UNIVERSITY OF CALIFORNIA
SANTA CRUZ

ANALYSES AND APPLICATIONS OF VISUAL SLAM FOR UAS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Gordon H. Keller

March 2025

The thesis of Gordon H. Keller
is approved:

Professor Mircea Teodorescu, Chair

Professor Gabriel Elkaim

Dr. Jonathan Glen

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Analyses and Applications of Visual SLAM for UAS

by

Gordon H. Keller

The ubiquity of Unoccupied Aerial Systems/Vehicles (UAS/V) in society has spurred many different functional purposes for them. In many of these applications, including visual sensing is advantageous for precise control and situational awareness. Visual "Simultaneous Localization and Mapping", or Visual SLAM, has come to the fore as an effective approach for proprioception in UAS. In this thesis, two explorations on the relationship between Visual SLAM and UAS are offered: (1) an in-depth analysis of the performance of the algorithm when flown in a sparse feature space onboard a multicopter platform, and (2) the design and simulation of an application example of Visual SLAM in a terrain-following modality of flight. Through these contributions, the viability and reliability of Visual SLAM is validated for practical use on small aerial vehicles.

Acknowledgments

Firstly, I would like to thank my graduate advisor, Mircea Teodorescu. He was an ever-patient and kind supporter throughout my graduate school experience and stuck with me through life's ups and downs. My soft skills and technical skills alike were greatly influenced and facilitated by him. He helped me to start on my journey as a confident and capable engineer. I also would like to extend my gratitude to Jonathan Glen and Gabriel Elkaim for being a part of this journey and for being a part of the reading committee.

Secondly, I would like to thank my family. I thank my father, Steve, who was my rock in writing this thesis and helped me repeatedly over the course of the program. I thank my mother, Erin, who supported me greatly through thick and thin. Her emotional and spiritual guidance were invaluable in crossing the finish line with school. I thank my siblings (Duncan, Griffin, Fiona, Francie, and Henry) for also being there for me and for helping me rally when times were tough.

Finally, I would like to thank my friends, new and old. All of the experiences with you have made me who I am, and I am grateful to know that so many of you will be life-long connections.

Chapter 1

Introduction

1.1 Motivation

The motivation behind this thesis is to better understand the effectiveness of ORB-SLAM2 onboard a UAS system. The requirements of UAS functionality in various capacities necessitate visual sensing. The ORB-SLAM family of algorithms has shown great promise in a multitude of applications, including use aboard UAS. However, in order to be sure that it can pass muster in UAS applications, studies should be undertaken to better understand how the motions of multicopter systems have an impact on the localization effectiveness and overall robustness. With the completion of such studies, applying the technology towards tangible applications is possible.

1.2 Contributions

The two primary contributions of this work are:

- A thorough investigation into the effects of multicopter motion on the effectiveness of the monocular ORB-SLAM2 algorithm in a feature-sparse indoor environment.
- A novel "Vision-assisted Terrain Following" algorithm and simulation which incorporates the use of stereo ORB-SLAM2 in a feature-rich outdoor environment.

1.3 Background

1.3.1 Advent of UAS

Unoccupied Aerial Systems (UAS), also known as Unoccupied Aerial Vehicles (UAV), have radically changed the ways in which we interact with the world. Brought about by the advancement of aeronautical control knowledge, the miniaturization of electronics, and improvements to battery technologies, UAS have enabled new modes of operation throughout many different disciplines. From defense to civilian use, for recreation and work alike, UAS have fundamentally reshaped how people utilize aircraft remotely to have fun and to get work done.

UAS are very commonly found used in research disciplines as they allow for expansive operations, efficiency with personnel use, and a wide variety of vehicle classes for different sized payloads or operational requirements. Archeological efforts employ them [11] [2], as do forestry efforts [68] [53], agricultural research [27] [69], marine studies [20] [6], and many more disciplines. The third chapter of this work will focus on their use in the geosciences.

1.3.2 Vision Aboard UAS

To enable the next generation of UAS, incorporating the ability to know how they are situated and how to traverse the airspace via direct environment sensing is paramount, and seamless integration into arbitrary environments (urban, suburban, rural, etc.) is only possible with several layers of assured localization accuracy and situational awareness (on top of complementary control algorithms). Use of digital cameras or other electo-optical devices enables a vision component to be employed and can constitute one such sensor component. Modern use of cameras and computer vision aboard small UASs is common – for instance, many commercial systems utilize optical flow for position hold and velocity estimation. Complete environment awareness via computer vision techniques is not yet fully realized however, meriting more research in the area. It is necessary to abstract away from algorithm-specific means of measurement for analysis to this end.

1.3.3 Visual SLAM

Visual Simultaneous Localization and Mapping, or Visual SLAM, is the problem of creating a map of the environment, usually as a point cloud representation, and calculating the ownship's (or observer's) pose at the same time [67].

There are so-called "indirect" and "direct" methods of conducting Visual SLAM [15]. Direct methods involve measuring the change in the pixel values directly when calculating the map and pose information. Indirect methods leverage some method of abstracting away from the image itself and instead focusing on some aspect of the

image, or features. Many indirect methods will use what are known as "keypoints", or points/regions of an image which have some distinguishing characteristic which will make them easier to track between frames. This could be the points defining either end of a line, or the "cornerness" of an area.

The problem of Simultaneous Localization and Mapping (SLAM) is a well-researched topic. Observer reprojection onto sparse point-clouds aggregated by sensors (e.g. LiDAR) are the primary means of addressing such a problem, and since its original incarnation more approaches have come to the forefront. Visual SLAM, replacing cameras for said sensors and performing computer vision processes to achieve the same task, is viewed as a sensing method that can radically reduce the costs associated with this problem while improving salient information. For the past decade, new Visual SLAM algorithms have come to the forefront, many of which build on each others' improvements. Non-filtering Visual SLAM algorithms are a subclass within this area of development that are best suited for most applications [65] - of these, several algorithms have signified milestone improvements to the state-of-the-art [38] [19] [28] [17]. ORB-SLAM [50] is one such method (or class of methods in the case of ORB-SLAM2 [51]). In their approach, Oriented FAST and Rotated BRIEF (ORB) features [58] are used to identify keypoints with descriptors to build and reference a sparse point cloud representation of the 3D structure of the environment. As with most other indirect SLAM methods, this works via Bundle Adjustment (BA): an optimisation problem wherein the reprojection error is minimised between the features detected and point cloud, and it continually improves the accuracy and efficiency in several ways including culling keyframes and

keypoints and performing loop closure where necessary.

1.4 Thesis Outline

This thesis explores the use of Visual SLAM, specifically ORB-SLAM2, onboard a multicopter platform in simulation and on a physical aircraft. The next chapter of this work is solely focused on the reliability of performing localization with monocular ORB-SLAM2 in a feature-sparse indoor area wherein localization error is assessed in relation to several parameters and state elements. The following chapter then offers an application example of how Visual SLAM can enable advanced flight modes via the implementation of a "Vision-assisted Terrain Following" mode of operation with accompanying analysis. The thesis is concluded with final thoughts and future work suggestions which can springboard from this work into a variety of projects.

Chapter 2

Visual SLAM aboard UAS: Localization and Mapping Analyses

2.1 Problem Statement

ORB-SLAM is attractive for UAS localization for several key reasons: (i) flexibility of camera type, (ii) computational efficiency, (iii) robustness to rotation, and (iv) ability to recover from localization loss. However, as sensors for aircraft require failure assurance of approx 10e-9 and the reliability of ORB-SLAM, nor most any other Visual SLAM approach, is of this caliber, additional investigation into quantifying expected success is pertinent to gauging appropriate peripheral sensing compensation and redundancy. The datasets used to validate ORB-SLAM exclude any multirotor-aggregated image sequences which are inherently different from handheld, car, and robot sequences within TUM and KITTI [50]. Implementation as a localization method for multirotors

merits extending validation via UAS implementation. We seek to investigate how well-suited ORB-SLAM is in its current form for UAS-borne sensing. In our work, we analyze the recorded egomotion of various flights with adjustments to ORB-SLAM and autopilot parameters. By mainly focusing on the long-term flight localization accuracy within a constrained space and analyzing the scenarios where poor ORB-SLAM performance manifests - the conditions under which BA converges to unreasonable reprojection error minimums without altogether failure - we emulate a likely environment for ORB-SLAM to have its limits tested in application. The corresponding recovery periods, if recovery occurs, is taken into account as well.

2.2 Background

2.2.1 ORB-SLAM

2.2.1.1 Overview of the algorithm

One very successful attempt at solving the Visual SLAM problem is an algorithm known as ORB-SLAM [50], based on local and global bundle adjustment of feature points. This approach uses ORB, or "Orientation Robust BRIEF", feature points, and with successful initialization will begin to draw out the three-dimensionality of the environment. This algorithm incorporates an aspect of "self-culling", meaning that as the map grows and there are more keyframes, or saliently-identified vantage points which comprise the nodes of a graph, the system begins to remove the less effective map points from the point cloud to keep the amount of memory use within reason. For the research

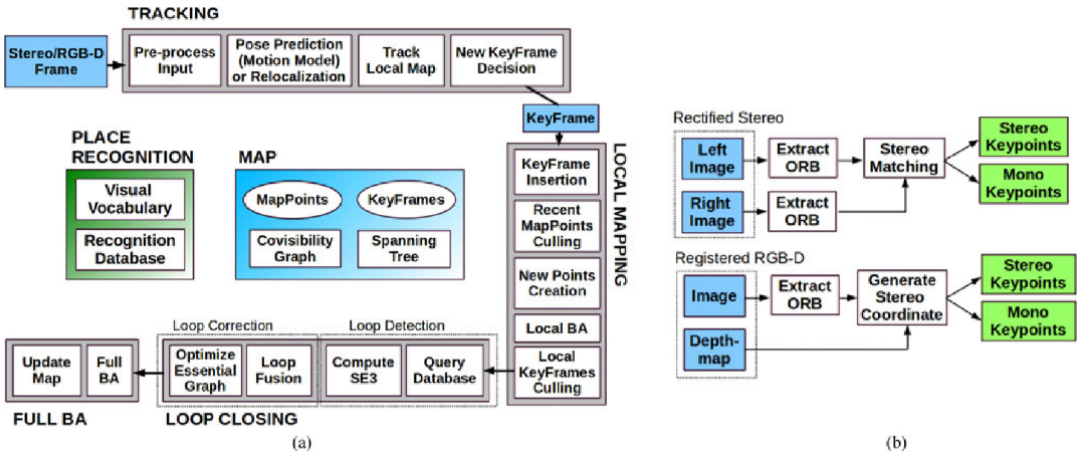

Figure 2.1: ORB-SLAM2 algorithm's high-level block diagram for Stereo and RGB-D operation from [51]

efforts of this thesis, we use the second variant "ORB-SLAM2" which introduced stereo and RGB-D operation in addition to the pre-existing monocular mode [51]. The pipeline for computation of this variant is shown in Figure 2.1.

ORB-SLAM works first by detecting the ORB features in the image frame. Camera calibration, for example, using Kalibr, is done before running the algorithm in order to establish both the intrinsic and extrinsic transforms for the image to the mounted camera coordinate system and to determine distortion parameters to rectify the image. In the case of monocular slam, initialization is successful upon moderate motion of the system wherein a satisfactory number of points can be detected and the parallax gauged to begin the mapping process. For stereo, the detected map points are matched between the left and right camera of the system via their relative pose to each other (also easily detected when conducting automated calibration in Kalibr). This gives

ORB-SLAM information on the three-dimensionality of the feature points with respect to the observer which, if it is within a distance reasonable for a given baseline length (i.e., the distance between the stereo cameras), helps inform the depth information. A process minimizing the feature point reprojection error, or the agreement between the existing map points and those projected onto the scene from the current vantage point, is undergone which refines the existing map points with the new observations through a process called "bundle adjustment".

Note that Chapter 2 of this thesis applies monocular ORB-SLAM2, and Chapter 3 utilizes stereo ORB-SLAM2.

The map, composed of the 3D-projected feature points generated within ORB-SLAM, is gradually culled and refined as the algorithm runs. When looped trajectories are detected, a process of loop closure optimizes the existing map and trajectory to resolve any discrepancies between the ends of the loop. Keyframes (key vantage points which are saved with the progression of the algorithm and linked together via a graph of transforms) are referenced as the system gets closer to them. This scheme aids in the computational efficiency of the algorithm as opposed to re-projecting from scratch every time. In other words, having vantage points as reference makes the point cloud projection process more predictable and easier to arrive at a solution in real-time.

Visual SLAM was selected as an approach to sensing the ownship pose and the map of the environment because of the high fidelity of the algorithm selected and the ease of backing out map sections for use in influencing the height of the aircraft.

2.2.1.2 ORB features

Because this work involves the loss and acquisition of features as they are related to ORB-SLAM2 positioning error, it is necessary to give a short overview of how ORB itself works. ORB may be understood in its two primary functions: feature detection and matching.

2.2.1.3 Feature Detection

ORB utilizes an adaptation on FAST (Features from Accelerated and Segmented Tests) [58]. Regularly, FAST identifies corners reliably and quickly using a combination of image subsection pixel intensity thresholding and machine learning. As identified in the literature, the feature detection repeatability falloff from higher noise content in an image is more drastic with FAST than some other feature detection algorithms [41][63] though having better overall performance in general [57]. This is relevant to the problem at hand as motion blur may affect algorithmic performance in a similar way.

2.2.1.4 Feature Matching

ORB's adaptation of BRIEF improves upon rotation sensitivity by adding a "steering" component to the descriptor and adding a learning stage to minimize correlated tests. The tests here are binary pixel intensity tests using a smoothed image

patch p for the following equation.

$$\tau(\mathrm{p};\mathrm{x},\mathrm{y}) := \begin{cases} 1 & : \mathrm{p}(\mathrm{x}) < \mathrm{p}(\mathrm{y}) \\ 0 & : \mathrm{p}(\mathrm{x}) \geq \mathrm{p}(\mathrm{y})' \end{cases}$$

The descriptor is then a vector of the form:

$$f_n(\mathrm{p}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathrm{p};\mathrm{x}_i,\mathrm{y}_i)$$

2.3 Analytical Aims

This work aims to characterize ORB-SLAM performance in a sparsely-featured environment. As such, the Root Mean Squared (RMS) error for the position estimate generated by ORB-SLAM is calculated based on an external motion tracking ground truth system (details outlined in Section 2.4.4. There is also an emphasis on the discrepancies between different mapping instances, and comparisons are drawn between different performance levels when calling upon these keyframes and point clouds for reference. There are experiments included which test both a manually flown system in addition to autonomous flight trajectories to provide variety in dynamics imposed on the system. Finally, there is an effort to look for certain vantage points within a map to see if viewing from certain positions tend to have higher error propensities.

2.4 Methods

2.4.1 Pre-experiment work

2.4.1.1 Pre-mapping process

The experiments in this chapter were conducted in a "localization-only" capacity, meaning that the mapping happens prior to localization testing. Pre-mapping occurs by manually moving the UAS throughout the space to be flown facing nominally in the forward direction. The entire field of view is depicted in Figure 2.2. Being an unaugmented version of monocular ORB-SLAM2 with no additional sensing incorporated, initialization requires a satisfactory number of ORB feature matches in the space to be tracked throughout movement for a starting set of map points (i.e. the first keyframe references). Once accomplished, ORB-SLAM2 continues to aggregate new map points and keyframes while culling the superfluous map points and keyframes. The map is saved when a desired quality is achieved. This map will then have a somewhat arbitrary orientation and scale which we must reconcile with our ground truth. The coordinate system in ORB-SLAM2 is measured in relation to the motion capture coordinate system. The motion capture has a manually set origin and orientation within the room which we align ORB-SLAM2 to by applying a scaling/rotating and translating operation to prior to experiments. With the two coordinate systems aligned and scaled, we may conduct the flights.

2.4.2 Experimental overview

2.4.2.1 Manual flights

The primary focus in experimentation is localization error and where it appears most prominently with changes to ORB properties. Three point maps of changing size (measured in map points) were created before each set of measured runs. For each of these three, four values of ORB feature counts are tested by tracking localization values from ORB SLAM2 and our ground truth (NaturalPoint OptiTrack). The flights conducted are manually flown, semirandom patterns tending towards forward-facing circular traversal of the space between 0.5m and 2.0m in height lasting approximately one minute. In addition to the UAS pose, map point matches (i.e. features successfully reprojected) are recorded throughout each flight.

2.4.2.2 Autonomous flights

Autonomous flights conducted in this work entail issuing a stream of setpoints to the aircraft while in PX4's "Offboard" mode. The flight patterns are loaded as CSV files to a ROS node which parses the flights and commands the aircraft to traverse the points using time elapsed and the waypoint hit status as the means of transitioning to the next point. Three patterns of flight were executed, generally described as: (1) side-to-side, (2) forward-and-back, and (3) square. The resulting flight patterns themselves can be seen in Figure 2.11.

Figure 2.2: Forward-facing FoV imaged when running ORB-SLAM2.

2.4.3 Environment and platform

The environment for experimentation is a contained space approximately 10m x 10m x 8m. The observable end of the room (the "front" from the perspective of the aircraft constrained to this heading) is populated with assorted boxes and other feature-bearing objects (e.g. a low railing, a fire-extinguisher case, etc.). These elements comprise a semi-structured environment for Visual SLAM to be conducted in reference to.

The environment flown in is a relatively feature-sparse indoor flying arena space. All experiments are conducted such that the multicopter system faces directly forward as it flies. A panoramic picture of the scenery which the drone observes is shown in Figure 2.2. Stacks of cardboard boxes are included in the space to give a bit more variation in feature presence than would otherwise be the case with the built in elements of the wall and floor of the room.

Our aerial system is depicted in Figure 2.3. The aircraft used in experimentation was a hexacopter controlled via mRo Pixhawk, an open-source autopilot hardware

Figure 2.3: UAS used for ORB SLAM2 experiments. The payload includes an Intel NUC, mRo Pixhawk, Monocular Arducam, and various peripheral components.

system [46], running PX4 [45]. On-board vision processing is achieved with the use of an Intel NUC7CJYH companion computer running Ubuntu 16.04. The companion computer communicates with the autopilot module over a CP2102 USB-to-UART bridge. An Arducam AR0134 1.2MP monochromatic camera is mounted on the front of the aircraft by a 3D printed vibration- damping fixture and transfers image data via USB to the companion computer, which is seen in Figure 2.4.

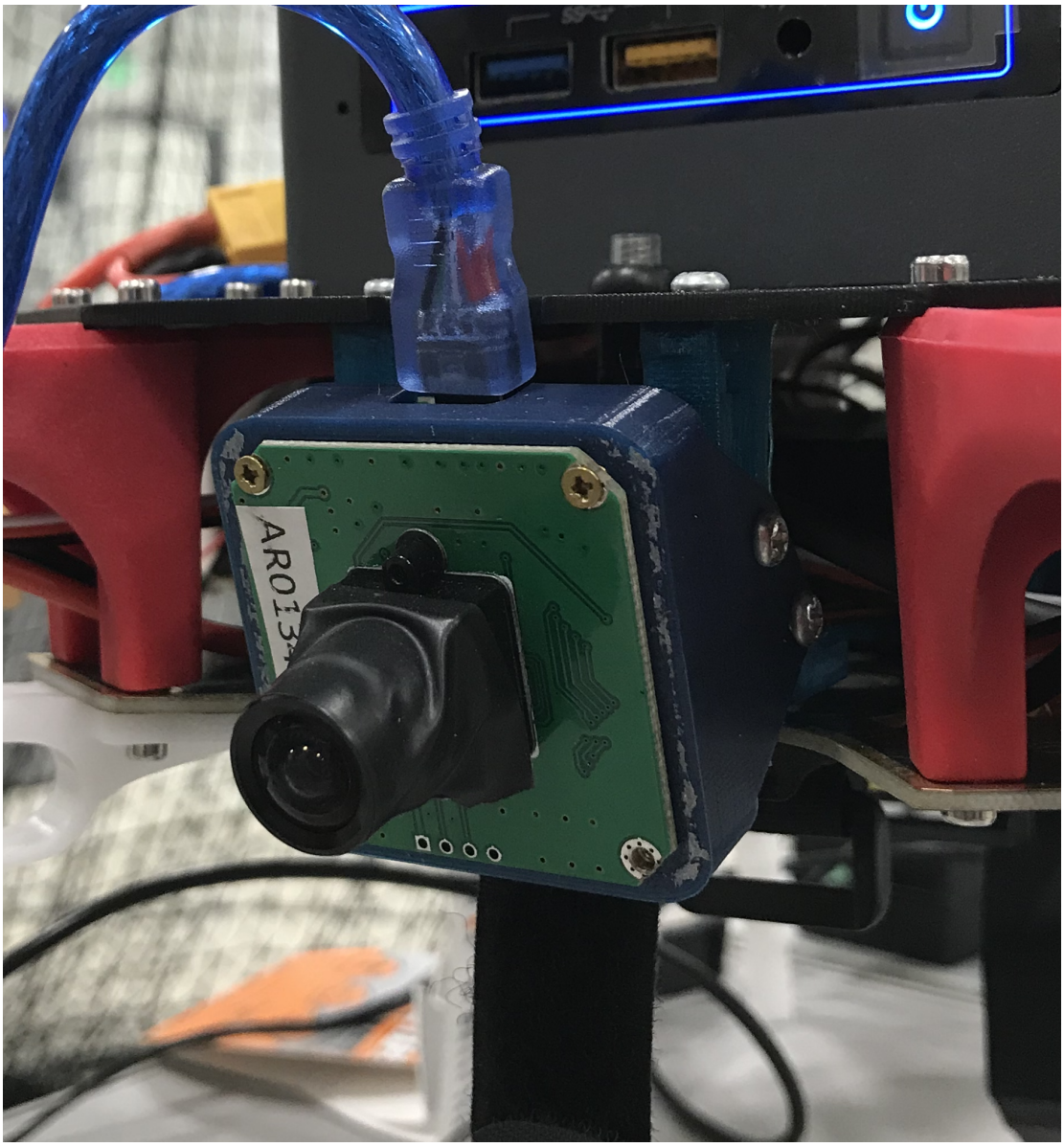

Figure 2.4: Drone front camera mounting scheme.

2.4.3.1 ROS integration

The ROS interface is used to simplify component integration and because of its interoperability with MAVLink (MAVROS), OptiTrack (rpnclient), and peripheral sensors.

2.4.4 Ground truth acquisition

Ground truth was captured with a motion capture system. OptiTrack Prime 17W 1.7MP cameras surrounding the flight space work with Motive software to track infrared markers mounted on the aircraft. The markers comprise the rigid-body representation within Motive. The position of the centroid (with respect to the rigid body's convex hull) and attitude relative to the initialized orientation comprise the ground truth pose.

Preconstructed maps are created using manually piloted, forward-oriented flights were used for ORB feature number experiment flights. In this way, a static map structure was used for multiple flights and allowed for performance assessment amongst them. Due to the initialization phase of ORB-SLAM deciding the origin position and orientation based on first successful reprojection, agreement between the coordinate systems for it and OptiTrack must be reconciled. Aligning the coordinate systems for the experimental and ground truth data is achieved in two ways.

For experiments where exact accuracy is less pertinent and the qualitative relation between these data is the focus, a retroactive linear best fit is performed on the experimental data. This process consists of discrete rotation, scaling, and translation

optimization between the vehicle location point distributions for ORB-SLAM and OptiTrack. Doing so preserves the high frequency noise and large error present in the experimental data while approximating the alignment for better assessment. These stages are based on [3].

The second method for grid alignment approximation of the two coordinate systems is performed after map creation and prior to experiments. Ten to 15 manually measured positions are recorded along with their respective ORB-SLAM output poses. As in the previous method, transforms between each system basis is done by capturing the scaling, rotation, translation, or combination matrices therein; however, in this instance, the matrices for the transform are computed via pseudo-inverse from the recorded measurements. Doing so minimizes residual discrepancy and approximates the error more exactly allowing for precise quantitative relationships to be formed between localization error and contributing factors from aircraft movement and ORB-SLAM performance.

2.4.5 Feature number variation

Being that evaluation of the ORB-SLAM algorithm's performance when run on a UAS is at the fore of this work, the key parameters for the algorithm must be considered. A primary parameter known as the "Feature Number" dictates what the maximum amount of features per frame may be detected when running the algorithm. This parameter affects the performance in that higher feature numbers allow for a more rich understanding of the environment, yet it can be of lower computational efficiency

and lead to insignificant or weakly-associated features to be entered into consideration for the 3D point cloud.

The parameter is altered using a YAML file which is loaded upon the algorithm's initialization. This parameter was never changed in the midst of a flight, only between flights conducted.

2.5 Results

2.5.1 ORB-SLAM outputs

The ORB-SLAM2 monocular variant was run in localization-only mode aboard our aircraft. An example of the view of the feature point detection and correspondence matching is seen in Figure 2.5. Green dots within the image represent the feature points which are being tracked between images, and the dots which have boxes around them correspond to the features which have been successfully mapped onto the reference point cloud, or "map". An example of the map is depicted in Figure 2.6. Points which are red correspond to the boxed feature points mentioned in 2.5.

2.5.2 Manual flight ORB-SLAM performances

The trajectories believed to have been traversed by ORB-SLAM2 (blue) and the comparison with the ground truth (orange) are seen in Figures 2.7, 2.8, and 2.9. A breakdown of the performance statistics is offered in Table 2.1, where the flights which were considered to be complete localization failures in a post-experiment evaluation are

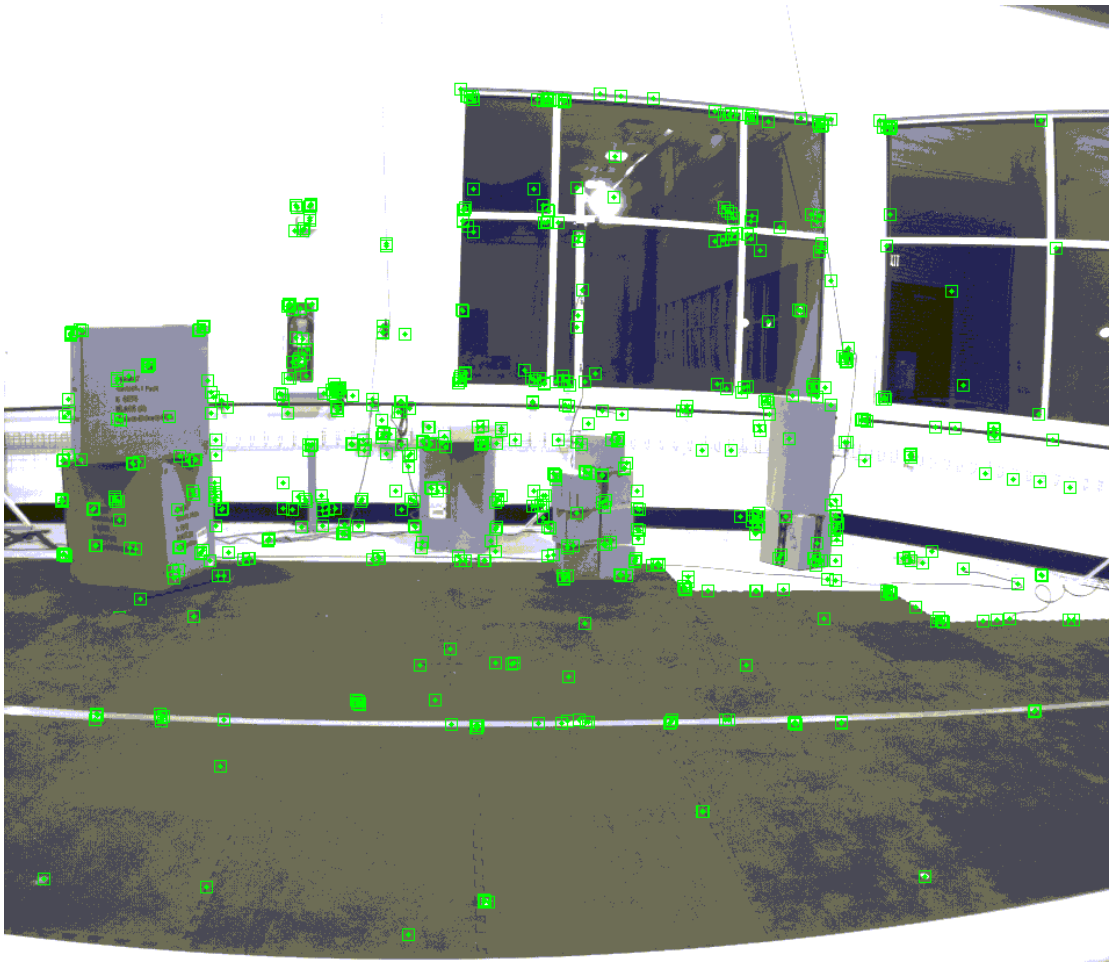

Figure 2.5: Image view rendered with salient features identified and matched to point map via the ORB-SLAM2 output.

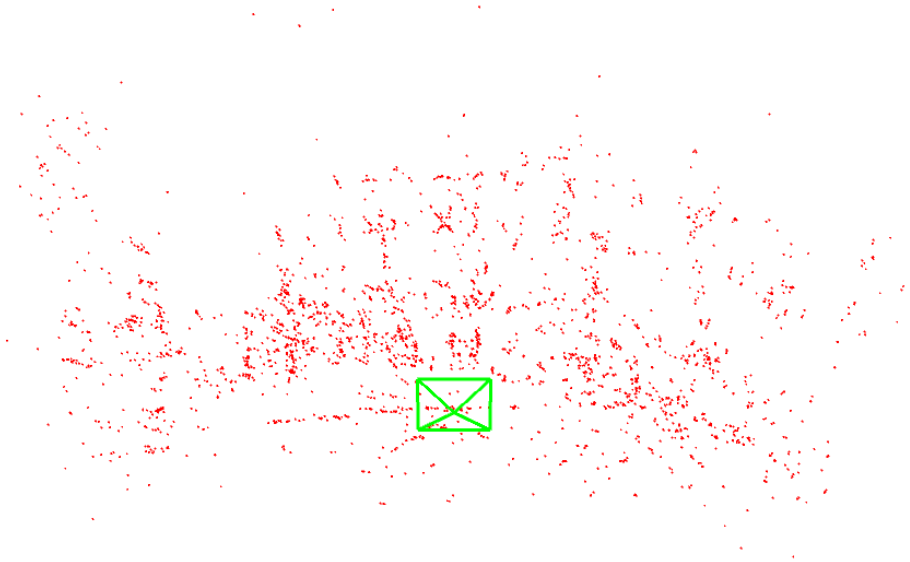

Figure 2.6: Visualization of Map 3 point cloud.

highlighted red. An accompanying bar plot presents these results graphically in Figure 2.10.

Flights with sparse high tracking failures that the system recovered from reasonably have noteworthy maximum errors and slightly higher than average standard deviations, but otherwise have unremarkable tracking metrics. Nominal errors for misalignment between OptiTrack (ground truth) rigid body centroid and camera-centric SLAM must be factored into the analysis as well; the average RMS error across all flights not considered instances of tracking failure is 0.1068 m, approximately 1/5th of the diameter of the aircraft.

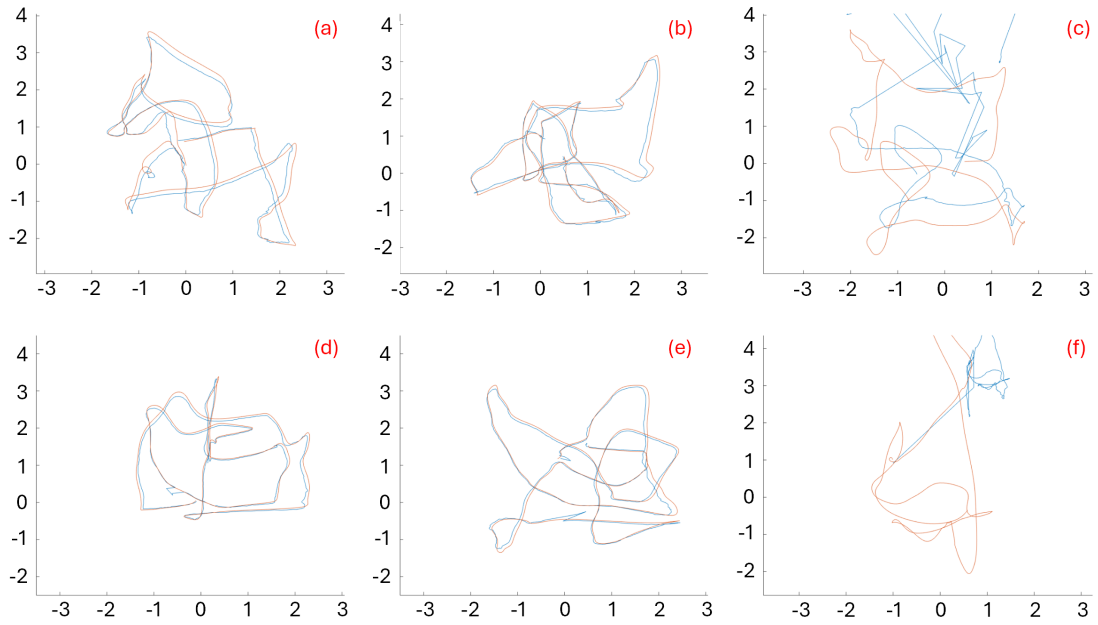

Figure 2.7: Map 1 Flights; Number of ORB features extracted per image (N) changes. (a) N = 1000, (b) N = 1200, (c) N = 1500, (d) N = 2000, (e) N = 2500, (f) N = 3000. Blue is ORB SLAM2 reported position, Orange is Ground Truth. Axis units are in meters.

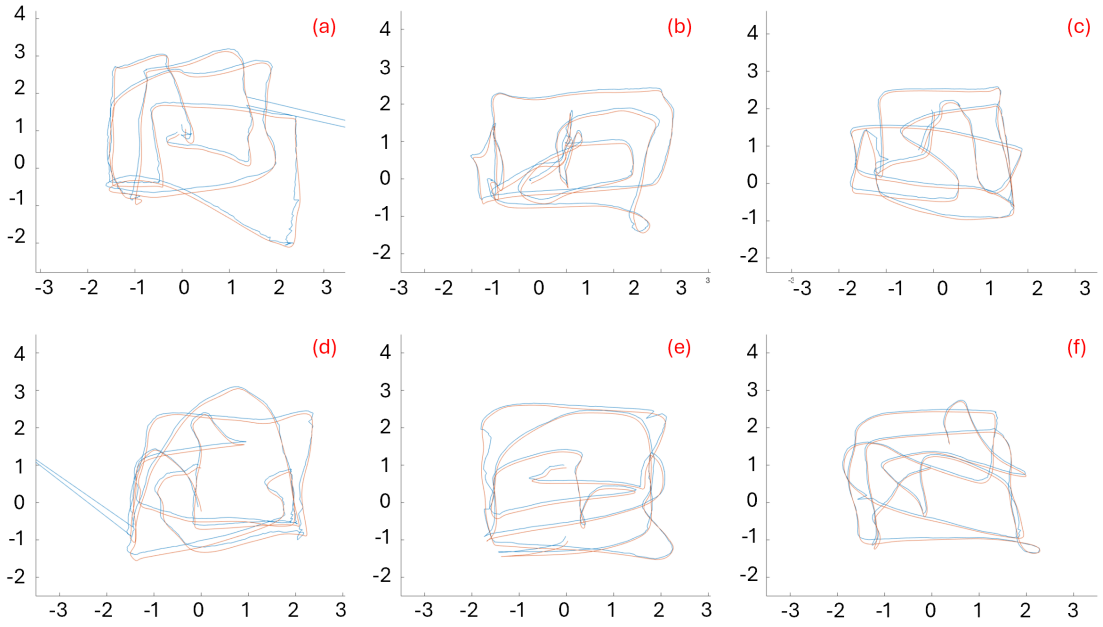

Figure 2.8: Map 3 Flights; Number of ORB features extracted per image (N) changes. (a) N = 1000, (b) N = 1200, (c) N = 1500, (d) N = 2000, (e) N = 2500, (f) N = 3000. Blue is ORB SLAM2 reported position, Orange is Ground Truth. Axis units are in meters.

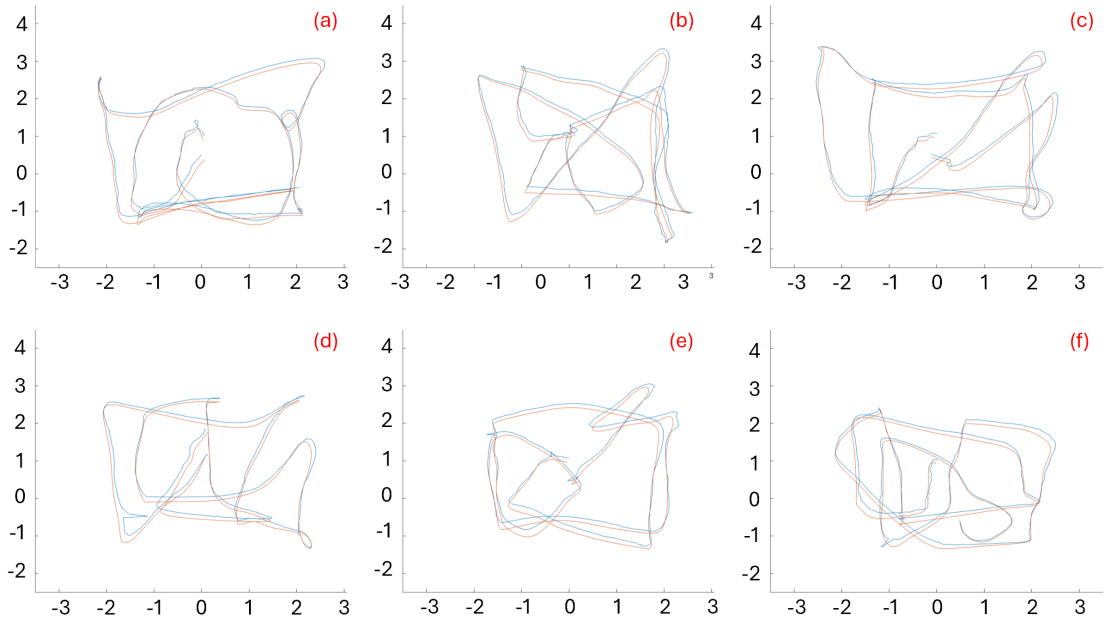

Figure 2.9: Map 3 Flights; Number of ORB features extracted per image (N) changes. (a) N = 1000, (b) N = 1200, (c) N = 1500, (d) N = 2000, (e) N = 2500, (f) N = 3000. Blue is ORB SLAM2 reported position, Orange is Ground Truth. Axis units are in meters.

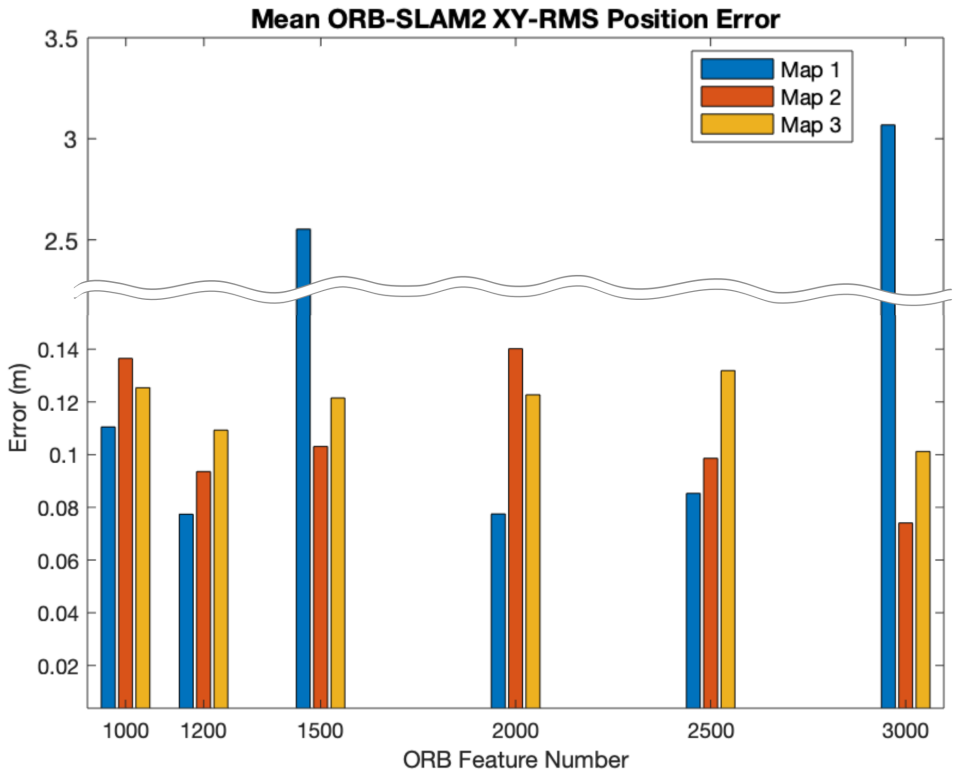

Figure 2.10: Mean RMS Errors for ORB-SLAM position tracking grouped by ORB Feature Number.

2.5.3 Autonomous flight testing

Figure 2.11 gives a comprehensive view of the localization effort from ORB-SLAM2 and the ground truth data for comparison for all of the autonomous flights. This includes all front-to-back, side-to-side, and square flights. An example of the square flight statistics for RMS error can be found in Figure 2.12. The same analytical technique was applied to all autonomous flights and aggregated into the evaluation schemes of Figures 2.13 and 2.14.

Figure 2.13 is a means by which to evaluate how the ORB feature count in any given frame cross-referenced with the velocity of the system may correspond to positioning error. Note that at rest or at nominal lateral traversal velocity, we see less positioning error at low feature point counts than in the middle. This manifests due to the fact that the most dynamic moments of the flight from the viewpoint of image blurring and rotational disturbances occur in the transitions to and from each waypoint – when steadily resting at a waypoint or with a nominally consistent velocity, the system suffers less error. This is further explored in Figure 2.14, where we consider the combination of instantaneous lateral acceleration and velocity as they are associated with positioning errors. Again, lower acceleration values, independent of velocity, tend to have lower positioning errors than moments of high acceleration values.

2.5.4 Area characterization

A consideration made in this work was the suitability of the flight space itself. In particular, it was of interest whether there were certain locations that, when flown to,

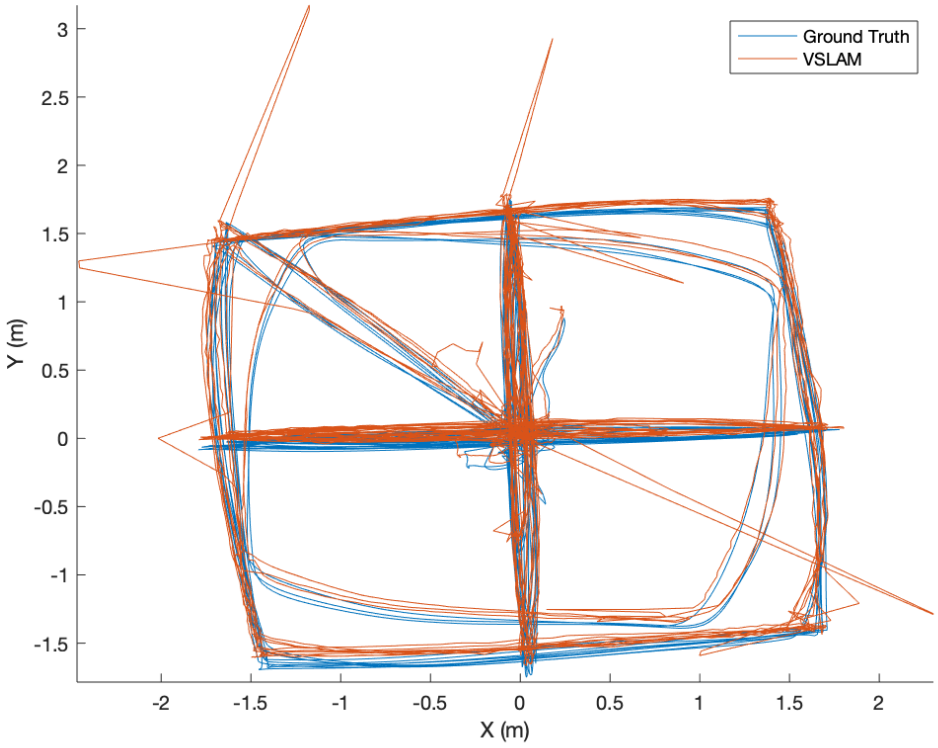

Figure 2.11: All autonomous flights conducted for these experiments.

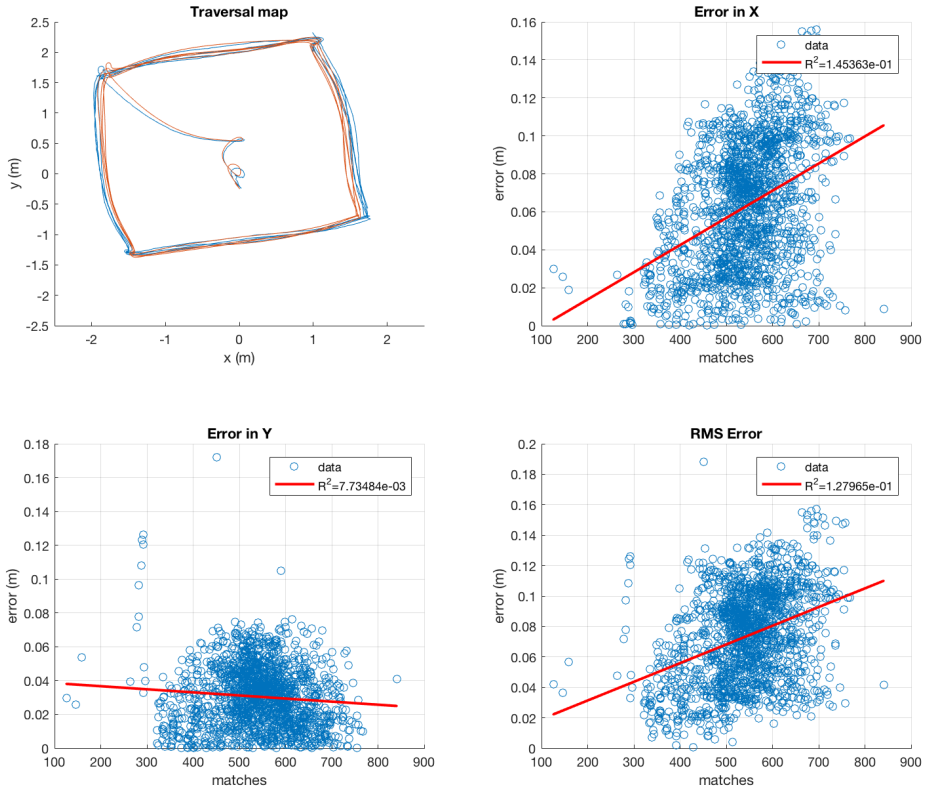

Figure 2.12: Square flight with error evaluation. Note that there is weak, if not no, apparent linear correspondence of the number of feature points detected to the error calculated.

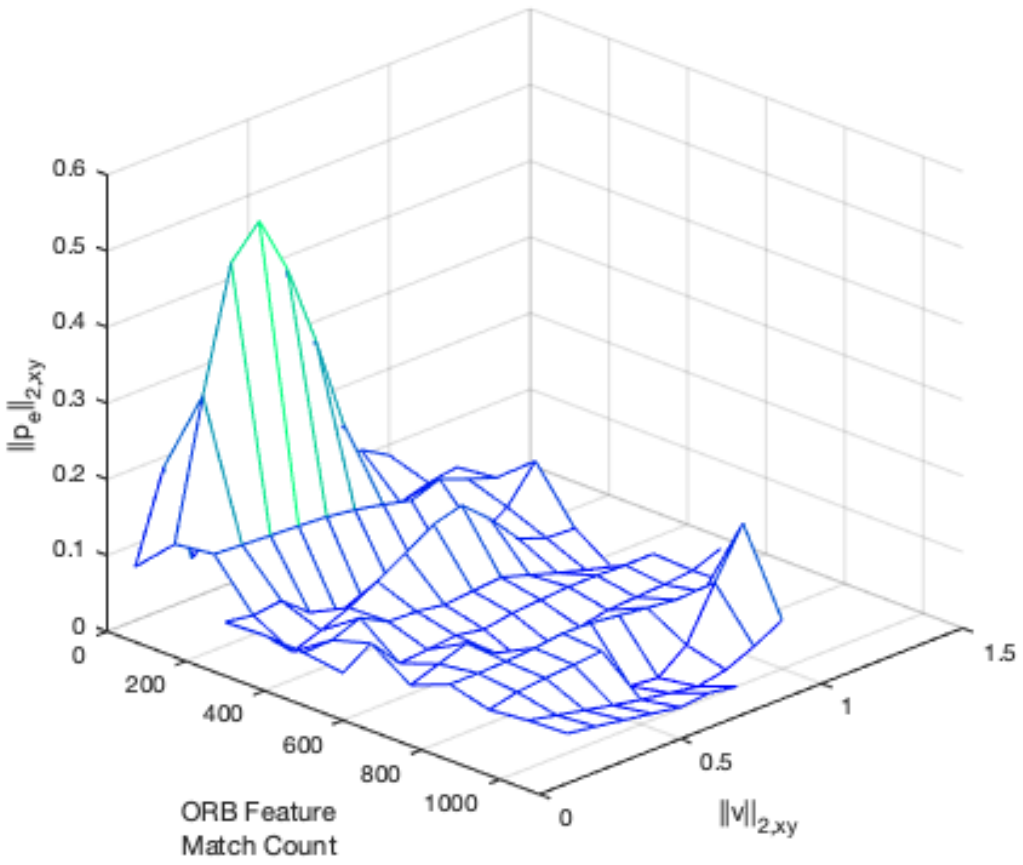

Figure 2.13: Match count versus RMS velocity versus RMS position error. Data is an aggregate of all autonomous flights.

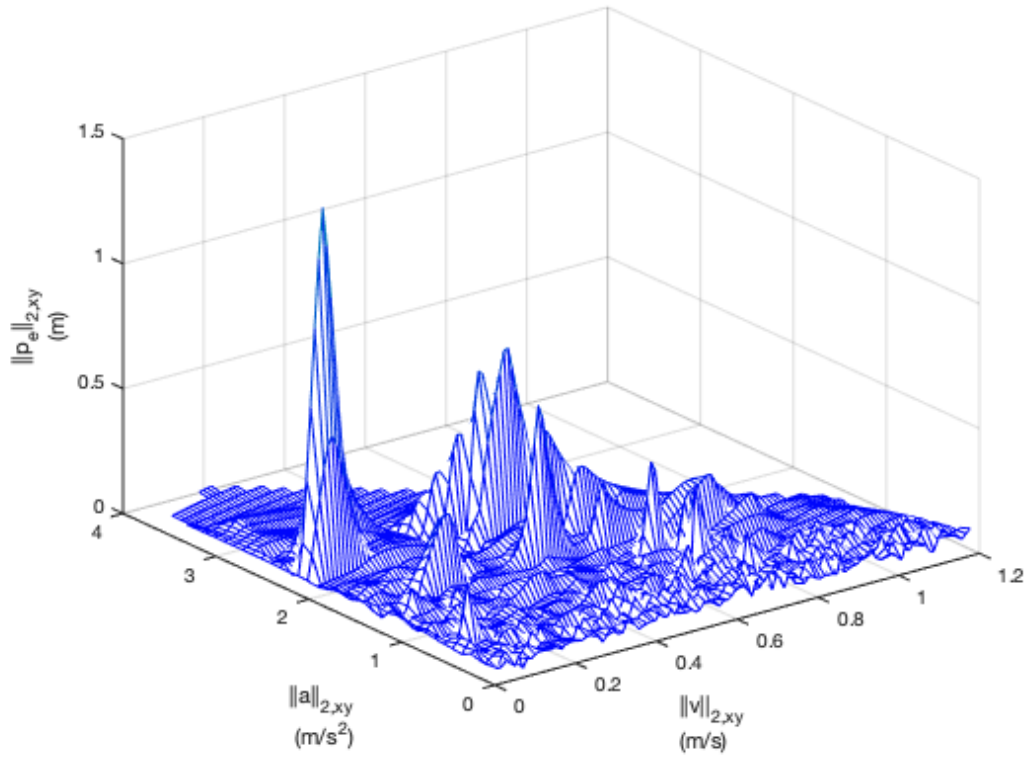

Figure 2.14: RMS velocity versus RMS acceleration versus RMS position error. Data is an aggregate of all autonomous flights.

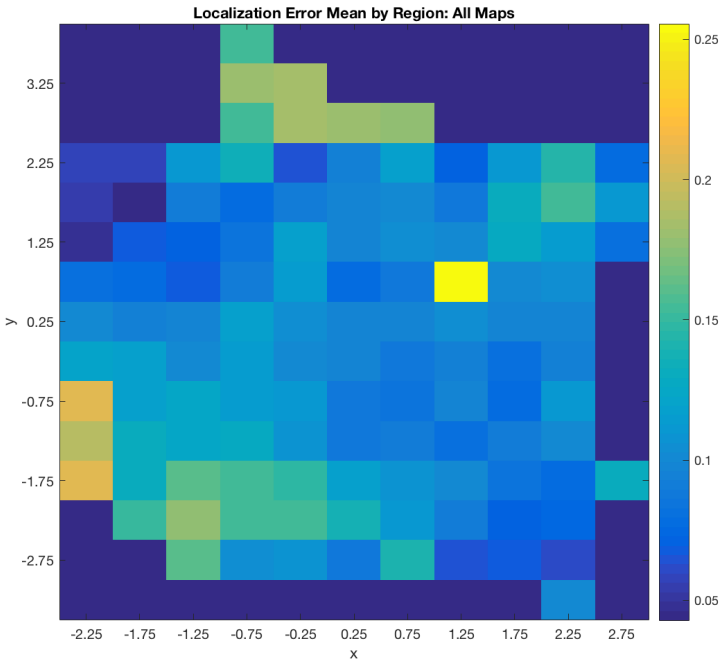

Figure 2.15: Area characterization effort to assess whether or not there is a correspondence between certain regions of the flight space and positioning error susceptibility.

tended to be more or less prone to positioning error. The rationale here is that, should there be a certain keyframe or region of perspective where the bundle adjustment and reprojection effort are less effective, calculating the mean and standard deviation of the errors reported in laterally-spread cells should be evident when plotted. Said plots are depicted in Figures 2.15 and 2.16.

All axes and values on these plots are in units of meters.

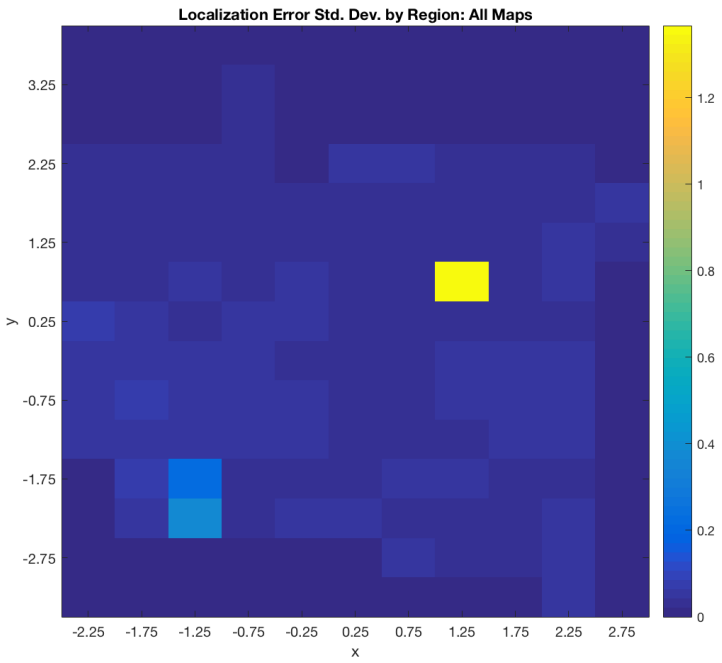

Figure 2.16: Standard deviations from the area characterization study, associating lateral location in the flight space with positioning errors.

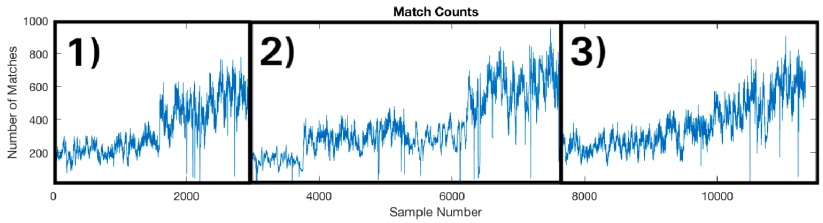

Figure 2.17: Progression of the number of features detected for three separate flights. Note that the feature numbers are consistently less than one thousand per frame.

2.6 Discussion

Overall, the results of varying the maximum number of feature points were not conclusive with respect to how they influenced the effectiveness of the algorithm. This is seen in that high positioning errors seem relatively sporadic regardless of the maximum number of feature points imposed in the algorithm's pre-configuration. Were this experiment to be conducted again, there would be a lower maximum number of feature points imposed. As seen in Figure 2.17, the number of feature points actively detected in three separate run instances does not reach numbers which would contest this parameter (i.e., the maximum number of features detected at any point in time is much less than the maximum feature parameter, so there is no clipping). As such, to further test the theory that a lower maximum set for the feature point count will effect the frequency of significant positioning errors, maximum feature numbers on the order of low to high hundreds of features would be varied in lieu of low thousands.

The primary takeaway from this work was regarding the higher dynamic moments of the aircraft moving corresponding to reprojection issues and higher positioning

error. As seen in the previous figures relating the velocity and acceleration to the feature loss, it is apparent that the moments of change between stationary and cruising and visa versa are those in which higher positioning errors occur. This is consistent with the notion that ORB-SLAM, or any Visual SLAM, especially those without an inertial component, will struggle in moments which are primarily rotational and which impart feature blur in the images observed by the drone.

Map	ORB #	Err Mean	Std Dev	Err Max	Err Med
1	1000	0.1105	0.0520	0.2393	0.1112
	1200	0.0774	0.0431	0.2282	0.0704
	1500	2.5525	4.0560	17.4018	0.7784
	2000	0.0775	0.0455	0.2223	0.0721
	2500	0.0853	0.0504	0.5854	0.0790
	3000	3.0684	1.1718	4.7867	3.6008
2	1000	0.1365	0.5971	12.0714	0.1060
	1200	0.0936	0.0302	0.2226	0.0947
	1500	0.1031	0.0487	0.3000	0.1057
	2000	0.1402	0.2410	4.0011	0.1293
	2500	0.0986	0.0419	0.2837	0.0952
	3000	0.0741	0.0348	0.2879	0.0684
3	1000	0.1254	0.0505	0.2579	0.1263
	1200	0.1093	0.0466	0.2861	0.1104
	1500	0.1215	0.0455	0.3244	0.1244
	2000	0.1227	0.0551	0.6263	0.1201
	2500	0.1319	0.0564	0.2927	0.1224
	3000	0.1012	0.0460	0.2689	0.0940

Table 2.1: Localization metrics for experiments. Each column represents the map ID, ORB feature number, XY-RMS average error, XY-RMS error standard deviation, XY-RMS error maximum, and XY-RMS error median. Rows colored red indicate flights classified as failures w.r.t. tracking.

Chapter 3

Design and Implementation of a Vision-Assisted Terrain Following Flight Mode

3.1 Motivation

3.1.1 UAS use in the geosciences

Two applications within the geosciences that benefit from UAS are associated with mapping and monitoring efforts [24] [18] [33] [34]. The specific survey methods towards these goals, including hyperspectral [32] [31] [26], thermal [16] [70], magnetic [31] [73], gas [36] [76] [40], and other analyses, provide insights as studies pertaining to resource (water, energy, mineral) and hazard investigations (volcano, earthquake).

The United States Geological Survey, for example, conduct missions of the

aforementioned types. Most of the missions conducted occur anywhere from just above ground level to approximately 400 ft. above ground level. Many of these missions occur on the lower side in altitude which permits higher resolution imaging of features of interest. This low flight regime enables gas sampling near the surface before it diffuses too much into the atmosphere, or maintains the magnetometer close to shallow crustal magnetic sources, enabling detailed mapping of their associated magnetic fields. Rovers, or Unoccupied Ground Vehicles (UGV), may be deployed in some cases towards these or similar initiatives, but sometimes the terrain is too treacherous/varying for UGV or the interaction between the rover and the ground is undesirable in performing the data collection [4]. For missions involving very low terrain-clearance operations, "terrain following" functionality is necessary.

Terrain following, a mode of UAS operation wherein the aircraft keeps a uniform distance above ground level, changing with the terrain, is a very useful means of conducting surveys, especially when staying close to the ground is important. The ability to contend with changing, undulating ground is useful for the mission types mentioned above.

3.1.2 Terrain Following Methods

There are a handful of approaches which allow a UAS to track the terrain as it flies.

3.1.2.1 Using DEM

One of these approaches involves using a Depth Elevation Map, or DEM, otherwise known as a Digital Surface Model, or DSM, as height data to affect the altitude of the aircraft [60] [47]. This involves downloading or otherwise collecting a DEM for an area which is to be surveyed, leveraging software which can mold the trajectory of the UAS to the terrain via the DEM data such as the program UGCS (which is frequently employed by geoscientific researchers flying UAS).

A problem with this approach is that, depending on when the DEM was collected, the surface of the area being surveyed may have changed. Because surveys are often conducted in very dynamic environments with rough weather patterns, what was once a very accurate and reliable DEM may prove to not be useful at the site of interest when there are unexpected obstacles, debris, etc. to mitigate collisions with. These DEMs also do not account for vegetation, as well as topographic features whose spatial extents are below the resolution of the DEM.

3.1.2.2 Using Altimeter

Another way to achieve terrain following is to use a laser- or radar-altimeter system [37] [52] [62]. This device measures the distance between its mounting point on the UAS and the ground by sending pulses of light and measuring the amount of time taken for the light to return. These sensors are generally placed on the underside of the UAS, oftentimes rigidly affixed, but sometimes mounted with the use of a gimbal. Especially when rigidly affixed in a nadir configuration, the sensed point lags the vehicle

due to the inherent pitching of a traditional rotorcraft system. This means that if the system is flying quickly and encounters steep terrain, there is a possibility that the altitude measured will be measured too late and the aircraft will then collide with the steep surface.

3.1.3 Using LiDAR

Lastly mentioned here as a means towards terrain following is via LiDAR [10] [25]. This sensing method is similar to laser altimeter but differs in that it sends out many light beams, aggregating a field of points to use for consideration in traversing the terrain. LiDAR tends to be a fairly expensive technology, and the high-performance modules can be bulky depending on the size of the UAS you are deploying. This takes up payload weight and power which could otherwise be occupied by the scientific sensing equipment desired.

3.1.4 Vision-based approaches to Terrain Following

This thesis proposes that a vision component be included to aid in terrain following, in a so-called "Vision-assisted Terrain Following" or VaTF modality. In this method, a camera system is included onboard the UAS to "see" what lies ahead terrain-wise. This approach gives the benefit that it can actively compute terrain height for wider areas, as if generating a small-scale DEM on-the-fly which then can be used in lieu of or alongside a laser altimeter to perform terrain following. Using the computer vision approach known as "Visual SLAM", we are able to draw out feature points representing

the convex hull of the environment similar to LiDAR and determine how terrain features map three-dimensionally.

Camera systems are desirable because they are often very inexpensive for the resolution/quality needed for effective Visual SLAM computation, they are also generally very lightweight and small which saves payload space for other instruments. One can achieve high-fidelity Visual SLAM with the equivalent of "webcams", or very cheap and relatively low resolution (on the order of hundreds of pixels) imaging systems.

Garratt and Chahl demonstrated the viability of a vision-based approach to terrain following in which they employed an imaging system (both in a downward- and forward-facing configuration) and demonstrated its effectiveness in comparison to using a laser range finder [22]. Their approach leverages optical flow to accomplish this via their "Iterative image interpolation algorithm", or "I^3A", which itself is based on the "Image Interpolation Algorithm", or "I^2A". They deploy their helicopter-based system in an outdoor environment and are able to fly at very low altitudes on the order of 1.5 - 2 m at relatively high forward velocity speeds from 5 to 8 m/s.

Campos et al. designed a method of visually-gauged height detection for use with aircraft employing monocular vision systems [12]. Their work combines the use of optical flow with feature recognition and projection in order to determine the relative height of the terrain and influence the flight of the multicopter system. They present work completed using a physical platform as well as in a simulated environment. Their system factors in consideration of highly-dynamic maneuvers as it affects the performance of the algorithm.

Srinivasan et al. created an optical-flow based method to terrain following wherein a reflective surface is imaged off of in order to get a wider view of the ground below [64]. The design of the distortion of the reflective imaging surface is shaped in order to reflect equal distances on the ground to equal distances within the image in the direction of the forward velocity. Their system is tested and characterized optically indoors and then further tested in an outdoor environment. An image remapping process is conducted prior to performing the optical flow.

3.2 Background

3.2.1 Multicopter dynamics

Given that this research is centered around multicopter control, it is necessary to give a brief overview of their kinematics and dynamics, and to describe how control is achieved in the inner loops of this work.

Multicopter platforms are distinct from fixed-wing platforms in that they generate lift directly from the propellers instead of from a wing. The "rotors", composed of the motor and propeller, vary their speeds to create thrust which, when combined with a flight controller, direct the aircraft's rolling, pitching, and yawing moments in addition to the overall thrust. Multicopters are inherently underactuated systems in their most generic configurations (all propellers spinning in approximately the same plane), meaning that the control of the system within its six degrees of freedom harbors coupling between different fundamental motions. The figure above depicts how the hierarchy of

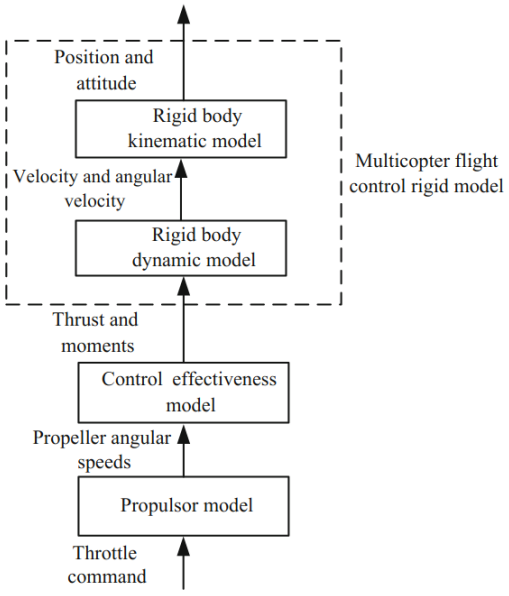

Figure 3.1: Figure 6.1 from [54] depicting the abstraction layers for the kinematics and dynamics for multicopter systems.

control is abstracted to get from individual motor motions to entire system motions (dynamics and kinematics) from [54].

For this work, we look at the pertinent physics that do not include individual rotor outputs to avoid loss-of-generality, i.e., assuming the autopilot used reasonably maintains control agnostic to rotor count – developing Visual SLAM enabled multirotor systems should not be restricted to quadcopters alone, but instead any vehicle with the same underactuated makeup. As such, we start from the vector representations of Euler translational and rotational physics. Given the variable representations for inertial frame position and orientation with respective velocities:

$$x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta \\ \psi \\ \phi \end{bmatrix}, \omega = \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\phi & cos\theta sin\phi \\ 0 & -sin\phi & cos\theta sin\phi \end{bmatrix}$$

Assuming that the multicopter in question is a rigid body, Equation 3.1 represents the translational mechanics.

$$m\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R_{BI}T_B + F_D \tag{3.1}$$

where m is the mass of the aircraft, x is the position of the aircraft in the inertial frame, g is gravity, R_{BI} is the rotation matrix from the body frame to the inertial frame, T_B is the thrust vector in the body frame, and F_D is drag.

Focusing on rotational mechanics, we start with the assumption that the quadcopter is somewhat inertially consistent with weight distributed evenly along each axis. In practice, this will vary from airframe to airframe, but for the purposes of this research is acceptable. As such, the moments of the inertia for the aircraft is described by I:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.2}$$

$$I\dot{\omega} + \omega \times (I\omega) = \tau \tag{3.3}$$

Combined with Equation 3.2, Equation 3.3 becomes:

$$\dot{\omega} = \begin{bmatrix} \tau_\phi I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\psi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix} \tag{3.4}$$

Collecting Equations 3.1 and 3.4 into their state-space space representation, we get Equation 3.5

$$
\begin{aligned}
\dot{x_1} &= x_2 \\
\dot{x_2} &= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R_{BI} T_B + \frac{1}{m} F_D \\
\dot{x_3} &= \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\phi & cos\theta sin\phi \\ 0 & -sin\phi & cos\theta sin\phi \end{bmatrix}^{-1} x_4 \\
\dot{x_4} &= \begin{bmatrix} \tau_\phi / I_{xx} \\ \tau_\theta / I_{yy} \\ \tau_\psi / I_{zz} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy} - I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz} - I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx} - I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix}
\end{aligned}
\tag{3.5}
$$

where x_1 is position, x_2 is velocity, x_3 is angle, and x_4 is angular velocity.

Changes in the thrust vector T_B rotated into the inertial frame via R_{BI} are coupled with changes to angular velocity. We then expect limitations on velocity in autopilot parameters to affect the induced vehicle rotations, and this is of interest when analyzing motion effects on ORB performance.

3.3 Methods

3.4 System architecture

Simulation of the system is achieved via the use of a collection of ROS nodes running in tandem as shown in the figure below. Being that this system was created within the ROS1 framework, a "roscore" node runs centrally which serves as the hub for the communication of the ROS nodes. The node demarcated below as "MAVROS + Gazebo launch" is a node provided by the PX4 repository which launches a MAVROS instance for communication with the simulated vehicle and a Gazebo simulation including the terrain/setting population and which spawns the vehicle. The "Offboard terrain following" node has already been described above. The "Pt cloud local height estimator" node is the node responsible for listening for the ORB-SLAM2-published point cloud and the ORB-SLAM2 estimation of the aircraft pose. This node creates the k-d tree from the map points and executes a radius search in the proximity of the aircraft as previously described. This node then publishes the local height estimate as well as the subset of map points which were identified as being within the radius of the aircraft. The "ROSBag recording" node simply listens for the topics of interest, including the Gazebo model states, the point cloud data, and the ORB-SLAM pose estimation. The "RViz" node generates the GUI viewing aid for the simulation data and allows for easier monitoring/debugging of the system. Finally, the "Stereo ORB-SLAM2" node is the node provided by the ORB-SLAM2 repository which listens for the stereo camera images generated within the simulation.

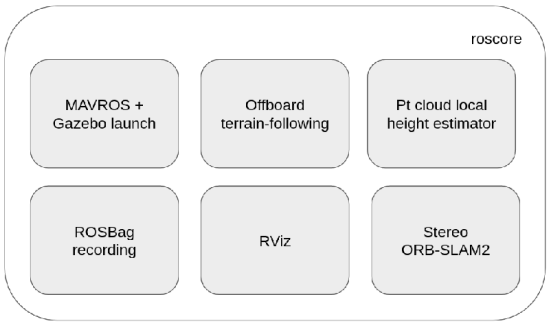

Figure 3.2: Diagram of the nodes/subsystems running in order to accomplish the vision-assisted terrain following. The architecture of the simulation is primarily based on interconnected ROS nodes.

3.4.1 Origin Placement

Conducting the terrain-following missions does not necessarily require the ability to freely define the origin of the local map. If using the local origin set by PX4, this point is defined by the location at which takeoff occurs. Within ORB-SLAM, it is the point wherever initialization first succeeds. In practice, being beholden to either of these two methods is not desirable because of the limitations they place on pre-planning and coordinate system customizability. For this reason, I implemented an optional process wherein an ArUco marker is recognized by the aircraft's imaging system and the appropriate transform from the observer to the desired origin-point is factored into the transform tree.

The OpenCV library used frequently throughout this work offers a method for detecting ArUco markers [9]. This method of augmentation to a visually-observed space allows for the relative pose of the observer to be calculated. The detection of

these markers requires that the intrinsics of the imaging system are known a priori, and the equation representing how the image space (the space defined by the upper lefthand corner of the imaging sensor) to camera space (the projection of the image space onto the plane at the focal point of the camera with pixels mapping onto their equivalent points in physical space at the intersection of the focal point plane) is shown below. In addition, the figure below shows an interpretation of this transform wherein a nadir-facing camera "sees" the ArUco marker first with respect to the image plane, and then the projection onto the camera plane via the intrinsics. The following figure shows a representation of the aircraft in gazebo with a depiction of the virtual "frustom" onto which the projection occurs.

The sequence of transforms to measure the marker origin relative to the local home of the drone is marker to image plane to camera plane to drone frame to local home. Note that this transform tree only needs to be calculated once as both the local home and the ArUco origin points are static with respect to the environment. As such, a point relative to the ArUco marker may then have its frame of reference changed by the transform shown here, represented with a single matrix M_a^h, where 'h' refers to the local home's frame and 'a' refers to the ArUco frame:

$$p_a = M_a^h p_h$$

, where

$$M_a^h = T_a^i T_i^c T_c^b T_b^h$$

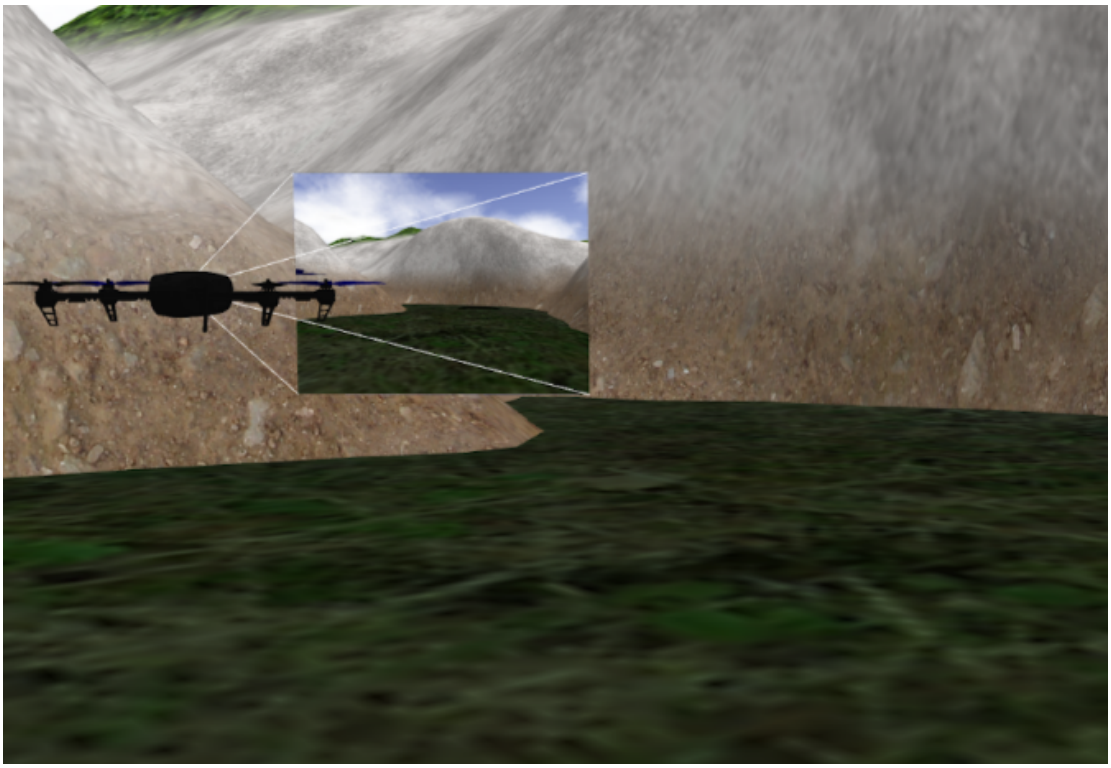

Figure 3.3: The virtual frustom depicting how the image plane values are projected onto the camera frame. Flying the IRIS UAS model within Gazebo. It is running the PX4 flight stack. The simulated frustom with an image representing what the UAS sees.

Eq. Process for transforming a point in the local home space to the ArUco-oriented space.

Here, p_a is the 3D point measured with respect to the ArUco marker's coordinate system, p_h is the 3D point measured with respect to PX4's local home position, M_a^h is the matrix transform from the local home to the ArUco frame, T_a^i is the transform from the ArUco frame to the image frame, T_i^c is the transform from the image frame to the camera frame, T_c^b is the transform from the camera frame to the body frame, and T_b^h is the transform from the body frame to the local home frame.

3.4.2 Feature Point Downselection

This point downselection process is implemented via a ROS node, identified here as "Point cloud local height estimator", which listens to the map point data published by ORB-SLAM and from which the downselection of points as well as the relative height is published. The looping process listens for "PointCloud" messages, a representation encoded to work with the Open Point Cloud, or "OPC", library [59]. This node constructs a k-d tree from this received map point data using the aforementioned OpenCV approach. The pose of the ownship as estimated by ORB-SLAM is used to query which map points fall within a given distance, e.g. a 3 m radius, of the aircraft. This is accomplished by using the "radiusSearch" method of the k-d tree class.

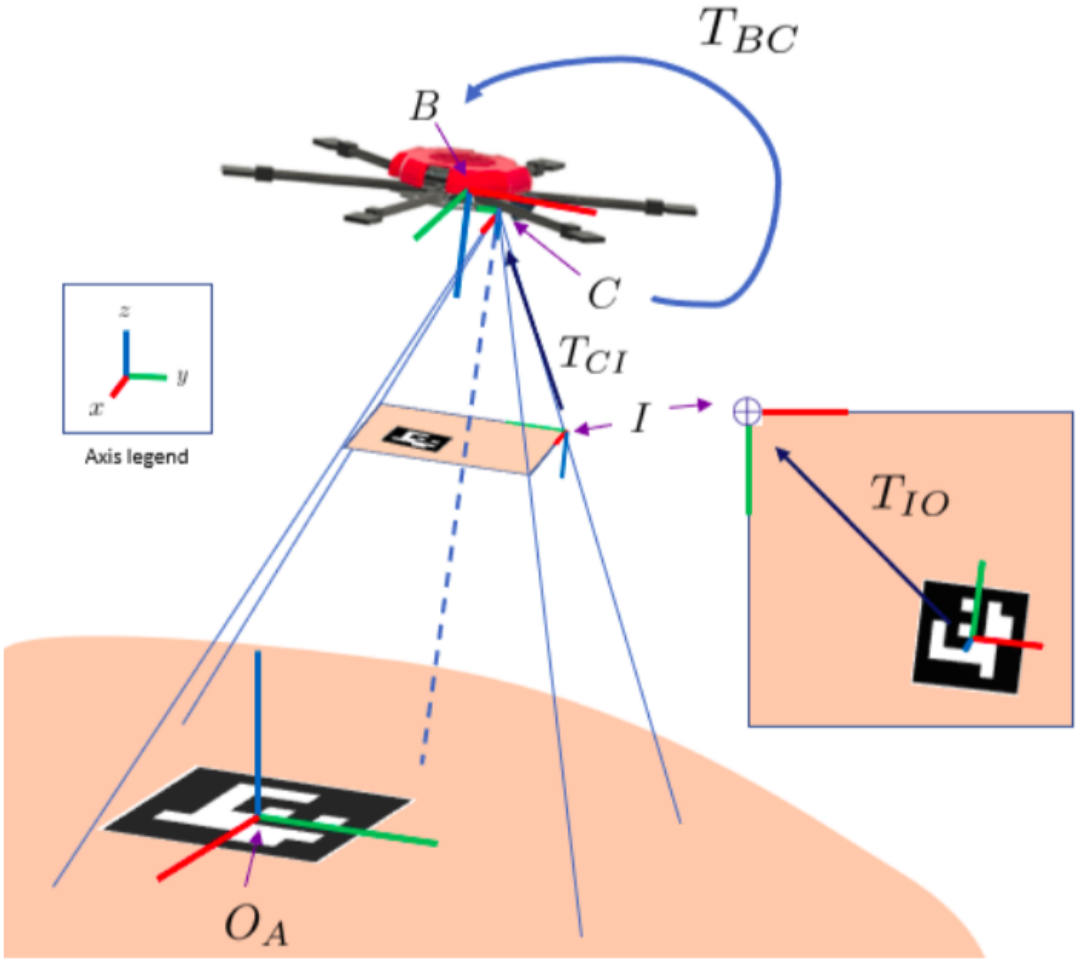

Figure 3.4: Illustration of the frame transforms undergone when calculating our home-to-ArUco frame manipulation. "C" represents the camera frame, "I" represents the image frame, "B" represents the body frame.

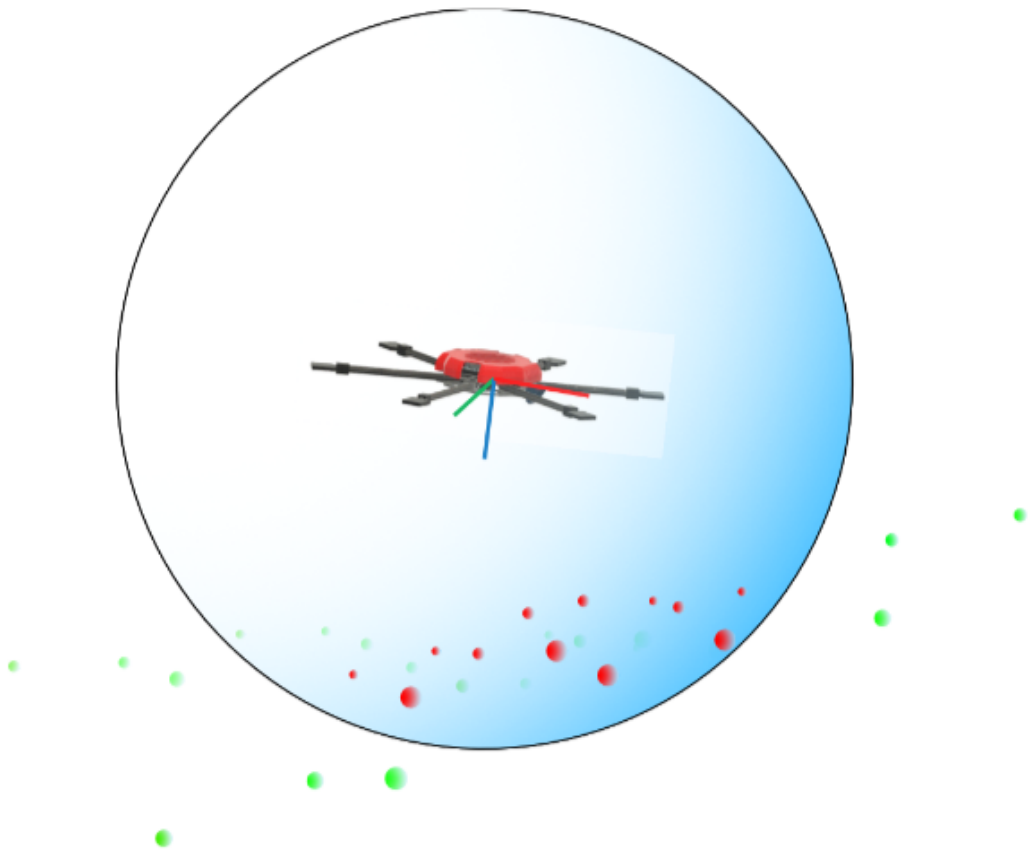

Figure 3.5: Illustration of the ORB-SLAM2-generated map points being downselected by radiusSearch using the estimated pose of the ownship. The red points shown are those which the search has rendered as being within the given radius.

3.4.3 Height Estimation

There is then the task of determining the relative height of the aircraft. This is done by calculating the average height of the map points detected within the radius of the aircraft, i.e., their encoded 'z' values, via the equation below. This aggregate height estimation of the terrain relative to the origin point set within ORB-SLAM is then subtracted from the reported height of the observer, again relative to the coordinate system of ORB-SLAM, rendering the height-above-terrain metric that we are interested in for terrain following purposes.

$$\hat{h}_r = \hat{h}_o - \sum_{i=0}^{j} p_{z,i}$$

Here, h_r is the relative height estimate of the UAS to the terrain, h_o is the estimated height of the UAS rendered by ORB-SLAM relative to its origin, and p_z, i is the i-th point within the downselection's z component.

The output of this search is also then encoded with the proper formatting in order to publish a new PointCloud message which only includes those points whose indices are identified by the radiusSearch method. This is done primarily for the purposes of visualization so that we may inspect which points are being considered by the terrain following algorithm. Pictures of this effect are presented in the "Results" section of this thesis.

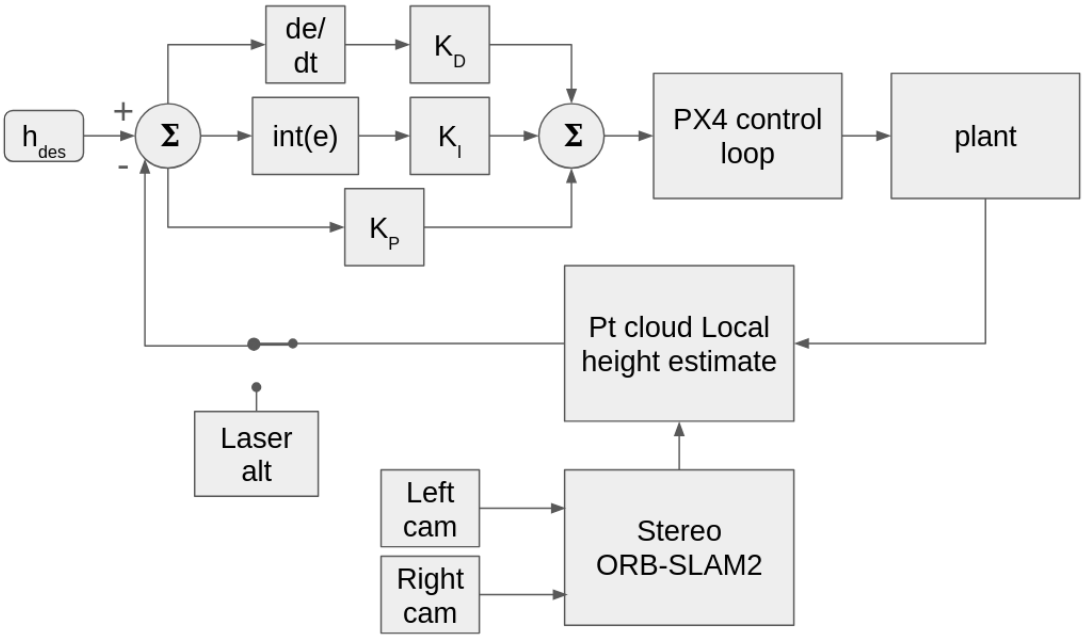

Figure 3.6: The outer loop PID controller for the vision-assisted terrain following algorithm.

3.4.4 Outer-loop PID Control

The relative height information generated in the manner outlined in the previous section is fed back to the system in the form of an outer-loop PID controller. The basic structure of this control loop is depicted in the block diagram below.

The setpoint for our control loop is the desired height which we want to follow the terrain with. The estimated height, provided by the point cloud local height estimator unless switched to the laser altimeter block, is subtracted from the setpoint to generate the error term. This error term is differentiated and integrated in order to generate the three branches necessary for PID. Each of these branches has a corre-

sponding gain value K which allows for tuning of the PID controller. Note that there is a switch which can toggle between the point cloud local height estimate block and the laser altimeter block – this is a dual purpose switch in that it (a) allows for a fallback functionality for when the ORB-SLAM2-sensed height calculation fails and (b) it is a convenient means of having an alternate system of terrain height estimation to calculate a baseline from.

An instance of a custom ROS node integrated with the MAVROS library functions and structures necessary to drive the aircraft's motions. This node first initializes the connection with the core MAVROS node, confirming that the aircraft is connected, after which it then prompts the system to arm, takeoff, and enter "Offboard" mode. This mode of operation allows for the custom node to send setpoints to the aircraft. The system then enters our state machine which will conduct the terrain following.

In most of our experiments using this custom driver node, we instruct the aircraft to go to a desired region to conduct the terrain following, after which we drop into a constant forward velocity loop which is repeatedly running our control loop to keep our estimated separation from the terrain. Both the laser altimeter-based approach implemented and the VaTF approach use this same scheme. The primary distinction between the two is in which type of sensed data is used to achieve terrain following.

For VaTF, the system listens for the topic publishing the relative height information estimated in the manner discussed previously via the point cloud processing node. The messages on this topic are fed into the PID control loop as an error calculation (i.e., the desired setpoint for our height has the estimated height based on the

point cloud calculation subtracted from it). This error is used in the context of the previously described PID equation. As the system flies and experiences deviations from the desired setpoint, the system self corrects and keeps a consistent height from the terrain as it goes.

3.5 Simulation Components

3.5.1 Gazebo

Gazebo is used as the simulation package. This simulator works well in tandem with ROS, and PX4 has several models which can be used which are provided by the repository. We used Gazebo 11, which works well with ROS Melodic.

3.5.2 PX4

We use the PX4 firmware as the primary firmware for the system. This is an autopilot firmware built on the NuttX operating system. It is one of the de facto firmwares for Pixhawks, along with Ardupilot.

3.5.2.1 Simulated Vehicle

The simulated vehicle in the Gazebo simulations is the Iris model which ships with the PX4 firmware repository. A stereo camera instance is attached to the front of the model in an oblique-nadir configuration. A laser altimeter is modeled using the LiDAR SDF model with only a single beam, and it is attached to the simulated drone

to point directly nadir.

3.5.3 ROS Integration

ROS, the "Robotic Operating System" is used as a means of interconnecting several components of the system. A diagram of the layout of ROS nodes for our system is shown in a section below. This framework is especially convenient because ORB-SLAM2 has ready-made ROS nodes.

3.5.3.1 MAVROS

MAVROS is the ROS package used for interfacing with PX4. We are able to send setpoints for our control loop by prompting the system to go into "Offboard" mode making it able to receive directives from outside the flight controller.

We use a combination of ROS + MAVROS as a way to connect the flight controller to our ROS system. The mode of operation that we are most interested in when using this scheme is "Offboard" mode. This mode is activated in any fashion where a MAVLink message can be sent to the flight controller, including via MAVROS itself (further discussed in the "Offboard node" section). When active, the flight controller accepts setpoints from outside of itself. As we use a "companion computer", we may then send setpoints from our MAVROS-linked node which are ingested by the flight controller.

3.5.3.2 Offboard Operation

The primary node of the system architecture is the node responsible for providing the control loop to accomplish the vision-assisted terain following. This node is initialized as a ROS1 node wherein the subscriptions, publications, and service clients are first set up. The node then attempts a connection with the MAVROS instance (i.e., the ROS node which is directly connected to the flight controller, being either simulated or physical). Upon connection, a stream of empty setpoints are pre-loaded to buffer the stream for when we begin running in "Offboard" mode. The node then requests that we enter "Offboard" mode. Upon successful engagement of "Offboard" mode, the system arms and takes off to a predetermined intermediary start point. After a short delay, the system traverses normally to the actual start point of the terrain following mission, at which point the VaTF algorithm runs. A flow chart is included below to depict this whole process.

3.6 Results

3.6.1 ArUco-based Origin Placement

As mentioned previously in the thesis, the ArUco marker was used to set an auxiliary origin point optionally when conducting the terrain following missions. Seen below is the representation of an aruco marker in the simulated environment. Region (a) is the view from within Gazebo of our simulation in which we see an Iris drone and its virtual frustom positioned looking at a virtual ArUco marker which is placed on the

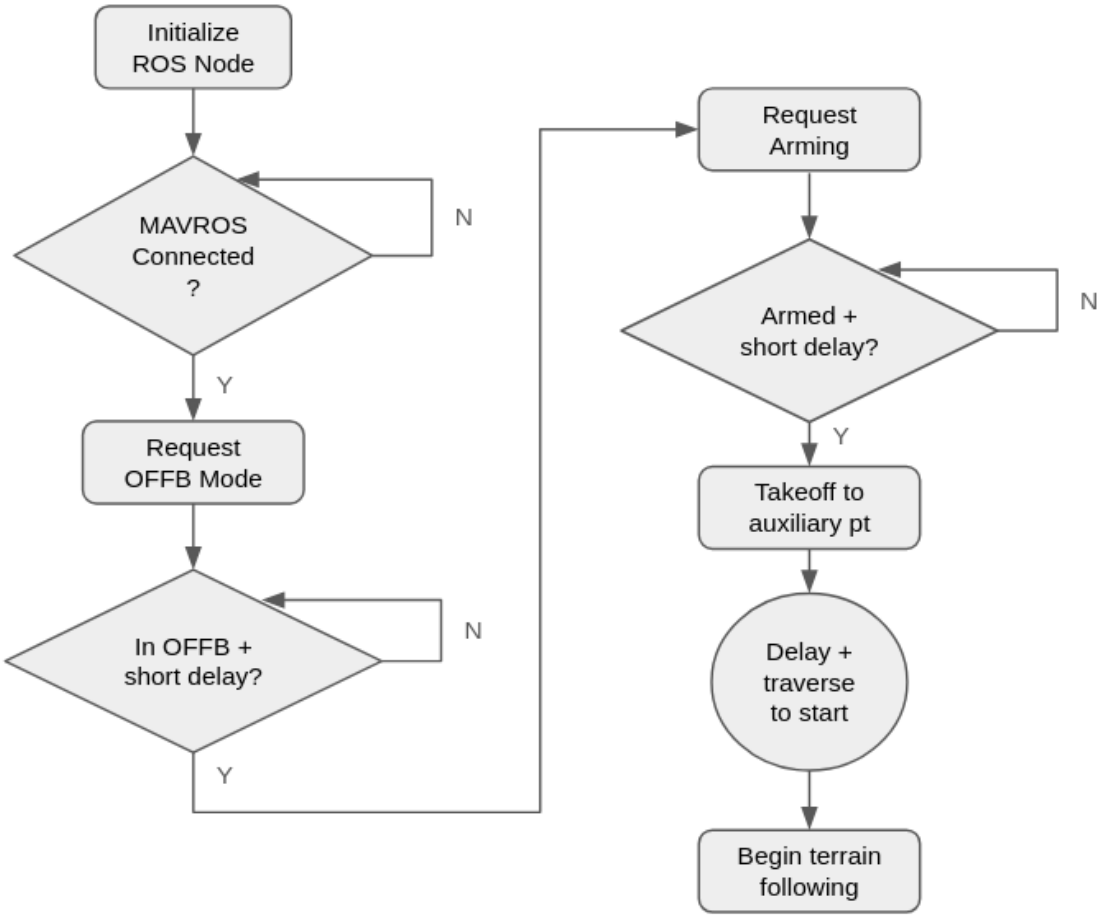

Figure 3.7: Flow chart representing the execution process for the offboard control node.

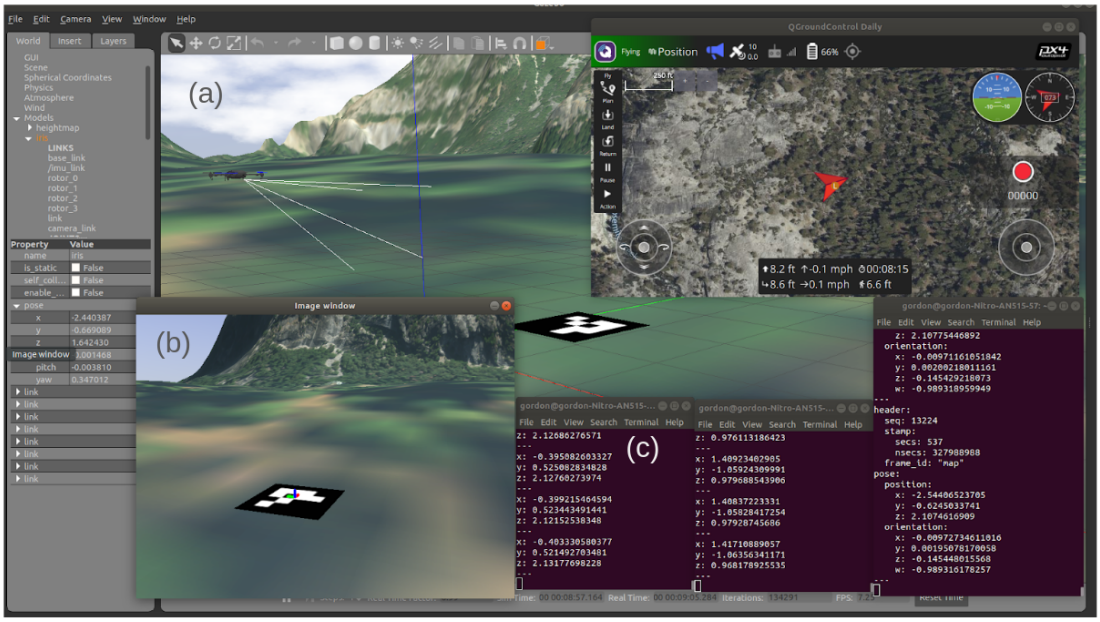

Figure 3.8: The UAS registering an ArUco marker within the simulation. This is used for the option of assigning a freely-selected origin point to the ORB-SLAM map by placing the marker in the desired spot when in the field. Region (a) depicts the simulator view of the scenario. Region (b) is the view from the vantage point of the drone with the ArUco marker identified as evidence by the small set of axes placed on the marker. Region (c) depicts the outputs of the calculations for the coordinate transforms applied.

ground. Region (b) of the figure shows what the drone is seeing and has a small set of axes placed on the detected ArUco marker which indicates that the relative pose of the marker with respect to the aircraft has been successfully computed. Finally, region (c) shows the output values of the relative pose of the marker and the drone.

3.6.2 Vision-assisted Terrain Following

The outer-loop PID controller of the system was tuned by iterative refinement. Seen below are some example trajectories which were flown with the described k values.

[put side-by-side examples of underdamped, critically damped, and over-damped PID gain values with the trajectories flown]

Multiple runs of the simulation were conducted in which the system repeatedly visited a part of the map. An aggregation of the heights and the deviations from the height is presented. This is demonstrating the performance versus using the simulated laser altimeter device.

3.7 Discussion

Outer-loop controller design One consideration for the performance of this system is that we do not vary the "forward" velocity for different terrain types and scenarios. It would benefit the research here to experiment with different speeds and to see how it affects the flight of the drone for different k values in our control loop. It very likely would be the case that the system would need to have different tunings for different velocity requirements. A discussion of the potential improvements to this controller is offered in the "Future works" section which describes how the control loop may be replaced by Model Predictive Control, which would very likely mitigate this discrepancy.

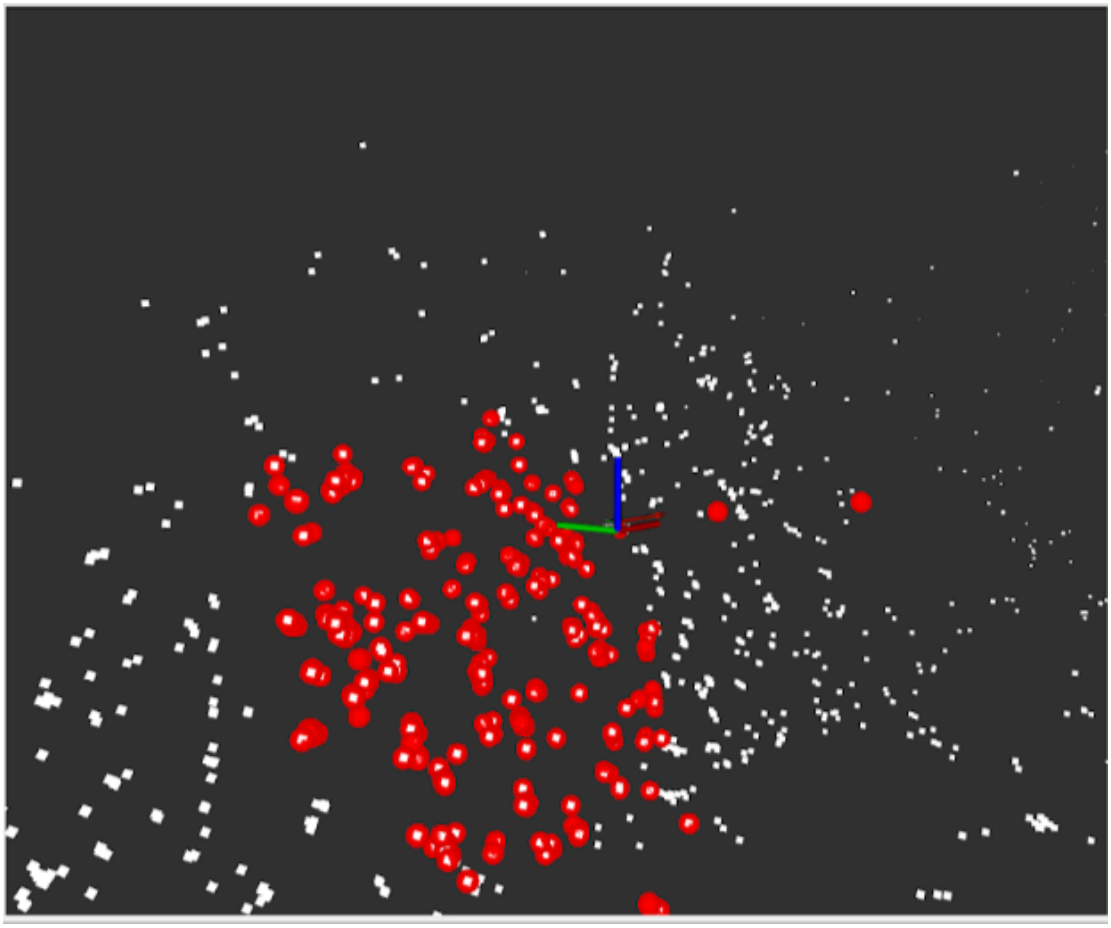

Figure 3.9: Fig. Downselection of the nearby map points generated by stereo ORB-SLAM2. Points highlighted in red fall within a 5 m radius of the multicopter system, represented here by the floating set of axes. The images on the bottom left of the image represent what the UAS sees with one of the two nadir-oblique facing cameras.

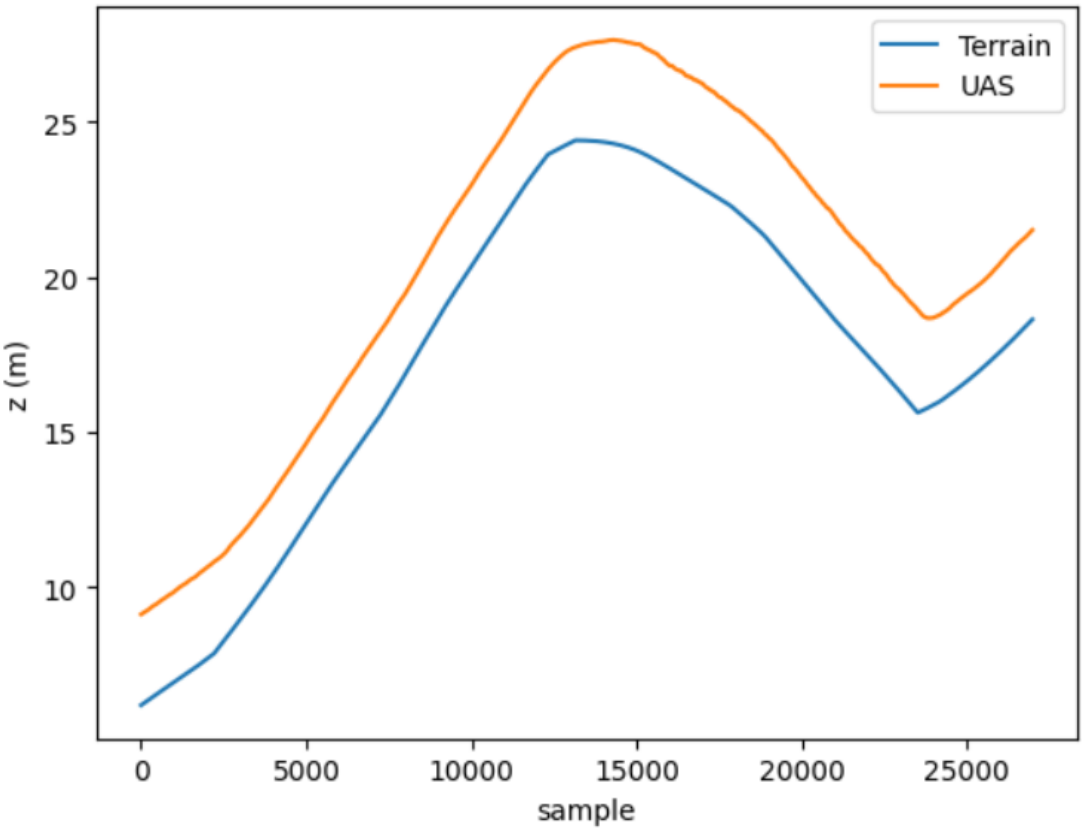

Figure 3.10: Fig. UAS trajectory while in the vision-assisted terrain following mode. The orange line depicts the z value of the aircraft, and the blue line is the height of the terrain. The forward velocity for this case was 0.5 m/s with a tracking height of 3m.

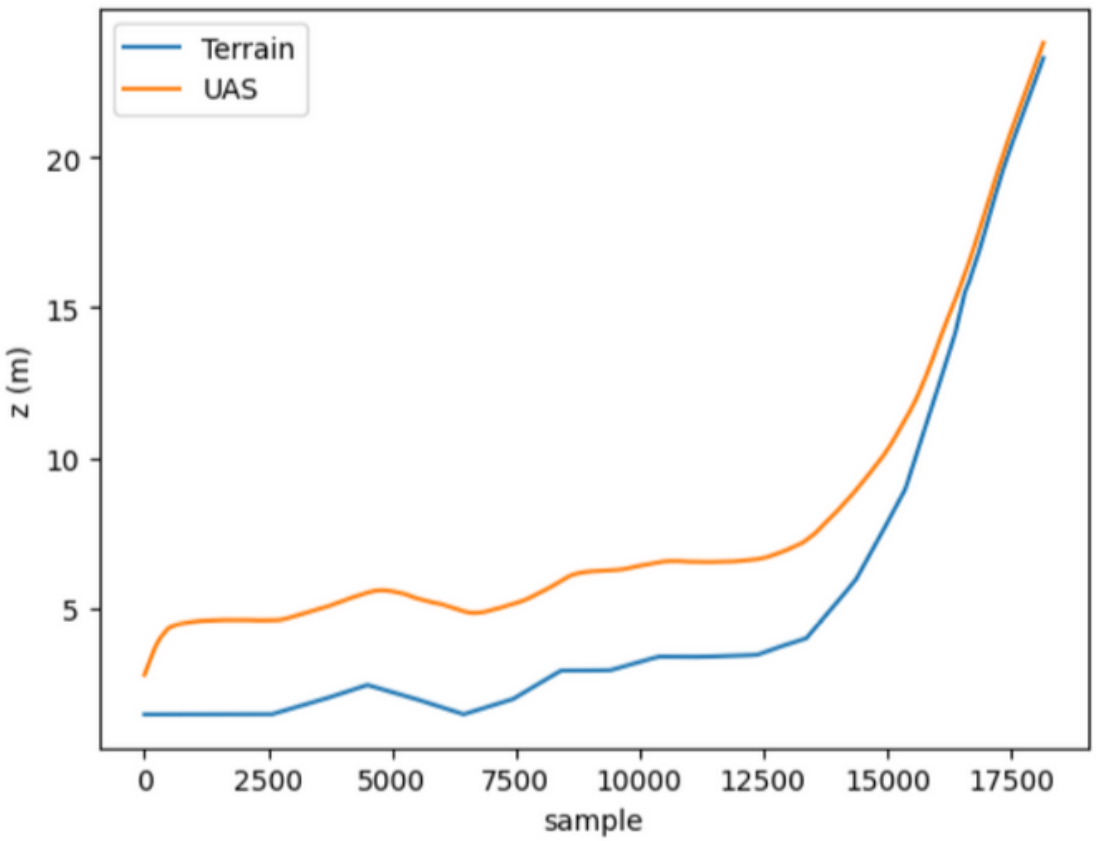

Figure 3.11: Fig. Another UAS trajectory while in the vision-assisted terrain following mode. The forward velocity for this case was 2 m/s with a tracking height of 3m.

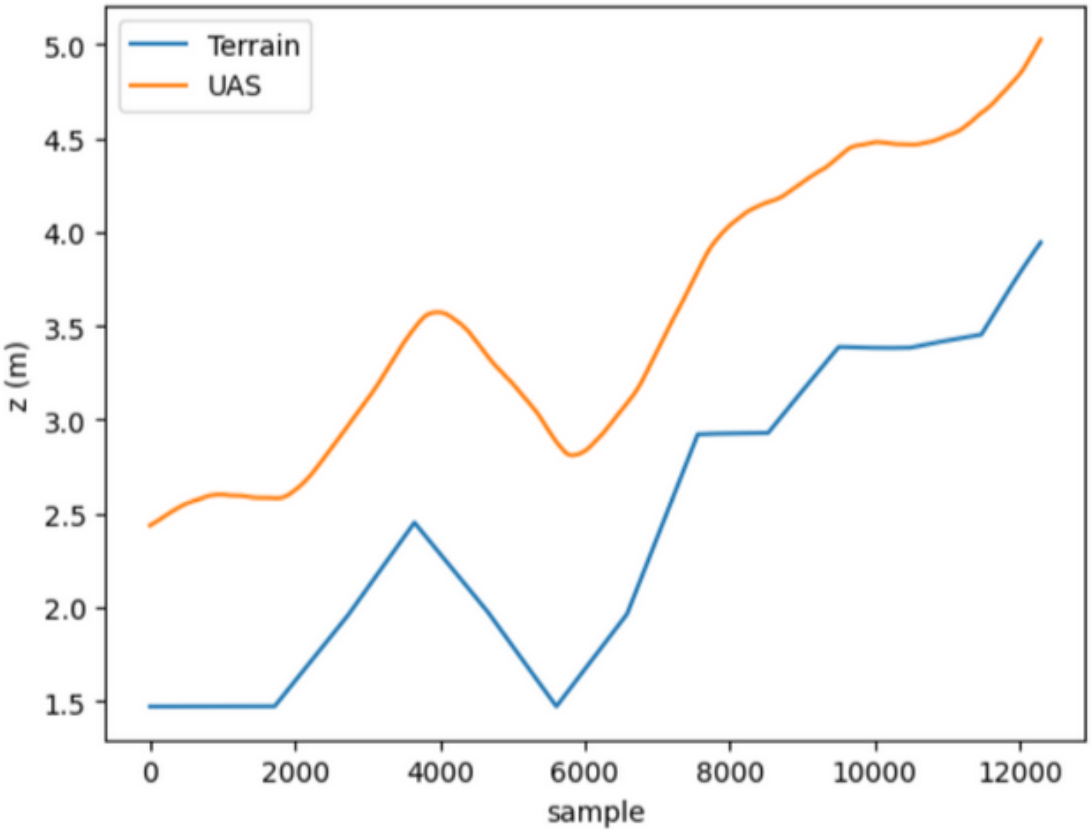

Figure 3.12: Fig. Another UAS trajectory while in the vision-assisted terrain following mode. The forward velocity for this case was 2 m/s with a tracking height of 1m.

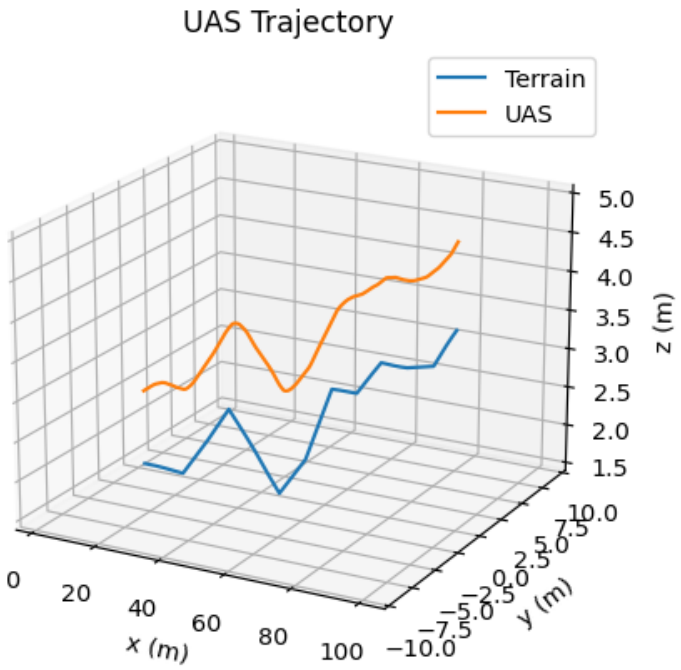

Figure 3.13: Fig. 3D-view of the previous trajectory of the aircraft conducting terrain following.

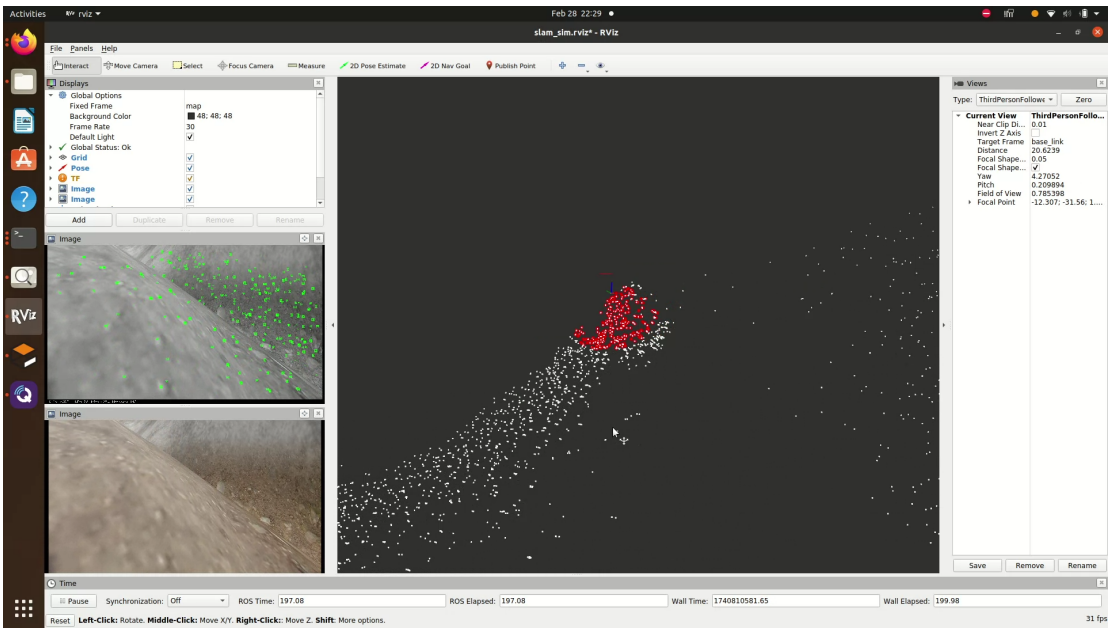

Figure 3.14: Fig. RViz view of the VaTF experiment execution. Region (a) shows the feature point mapping process of ORB-SLAM on the stereo images aggregated by the system in flight, for which region (b) shows the raw image. Region (c) shows the map points representing the convex hull of the terrain which are produced and maintained by ORB-SLAM, and the points colored red are those which fall within the predetermined distance of the aircraft, in this case being 3 m.

Figure 3.15: Fig. The simulated Iris aircraft conducting a terrain following mission in the Gazebo map “Yosemite”.

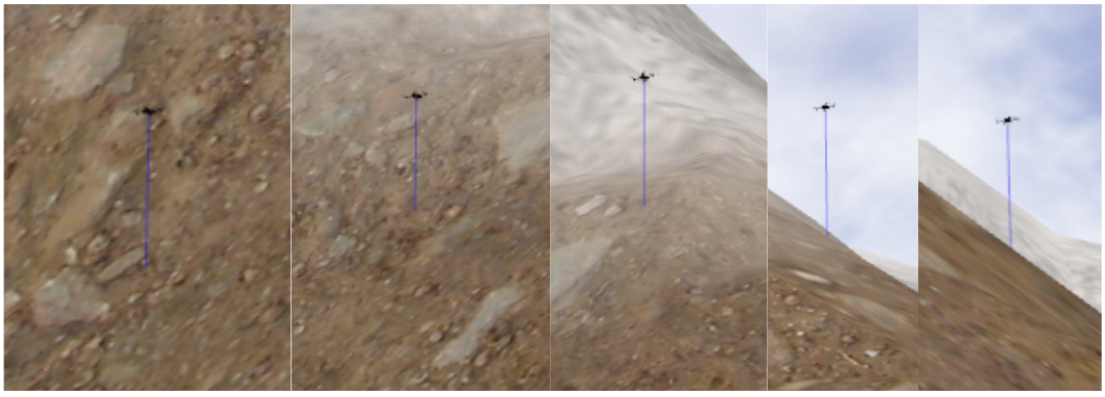

Figure 3.16: FIg. Stitched together images of the multirotor system at different moments in its terrain-following trajectory.

3.7.1 Sensor mode switching

The capability to switch between the vision-based height estimate method and the laser altimeter readings was implemented in this work to allow for an operational fallback mode. However, when running the vision-assisted terrain following method, it was found that the laser altimeter was being used in lieu of the vision estimate approximately half of the time. The reason for this is that the speed of the point cloud local height estimate node was looping at approximately half the rate of the PID control loop. This suggests that optimization around the point cloud down-selection process and height estimation portion of the algorithm needs to be optimized to update the PID at its same frequency or faster.

3.7.2 Vertical clearance

One aspect of the vision-assisted terrain following methodology which could be improved upon is to keep an absolute distance from the map points detected rather than simply gauging height. In this configuration, the map points would each impart a virtual force on the aircraft to try to keep it within the desired distance, similar to a potential field-based scheme (Hwang Ahuja, 1992).

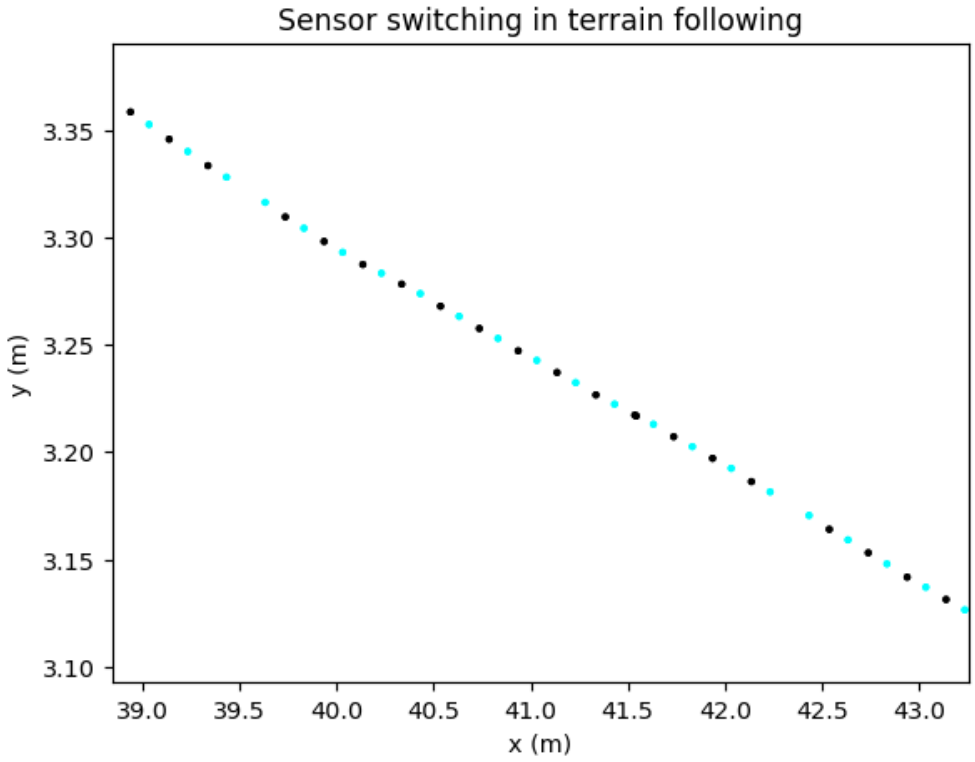

Figure 3.17: Fig. Depiction of the mode switching which occurs with the current implementation of the VaTF algorithm. The vision detection is valid 50% of the time, and the laser altimeter is fallen back upon 50% of the time. The blue dots indicate moments when the vision height estimate is used in the control loop, and the black dots are the times where laser altimeter is used.

Chapter 4

Conclusion

In this work, we explored the use of Visual SLAM onboard multicopter platforms. This was accomplished by undertaking an analysis of localization performance for the monocular ORB-SLAM2 algorithm when flown onboard a hexacopter platform in an indoor environment. Several different flight patterns and modes of control (i.e., manual and autonomous) were flown to aggregate a large dataset to conduct the analyses. The outcomes of these analyses revealed the overall reliability of the algorithm, and it informed next steps on maximum feature number experiments to conduct to further stress-test its performance. Various analytical methods were employed to coalesce findings from both manual and autonomous flights within the indoor flight space. In particular, we observed the interrelatedness between feature number detection and multiple state variables of the aircraft.

This was then followed by the creation of a Vision-assisted Terrain Following mode of flight leveraging stereo ORB-SLAM2. We pursued this initiative from the

perspective of its application in the geosciences to push the limits on existing techniques for terrain following. The algorithm developed here was tested in the Gazebo simulation environment, and it entailed the efficient downselection of map points generated by ORB-SLAM2 via a k-d tree search with a local height estimation calculation to factor in the surrounding terrain. This methodology was tested at different maintained heights and varying speeds to get a sense of how the system works in these different situations.

Ultimately, wem created a work combining the in-depth analysis of the functionality of ORB-SLAM2 aboard multicopter platforms and creating an applied example of how the algorithm may be useful to researchers employing UAS.

Chapter 5

Future works

One of the major developments to be made were this research continued would be to test the effectiveness of the VaTF algorithm onboard a physical platform. Seen below is the system which was constructed using a Pixhawk 4 as the flight controller unit, an Intel NUC as the companion computer, a Holybro GPS module, an Arducam camera (however, this would be doubled to do stereo ORB-SLAM), a Holybro SiK radio telemetry module, PX4Flow optical flow module, and two 4S LiPo batteries. The algorithm would be conducted in an area with undulating terrain to demonstrate how well the system tracks it.

Several improvements could be made to this system to make it perform better. The first of these changes which could be made is to use stereo visual inertial slam. The "inertial" part of this system is the IMU data which is factored into the calculation of the map point cloud and observer pose within SLAM. Even though the system does incorporate IMU data within the external kalman filter of the autopilot, the ORB-

Figure 5.1: Fig. Physical hexacopter platform constructed to evaluate the effectiveness of the algorithm with. The system was fully developed, however experiments using the VaTF algorithm were not exercised.

SLAM mapping would see improvement likely leading to better performance of the overall system.

Another aspect of the system that could be improved upon is the means of control. We used an outer loop PID controller for the incorporation of the sensed relative height into the multicopter outermost layer, known as offboard control. However, the plan for the system long-term was to use a more sophisticated method of control known as Model Predictive Control, or MDP. The implementation of this would involve creating an iteratively updated simulation of what a string of control commands would do to the system, and the most optimal set of commands within this time horizon is used. Only the first input command of this sequence is executed, and the whole process is started again. This method of control makes sense for conducting precision terrain following for a number of reasons. First, decisions can be factored into the construction of the algorithm in the definition of the optimization function(s) used. For example, if there were enough reason to do so, having the system consider energy expenditure and allowing room for deviation from the intended trajectory could mean smoothly switching between following terrain vertically and conducting avoidance or circumnavigating regions for which the cost of energy expenditure isn't worth the ascent and descent. Another reason that this approach would be useful is that it could allow for variability in some of the operational parameters, such as speed. When the system recognizes that it is safe and advantageous to do so, in theory, one could implement the logic in this controller to vary the velocity.

I would have liked to have included multilateration localization in this work. This would have involved using UWB nodes to aid in the localization of the system

so that it would be able to geotag data collected during missions very accurately. In addition, this multilateration approach could further improve the fidelity of the control of the system by serving as another sensor input.

Lastly, to finish designing a highly capable aircraft for geoscientific research, we would have wanted to incorporate so-called "Payload Directed Flight" into the functionality (Ippolito et al., 2009). This mode of operation uses the payload data (for example, gas samples aggregated or magnetic signatures in studying magma at depth) to inform where the system should be flown. This introduces a layer of autonomy to the system which can enhance the efficiency of operations undergone while out in the field.

Bibliography

[1] Oaisys: A photorealistic terrain simulation pipeline for unstructured outdoor environments.

[2] Efstathios Adamopoulos and Fulvio Rinaudo. Uas-based archaeological remote sensing: Review, meta-analysis and state-of-the-art. *Drones*, 4(3):46, 2020.

[3] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[4] Noah D Athens, Jonathan MG Glen, Robert L Morin, and Simon L Klemperer. Atv magnetometer systems for efficient ground magnetic surveying. *The Leading Edge*, 30(4):394–398, 2011.

[5] PX4 Autopilot. Mavros offboard control example (c++), 2023. `https://docs.px4.io/main/en/ros/mavros_offboard_cpp.html` [Accessed: 8/24/2023].

[6] Jonathas Barreto, Luciano Cajaíba, João Batista Teixeira, Lorena Nascimento, Amanda Giacomo, Nelson Barcelos, Ticiana Fettermann, and Agnaldo Martins.

Drone-monitoring: Improving the detectability of threatened marine megafauna. *Drones*, 5(1):14, 2021.

[7] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1000–1006. IEEE, 1997.

[8] Jon Louis Bentley. K-d trees for semidynamic point sets. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, 1990.

[9] Gary Bradski, Adrian Kaehler, et al. Opencv. *Dr. Dobb's journal of software tools*, 3(2), 2000.

[10] Frédéric Bretar and Nesrine Chehata. Terrain modeling from lidar range data in natural landscapes: A predictive and bayesian framework. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1568–1578, 2009.

[11] Stefano Campana. Drones in archaeology. state-of-the-art and future perspectives. *Archaeological Prospection*, 24(4):275–296, 2017.

[12] Igor SG Campos, Erickson R Nascimento, Gustavo M Freitas, and Luiz Chaimowicz. A height estimation approach for terrain following flights from monocular vision. *Sensors*, 16(12):2071, 2016.

[13] Steven Carey and Marcus Bursik. Volcanic plumes. In *The encyclopedia of volcanoes*, pages 571–585. Elsevier, 2015.

[14] CW Chan and TY Kam. A procedure for power consumption estimation of multi-rotor unmanned aerial vehicle. In *Journal of Physics: Conference Series*, volume 1509, page 012015. IOP Publishing, 2020.

[15] Weifeng Chen, Guangtao Shang, Aihong Ji, Chengjun Zhou, Xiyang Wang, Chonghui Xu, Zhenxiong Li, and Kai Hu. An overview on visual slam: From tradition to semantic. *Remote Sensing*, 14(13):3010, 2022.

[16] Shih-Hong Chio and Cheng-Horng Lin. Preliminary study of uas equipped with thermal camera for volcanic geothermal monitoring in taiwan. *Sensors*, 17(7):1649, 2017.

[17] Alejo Concha and Javier Civera. Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693. IEEE, 2015.

[18] Jill J Cress, Michael E Hutt, Jeff L Sloan, Mark A Bauer, Mark R Feller, and Susan E Goplen. Us geological survey unmanned aircraft systems (uas) roadmap 2014. Technical report, US Geological Survey, 2015.

[19] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[20] Lorenzo Fiori, Ashray Doshi, Emmanuelle Martinez, Mark B Orams, and Barbara

Bollard-Breen. The use of unmanned aerial systems in marine mammal research. *Remote Sensing*, 9(6):543, 2017.

[21] Arnau Folch, Antonio Costa, and Giovanni Macedonio. Fall3d: A computational model for transport and deposition of volcanic ash. *Computers & Geosciences*, 35(6):1334–1342, 2009.

[22] Matthew A Garratt and Javaan S Chahl. Vision-based terrain following for an unmanned rotorcraft. *Journal of Field Robotics*, 25(4-5):284–301, 2008.

[23] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A research platform for visual-inertial estimation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[24] Daniele Giordan, Marc S Adams, Irene Aicardi, Maria Alicandro, Paolo Allasia, Marco Baldo, Pierluigi De Berardinis, Donatella Dominici, Danilo Godone, Peter Hobbs, et al. The use of unmanned aerial vehicles (uavs) for engineering geology applications. *Bulletin of Engineering Geology and the Environment*, 79:3437–3481, 2020.

[25] Pan Hai-yang and Liu Shun-an. A special terrain-following system based on flash lidar. In *2012 IEEE International Conference on Mechatronics and Automation*, pages 653–657. IEEE, 2012.

[26] Soufiane Hajaj, Abderrazak El Harti, Amin Beiranvand Pour, Amine Jellouli, Zakaria Adiri, and Mazlan Hashim. A review on hyperspectral imagery application

for lithological mapping and mineral prospecting: Machine learning techniques and future prospects. *Remote Sensing Applications: Society and Environment*, page 101218, 2024.

[27] Samuel C Hassler and Fulya Baysal-Gurel. Unmanned aircraft system (uas) technology and applications in agriculture. *Agronomy*, 9(10):618, 2019.

[28] Daniel C Herrera, Kihwan Kim, Juho Kannala, Kari Pulli, and Janne Heikkilä. Dt-slam: Deferred triangulation for robust slam. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 609–616. IEEE, 2014.

[29] Yong Koo Hwang, Narendra Ahuja, et al. A potential field approach to path planning. *IEEE transactions on robotics and automation*, 8(1):23–32, 1992.

[30] Corey Ippolito, Yoo-Hsiu Yeh, and Stefan Campbell. A trajectory generation approach for payload directed flight. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1351, 2009.

[31] Robert Jackisch, Yuleika Madriz, Robert Zimmermann, Markku Pirttijärvi, Ari Saartenoja, Björn H Heincke, Heikki Salmirinne, Jukka-Pekka Kujasalo, Louis Andreani, and Richard Gloaguen. Drone-borne hyperspectral and magnetic data integration: Otanmäki fe-ti-v deposit in finland. *Remote sensing*, 11(18):2084, 2019.

[32] Sandra Jakob, Robert Zimmermann, and Richard Gloaguen. Processing of drone-borne hyperspectral data for geological applications. In *2016 8th Workshop on*

Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), pages 1–5. IEEE, 2016.

[33] Mike R James, Brett Carr, Fiona D'Arcy, Angela Diefenbach, Hannah Dietterich, Alessandro Fornaciai, Einat Lev, Emma Liu, David Pieri, Mel Rodgers, et al. Volcanological applications of unoccupied aircraft systems (uas): Developments, strategies, and future challenges. *Volcanica*, 3(1):67–114, 2020.

[34] Benjamin R Jordan. Collecting field data in volcanic landscapes using small uas (suas)/drones. *Journal of Volcanology and Geothermal Research*, 385:231–241, 2019.

[35] Konstantinos Karydis and Vijay Kumar. Energetics in robotic flight at small scales. *Interface focus*, 7(1):20160088, 2017.

[36] Christoph Kern, Allan H Lerner, Tamar Elias, Patricia A Nadeau, Lacey Holland, Peter J Kelly, Cynthia A Werner, Laura E Clor, and Mike Cappos. Quantifying gas emissions associated with the 2018 rift eruption of kīlauea volcano using ground-based doas measurements. *Bulletin of Volcanology*, 82(7):55, 2020.

[37] Chong-sup Kim, In-je Cho, Dong-Kyu Lee, and Im-Ju Kang. Development of low altitude terrain following system based on terain profile matching. *Journal of institute of control, robotics and systems*, 21(9):888–897, 2015.

[38] 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*, pages 225–234. IEEE, 2007.

[39] Jongwon Lee, David Hanley, and Timothy Bretl. Extrinsic calibration of multiple inertial sensors from arbitrary trajectories. *IEEE Robotics and Automation Letters*, 7(2):2055–2062, 2022.

[40] Emma J Liu, Kieran Wood, Emily Mason, Marie Edmonds, Alessandro Aiuppa, Gaetano Giudice, Marcello Bitetto, Vincenzo Francofonte, Steve Burrow, Thomas Richardson, et al. Dynamics of outgassing and plume transport revealed by proximal unmanned aerial system (uas) measurements at volcán villarrica, chile. *Geochemistry, Geophysics, Geosystems*, 20(2):730–750, 2019.

[41] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[42] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22(22), 2011.

[43] Chenxiang Ma, You Zhou, and Zhiqiang Li. A new simulation environment based on airsim, ros, and px4 for quadcopter aircrafts. In *2020 6th international conference on control, automation and robotics (ICCAR)*, pages 486–490. IEEE, 2020.

[44] Muhammad Maaruf, Magdi Sadek Mahmoud, and Alfian Ma'arif. A survey of control methods for quadrotor uav. *International Journal of Robotics and Control Systems*, 2(4):652–665, 2022.

[45] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.

[46] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 ieee international conference on robotics and automation*, pages 2992–2997. IEEE, 2011.

[47] Carmelo Donato Melita, Dario Calogero Guastella, Luciano Cantelli, Giuseppe Di Marco, Irene Minio, and Giovanni Muscato. Low-altitude terrain-following flight planning for multirotors. *Drones*, 4(2):26, 2020.

[48] Marcus G Müller, Samantha Stoneman, Ingo von Bargen, Florian Steidle, and Wolfgang Stürzl. Efficient terrain following for a micro aerial vehicle with ultra-wide stereo cameras. In *2020 IEEE Aerospace Conference*, pages 1–9. IEEE, 2020.

[49] Marcus Gerhard Müller, Maximilian Durner, Abel Gawel, Wolfgang Stürzl, Rudolph Triebel, and Roland Siegwart. A photorealistic terrain simulation pipeline for unstructured outdoor environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9765–9772. IEEE, 2021.

[50] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam:

A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[51] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[52] PC Oxley. Terrain following and terrain avoidance algorithms. In *IEE Colloquium on Navigation, Guidance and Control on Aerospace*, pages 2–1. IET, 1989.

[53] Luís Pádua, Jakub Vanko, Jonáš Hruška, Telmo Adão, Joaquim J Sousa, Emanuel Peres, and Raul Morais. Uas, sensors, and data processing in agroforestry: A review towards practical applications. *International journal of remote sensing*, 38(8-10):2349–2391, 2017.

[54] Quan Quan. *Introduction to multicopter design and control*, volume 10. Springer, 2017.

[55] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[56] Guilherme V Raffo, Manuel G Ortega, and Francisco R Rubio. An integral predictive/nonlinear h control structure for a quadrotor helicopter. *Automatica*, 46(1):29–39, 2010.

[57] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 430–443. Springer, 2006.

[58] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[59] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[60] Raza Samar and Abdur Rehman. Autonomous terrain-following for unmanned air vehicles. *Mechatronics*, 21(5):844–860, 2011.

[61] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[62] Chandra Has Singh, Vishal Mishra, and Kamal Jain. High-resolution mapping of forested hills using real-time uav terrain following. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 10:665–671, 2023.

[63] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[64] Mandyam V Srinivasan, Saul Thurrowgood, and Dean Soccol. An optical system for guidance of terrain following in uavs. In *2006 IEEE International conference on video and signal based surveillance*, pages 51–51. IEEE, 2006.

[65] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664. IEEE, 2010.

[66] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, 2018.

[67] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ transactions on computer vision and applications*, 9(1):16, 2017.

[68] Chiara Torresan, Andrea Berton, Federico Carotenuto, Salvatore Filippo Di Gennaro, Beniamino Gioli, Alessandro Matese, Franco Miglietta, Carolina Vagnoli, Alessandro Zaldei, and Luke Wallace. Forestry applications of uavs in europe: A review. *International journal of remote sensing*, 38(8-10):2427–2447, 2017.

[69] Dimosthenis C Tsouros, Stamatia Bibi, and Panagiotis G Sarigiannidis. A review on uav-based applications for precision agriculture. *Information*, 10(11):349, 2019.

[70] Ryan M Turner, Mary M MacLaughlin, and Stephen R Iverson. Identifying and

mapping potentially adverse discontinuities in underground excavations using thermal and multispectral uav imagery. *Engineering geology*, 266:105470, 2020.

[71] Emmanouil Tzorakoleftherakis and Todd D Murphey. Iterative sequential action control for stable, model-based control of nonlinear systems. *IEEE Transactions on Automatic Control*, 64(8):3170–3183, 2018.

[72] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[73] Callum Walter, Alexander Braun, and Georgia Fotopoulos. High-resolution unmanned aerial vehicle aeromagnetic surveys for mineral exploration targets. *Geophysical Prospecting*, 68(1-Cost-Effective and Innovative Mineral Exploration Solutions):334–349, 2020.

[74] Thomas C Wilkes, Tom D Pering, and Andrew JS McGonigle. Semantic segmentation of explosive volcanic plumes through deep learning. *Computers & Geosciences*, 168:105216, 2022.

[75] Georges Younes, Daniel Asmar, and Elie Shammas. A survey on non-filter-based monocular visual slam systems. *arXiv preprint arXiv:1607.00470*, 413:414, 2016.

[76] Laurie Antoinette Zielinski, Jonathan MG Glen, Tait E Earney, Grant H Rea-Downing, R Greg Vaughan, Peter J Kelly, Gordon H Keller, Branden James

Dean, and William Schermerhorn. Uas-based tools for mapping and monitoring hydrothermal systems: An example from mammoth lakes, california. *Geothermal Resources Council Transactions*, 46:1618–1637, 2022.