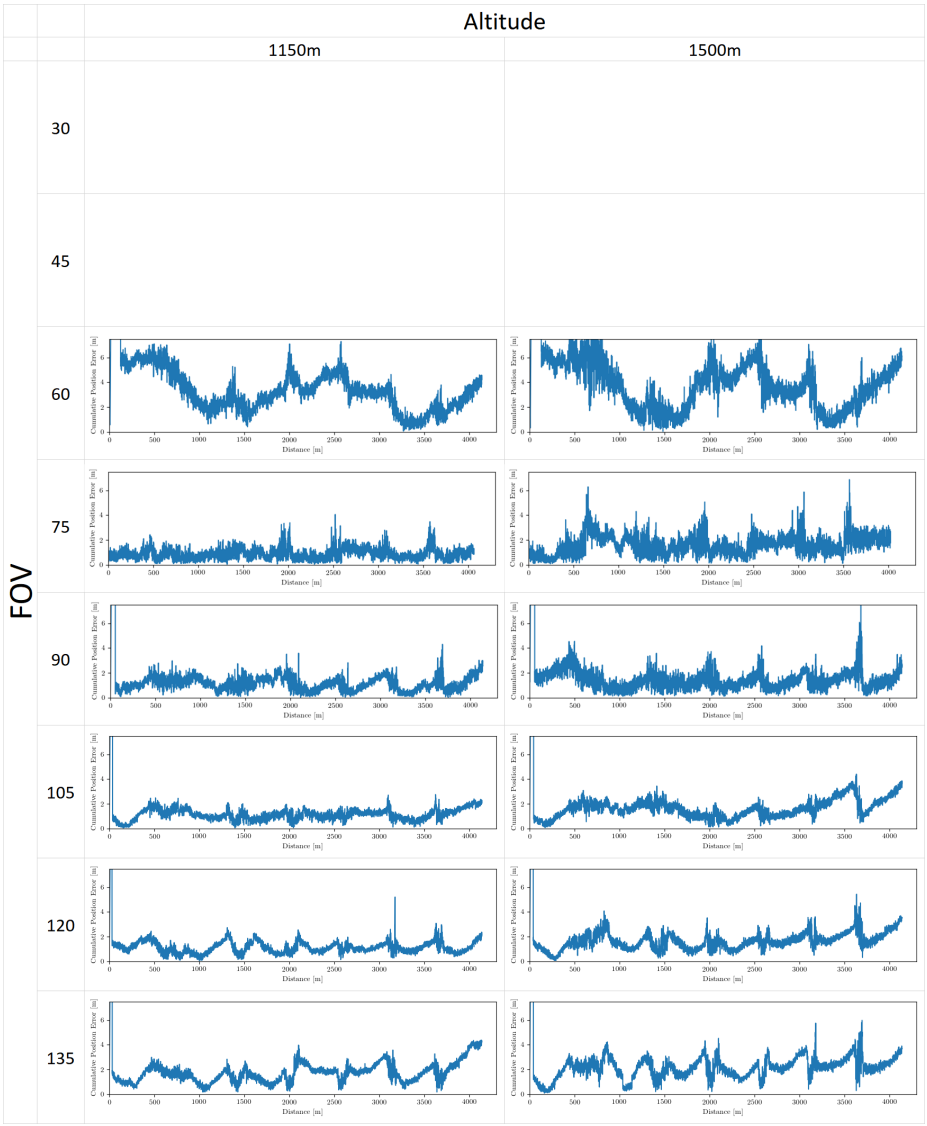


Figure 16: ORB-SLAM2 error accumulation graphs for dataset group 1 altitudes 1150m and 1500m, showing total error accumulated at various altitudes and camera FOVs.

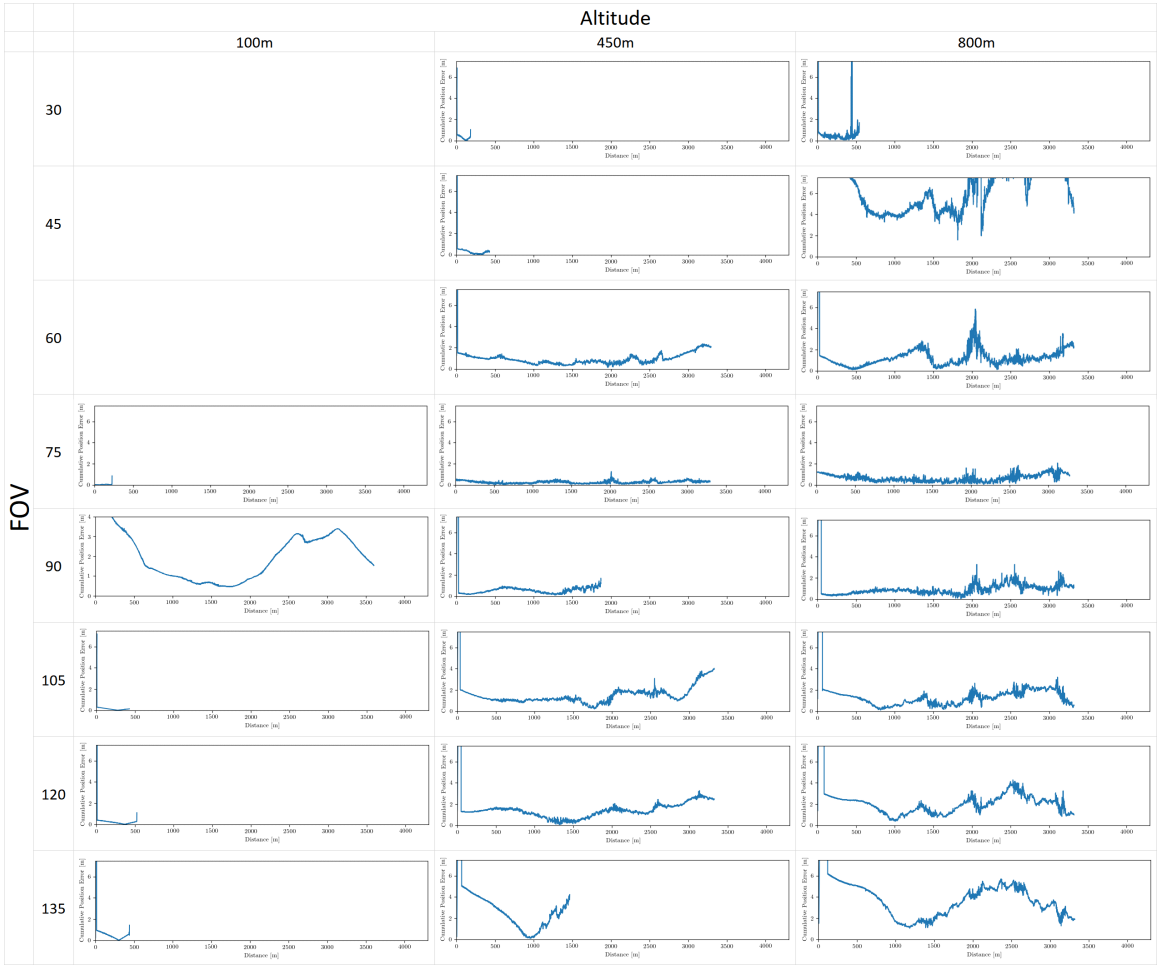


Figure 17: SVO error accumulation graphs for dataset group 1 altitudes 100-800m, showing total error accumulated at various altitudes and camera FOVs.

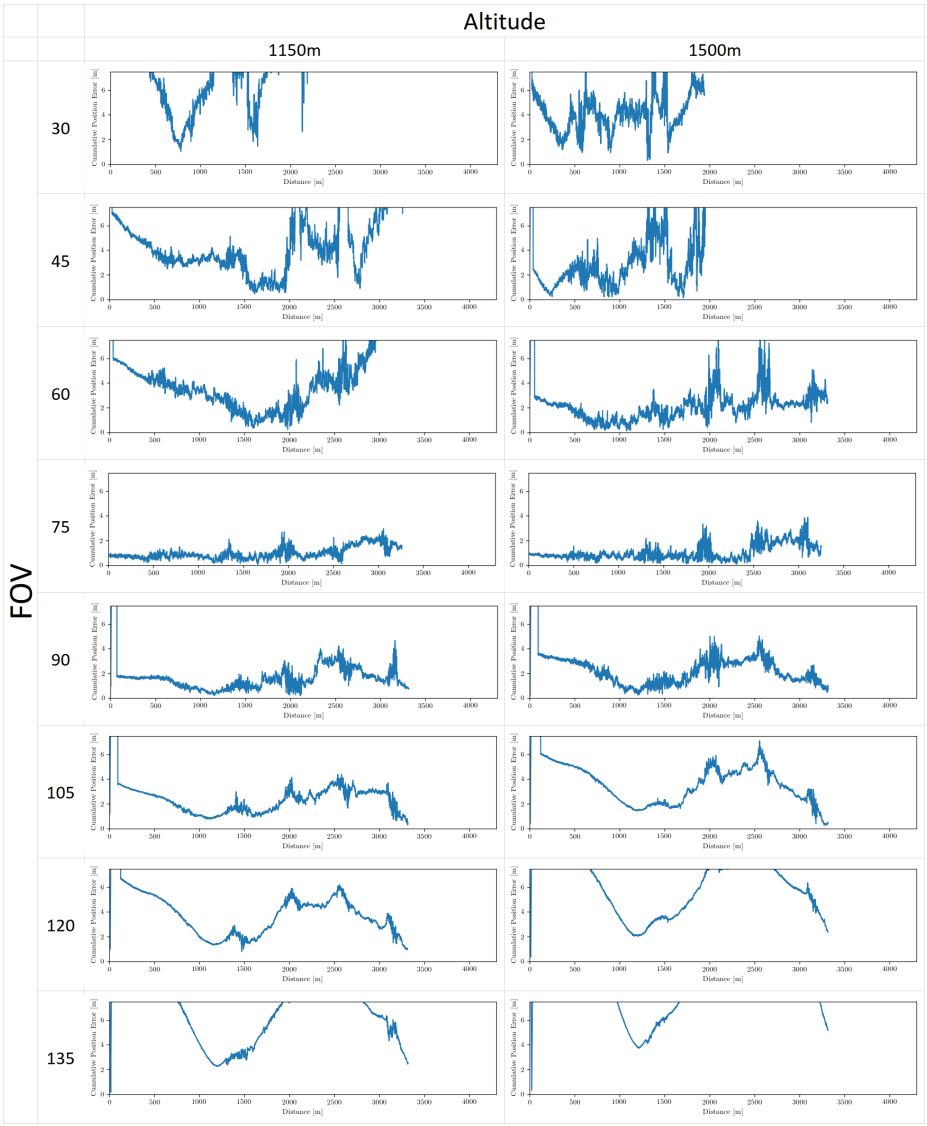
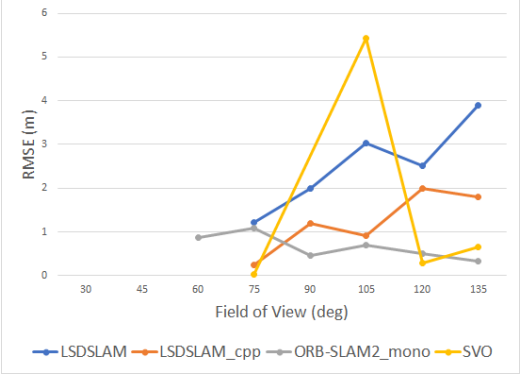


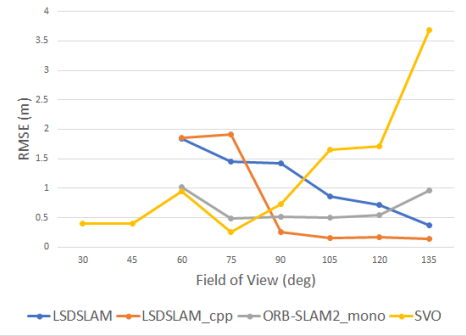
Figure 18: SVO error accumulation graphs for dataset group 1 altitudes 1150m and 1500m, showing total error accumulated at various altitudes and camera FOVs.

Altitude (m)	FOV (deg)	Position RMSE (m) / Maximum Successful Bank Angle Turn (deg)			
		LSDSLAM	LSDSLAM_cpp	ORBSLAM2_mono	SVO
100	30	X	X	X	X
	45	X	X	X	X
	60	X	X	0.860 / 30	X
	75	1.211 / 0	0.236 / 0	1.095 / 30	0.029 / 0
	90	2.003 / 25	1.190 / 10	0.455 / 30	X
	105	3.030 / 30	0.907 / 10	0.699 / 30	5.428 / 0
	120	2.506 / 30	1.998 / 30	0.502 / 30	0.292 / 0
	135	3.890 / 30	1.793 / 30	0.330 / 30	0.660 / 0
450	30	X	X	X	0.399 / 0
	45	X	X	X	0.400 / 0
	60	1.837 / 25	1.860 / 30	1.026 / 30	0.954 / 25
	75	1.451 / 30	1.909 / 30	0.493 / 30	0.256 / 25
	90	1.426 / 30	0.257 / 30	0.521 / 30	0.736 / 10
	105	0.862 / 30	0.153 / 30	0.498 / 30	1.660 / 25
	120	0.718 / 30	0.163 / 30	0.548 / 30	1.715 / 25
	135	0.372 / 30	0.136 / 30	0.956 / 30	3.688 / 10
800	30	X	X	X	0.631 / 0
	45	3.834 / 20	0.762 / 5	X	7.316 / 25
	60	1.507 / 25	0.509 / 25	3.301 / 30	1.320 / 25
	75	2.471 / 30	0.508 / 30	0.647 / 30	0.630 / 25
	90	0.848 / 30	0.422 / 30	0.793 / 30	1.232 / 25
	105	0.642 / 30	0.263 / 30	0.778 / 30	1.821 / 25
	120	0.412 / 30	0.170 / 30	1.184 / 30	2.963 / 25
	135	0.322 / 30	0.240 / 30	1.085 / 30	5.444 / 25
1150	30	1.739 / 10	2.165 / 10	X	11.268 / 15
	45	2.140 / 20	0.436 / 5	X	4.876 / 25
	60	2.271 / 25	1.534 / 25	4.839 / 30	4.524 / 25
	75	0.927 / 30	0.366 / 30	0.875 / 30	0.964 / 25
	90	0.839 / 30	0.325 / 30	1.467 / 30	2.229 / 25
	105	0.349 / 30	0.207 / 30	1.257 / 30	3.375 / 25
	120	0.447 / 30	0.134 / 30	1.215 / 30	5.619 / 25
	135	0.425 / 30	0.307 / 30	1.835 / 30	10.386 / 25
1500	30	X	X	X	4.311 / 10
	45	5.538 / 20	3.924 / 25	X	3.138 / 10
	60	1.920 / 30	0.717 / 25	5.385 / 30	2.454 / 25
	75	1.446 / 30	0.824 / 30	1.617 / 30	1.027 / 25
	90	0.740 / 30	0.340 / 30	1.752 / 30	3.313 / 25
	105	0.343 / 30	0.224 / 30	1.834 / 30	5.404 / 25
	120	0.384 / 30	0.180 / 30	1.804 / 30	9.277 / 25
	135	0.504 / 30	0.380 / 30	2.317 / 30	17.072 / 25

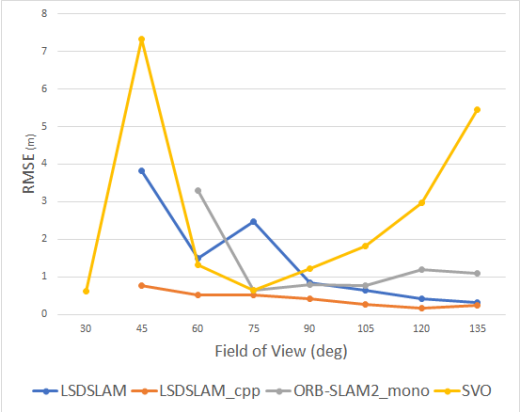
Table 2: Altitude vs FOV results using dataset group 1. RMSE and the maximum bank angle turn reached by each algorithm is reported. Reaching the 30 degree turn indicates that the algorithm successfully tracked the entire dataset. An X denotes when the algorithm failed to initialize for that dataset. These statistics only take the portion of the trajectory that was successfully tracked into account. (e.g. if an algorithm failed on the 10 degree turn of a dataset, only the error statistics up to that turn are reported.) Green boxes highlight the best RMSE for each altitude/FOV combination.



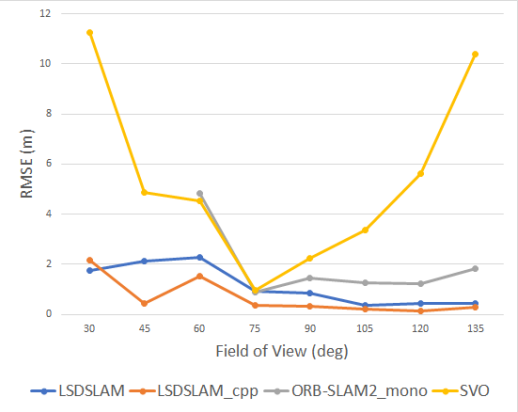
(a) 100m altitude



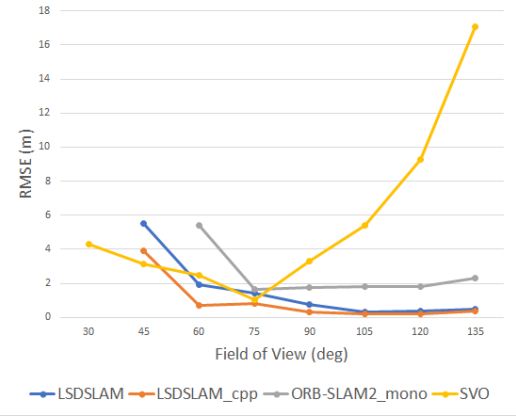
(b) 450m altitude



(c) 800m altitude

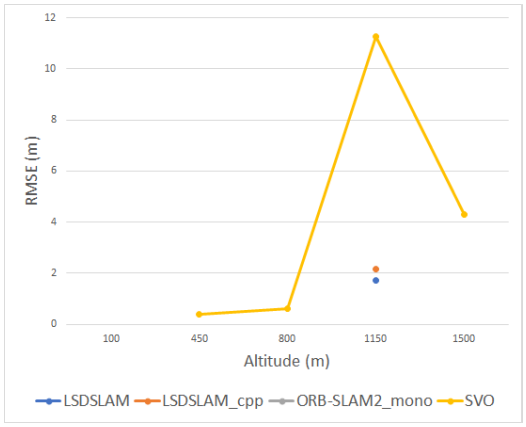


(d) 1150m altitude

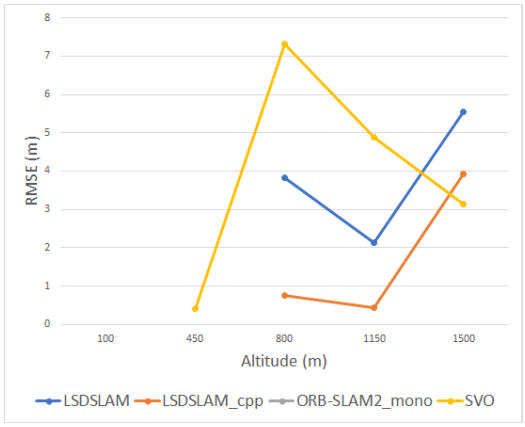


(e) 1500m altitude

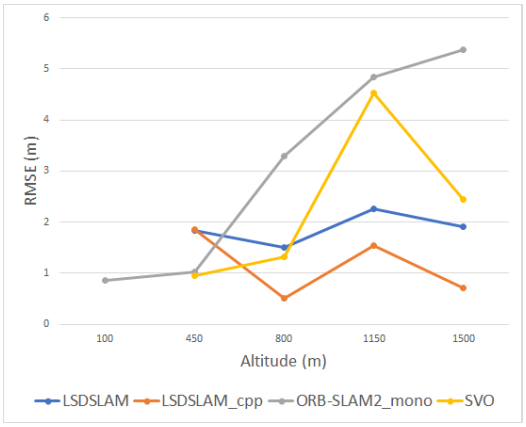
Figure 19: Field of View vs RMSE for each dataset at each altitude.



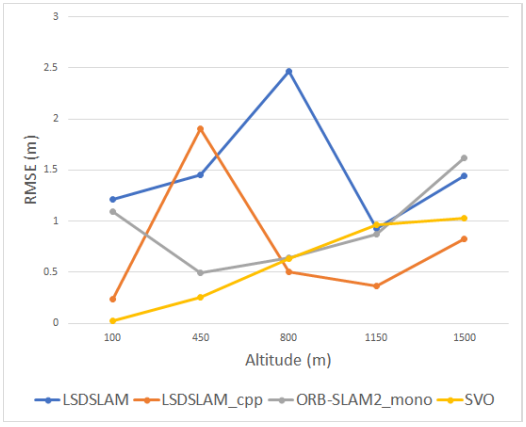
(a) 30 degree Field of View



(b) 45 degree Field of View

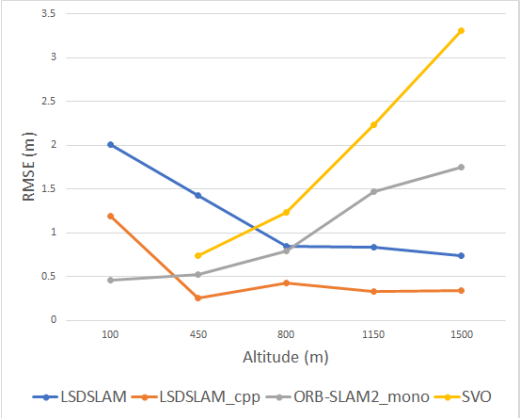


(c) 60 degree Field of View

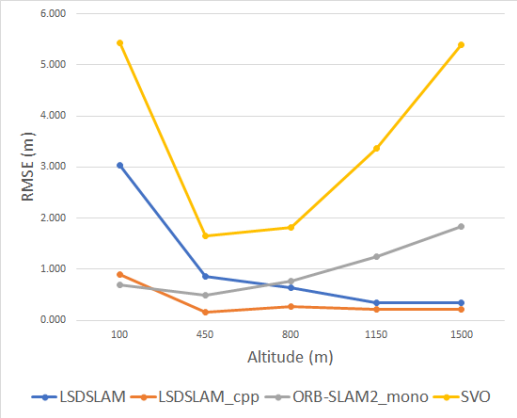


(d) 75 degree Field of View

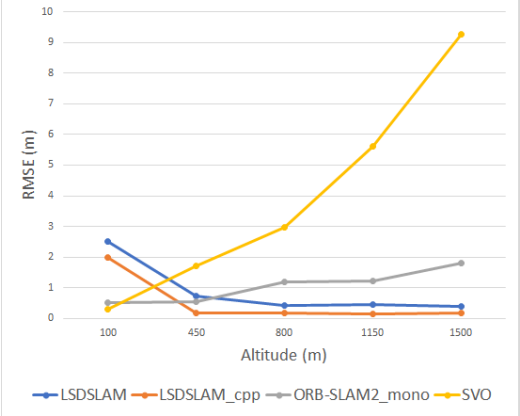
Figure 20: Altitude vs RMSE for 30 - 75 degree Field of View.



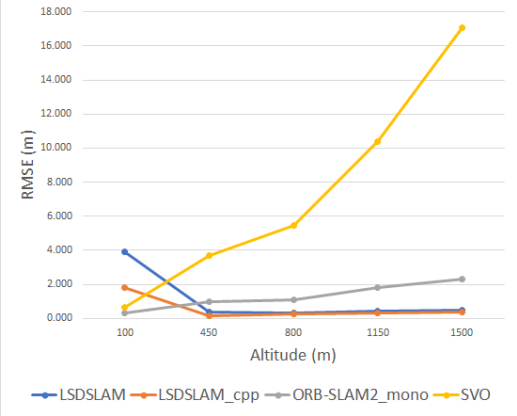
(a) 90 degree Field of View



(b) 105 degree Field of View



(c) 120 degree Field of View



(d) 135 degree Field of View

Figure 21: Altitude vs RMSE for 90-135 degree Field of View.

		Position RMSE (m) / Maximum Successful Bank Angle Turn (deg)							
Algorithm:		LSDSLAM		LSDSLAM_cpp		ORB-SLAM2		SVO	
Time to Bank:		1.5s	3s	1.5s	3s	1.5s	3s	1.5s	3s
FOV (deg)	30	X	X	X	X	X	X	X	X
	45	X	X	X	X	X	X	X	0.524 / 10
	60	1.987 / 0	1.898 / 0	1.803 / 0	1.829 / 0	0.823 / 60	0.975 / 60	0.105 / 0	0.097 / 0
	75	2.923 / 45	3.213 / 45	2.268 / 45	2.441 / 45	0.850 / 60	0.681 / 60	0.791 / 20	1.608 / 35
	90	0.836 / 45	1.306 / 45	0.254 / 45	1.392 / 45	0.715 / 60	0.637 / 60	0.475 / 45	1.664 / 45
	105	0.725 / 50	0.872 / 50	0.425 / 50	0.284 / 60	0.733 / 60	0.941 / 60	0.786 / 45	0.667 / 45
	120	0.456 / 55	0.673 / 60	0.171 / 60	0.144 / 60	1.075 / 60	1.305 / 60	0.543 / 45	0.584 / 45
	135	1.239 / 60	1.062 / 60	0.344 / 55	0.388 / 60	1.505 / 60	1.398 / 60	0.565 / 45	0.621 / 45

Table 3: Algorithm RMSE results when tested on dataset group 2. RMSE and the Maximum Bank Angle Turn reached by each algorithm is reported. Reaching the 60 degree turn indicates that the algorithm successfully tracked the entire dataset. "X"s indicate where the algorithm failed to initialize. Green boxes highlight the best RMSE for each altitude/FOV combination.

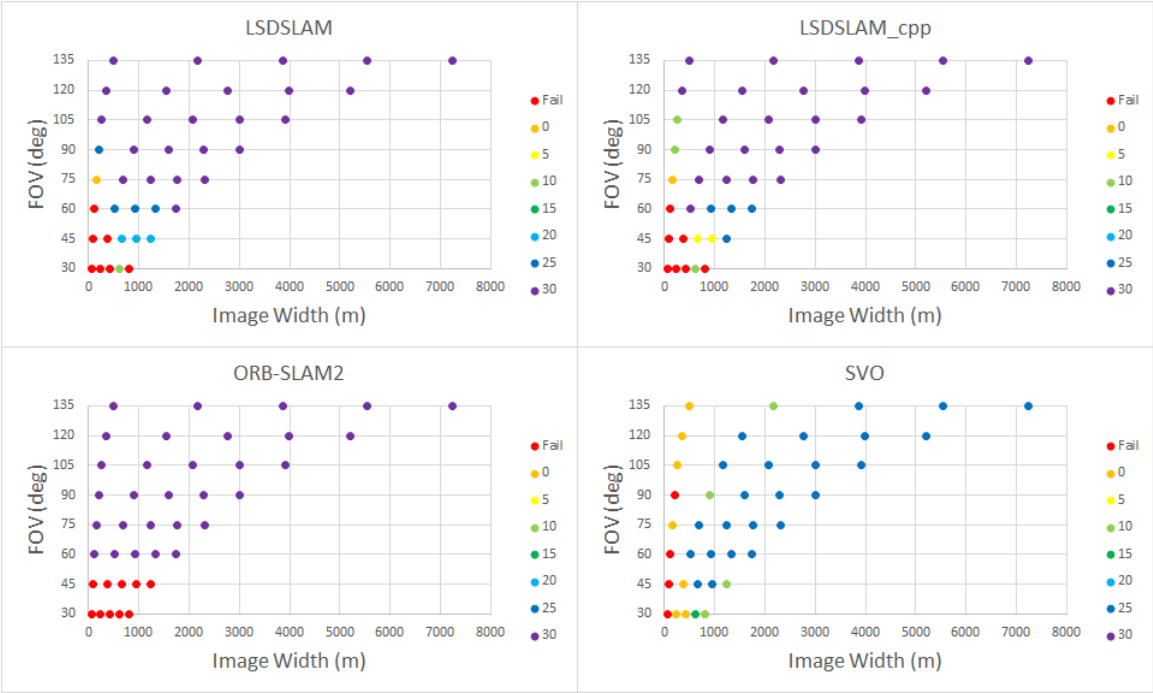


Figure 22: Image Width vs FOV - Bank Angle Completion. Graphs show where each algorithm failed, if applicable, for a given Image Width and FOV.

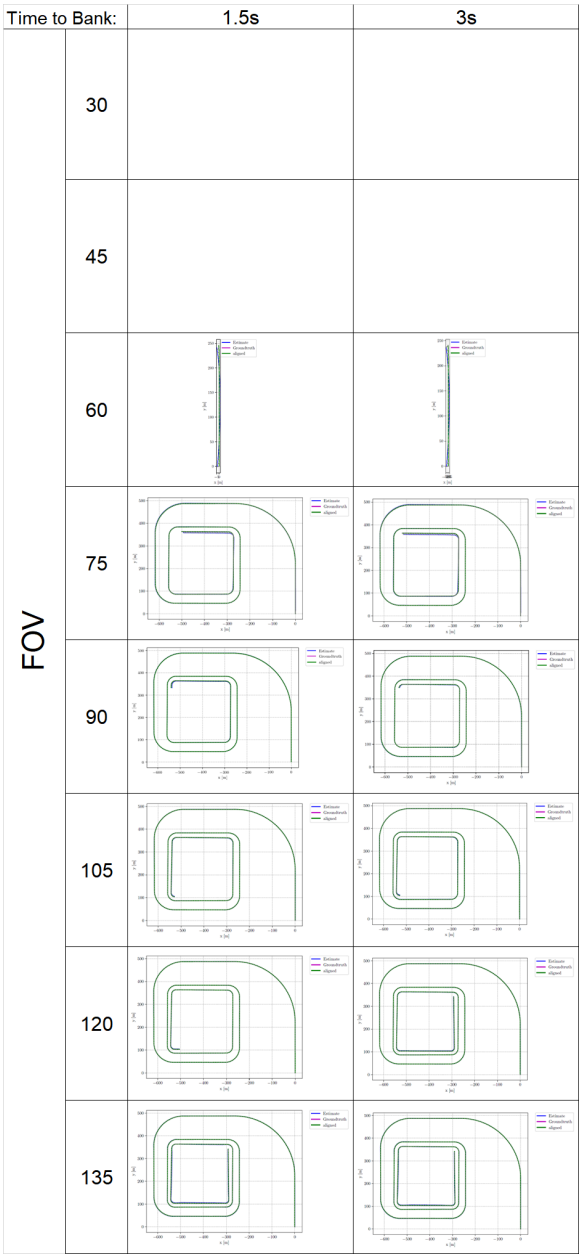


Figure 23: LSDSLAM results using dataset group 2 to test the effect of aircraft bank angle vs bank rate. Blank boxes indicate where the algorithms failed to initialize.

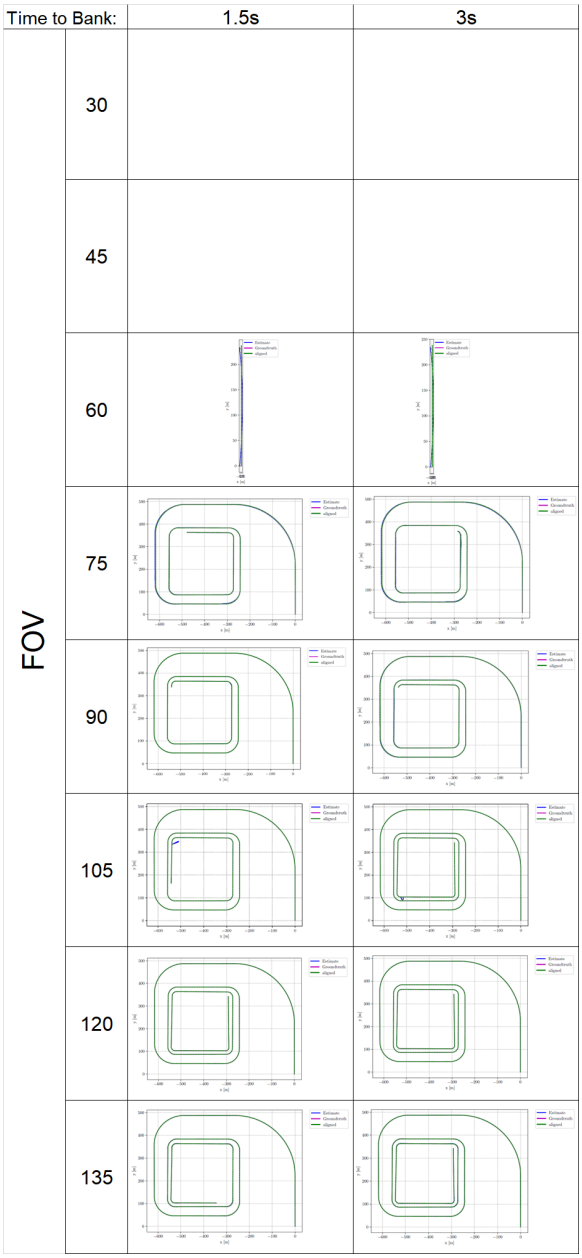


Figure 24: LSDSLAM_cpp results using dataset group 2 to test the effect of aircraft bank angle vs bank rate. Blank boxes indicate where the algorithms failed to initialize.

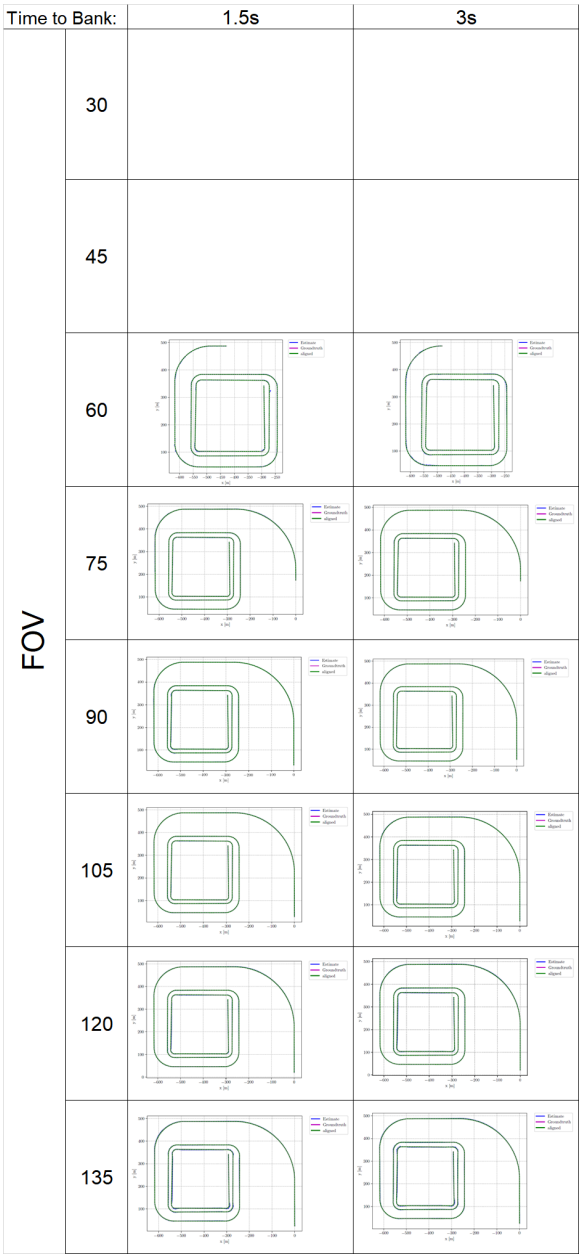


Figure 25: ORB-SLAM2 results using dataset group 2 to test the effect of aircraft bank angle vs bank rate. Blank boxes indicate where the algorithms failed to initialize.

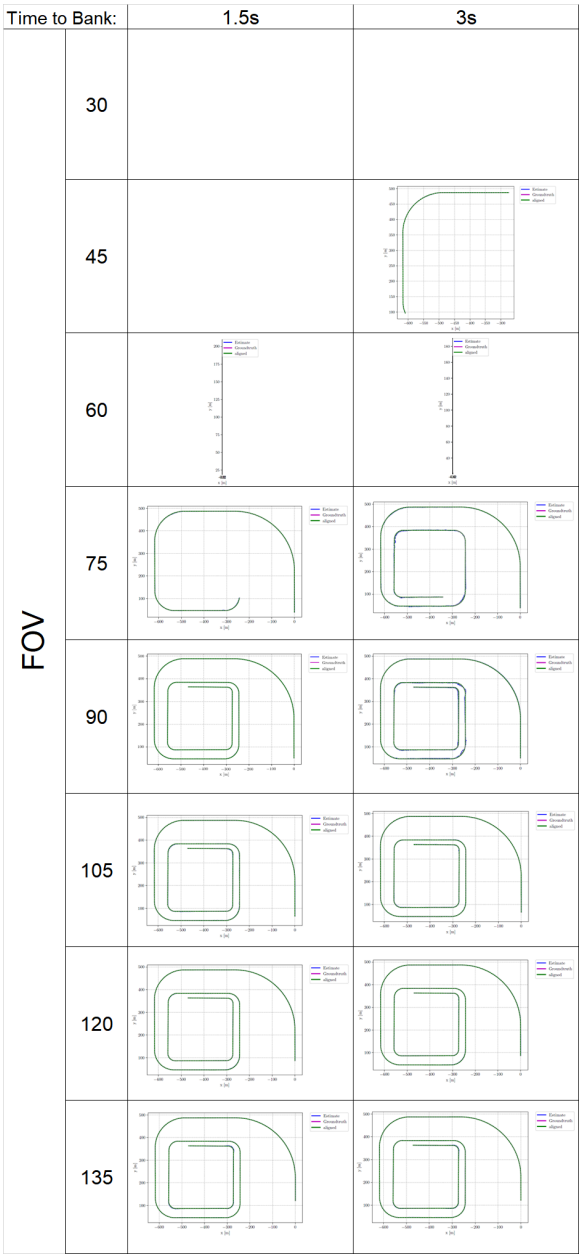
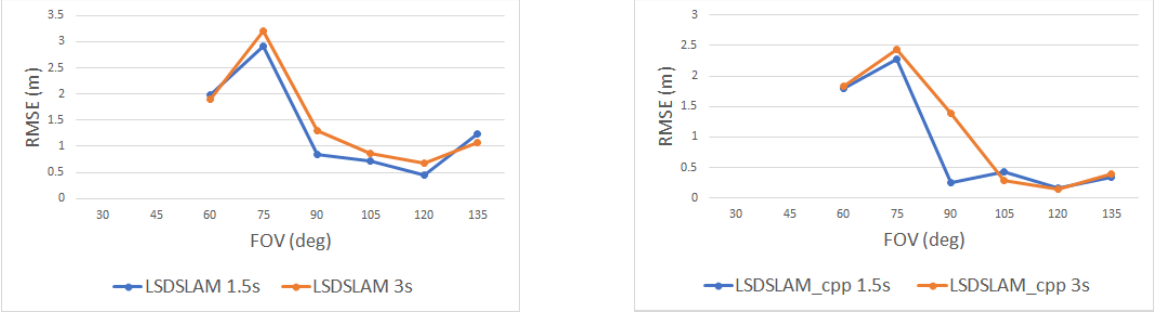


Figure 26: SVO results using dataset group 2 to test the effect of aircraft bank angle vs bank rate. Blank boxes indicate where the algorithms failed to initialize.



(a) LSDSLAM results

(b) LSDSLAM_cpp results

(c) ORB-SLAM2 results

(d) SVO results

Figure 27: Field of View vs RMSE for each dataset in group 2.

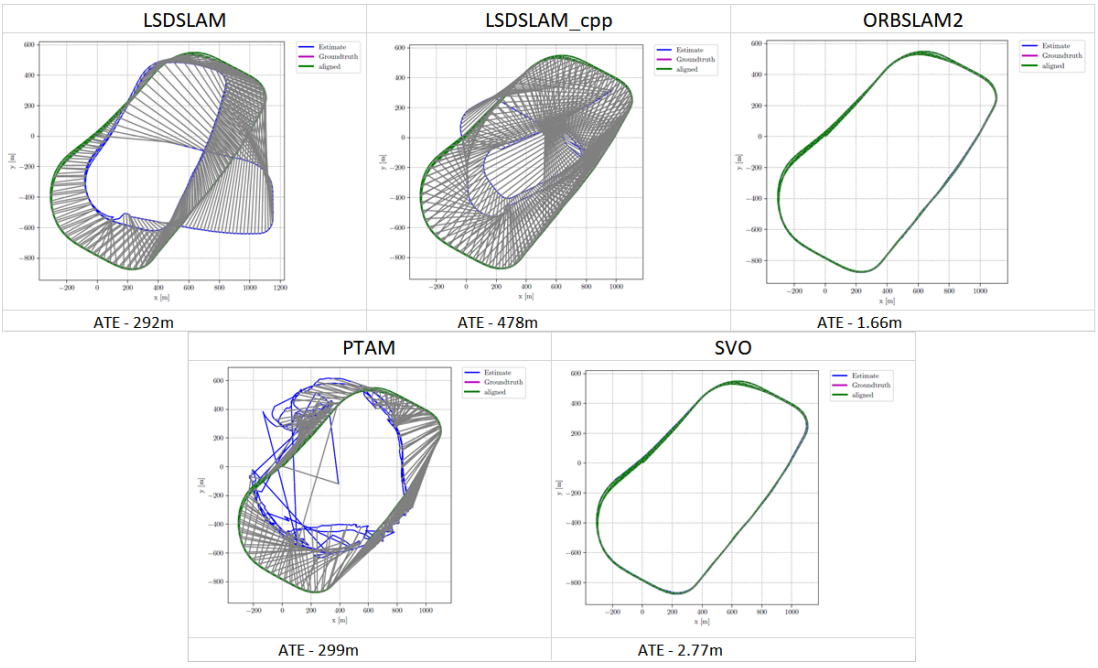


Figure 28: Position solutions of each algorithm on the simulated SUSEX Flight 12 dataset.

4.2 Analysis

This section analyzes the results produced in this research. It looks at the impact of each of the variables tested, including algorithm choice, aircraft altitude, bank angle, bank rate, camera FOV, and any interdependencies between these.

4.2.1 Impact of FOV and Image Width

Camera FOV in general had the largest impact on each algorithm. At 30 and 45 degree FOV, every algorithm failed to completely track the datasets, with only SVO and LSDSLAM able to initialize on some of the datasets. This is shown in Fig. 7 through Fig. 18. With this narrow FOV, ORB-SLAM2 failed to initialize, LSDSLAM and LSDSLAM_cpp only initialized on 4 of 13 datasets, and SVO did the best by initializing on 10 of the 13, though 5 of those failed after the first straight section of the dataset. The remaining 5 datasets had a larger RMSE than higher FOV datasets, showing that even for SVO, the small field of view caused a significant degradation in the performance. At 60 degrees FOV, all of the algorithms started to track successfully, as can be seen in Table 2. All of the algorithms tended to be most accurate at 75 degree FOV and above, although SVO dropped in accuracy at 120 degrees and above, as shown in Fig. 19. FOV also showed an inter-dependency with bank rate and bank angle, which is explored in the next subsections.

To make sure the effect of FOV was truly uncoupled from the effect of Altitude, FOV was also compared to Image Width in meters, as shown in Fig. 22. In this chart it can be seen that ORB-SLAM2 is purely dependent on FOV, only failing to initialize when the FOV is 45 degrees or below. SVO, on the other hand, looks to be almost entirely dependent on the Image Width, only failing to track the dataset to completion when the Image Width is less than 500 meters. It did have a slight dependence on FOV being above 30 degrees, though it is within the margin of error.

Finally, LSDSLAM and LSDSLAM_cpp showed mixed dependence between FOV and Image Width. Below about 260m Image Width they both struggled to initialize or fully complete the datasets. Above 260m, they seemed to be FOV dependent, with FOVs of 60 degrees and below either failing to initialize, or failing to track the entire dataset. Above 260m Image Width, higher FOVs were consistently able to track the entire dataset, even with similar Image Widths as lower FOV datasets that had failed early.

4.2.2 Impact of Bank Angle vs FOV

Depending on the specific algorithm, the bank angle used during the 90 degree turns had a significant effect on the performance of the algorithm. While ORB-SLAM2 showed no dependence on the bank angle, LSDSLAM and LSDSLAM_cpp show a dependency between the aircraft's bank angle and camera FOV, namely failing when the bank angle exceeds more than half of the camera's FOV, as seen in Table 2 and Table 3. SVO seems to show the same relationship as LSDSLAM, but is inconclusive due to SVO being unable to track the full dataset, as discussed in the algorithm choice section below.

4.2.3 Impact of Bank Rate vs FOV

Dataset group 2 was used to test the effect of bank rate, as seen in Fig. 23 through Fig. 26. Namely, if the algorithm consistently tracks more turns when the time to achieve a certain bank angle is doubled, then we can assume the algorithm is sensitive to bank rate. The RMSE results can be seen in Table 3.

LSDSLAM tended to fail on the same turn in both of its implementations, although it did have two pairs where the slower bank rate tracked an extra turn. It also had a pair of datasets where the sharper bank rate went an extra turn, so we

assume that the differences are more due to random perturbations than algorithmic attributes. Overall, LSDSLAM did not seem to have a dependence on the bank rate itself.

ORBSLAM2 was also independent of bank rate, with near identical behavior between the the slower and faster datasets.

SVO seemed to have a tenuous dependence on bank rate, with the 75 degree dataset failing when the bank rates were similar instead of the same bank angle turn, as seen in Table 3 and Fig. 26. As SVO failed prematurely, however, that is the only dataset pair that had the range to show this result, and is thus taken to be inconclusive.

As shown by Fig. 27 and Table 3, the accuracy of each algorithm was not noticeably impacted by the change in bank rate, showing similar RMSE across algorithms.

4.2.4 Impact of Altitude

The impact of altitude on the VSLAM algorithms is highly dependent on the algorithm. ORB-SLAM2 appeared to be fairly altitude agnostic, with only a small effect on RMSE as seen in Table 2. Namely, ORB-SLAM2 seems to be the most accurate at lower altitudes with a high FOV. as seen in Fig. 20 and Fig. 21.

LSDSLAM and LSDSLAM_cpp have a slightly worse RMSE at 100 meters than the rest of the altitudes, failing altogether at 100m and 60 degrees FOV, as shown in Fig. 7 and Fig. 8. LSDSLAM is slightly more accurate at higher altitudes and higher FOVs, while LSDSLAM_cpp is slightly more accurate at 450-800m and high FOVs, as seen in Fig. 20 and Fig. 21. The difference in accuracy is relatively small, however. LSDSLAM and LSDSLAM_cpp also were able to track further into the dataset as the altitude increased, shown in Table 2.

SVO is drastically effected by altitude, as shown in Fig. 10, where it struggles

to track in the 100m and 450m datasets. This is a similar result to the research by Kim [6], where SVO failed to track below 300m altitude. On the other hand, the accuracy of SVO tended to be better at lower altitudes, as seen in Table 2, though this may be due to the algorithm failing earlier, thus only including the error for the first straight section. SVO also tended to track more turns in each dataset as the altitude increased, as shown in Table 2, though this is explained by the Image Width results mentioned earlier.

4.2.5 Real World vs Simulation

The algorithms were unable to track either the real world or simulated SUSEX Flight 12 dataset with only a 25 degree FOV. This reinforces the previous results of algorithms generally failing on datasets with an FOV of below 60 degrees. When the simulated dataset FOV is increased to 90 degrees, LSDSLAM, LSDSLAM_cpp, ORB-SLAM2, PTAM, and SVO were able to initialize, as shown in Fig. 28. PTAM lost tracking almost immediately, degrading into noise that follows a vaguely loop shape. LSDSLAM tracks most of the dataset, only failing on the last loop, which throws off the trajectory alignment. LSDSLAM_cpp similarly fails, throwing off the alignment even further. ORB-SLAM2 and SVO both track the loop completely, with ORB-SLAM2 being slightly more accurate.

These results suggest that use on a real world dataset is plausible, if given a wider FOV camera and a robust algorithm.

4.2.6 Algorithm Choice

As shown in the previous sections, the different algorithms tested had wildly different results. MonoSLAM, PTAM, and OKVIS failed entirely, and are thus considered to be poor candidates for aerial vehicles. SVO seemed rather robust, initializing even

at some of the lowest FOVs, however proved to be less accurate than the other algorithms. SVO also seemed to have a memory leak, as it failed in each dataset at the same point after linearly consuming all 24 GB of the test system's RAM. It is unknown whether this memory leak is a result of the SVO algorithm itself, or if it is a result of SVO's implementation within the SLAMBench2 benchmarking software. Unfortunately, this memory leak served to cut off all higher bank angle turns in each dataset, leading to inconclusive results for the algorithm. This can be seen in Table 2, where the maximum bank angle turn reached is 25 degrees, as well as in Table 3, where the maximum bank angle turn reached is 45 degrees. ORB-SLAM2 also seemed to be robust, tracking all datasets as long as the camera FOV was 60 degrees or above, and performing quite well at 90 degrees and above. LSDSLAM proved to be the most accurate algorithm, with both implementations having a lower RMSE than the other algorithms. LSDSLAM, however, proved to be the least robust, with a strong dependency on the bank angle staying under half of the camera's FOV. LSDSLAM was also sensitive to it's implementation, with LSDSLAM_cpp yielding more accurate results, as shown in Table 2 and Table 3.

4.2.7 Impact of Tuning Parameters

As discussed in Chapter 2, many of these algorithms do have tuning parameters that can be changed to help improve performance. This research takes all parameters at their default values for the above results, but some work was done to determine whether or not the parameters would drastically change the results. Through tweaking of all available parameters for LSDSLAM, ORB-SLAM2, and SVO, it was found that changing these parameters did not change the initialization results for any of the algorithms (i.e. none of the algorithms that failed to initialize under default parameters would initialize when the parameters were changed). Of the algorithms that did

initialize, RMSE was minimally affected, having a variance of less than a meter, even with drastic parameter changes. It should be noted that this parameter search was not exhaustive, and is considered a limitation of this research and a subject for future study.

4.3 Summary

This chapter provided the results of both dataset groups, as well as the simulated SUSEX Flight 12 dataset. It then analyzed these results, showing the impact of camera FOV, Image Width, aircraft altitude, bank angle, bank rate, and choice of algorithm on each of the results. Overall, it was shown that LSDSLAM, ORB-SLAM2, and SVO were all viable for aerial vehicle navigation, with each having different strengths and weaknesses. All algorithms strongly depend on the FOV of the capturing camera, failing entirely when the FOV is below 60 degrees, and performing best at a 75 degree FOV. LSDSLAM was found to be the most accurate algorithm, but was less robust, with a strong dependence on aircraft bank angle. ORB-SLAM2 was found to be the most robust algorithm as long as the FOV was above 45 degrees, but was less accurate than the other algorithms. SVO was found to be robust at lower FOVs, as well as accurate at higher altitudes, but required an altitude of at least 450m, and suffered from a memory leak of unknown origins, preventing further analysis.

V. Conclusions

This chapter summarizes this research. It reviews and discusses the results obtained, as well as the key contributions of this work. It concludes with suggestions for future work in this area.

5.1 Results Discussion

Six current monocular Visual Simultaneous Localization and Mapping (VSLAM) algorithms, MonoSLAM, PTAM, OKVIS, LSDSLAM, ORB-SLAM2, and SVO, were tested on 59 datasets designed to determine the algorithms viability for use on aerial vehicle datasets, as well as their dependence on camera field of view (FOV), aircraft altitude, bank angle, and bank rate. They were also tested against a real world SUSEX flight dataset, along with two corresponding simulated datasets, to determine if the simulated results are indicative of the real world scenario.

It was found that MonoSLAM, PTAM, and OKVIS were not viable for these datasets, as they could not initialize tracking consistently. All other algorithms either failed entirely or the solutions were extremely degraded when the camera FOV was below 60 degrees, with the algorithms becoming most accurate around 75 degrees FOV, as shown in Fig. 19. LSDSLAM_cpp was found to the most accurate, ass seen in Table 2, but reliant on the aircraft bank angle staying under half of the camera's FOV to maintain tracking. ORB-SLAM2 was found to be robust, not dependent on altitude, bank angle, or bank rate, but did require at least a 60 degree camera FOV to track at all. It was also found to be slightly less accurate than the other algorithms. SVO was found to of middling accuracy, being more robust at low camera FOVs than the other algorithms, but required a minimum altitude of 450m to track and a minimum Image Width of 500m to perform well. SVO's dependence on the bank rate

and angle were inconclusive due to a memory leak in the SVO implementation.

Across all algorithms, tracking quality was either completely broken or extremely degraded when the camera FOV was below 60 degrees. The quality of the tracking solution tended to be best around 75 degrees FOV and above, though extremely wide angles such as 120 degrees degraded some of the tracking solutions.

5.2 Future Work

There are many opportunities for further research in the area of using VSLAM on aerial vehicles. First and foremost, confirming the conclusions of this research on real-world datasets would further contribute to this field. While this work strives to be applicable to real world implementations, the simulated environment does purposely provide ideal conditions in order to isolate the variables of interest.

More testing could also be done with more modern VSLAM algorithms not included in SLAMBench2, which may give benefits to both accuracy and robustness as the field has advanced rapidly within the last few years. A promising field that is emerging is the use of machine learning in VSLAM algorithms. This could potentially allow the algorithms to be more robust and adapt to the operating environment.

A wider look at the effect of tuning parameters could be performed, to conclusively determine how each parameter effects the performance on this use-case. This would have to be done on a per-algorithm basis, preferably focusing on the most promising algorithms to help improve performance.

The effect of including other sensors, such as inertial measurement unit (IMU) or ground facing LIDAR measurements, could be analysed using algorithms such as VINS Fusion [25]. Adding other sensors could increase the robustness of the tracking, especially during large bank angle turns. Similarly, different camera arrangements could be tested for effectiveness, such as having multiple cameras at different poses,

or even having a full 360 degree spherical image to allow ground tracking at any aircraft pose. This could potentially also allow for more advanced features, such as extracting the rotational information of the aircraft from horizon tracking.

The performance of these various algorithms on different hardware is also a line of inquiry that could be followed, namely focusing on hardware that could readily be adapted for use in aerial vehicles. This would determine the viability of different algorithms, as only algorithms that can be run in real time on reasonable hardware can be seriously considered for implementation.

Current algorithms could be analyzed for their uncertainty, to test the actual accuracy of each algorithm and confirm if uncertainty figures given in their original papers hold under the conditions presented in the aerial vehicle navigation case.

Eventually, these algorithms will also need to be tested for performance under different environmental conditions aircraft might face, such as the introduction of clouds and other weather, different geological locations and the features they might present, and even time of day. To be a viable navigation technique, the VSLAM algorithms will need to be able to provide tracking over a wide variety of these conditions and environments.

Bibliography

1. Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015.

2. Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 2017.

3. Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8690 LNCS, pages 834–849. 2014.

4. Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, dec 2016.

5. Daniel J. Carson, John F. Raquet, and Kyle J. Kauffman. Aerial Visual-Inertial Odometry Performance Evaluation. In *Proceedings of the ION 2017 Pacific PNT Meeting, Honolulu, Hawaii, May 2017*, pages 137–154. Air Force Institute of Technology, jun 2017.

6. Kyung M Kim. MONOCULAR VISUAL ODOMETRY FOR FIXED-WING SMALL UNMANNED AIRCRAFT SYSTEMS. Master's thesis, Air Force Institute of Technology, 2019.

7. Benjamin M Fain. *SMALL FIXED-WING AERIAL POSITIONING USING INTER-VEHICLE RANGING COMBINED WITH VISUAL ODOMETRY*. PhD thesis, Air Force Institute of Technology, 2017.

8. Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

9. Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

10. Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334, 2015.

11. Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2014.

12. Scott Nykl, Chad Mourning, Mitchell Leitch, David Chelberg, Teresa Franklin, and Chang Liu. An overview of the STEAMiE Educational game engine. *Proceedings - Frontiers in Education Conference, FIE*, (November), 2008.

13. Bruno Bodin, Harry Wagstaff, Sajad Saeedi, Luigi Nardi, Emanuele Vespa, John H Mayer, Andy Nisbet, Mikel Luján, Steve Furber, Andrew J Davison, Paul H. J. Kelly, and Michael O'Boyle. SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM. aug 2018.

14. Zichao Zhang and Davide Scaramuzza. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. *IEEE International Conference on Intelligent Robots and Systems*, pages 7244–7251, 2018.

15. Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, mar 2004.

16. Satoshi Fujimoto, Zhencheng Hu, Roland Chapuis, and Romuald Aufrère. ORB-SLAM map initialization improvement using depth. *Proceedings - International Conference on Image Processing, ICIP*, 2016-Augus:261–265, 2016.

17. Ankur Handa, Thomas Whelan, John Mcdonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1524–1531, 2014.

18. Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. *IEEE International Conference on Intelligent Robots and Systems*, 2012.

19. Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *International Journal of Robotics Research*, 35(10):1157–1163, 2016.

20. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

21. Richard A Newcombe, David Molyneaux, David Kim, Andrew J Davison, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion : Real-Time Dense Surface Mapping and Tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 127–136, 2011.

22. Ben Glocker, Renato Salas Moreno, Thomas Whelan, Andrew Davison, and Stefan Leutenegger. ElasticFusion: Dense SLAM Without A Pose Graph. *Robotics: Science and Systems XI*, 2016.

23. Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1241–1250, 2015.

24. Gary Ellingson, Kevin Brink, and Tim McLain. Relative visual-inertial odometry for fixed-wing aircraft in GPS-denied environments. *2018 IEEE/ION Position, Location and Navigation Symposium, PLANS 2018 - Proceedings*, pages 786–792, 2018.

25. Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors. 2019.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2020	Master's Thesis	Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE

Comparison of Visual Simultaneous Localization and Mapping Methods for Fixed-Wing Aircraft Using SLAMBench2

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Latcham, Patrick R., 2d Lt USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-20-M-034

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Visual Simultaneous Localization and Mapping (VSLAM) algorithms have evolved rapidly in the last few years, however there has been little research evaluating current algorithm's effectiveness and limitations when applied to tracking the position of a fixed-wing aerial vehicle. This research looks to evaluate current monocular VSLAM algorithms' performance on aerial vehicle datasets using the SLAMBench2 benchmarking suite. It does so by using simulated datasets generated in the AftrBurner Engine to test the quality of each algorithm's tracking solution, as well as finding any dependence on camera Field of View (FOV), aircraft altitude, bank angle, and bank rate.

15. SUBJECT TERMS

Visual Simultaneous Localization and Mapping, Visual Odometry

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Clark N. Taylor, AFIT/ENG
U	U	U	UU	73	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, ext 4614; Clark.Taylor@afit.edu