ESTIMATION, MAPPING AND NAVIGATION WITH MICRO AERIAL VEHICLES FOR INFRASTRUCTURE INSPECTION

Tolga Özaslan

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2020

Dedicated to my beloved wife Elif

Acknowledgments

Two sentences from Nursi's masterpiece *The Words* summarize my journey: *"Yes, one who acquires true belief may challenge the whole universe and be saved from the pressure of events in accordance with the strength of his belief"* and, *"Belief (iman) makes man into man, indeed, it makes him into a sultan"*. His *Words* shed light on my path throughout this long journey and saved me from being exhausted many times. Hence my greatest thanks go to my mentor Said Nursi.

This thesis would not have been possible without the encouragement and support of many people. Foremost, I would like to express my sincere gratitude and thanks to my advisers, Dr. Vijay Kumar, and Dr. CJ Taylor, for their endless support, patience, and guidance throughout my graduate studies. I feel very fortunate and privileged for being a student of such great mentors and roboticists.

I also owe my special thanks to my thesis committee members Dr. Kostas Daniilidis, Dr. Mark Yim and Dr. Nicholas Roy for their invaluable feedback and light-shedding guidance during the preparation of my thesis.

I would also like to thank my fellow lab members, for fields trips and experiments as well as many intellectual and scientific discussions over the years: James Keller, Giuseppe Loianno, Kartik Mohta, Omur Arslan, Shaojie Shen. Especially, I am very much indebted to James Keller and Giuseppe Loianno for their great help during the ardous, tiring, dirty and dusty flight tests inside penstocks.

I also acknowledge the scholarship by Republic of Turkey Ministry of National Education (MEB) during my graduate studies at University of Pennsylvania.

I would like to thank all of my friends, especially the Upper Darby crew, Abdurrahman

Yilmaz, Hayrettin Unsal, Mikail Temirel, Musa Dankaz, Sadik Seymen. And Mustafa Dogan, Haris Toprak, Celal Alagoz, Birkan Tunc made Philly feel like home for me. I can never forget their loyalty, cordiality and providing priceless moral support at all my hard times.

My special thanks to my family... No words is capable of expressing how grateful I am to my mother, father, sister, mother-in-law and father-in-law for all the sacrifices that you have made on my behalf. Your patient prayers and never-ending encouragement for me were what sustained me thus far.

The best outcome of my years in the US is finding my best friend, soul-mate and other-half. I would like to thank my loving and patient wife who never ceased her faithful support during my exhausting and grueling journey; especially in the moments when everyone else was unable to answer my queries.

ABSTRACT

ESTIMATION, MAPPING AND NAVIGATION WITH MICRO AERIAL VEHICLES FOR INFRASTRUCTURE INSPECTION

Tolga Özaslan

Vijay Kumar

Camilo J. Taylor

Multi-rotor Micro Aerial Vehicles (MAV) have become popular robotic platforms in the last decade due to their manufacturability, agility and diverse payload options. Amongst the most promising applications areas of MAVs are inspection, air delivery, surveillance, search and rescue, real estate, entertainment and photography to name a few. While GPS offers an easy solution for outdoor autonomy, using onboard sensors is the only solution for autonomy in constrained indoor environments. In this work, we study onboard state estimation, mapping and navigation of a small MAV equipped with a minimal set of sensors inside GPS-denied, axisymmetric, tunnel-like environments such as penstocks. We primarily focus on state estimators formulated for different sensor suits which include 2D/3D lidars, cameras, and Inertial Measurement Units (IMU). Penstocks are pitch dark environments and offer very weak visual texture even with onboard illumination, hence our estimators primarily rely on lidars and IMU. The point cloud data returned by the lidar consists of either elliptical contours or indiscriminate partial cylindrical patches making localization along the tunnel axis theoretically impossible. Cameras track features on the walls using the onboard illumination to estimate the velocity along the tunnel axis unobservable to range sensors. Information from all sensors are then fused in a central Kalman Filter for 6 Degrees-of-Freedom (DOF) state estimation. These approaches are validated through onsite experiments conducted in four different dams demonstrating state estimation, environment mapping, autonomous and shared control.

Contents

List of Figures

Chapter 1

Introduction

Multi-rotor Micro Aerial Vehicles (MAV) equipped with onboard sensors have become ideal platforms for autonomous navigation in complex and confined environments. This is due to multiple reasons with the most prominent a few being their ease of manufacturability with only off-the-shelf equipment, and superiority to ground vehicles in terms of their ability to traverse the 3D space with great ease. Multi-rotor MAVs are also much easier to maintain and safer for indoor operation compared to single-rotor helicopters due to their simple design and smaller propeller sizes respectively. Consequently, MAVs are platforms cut out for real-life applications which include exploration [94], inspection [101], [127], [131], mapping [120], interaction with the environment [128], agricultural inspection, pest control [116], search and rescue, and tactical engagement just to name a few.

MAVs, on the other hand, can be exploited in real-life scenarios only when equipped with on-board autonomy primarily due to safety concerns and their unstable dynamics. For instance, it is almost practically impossible to control an MAV without an onboard attitude stabilizer which is the lowest level of autonomy required for human operation. More complicated cases such as infrastructure inspection and air delivery, require more sophisticated components to attain safe autonomy such as full state estimation, mapping of the environment, obstacle avoidance and high-level human-robot interaction.

The full state estimation and environment mapping topics are studied under Simultaneous Localization and Mapping (SLAM) which has been a major problem of interest for the robotics society for more than three decades [25], [40], [41]. Various successful probabilistic methods [22] as well as graph-based solutions [50], [79] have been proposed that brought the literature to a saturation point. These methods temporally integrate state and map information obtained from sensory data over a certain time interval. While a single sensor such as a 2D/3D lidar, a color or RGB-D camera might be sufficient for state estimation and mapping in many scenarios, there are also methods that do sensor fusion for improved robustness and fail-safe operation. On the other hand, path planning and obstacle avoidance are possible only with an accurate and precise SLAM output. Given a dense map, a planner algorithm builds a path assuring a collision free trajectory [86] hence inherently avoiding obstacles. There are also studies such as [102], [103] which propose methods for navigating through narrow passages respecting the nonholonomic dynamics of these platforms.

In this work we focus on the design of state estimation and mapping algorithms using different sensor modalities for axisymmetric tunnel-like environments such as penstocks and corridors. As opposed to existing studies which consider these problems in feature-rich environments, we propose methods which assume a specific geometry for the environment that can be represented parametrically. Furthermore, we study the same problem for different sensor payloads and compare their capabilities and performance. The proposed methods are extensively tested inside four different penstocks. To our knowledge, this is the first robotics field study in the literature that focuses on this specific problem.

1.1 Motivation

A penstock is the water way of a dam that extends from the electric generator turbines located below the river level up to the lake level. Penstocks are usually made out of riveted steel plates or steel-reinforced concrete. These structures have diameters ranging between 5 to 20 meters and lengths between 70 to 250 meters.

Since they are exposed to huge, oscillating loads for long periods, regular inspection and maintenance of penstocks are vital. If crack and rust formations are not treated timely, catastrophic consequences are inevitable such as collapse of the penstock, flood and fire. Current inspection and maintenance practices are carried manually by dam workers either by swinging down from the gate, tethering carts or building scaffolds. However, these conventional methods are dangerous, potentially inaccurate, labor and cost intensive. This work proposes a solution that utilizes MAVs to replace the conventional methods for visual inspection of penstocks.

Penstocks pose unique practical and theoretical challenges encountered very rarely in a typical robotics field application. Since penstocks are steel structures, neither a Global Positioning System (GPS) nor a compass can be used for state estimation. A penstock has almost no geometric cues except for changes in the bending profile and tunnel diameter. Lidars sometimes fail measuring distant points since wet and muddy surfaces behave like a mirror and black tar coating absorbs the laser beam. Also RGB-D sensors completely fail when oriented towards wet surfaces. Furthermore, the visual texture on the tunnel surface is very weak in most parts of the tunnel. Visual cues are usually due to either rivets, welding lines which may be meters apart from each other, rust spots or peeled off protective tar coating. This ironically means that a penstock in good condition has almost no visual texture prohibiting autonomous inspection. Since there is no external illumination inside a tunnel, vision-based algorithms can work only with powerful onboard illumination which complicates the platform design. Besides, experimentation inside a penstock is grueling and arduous for researchers due to continuous water drainage, slippery mud covering the tunnel floor, steep inclination and rust dust. As a result, autonomous navigation of an MAV inside penstocks is, on its own, a valuable field robotics study with both unique theoretical and practical challenges. In this work, we seek for solutions to these challenges.

1.2 Problem Statement

In a SLAM framework, front-end is responsible for extraction of salient features from the sensor data such as image or point cloud features, and association of these features accross frames. Typical indoor and outdoor settings for robotics applications are rich in geometric and visual cues such as an office or an urban area. Hence, most SLAM front-end methods (*i.e.* feature extraction and matching) are designed with this presumption. To illustrate, range-based methods require geometric cues such as non-degenerate set of planar surfaces [90], [95], [136], edges and corners [134] for data association. This is then used for point cloud registration and relative pose estimation. Similarly, vision-based methods either use salient image features [59], [121] or pixel intensity values [107] for data association. Camera displacement is then estimated as the relative pose which minimizes either the total projection error or intensity difference respectively. Benchmarking studies such as [88], [93], [96] show state of the art in accuracy and robustness *when* data association *is* possible.

A typical penstock neither has geometric cues, nor offers imagery with sufficient texture which are required by most front-end methods. To illustrate, point cloud data collected with a 3D lidar consists of indiscriminate partial cylindrical patches from which no edge or corner can be extracted. Furthermore, consecutive point cloud data often do not overlap sufficiently when the robot pitches more than several degrees. Lastly, the long tunnel geometry prohibits measuring position along the tunnel axis. Although the geometry of a penstock does not pose any problems for a vision-based solution, lack of sufficient illumination or features may become prohibitive. Also, in most penstocks there is always mist due to continuous water drainage, rust and tar dust as well. The propeller downwash kicks up mist and dust which often completely blocks the field of view of onboard cameras for several seconds long sufficient to lose tracking. Due to the reasons only some of which we listed here, existing SLAM methods cannot be *directly* applied in this scenario. This work aims to fill this gap in the literature with methods *crafted* for this specific field robotics scenario.

1.3 Research Problems

1.3.1 Range-Based SLAM

Many variants of range-based SLAM solutions rely on the presence of geometric cues for feature extraction and matching [13], [134], while others use point-to-(point/line/plane) (P2X) approach for data association [30], [64], [105]. These two types of methods are often used in combination where a feature-based method finds an initial alignment, and a P2X-based method refines the alignment. This scheme speeds up the convergence rate and also proves successful on fast moving platforms [64] and offers a solution to the kidnapped-robot case [69] as well. In a GPS-denied, dark tunnel environment the wisest choice of sensor is a lidar. Although an RGB-D sensor provides data in a similar format, due to its working principle, RGB-D cameras perform very poorly inside these environments. Obviously, a feature-based method would quickly diverge after finding no or too few, yet low quality, features in a long axisymmetric tube. On the other, the latter approach usually converges quickly when initialized close to the actual robot pose, but it gets trapped in local minima due to the non-convex nature of orientation estimation. In this work, we evaluate these two types of methods in a tunnel setting. We also approach this problem from a different perspective where we parametrically model the environment.

1.3.2 Sensor Fusion

Sensors provide useful information only when they are used within their design limitations. Therefore, a sensor might dysfunction when one or more of the working conditions are not satisfied. For example, in a setting with insufficient illumination, repetitive texture or when a trajectory that does not offer sufficient parallax is followed, a single camera provides almost no 3D pose information. Similarly, a lidar in an environment covered with light absorbing or reflective materials would fail to take measurement at all, or data association on the point cloud might be impossible in case there is too much clutter. In case of an IMU, very fast motions for long periods of time might render it useless due to sensor saturation or lack of gravity correction. A solution for such cases is to use a heterogeneous suit of exteroceptive sensing modalities. Sensor fusion algorithms can be used to incorporate information from each sensor to increase the robustness of the estimator and also save the platform from failing. In our case, none of the above three types of sensor suffice for 6 DoF state estimation. This is because an MAV equipped with only a range sensor cannot localize itself along the long, axisymmetric tunnel axis. Secondly, onboard cameras are unreliable due to weak illumination and texture on the tunnel walls. However, full state estimation can be achieved with the fusion of these sensors.

1.3.3 Tunnel Mapping

Mobile robotics applications that use *relative* measurement sensors such as lidars and cameras require a map as a reference for global localization. Information from these sensors is aggregated on a map in various formats such as a point cloud [43], [84], a set of keyframes with image features [59], [107], [121] or a textured mesh. The map can then be used for path planning and obstacle avoidance, 3D reconstruction of real sites such as mines and ship hulls. The latter purpose is especially intriguing to us since the labor intensive and dangerous task of visual inspection of a penstock can be performed with an autonomous MAV. The maintenance engineer can simply use either the 3D virtual reconstruction of the tunnel or the imagery collected with onboard cameras to localize regions that require maintenance. Also either a parametric or point cloud representation of the tunnel can be used to avoid obstacles such as human operators, scaffolding or tunnel walls.

Chapter 2

Related Work

2.1 Inspection Robotics

After more than a decade of research on MAVs, they have become an important robotic platform employed in many real-life applications some of which are infrastructure inspection, air delivery, photography, search and rescue. Among many interesting application areas is ship vessel inspection as in [110]. The authors autonomously fly an MAV equipped with an IMU, downward facing camera and a 2D lidar inside and outside of a ship hull to find defects in the metal structure such as peeled off coating, corrosion and cracks. The onboard state estimator either uses the lidar or the camera exclusively while moving horizontally and vertically respectively. Hence [110] switches between two types of estimators rather than implementing a sensor fusion algorithm. The additional two cameras are used to collect imagery during the inspection process.

In an industrial boiler inspection scenario, [87], [101] use a quadrotor equipped with an IMU and a stereo rig tightly coupled to estimate 6 DoF robot state. Equipped with onboard power-LEDs, the platform does not require any external illumination for the onboard cameras to work. The researchers adopt an existing method for state estimation details to which is left to the original paper. The robot is deliberately commanded to fly close to the walls so as to sufficiently illuminate the FOV of the stereo rig. If the robot moves away from the walls to a distance where the onboard LEDs cannot sufficiently illuminate the camera FOVs, it can easily diverge and lose control.

In a similar study, the authors of [117] inspect mine shafts with a manually controlled MAV. This work does not consider the online state estimation and mapping problems, but rather focuses on evaluating the platform under the challenging hot, wet and dusty mine conditions as well as strong vertical air flows along mine shafts. The 2D lidar and stereo image data is then post-processed to reconstruct the 3D model of mine shafts. However, this interesting and challenging MAV application requires an expert pilot and clear view of the platform throughout the flight.

[115] also considers mapping of mines. This study uses a pair of rotating 2D lidars and an IMU mounted on a truck that drives through a 17 km long mine to collect data. The conventional surveying methods can be replaced by this much faster method regardless of how long the mine corridors are. In their results, the authors claim that the resultant map offers accuracy better than that of a lidar. This was possible through using continuous-time non-rigid registration, scalable place recognition, and robust pose graph optimization. The total time for 3D offline reconstruction takes only half the time for data collect. This is perhaps the closest study to ours. But in this case, the mine walls have enough geometric texture using which motion along the shaft axis can be inferred. However, this is not the case inside a steel or concrete penstock.

In studies similar to ours [77], [78], [118], Hansen *et al.* uses a small wheeled robot that can fit into 400 mm diameter steel natural gas pipes for inspection purposes. This robot performs visual SLAM using an onboard fisheye camera and sparse structured light. While the robot generates a sparse map for localization, it also constructs a textured 3D mesh for offline visual inspection. The authors use prior knowledge about the geometry of the pipes to increase reconstruction quality as we do in penstock inspection. It may be argued that a similar approach could be used inside a penstock. However, ground vehicles such as proposed in [118] cannot attain sufficient traction on the slippery walls along the inclination.

2.2 Simultaneous Localization and Mapping

Mobile robotic autonomy is possible only if the robot can infer its pose using onboard sensors only. For this, a robot requires a map of its surroundings as well as robust algorithms to localize itself within that map. However, maps such as building blueprints or 3D meshes are very rarely available. Hence, in order to attain a higher level autonomy and independence from human intervention, the map of the environment has to be constructed by the robot. Simultaneous Localization and Mapping, *SLAM*, is a topic in robotics that studies these two important and crucial requirements of mobile robotic autonomy [39]–[41].

The SLAM topic has many theoretical and practical challenges making it an attractive field to researchers for the past three decades. In an early study, Thrun [26] lists the key challenges three of which are sensory noise, high dimensionality (robot and map states), and data correspondence (association). These three challenges shaped the direction the literature pursued. For example, the sensory noise as well as pose and map uncertainties are almost always modeled as Gaussian although in reality this rarely holds. This choice is due to the fact that convolution of two Gaussian distributions gives another Gaussian which is fundamental in the derivation of the renowned Kalman Filter (KF) [1], [39].

On the other hand, there are many systems with multi-modal, widely spread uncertainty characteristics that cannot be modeled as a Gaussian. The nonparametric particle filter-based (PF) approach and its variants, also known as Monte Carlo methods [18], [25], [61] provide approximate representations of arbitrary distributions. Although PF-based methods are more powerful compared to the parametric KF-based approaches in this respect, they suffer from the *curse of dimensionality* [39]. As the dimension of the state space increases, the number of particles required to represent the uncertainty increases dramatically making PF computationally intractable. The Rao-Blackwellized particle filter (RBPF) [19], [56] remedies this shortcoming of vanilla PF by splitting the state space into parts. A small subset of state parameters are tracked by a PF while the remaining is filtered using a suitable KF variant for each particle. Depending on the choice of state partitioning, this significantly reduces the time complexity.

Until recently, KF and its variants have been the *de facto* tool for inference. However, as the dimension of the pose and map increases, KF becomes intractable for real-time onboard computation due to a matrix inversion in KF. Some researchers attempted to surmount this bottleneck through splitting the map into parts [36], [54], [66] or using matrix-inversion-free information filter [35], [63]. These elegant approaches however do not offer a solution to the quadratic growth in time and space complexity with the number of landmarks in the map. Eventually, the *filtering approach* started to be questioned [50], [92]. This is because a KF relates every state entity to all others (the covariance matrix) and this is really not necessary. [29] was one of the earliest attempts to remove *weak connections* between state entities. Using junction trees, [29] is able to reduce the complexity to linear. This approach periodically *thins* the tree to maintain the filter tractable. Studies such as [37], [45] reduced the time complexity using similar approaches.

An intuitive way to formulate the SLAM problem is to build a graph with its nodes being the robot poses at different times and the edges being constraints between the poses [70]. Either odometry or landmark observations are used to define these constraints. The configuration of robot poses which satisfies the constraints gives the robot trajectory. This can then be used to reconstruct the map by aggregating sensory measurement at their corresponding robot poses [50] after optimization.

[50] is one of the earliest attempts to formulate SLAM as a graph optimization. This method is several folds faster than filtering methods such as Extended Kalman Filter (EKF) as claimed by the authors. Instead of finding a most likely robot trajectory at once, [50] uses Stochastic Gradient Descent (SGD) to tackle the problem in real-time at the cost accuracy. Just a year later, Grisetti *et al.* published their solver, TORO [57], which improves over the work of Olson [50]. Instead of using a graph as the data structure, TORO uses a spanning tree of this graph. This way, upon a loop closure the optimizer updates only the landmarks within the loop reducing the time complexity drastically. It also distributes the error onto all the affected nodes using the slerp algorithm [8] which results in smoother trajectories.

Further improvements were obtained by [71] using hierarchical pose-graphs. This work is motivated by the idea that for loop closure, the SLAM front-end should not need to search for correspondence over all other poses. Their hierarchical map groups poses and landmarks into chunks based on a distance metric to reduce the search space for loop closure. Furthermore, only higher level nodes of the graph are updated as long as there is no inconsistency in the lower-level nodes. Grisetti *et al.* later came up with an improved implementation named g^2o [79]. As opposed to their earlier work [57], g^2o does not assume the uncertainties to be spherical and can also handle ill-conditioned measurement covariances. Similar formulations are presented by Kaess *et al.* dubbed as Square-Root SAM [44], iSAM [65] and iSAM2 [91].

In this section we reviewed some of the important studies which consider the *back-end* of the SLAM problem. The primary concern of these methods are handling uncertainties and improving time complexity which are the first two key challenges as listed by Thrun in [26]. In the following sections, we will consider the SLAM *front-end*, *i.e.* the data association problem, which is the third item in this list.

2.3 Range-Based Methods

Lidars and RGB-D cameras are the most widely used range sensors in robotics applications. These sensors provide relative pose information with respect to previous range readings and can also be used for global localization if a map is available.

In an early work, Lu *et al.* [13] uses 2D range measurements to obtain optimal pose estimates. The range readings are aligned using either point-to-point (P2P) or line-to-line (L2L) correspondences. A network of relations (*i.e. graph*) is constructed from wheel and range odometry with connection strengths defined using uncertainties. In this respect, [13] is one of the earliest studies that uses pose-graphs.

The family of point cloud registration algorithms is usually referred to as Iterative Closest Point (ICP). Many variants of ICP algorithms have been proposed with different metrics for data association, heuristics to improve convergence characteristics and reduce computational requirements. A good comparative study on efficient derivatives of ICP algorithm has been presented by Rusinkiewicz *et al.* [23]. A recent and very detailed survey on ICP algorithms and point cloud registration in general is also presented by Pomerleau *et al.* [122]. In an outdoor SLAM application using a 3D lidar mounted on an autonomous ground vehicle, [43] uses an ICP variant that associates a point to its closest model point. In this work, alignment is obtain using a Levenberg-Marquardt non-linear optimization. The odometry results at each step is tested against a classifier trained with manually labeled data.

Censi proposes a method in [64] for 2D scan alignment based on a point-to-line (P2L) metric with a closed-form solution. Although this method is not robust to large initial relative rotations, its convergence rate is much better than P2P metric otherwise. This is basically because the P2L metric quadratically reduces the error and also point-line correspondences usually do not change after an iteration reducing time required for correspondence search. On two other papers [52], [53], Censi proves uncertainty bounds of an ICP algorithm and provides a covariance estimate for a planar ICP.

A completely different data association scheme from the P2X heuristic is proposed by Magnusson *et al.* in [60]. In this work, the authors use the Normal Distribution Transform (NDT) which transforms the raw point cloud data into a collection of 3D normal distributions. To each set of points that fits into a voxel of a 3D occupancy grip is fitted a 3D normal distribution. The minimization function is then written using the Mahalanobis distance between two point clouds. The authors test their method in a challenging mine survey application.

Olson, in his work [69], presents a probabilistically-motivated 2D scan-matching algorithm. This method is built upon the idea that two scans are aligned the best when their cross-correlation is maximum. He claims that although his method is computationally more demanding, the results are better in quality. Using a multi-resolution 3D grid (two translations and a rotation), this method can do global localization.

Until MAVs became more popular, SLAM experiments used to be carried on only ground vehicles. [68] is one of the early studies which applies the existing mapping and localization algorithms on a small MAV indoors. This work proposes a navigation system consisting of a quadcopter equipped with an IMU, a 2D lidar with a mirror setup to deflect some of the lidar rays to the floor for altitude measurement and a Gumstick embedded CPU. Shen *et al.* [84] uses a similar setup for mapping a multi-floor building. Despite low computational budget, this system was capable of running all computation onboard including localization and mapping, planning and controller.

There are several other studies which focus on utilizing range sensor for MAV control and navigation. [86] uses an MAV with an onboard RGB-D sensor. A vision-based odometry is used for localization and the depth data is used to construct a 3D occupancy grid map which is used for path planning. Due to low computational power, mapping is done on a remote workstation. With a similar configuration [120] flies multiple MAVs for collaborative mapping. This work simultaneously localizes multiple MAVs using a vision-based method inferring the scale factor from the depth data. The swarm of MAVs collaboratively builds a dense map of the environment. Finally, Zhang *et al.* [133], [134] uses an onboard 3D lidar for mapping and navigation with a large MAV outdoors. The resultant point cloud map of this algorithm is very detailed allowing for safe navigation and accurate path planning.

2.4 Vision-Based Methods

Due to their high cost, weight and narrow FOV (*e.g.* 2D lidars) lidars are often not considered in MAV applications. Especially data association on lidar data collected from unstructured, cluttered environments is a difficult problem. On the other hand, cameras are cheap, light-weight and provide rich information about the environment. Due many factors these being only a few, cameras are one of the mostly studied sensors for localization and mapping.

There are two approaches to data association between images : feature-based and intensity-based. A feature-based method requires extraction and matching of features which may introduce errors due to feature position inaccuracies [62]. Furthermore, this causes extra burden on the CPU which is an undesired effect especially for small MAVs. On the other hand, the latter family of methods use the pixel intensities and do normalized cross-correlation search to associate pixels across frames [20], [55]. Thus, Visual Odometry (VO) methods using this approach is often called *direct methods* [106], [107], [129].

There is huge literature on feature extraction and matching. Some of the most widely used feature extraction methods are Harris [9], FAST [51], [75], SIFT [31], SURF [42] to name a few. The latter two methods also provide feature descriptors for feature matching across frames or object/scene recognition. For example, the authors of [38] use SIFT features for localization and sparse mapping on a ground vehicle in an office setting. Tuytelaars *et al.* [62] provides a very detailed study on local invariant feature detectors. This work discusses the qualities of a good feature detector and reviews prominent studies of its time.

Vision-based methods are applicable wherever the robot can extract salient features. Especially for outdoor settings, Visual SLAM (VSLAM) offers a promising alternative to GPS-based navigation systems such as in [85]. The authors claim their work does not suffer from drift, estimates 6 DoF MAV state onboard and to be the first MAV VSLAM application. Shen [113] studies VSLAM on an MAV with transitions between indoor and outdoor settings. This is especially a difficult task due to saturation and slow exposure adjustment of the cameras. In a setting close to ours, Alismail *et al.* [129] uses binary descriptors for localization and mapping inside a very poorly illuminated mine.

Chapter 3

Preliminaries

3.1 Experimentation Environment

Methodologies proposed in this work focus on state estimation and mapping inside axisymmetric, long, tunnel-like environments. In particular, we are interested in flying MAVs autonomously inside penstocks. We also tested these methods inside long corridors in university buildings during the software development cycle.

Penstocks are either steel or concrete tubes that conveys water from a lake to the electric generator turbines at the bottom of a dam (Fig. 3.1). Penstocks are almost perfectly

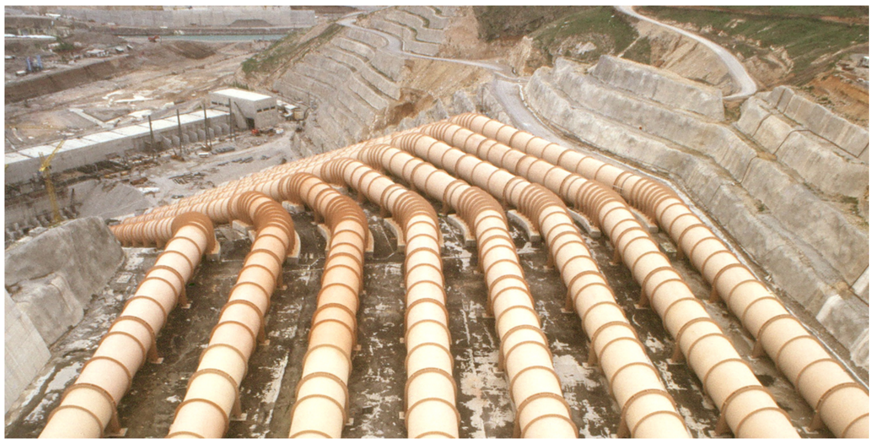


Figure 3.1: An array of steel penstocks.

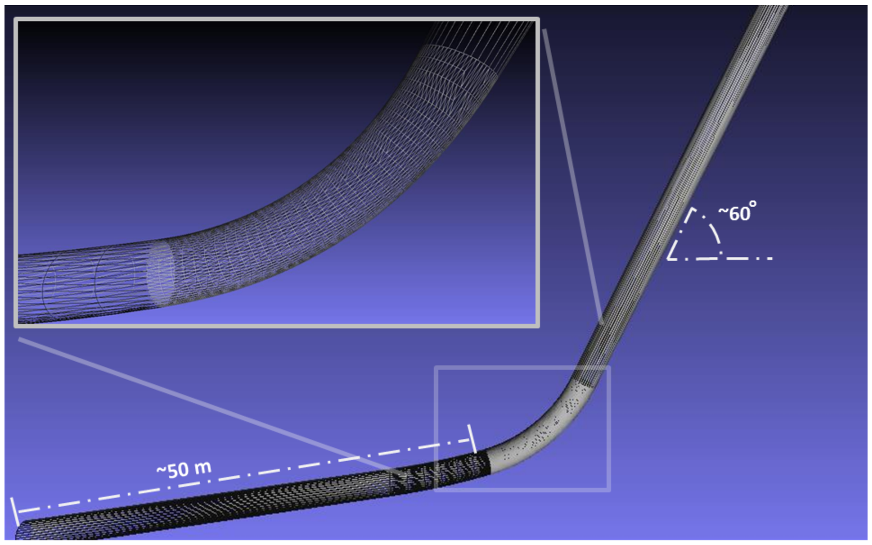


Figure 3.2: A partial CAD model of a penstock in Glen Canyon Dam, AZ. This penstock is one of the largest in the US with a total length of more than 250 meters and a steep inclination of ~ 60 degrees.

cylindrical in cross-section, and have multiple sections with different orientations. A sample CAD model of a penstock in Glen Canyon Dam, AZ is shown in Fig. 3.2. Their diameters range between 5 to 20 meters depending on the size of the dam. In a typical penstock, the first section is the scroll cage that wraps around the giant turbine of diameter ~ 5 meters (Fig. 3.3). Following that is usually a horizontal section which might bend to the sides depending on the shape and width of the river bed. The next section has an inclination that ranges from gentle slopes to close-to-vertical as in Glen Canyon Dam, AZ. Depending on the depth of the dam wall, a penstock may extend further after the inclined section. The end of a penstock is blocked by a gate. The shortest penstock we experimented is ~ 70 meters while the longest is longer than 200 meters.

Penstocks are pitch-dark, uniaxial and axisymmetric tunnels with no geometric features except for changes in the bending profile and diameter (Fig. 3.5-3.6). In addition to this, close to the gate, tunnel cross-section becomes rectangular offering some geometric cues. Besides being pitch-dark, penstock walls do not offer visual texture due to uniform protective tar coating. However, penstocks with peeled off coating offers some visual texture such as shown in Fig. 3.4. The walls of steel penstocks are cylindrically bent steel plates riveted along a seam line offering some visual texture too. Lastly, in concrete penstocks, cracks and

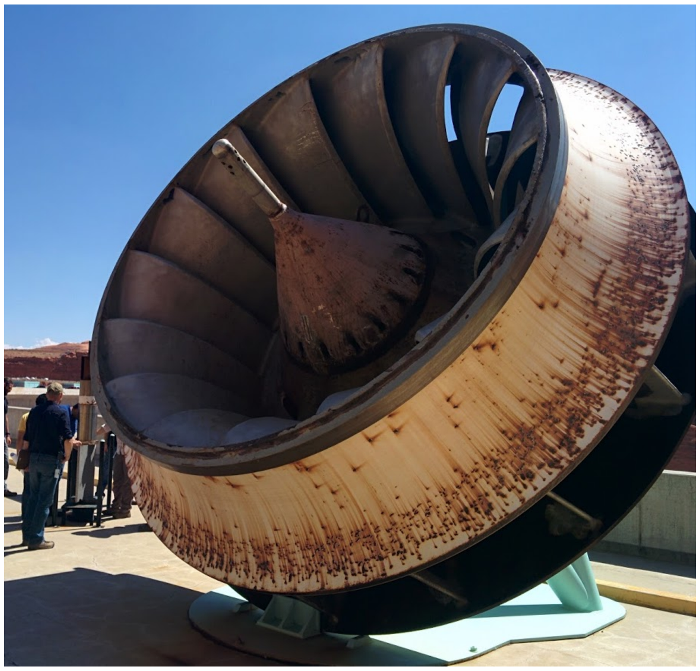


Figure 3.3: A turbine in Glen Canyon Dam, AZ exhibited for visitors. Its size can be compared with a visitor standing next to it.

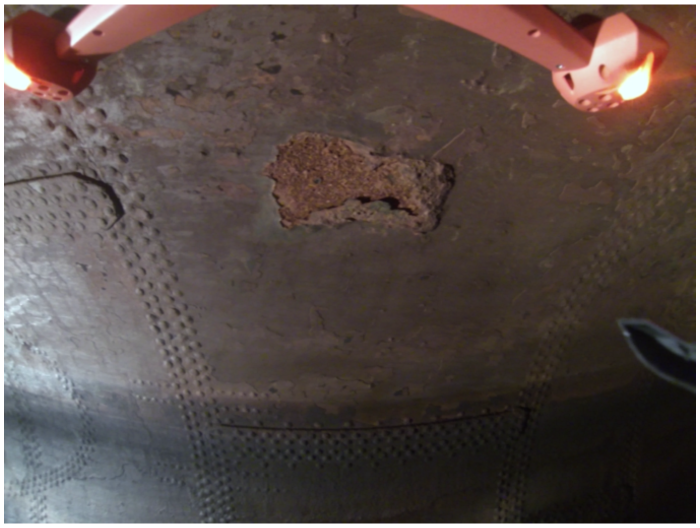


Figure 3.4: An image from inside a penstock in Allatoona Dam, GA. This image was taken from a quadcopter flown manually close to a patch that needs maintenance.

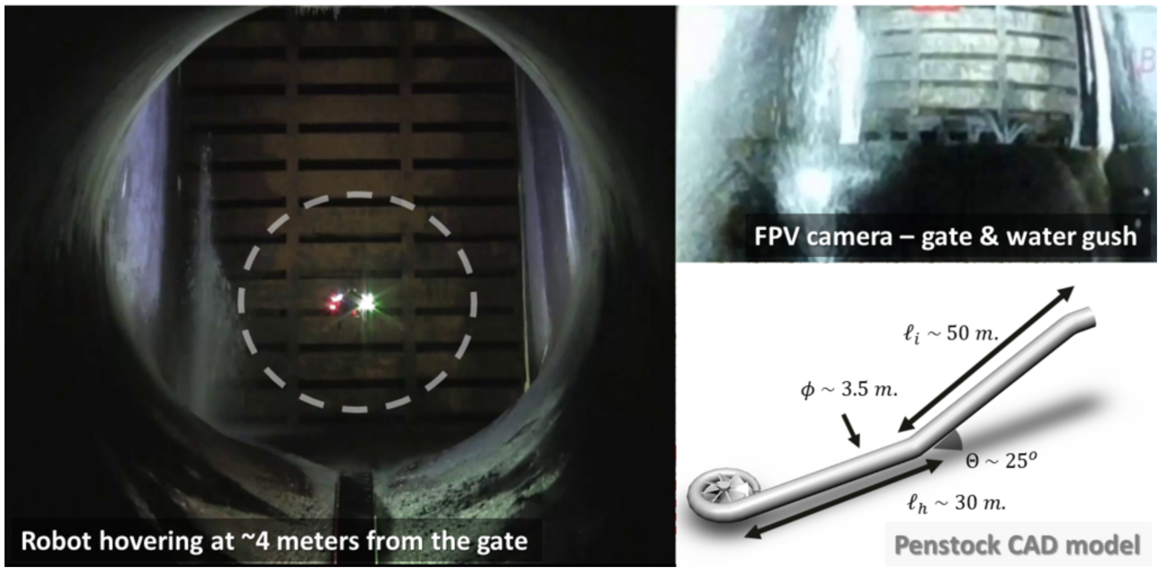


Figure 3.5: (Left) Custom-designed hex-rotor platform hovering $\sim$ 4 meters from the gate in shared-control mode. (Top-right) FPV camera snapshot showing the gate and the water gush. (Bottom-right) CAD model of a penstock at Center Hill Dam, TN.

limestone formation provide richer visual cues compared a steel tunnel.

The other three types of sensors widely used in MAV automation are magnetometer, GPS and barometers. Due to the obvious reason that penstocks are made of steel or steel-reinforced concrete, we never consider magnetometer as a alternative sensor. The considerations apply to GPS. Finally, due to the strong vortices formed by the propellers, barometer measurements are prohibitively noisy.

In conclusion, the characteristics of penstocks and their effects of widely used sensor modalities pose a number challenges in state estimation and mapping. It is theoretically impossible do localization using range sensors along the tunnel axis since either the length of the tunnel is longer than the range of the sensor or the ends of the tunnel are outside the FOV of the lidar such as in the case while the robot is flying along the inclination. A sample data is shown in Fig. 3.7. Furthermore, due to the uniaxial and axisymmetric cross-section, roll and pitch angles cannot be estimated using only relative measurement sensors (*i.e.* lidars and cameras). Lastly, inside steel penstocks, neither GPS, compass nor barometer works properly.

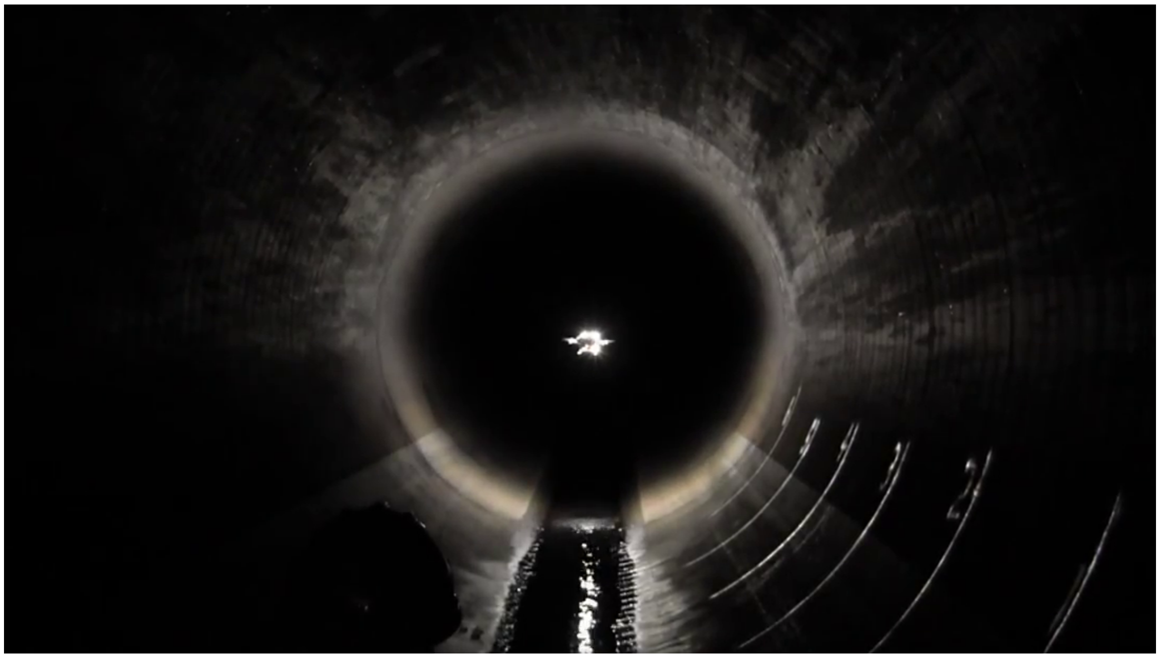


Figure 3.6: A photo taken from inside a penstock at Glen Canyon Dam, AZ while the KHex platform is flying autonomously. The robot is tested for visual odometry with onboard LEDs as the only source of illumination.

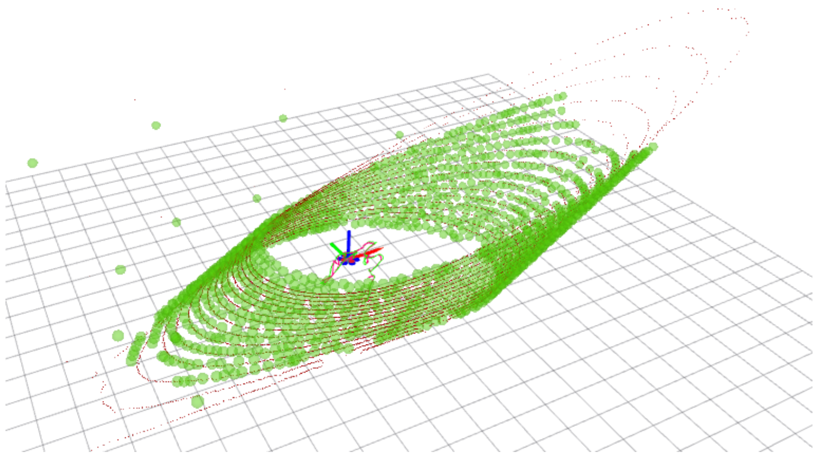


Figure 3.7: A snapshot from RViz captured when the robot is flying along the inclined section of a penstock at Center Hill Dam, TN. The shape of the point cloud data is almost the same throughout the inclined section hence providing no information about robot's position along the tunnel axis.

In the past five years, we visited five different sites to test and improve various approaches to the state estimation and mapping problems. These site are :

- Carters Dam, GA
- Allatoona Dam, GA
- Glen Canyon Dam, AZ
- Center Hill Dam, TN
- Walter Dam, PA

In the subsequent chapters, we will explain experiment results on these sites.

3.2 Experiment Platforms

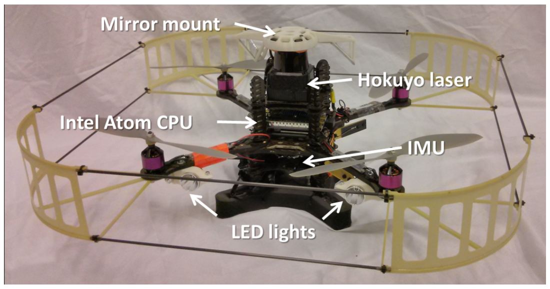
3.2.1 Platform Requirements

While the size of an MAV almost proportionally affects the maximum payload, the same factor also brings about safety concerns due to the platform weight and larger propeller sizes. For example, a platform with larger propellers can carry more sensory payload such as cameras and lidars. However, an MAV with large effective diameter is undesired when flying inside a tunnel with a relatively smaller diameter since the propellers may clip the walls. Also, when flying close to the walls, external disturbance due to the vortices generated by the propeller downwash disturbs the stability. The flight time should be long enough to traverse a penstock of length ~ 300 meters one-way with steep inclination such as in Glen Canyon Dam, AZ. This requires a larger battery hence a larger platform. A design that allows for easy maintenance is also desirable since the adverse conditions inside a penstock may require replacement of some mechanical and electronic components after an experiment. Lastly, the choice of minimal onboard sensor suit sufficient to autonomously fly an MAV through a pitch dark, long and featureless tunnel is maybe the most important aspect of platform design. The sensor pack may included one or multiple range sensors and cameras depending on the specific SLAM formulation. Hence, finding the best design is a constrained optimization problem with many equally admissible solutions.

In field robotics, the platform choice, its capabilities and sensory payload play a crucial role. Only very recently professional commercial MAVs such as Ascending Technologies' Pelican and DJI's Flame Wheel series became available to the researchers. With the emerge of alternative commercial MAVs, we updated our platform design over the years. We also redesigned our MAVs as new sensors such as Velodyne's 3D Lidar became available to match application requirements. In this section, we go through three different platforms that we used over the past five years.

There are a number of MAV design parameters which are propeller count, size and weight, ease of maintenance and stock spare part availability, battery type, sensor pack such as IMU, cameras and lidars. Unfortunately, there is no simple way to determine the best design choice since these parameters are coupled with each other. For example, with the number of propellers, the maximum thrust also increases. However, this does not necessarily increase the thrust-to-weight ratio since motors and Electric Speed Controllers (ESC) constitute a significant percentage of the total platform weight. Also, penstocks are confined spaces making human safety a bigger concern. In case of a crash, a heavy platform may cause severe damage to the researchers since there is not plenty of space to escape from the robot. On the other hand, a small platform cannot carry the sensor payload required for autonomy and visual inspection.

Flight time is another important parameter that needs to be maximized while observing the other constraints. This is important because, while a typical, medium-sized penstock is 100 meters long, larger dams have penstocks which are $\sim$ 300 meters long with steeper inclinations. An MAV should be able to traverse such challenging penstocks before the battery voltage drains below a certain threshold. Otherwise, the MAV would lose attitude since ESCs would saturate at lower input voltages. Finally, we also have to consider the choice of peripherals such as onboard illumination, first person view video downstream which



(a) Ascending Technologies' Pelican equipped with an Intel Atom processor, 2D Hokuyo lidar, an IMU and two LED lights. A 3D printed mirror mounted on top of the lidar redirects some of the laser beams upward and downward.

(b) Our Pelican platform flying inside a penstock in Allatoona Dam, GA. In this photo, the robot is also collecting imagery with a uEye camera mounted on top of the lidar.

Figure 3.8: Our Pelican platform from different perspectives.

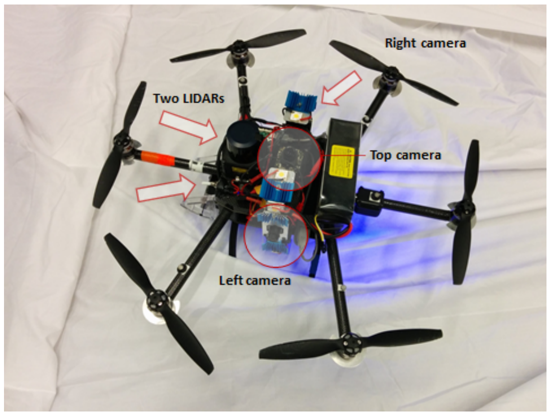
further complicates the platform design.

3.2.2 Pelican

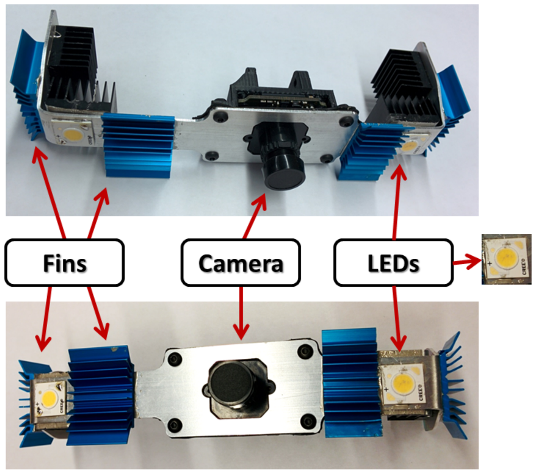
In the earlier phases of experimentation, we used Ascending Technologies' Pelican platform. The frame is made of carbon fiber which reduces the total weight of the platform significantly compared to aluminum or plastic arms. This robot is equipped with a low power 1.6 GHz Intel Atom processor. The major sensors we used for estimation are an IMU and a Hokuyo 2D lidar (Fig. 3.8). The autopilot runs a proprietary software provided by the manufacturer. We retrofitted the robot with a 3D printed mirror setup to redirect some of the lidar beams to the floor and ceiling for altitude measurement. The weight of the platform was 1.8 kg. and it can fly about 10 minutes with a 3 cell Li-Po battery.

3.2.3 KHex

The second platform we used is *KHex* designed by KMel Robotics. KHex can fly approximately 8 minutes with a four-cell 4500 mAh Li-Po battery and a total payload of 2.6 kilograms. Fig. 3.9a shows KHex platform equipped with an Intel i7 board, two Hokuyo UST20-LX lidars and four XGA resolution grayscale BlueFox cameras. KHex is equipped with redundant sensors in order to reduce the risk of sensor related failures and collect



(a) The hex-rotor platform equipped with an Intel i7 computer, IMU, two Hokuyo UST20-LX lidars and four Bluefox XGA cameras. This design uses eight 10 W LEDs placed around the cameras to provide onboard illumination. KHex weighs 2.6 kg and can fly about 8 minutes with a four-cell 4500 mAh battery.



(b) The Camera-LED setup. We use eight Cree power-LEDs to provide onboard illumination for imagery collection and visual odometry. Each LED is 10 W and has 5000 K color temperature.

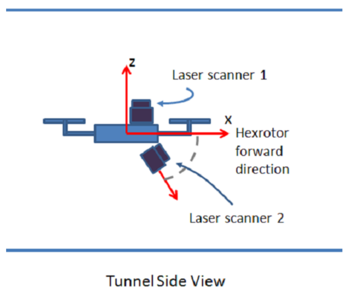
Figure 3.9: KHex platform and onboard illumination setup.

detailed panoramic imagery from inside the penstock.

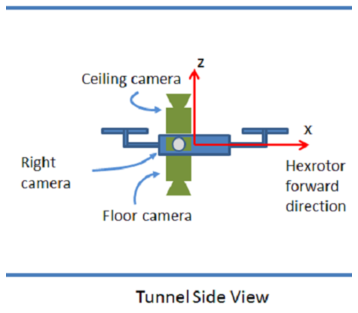
In our previous platform, Pelican, we retrofitted the robot with a mirror setup to redirect some of the lidar rays to the floor and the ceiling to measure the altitude. However puddles, continuous water drainage and wet surfaces often fail the altitude measurements. This problem is solved by mounting a second 2D lidar tilted downwards to measure the elevation as illustrated in Fig. 3.10a.

KHex runs KMel's proprietary onboard attitude estimator. After gravity correction, roll and pitch angle estimates exhibit low drift and noise. The two lidars send data through two separate Ethernet ports at 20 Hz and each has a 270 degrees span. Landing gears and booms only partially occlude the view of the bottom lidar. However this does not affect the overall performance of the onboard estimator since the FOV of the two lidars overlap along the occluded region.

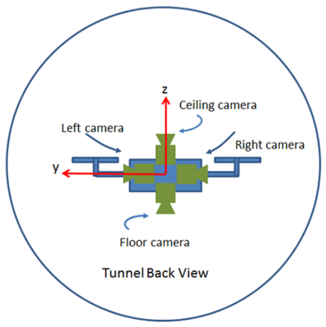
The four cameras are used both for visual odometry (VO) and imagery collection for offline visual inspection. At the time we built this platform, high speed USB-3 global shutter



(a) Two 2D Hokuyo lidars.



(b) Four BlueFox cameras - side view.



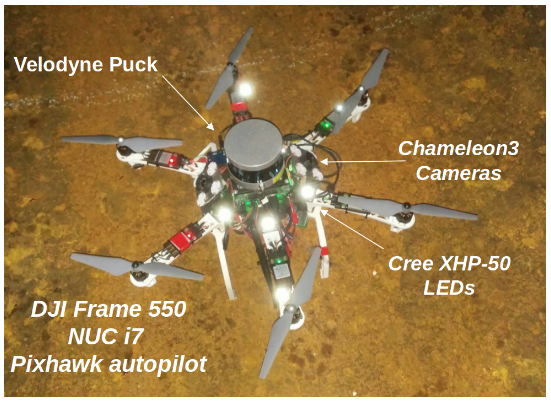
(c) Four BlueFox cameras - back view.

Figure 3.10: KHex platform and drawings showing the sensor placement from different perspectives.

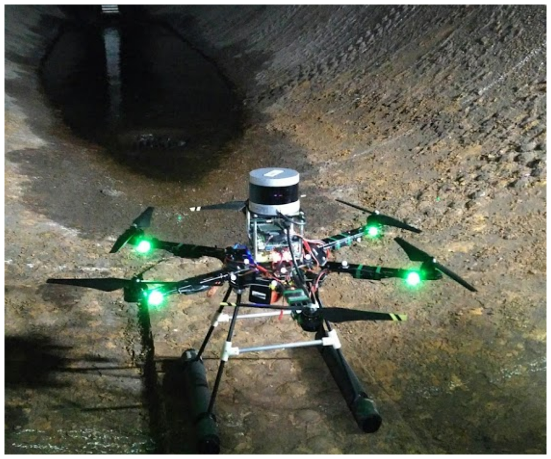
cameras were not available yet. Hence, the bandwidth limit of USB-2 imposes a constraint on the maximum frame rate. Because of this, KHex could not run all cameras at full speed. The schematic showing the sensor placement and the preferred robot orientation during flight is shown in Fig. 3.10. In order to obtain sufficiently bright and textured images for both inspection and VO, we equipped the robot with power LEDs (Fig. 3.9b). This removes the requirement of external illumination and reduces the labor requirement significantly.

3.2.4 DJI Hexrotor

After its release, we replaced the 2D lidars with a Velodyne Puck 3D lidar. In order to carry the extra 800 grams payload, we upgraded our platform to a larger hex-rotor MAV. We built two platforms with DJI Flame Wheel F550 frame with stock arms and extended arms. These two versions use DJI E310 and DJI E600 propulsion systems respectively. Since the former propeller-motor combination (Fig. 3.11a) did not provide sufficient thrust, we continued with the latter, larger platform (Fig. 3.11b). The thrust to weight ratio of this platform is $\sim$2. Its total weight is $\sim$4.5 kg when equipped with a Velodyne Puck LITE, a Pixhawk autopilot, a 5th generation Intel i7 NUC board, a 6S 5000mAh 50C LiPo battery and a custom-designed power distribution board. The robot also carries four Chameleon3 1.3MP USB-3 color cameras and high-power Cree XHP-50 LEDs for onboard illumination and imagery collection (Fig. 3.11). The total flight time is more than 12 minutes depending on



(a) DJI platform with shorter stock arms and E310 propulsion system.



(b) DJI platform with extended arms and E600 propulsion system.

Figure 3.11: Two iterations of the DJI platform during experimentation inside Center Hill Dam, TN.

the aggression level. This platform is larger in diameter than our previous designs primarily due to larger 12″ DJI E600 propellers.

Chapter 4

Localization and Control Using a Single 2D Lidar

In this chapter, we present our first estimator design based on a single 2D lidar and an IMU. Our purpose is to enable a computationally constrained Pelican MAV (Sec. 3.2.2) fly semi-autonomously inside long featureless tunnels. Due to the limited information from a single 2D lidar, we only focus on localization assuming that the map of the tunnel is provided in a parametric form which can be converted into a point cloud or a mesh. The proposed method uses a Rao-Blackwellized Particle Filter (RBPF) with the particles modeling the position uncertainty along the tunnel axis and an Unscented Kalman Filter (UKF) tracking the remaining degrees of freedom. The measurement update is based on an efficient ICP derivative with outlier rejection. Except for when the robot is close to the terminals of the tunnel or there is significant change in the tunnel profile, the position estimate is uncertain along the tunnel axis. Hence, we assume an operator manually controls the robot pitch throughout most of the flight.

Details presented in the chapter are based on our previous study at [111]. To our knowledge, this is the very first study in the robotics literature focusing on localization and autonomous control of an MAV in 3D, featureless tunnel-like environments. We present results from experiments in Carters and Allatoona Dams, GA and university building corridors showing that the proposed method operates robustly in different settings.

4.1 Map and Frame Definitions

Penstocks we experimented inside consist of cylindrical and conical sections with no or slight bends. Hence, each such penstock segment can be approximated by simple parametric mathematical structures. Parameters defined for the two end points of a given section can be used to obtain intermediate points by linear interpolation.

4.1.1 Map as a List of Joints

We define a joint, J, as an aggregate structure consisting of a position, a radius and a flag to determine whether it is blocked or not. This can be written as $J := \{\mathbf{r}, \rho, \beta\}$ where $\mathbf{r} \in \mathbb{R}^3$, $\rho \in \mathbb{R}$ and $\beta \in \{true, false\}$. The last parameter, β, is set true for the end joints of a penstock if it is blocked by a gate. An ordered list of joints approximates a cylindrical tunnel which we represent as $M := \{J_0, J_1, ...J_n\}$. The line segments formed by connecting the positions of adjacent joints form the *center line*, χ, of a tunnel. The circular contour swept along the center line gives the surface of the tunnel which we convert into a mesh or a point cloud depending on the requirements of the process that consumes this structure. Furthermore by simply changing the definition of the contour represented by the joints, one can approximate the map of a building corridor or another axisymmetric tunnel-like structure. Depending on the context, M may refer to either of the list of joints, the point cloud or the mesh obtained from the list of joints. We present a point cloud approximation to a penstock in Fig. 4.1.

We parametrize the center line such that $\chi := \mathbb{R} \rightarrow \mathbb{R}^3$ where $\chi(s) = \chi(0) + \int_0^s \frac{\chi(\sigma)}{d\sigma} d\sigma$. $\chi(s)$ maps a given scalar, s, uniquely to a point along the tunnel axis for $s \in [0, s_{max}]$ where s_{max} is the length of the center line. The inverse function is defined as $\chi^{-1}(\mathbf{r}) = \underset{s}{\operatorname{argmin}} \, ||\chi(s) - \mathbf{r}||_2$ where $||\cdot||_2$ is the L_2 norm. Although χ is a function, its inverse may not always uniquely

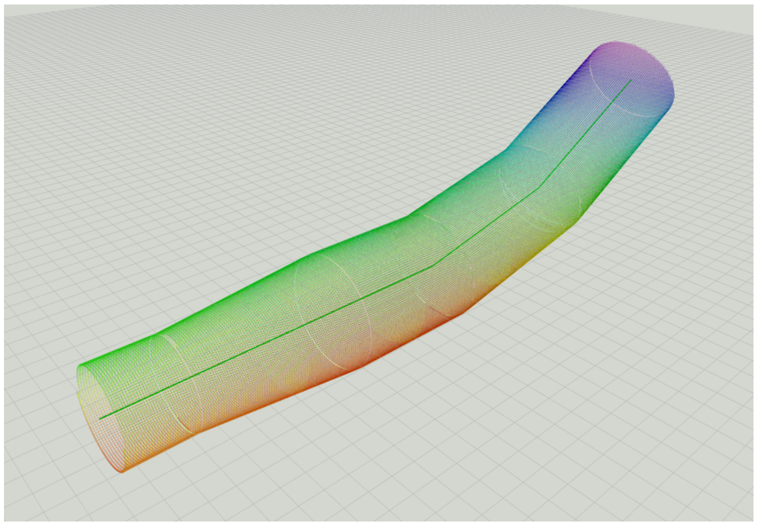


Figure 4.1: A sample tunnel reconstruction from a set of ordered joints. Each joint has a associated 3D position and a radius. The junctions at the terminals are marked as *blocked* to simulate gates or other blockages. The map is represented as a point cloud.

map a given 3D point to a scalar. Depending on the curvature, radius and query point, χ^{-1} may yield multiple solutions. We assume that among the possible solutions one can be unambiguously chosen from the context (*i.e.* robot state).

4.1.2 Reference Frame Definitions

A reference frame is defined by an orthonormal triplet of basis vectors and an origin, *i.e.* $\mathcal{F} := \{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, \mathcal{O}\}$. Using the parameterization explained in the previous section, we can define two local coordinate systems for each point along the tunnel axis. These two frames are depicted in Fig. 4.2. One of these frames is aligned with the center line tangent and the other is with the gravity vector.

We prefer defining multiple local frames to simplify the formulation of the pose estimation and filtering as well as control and path following as will be explained in the subsequent sections. We designate these two frames with $\mathcal{A}(s)$ and $\mathcal{G}(s)$ where the former is used to represent the robot position and the latter for its orientation. Both reference frames are functions of their axial positions (*i.e.* their position along the centerline). The common origin of both reference frames is defined to be at the center line, *i.e.* $\mathcal{A}(s)_{\mathcal{O}} = \mathcal{G}(s)_{\mathcal{O}} = \chi(s)$.

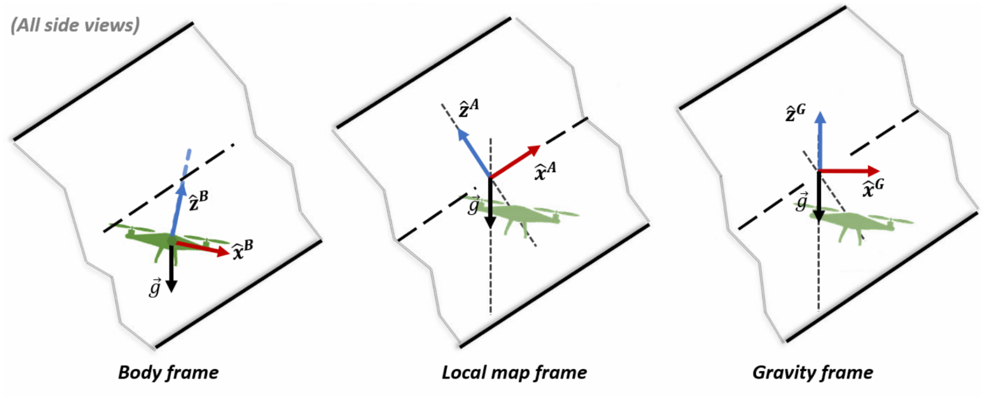


Figure 4.2: An illustration of the three coordinate frames which are the body ($\mathcal{B}$), local map ($\mathcal{A}$) and gravity ($\mathcal{G}$) frames. These figures show the relation between these frames with the robot state is fixed. The difference between the local map and the gravity frames is a rotation around the *common* y axis.

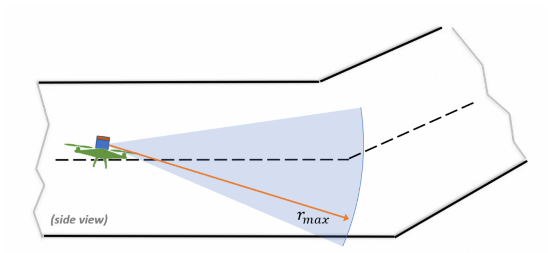
$\mathcal{A}(s)$ is oriented such that it is aligned with the cross-section of the tunnel at the frame origin. This can be attained simply by defining its basis vectors as

$$\begin{aligned}
\mathcal{A}(s)_{\hat{\mathbf{x}}} &\parallel \frac{d\chi(s)}{ds} \\
\mathcal{A}(s)_{\hat{\mathbf{y}}} &\parallel \mathcal{A}(s)_{\hat{\mathbf{x}}} \times \mathbf{g} \\
\mathcal{A}(s)_{\hat{\mathbf{z}}} &\parallel \mathcal{A}(s)_{\hat{\mathbf{x}}} \times \mathcal{A}(s)_{\hat{\mathbf{y}}}
\end{aligned}$$

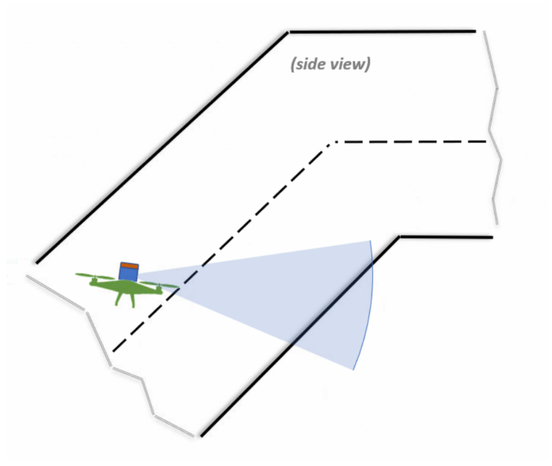
where $\mathbf{g}$ is the gravity vector, each basis vector is normalized and the binary operator $\cdot \parallel \cdot$ implies the two vectors are parallel. Orientation of $\mathcal{G}(s)$ is obtained simply by rotating $\mathcal{A}(s)$ around $\mathcal{A}(s)_{\hat{\mathbf{y}}}$ such that $\mathcal{G}(s)_{\hat{\mathbf{z}}} = -\mathbf{g}$ (Fig. 4.2). The benefits of using these two frames will be explained in the following sections. Lastly, we define the map frame as $\mathcal{M} := \mathcal{G}(s = 0)$. In the following sections we will omit the axial position parameter with reference frames for clarity.

4.2 Filtering-Based Localization

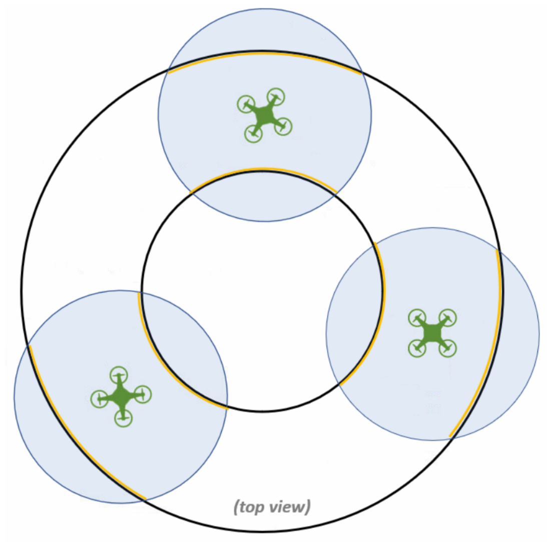
Estimating the robot pose with respect to a global reference frame is impossible since the experiment sites are both GPS and compass-denied. Furthermore estimation using only range sensors with respect to a fixed world frame is also impossible due to the shape of the



(a) An illustration of a robot flying along a horizontal section of a long tunnel. Since the maximum range of the laser scanner is less than the distance to the start of the inclination, the robot is unable to localize itself with respect to a global reference frame using solely the onboard range sensor.



(b) An illustration of a robot flying along an inclined section of a tunnel. Since the vertical field of view of the laser scanner is small, the robot cannot measure where the inclination ends. This prohibits the robot from localizing itself along the tunnel axis using only a range sensor.



(c) An illustration of three robots flying in a hypothetical toric shaped tunnel. Due to the shape of the tunnel, all the robot take the exact same measurements using their onboard range sensors. Hence, the true position of a robot cannot be inferred in such an environment using only range sensors.

Figure 4.3: Three illustrations explaining the three limiting cases where a range-based estimator cannot localize a robot inside a axisymmetric tunnel.

tunnel. This can be understood by observing the following three cases.

1. **Long tunnel** : If the tunnel is longer than the maximum sensing range of the laser scanner, the robot cannot estimate its position along the tunnel axis (Fig. 4.3a)
2. **Inclined tunnel** : This case is similar to the long tunnel case. Furthermore, if the field of view of the range sensor is narrow, the robot cannot measure its distance to where the tunnel bends (Fig. 4.3b)
3. **Torus tunnel** : Even if the field of view and the range of the range sensor are sufficient, the range measurements are not informative since the robot would get the same measurements everywhere in the tunnel (Fig. 4.3c-3.7).

In the following sections we define the robot pose and design the filter based upon these observation.

4.2.1 Robot State

The featureless, long tunnel geometry renders the position along the tunnel axis unobservable to a range sensor as explained in the previous section. For this reason we prefer separating the unobservable and observable states of the robot and also design our estimator and controller accordingly. The robot state is defined as

$$\mathbf{x} := [s, \mathbf{r}_{y,z}, \boldsymbol{\Omega}, \mathbf{v}_{y,z}, \mathbf{b}_{\mathbf{a}}, \mathbf{b}_{\boldsymbol{\omega}}]^\top \tag{4.1}$$

where s is the position along the center line, $\mathbf{r}_{y,z}$ and $\mathbf{v}_{y,z}$ are the y and z coordinates of the position and velocity defined in $\mathcal{A}(s)$, $\boldsymbol{\Omega}$ is the orientation defined in $\mathcal{G}(s)$, $\mathbf{b}_{\mathbf{a}}$ and $\mathbf{b}_{\boldsymbol{\omega}}$ are the accelerometer and gyroscope biases given in the IMU frame, $\mathcal{I}$. $\boldsymbol{\Omega} := \{\phi, \theta, \psi\}$ is the Euler angles in XYZ order which can be written in matrix form as ${}^{\mathcal{G}(s)}\mathbf{R}_{\mathcal{B}(s)} = \mathbf{R}_z(\mathbf{x}_\psi)\mathbf{R}_y(\mathbf{x}_\theta)\mathbf{R}_x(\mathbf{x}_\phi)$ where $\mathbf{x}_\phi, \mathbf{x}_\theta, \mathbf{x}_\psi$ are the roll, pitch and yaw angles respectively, and $\mathbf{R}_i(\gamma)$ is the rotation matrix around axis i by γ radians. All degrees of freedom except for s is observable to the range sensor and the IMU. We will refer to s as the *axial position.*

Lastly, we assume that the IMU and body frames coincide and will use interchangeably, *i.e.* $\mathcal{I} \equiv \mathcal{B}$.

A Discussion on Reference Frame Choices

Defining the robot position, $\mathbf{x}_{y,z}$ and velocity $\mathbf{x}_{\dot{y},\dot{z}}$ in frame $\mathcal{A}(s)$ has multiple benefits. The position of the robot in $\mathcal{M}$ for a given state can be written as

$$\mathbf{x}_{\mathbf{r}}^{\mathcal{M}} = \chi(s) + {}^{\mathcal{M}}\mathbf{R}_{\mathcal{A}(s)} \begin{bmatrix} 0 \\ \mathbf{x}_{y,z} \end{bmatrix}. \tag{4.2}$$

Here, with the abuse of notation, $\mathbf{x}_{\mathbf{r}} \in \mathbb{R}^3$ refers to the complete position of the robot. The robot position as defined in the original state space can be recovered by first projecting $\mathbf{x}_{\mathbf{r}}^{\mathcal{M}}$ onto the tunnel center line as

$$s = \chi^{-1}(\mathbf{x}_{\mathbf{r}}^{\mathcal{M}}). \tag{4.3}$$

We assume that either there is a single solution or we can infer the right one from the context, *i.e.* the most recent robot state. Then the remaining unknown position variables can be found as

$$\mathbf{x}_{\mathbf{r}}^{\mathcal{A}(s)} = {}^{\mathcal{A}(s)}\mathbf{R}_{\mathcal{M}} \left(\mathbf{x}_{\mathbf{r}}^{\mathcal{M}} - \chi(s)\right). \tag{4.4}$$

Due to how χ^{-1} is defined, $\mathbf{x}_{x}^{\mathcal{A}(s)} = 0$ always holds. It should be noted that any reference frame that has one of its basis vectors tangent to the center line supports this property up to a permutation.

Let's consider the same operations assuming $\mathcal{G}(s)$ is used for the robot position. Then the transformation

$$\mathbf{x}_{\mathbf{r}}^{\mathcal{G}(s)} = {}^{\mathcal{G}(s)}\mathbf{R}_{\mathcal{M}} \left(\mathbf{x}_{\mathbf{r}}^{\mathcal{M}(s)} - \chi(s)\right) \tag{4.5}$$

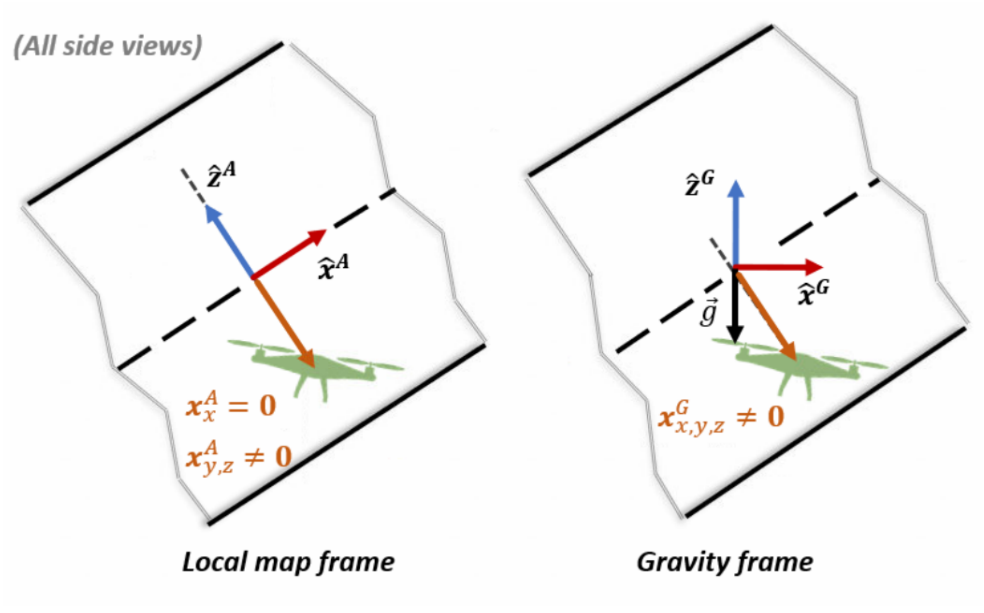


Figure 4.4: This figure illustrates the local and gravity frames along with the robot position written in these frames. Note that the origin of a frame is the point closest to the robot along the center line. $\mathbf{x}_x = 0$ always holds in the local frame case due to how the frame origin is chosen. This equality holds for a gravity frame only if the center line is horizontal.

would result in $\mathbf{x}_x^{\mathcal{G}(s)} \neq 0$ *iff* ${}^{\mathcal{G}(s)}\mathbf{R}_{\mathcal{A}(s)} \neq \mathbf{I}$. The given inequality holds when the tunnel is inclined. This case is depicted in Fig. 4.4. As a result, for the given χ and χ^{-1}, the robot position can be recovered after switching between such two reference frames only with a local frame which is aligned with the center line tangent, such as $\mathcal{A}$.

The second benefit in defining the robot position in $\mathcal{A}(s)$ is related to navigation and path planing. In a typical inspection scenario, the robot must traverse the tunnel at a fixed distance from either the center line or the tunnel surface. When the proposed reference frame definition for the robot position is used, this can be achieved simply by fixing $\mathbf{x}_{y,z}$ to a constant or a value which is a function of the axial position and local radius respectively. However, in case $\mathcal{G}(s)$ were used, the distance to the center line would change according to the variations in the tunnel inclination along the robot's path. These two cases are depicted in Fig. 4.5.

Robot orientation is represented in $\mathcal{G}(s)$ rather than $\mathcal{A}(s)$ due to similar considerations as for the robot position. Due to the dynamics of multi-rotor aerial vehicles, the orientation

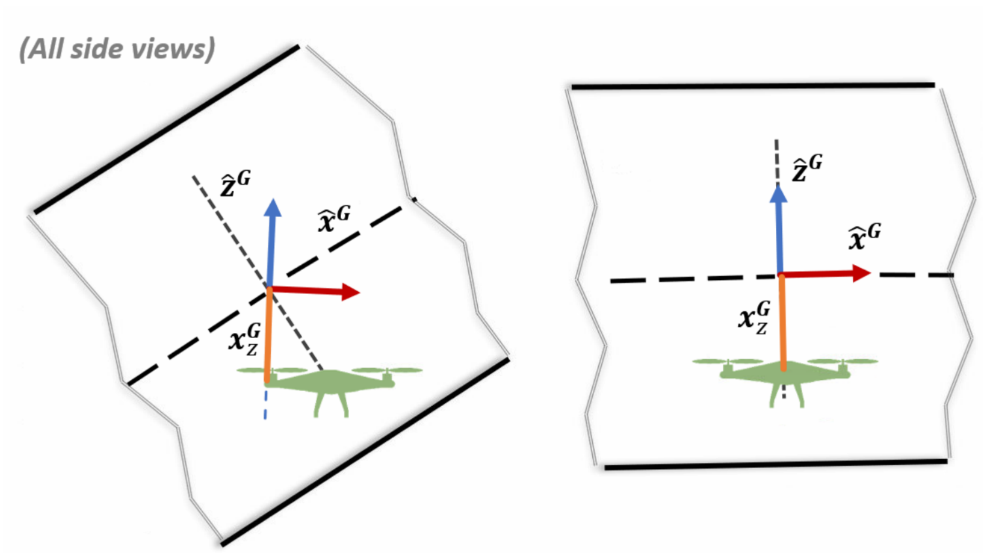


Figure 4.5: This figure shows the position of a robot at two different sections of the tunnel. In between these two states the robot is commanded to follow a trajectory with $\mathbf{x}_z$ being constant. When the inclination changes, in order to keep $\mathbf{x}_z$ constant (orange lines), the distance of the robot from the centerline adjusts accordingly. This is not a desired effect since in a typical inspection scenario, the robot is commanded to follow a path at a constant distance from either the centerline or the tunnel walls.

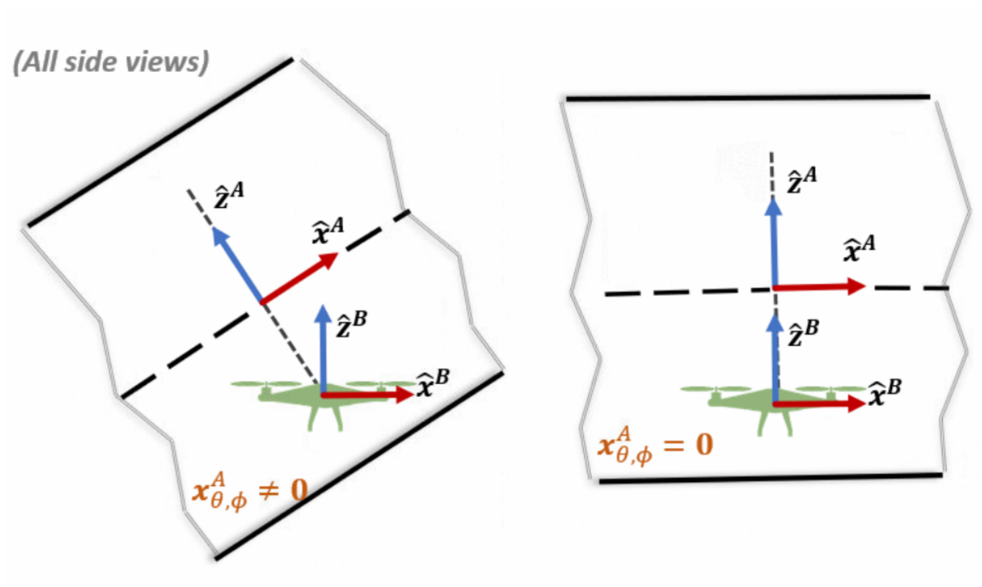


Figure 4.6: This figure illustrates a *hovering* robot in two different sections of a tunnel. The robot orientations as written in their corresponding local map frames differ since the inclinations are different. Hence, the robot orientation is represented in a gravity frame.

commands to the onboard auto-pilot must be given with respect to the gravity vector. For this reason, we represent orientations with respect to $\mathcal{G}(s)$ which is aligned with the gravity vector. Furthermore, as can be seen in Fig. 4.6, the choice of reference frame affects the robot orientation, $\mathbf{x}_{\boldsymbol{\Omega}}$, by a rotation around $\mathcal{A}(s)_{\hat{\mathbf{y}}}$ $\left(= \mathcal{G}(s)_{\hat{\mathbf{y}}}\right)$ at a given axial position, s. This implies that two rotations that are *numerically equal* at different axial positions, s_i and s_j, represent two different orientations if ${}^{\mathcal{G}(s_i)}\mathbf{R}_{\mathcal{A}(s_i)} \neq {}^{\mathcal{G}(s_j)}\mathbf{R}_{\mathcal{A}(s_j)}$. This means that the robot orientation would be a function of axial position if a reference frame which has none of its basis vectors aligned with the gravity vector. This is definitely an unnecessary coupling between state variables.

4.2.2 Rao-Blackwellized Particle Filter

We estimate the robot state using a Rao-Blackwellized Particle Filter (RBPF). A RBPF estimates a subset of its states using particles and the remaining using a Kalman Filter (KF). In this particular case, we run a particle filter (PF) for estimating the axial position of the robot, s, and an Unscented Kalman Filter (UKF) to estimate the remaining state variables. The overall system design is shown in Fig. 4.7.

Unless the robot is close to the terminals of the tunnel or in a region where the bending profile significantly differs from the rest of the tunnel, a range sensor cannot provide informative measurements to estimate $\mathbf{x}_s$. Possible scenarios supporting this observation have been explained in Sec. 4.2. This implies that in such cases uncertainty along the center line is a widely spread distribution not exhibiting a shape of any closed-form distribution. For this reason, we use a PF for position estimation along this specific DoF . Each particle also runs a separate UKF to estimate the rest of the state variables, *i.e.* position and velocity along the tunnel cross-section, orientation and IMU biases. In the following sections, we explain the process and measurement models in detail.

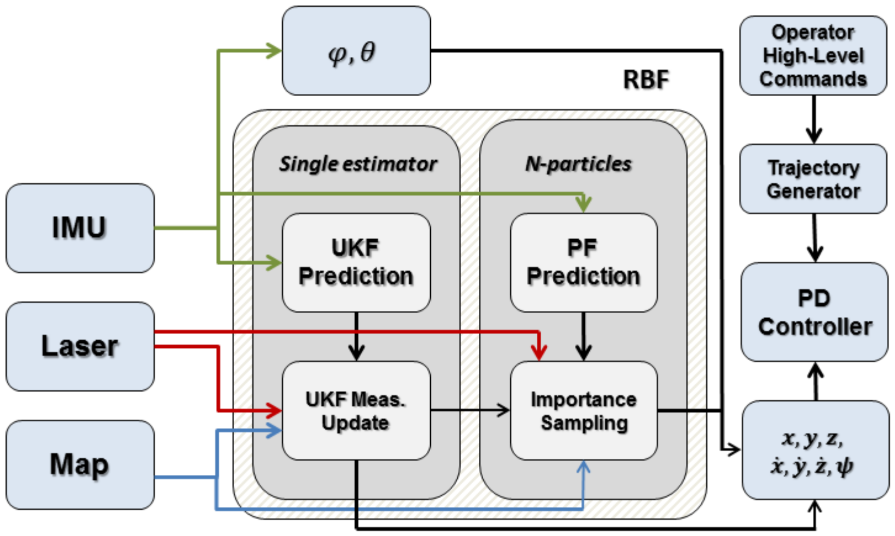


Figure 4.7: The estimator based on a Rao-Blackwellized Particle Filter and a PD controller for autonomous flight in a tunnel of known cross section. A PF with N particles is used to model the propagation of the axial position and its uncertainty, while a UKF is used to estimate the remaining states.

4.2.3 Process Model

The part of the robot state tracked by a UKF is

$$\check{\mathbf{x}} := [\mathbf{r}_{y,z}, \mathbf{\Omega}, \mathbf{v}_{y,z}, \mathbf{b}_{\mathbf{a}}, \mathbf{b}_{\boldsymbol{\omega}}]^\top \tag{4.6}$$

The UKF process model is defined as

$$\check{\mathbf{x}}_{t+1} = f(\check{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{c}_t) \tag{4.7}$$

where $\mathbf{c}$ is the process noise drawn from a zero-mean normal distribution, $\mathbf{u}$ is the control input derived from the raw IMU data. The control input is defined in the IMU frame, $\mathcal{I}$, and writes as:

$$\mathbf{u} = [\mathbf{a}, \boldsymbol{\omega}]^\top \tag{4.8}$$

where $\mathbf{a} = [\ddot{x}, \ddot{y}, \ddot{z}]^\top$ is the body acceleration vector, and $\boldsymbol{\omega} = [p, q, r]^\top$ is the body rotational velocity. The process model implements dynamics of a quadrotor details of which we leave to [74]. Finally the process model, $f(\check{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{c}_t)$, writes as

$$\begin{bmatrix} \mathbf{x}_{y,z} \\ \mathbf{x}_{\dot{y},\dot{z}} \\ \mathbf{x}_{\boldsymbol{\Omega}} \\ \mathbf{x}_{\mathbf{b}} \end{bmatrix}_{t+1} = \begin{bmatrix} \mathbf{x}_{y,z} + \mathbf{x}_{\dot{y},\dot{z}}\,\Delta t + \frac{1}{2}\,\mathbf{P}\,\left({}^{\mathcal{A}}\mathbf{R}_{\mathcal{I}}\,(\mathbf{u_a} - \mathbf{x_{b_a}}) - {}^{\mathcal{A}}\mathbf{R}_{\mathcal{G}}\,\mathbf{g}\right)\Delta t^2 \\ \mathbf{x}_{\dot{y},\dot{z}} + \mathbf{P}\,\left({}^{\mathcal{A}}\mathbf{R}_{\mathcal{I}}\,(\mathbf{u_a} - \mathbf{x_{b_a}}) - {}^{\mathcal{A}}\mathbf{R}_{\mathcal{G}}\,\mathbf{g}\right)\Delta t \\ rpy\Big(\mathbf{R}(\mathbf{x}_{\boldsymbol{\Omega}})\,\left(\mathbf{I} + (\mathbf{u}_{\boldsymbol{\omega}} - \mathbf{x}_{\mathbf{b}_{\boldsymbol{\omega}}})_\times \Delta t\right)\Big) \\ \mathbf{x}_{\mathbf{b}} \end{bmatrix}_t + \mathbf{c}_t \qquad (4.9)$$

where Δt is the control input update period, the function $\mathbf{R}(\cdot)$ gives the rotation matrix corresponding to the given Euler angles, $rpy(\cdot)$ outputs the Euler angles corresponding to the input rotation matrix, $\mathbf{n}_\times$ is the skew-symmetric written as

$$\mathbf{n}_\times = \begin{bmatrix} 0 & -\mathbf{n}_2 & \mathbf{n}_1 \\ \mathbf{n}_2 & 0 & -\mathbf{n}_0 \\ -\mathbf{n}_1 & \mathbf{n}_0 & 0 \end{bmatrix} \qquad (4.10)$$

with the subscripts being zero-based indices, $\mathbf{g}$ is the gravity vector and lastly $\mathbf{P} = [\mathbf{0}_{2\times1},\ \mathbf{I}_{2\times2}]$. It is also obvious that $\mathbf{R}(\mathbf{x}_{\boldsymbol{\Omega}}) = {}^{\mathcal{G}}\mathbf{R}_{\mathcal{B}}$. Finally, we note that for each particle the frames for which the rotation matrices defined are dependent on the corresponding axial position, $\mathbf{x}_s$.

4.2.4 Measurement Model

The measurement from range-based position and yaw estimation algorithm (Sec. 4.3) is

$$\mathbf{z} = [\mathbf{x}_{y,z},\ \mathbf{x}_\psi]^\top \qquad (4.11)$$

where the position estimates are given in the local frame $\mathcal{A}(\mathbf{x}_s)$ for a given particle.

The measurement model is linear which writes as

$$\mathbf{z}_t = \mathbf{H}\check{\mathbf{x}}_t + \mathbf{m}_t \qquad (4.12)$$

where **H** extracts the robot position along the tunnel cross-section at a given axial position, $\mathbf{x}_s$, as well as its yaw. $\mathbf{m}_t$ is the additive noise modeled as a zero-mean normal distribution. The matrix **H** is defined as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & & 0 & \\ 0 & 1 & \mathbf{0}_{3\times2} & 0 & \mathbf{0}_{3\times8} \\ 0 & 0 & & 1 & \end{bmatrix} \tag{4.13}$$

4.3 2D Laser Processing

The onboard 2D laser scanner provides data as an ordered array of range measurements which in its raw form is not very useful. The general approach is to convert range measurements to a point cloud to perform scan registration, mapping etc. This, however, causes information loss since the invalid measurements which may be due to non-reflective surfaces, or ranges longer than the sensor maximum range are not copied into the point cloud at all. To illustrate, lack of a return for a particular laser beam means that the space along the beam direction is unoccupied up to the sensor maximum range. Obviously, this is a valid statement if we are sure that the environment does not contain objects made of materials that completely absorb the laser beams. In order to overcome this shortcoming of the point cloud data type, we propose using a *laser image* which is the counterpart of a camera image.

We define a laser image, $\mathcal{I}$, as an ordered, enumerable, fixed-sized list of equally-sized pixels. In particular, for a 2D laser scanner, these pixels are positioned along a circle centered at the sensor origin. A pixel is defined as $p := \{\hat{\mathbf{p}}, \rho\}$ where $\hat{\mathbf{p}}$ is the unit 3-vector representing its orientation in the sensor frame and ρ is inverse its range. This representation allows for encoding the two special states of a pixel, which occur if a pixel is out of FOV of the sensor or when the range measurement saturates. These two pixel states are represented simply by setting the inverse range component of a pixel to NaN - *not-a-number*, and 0 respectively. With the abuse of notation, when a pixel is used in an algebraic expression we intend its corresponding 3-vector which is $\frac{\hat{\mathbf{p}}}{\rho}$. A careful reader should notice the similarity between

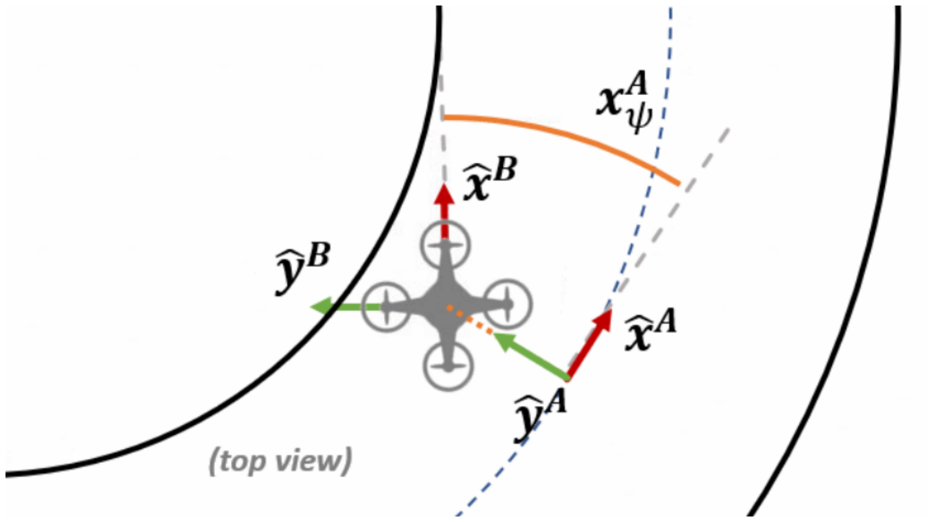


Figure 4.8: This figure illustrates the reference orientation with respect to which the robot yaw angle, $\mathbf{x}_\psi$, is defined. Since the robot yaw angle is defined with respect to the center line tangent, independent of the changes in the tunnel shape, the robot can follow a trajectory with constant yaw.

pixel representation and the homogeneous coordinates of the projective space. Finally, a laser image is a set of pixels, $\mathcal{I} := \{p\}$, defined by its resolution, *i.e.* the number of pixels, $|\{p\}|$. A given pixel, p_i, spans the range $\frac{2\pi}{|\{p\}|}[i, i+1)$ radians where i is the zero-based pixel coordinate. Modeling a 2D laser scanner in this way prevents information loss while preserving sensor layout.

4.3.1 Yaw Estimation

The onboard attitude estimator provides roll and pitch estimates with gravity correction. On the contrary, the yaw angle, $\mathbf{x}_\psi$, drifts unboundedly due to lack of a global reference frame. This problem could have been overcome using a magnetometer, however the metal tunnel prohibits this. Instead, we correct the yaw angle with respect to the tunnel local axis using the onboard laser scanner (Fig. 4.8)

We propose a geometric solution to the yaw estimation problem using the fact that the intersection of a cylinder and a plane is always an ellipse. It is easy to see that the intersection of a plane and a cylinder can result in three different curves which are a circle, an ellipse or two parallel lines. This curve is a circle only when $\mathcal{A}_{\hat{\mathbf{x}}} \parallel \mathcal{B}_{\hat{\mathbf{z}}}$ which is very unlikely to happen in our case. Other two cases are more likely to be observed and both can be treated as ellipses since two parallel lines correspond to the special case of an ellipse with infinite major axis length. The orientation of the major axis of the ellipse is negative the yaw angle up to π radians ambiguity.

We define $\mathbf{x}_\psi = 0$ to be the orientation where $\mathcal{A}_{\hat{\mathbf{y}}} \perp \mathcal{B}_{\hat{\mathbf{x}}}$. Due to the shape of the tunnel, there is no cue using which a range-based estimator could distinguish whether $\mathbf{x}_\psi = \psi_0$ or $\mathbf{x}_\psi = \psi_0 + \pi$. This is because in both cases the laser contours are exactly the same. The ambiguity is resolved by each UKF comparing its most recent yaw estimate to these two alternatives and choosing the closest at the measurement update.

We use the method proposed in [16] which studies ellipse fitting as a linear least-squares problem. The equation of a conic is

$$\begin{aligned} F(\mathbf{q}, \mathbf{d}) &= \mathbf{q} \cdot \mathbf{d} && (4.14) \\ &= ax^2 + bxy + cy^2 + dx + ey + f = 0 && (4.15) \end{aligned}$$

where $\mathbf{q} = [a, b, c, d, e, f]^\top$ are the conic parameters and $\mathbf{d} = \left[x^2, xy, y^2, x, y, 1\right]^\top$. $F(\mathbf{q}, \mathbf{d})$ evaluates to the *algebraic distance* of a given point $[x,\ y]^\top$ to the conic defined by $\mathbf{q}$. [16] constrains the conic parameters such that $b^2 - 4ac < 0$ to guarantee that they represent an ellipse. Since $F(\mathbf{q}, \mathbf{d}) = 0$ can be scaled, this constraint can be converted into an equality as $\mathbf{q}^\top \mathbf{C} \mathbf{q} = 1$ where

$$\mathbf{C} = \begin{bmatrix} \begin{matrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{matrix} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}. \qquad (4.16)$$

The method minimizes the sum of algebraic distances of each point to a given ellipse which can be written as $\|\mathbf{D}\mathbf{q}\|^2$ where $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_N]^\top$. The constraint is incorporated into the problem through a Lagrange multiplier. This equation is then differentiated to get the following system of equations

$$\mathbf{D}^\top \mathbf{D} - \lambda \mathbf{C}\mathbf{q} = 0 \tag{4.17}$$

$$\mathbf{q}^\top \mathbf{C}\mathbf{q} = 1. \tag{4.18}$$

This is a generalized eigenvalue problem which we solve using the Eigen C++ library [72]. The eigenvector corresponding to the only negative eigenvalue is the ellipse parameters.

In its original form, the ellipse equation is not very useful since neither the orientation nor the center of the ellipse are obvious. This equation can be rewritten as

$$\mathbf{r}^\top \begin{bmatrix} a & b/2 \\ b/2 & a \end{bmatrix} \mathbf{r} + \begin{bmatrix} d \\ e \end{bmatrix}^\top \mathbf{r} + f = 0. \tag{4.19}$$

where $\mathbf{r} = [x,\ y]^\top$. In order to find the orientation of the ellipse we first apply eigenvalue decomposition on the 2×2 matrix. The eigenvector corresponding to the larger of the two eigenvalues corresponds to the major axis. We define the orientation of an ellipse with its major axis with π radians of ambiguity. The center of the ellipse is obtained by first differentiating the above equation with respect to $\mathbf{r}$, and then solving for the free variable. The ellipse center is found as

$$\mathbf{r}_o = \frac{1}{2} \begin{bmatrix} a & b/2 \\ b/2 & a \end{bmatrix}^{-1} \begin{bmatrix} d \\ e \end{bmatrix}. \tag{4.20}$$

As seen in Fig. 4.9, laser data may be noisy due to unmodeled obstacles in the environment, inherent noise in the laser scanner or sensor failures such as due to reflective surfaces. A direct fit to raw data is very likely to yield to wrong estimates which may result in complete failure of the estimator and crash the robot. In order to mitigate this problem, we remove the outliers using the robust estimator MLESAC [14].

Most of the studies in the literature prefer RANSAC [5] which is a proven robust estimator. The problem with RANSAC is that a data point is penalized only if it is at a distance to the

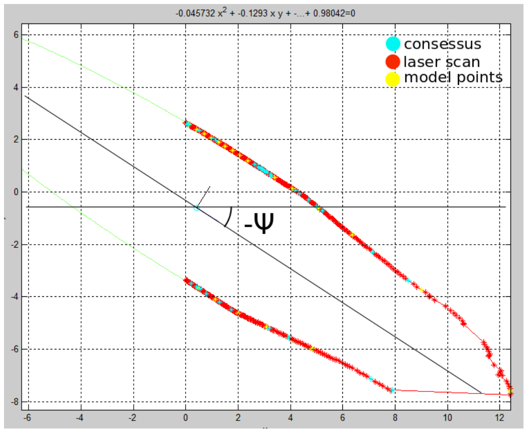


Figure 4.9: A sample laser scan data. The ellipse is fit using the method in [16]. In order to eliminate outliers, we use MLESAC [14]. Outliers are due to operators walking next to the MAV, noise and lidar failures.

model greater than a fixed threshold. This brings up the question of what a good threshold is. Setting the threshold too high may result in poor performance while setting this too small would flag most good data points as outliers. On the other hand, MLESAC [14] defines a cost for each data point which when summed gives a likelihood for a given model parameter set. By simply choosing the parameter set which yields the highest likelihood will perform at least as good as RANSAC. We leave the details of the MLESAC algorithm to the original work [14].

4.3.2 An ICP Algorithm for Position Estimation

In this section we propose an ICP algorithm which uses a minimal set of laser points to estimate the vertical and lateral positions of the platform along the cross-section of the tunnel (*i.e.* the $\mathcal{A}_{\hat{\mathbf{y}}} - \mathcal{A}_{\hat{\mathbf{z}}}$ plane) at a given axial position, $\mathbf{x}_s$. Due to the axisymmetric shape of the tunnel, it is theoretically impossible to estimate the roll angle using only range sensors. The same applies to pitch angle since the robot is not aware of the tunnel inclination. For this, our estimator relies on the IMU measurements for roll and pitch angle estimation which affects the range-based localization.

The onboard 2D Hokuyo laser scanner has ~ 210 degrees of FOV on the $\mathcal{B}_{\hat{\mathbf{x}}} - \mathcal{B}_{\hat{\mathbf{y}}}$ plane. A 3D printed dual-mirror mount is fixed on top of the laser scanner to reflect rays in upward

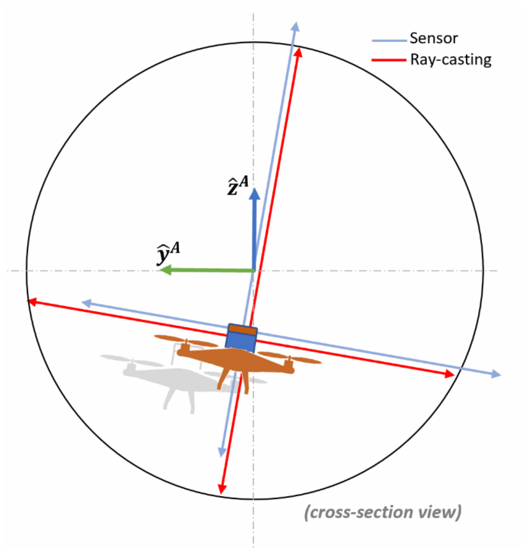


Figure 4.10: Starting from an initial pose, an ICP iteratively refines $\mathbf{x}^{\mathcal{A}}_{y,z}$ to reduce discrepancy between the actual range measurements and the rays cast against the map. This figure shows the process on a tunnel cross-section which can be at either a horizontal or inclined section.

and downward ($\pm\mathcal{B}_{\hat{\mathbf{z}}}$) directions [68] (Fig. 3.8a). These measurements together with the orientation estimate and the knowledge of the map are used to localize the robot on the $\mathcal{A}_{\hat{\mathbf{y}}} - \mathcal{A}_{\hat{\mathbf{z}}}$ plane of the tunnel using a derivative of the ICP algorithm.

In order to attain real time performance, we first convert the raw measurements into a laser image and use only a subset of its pixels. We define the *effective FOV* of the laser scanner as the intersection of its original FOV and two cones. These cones have fixed apex angles, their apexes at the sensor origin and are oriented along $\pm\mathcal{A}_{\hat{\mathbf{y}}}$ directions. It should be noted that the intersection of a cone with the laser scanner FOV may be empty if the apex angle is small or the platform orientation about $\mathcal{G}_{\hat{\mathbf{x}}}$ is large (*i.e.* large roll angle). The intersection of the sensor FOV and the cones can be easily found by iterating over all the pixels of the laser image and marking the range component of the pixels that fall outside both cones with NaN as discussed previously. The beams that are reflected by the two mirrors are also converted into a laser image. This image has only two pixels with their centers pointing along $\pm\mathcal{B}_{\hat{\mathbf{z}}}$ directions. Since the number of pixels is already low, we do not apply any filters on the second laser image. We call the two laser images as $\mathcal{I}_l := \{p_l\}$ and $\mathcal{I}_v := \{p_v\}$ where superscripts refer to *lateral* and *vertical*, $p_{\alpha,i}$ is the i^{th} pixel of the corresponding laser image.

The *closest* point on the map to a range measurements is found by casting a ray against the occupancy grid map, M, along the measurement direction emanating from the sensor origin. We call the resultant set of vectors as $\{\tilde{p}\}$. All pixels of the two laser images are matched to map points in this way. We then iteratively update the robot position on the tunnel cross-section to reduce the discrepancy between the measurement and the cast rays. Since this method localizes the robot only along the cross-section of the tunnel, the residuals due each measurement is projected onto the $\mathcal{A}_{\hat{\mathbf{y}}} - \mathcal{A}_{\hat{\mathbf{z}}}$ plane. The residual can be calculated as

$$\mathbf{r}_i = \left(\mathbf{I} - \hat{\mathbf{x}}\, \hat{\mathbf{x}}^\top\right)(p_i - \tilde{p}_i) \tag{4.21}$$

where all vectors are in $\mathcal{A}(\mathbf{x}_s)$. The measurement uncertainty is the covariance of the residuals calculated after convergence. Algo. 1 explains this method. Also a snapshot of this procedure is illustrated in Fig. 4.10. Finally, due to the convexity of the tunnel cross-section, this algorithm is guaranteed to converge.

Algorithm 1: Iterative Closest Point for Localization on $\hat{\mathbf{y}}^{\mathcal{A}} - \hat{\mathbf{z}}^{\mathcal{A}}$

```
Data: $[\mathbf{x}, \{p_l\}, \{p_v\}, M]$
Result: $[\mathbf{x}_{y,z}, \mathbf{\Sigma}_{y,z}]$
/* residual projection matrix */
$\mathbf{P} \Leftarrow [\mathbf{0}_{2\times 1},\ \mathbf{I}_{2\times 2}] \left(\mathbf{I} - \mathcal{A}(\mathbf{x}_s)_{\hat{\mathbf{x}}}\, \mathcal{A}(\mathbf{x}_s)_{\hat{\mathbf{x}}}{}^{\top}\right)$
/* merge the two pixel sets */
$\{p\} \Leftarrow \{p_l\} \cup \{p_v\}$
/* transform measurements from $\mathcal{B}$ to $\mathcal{A}$ */
foreach $p_i \in \{p\}$ do
    $p_i \Leftarrow {}^{\mathcal{A}(\mathbf{x}_s)}\mathbf{R}_{\mathcal{B}(\mathbf{x})}\ p_i$
end
$\delta_{y,z} \Leftarrow \infty$
while $iter < iter_{max} \ \wedge\ \epsilon < ||\delta_{y,z}||_2$ do
    $\delta_{y,z} \Leftarrow \mathbf{0}_{2\times 1}$
    $\mathbf{\Sigma}_{y,z} \Leftarrow \mathbf{0}_{2\times 2}$
    $n \Leftarrow 0$ /* number of successful ray-casts */
    foreach $p_i \in \{p\}$ with $p_{i,\rho} \neq NaN$ do
        /* ray_cast(origin, direction, occupancy grid map) */
        $\tilde{p_i} \Leftarrow ray\_cast\left(\mathcal{B}(\mathbf{x}_s)_{\mathcal{O}}, p_i, M\right) \ - \ \mathcal{B}(\mathbf{x})_{\mathcal{O}}$
        if ray-casting successful then
            $n \Leftarrow n \ + \ 1$
            $\delta = \mathbf{P}\left(p_i - \tilde{p_i}\right)$
            $\delta_{y,z} \Leftarrow \delta_{y,z} \ + \delta$
            $\mathbf{\Sigma}y, z \Leftarrow \mathbf{\Sigma}y, z + \delta\delta^{\top}$
    end
    if $n = 0$ then
        break
    $\delta_{y,z} \Leftarrow \frac{1}{n}\delta_{y,z}$
    $\mathbf{x}_{y,z} \Leftarrow \mathbf{x}_{y,z} \ + \ \delta_{y,z}$
    $\mathbf{\Sigma}y, z \Leftarrow \frac{1}{n}\mathbf{\Sigma}y, z$
end
```

4.4 Particle Weighing and Resampling

The objective of a PF is to represent a continuous distribution with a set of discrete samples. These samples, *i.e.* particles, are composed of a state and a weight. Normally, the weights of all particles should be equal if sufficiently many of them could be sampled. However, for practical concerns, an approximation to this effect is achieved by sampling a finite, computationally tractable number of particles with weights proportional to their state likelihoods [119]. In our case, the weight of a particle is proportional to the likelihood of the laser scanner measurements at its state for the given map.

4.4.1 Particle Weighing

The PF estimates the axial position, $\mathbf{x}_s$, of the robot which is a non-negative scalar upper-bounded by the length of the tunnel centerline, *i.e.* $\mathbf{x}_s \in [0, s_{max}]$. Each particle is assigned a weight according to two factors. One of these factors is the covariance of the ICP algorithm explained in the previous section. Since the inclination changes along the tunnel axis, and the sections of the tunnel might have different radii as well as bending profiles, residual errors in the ICP algorithm yield to be higher for particles sampled at sections of the tunnel dissimilar to where to robot actually is. Secondly, we evaluate the likelihood of the laser beams that are aligned with the tunnel axis. This set of beams (*i.e.* laser image pixels) are obtained by filtering the original laser image with a cone pair centered at the sensor origin, oriented along $\pm\mathcal{G}_{\hat{\mathbf{x}}}(\mathbf{x}_s)$, and a certain apex angle. These beams *see through* the tunnel and provide the most information especially if the robot is flying along a horizontal section and sufficiently close to one end of it. If the robot is at a distant from the end of a horizontal section closer than the maximum range of the laser scanner, the robot can be localized accurately. Furthermore, lack of measurements also provides valuable information which basically means that the robot is far from end of all horizontal sections. The laser image data structure makes such queries possible since it stores the saturated measurements as well as other successful ones. Similar considerations apply to cases where the robot flies along an inclined section.

The weight, w, of a given particle is calculated by incorporating information from the two factors as

$$-\log(w) = \sqrt{tr\left(\mathbf{\Sigma}_{y,z}\right)} + \frac{1}{|\{p\}|}\sum_{j}^{|\{p\}|} L(p_j, \mathbf{x}, M) \tag{4.22}$$

where $\{p\}$ is the set of laser image pixels after filtering as explained above and $|\{p\}|$ is its cardinality, $L(p, \mathbf{x}, M)$ is the negative log-likelihood of a single measurement which is defined as

$$L(p, \mathbf{x}, M) = \begin{cases} 0 & p_\rho = 0,\ \tilde{p}_\rho = 0 \\ L_{min} & p_\rho \neq 0,\ \tilde{p}_\rho = 0 \\ L_{min} & p_\rho = 0,\ \tilde{p}_\rho \neq 0 \\ \gamma\left|p_\rho^{-1} - \tilde{p}_\rho^{-1}\right| & otherwise \end{cases} \tag{4.23}$$

where $\tilde{p}$ is the supposed laser measurement (as a laser pixel) obtained by casting a ray from the particle state, $\mathbf{x}$, against the map, M, p_ρ is its inverse range and $\gamma \in \mathbb{R}^+$.

4.4.2 Particle Resampling

The two basic steps of a PF are the *particle propagation* and the *particle weight computation*. Typically, the particle propagation step applies the process model onto each particle [10]. In our case, the process model is identity for $\mathbf{x}_s$ since there is no higher order terms such as axial velocity that will accumulate on this over time or coupling with the other state variables. The weight computation step is already considered in the previous section. A general problem of PFs is *degeneracy* which happens when a subset of particles have zero weight, or a few particles have much larger weights such that the rest of the particles have no effect. When only these two steps are applied, degeneracy is, in most cases, inevitable. This problem is solved with resampling the particles according to their weights, which is called the *importance resampling* [119].

There are various approaches to the resampling problem which is covered in detail in the survey paper [119]. In this work we employ the *systematic resampling* approach due to its lower CPU demand. This method divides the interval $(0, 1]$ into N disjoint equal subintervals. Only a single random value in $\left(0, \frac{1}{N}\right]$ from a uniform distribution is sampled and the remaining $N - 1$ values are obtained deterministically as

$$u_0 \sim U\left(0, \frac{1}{N}\right] \tag{4.24}$$

$$u_n = u_0 + \frac{n}{N} \quad , \quad n = 1, ..., N - 1. \tag{4.25}$$

Algo. 2 provides a pseudo code of this approach.

Algorithm 2: Systematic Particle Resampling

```
Data: $[\{\mathbf{x}, w\}]$
Result: $[\{\tilde{\mathbf{x}}\}]$
$\{Q\} = cumulativeSum\left(\{w\}\right)$ /* find the cumulative sum of the particle weights */
$n \Leftarrow 0$ /* count the processed number of particles */
$m \Leftarrow 0$ /* book-keep the bin in the cumulative dist.  */
/* the only random value sampling from a uniform dist.  */
$u_0 \Leftarrow sampleUniform\left(0, \frac{1}{N}\right)$
/* loop until $N$ particles are resampled */
while $n < N$ do
    $u \Leftarrow u_0 + \frac{n}{N}$
    /* find the bin that $u$ falls inside */
    while $Q_m < u$ do
        $m \Leftarrow m + 1$
    end
    $n \Leftarrow n + 1$
    $\tilde{\mathbf{x}}_n \Leftarrow \mathbf{x}_m$
end
```

4.5 Experimental Results

In this section we present and interpret results of our experimental work. Dataset were collected in three different sites: Carters Dam, GA and the Allatoona Dam, GA, and in a long university building hallway.

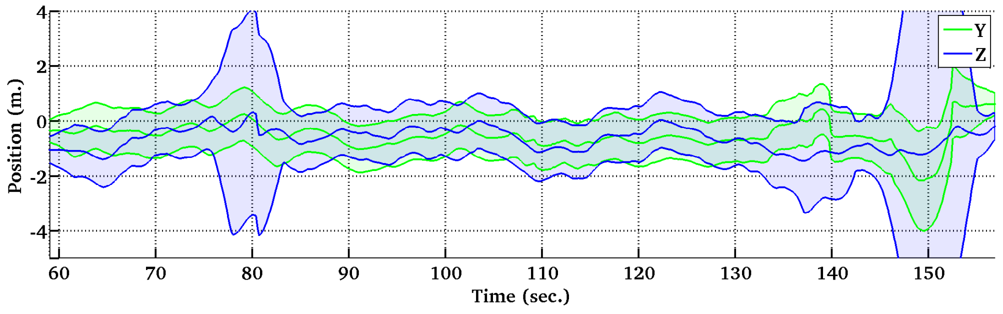
In Fig. 4.11-4.12-4.13 we give results for our Allatoona Dam, Carters Dam and university building experiments respectively. These experiments show that quadrotors programmed to fly semi-autonomous should be considered as the preferred platform for inspection of tunnel-like environments. Only with a laser scanner and an IMU robust localization and control along the tunnel cross-section can be performed in real-time. Lastly, when a tunnel terminal is within the range of the onboard laser scanner, localization along the tunnel axis is achieved as well.

In our initial visit to Carters Dam, we collected two datasets flying the robot manually. In the first flight the platform traversed along only the horizontal section of a penstock. The second dataset was collected while the robot flew close to the junction region and also into the inclined section. Since the human operator could not climb along the inclination due to steep inclination, wet and slippery surface, the operator could not follow the robot which prohibited flying the robot further along the inclined section. We processed these first datasets offline to assess the quality of our estimator. Results for these dataset are shown in Fig. 4.11

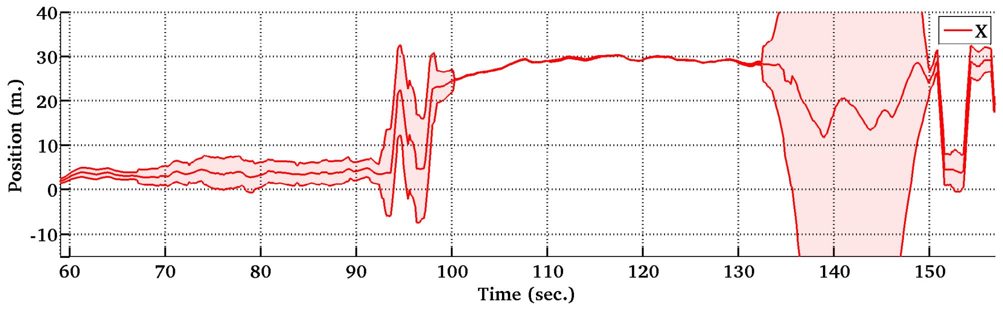
In our field tests at Allatoona Dam, GA, we flew the robot in semi-autonomous mode where the position along the tunnel cross-section and the heading are commanded through a radio control by a human operator. These high level commands given by the operator were executed by the low level onboard controllers. The operator also controls the acceleration along the center line direction by adjusting the pitch angle. These semi-autonomous flight tests prove accuracy and stability of our estimator. Unlike a ground robot, flaws in the controller or estimator would cause instabilities and cause the robot to crash. Results for these dataset are shown in Fig. 4.12

In Fig. 4.12b at the 40^{th} second, increase in the covariance is due to a researcher walking near by the quadrotor. However, the estimator could handle this case and position estimate was not affected significantly. In Fig. 4.11b-4.13a, the covariance increases during the periods when the robot is away from the tunnel/corridor terminal with the following exceptions. In Fig. 4.11b around the 160^{th} second increase in uncertainty is because of sensor failure due to water drainage behaving as a mirror. Lastly the increase in variance in Fig. 4.13a around the $100^{th} - 120^{th}$ and 160^{th} seconds is because the quadrotor pitched such that the laser scanner could only see the floor. In this case the marble floor tilings fail the laser scanner due to the mirror effect.

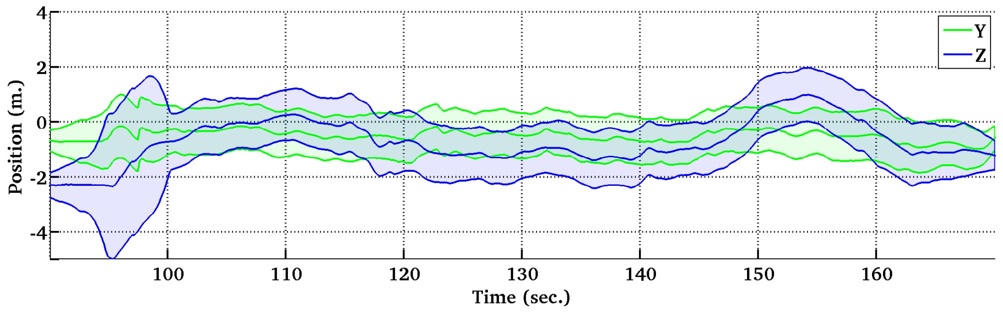
We conducted a third experiment in a building at the University of Pennsylvania, along a $\sim$42 meters long corridor where the quadrotor flies semi-autonomously. In the corridor experiment, although there are geometric features such as pillars and doors, the point cloud map is a featureless rectangular prism. So there is no feature in our map that would help in estimating position along the center line. Indeed, those features behave like noise for yaw estimation which shows robustness of our estimator.



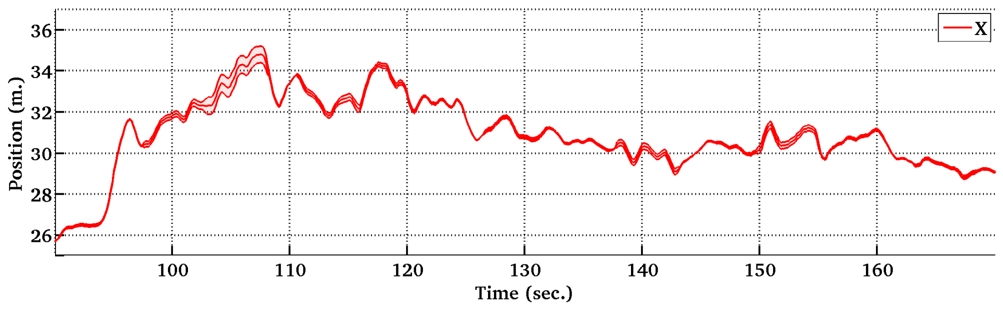
(a) Experiment #1 at Carters Dam, GA.



(b) Experiment #1 at Carters Dam, GA.

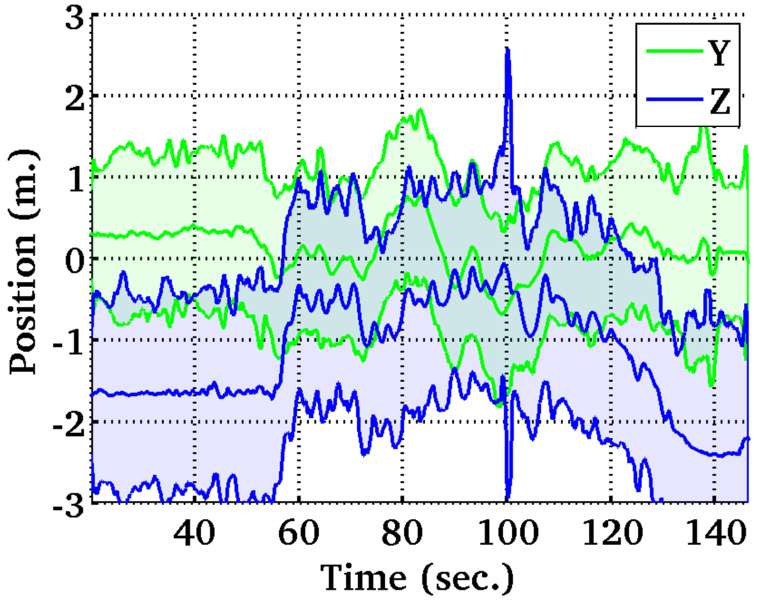


(c) Experiment #2 at Carters Dam, GA.

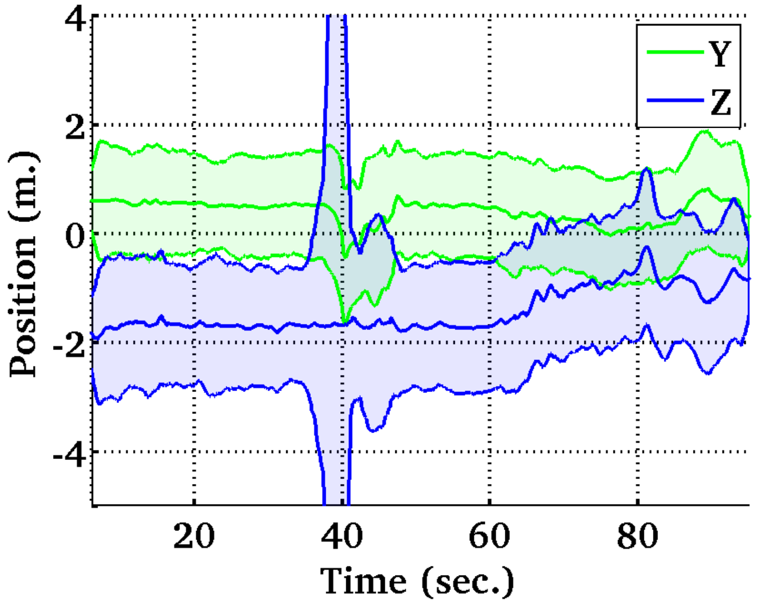


(d) Experiment #2 at Carters Dam, GA.

Figure 4.11: These figures show robot position estimates along with their respective inflated covariances. y and z positions are with respect to the local center line. Unlike the tests at Allatoona Dam (Fig. 4.12), the tunnel walls were not wet and reflective, hence we could collect a good dataset close to the junction. Fig. 4.11b clearly shows the period that the robot is localized along the center line. This period is longer in Fig. 4.11d since the robot flies close to the junction. Localization fails in high covariance regions for both tests.

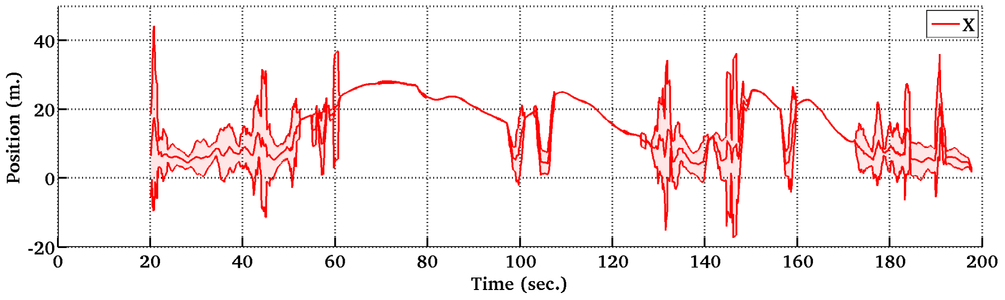


(a) Experiment #1 at Allatoona Dam, GA.

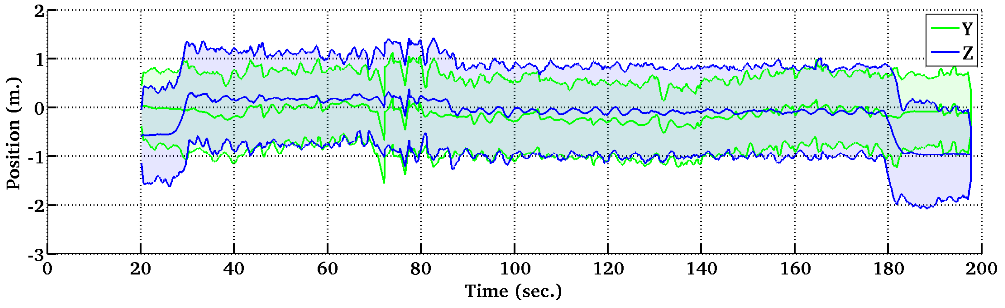


(b) Experiment #2 at Allatoona Dam, GA.

Figure 4.12: These figures show position estimates of $\mathbf{x}^{\mathcal{A}}_{y,z}$ along with their respective inflated covariances. In these experiments, the platform flies semi-autonomously. Due to wet, reflective surfaces, laser scanner failed to take measurements along the center line direction. Hence we cannot estimate position along the tunnel axis. Failure in the dataset #2 at the 40^{th} second is due to occlusion.



(a) Experiment in a university building corridor.



(b) Experiment in university building.

Figure 4.13: These figures show results for tests carried in a corridor of length ∼42 meters in a building at University of Pennsylvania. 3D position estimates along with their inflated covariances are presented. Videos of this experiment can be found at: http://mrsl.grasp.upenn.edu/tolga/FSR2013.mp4

Chapter 5

State Estimation Using a Heterogeneous Sensor Suit

In this chapter we study the state estimation problem inside a tunnel using the KHex platform (Sec. 3.2.3) equipped with a richer set of sensors. The onboard sensor suite includes two 2D Hokuyo lidars, four BlueFox XGA grayscale cameras one of which is used for velocity estimation along the tunnel axis and an IMU. One of the lidars is tilted downwards to measure the altitude in addition to contributing to the lateral position and orientation estimation. The tilted lidar replaces the dual-mirror mount of the Pelican platform to provide more reliable and richer measurements for altitude estimation. Since the tunnel inclination changes, this laser is tilted in such a way that at all sections of the tunnel it sees the tunnel floor at a small angle of incidence on the average.

Similar to the approach of the previous chapter we assume that a parametric map of the environment is provided. Hence, this chapter studies the *localization* problem as well rather than the SLAM problem. The most significant improvement over the previous approach is the velocity estimation along the tunnel axis which in turn enables the full state estimation using a similar filtering mechanism. Briefly, information from all the sensors are fed into a central UKF to estimate the 5 DoF robot state as well as its axial velocity. We present results from experiments in Carters Dam, GA and Glen Canyon Dam, AZ for range-based estimation and visual odometry. This chapter is based on our previous work at [127].

5.1 A Discussion on The Requirement of a Prior Map

Two 2D laser scanners with fixed relative poses theoretically provide sufficient information for fitting a cylinder. This could have been used to eliminate the requirement for a prior map. A cylinder can be defined by 5 parameters which are its radius (1), normalized axis (2) and plane-intercept of its axis (2). While only 5 points are sufficient to fit a cylinder, there are 6 possible solutions with only an even number them being real (*i.e.* non-complex) cylinders [49]. Therefore, multiplicity of the solutions prohibits direct application of a 5-point approach to a robotics application with real-time performance requirements. This situation can be remedied through employing a filtering mechanism that chooses among the possible solutions provided that certain priors are available. Another alternative is to run the 5-point algorithm for multiple point tuples and choose the similar solutions from each run. Although it is beyond the scope of this work, it can be easily shown that slight perturbations of point positions cause significant changes in some of the solutions reducing the likelihood of obtaining multiple *common* solutions from multiple runs. Regardless, algebraic methods such as [49], [124] cannot be directly used to solve the cylinder fitting problem in a robotics scenarios since these family of methods assume absolute point accuracy.

The robotics community has produced works that can handle noise in the point cloud data such as [21], [34], [98], [112]. However these methods assume a dense point cloud is provided. To illustrate [98] fits cylinders to a point cloud reconstruction of a piping system which consists of several millions of points. Since this approach relies on surface normals for tracking pipe directions, it cannot be applied to our case. Likewise, [112] approximates the skeleton of a piping system from a point cloud map consisting of several millions of points. Due to these reasons and our safety concerns pertaining to stability issues, we assume, for this work, that the map of the tunnel is provided, and leave online cylindrical map estimation using 2D laser scanners as a future work.

5.2 Nomenclature and Definitions

This work assumes that a map of the tunnel is given parametrically which can be converted into a mesh or a point cloud similar to the previous chapter. In particular, the map is converted into an occupancy grid with the cell size of 5 cm.

Since the sensor suit includes multiple range sensors and cameras, each is given a unique name to prevent ambiguity. The frames of the two laser scanners are denoted as $\mathcal{L}_b$ and $\mathcal{L}_t$ where the subscripts b and t refer to bottom and top respectively. As explained in Sec. 3.2.3, the cameras are oriented to the right, left, top and bottom of the platform with respect to the body frame. We refer to the reference frames of each camera as $\mathcal{C}_r$, $\mathcal{C}_l$, $\mathcal{C}_t$ and $\mathcal{C}_b$ respectively. The IMU frame is denoted as $\mathcal{I}$ which we assume to be coincident with the body frame, $\mathcal{B}$. The relative pose of a sensor, S_a, with respect to another sensor or the robot body, S_b, is given with the rotation matrix and the translation vector pair as $\left\{ {}^{\mathcal{S}_b}\mathbf{R}_{\mathcal{S}_a},\ \mathcal{O}_{\mathcal{S}_a}^{\mathcal{S}_b} \right\}$ where the first element of the tuple is the rotation from S_a frame to S_b frame, and the second element is the origin of S_a with respect to S_b origin.

Data structure of both sensors are modeled as a *set of pixels* similar to the previous chapter. Camera and laser scanner pixels are *geometrically* equivalent hence using the same mathematical pixel structure facilitates the formulations. The properties that are not common to both pixel types are the color and the intensity values of camera and laser scanner pixels respectively, however these are not used by the proposed estimation methods. We discriminate between pixels from different sensor by a proper notation such as $\left\{ p_{\mathcal{C}_i} \right\},\ i \in \{r, l, t, b\}$ and $\left\{ p_{\mathcal{L}_i} \right\},\ i \in \{t, b\}$ which refer to a camera and a laser image respectively.

5.3 Sensor Fusion for State Estimation

This section explains the system and filter designs as well as the measurement models for the two exteroceptive sensors. The state estimator introduced in the previous chapter is improved by a vision-based velocity estimator complementing the range sensors to give a

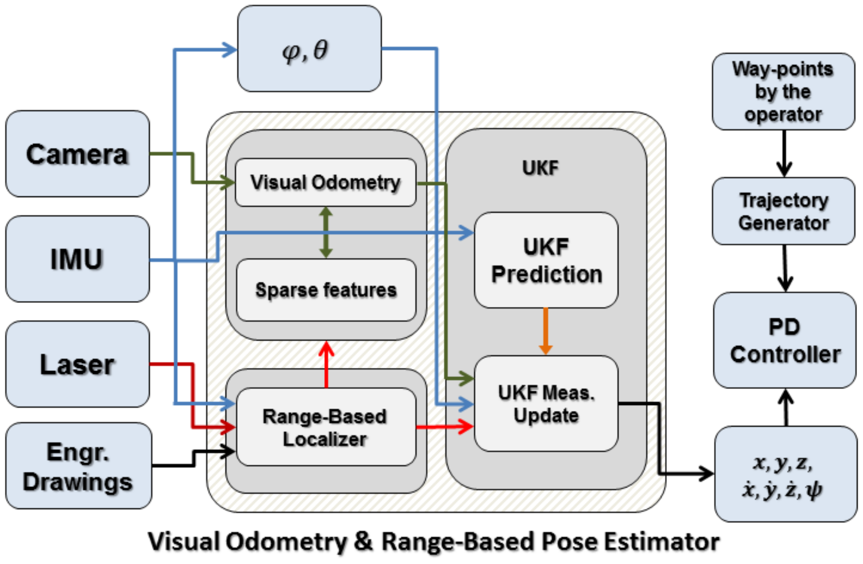


Figure 5.1: This figure shows processes in a data flow diagram. Inputs to the system are the IMU, lidar and camera measurements, and the map of the tunnel. Partial pose estimates from the range-based localizer and the visual odometry are fused by the UKF node. The operator gives way-point commands using an RC to the trajectory generator output of which is fed to the onboard PD controller.

full state estimation. The frequency of the estimator is determined by the sensor rates. The UKF prediction step runs at the 100 Hz IMU rate, and the range and vision-based measurement updates run at 40 Hz and 24 Hz respectively.

The robot lateral position and yaw angle are estimated through a robust iterative closest point algorithm that runs on the two laser scanners. The most recent filtered robot state is fed to the vision-based estimator as its initialization point. The velocity estimate is then fed back to the UKF to update the robot axial position.

In Fig. 5.1 we show the data flow diagram with each box corresponding to a process. The inputs to the system are the map, M, IMU data, frames from the right camera and range measurements from the two lidars. The UKF node outputs 6 DoF pose estimate which is fed to a PD controller. The operator gives way-point commands to the trajectory generator using a remote control. Finally, the onboard PD controller generates low-level controller commands in accordance with the pose estimate and the trajectory.

On this platform, only one of the four cameras is used during flight tests since the USB 2 bandwidth is not enough to grab frames from all the four cameras at high frame rates. We also collected additional datasets with all the four cameras active to grab images from the four sides of the robot which we then use to form 360 degrees image panoramas as will be explained in the subsequent sections. These images can later be used to facilitate locating cracks and rusty spots by the maintenance engineers.

5.3.1 Robot State

We adopt the state definition of the previous chapter with the addition of position and speed along the tunnel axis and bias terms for roll and pitch which writes

$$\mathbf{x} := [s, \dot{s}, \mathbf{r}_{y,z}, \boldsymbol{\Omega}, \mathbf{v}_{y,z}, \mathbf{b}_{\phi,\theta}, \mathbf{b}_{\mathbf{a}}, \mathbf{b}_{\boldsymbol{\omega}}]^{\top} \tag{5.1}$$

where s and $\dot{s}$ are the position and speed along the center line, $\mathbf{r}_{y,z}$ and $\mathbf{v}_{y,z}$ are the y and z coordinates of the position and velocity defined in $\mathcal{A}(s)$, $\boldsymbol{\Omega} := \{\phi, \theta, \psi\}$ is the orientation defined in $\mathcal{G}(s)$, $\mathbf{b}_{\phi,\theta}$ is the roll and pitch biases of the onboard attitude estimator, $\mathbf{b}_{\mathbf{a}}$ and $\mathbf{b}_{\boldsymbol{\omega}}$ are the accelerometer and gyroscope biases given in the IMU frame, $\mathcal{I}$. $\boldsymbol{\Omega}$ is the Euler angles in XYZ order which can be written in matrix form as ${}^{\mathcal{G}(s)}\mathbf{R}_{\mathcal{B}(s)} = \mathbf{R}_z(\mathbf{x}_\psi)\mathbf{R}_y(\mathbf{x}_\theta)\mathbf{R}_x(\mathbf{x}_\phi)$ where $\mathbf{x}_\phi, \mathbf{x}_\theta, \mathbf{x}_\psi$ are the roll, pitch and yaw angles respectively, and $\mathbf{R}_i(\gamma)$ is the rotation matrix around axis i by γ radians. The axial speed, $\dot{s}$, is observable to the vision-based estimator and all other state variables except of the axial position, s, are observable to the range sensor and the IMU. The axial position, s, is obtained by integrating the axial speed over time.

5.3.2 Process Model

The UKF process model is defined as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{c}_t) \tag{5.2}$$

where the definitions of the arguments are the same as the process model of the previous chapter (Equ. 4.9). With the addition of the full velocity estimate, the process model is updated as

$$\begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_{\dot{s}} \\ \mathbf{x}_{y,z} \\ \mathbf{x}_{\dot{y},\dot{z}} \\ \mathbf{x}_{\mathbf{\Omega}} \\ \mathbf{x}_{\mathbf{b}} \end{bmatrix}_{t+1} = \begin{bmatrix} \mathbf{x}_s + \mathbf{x}_{\dot{s}}\ \Delta t \\ \mathbf{x}_{\dot{s}} \\ \mathbf{x}_{y,z} + \mathbf{x}_{\dot{y},\dot{z}}\ \Delta t + \frac{1}{2}\,\mathbf{P}\ \Big({}^{\mathcal{A}}\mathbf{R}_{\mathcal{I}}\ (\mathbf{u}_{\mathbf{a}} - \mathbf{x}_{\mathbf{b}_{\mathbf{a}}}) - {}^{\mathcal{A}}\mathbf{R}_{\mathcal{G}}\ \mathbf{g}\Big)\Delta t^2 \\ \mathbf{x}_{\dot{y},\dot{z}} + \mathbf{P}\ \Big({}^{\mathcal{A}}\mathbf{R}_{\mathcal{I}}\ (\mathbf{u}_{\mathbf{a}} - \mathbf{x}_{\mathbf{b}_{\mathbf{a}}}) - {}^{\mathcal{A}}\mathbf{R}_{\mathcal{G}}\ \mathbf{g}\Big)\Delta t \\ rpy\Big(\mathbf{R}(\mathbf{x}_{\mathbf{\Omega}})\ \big(\mathbf{I} + (\mathbf{u}_{\boldsymbol{\omega}} - \mathbf{x}_{\mathbf{b}_{\boldsymbol{\omega}}})_{\times}\Delta t\big)\Big) \\ \mathbf{x}_{\mathbf{b}} \end{bmatrix}_{t} + \mathbf{c}_t \tag{5.3}$$

where $\mathbf{x}_{\mathbf{b}}$ refers to all three bias terms.

5.3.3 Fusing Multiple Sensory Information

The proposed filter performs separate measurement updates for the IMU, the laser scanners and the camera since each sensor runs at a different rate. The thermally calibrated IMU and the proprietary onboard controller of the KHex platform provides accurate roll and pitch estimates with gravity correction. In order to benefit from the onboard attitude estimate (implementation and calibration details of which is unavailable), our UKF implements a measurement update for the roll and pitch angles at a reduced frequency of 20 Hz. An IMU measurement is defined as

$$\mathbf{z}^{\mathcal{I}} = [\phi,\ \theta]^{\top} \tag{5.4}$$

where both angles are given in the IMU frame, $\mathcal{I}$, which is coincident with the body frame, $\mathcal{B}$. The measurement from the range-based localization algorithm (Sec. 5.4) is defined as

$$\mathbf{z}^{\mathcal{L}} = [y,\ z,\ \psi]^{\top} \tag{5.5}$$

where the position estimates are given in the local frame $\mathcal{A}(\mathbf{x}_s)$. Lastly, the vision-based axial speed measurement is defined as

$$\mathbf{z}^{\mathcal{C}} = \dot{s}. \tag{5.6}$$

The measurement models for each sensor are linear which write as

$$\mathbf{z}_t^{\mathcal{I}} = \mathbf{H}^{\mathcal{I}}\,\mathbf{x}_t + \mathbf{m}_t^{\mathcal{I}} \tag{5.7}$$

$$\mathbf{z}_t^{\mathcal{L}} = \mathbf{H}^{\mathcal{L}}\,\mathbf{x}_t + \mathbf{m}_t^{\mathcal{L}} \tag{5.8}$$

$$\mathbf{z}_t^{\mathcal{C}} = \mathbf{H}^{\mathcal{C}}\,\mathbf{x}_t + \mathbf{m}_t^{\mathcal{C}} \tag{5.9}$$

where the matrices are, respectively, defined as

$$\mathbf{H}^{\mathcal{I}} = [\mathbf{0}_{2\times4},\ \mathbf{I}_{2\times2},\ \mathbf{0}_{2\times11}] \tag{5.10}$$

$$\mathbf{H}^{\mathcal{L}} = \begin{bmatrix} & 1 & 0 & & 0 & \\ \mathbf{0}_{3\times2} & 0 & 1 & \mathbf{0}_{3\times2} & 0 & \mathbf{0}_{3\times10} \\ & 0 & 0 & & 1 & \end{bmatrix} \tag{5.11}$$

$$\mathbf{H}^{\mathcal{C}} = [0,\ 1,\ \mathbf{0}_{1\times15}] \tag{5.12}$$

$\mathbf{m}_t^{\mathcal{I}}$, $\mathbf{m}_t^{\mathcal{L}}$ and $\mathbf{m}_t^{\mathcal{C}}$ are the additive noise modeled as a zero-mean normal distribution.

5.4 Range-Based Partial State Estimation

This section gives the details of the range-based estimation algorithm which aligns the robot with respect to its current local frame derived from the map, M, and the current axial position of the robot, $\mathbf{x}_s$. Secondly, an uncertainty estimate of the partial state is also derived which is a requirement of the filtering framework. The proposed algorithm employs multiple sensors with their extrinsic calibrations defined with respect to the body frame which coincides with the IMU frame. This is because the onboard attitude controller expects the commands in the IMU frame. The extrinsic calibration of each lidar is represented by rotation-translation pairs, *i.e.* $\{{}^{\mathcal{B}}\mathbf{R}_{\mathcal{L}_t}, \mathcal{O}^{\mathcal{B}}_{\mathcal{L}_t}\}$ and $\left\{{}^{\mathcal{B}}\mathbf{R}_{\mathcal{L}_b}, \mathcal{O}^{\mathcal{B}}_{\mathcal{L}_b}\right\}$ for the top and the bottom laser scanners respectively. Fig. 3.10 illustrates the sensor configuration.

The raw range readings from each laser scanner are first converted into two laser images (Sec. 4.3). Indefinite measurements and rays with ranges larger than $6m$ are filtered out at the preprocessing stage before being fed into the ICP algorithm. The latter filter is important since it reduces the effect of noisy measurements in the yaw estimation ($\mathbf{x}_\psi$) which in tightly coupled with the lateral position estimation ($\mathbf{x}_{y,z}$). Details of this behavior is explained in the iterative least squares formulation. Furthermore, converting the dense range measurements to a laser image also both smooths out noise and downsamples the data hence saves CPU time.

5.4.1 Iterative Least-Squares Formulation

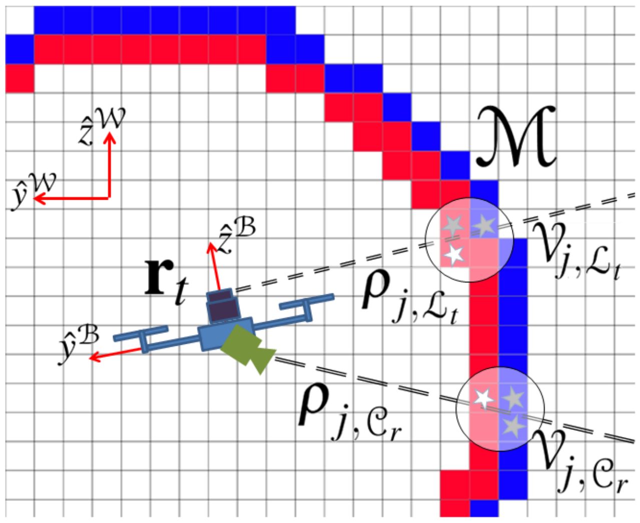
The proposed ICP algorithm performs data association by projecting each laser beam onto the grid map, M, and assigning the first hit voxel center to the corresponding beam. In other words, we define the closest map point to be the voxel center closest to the beam origin that intersects with the laser beam represented as

$$\begin{aligned} \mathcal{V}_{i,\mathcal{L}_j} &= \pi\left(\mathbf{x}, p_{\mathcal{L}_j,i}, M\right) \\ &= \underset{\mathcal{V}_k \in M,\, t \in \mathbb{R}^+}{\operatorname{argmin}} \left(\left|\left| \mathcal{O}^{\mathcal{M}}_{\mathcal{L}_j} + t\, \hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{L}_j} - \mathcal{V}_k \right|\right|_2 \right) \end{aligned} \tag{5.13}$$

$\mathbf{x}$ is the state vector, $j \in \{t, b\}$, $p_{i,\mathcal{L}_j}$ is the i^{th} pixel of the laser image from laser scanner $\mathcal{L}_j$, M is the 3D occupancy grid approximation of the map, $||\cdot||_2$ is the L_2 norm and

$$\mathcal{O}^{\mathcal{M}}_{\mathcal{L}_j} = \chi(\mathbf{x}_s) + {}^{\mathcal{M}}\mathbf{R}_{\mathcal{A}(\mathbf{x}_s)} \left(\begin{bmatrix} 0 \\ \mathbf{x}_{y,z} \end{bmatrix} + {}^{\mathcal{A}(\mathbf{x}_s)}\mathbf{R}_{\mathcal{B}}\, \mathcal{O}^{\mathcal{B}}_{\mathcal{L}_j} \right) \tag{5.14}$$

$$\hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{L}_j} = {}^{\mathcal{M}}\mathbf{R}_{\mathcal{A}(\mathbf{x}_s)}\; {}^{\mathcal{A}(\mathbf{x}_s)}\mathbf{R}_{\mathcal{B}}\; {}^{\mathcal{B}}\mathbf{R}_{\mathcal{L}_j}\; \hat{\mathbf{p}}_{i,\mathcal{L}_j} \tag{5.15}$$



$\mathcal{M}$
$\hat{z}^{\mathcal{W}}$
$\hat{y}^{\mathcal{W}}$
$\hat{z}^{\mathcal{B}}$
$\mathbf{r}_t$
$\boldsymbol{\rho}_{j,\mathcal{L}_t}$
$\mathcal{V}_{j,\mathcal{L}_t}$
$\hat{y}^{\mathcal{B}}$
$\boldsymbol{\rho}_{j,\mathcal{C}_r}$
$\mathcal{V}_{j,\mathcal{C}_r}$

Figure 5.2: This schematic depicts the parameters and vectors explained in Equ. 5.13-5.35. The bright stars represent the closest voxels among possible other that each pixel intersects.

are the sensor origin and pixel orientation both in the map frame. Fig. 5.2 depicts these parameters and vectors in a schematic.

Having the data association defined, we formulate the problem as an iterative weighted least squares problem without regularization written as $\mathbf{WA}x = \mathbf{Wb}$ where

$$x = [\cos(\Delta\mathbf{x}_\psi),\ \sin(\Delta\mathbf{x}_\psi),\ \Delta\mathbf{x}_y]^T, \tag{5.16}$$

$\mathbf{W}$ is the diagonal weight matrix, $\mathbf{A}$ is an $N \times 3$ matrix and $\mathbf{b}$ is an N vector where N is the total number of data points from both of the lidars. We exclude the roll and pitch angles from the solver assuming that the UKF estimates these accurately. Indeed, due to the symmetric tunnel geometry, absolute roll and pitch angles cannot be measured with range sensors. **A** and **b** are defined as

$$p_{i,\mathcal{L}_j}^{\mathcal{M}} = \mathcal{O}_{\mathcal{L}_j}^{\mathcal{M}} + \frac{1}{\rho_{i,\mathcal{L}_j}} \hat{\mathbf{p}}_{i,\mathcal{L}_j}^{\mathcal{M}} \tag{5.17}$$

$$\mathbf{A}_n = \left[\left\{ \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} p_{i,\mathcal{L}_j}^{\mathcal{M}} \right\}^{\top}, 1 \right] \tag{5.18}$$

$$\mathbf{b}_n = \left(\mathcal{V}_{i,\mathcal{L}_j}\right)_y - \mathbf{x}_y \tag{5.19}$$

$$\tag{5.20}$$

where the subscript n refers to the n^{th} row of the tensor and there is a one-to-one correspondence between each pixel from both sensors and tensor row indices. Each data point is assigned a weight such that $\mathbf{W}_{n,n} = w_n$ as a function of the alignment error ε_n. These parameters are defined as

$$\varepsilon_n = p_{i,\mathcal{L}_j}^{\mathcal{M}} - \mathcal{V}_{i,\mathcal{L}_j} \tag{5.21}$$

$$w_n = e^{-||\varepsilon_n||_2^{\gamma}} \tag{5.22}$$

where we choose $\gamma = 3$. This way, correspondences with large initial residuals are penalized more and lose their contribution to the least squares solution. Finally the partial solution becomes

$$x = (\mathbf{A}^{\top}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{W}\mathbf{b} \tag{5.23}$$

therefore

$$\Delta\mathbf{x}_y = x_3 \tag{5.24}$$

$$\Delta\mathbf{x}_{\psi} = atan2(x_2^*, x_1^*). \tag{5.25}$$

Since x_2 and x_1 might not be valid cosine and sine values, we clamp them to the $[-1, 1]$ inclusive range which is denoted by $\cdot^*$.

The above formulation solves only for $\mathbf{x}_y$ and $\mathbf{x}_\psi$ simultaneously because of their strong coupling. Whereas, due to the geometry of M and the way we formulate the least squares solution, $\mathbf{x}_z$ and $\mathbf{x}_\psi$ do not correlate significantly. $\mathbf{x}_z$ is mostly a function of the ranges from the bottom lidar. For each iteration of the pose refinement we define the $\mathbf{x}_z$ update as

$$\Delta\mathbf{x}_z = -\frac{1}{N}\left\{\sum_{n=1}^{N}(\varepsilon_n)_z \ \exp\left(\frac{\left|\left(p_{i,\mathcal{L}_j}^{\mathcal{M}}\right)_z\right|}{\left\|p_{i,\mathcal{L}_j}^{\mathcal{M}}\right\|_2} - 1\right)\right\}. \tag{5.26}$$

5.4.2 Uncertainty Estimation

Censi [53] models the 2D scan registration and its uncertainty estimation problems based on a polygonal environment assumption. This approach reduces the infinite dimensional map into a finite space. The unknown polygon is approximated by connecting ray tips of the sub-sampled scan data which is a fair heuristic with modern, accurate laser scanners. [53] approaches the uncertainty estimation problem from an information theoretic point of view and uses the Fisher information to calculate this. In particular, Cramer-Rao bound-like inequalities are used to statistically bound the pose covariance from below. A lower bound for the pose uncertainty is proven to be the inverse Fischer Information Matrix (FIM). In this work, we employ the same approach with improvements in the environment model for estimating the uncertainty in the partial state estimate given by the proposed ICP algorithm.

The information carried by each range measurement is summed to give the Fisher information of a single laser scan. The symmetric 3×3 FIM as defined in the source sensor frame is calculated as

$$\mathbf{I}\left(\mathcal{L}_j\right) = \sum_{i=0}^{|\{p_{\mathcal{L}j}\}|} \frac{1}{\sigma^2\cos^2(\beta_i)} \begin{bmatrix} \mathbf{a}(\alpha_i)\mathbf{a}(\alpha_i)^\top & \frac{1}{\rho_i}\sin(\beta_i)\mathbf{a}(\alpha_i) \\ * & \frac{1}{\rho_i^2}\sin^2(\beta_i) \end{bmatrix} \tag{5.27}$$

σ^2 is the variance of the additive Gaussian noise of each range measurement which we assume to be the same for each beam. The dependency of FIM on the environment geometry is encoded through the following variables: α_i is the surface (line) normal direction in the

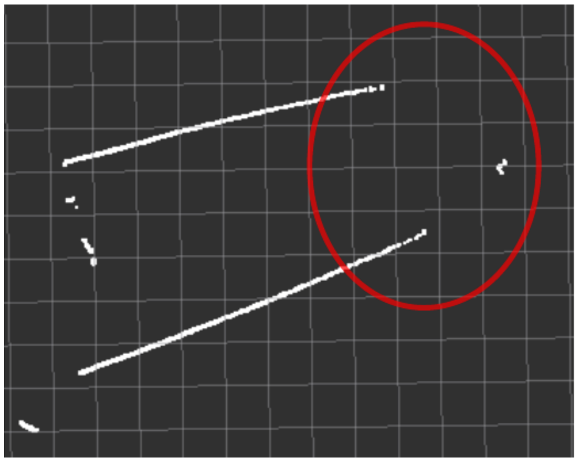


Figure 5.3: Sample laser scanner contour from inside a penstock at Carters Dam, GA. The two straight segments are from the walls of the tunnel. Since the laser scanner cannot see the end of the tunnel, contour interrupts (circled). A FIM is estimated as proposed in [53] separately for each segment and summed to give the measurement covariance.

sensor frame that the i^{th} ray intersects, $\mathbf{a}(\alpha_i) = [\cos(\alpha_i),\, \sin(\alpha_i)]^\top$ and β_i is the angle of incidence of the i^{th} ray with respect to the surface it intersects, *i.e.* the angle between the ray and the surface normal directions.

As shown in Fig. 5.3 assuming a single polygonal environment does not work for certain cases. For this reason, we first segment the scan into clusters according to the discrepancy between consecutive readings and represent the environment with *multiple polygons.* FIM is estimated for each polygon separately which are then summed to give a more accurate covariance estimate. The FIM for each laser scanner is defined as

$$\mathbf{I}\left(\mathcal{L}_j\right) = \begin{bmatrix} \mathbf{I}_{x,x} & \mathbf{I}_{x,y} & \mathbf{I}_{x,\psi} \\ \mathbf{I}_{y,x} & \mathbf{I}_{y,y} & \mathbf{I}_{y,\psi} \\ \mathbf{I}_{\psi,x} & \mathbf{I}_{\psi,y} & \mathbf{I}_{\psi,\psi} \end{bmatrix}. \tag{5.28}$$

written in the given laser frame hence each such information matrix should be transformed into a common frame before being summed.

Information from multiple lidars is merged by transforming each FIM into the body frame, $\mathcal{B}$. A FIM after transformation is denoted as $\mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{B}}$. The upper-left 2×2 block, *i.e.* the translational components, denoted as $\mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{L}_j}_{x,y}$, can be transformed simply by multiplying with proper rotation matrices. We expand this block to a 3×3 matrix for compatibility by appending the information for the z dimension which is 0. The FIM after transformation writes

$$\mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{B}}_{x,y,z} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{L}_j} \begin{bmatrix} \mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{L}_j}_{x,y} & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 0 \end{bmatrix} {}^{\mathcal{L}_j}\mathbf{R}_{\mathcal{B}}. \tag{5.29}$$

Angular uncertainties however are not easy to project, and finding a solution for this is beyond the concerns of this work. Therefore we approximate $\mathbf{I}^{\mathcal{B}}_{\psi,\psi}$ as

$$\mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{B}}_{\psi,\psi} = \left| \left({}^{\mathcal{M}}\mathbf{R}_{\mathcal{L}_j}\right)_{(3,3)} \right| \mathbf{I}\left(\mathcal{L}_j\right)^{\mathcal{L}_j}_{\psi,\psi} \tag{5.30}$$

where $\cdot_{(3,3)}$ is the bottom-right element of the rotation matrix and it is a measure of how much the laser scanner is tilted with respect to the target frame. The off-diagonal elements for the ψ information are ignored which does not affect the performance of the estimator significantly.

The measurement uncertainty of the ICP is approximated as the inverse of the total information from both laser and is written as

$$\mathbf{\Sigma}\left(\mathcal{L}\right)^{\mathcal{B}}_{x,y,z} = \left(\mathbf{I}\left(\mathcal{L}_t\right)^{\mathcal{B}}_{x,y,z} + \mathbf{I}\left(\mathcal{L}_b\right)^{\mathcal{B}}_{x,y,z}\right)^{-1} \tag{5.31}$$

$$\mathbf{\Sigma}\left(\mathcal{L}\right)^{\mathcal{B}}_{\psi} = \left(\mathbf{I}\left(\mathcal{L}_t\right)^{\mathcal{B}}_{\psi} + \mathbf{I}\left(\mathcal{L}_b\right)^{\mathcal{B}}_{\psi}\right)^{-1}. \tag{5.32}$$

Equ. 5.5 requires the position measurements to be given in the local axis frame. This conversion can be performed as

$$\mathbf{\Sigma}\left(\mathcal{L}\right)^{\mathcal{A}(\mathbf{x}_s)}_{y,z} = \mathbf{P}\ \left({}^{\mathcal{A}(\mathbf{x}_s)}\mathbf{R}_{\mathcal{B}}\right)\ \left(\mathbf{\Sigma}\left(\mathcal{L}\right)^{\mathcal{B}}_{x,y,z}\right)\ \left({}^{\mathcal{B}}\mathbf{R}_{\mathcal{A}(\mathbf{x}_s)}\right)\ \mathbf{P}^{\top} \tag{5.33}$$

$$\mathbf{P} = \left[\mathbf{0}_{2\times 1},\ \mathbf{I}_{2\times 2}\right]. \tag{5.34}$$

5.5 Vision-Based Axial Speed Estimation

This section gives the details of the visual odometry (VO) algorithm that we employ for estimating the axial speed of the robot. By integrating over time, axial speed can be used to estimate the only unknown state variable, the axial position, $\mathbf{x}_s$. Secondly, an uncertainty estimate of the axial speed is also provided which is a requirement of the filtering framework. The two other major topics of this section are the image enhancement and panoramic image generation for offline inspection purposes.

The proposed algorithm is explained assuming only one of the cameras is active. However, this method can easily be adapted to multi-camera scenarios by simply redefining the corresponding measurement model of the Kalman filter. The major reason for using a single camera although the platform is equipped with four, is lack of sufficient bandwidth to grab frames from all cameras at sufficiently high frame-rates. On the other hand, the panoramic image generation algorithm stitches all four images collected in a separate experiment where the robot was flown with only the range-based estimation active.

Similar to the previous section, the extrinsic calibration of each camera is represented by rotation-translation pairs, *i.e.* $\left\{ {}^{\mathcal{B}}\mathbf{R}_{\mathcal{C}_i}, \mathcal{O}^{\mathcal{B}}_{\mathcal{C}_i} \right\}$ where $i \in \{r, l, t, b\}$. Fig. 3.10 illustrates the sensor configuration.

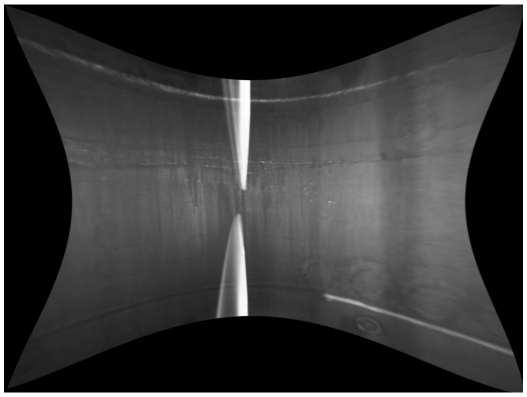
5.5.1 Full State vs Axial Speed Estimation

The rest of this section details the image enhancement, feature extraction, tracking and, finally, axial speed estimation steps. Each step of the estimation pipeline introduces errors into the process such as inaccurate image feature coordinates and tracking. The performance of imagers in low light settings available at the development time of the approach explained in this chapter was a limiting factor affecting the image quality, hence the whole visual odometry pipeline. The errors inevitably introduced at each step of the proposed pipeline is amplified in proportion to the performance of the onboard cameras in low light conditions. Output of each step of the pipeline shown in Fig. 5.5 suffices to prove the severity of the situation. This raises concerns about the reliability of cameras as a major source of measurement for a state estimation framework deployed on a MAV flying inside a confined penstock setting. On the other hand, laser scanners are not affected by the adverse and challenging conditions in penstocks. Due to these concerns, the approach presented in this chapter prefers using laser scanners as the major source of information for state estimation, and uses cameras only in the estimation of just a single degree of freedom, *i.e.* axial speed, failure in accurate estimation of which does not result in catastrophic failure of the state estimation.

On the other hand, a platform equipped with more powerful onboard illumination and cameras with better performance in low light conditions can be deployed with an estimator that relies more on visual information. In particular, a loosely-coupled range-visual odometry framework which adaptively fuses information from both type of sensors depending on the image quality could perform better than the proposed method. Open source visual odometry frameworks such as [106], [107], [121] could be used for this purpose. Also, performance of these can be enhanced when used in combination with range sensors since depth of pixels/features that overlap with the field of view of laser scanners can be estimated without any latency.

5.5.2 Image Enhancement

The accuracy of the proposed VO method heavily depends on the quality of the onboard illumination since there is no external illumination in a typical penstock. As shown in Fig. 5.4 the images have very low contrast and weak texture. In our tests, none of the feature extraction implementations of the OpenCV library such as FAST, Harris and Shi-Tomasi [51], [62] could find any features or they fail persistence. Furthermore, the nonuniform lighting pattern such as concentric brightness rings and lens flare generate artificial intensity gradient which adversely affects both feature extraction and tracking performance. We overcome these problems by applying a set of image filters as shown in Fig. 5.5 to amplify


(a) An image grabbed using one of the side cameras. The white blades are the propellers partially occluding the camera view. The white line at the bottom-right corner is painted for manual ground truth comparison (Sec. 5.8). Note that this image is rotated by 90 degrees.

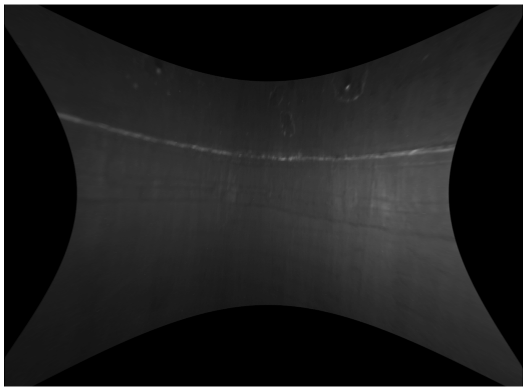

(b) An image grabbed by the top camera. The white line is the welding seam line. Since the robot is flying below the center line level, the LED fall short in illuminating the ceiling sufficiently resulting in a dark, low contrast image.

Figure 5.4: These two frames grabbed using the onboard cameras show the typical level of image detail.

the weak texture and suppress the effect of intensity pattern artifacts.

In order to enhance the contrast we apply histogram equalization followed by Gaussian smoothing to suppress local abrupt changes introduced by stretching the image histogram. The resultant image has higher contrast hence stronger gradients which is a necessity for robust feature extraction and tracking. Then we perform adaptive thresholding on this image with a Gaussian kernel. A pixel is set white if its intensity is greater than the weighted average of its neighboring pixels. This methods outputs a binary image (Fig. 5.6). An interesting property of this binary image is that white pixels adhere to each other and form small blobs which persist for a couple of frames. These blobs define small hills and the black regions form valleys around that these which Kanade-Lucas-Tomasi (KLT) tracker [20] can easily track. Sample binary frames and raw images with optical flow fields overlaid are shown in Fig. 5.7. The binary image exhibits this behavior due to the speckled tunnel surface caused by rust formation, peeled-off surface coating, rivets and irregular painting. Despite the short blob adhesion periods of just a couple of frames, sufficient optical flow field density is attained thanks to the new blobs forming at every frame. In some cases, only a small subset of the image exhibit sufficient blob quality and persistence. This happens

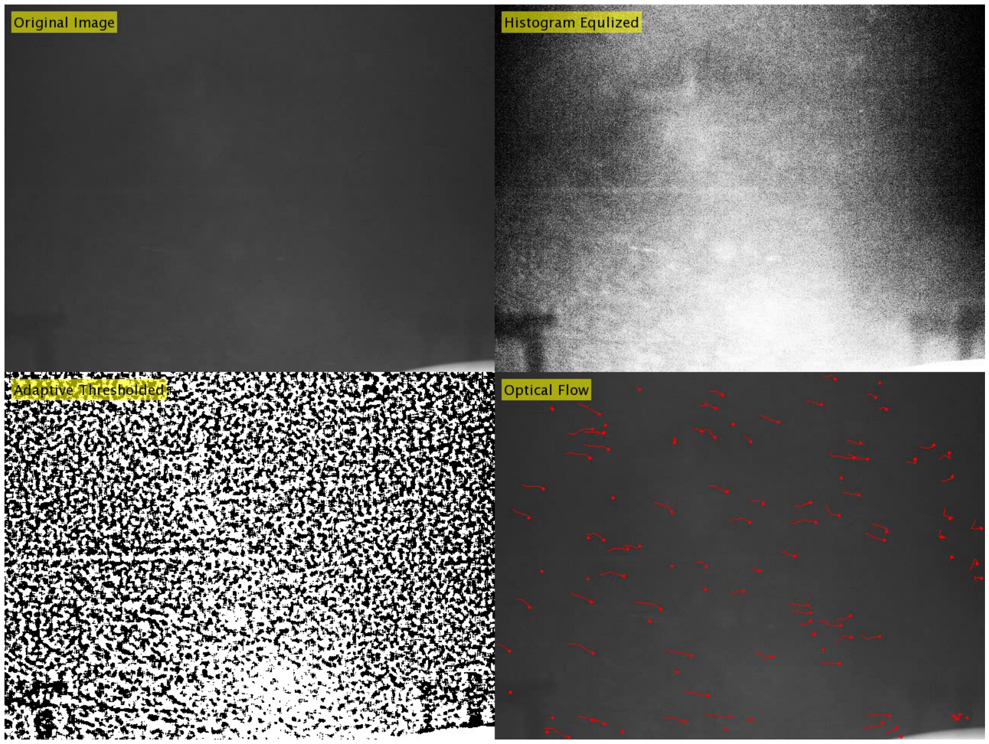


Figure 5.5: This figure shows the output of the image processing pipeline at each step and the resultant optical flow field from our visit to Carters Dam, GA. At the top-left is the raw image. This is a pale image with almost no significant texture. In order to amplify the texture gradient, we used histogram equalization as shown on the top-right. Next, an adaptive threshold is applied to get a black-white image as in the bottom-left. FAST features and KLT tracker are used to extract and track features on this image. The bottom-right image shows the tracked feature points and their corresponding optical flow trails after initial outlier elimination. The corresponding video can be found at http://mrsl.grasp.upenn.edu/tolga/iros2016/

when the tunnel surface is clean, the coating is uniform and well maintained. In such cases, we set the region of interest to image patches with higher average brightness to increase the effect of histogram equalization.

5.5.3 Feature Extraction and Tracking

The spatial distribution of the feature points extracted from the final image shape according to the saliency of the image regions. This in turn results in clustering of the features around high texture regions which is not preferable since this would cause singularity in the pose estimation. This effect can be mitigated by means of a method called bucketing [11], [73].

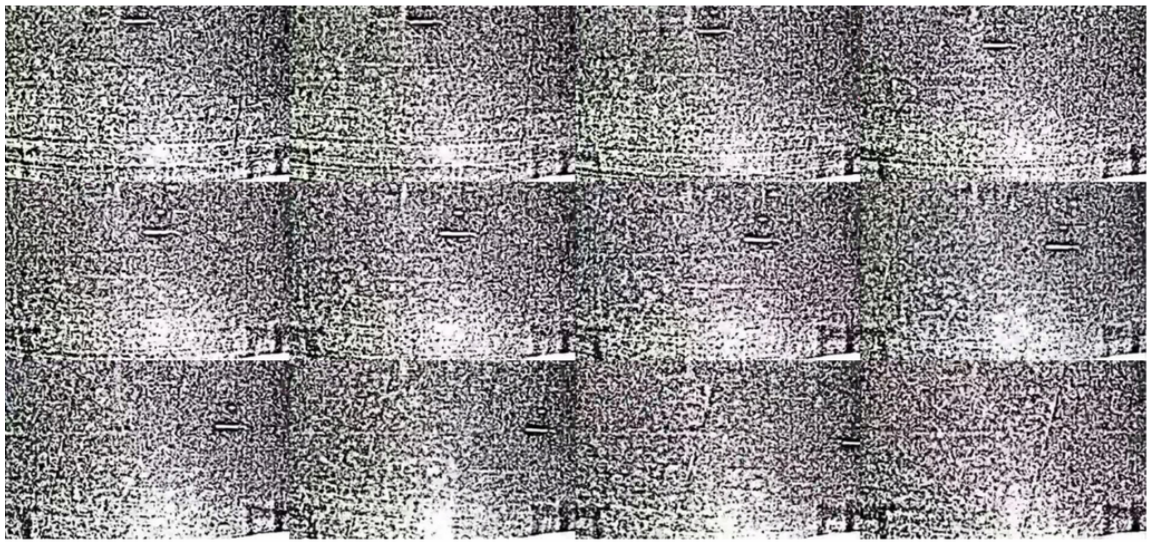


Figure 5.6: This figure shows consecutive binary images obtained after the adaptive thresholding stage of the image processing pipeline. It can be observed that the effect of irregular illumination is partially recovered. Due to the low resolution of the images, it is difficult for a human to track blobs across frames. However a few blobs due to scratches and spots preserve their shapes which can be easily observed. The KLT tracker can track even the smaller blobs for 3-5 frames which is sufficient for VO.

The image is divided into nonoverlapping rectangular patches and the maximum number of features in each bucket is limited. In this way, the resultant feature point set is more uniformly distributed over the whole image

Among the benefits of this approach is the reduction in the number of features while keeping the most salient ones. This helps reducing the CPU load which is a requirement for real-time performance. Secondly, a uniformly distributed set of features can capture the actual optical flow field much better compared to the case where feature points are clustered around a relatively small, high texture image region. Lastly, the adverse effect of intensity artifacts due to nonuniform illumination is mitigated. Without bucketing, a feature detector would extract most features along the artificial intensity gradient.

Due to its efficiency, we use FAST features [75]. The feature extractor maintains a constant number of features while guaranteeing a certain minimum separation between features. We keep this value in 3-7 pixels range. Existing features are tracked using the KLT tracker for temporal data association. We prefer the KLT tracker over descriptor based feature

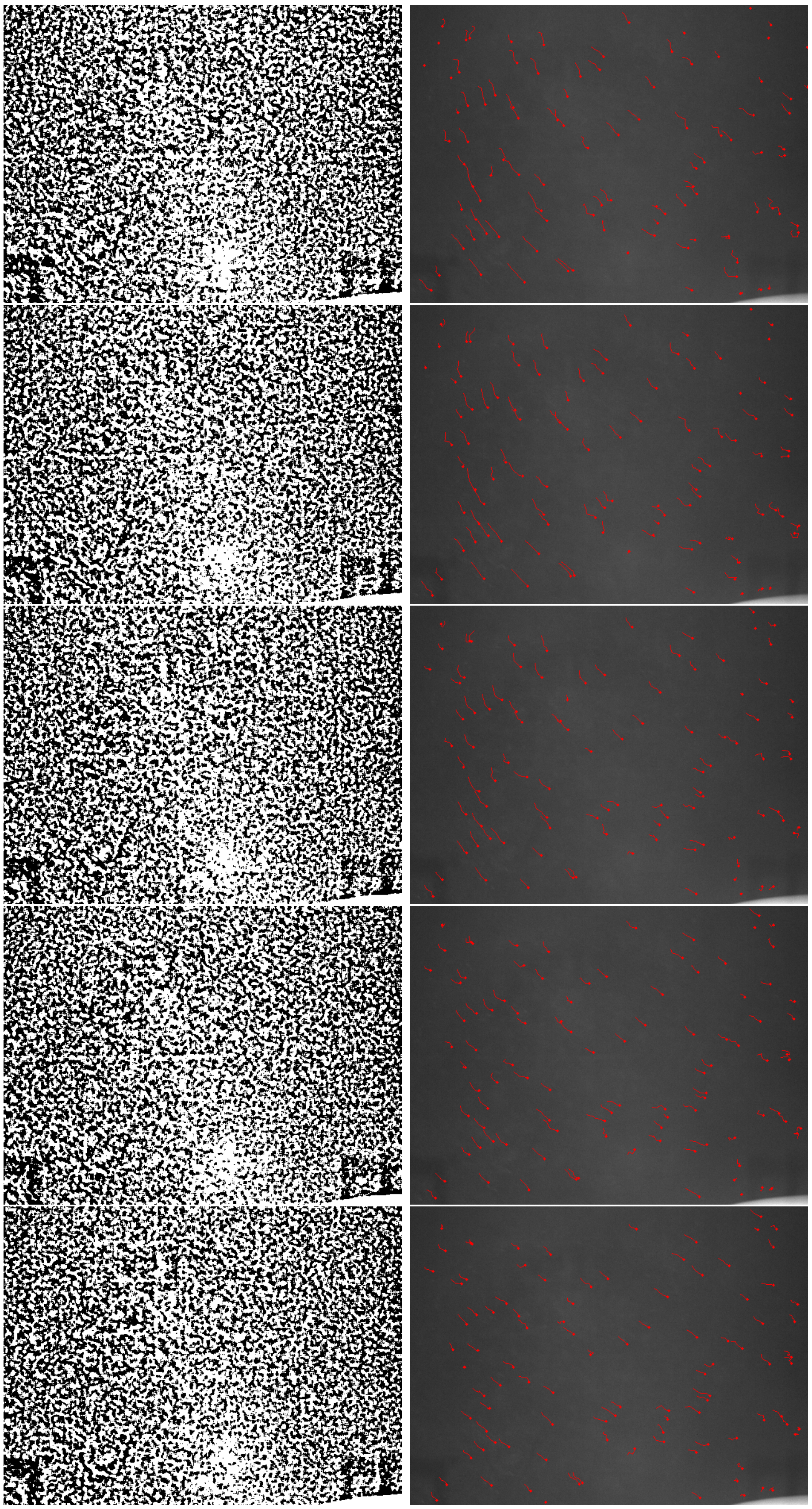


Figure 5.7: Sample binary images and optical flow fields.

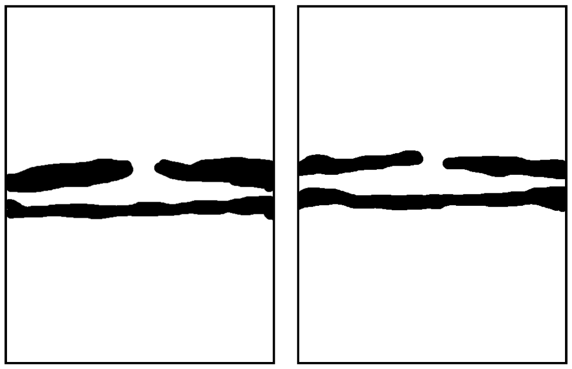


Figure 5.8: Masks applied to the images captured from the right and left cameras. The half strips cover the image areas corresponding to the propellers. When the robot is flying close to the centerline vertically, the side cameras are exposed to specular reflection from the wet tunnel surfaces which degrades the feature extractor and tracker performance. The full strips mask out these image regions.

matching approaches since the latter requires descriptor extraction and matching both of which induce more CPU usage. According to our tests, the former approach performs well enough for optical flow vectors a few tens of pixels long which is satisfied with moderate flight speeds (~ 2 m/s) and camera high frame rates (> 20 fps). History of each feature position is kept for a certain maximum number of frames which typically ranges between 3 and 7.

The propellers are partially visible to side cameras as shown in Fig. 5.4a. The edges of the propellers offer high intensity gradient which causes the feature extractor to detect most features in the corresponding bucket around the propeller edges. In order to prevent this, we mask the image from the side cameras with masks as shown in Fig. 5.8

We employ three types of outlier rejection procedures. Only the features that are tracked successfully for more than a given number of frames are used. Secondly, features with optical flow vectors longer than a threshold are rejected. These usually correspond to cases where KLT tracker fails to track a feature and ends up in abrupt jumps in the feature location. The typical threshold is between 30 and 100 pixels depending on the flight speed. Lastly, we apply an adaptive thresholding on the optical flow vector lengths. The mean flow vector and its variance are calculated which are then used to eliminate flow vectors that are farther from this model than a certain number of standard deviations.

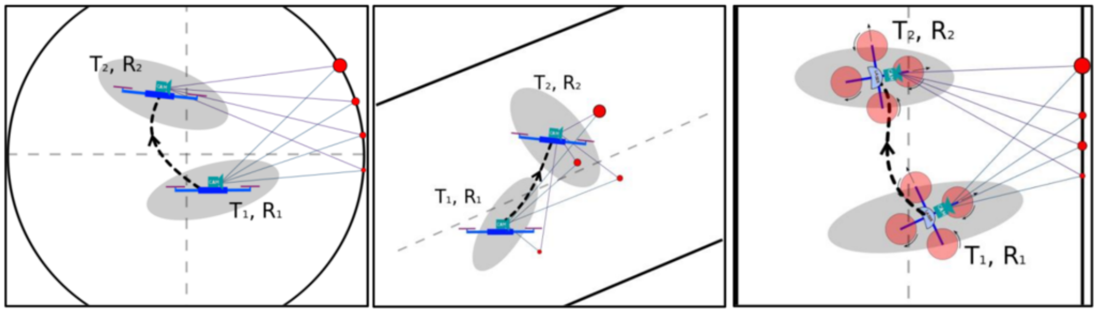


Figure 5.9: These figures show snapshots from three perspectives of the two-step VO displacement estimation. From left to right are back, right and top views of the tunnel along the inclined section. The robot poses are denoted as rotation-translation pairs (R_i, T_i) $i \in \{1, 2\}$. Grey shades represent the pose uncertainty. The red dots are the back-projected feature points. The range-based localizer can only provide lateral and vertical position estimates. The missing DoF is estimated using VO.

5.6 Visual Odometry

Position measurement using a laser scanner is, in most cases, much more reliable and accurate compared to a camera since vision-based position estimation is possible only after observing a scene for multiple frames while moving along a path that lends to sufficient parallax. Furthermore, a visual odometry pipeline involves highly non-linear optimization procedures or linearization of systems of equations which when not implemented robustly or fed with inaccurate or faulty measurements may diverge and fail catastrophically. Particular to our case, provision of reliable measurements to the VO pipeline is maybe the most fragile component due to factors such as low texture surfaces, insufficient illumination and occlusion by dust particles and mist which degrade the image quality. Pursuant to safety concerns in the confined penstock environment, we designed the VO pipeline to estimate only the axial speed and rely on the partial state estimates from the laser scanners as its initialization point.

The epipolar constraint [15] provides a mathematical relation for estimating relative orientation and translation *direction* using solely 2D image features. Successive application of this method along with estimating the 3D positions of the landmarks corresponding to the image features is central to most VO systems [32], [33], [83]. In case 3D position estimates of the landmarks are available, the unknown translational scale can also be recovered using the family of Perspective-n-Point (PnP) algorithms [27]. The knowledge of pixel metric depth facilitates the pose estimation problem making it less sensitive to inaccurate measurements. In this work, we adopt the latter approach since a partial state estimate of the robot is available which, in combination with the map, can be used to reliably estimate feature depths.

We estimate the inter-frame displacement along the tunnel axis by analyzing the back-projection of the tracked features onto M extracted from consecutive frames pairs. This operation is similar to the ray-casting function $\pi(\cdot)$ defined for the range-based localizer with minor modifications to improve accuracy. Features in the 1st and 2nd frames are referenced by $\left\{p_{\mathcal{C}_j}\right\}^1$ and $\left\{p_{\mathcal{C}_j}\right\}^2$ respectively. The 2nd frame is the most recently grabbed frame and the 1st frame is the previous one. Note that, since laser scanners and cameras are not working with the same rate, we do not use a common time index. i is the tracked feature index in the camera with frame $\mathcal{C}_j$ where $j \in \{r, t, l, b\}$. In fact, due to low USB 2 bandwidth, we use only the right camera. The 3D projected point on M for each feature is found as

$$\begin{aligned}\mathcal{V}_{i,\mathcal{C}_j} &= \pi\left(\mathbf{x}, p_{i,\mathcal{C}_j}, M\right) && (5.35)\\ &= \underset{\mathcal{V}_k \in M,\, t \in \mathbb{R}^+}{\operatorname{argmin}} \left(\left|\left| \mathcal{O}^{\mathcal{M}}_{\mathcal{C}_j} + t\, \hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{C}_j} - \mathcal{V}_k \right|\right|_2 \right). && (5.36)\end{aligned}$$

The behavior of the $\pi(\cdot)$ function and the definition of the other variables are similar to those in Equ. 5.13. As shown in the system architecture (Fig. 5.1), the pose at which the $\pi(\cdot)$ is applied is fed from the UKF which is denoted as $\mathbf{x}$. $\hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{C}_j}$ is the unit vector corresponding to the image feature written in the map frame as

$$\hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{C}_j} = \gamma\ {}^{\mathcal{M}}\mathbf{R}_{\mathcal{C}_j}\ \mathbf{K}^{-1} \begin{bmatrix} \left(p_{i,\mathcal{C}_j}\right)_x \\ \left(p_{i,\mathcal{C}_j}\right)_y \\ 1 \end{bmatrix} \tag{5.37}$$

which is also depicted in Fig. 5.2. Here $\gamma \in \mathbb{R}^+$ is the normalization factor, $\mathbf{K}$ is the camera calibration matrix and the top two elements of the 3-vector are the coordinates of the corresponding image feature.

Since M is a finite resolution occupancy grid, two back-projected feature points at different but close image coordinates may be matched to the same voxel. Thus, direct use of the output of the projection function loses the precision required for accurate estimation of the *continuous* robot pose. In order to mitigate this problem we fix $\mathcal{V}_{i,\mathcal{C}_j}$ as follows

$$\mathcal{V}^{*}_{i,\mathcal{C}_j} = \left|\left|\mathcal{V}_{i,\mathcal{C}_j} - \mathcal{O}^{\mathcal{M}}_{\mathcal{C}_j}\right|\right|_2 \hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{C}_j} + \mathcal{O}^{\mathcal{M}}_{\mathcal{C}_j}. \tag{5.38}$$

This operation basically fixes the orientation of the corresponding 3D point to that of the high resolution image feature, *i.e.* $\hat{\mathbf{p}}^{\mathcal{M}}_{i,\mathcal{C}_j}$, and sets the depth of the feature, *i.e.* $\rho^{-1}_{i,\mathcal{C}_j}$, to the distance between the camera origin and the initially matched voxel center, $\mathcal{V}_{i,\mathcal{C}_j}$. Since the voxel size is small ([3-5] cm) and the tunnel surface is smooth, we assume the change in the depth of a feature is insignificant after the correction. Fig. 5.9 shows an instance of the robot from three perspectives with the back-projected rays.

The direction of the motion unobservable to the range-based localizer can be inferred from the state vector as $\gamma\frac{d\chi(\mathbf{x}_s)}{d\mathbf{x}_s} = \mathcal{A}(\mathbf{x}_s)^{\mathcal{M}}_{\hat{\mathbf{x}}}$ where $\gamma \in \mathbb{R}^+$ is the normalization factor. For brevity, we will denote this vector as $\hat{\boldsymbol{\alpha}}(\mathbf{x}_s)$. This motion can be estimated on a frame-by-frame basis by analyzing the difference between the back-projected vectors of consecutive frames. Each such vector is defined as

$$\mathbf{e}_{i,\mathcal{C}_j} = \left(\mathcal{V}^{*2}_{i,\mathcal{C}_j} - \mathcal{V}^{*1}_{i,\mathcal{C}_j}\right) \hat{\boldsymbol{\alpha}}(\mathbf{x}_s)^{\top} \hat{\boldsymbol{\alpha}}(\mathbf{x}_s) \tag{5.39}$$

where the superscript numbers denote the frames numbers. The set of error vectors is denoted with the list notation as $\{\mathbf{e}_{\mathcal{C}_j}\}$.

Binarization of the raw images at the preprocessing stage degrades the *accuracy* and *locality* of the features which are listed among the properties of the ideal local feature by [67]. As seen in Fig. 5.6-5.7, at each frame a random subset of blobs either merge with its neighboring blobs or split into smaller ones in the next frame. This results in false optical flow vector estimation with the vector norms differing by multiples of a typical blob diameter. The noise with such characteristics can easily be filtered by a histogram-based filter. A histogram of the projection errors, $\left\{ \left|\left|\mathbf{e}_{\mathcal{C}_j}\right|\right|_2 \right\}$, is generated with 5 bins maximum. Features in the highest percentage bin are regarded as inlier. The axial speed measurement, $\mathbf{z}^{\mathcal{C}}$, is estimated as the mean of the inliers in this bin which is then fed into the UKF. Finally, the measurement uncertainty is estimated as the variance of the same set of projections errors.

5.7 Panoramic Image Generation

In a typical image processing pipeline for panoramic image generation, features with descriptors are extracted from images which are then used to find an initial alignment of adjacent images pairs. However, these steps are not required if the relative camera poses and scene depth is known. In out case, the external calibration of the cameras as well as the relative pose of the cylindrical structure around the cameras are known thanks to the *a prior* map the pose estimate. A panoramic image is obtained simply by back-projecting images from all cameras onto the surface of the tunnel and then projecting them back onto a hypothetical cylindrical imager.

A cylindrical imager with square pixels is modeled by its circumferential and axial resolution, *i.e.* $c \times a$. The azimuthal FOV (around the axis) of this camera is $FOV_\theta = 360^o$ and its elevation FOV can be found as

$$FOV_\phi = 2\ atan\left(\frac{a\pi}{c}\right). \tag{5.40}$$

We assume that the cylindrical images is fixed to the body frame at its origin, $\mathcal{B}_{\mathcal{O}}$, and its axis is aligned with $\mathcal{B}_{\hat{\mathbf{x}}}$. The origins of azimuthal and elevation coordinates are respectively coincident with $\mathcal{B}_{\hat{\mathbf{z}}}$ and the base of the cylinder at its $-\mathcal{B}_{\hat{\mathbf{x}}}$ end. The camera function,

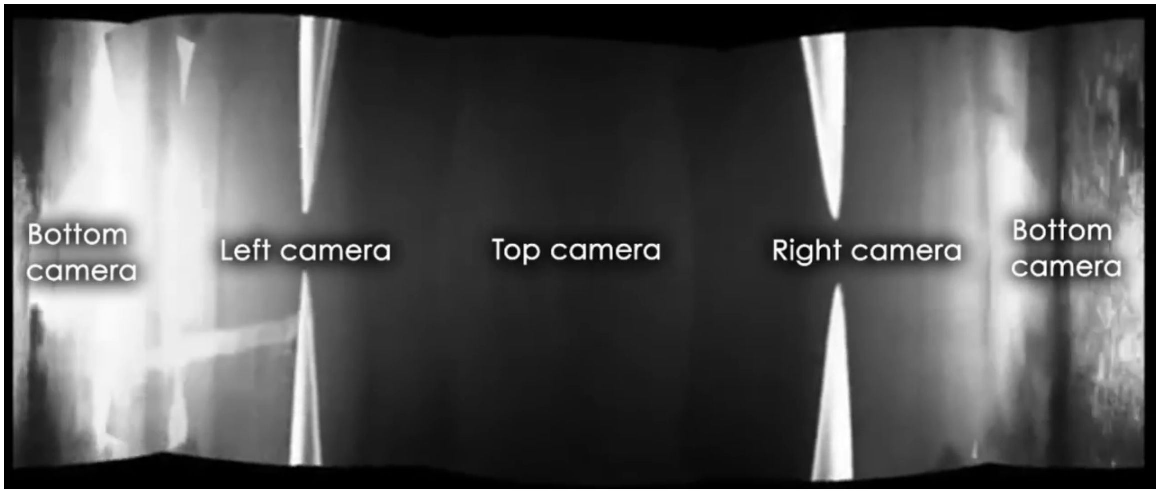


Figure 5.10: This figure shows the 360 degrees panoramic image reconstruction obtained using images from the four onboard cameras. Regions of the panoramic image is labels with the source camera.

analogous to the camera matrix, is defined as

$$\mathbf{K}(\mathbf{p}) = \begin{pmatrix} atan2\left(\mathbf{p}_y, \mathbf{p}_z\right) \\ \frac{\mathbf{p}_x}{\sqrt{\mathbf{p}_y^2+\mathbf{p}_z^2}}\frac{c}{2\pi} + \frac{a}{2} \end{pmatrix} \tag{5.41}$$

where $\mathbf{p}$ is a 3-vector given in $\mathcal{B}$, and $c/2\pi$ is the cylinder radius in pixels.

The cylindrical image is constructed by projecting each pixel from all four cameras onto the cylindrical images surface and blending their intensities with equal alpha values. Depth of each pixel is estimated as explained in Equ. 5.38. The projection of a real pixel onto the panoramic image is then calculated as

$$\mathbf{K}\left(\rho_{i,\mathcal{C}_j}^{-1}\hat{\mathbf{p}}_{i,\mathcal{C}_j}^{\mathcal{B}}\right). \tag{5.42}$$

Fig. 5.10 shows a panoramic image generated using the images grabbed from the four on-board during the experiments at Glen Canyon Dam. A 3D color point cloud obtained by back-projecting the raw images onto the tunnel walls is shown in Fig. 5.11.

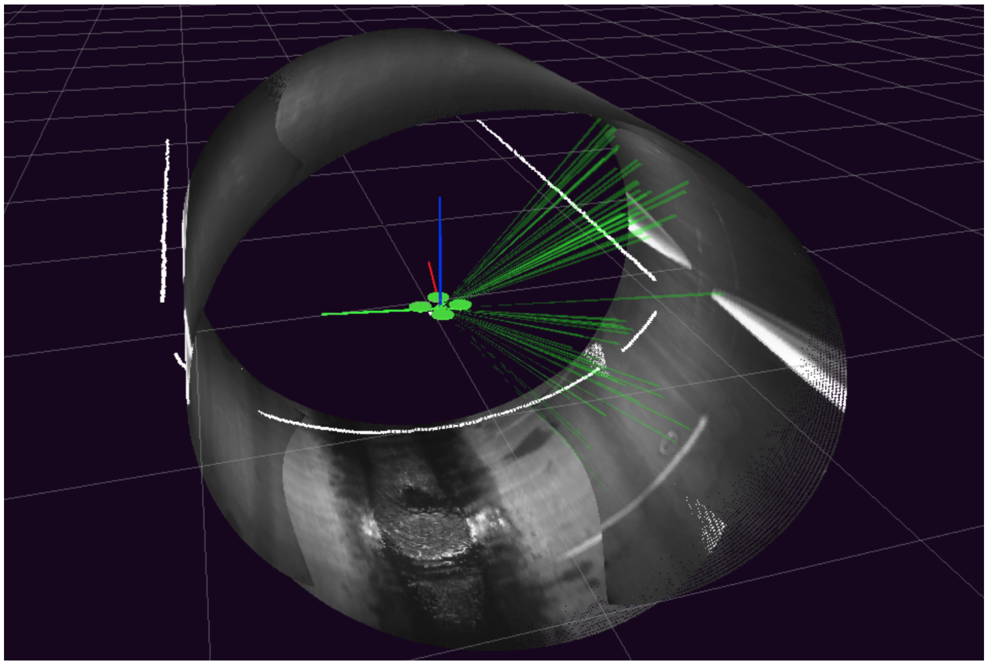
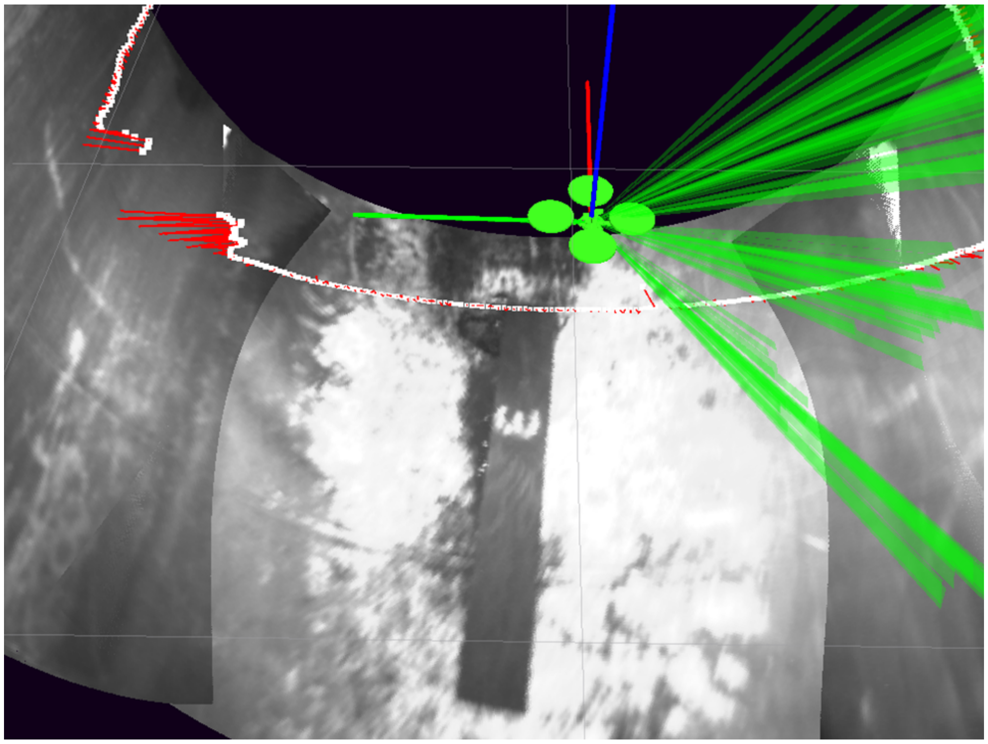



Figure 5.11: These images show 3D color point clouds plotted in RViz obtained by back-projecting images from the four onboard cameras onto the tunnel walls. The water drainage, propellers and peeled off coating can be clearly seen.

5.8 Experimental Results

In this section we present results from experiments conducted in penstocks at Carters Dam, GA and Glen Canyon Dam, AZ. Results we present here include estimation results with ground truth comparisons when possible such as in Fig. 5.14. In particular, at Glen Canyon Dam, AZ, we marked the walls vertically with spray paint with 2 meters of separation visible from the camera which we used for ground truth comparison. We collected this dataset while the robot was flying along the horizontal section as can be seen in Fig. 5.12. The position estimates along the tunnel axis are manually recorded at instants when the right side camera sees the markings right at the center of the images. The displacements are then compared against the ground truth 2 meters separation between markings. The time axis refers to instants the painted markers are seen right at the center of the right camera. Videos related to this experiment can be found at http://mrsl.grasp.upenn.edu/tolga/iros2016/

In Fig. 5.13, we show the lateral and vertical position estimates along with their inflated variances. Throughout the flights we commanded the robot to follow a straight line parallel to the centerline which can also be observed in these figures. The oscillations in these graphs are due to non-optimal controller parameter values.

In other three datasets collected in a penstock at Carters Dam, GA, the robot flew along the inclination. Since the steep inclination prohibits climbing along this section of the tunnel, we could not follow a similar approach as in the previous case for ground truth comparison. Instead, we assessed the visual odometry performance by observing scratches on the walls. Fig. 5.15-5.16 show 3D position estimation results from the Carters Dam experiment. We particularly focus on the VO results in these figures. In order to assess the performance of VO, we manually match scratches on the wall at the start and end of each flight, and also at intermediate locations if possible. In the first experiment (Fig. 5.15) the drift after a ~ 40 meter flight is less than 1 meter which corresponds to $< 2.5\%$ error. In this particular dataset, we could manually detect loop closured and confirmed that the drift is small also at intermediate points.

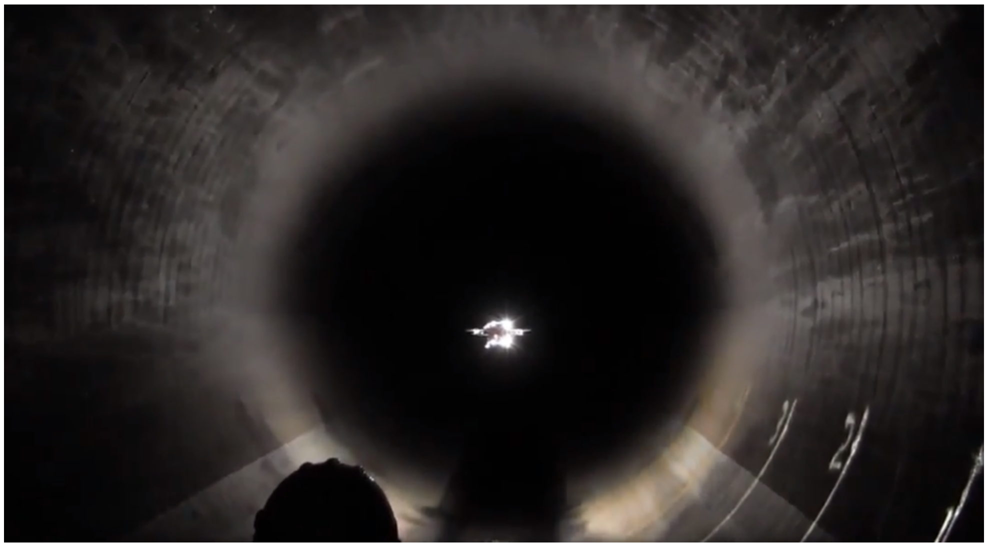
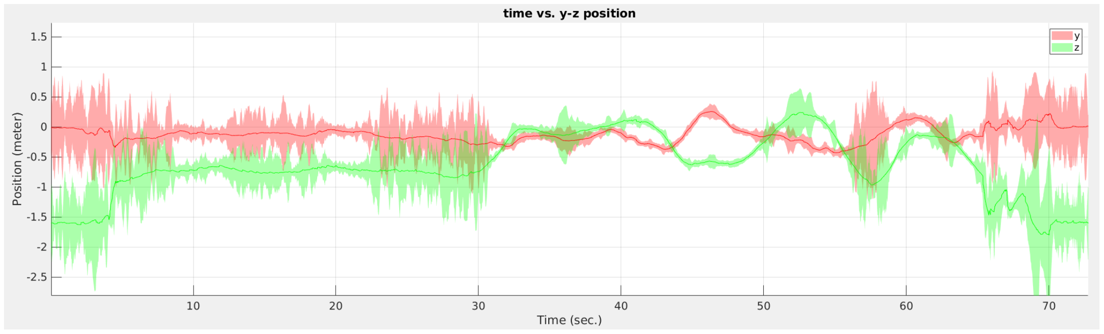
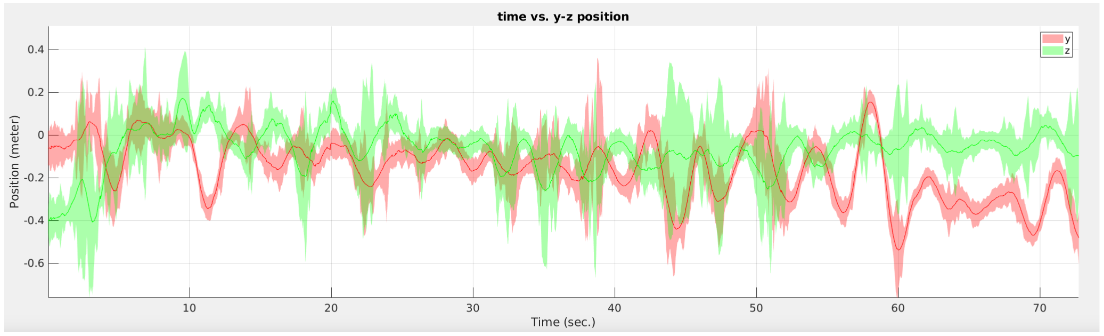


Figure 5.12: A snapshot from the experiments inside a penstock at Glen Canyon Dam, AZ. The robot is flying fully autonomously using onboard illumination. Also, in Fig. 5.11 and Fig. 5.10, we show the local 3D reconstruction and the 360 degrees panoramic image generated using the images from the onboard four cameras.

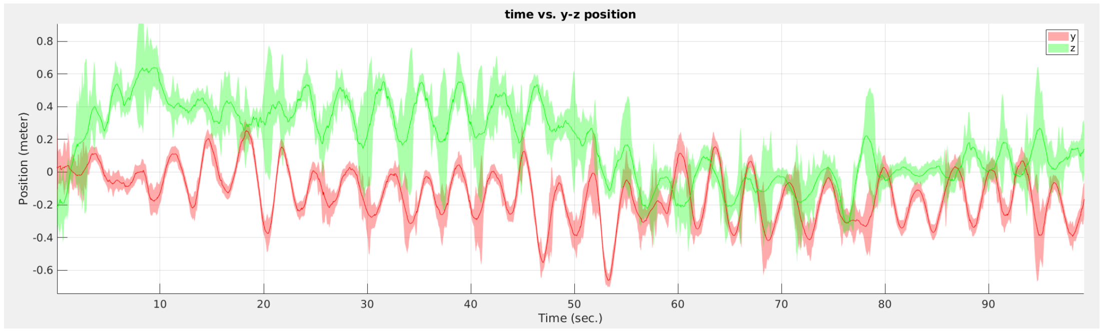
The second dataset (Fig. 5.16a) shows $\sim$ 7.5 meters of drift at the end of the flight which is significantly larger compared to the first test. It should be noted that starting at $\sim 65^{th}$ second, the axial position estimate oscillates for more than 20 seconds. During this flight, when the robot reached at the highest elevation (at $\sim 65^{th}$ second), it started to yaw randomly which was due a failure in the yaw estimation. The yawing motion in turn induced large optical flow vectors. However, since the range-based state estimate was not correct, the rotation-induced optical flow field was not compensated properly which resulted in very larger axial velocity estimates. Lastly, the results for the third dataset (Fig. 5.16b) are similar to the first dataset in terms of drift and the estimated trajectory.



(a) $y - z$ position estimates from experiment #1.

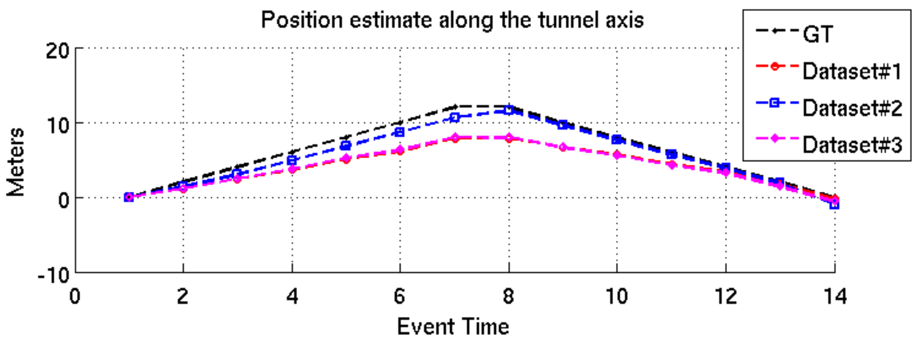


(b) $y - z$ position estimates from experiment #2.

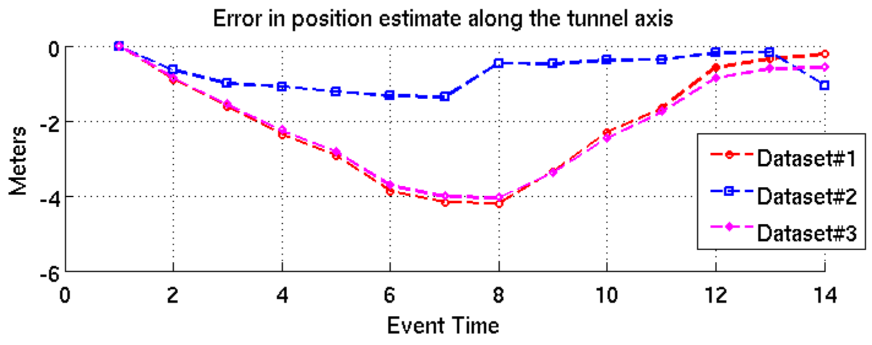


(c) $y - z$ position estimates from experiment #3.

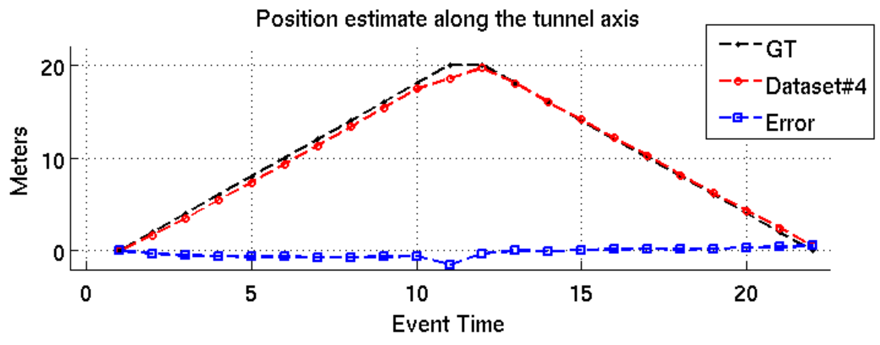
Figure 5.13: These plots show the lateral and vertical position estimates of the robot along with their corresponding inflated variances. During these flights, the robot was commanded to follow a straight path at a constant distant from the centerline. The oscillations are due to suboptimal onboard controller parameters.



(a) Position estimate along the tunnel axis for experiment #1.

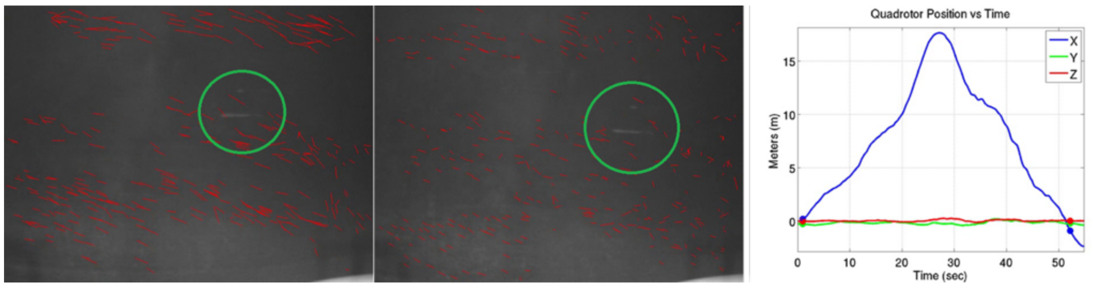


(b) Position estimation *errors* along the tunnel axis for experiment #2.

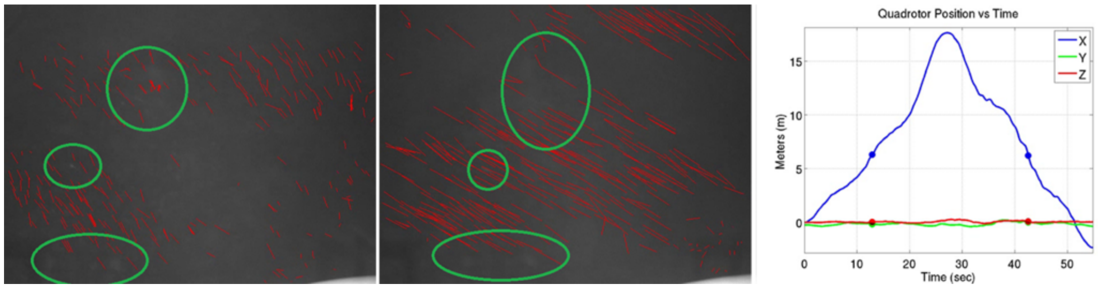


(c) Position estimate and error along the tunnel axis for experiment #3.

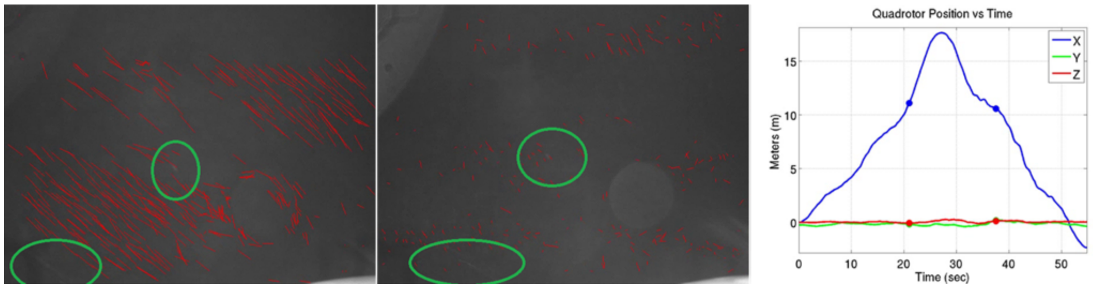
Figure 5.14: These plots compare the VO results with ground truth data on datasets collected in a penstock at Glen Canyon Dam, AZ. The x-axis enumerates the instants that the markers on the walls are seen at the center of the right camera.



(a) A comparison of axial position estimates at the take off position. We deduce that the robot revisits the same position from the horizontal white scratches circled in green.

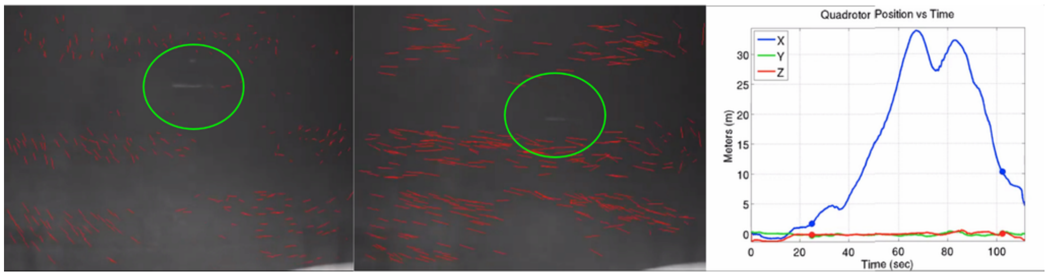


(b) The robot is at the $\sim 7^{th}$ meter from the take off position. The axial position estimates differ only by a few centimeters.

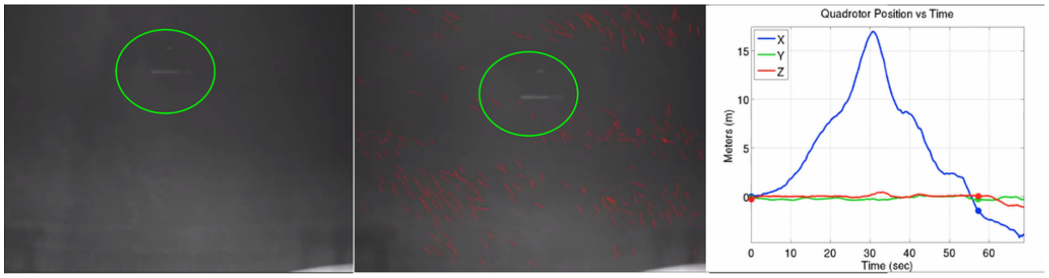


(c) The robot is at the $\sim 11^{th}$ meter from the take off position. The axial position estimates differ only by a few centimeters. Note how lens flare distorts the optical flow field.

Figure 5.15: These figure show 3D position estimation results on a dataset collected along the inclined section of a penstock at Carters Dam, GA. $y - z$ positions are estimated by the range-based estimator, and x (axial position) is estimated using visual odometry. The left and middle images show the flow field overlaid on the camera view. Green circles focus on the scratches on the wall which we use for manual loop closure. The two blue dots on the plot show axial position estimates at the loop closure. The drift along the ~ 40 meter flight is < 1 meter.



(a) In this test, around the 65th second, the yaw estimation failed for a couple of second causing the robot to oscillate. This motion in turn induced large optical flow vectors. Since the orientation estimate was wrong, during this period the optical flow field was not compensated resulting in larger axial velocity estimates.



(b) The drift in the axial position after a $\sim$ 40 meters of flight is $\sim$ 2 meters.

Figure 5.16: 3D position estimation results from two experiments at Carters Dam, GA. $y-z$ positions are estimated by the range-based estimator and x (axial position) is estimated using visual odometry. Scratches on the tunnel wall that we use to manually detect loop closure are circled in green.

Chapter 6

Local Mapping and Estimation with a 3D Lidar

Lightweight and high scan rate 3D lidars, after very recently becoming available to the robotics researchers, greatly facilitates robot perception in 3D settings. Prior to these devices, in order a robot to build 3D maps with comparable level of density and also attain omnidirectional awareness, the onboard 2D laser scanners were required to be retrofitted with electro-mechanical actuators or the robot had to execute sophisticated path planning algorithms. This chapter describes a state estimation and local mapping approach in penstocks using the DJI platform equipped with an IMU and a 3D lidar (Sec. 3.2.4). In particular, we are interested in exploiting the capabilities of the 3D lidar to eliminate the requirement of a prior map. The only requirement regarding the tunnel geometry is it to be in the shape of a generalized cylinder.

The proposed approach models the environment with a minimal set of parameters rather than using a prior point cloud map significantly reducing the memory and CPU required for representation as well as processing. Our system estimates the local map and the robot pose in real-time with onboard resources only. Furthermore, obstacles which are often not modeled in a prior map can be easily detected since comparison of the raw lidar data against the parametric map would reveal these anomalies. This enables us to generate trajectories with obstacle avoidance offering a safer experimental platform which is of utmost importance in the confined penstock setting. Since it does not require a prior map and is capable of detecting unmodeled obstacles, this system is superior compared to those described in the previous chapters. We demonstrate results from experiments conducted in Center Hill Dam, TN to show the performance of our estimator and the local mapper. At Center Hill Dam, TN we reached speeds of more than 3 m/s. Finally, this chapter is based on our previous work at [131].

6.1 Local Map Representation and Robot State

The local map, L, is represented as a list of cylindrical segments, $\{S\}$. A segment is an aggregate mathematical structure with three components : local frame, $\mathcal{L}$, radius ρ, and source point cloud, P, *i.e.* $S_s := \{\mathcal{L}_s, \rho_s, P_s\}$. Since each segment, by design, has the same length, we exclude this from the segment definition. The point cloud component is the subset of the raw point cloud, P, used to estimate the given segment. A point cloud is a list of pixels, $P := \{p\}$, which we also call a 3D range image. We follow the same pixel definition as in Sec. 4.3. The segment fitting procedure will be explained in the following sections.

The robot pose, except for the axial position, $\mathbf{x}_s$, is a function of the robot's immediate surroundings. Hence the knowledge of only the *local map* suffices for autonomy. Given this, the segment inside the convex hull of which the robot currently flies is defined to be the center segment. This segment is denoted as S_0 and is special since it also defines the *local map* coordinate frame, *i.e.* $\mathcal{L} := \mathcal{L}_0$. The rest of the segments are represented with respect to this frame.

The local map is re-estimated at each frame using the 3D raw point cloud, and is not a constant structure unlike the map used in the previous chapters. Hence, each segment is a function of time, *i.e.* $S_s = S_s(t)$. However, we drop the time argument for clarity. The segments of the local map are indexed with respect to the center segment as $L := \left\{..., S_{(-2)}, S_{(-1)}, S_0, S_1, S_2, ...\right\}$. The sign of an index is determined according to the corresponding segment's relative position with respect to the center segment which is obtained as $sign\left((\mathcal{L}_0)_{\hat{\mathbf{x}}}^{\top}(\mathcal{O}_s - \mathcal{O}_0)\right)$. In order to facilitate indexing adjacent segments, we define the following shorthand notation : the preceding and the succeeding segment indices are obtained, respectively, as

$$s^{-} = s - sign\,(s) \tag{6.1}$$

$$s^{+} = s + sign\,(s) \tag{6.2}$$

for $|s| > 0$.

We use the same state vector definition as in Chap. 5 which is

$$\mathbf{x} := [s, \dot{s}, \mathbf{r}_{y,z}, \boldsymbol{\Omega}, \mathbf{v}_{y,z}, \mathbf{b}_{\phi,\theta}, \mathbf{b}_{\mathbf{a}}, \mathbf{b}_{\boldsymbol{\omega}}]^{\top}. \tag{6.3}$$

Here, the cross-sectional position and velocity, $\mathbf{r}_{y,z}, \mathbf{v}_{y,z}$, are defined in the center segment local frame, $\mathcal{L}_0$. The Euler angles, $\boldsymbol{\Omega}$, is written with respect to the gravity frame, $\mathcal{G}$, where $\mathcal{G}_{\hat{\mathbf{z}}} = -\mathbf{g}$ and $\mathbf{g}$ is the gravity vector. Since the proposed estimator neither tracks a fixed world frame nor relies on a fixed prior map, the orientation of $\mathcal{G}$ around $\mathcal{G}_{\hat{\mathbf{z}}}$ is unknown. However, this is neither a deficiency nor causes ambiguity since a local map is sufficient to describe the *partial* state of the robot that the controller and the path planner requires. Indeed, the gravity frame is fixed such that $\mathcal{G}_{\mathcal{O}} = \mathcal{O}_0$ and $\mathcal{G}_{\hat{\mathbf{x}}}^{\top}(\mathcal{L}_0)_{\hat{\mathbf{y}}} = 0$. The definition of the rest of the state vector components are the same as in Equ. 5.1.

When the system is first activated, the gravity direction and its magnitude are calibrated by simply averaging the accelerometer measurements for a certain duration while the robot is kept stationary. This way the gravity vector is *indirectly* tracked in the form of the robot roll and pitch angles. Once this preemptive calibration step is complete, the gravity frame is defined as explained in the previous paragraph.

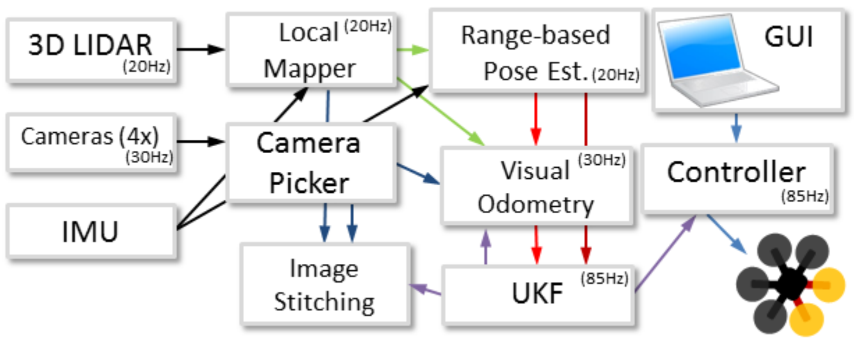


Figure 6.1: The overall system diagram. The inputs are data from three types of sensors and the user commands given through the GUI or the radio control. *Local Mapper* and *Range-Based Pose Est.* uses the IMU and 3D point cloud data to generate a local map of the tunnel and localize the robot within that map. *Camera Picker* chooses one or more of the cameras and relays the corresponding frames to the *Visual Odometry* block. The details of *Visual Odometry* is presented in the Chap. 5.

6.2 System Design

The system is composed of several building blocks which include sensory inputs, estimators, controller and user interface. Fig. 6.1 shows the inputs to the system which are IMU, 3D point cloud and image data from the onboard sensor suit. Partial state estimates from each estimator can be fused in a central UKF to estimate the 6 DoF state. Since the details of a possible sensor fusion algorithm has already been studied in the previous chapter, in this chapter we are focusing on the 3D local map generation and range-based partial state estimation. Furthermore, we implemented a shared controller such that the operator can give high-level way-point commands using the radio control or the GUI application running on the base station. The onboard controller can override the operator commands in case it detects a possible collision to prevent crashes caused by the human operator.

6.3 3D Point Cloud Preprocessing

In this work, we use the DJI platform (Sec. 3.2.4) which is equipped with a Velodyne Puck Lite 3D lidar as its primary exteroceptive sensor. The FOV of this sensor is 360 and 30 degrees in the azimuthal and the elevation directions respectively. Although the horizontal

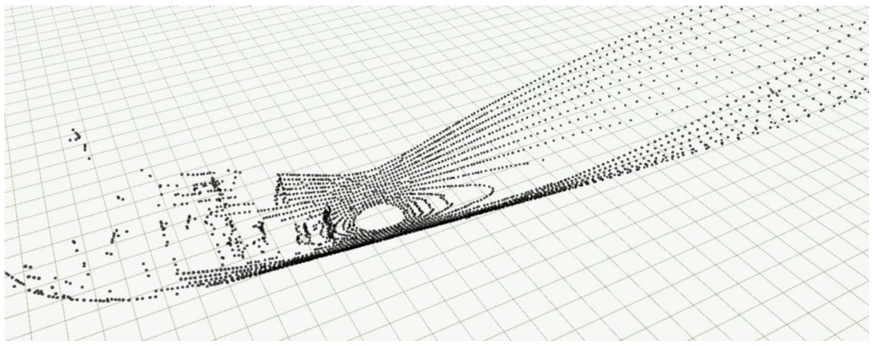


Figure 6.2: A typical point cloud data captured from inside a penstock at Center Hill Dam, TN. The raw point cloud is subsampled using a voxel filter with a cell size of 5 cm. Since the point density is small in the elevation direction, this filter affects only the point density along the azimuthal direction.

FOV lends to omnidirectional awareness with only a single range image, this is not the case for the vertical direction due to the narrow FOV. This prohibits reconstruction of the robot's surroundings directly from a single raw point cloud data. Only through assuming a model for the tunnel and then estimating its parameters, the environment can be reconstructed with a single lidar range image. The obvious downside of this approach is that predominance of discrepancies between the actual environment and the model may fail the downstream estimator catastrophically. We refer to all the unmodeled objects as *obstacles* and assume that the range and FOV of the lidar is sufficient to detect these from a safe distance. This way, the path planner can prevent the human operator flying the robot closer than a safe distance.

At the very first stage of the point cloud processing pipeline, the raw point cloud is downsampled using a voxel filter with its cell size being in the range of $[3 - 5]$ cm. The vertical resolution of the 3D lidar is $2^o/pixel$ which is much sparser compared to the horizontal resolution of $0.4^o/pixel$. Hence the main purpose of the downsampling is to reduce the unnecessarily high point density along the azimuthal direction. Since the vertical resolution is already low and the voxel size is small, this filter only reduces the horizontal point density. Downsampling the point cloud, significantly reduces the compute time and memory usage. A typical point cloud data collected while the robot was landed is shown in Fig. 6.2

The 100 meters maximum range of the Velodyne 3D lidar is much longer than the longest

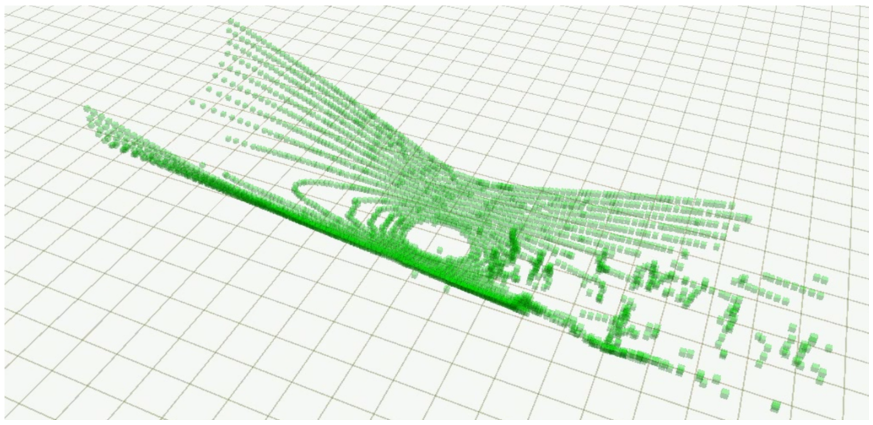


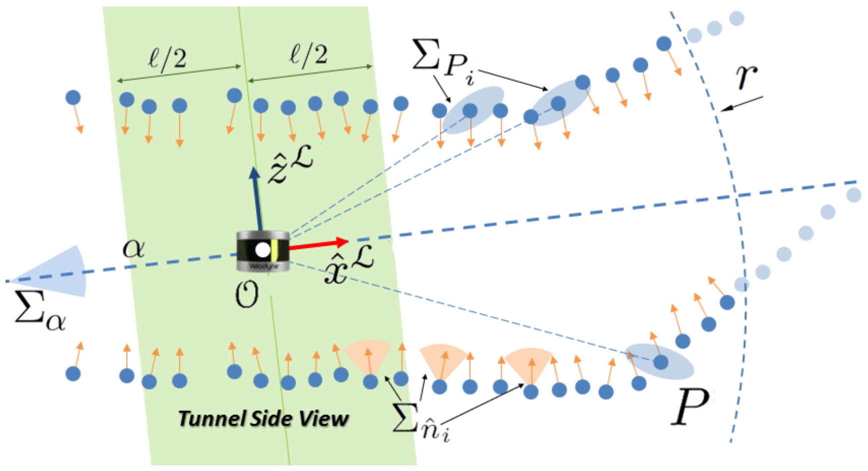
Figure 6.3: Accurate surface normal estimation is not possible beyond a certain distance from the sensor origin since the points get prohibitively sparse. For this reason, the voxel filtered point cloud is trimmed.

line of sight inside the tunnels we experiment. Furthermore, range measurements beyond a certain distance do not contribute to the state and local map estimation at all since the Euclidean distance between points corresponding to adjacent lidar pixels grows with the range and cannot capture the required level of geometric detail. This effect is more prominent especially along the lidar vertical direction due to the lower resolution. For this reason, as well as to reduce the CPU load, we filter out the original point cloud to exclude all the points with ranges longer than a certain threshold which ranges from 5 to 12 meters. A sample trimmed point cloud data is shown in Fig. 6.3

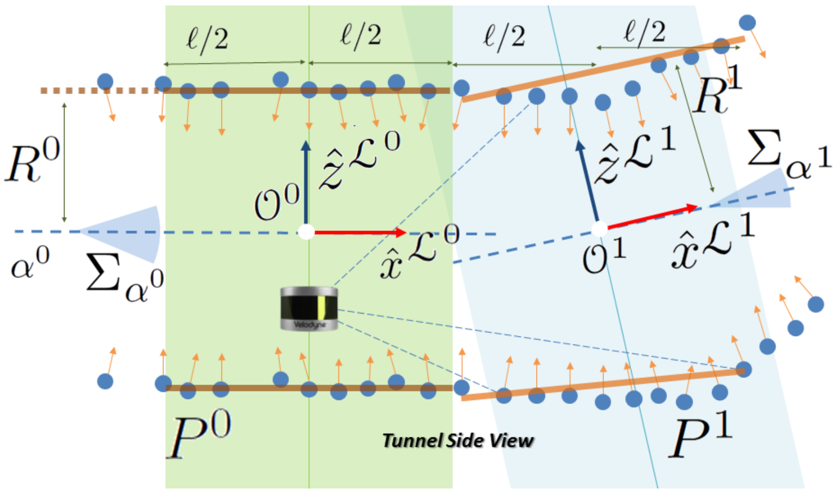
6.3.1 Surface Normal and Uncertainty Estimation

Surface normal estimation is at the core of the range-based estimation. Surface normals are used to estimate the axes of local map segments as will be discussed in the subsequent sections. The axis of a segment is estimated to be the vector which is perpendicular to all of its surface normals in the least squares sense. Unless otherwise stated, the calculations below are carried in the Velodyne frame, $\mathcal{V}$.

We define the normal of the point p_i, $\hat{\mathbf{n}}_i$, as the eigenvector corresponding to the smallest



(a) This figure illustrates point cloud data (blue circles) plotted in side-view. Samples for point position and normal uncertainties are shown in orange and blue shades. All the points within a certain radial distance from the lidar are used for fitting the the initial frame, $\mathcal{L}$.



(b) This figure shows the results of two iterations of local frame and segment estimation. Each frames is fitted using only the points that are inside a certain volume which are marked with green and blue shades.

Figure 6.4: Algo. 9 illustrated. These figures explain the components of the range-based pose estimation and the local mapping algorithm. Points farther than r meters from the sensor are drawn in light-blue and never used in the calculations. The inlier data (blue points) is denoted by P. Position uncertainties of sample points, formula of which is given in Equ. 6.8, are overlaid in light-blue. Normal vectors of points are plotted in orange. Uncertainties of sample normal vectors are also plotted in light-orange formula of which is given in Equ. 6.7. This figure also illustrates α and $\mathcal{A}$ further details for which are provided in Sec. 6.4.1. The origin of $\mathcal{L}$ is coincident with origin of the center segment, $\mathcal{O}_0$.

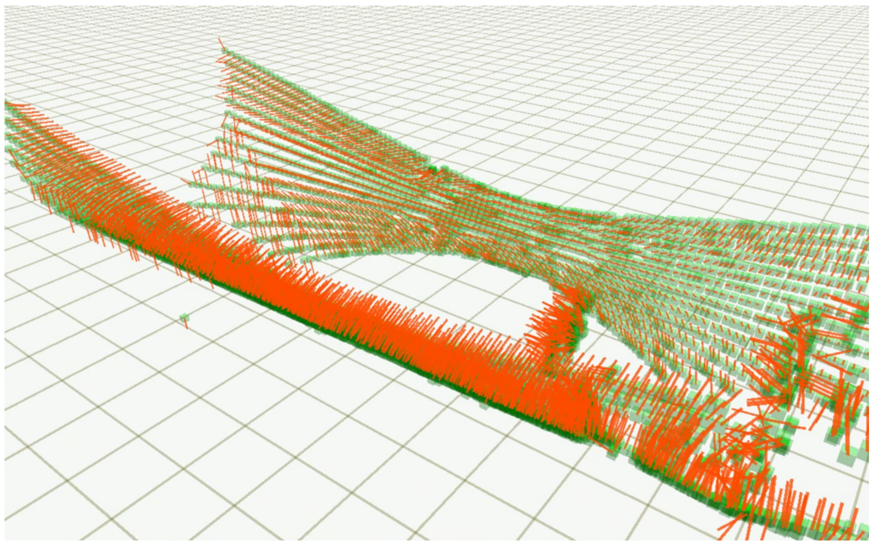


Figure 6.5: A sample point cloud data and surface normal estimates. The normals are estimated to be the eigenvector of the scatter matrix corresponding to its smallest eigenvalue (Equ. 6.6). The normal directions are fixed according to their relative orientation with respect to the view port. Although some of tunnel surface normals are pointing outwards, this does not affect the cylinder fitting process.

eigenvalue of the scatter matrix $\mathbf{S}_{p_i}$ [76] defined as

$$\{j\}_{p_i} := \left\{ j \;\;|\;\; ||p_j - p_i||_2 \leq r_{\hat{\mathbf{n}}} \right\} \tag{6.4}$$

$$\mu_{p_i} = \frac{1}{\left|\{j\}_{p_i}\right|} \sum_{j \in \{j\}_{p_i}} p_j \tag{6.5}$$

$$\mathbf{S}_{p_i} = \frac{1}{\left|\{j\}_{p_i}\right|} \sum_{j \in \{j\}_{p_i}} (p_j - \mu_{p_i})(p_j - \mu_{p_i})^\top \tag{6.6}$$

where $\{j\}_{p_i}$ is the set of indices, j, of the points p_j which are at most $r_{\hat{\mathbf{n}}}$ distant from p_i. For radius search we use the Kd-tree implementation of the PCL library [82]. Surface normals on a sample voxel filtered and trimmed point cloud is shown in Fig. 6.5.

We also estimate the uncertainty of $\hat{\mathbf{n}}_i$, by propagating the uncertainty of each point as [53]

$$\mathbf{\Sigma}_{\hat{\mathbf{n}}_i} = \left(\sum_{j \in \{j\}_{p_i}} \frac{\partial \hat{\mathbf{n}}_i}{\partial p_j} \left(\mathbf{\Sigma}_{p_j}^{-1} \right) \frac{\partial \hat{\mathbf{n}}_i}{\partial p_j}^\top \right)^{-1} \tag{6.7}$$

where $\mathbf{\Sigma}_{p_j}$ is the uncertainty of point p_j. This is found as

$$\mathbf{\Sigma}_{p_i} = {}^{\mathcal{V}}\mathbf{R}_{\mathcal{P}_i}\, \mathbf{\Sigma}_{p_i}^{\mathcal{P}_i}\, {}^{\mathcal{P}_i}\mathbf{R}_{\mathcal{V}} \tag{6.8}$$

where $\mathbf{\Sigma}_{p_i}^{\mathcal{P}_i}$ is the uncertainty of the corresponding point in the pixel frame, $\mathcal{P}_i$. The pixel frame relates to lidar frame, $\mathcal{V}$, as

$${}^{\mathcal{V}}\mathbf{R}_{\mathcal{P}_i} = \begin{bmatrix} \cos(\beta)\cos(\alpha) & -\cos(\beta)\sin(\alpha) & -\sin(\beta) \\ \sin(\alpha) & \cos(\alpha) & 0 \\ \sin(\beta)\cos(\alpha) & -\sin(\beta)\sin(\alpha) & \cos(\beta) \end{bmatrix} \tag{6.9}$$

where β and α are the elevation and azimuthal coordinates of p_i in lidar frame, $\mathcal{V}$, respectively. The uncertainty of this point is defined in the frame $\mathcal{P}_i$ is taken as

$$\mathbf{\Sigma}_{p_i}^{\mathcal{P}_i} = \frac{1}{\rho_i} \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_\beta^2 \end{bmatrix} \tag{6.10}$$

where $\sigma_r^2, \sigma_\alpha^2$ and σ_β^2 are the uncertainties in the range, azimuthal and elevation angles per unit range. $\mathbf{\Sigma}_{p_i}^{\mathcal{P}_i}$ models the measurement uncertainty to be linearly increasing with the range of the point. It should be noted that vectors and uncertainty matrices are defined in the lidar frame, $\mathcal{V}$, unless another frame is explicitly expressed in the equations as stated earlier.

In order to complete the derivation of normal uncertainty, $\mathbf{\Sigma}_{\hat{\mathbf{n}}_i}$, we have to determine the Jacobian $\frac{\partial \hat{\mathbf{n}}_i}{\partial p_j}$ where $j \in \{j\}_{p_i}$. Let λ_i be the smallest eigenvalue of $\mathbf{S}_{p_i}$ with its corresponding eigenvector $\hat{e}_i = \hat{\mathbf{n}}_i$. Each column of the partial derivative of $\hat{\mathbf{n}}_i$ is given as [7]

$$\frac{\partial \hat{\mathbf{n}}_i}{\partial p_{j,c}} = \frac{1}{\left|\{j\}_{p_i}\right|} (\lambda_i \mathbf{I} - \mathbf{S}_{p_i})^{\dagger} \frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,c}} \hat{\mathbf{n}}_i \tag{6.11}$$

$$\tag{6.12}$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\cdot^{\dagger}$ is the Moore-Penrose inverse, $p_{i,c}$ is the c^{th} component of p_i where $c \in \{x, y, z\}$. The partial derivatives of the scatter matrix, $\mathbf{S}_{p_i}$, for each c are

$$\frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,x}} = \frac{\begin{bmatrix} (p_j - \mu_{p_i})^\top \\ \mathbf{0}_{2\times 3} \end{bmatrix} + \begin{bmatrix} (p_j - \mu_{p_i})^\top \\ \mathbf{0}_{2\times 3} \end{bmatrix}^\top}{\left|\{j\}_{p_i}\right|} \tag{6.13}$$

$$\frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,y}} = \frac{\begin{bmatrix} \mathbf{0}_{1\times 3} \\ (p_j - \mu_{p_i})^\top \\ \mathbf{0}_{1\times 3} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{1\times 3} \\ (p_j - \mu_{p_i})^\top \\ \mathbf{0}_{1\times 3} \end{bmatrix}^\top}{\left|\{j\}_{p_i}\right|} \tag{6.14}$$

$$\frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,z}} = \frac{\begin{bmatrix} \mathbf{0}_{2\times 3} \\ (p_j - \mu_{p_i})^\top \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{2\times 3} \\ (p_j - \mu_{p_i})^\top \end{bmatrix}^\top}{\left|\{j\}_{p_i}\right|}. \tag{6.15}$$

Finally, the Jacobian matrices

$$\begin{aligned} \frac{\partial \hat{\mathbf{n}}_i}{\partial p_j} =& \frac{1}{\left|\{j\}_{p_i}\right|} \left(\lambda_i \, \mathbf{I}_{3\times 3} - \mathbf{S}_{p_i}\right)^\dagger \\ & \left[\frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,x}} \hat{\mathbf{n}}_i, \; \frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,y}} \hat{\mathbf{n}}_i, \; \frac{\partial \mathbf{S}_{p_i}}{\partial p_{j,z}} \hat{\mathbf{n}}_i\right] \end{aligned} \tag{6.16}$$

can be plugged into Equ. 6.7 to propagate point uncertainties into normal uncertainties. This completes the normal vector and its uncertainty estimation.

6.4 Point Cloud Segmentation and Surface Fitting

The narrow FOV of the Velodyne Puck along the elevation dimension (±15 degrees) imposes a significant constraint on the robot's ability to sense and model its surroundings. However, this can be remedied by exploiting the symmetry of the tunnel through estimating the parametric representation of the tunnel surface and extrapolate the raw point cloud data by that. Fitting cylindrical segments to the point cloud practically provides a 360 degrees of FOV around the tunnel axis. As opposed to direct point cloud matching algorithms such as [60], [134], this method does not suffer from data association problems which arise due to non-overlapping point cloud data across lidar frames. Furthermore, the parametric surface can also be used to estimate depths of image features with zero latency provided that the relative calibration of the lidar and the cameras is available.

6.4.1 Local Frame Initialization

The segmentation and mapping process is an iterative algorithm with can be summarized as *initialize-refine-recur* until all the points in P are consumed. P is the point cloud obtained after radius filtering and downsampling applied on the raw data as explained in Sec. 6.3.

The first step is to use the filtered point cloud, P, to initialize a rough, but reliable initial local coordinate frame. This frame is denoted as $\mathcal{L}_*$ where $\mathcal{O}_* = \mathbf{0}$ by definition. In order to define $\mathcal{L}_*$, we first have to estimate the local tunnel axis $\hat{\boldsymbol{\alpha}}_* := \mathcal{L}_{*\hat{\mathbf{x}}}$, which is found as the eigenvector corresponding to the smallest eigenvalue of

$$\mathbf{M} = (\mathbf{W}\mathbf{N})^\top(\mathbf{W}\mathbf{N}) \tag{6.17}$$

where

$$\mathbf{N} = \left[\hat{\mathbf{n}}_0, \hat{\mathbf{n}}_1, ..., \hat{\mathbf{n}}_{|P|-1}\right]^\top \tag{6.18}$$

$$\mathbf{W} = diag\left[e^{-(\kappa_0/\kappa_\tau)^2}, e^{-(\kappa_1/\kappa_\tau)^2}, ..., e^{-\left(\kappa_{|P|-1}/\kappa_\tau\right)^2}\right]. \tag{6.19}$$

Here κ_i is the curvature of p_i which is defined as the ratio of the smallest eigenvalue of $\mathbf{S}_{p_i}$ to their summation [76] and κ_τ is a constant which we choose to be in the range $[0.1, 0.3]$. $\mathbf{W}$ assigns smaller weights to points at high curvature regions. Once the local map centerline tangent is estimated, we can then use the procedure explained in Sec. 4.1.2 to construct the frame, *i.e.* $\mathcal{L}_* := \left\{\mathcal{L}_{*\hat{\mathbf{x}}}, \mathcal{L}_{*\hat{\mathbf{y}}}, \mathcal{L}_{*\hat{\mathbf{z}}}, \mathcal{O}_*\right\}$. This process is explained in Algo. 3 and illustrated

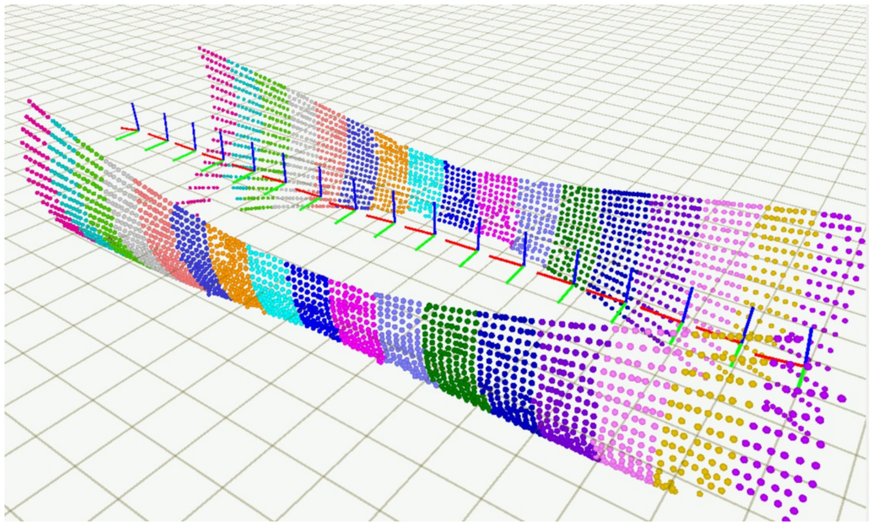


Figure 6.6: This figure shows a sample point cloud data and reference frame estimates along with their corresponding color-coded point cloud segments used in their estimation. The input point cloud is segmented and to each segment a reference frame is fitted as explained in Algo. 3-4. The point due to objects and human operators are filtered out from the point cloud segments as explained in Algo. 5-6.

in Fig. 6.4. Also a sample output with reference frames and the point cloud segments used for their estimation are shown in Fig. 6.6

6.4.2 Segment Initialization

Segment initialization is handled differently for the cases $s = 0$ and $|s| > 0$. S_0 is initialized as $\mathcal{L}_0 = \mathcal{L}_*$ and $\mathcal{O}_0 = \mathcal{O}_*$. P_s is defined as

$$P_s := \left\{ p_i \mid p_i \in P \ , \ \|p_{i,x}^{\mathcal{L}_s}\| \leq \frac{\ell}{2} \right\} \tag{6.20}$$

where

$$p_{i,x}^{\mathcal{L}_s} = [1, \ 0, \ 0] \ {}^{\mathcal{L}_s}\mathbf{R}_{\mathcal{V}} \ (p_i - \mathcal{O}_s) \, . \tag{6.21}$$

Algorithm 3: *estimateLocalFrame*

```
Data: $[\{\mathbf{S}\}]$ /* point scatter matrices */
Result: $[\mathcal{L}]$
/* Construct the least-squares system */
for $i = 0;\ i < |\{\mathbf{S}\}|\,;\ i{+}{+}$ do
    /* get the eigenvalues of the scatter matrix in ascending order */
    $\{\lambda\} \Leftarrow evals\,(\mathbf{S}_i)$
    /* get the eigenvector corresponding to the smallest eigenvalue */
    $\mathbf{N}_i \Leftarrow minevec\,(\mathbf{S}_i)^\top$
    /* approximate the curvature */
    $\kappa \Leftarrow \frac{\lambda_0}{\sum_j \lambda_j}$
    /* construct the weigh matrix */
    $\mathbf{W}_{i,i} \Leftarrow e^{-(\kappa/\kappa_\tau)^2}$
end
/* find the local map axis */
$\mathcal{L}_{\hat{\mathbf{x}}} \Leftarrow minevec\left((\mathbf{WN})^\top(\mathbf{WN})\right)$
/* construct the local map frame as explained in Sec.
4.1.2 */
$\mathcal{L} \Leftarrow constructFrame\,(\mathcal{L}_{\hat{\mathbf{x}}})$
```

Algorithm 4: *segmentPointCloud*

```
Data: $[P, \mathcal{L}]$
Result: $\left[\tilde{P}\right]$
/* retrieve the origin of the local frame */
$[\mathcal{O}] \Leftarrow \mathcal{L}$
/* initialize the output point cloud */
$\tilde{P} \Leftarrow \emptyset$
foreach $p_i \in P$ do
    /* transform the point from lidar frame to local map frame */
    $p_i^{\mathcal{L}} \Leftarrow {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}(p_i - \mathcal{O})$
    /* check if the point is within the radius threshold */
    if $\left|p_{i,x}^{\mathcal{L}}\right| \le \ell/2$ then
        /* add the point into the output point cloud */
        $\tilde{P} \Leftarrow \tilde{P} \bigcup p_i$
end
```

For other cases, $|s| \geq 1$, the coordinate frame is initialized as

$$\mathcal{L}_s = \mathcal{L}_{s^-} \tag{6.22}$$

$$\mathcal{O}_s = \mathcal{O}_{s^-} + sign\,(s)\ \ell\,(\mathcal{L}_{s^-})_{\hat{\mathbf{x}}}. \tag{6.23}$$

P_s can be obtained the same way as P^0 through proper coordinate frame substitution in Equ. 6.20. Point cloud segmentation is also explained in Algo. 4 and illustrated in Fig. . 6.4b. In our tests, we choose ℓ to be in the range of $[0.10 - 2]$ meters.

6.4.3 Segment Refinement

The segment initialization is followed by parameter refinement and outlier rejection. These steps are recursed to obtain S_s and $S_{(-s)}$ in pairs where the sign denotes the forward and backward direction with respect to the tunnel axis. The process continues until all the points in P are either segmented into tunnel sections or marked as outliers.

The quality of segment initialization is dependent on the estimation quality of the previous segment, $S_{(s^-)}$, as well as the noise level of sensor data which is not guaranteed for certain cases. For this reason, initialization is followed by refining the point set, P_s, as well as the coordinate frame definitions, $\mathcal{L}_s - \mathcal{O}_s$, through outlier rejection and refitting the model iteratively until convergence. The two assumptions we use for outlier detection are the agreement of local surface normals, $\hat{\mathbf{n}}_i$, with the local tunnel axis, $(\mathcal{L}_s)_{\hat{\mathbf{x}}}$, and the cylindrical surface model. The former assumption eliminates points with incompatible surface normals to obtain more accurate local frames and the latter removes regions of the point cloud that do not comply with the cylindrical surface assumption due to objects such as scaffolding and human operators. The two methods are given in Algo. 5 and Algo. 6 respectively.

Outlier rejection based on the surface normal assumes that each normal, is perpendicular to the local tunnel axis. This condition is written as

$$\left| \hat{\mathbf{n}}_i^\top (\mathcal{L}_s)_{\hat{\mathbf{x}}} \right| < \tau_{\hat{\mathbf{n}}}. \tag{6.24}$$

Algorithm 5: *outlierEliminationNormals*

```
Data: $[S]$
Result: $[S, n]$
/* retrieve the components of the input segment */
$[P, \mathcal{L}, \mathcal{O}] \Leftarrow S$
/* count the number of removed points */
$n \Leftarrow 0$
for $i = 0$ ; $i < |P|$ ; $i$++ do
    /* remove points not compliant with the centerline tangent */
    if $\left|\hat{\mathbf{n}}_i^\top \hat{\boldsymbol{\alpha}}\right| < \tau_{\hat{\mathbf{n}}}$ then
        $P \Leftarrow P \setminus p_i$
        $n$++
end
```

Algorithm 6: *outlierEliminationRadius*

```
Data: $[S]$
Result: $[S, n]$
/* retrieve the components of the input segment */
$[P, \mathcal{L}, \mathcal{O}, \rho] \Leftarrow S$
/* count the number of removed points */
$n \Leftarrow 0$
for $i = 0$ ; $i < |P|$ ; $i$++ do
    /* transform the point from lidar frame to local map frame */
    $p = {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}} \left(p_i - \mathcal{O}\right)$
    /* remove points far from annulus */
    if $\left|1 - \frac{||p_{y,z}||_2}{\rho}\right| \geq \tau_\rho$ then
        $P = P \setminus p$
end
```

We choose $\tau_{\hat{n}}$ in the range of $[0.1, 0.25]$. Both the surface normal compliance assumption and the low weight assigned to high curvature regions (Algo. 3) for tunnel axis estimation, increases the robustness of the local frame estimates.

Algorithm 7: *fitModel*

```
Data: $[S]$
Result: $[\mathcal{O}_c, \rho]$
/* retrieve the components of the input segment */
$[P, \mathcal{L}, \mathcal{O}] \Leftarrow S$
/* initialize solver */
$\mathbf{A} \Leftarrow \emptyset$
$\mathbf{b} \Leftarrow \emptyset$
for $i = 0$ ; $i < |P|$ ; $i$++ do
    $p = {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}} (p_i - \mathcal{O})$
    $\mathbf{A}_i \Leftarrow \left[p_{y,z}^\top, \ 1\right]^\top$
    $\mathbf{b}_i \Leftarrow -\left|\left|p_{y,z}\right|\right|_2^2$
end
$\mathbf{f} \Leftarrow \mathbf{A}^\dagger \mathbf{b}$
/* calculate the model center w.r.t. local map frame */
$\mathcal{O}_c \Leftarrow -\frac{1}{2}\left[0, \ \mathbf{f}_{1,2}^\top\right]^\top$
$\rho \Leftarrow \sqrt{\frac{||\mathbf{f}_{1,2}||_2^2}{16} - \mathbf{f}_3}$
```

As explained in Sec. 6.1, a segment is assumed to be cylindric with a fixed length, ℓ. We use the points in P_s to fit a cylindrical model assuming that the tunnel has a parametric cross-section which is circle in this case. In order to simplify the model fitting problem, we use the fact that, for a reasonably well estimated local frame $\mathcal{L}_s$, the projection of the points in P_s onto the local cross-sectional plane formed by $(\mathcal{L}_s)_{\hat{\mathbf{y}}} - (\mathcal{L}_s)_{\hat{\mathbf{z}}}$ forms a circle. The parameter estimation then can be formulated as a linear regression written as

$$\mathbf{A}_i = \left[\left(p_i^{\mathcal{L}_s}\right)_{y,z}^\top, \ 1\right] \tag{6.25}$$

$$\mathbf{b}_i = -\left|\left|\left(p_i^{\mathcal{L}_s}\right)_{y,z}\right|\right|_2^2 \tag{6.26}$$

$$\mathbf{f} = \mathbf{A}^\dagger \mathbf{b} \tag{6.27}$$

where $\mathbf{A}_i$ and $\mathbf{b}_i$ select the i^{th} row of the corresponding tensor, $\left(p_i^{\mathcal{L}_s}\right)_{y,z}$ is the y, z coordinates of the given point in the local map frame, $\mathcal{L}_s$, $\mathbf{f}$ is a 3-vector and $\cdot^\dagger$ is the Moore-Penrose

Algorithm 8: $refineSegment$

```
Data: $[S, \{\mathbf{S}\}]$
Result: $[S]$
/* retrieve the components of the input segment */
$[P, \mathcal{L}, \mathcal{O}, \rho] \Leftarrow S$
/* refine the segment iteratively */
for $outer = 0$ ; $outer < outer_{max}$ ; $outer{+}{+}$ do
    /* update local frame and eliminate outliers */
    for $inner = 0$ ; $inner < inner_{max}$ ; $inner{+}{+}$ do
        $\mathcal{L} \Leftarrow estimateLocalFrame(\{\mathbf{S}\})$
        $[S, n] \Leftarrow outlierEliminationNormals(S)$
        if $n = 0$ then
            break
    end
    /* update model parameters and eliminate outliers */
    for $inner = 0$ ; $inner < inner_{max}$ ; $inner{+}{+}$ do
        $[\mathcal{O}_c, \rho] \Leftarrow fitModel(S)$
        $\mathcal{O} \Leftarrow \mathcal{O} + {}^{\mathcal{V}}\mathbf{R}_{\mathcal{L}}\, \mathcal{O}_c$
        $[S, n] \Leftarrow outlierEliminationRadius(S)$
        if $n = 0$ then
            break
    end
end
```

inverse. The center and the radius of the model are found as

$$\tilde{\mathcal{O}}_s^{\mathcal{L}_s} = -\frac{1}{2}\begin{bmatrix}\mathbf{f}_{1,2}\\0\end{bmatrix} \tag{6.28}$$

$$\rho_s = \sqrt{\frac{||\mathbf{f}_{1,2}||_2^2}{16} - \mathbf{f}_3}. \tag{6.29}$$

Note that the model center, $\tilde{\mathcal{O}}_s$, is defined in its corresponding local frame, $\mathcal{L}_s$. At each refinement step, the model center is used to update the local frame origin, $\mathcal{O}_s$. The update is written as

$$\mathcal{O}_s \Leftarrow \mathcal{O}_s + \left({}^{\mathcal{V}}\mathbf{R}_{\mathcal{L}_s}\right)\tilde{\mathcal{O}}_s^{\mathcal{L}_s}. \tag{6.30}$$

The model fitting algorithm is given in Algo. 7. Also a sample output of this is given in Fig. 6.7. This equation indicates that the model center and the local frame have to be aligned

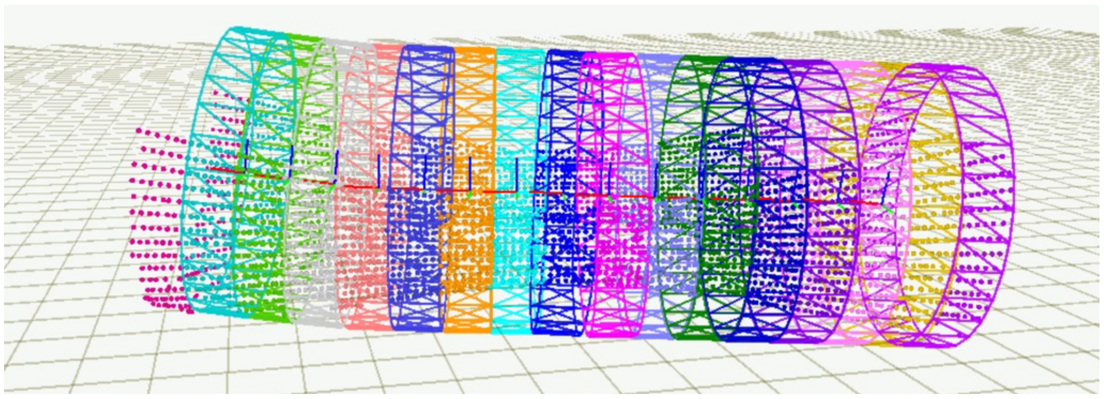


Figure 6.7: This figure shows a sample segmented point cloud data. To each segment a cylindrical surface is fitted using the method summarized in Algo. 7.

due to the cylindrical cross-section assumption. This always holds in straight sections of the tunnel whereas ℓ should be chosen to be smaller as the curvature of the tunnel increases, such as around bends.

The second outlier rejection uses the cylindrical surface assumption. Points that are far from the surface are marked as outliers. This criterion is given as

$$\left\|1 - \frac{1}{\rho_s}\left(p_{i,(y,z)}^{\mathcal{L}_s}\right)\right\|_2 \leq \tau_\rho \tag{6.31}$$

where ρ_s is the radius of the segment S_s and τ_ρ is a constant threshold that is in the range $[0.1, 0.2]$.

The segment refinement process is explained in Algo. 8 and the complete local map estimation process is given in Algo. 9. Also Fig. 6.4b illustrates these steps. In this figure, the green and light-blue shaded rectangles designate the regions of each local frame, $\mathcal{L}_0$ and $\mathcal{L}_1$ respectively. Points belonging to each region are denoted as P_0 and P_1. The algorithm starts with the initialization step as illustrated in Fig. 6.4a. The points inside the shaded region in Fig. 6.4a are then used to refine the local frame estimate $\mathcal{L}_0$. After the refinement, we obtain $\left\{(\mathcal{L}_0)_{\hat{\mathbf{x}}}, (\mathcal{L}_0)_{\hat{\mathbf{y}}}, (\mathcal{L}_0)_{\hat{\mathbf{z}}}\right\}$ and $\mathcal{O}_0$ defined in $\mathcal{V}$. Then, a cylindrical surface is fit to P_0 with all the corresponding parameter uncertainties as explained in the following sections. Once the 0^{th} segment is processed, an initial guess for $\mathcal{O}_1$ and $\hat{\boldsymbol{\alpha}}_1$ is made by extrapolating $\hat{\boldsymbol{\alpha}}_0$ by ℓ meters. This guess is then refined to obtain $\left\{(\mathcal{L}_1)_{\hat{\mathbf{x}}}, (\mathcal{L}_1)_{\hat{\mathbf{y}}}, (\mathcal{L}_1)_{\hat{\mathbf{z}}}\right\}$ and $\mathcal{O}_1$ using

Algorithm 9: *buildLocalMap*

```
Data: $[P]$
Result: $[\{S\}]$
/* apply radius and voxel filters Sec.
6.3 */
$P \Leftarrow downsample(P)$
/* estimate surface normals and uncertainties Sec.
6.3.1 */
$[\{\hat{\mathbf{n}}\}, \{\mathbf{S}\}] \Leftarrow estimateSurfaceNormals(P)$
/* estimate initial segment Algo.
3 */
$\mathcal{L}_* \Leftarrow estimateLocalFrame(\{\mathbf{S}\})$
$\mathcal{O}_* \Leftarrow \mathbf{0}$
for $s \Leftarrow 0$ ; $s \leq s_{max}$ ; $s{+}{+}$ do
    for $t \in [-s, s]$ do
        /* retrieve point cloud that makes the current segment */
        $P_t \Leftarrow segmentPointCloud(P, \mathcal{L}_{t^-})$
        /* initial current segment */
        $\mathcal{L}_t \Leftarrow \mathcal{L}_{t^-}$
        $\mathcal{O}_t \Leftarrow \mathcal{O}_{t^-} + sign\,(t)\ \ \ell\,(\mathcal{L}_{t^-})_{\hat{\mathbf{x}}}$
        $S_t \Leftarrow [P_t, \mathcal{L}_t, \mathcal{O}_t]$
        /* Algo.
        8 */
        $S_t \Leftarrow refineSegment(S_t, \{\mathbf{S}\})$
        /* removed used points */
        $P \Leftarrow P \backslash P_t$
        if $P = \emptyset$ then
            break;
    end
    $s{+}{+}$
end
```

the point set P_1 where P_1 consists of all the unused points that are at most $\ell/2$ meters away from the $(\mathcal{L}_1)_{\hat{\mathbf{y}}}$-$(\mathcal{L}_1)_{\hat{\mathbf{z}}}$ plane.

6.5 Uncertainty Estimation of Local Frames

The uncertainty of the tunnel axis, $\hat{\boldsymbol{\alpha}}$, is estimated in a similar way as the normal uncertainty estimation. We estimate this by propagating normal uncertainties through the equation

$$\Sigma_{\hat{\boldsymbol{\alpha}}} = \left(\sum_i \frac{\partial \hat{\boldsymbol{\alpha}}}{\partial \hat{\mathbf{n}}_i} \left(\boldsymbol{\Sigma}_{\hat{\mathbf{n}}_i}\right)^{-1} \frac{\partial \hat{\boldsymbol{\alpha}}}{\partial \hat{\mathbf{n}}_i}^{\top} \right)^{-1} \tag{6.32}$$

where the summation is over all surface normals that contribute to the estimation of the given local tunnel axis. The partial derivative of the local tunnel axis with respect to the normals is calculated in a similar way as the normal vector partials. Let λ_i be the smallest eigenvalue of $\mathbf{M}$ in Equ. 6.17 with its corresponding eigenvector being $\hat{\mathbf{e}} = \hat{\boldsymbol{\alpha}}$. Each column of the derivative of $\hat{\boldsymbol{\alpha}}$ is given as [7]

$$\frac{\partial \hat{\boldsymbol{\alpha}}}{\partial \hat{\mathbf{n}}_{i,c}} = (\lambda_i \mathbf{I} - \mathbf{M})^{\dagger} \frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,c}} \hat{\boldsymbol{\alpha}} \tag{6.33}$$

where $\mathbf{I}$ is an identity matrix, $\hat{\mathbf{n}}_{i,c}$ is the c^{th} component of $\hat{\mathbf{n}}_i$ with $c \in \{x, y, z\}$ and

$$\frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,x}} = \mathbf{W}_{i,i}^2 \left(\begin{bmatrix} \hat{\mathbf{n}}_i^{\top} \\ \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} \end{bmatrix}^{\top} + \begin{bmatrix} \hat{\mathbf{n}}_i \\ \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} \end{bmatrix} \right) \tag{6.34}$$

$$\frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,y}} = \mathbf{W}_{i,i}^2 \left(\begin{bmatrix} \mathbf{0}_{1\times 3} \\ \hat{\mathbf{n}}_i^{\top} \\ \mathbf{0}_{1\times 3} \end{bmatrix}^{\top} + \begin{bmatrix} \mathbf{0}_{1\times 3} \\ \hat{\mathbf{n}}_i \\ \mathbf{0}_{1\times 3} \end{bmatrix} \right) \tag{6.35}$$

$$\frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,z}} = \mathbf{W}_{i,i}^2 \left(\begin{bmatrix} \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} \\ \hat{\mathbf{n}}_i^{\top} \end{bmatrix}^{\top} + \begin{bmatrix} \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} \\ \hat{\mathbf{n}}_i \end{bmatrix} \right) \tag{6.36}$$

where $\mathbf{W}$ was previously defined for Equ. 6.17. The Jacobian in the above equation may be written

$$\frac{\partial \hat{\boldsymbol{\alpha}}}{\partial \hat{\mathbf{n}}_i} = (\lambda_i \mathbf{I} - \mathbf{M})^{\dagger} \left[\frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,x}} \hat{\boldsymbol{\alpha}},\ \frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,y}} \hat{\boldsymbol{\alpha}},\ \frac{\partial \mathbf{M}}{\partial \hat{\mathbf{n}}_{i,z}} \hat{\boldsymbol{\alpha}} \right] \tag{6.37}$$

This can be substituted in Equ. 6.32 to get $\boldsymbol{\Sigma}_{\hat{\boldsymbol{\alpha}}}$.

6.6 Robot State and Its Uncertainty

We employ the same state definition, process model and measurement updates as in Chap. 5. The lateral and vertical positions of the robot as well as the yaw angle are estimated *indirectly* by the segment fitting algorithm discusses in the previous section. This section explains the transformation from segment estimates to the measurement model domain.

Complete robot orientation can be estimated only with the presence of an IMU since roll and pitch angles are not observable to the range sensor as discussed previously. We integrate the rotational velocity from the IMU in our UKF at a high rate and also perform a measurement update at a lower rate using the onboard autopilot attitude estimate which applies gravity correction to eliminate possible error accumulation in the roll and pitch estimation. The process model and the corresponding measurement update are given by Equ. 5.2 and in Sec. 5.3.3.

The yaw component of the robot state, $\mathbf{x}_\psi$, can be inferred using the local tunnel axis estimate, $\hat{\boldsymbol{\alpha}}_0$. For clarity, we will omit the subscript in the following derivations. On the DJI platform the IMU frame, $\mathcal{I}$, according to which the controller input is defined, is coincident with the body frame, $\mathcal{B}$. The segments, on the other hand, are estimated with respect to the lidar frame, $\mathcal{V}$ and the state Euler angles relate the body and the gravity frames. Based on this, the relation that yields the yaw angle of the robot writes as

$$\mathcal{G}_{\hat{\mathbf{y}}}^{\top} \left({}^{\mathcal{G}}\mathbf{R}_{\mathcal{B}} {}^{\mathcal{B}}\mathbf{R}_{\mathcal{V}} \hat{\boldsymbol{\alpha}} \right) = 0 \tag{6.38}$$

where ${}^{\mathcal{B}}\mathbf{R}_{\mathcal{V}}$ is known due to the sensor external calibration, and ${}^{\mathcal{G}}\mathbf{R}_{\mathcal{B}}$ is the rotation matrix corresponding to the state Euler angles, $\mathbf{x}_{\boldsymbol{\Omega}}$. The orthogonality of the two vectors is due to how the gravity and local frames are defined as explained in Sec. 4.1.2. This equation can be rewritten as

$$\mathcal{G}_{\hat{\mathbf{y}}}^{\top}\mathbf{R}_z(\mathbf{x}_\psi)\mathbf{R}_y(\mathbf{x}_\theta)\mathbf{R}_x(\phi){}^{\mathcal{B}}\mathbf{R}_{\mathcal{V}}\hat{\boldsymbol{\alpha}} = 0 \tag{6.39}$$

after expanding ${}^{\mathcal{G}}\mathbf{R}_{\mathcal{B}}$ into its corresponding Euler angle rotations. The roll and pitch angles, $\mathbf{x}_\phi - \mathbf{x}_\theta$, are know from the robot state leaving the yaw angle, $\mathbf{x}_\psi$, as the only unknown. After performing the following substitutions

$$\mathcal{G}_{\hat{\mathbf{y}}} = [0,\ 1,\ 0]^{\top} \tag{6.40}$$

$$\mathbf{R}_z(\mathbf{x}_\psi) = \begin{bmatrix} \cos(\mathbf{x}_\psi) & -\sin(\mathbf{x}_\psi) & 0 \\ \sin(\mathbf{x}_\psi) & \cos(\mathbf{x}_\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.41}$$

$$\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}} := \mathbf{R}_y(\mathbf{x}_\theta)\mathbf{R}_x(\mathbf{x}_\phi){}^{\mathcal{B}}\mathbf{R}_{\mathcal{V}}\hat{\boldsymbol{\alpha}} \tag{6.42}$$

the Equ. 6.39 simplifies to

$$[\sin(\mathbf{x}_\psi),\ \cos(\mathbf{x}_\psi),\ 0]\,\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}} = 0. \tag{6.43}$$

Finally, $\mathbf{x}_\psi$ can be obtained as

$$\mathbf{x}_\psi = -atan2\left(\hat{\boldsymbol{\alpha}}_y^{\tilde{\mathcal{G}}}, \hat{\boldsymbol{\alpha}}_x^{\tilde{\mathcal{G}}}\right) \tag{6.44}$$

The uncertainty in $\mathbf{x}_\psi$ can be estimated by propagating the uncertainty in $\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}$ as (omitting the superscript)

$$\boldsymbol{\Sigma}_\psi = \left(\frac{\partial\psi}{\partial\hat{\boldsymbol{\alpha}}_{x,y}^{\tilde{\mathcal{G}}}}\left(\boldsymbol{\Sigma}_{\hat{\boldsymbol{\alpha}}_{x,y}^{\tilde{\mathcal{G}}}}\right)^{-1}\frac{\partial\psi}{\partial\hat{\boldsymbol{\alpha}}_{x,y}^{\tilde{\mathcal{G}}}}^{\top}\right)^{-1} \tag{6.45}$$

where

$$\frac{\partial \psi}{\partial \hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}_{x,y}} = \frac{1}{\left\|\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}_{x,y}\right\|_2} \left[\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}_y,\ -\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}_x\right]. \tag{6.46}$$

The uncertainty of the local tunnel axis in the intermediate $\tilde{\mathcal{G}}$ frame can be obtained as

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\alpha}}^{\tilde{\mathcal{G}}}} = \tilde{\mathbf{R}}\boldsymbol{\Sigma}_{\hat{\boldsymbol{\alpha}}}\tilde{\mathbf{R}}^\top \tag{6.47}$$

$$\tilde{\mathbf{R}} := \mathbf{R}_y(\mathbf{x}_\theta)\mathbf{R}_x(\mathbf{x}_\phi)^{\mathcal{B}}\mathbf{R}_{\mathcal{V}} \tag{6.48}$$

which completes the derivation of $\boldsymbol{\Sigma}_\psi$.

We prefer defining the position of the robot with respect to the local map origin, $\mathcal{O} := \mathcal{O}_0$ rather than a fixed world frame. This offers an intuitional way of defining way-points for the controller and also makes the estimator immune to drifts in vertical and lateral directions. Similar to the way we inferred the robot orientation from the central segment axis, we transform the central segment origin, $\mathcal{O}_0$, which is estimated with respect to the lidar frame, $\mathcal{V}$, to get the robot lateral and vertical positions in the local map frame, $\mathcal{L} := \mathcal{L}_0$. This is found as

$$\mathbf{x}_{y,z} = -\left({}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}\ \mathcal{O}_0\right)_{y,z}. \tag{6.49}$$

The uncertainty in $\mathbf{x}_{y,z}$ is a linear transformation of the uncertainty of $\mathcal{O}_0$, which writes (all in $\mathcal{V}$)

$$\boldsymbol{\Sigma}_{y,z} = \left(\sum_{p_i \in \{P_0\}} \left(\frac{\partial \mathcal{O}_{0,(y,z)}}{\partial p_i}\right) \boldsymbol{\Sigma}_{p_i}{}^{-1} \left(\frac{\partial \mathcal{O}_{0,(y,z)}}{\partial p_i}\right)^\top\right)^{-1} \tag{6.50}$$

where

$$\frac{\partial \mathcal{O}_{0,(y,z)}}{\partial p_i} = \frac{\partial \mathcal{O}_{0,(y,z)}}{\partial \mathbf{f}} \frac{\partial \mathbf{f}}{\partial p_i} \tag{6.51}$$

$$\frac{\partial \mathbf{f}}{\partial p_i} = \mathbf{A}^{\dagger} \left(\frac{\partial \mathbf{b}}{\partial p_i} - \frac{\partial \mathbf{A}}{\partial p_i} \mathbf{f} \right) \tag{6.52}$$

and

$$\frac{\partial \mathbf{A}}{\partial p_{i,x}} = \mathbf{0}, \quad \frac{\partial \mathbf{A}}{\partial p_{i,y}} = \delta_{i,0}, \quad \frac{\partial \mathbf{A}}{\partial p_{i,z}} = \delta_{i,1},$$

$$\frac{\partial \mathbf{b}}{\partial p_{i,x}} = \mathbf{0}, \quad \frac{\partial \mathbf{b}}{\partial p_{i,y}} = -2\delta_{i,0}\ p_{i,y}, \quad \frac{\partial \mathbf{b}}{\partial p_{i,z}} = -2\delta_{i,0}\ p_{i,z},$$

$$\frac{\partial \mathcal{O}_{0,y}}{\partial \mathbf{f}} = -\frac{1}{2} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \frac{\partial \mathcal{O}_{0,z}}{\partial \mathbf{f}} = -\frac{1}{2} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

and $\delta_{i,j}$ is a zero matrix except that it is 1 at (i, j). $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{f}$ are the tensors first introduced for the model fitting algorithm in Sec. 6.4.3. We can obtain the uncertainty in position in the lidar frame by chaining the above partials. Finally, this can be transformed into the local map frame as

$$\mathbf{\Sigma}^{\mathcal{L}}_{y,z} = \bar{\mathbf{R}} \mathbf{\Sigma}_{y,z} \bar{\mathbf{R}}^{\top} \tag{6.53}$$

where $\bar{\mathbf{R}}$ is the top left 2×2 block of ${}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}$. One should note that we silently assumed that $\mathcal{O}_{0,c} \equiv \mathcal{O}_0$ which is valid for a successful cylinder fitting.

6.7 A Discussion on Estimator Robustness

The previous section explains the procedure for inferring the robot state from the local map measurement. In summary, the state of the central segment is *inverted* and used in the measurement update of the underlying Kalman filter. The rest of the segments are not used at all for this inference which might raise questions about the necessity of building a larger map and robustness of the robot state measurement. We refer to the next chapter for detailed answers to these questions which explains an improved version of this estimator that utilizes *all* the segments to robustify this process. On the other hand, building a larger (*longer*) local map, even though not used in robot state estimation, is beneficial for detecting obstructions which can be used in path planning and shared control. A sample case where this feature is used to detect the end of a penstock and prevent the robot from crashing into the gate is shown in Fig. 6.9b. The number of segments estimated in forward and backward directions can also be adjusted independently depending on the direction of the robot motion along the axis if axial speed or, at least, its direction is available. Optical flow from the cameras or the pitch angle of the robot can be used for this purpose. Estimating only as many segments as required for shared control will also occupy less CPU time and battery power.

The latter point regarding the robustness is rather concerning. The circularity assumption does not hold for certain sections of penstocks such as along sharp bendings and at sections close to the gate. While a few glitches per second in the robot state measurement can be handled easily by the Kalman filter, erroneous measurements for longer periods might cause divergence of the filter. A relatively easy solution to this would be to increase the length of the central segment at the cost of measurement accuracy. When too few points are used, especially the segment orientation estimation may fail grossly (Fig. 7.8), and using more points would remedy this weakness of the optimizer (Sec. 6.4.3). Thus increasing the length of the central segment, hence using more points, reduces the likelihood of abrupt changes in the central segment estimation. Along non-straight sections of a penstock, increase in the segment length would degrade the measurement accuracy since the straight cylindrical segment assumption does not hold. Two segments, one short and the other long, can be fit and compared to check if the short segment is *close* to the longer one. If the short segment estimate passes this sanity check, then it can be used for the Kalman measurement update.

6.8 Obstacle Avoidance and Shared Control

The local map of the tunnel is used to detect free space for navigation as well as for state estimation. The points that are marked as outlier by the robust cylinder fitting algorithm explained Sec. 6.4.3 are treated as obstacles by the path planner. In this work, for robustness, a nonlinear controller is used based on [80], [100] details of which we leave to the original papers. The path planner is implemented as a line follower. In case the distance of the platform to an outlier point cluster is less than a certain threshold, the component of the planned motion towards the obstacle is canceled. For example, when the robot is flying close to the gate as shown in Fig. 6.9b, the obstacles in front of the robot prevent the robot from flying further only in forward direction while lateral and vertical motions are still allowed. This effect is achieved by the onboard controller programmed to override the high-level commands given by the human operator when the robot is close to an obstacle. The accuracy and robustness of the obstacle avoidance feature is pertinent to the accurate knowledge of the robot state. When the obstacle is within the range and field of view of the lidar, accurate control and robust obstacle avoidance can be achieved due to the accuracy of range sensors. This is always the case along the cross-section of the tunnel and when the robot is close to a terminus of the tunnel.

Accurate control along the axial direction would be possible all the time if other sensor modalities such cameras (Chap. 5) or a Ultra-Wideband distance sensors (UWB) are also used. Otherwise, this motion can only be controlled by *directly* adjusting the robot pitch angle which is observable only to the onboard IMU. However, measurement noise, drag effect and inaccuracy in IMU bias estimates render axial motion control in this way unreliable. For these reasons, we equipped the DJI platform with an analog FPV camera for human operator awareness.

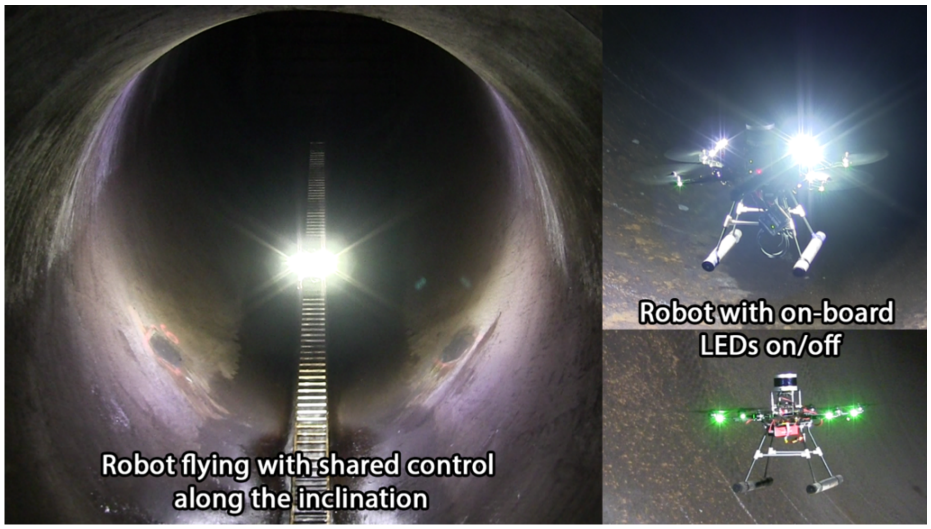

Robot flying with shared control along the inclination
Robot with on-board LEDs on/off

Figure 6.8: Photos of the DJI experiment platform flying inside a penstock at Center Hill Dam, TN. The onboard illumination is required for the testing. However both for safety concerns and for imagery collection, we kept the LEDs on.

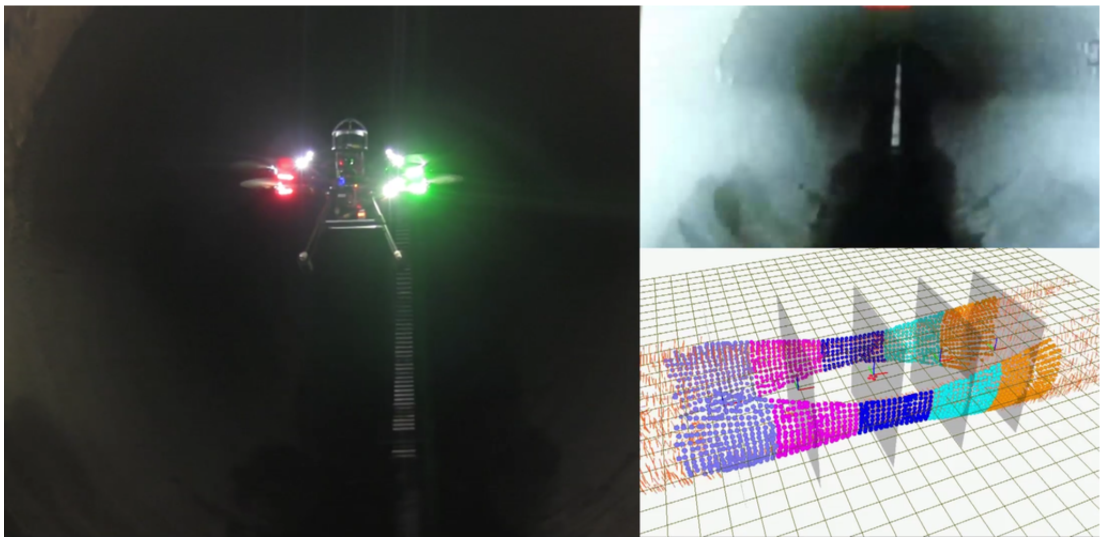
6.9 Experimental Results

In this section, we report the results of the experiments performed in a penstock at Center Hill Dam (CHD), TN. The DJI platform is shown from different views in Fig. 6.8. The goal of our experiments is to show that the robot can autonomously navigate to the end of the tunnel semi-autonomously, concurrently reconstructing the local environment. The human operator is provided with two alternative interfaces for controlling the robot. Through a GUI running on the base station, the operator can arm or disarm the robot, and also give high level position commands. Alternatively, the operator can control the robot through the radio control. In either case the operator has to continually adjust the pitch of the robot for motion along the tunnel axis.

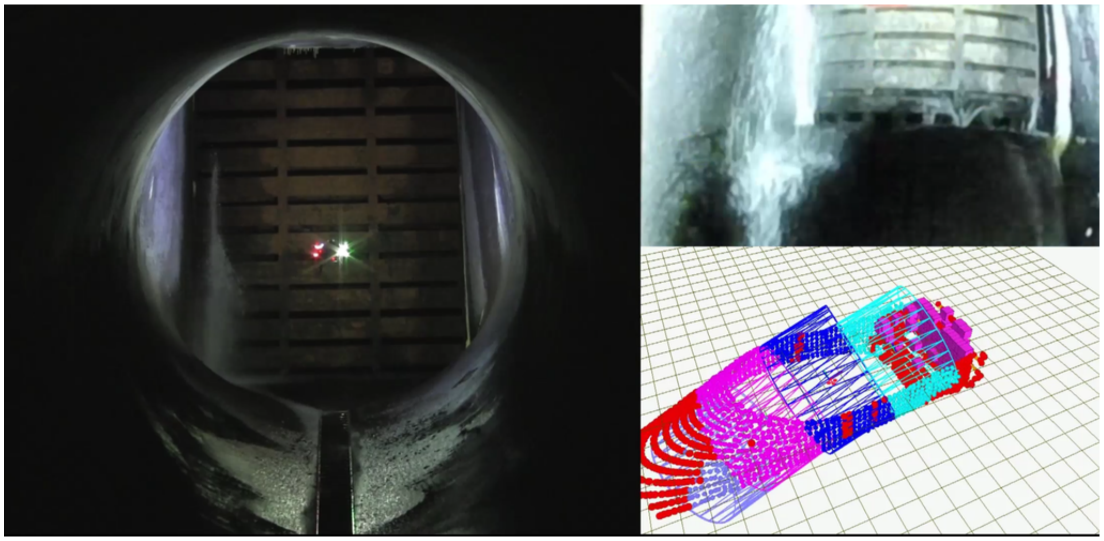
As the robot climbs along the inclined section, it gets out of line of sight of the operator. For real-time awareness of the operator we equipped the platform with an analog camera that works independent from the data link between the robot and the base station. Furthermore, in case the operator cannot perceive the field of depth from the monocular analog stream, or the video connection cuts off, we also implemented an onboard shared controller that can override the operator commands when required. For example, the robot can detect the gate at the upper terminal or any large obstacle blocking the tunnel and override the operator commands to avoid collision. Fig. 6.9 shows the video stream and screen shots from the RViz visualization tool while the robot is flying.

In all the experiments, the robot starts at an arbitrary point in the horizontal section and traverses the tunnel until the end of the inclined section. The one-way flight distance in the particular penstock is approximately 80 meters which our robot can traverse 2-3 times with a single battery pack. By traversing the penstock multiple times end-to-end, we show that the transition between the horizontal and the inclined sections are handled successfully. In some of the tests, the human operator stood closer to the opposite terminal of the tunnel and did not have the robot in his sight for most of the flight. The only source of awareness was the video stream from the onboard FPV camera. In these tests, we could assess the performance of the shared controller and the obstacle avoidance capability.

The flight starts with the operator commanding the robot to align itself with the tunnel centerline using either the GUI or the radio control. In order to reduce vortex formation which occurs when the robot is close to the walls, and also to maximize the image brightness, we command the robot to align with the center-line ($\mathbf{x}_{y,z,\psi} = \mathbf{0}$). Due to the robot state choice and the way we formulate the estimator, independent of the tunnel geometry, the robot always follows a path at a constant distance from the centerline. Then the operator can command the robot to go forward or backward along the tunnel by pitching it. The snapshots from the RViz visualization tool in Fig. 6.10-6.11 show the processed point cloud at different sections of the tunnel. In Fig. 6.12 it can be seen that the robot closely follows these commands except for oscillations and a constant offset in $\mathbf{x}_z$ due to imprecise controller parameter tuning. Lastly, the actual diameter of the penstock is 5.5 meters and our algorithm can estimate it within a 5% error.



(a) In the left image, the robot is about the start climbing through the inclined section. The top-right image shows the tunnel sloping up from the robot's perspective. The point cloud segments with distinct color and segment separator planes plotted in RViz.



(b) The left image is captured when the robot is ~ 4 meters from the gate. At this moment the onboard controller overrides the operator commands to fly the robot further forward since the gate is at a critical distance from the robot. The gate can be clearly seen in the top-right image. The purple blocks in the bottom-right image are the points classified as obstacles.

Figure 6.9: These figures show two instants captured during our tests at Center Hill Dam, TN. In both cases the robot is flying semi-autonomously in shared-control mode.

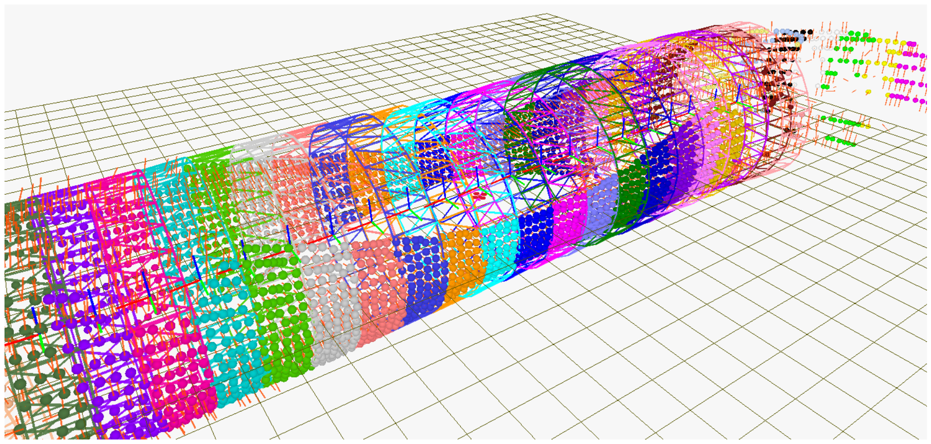


Figure 6.10: Screenshot from the RViz visualization tool showing the robot flying with shared control along the horizontal section of the tunnel. The colored point cloud and their corresponding meshes demonstrate the output of the segmentation and the cylinder fitting algorithms. The robot is shown with a red CAD model at the very center of the meshes.

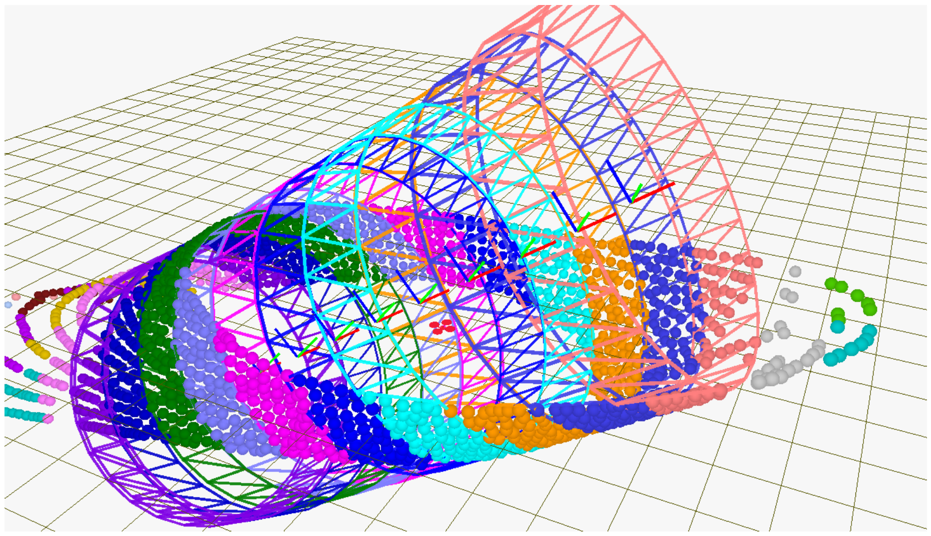
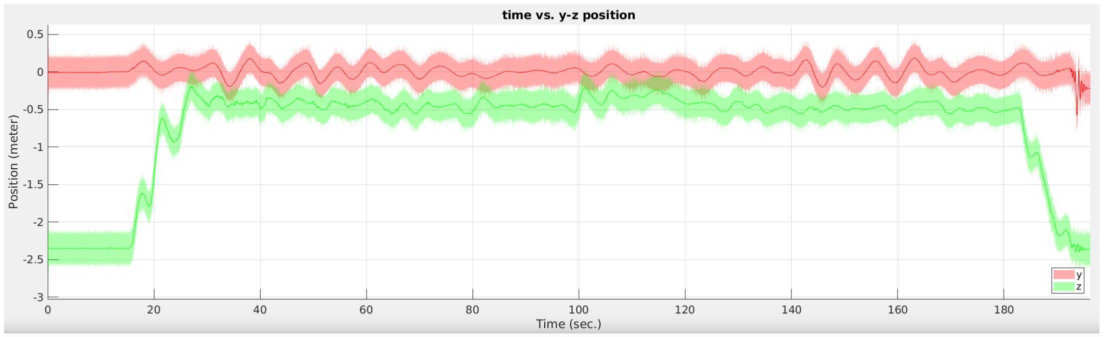
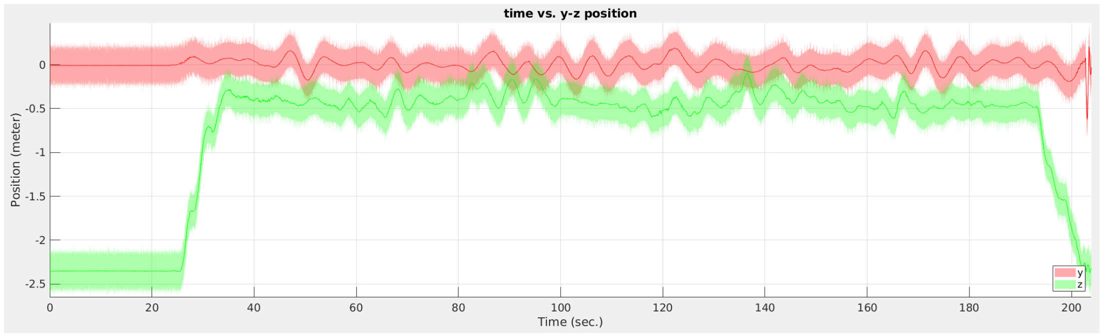


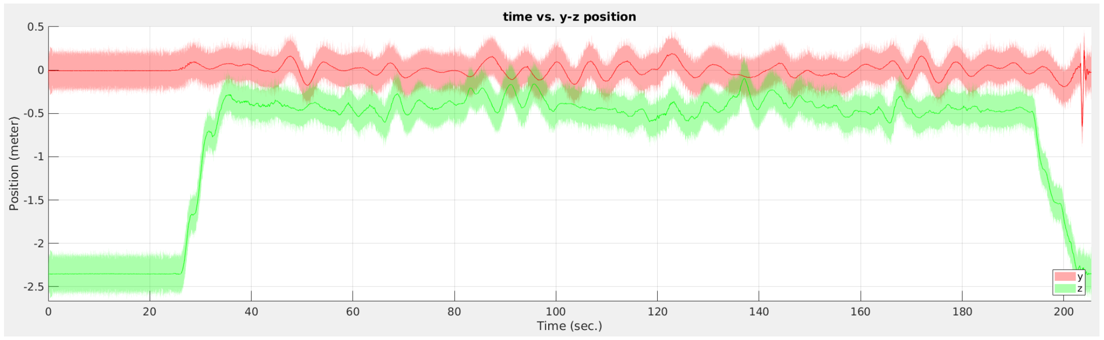
Figure 6.11: Screenshot from the RViz visualization tool showing the robot flying with shared control along the inclined section of the tunnel after $\sim$ 20 seconds after the take off. The estimation results of this experiment are presented in Fig. 6.12c. The algorithms does not require any modifications to handle the transition between the horizontal and the inclined section.



(a) Experiment #1



(b) Experiment #2



(c) Experiment #3

Figure 6.12: Vertical and lateral position, $\mathbf{x}_{y,z}$, of the robot while traversing entire penstock. Shades around the plots are the corresponding inflated standard deviations. In these tests, the robot was commanded to follow a straight path at a constant distant from the centerline. The offset in the z position is due to the inaccurate controller parameters.

Chapter 7

Modeling Tunnels as Smooth Generalized Cylinders

The work presented in this chapter focuses on the state estimation and local mapping of an MAV equipped with a 3D Lidar and an IMU for autonomous navigation inside penstocks with improvements over the approach presented in the previous chapter. We attain a higher level of reliability and robustness through modeling the tunnel as a parametric *piecewise-smooth-generalized-cylinder (PSGC)* [6], [48]. This improvement is of great importance due to possible safety issues unique to the confined penstock setting. Another significance of this model and the particular solution we propose is due to most of the range-based methods such as [114], [123], which rely on the presence of geometric cues, fail in the featureless tunnel setting.

Our claim is that *uniaxial*, *axisymmetric* and *featureless* tunnels can be mapped more accurately and robustly if represented as a smooth, connected set of parametric cylindrical segments compared to the common choice of raw point cloud representation or as a set of *disconnected* segments as in the previous chapter. We formulate the segment measurement model as a constrained optimization problem that respects the axisymmetric and smooth tubular geometry of the environment. To achieve this, this optimizer fits a chain of cylindrical segments defined by their *position*, *axis* and *radius* from the raw point cloud data. The optimizer imposes rotational and translational constraints in addition to penalizing other possible solutions that do not comply with a probability distribution derived from the raw point cloud data. A novel feature of this work is the use of spherical data analysis tools from the *directional statistical literature* for noise filtering and constrained optimization on $\mathbb{S}^2$ [17], [125]. Lastly, the segments are tracked by a constrained UKF running on a manifold [137] to filter the effects of noise temporally.

The method presented in this chapter differs from the previous approaches in various aspects. Our work described in Chap. 4 relies on a single 2D laser scanner retrofitted with a mirror setup to reflect a subset of its rays to the ceiling and floor. The approach of Chap. 5 uses two 2D laser scanners one of which is tilted downwards and the other facing forward. Unlike this approach, both methods assume a map of the environment is given, hence only perform localization. This is due to the fact that building a 3D map using only 2D range measurements in a tunnel setting is either infeasible or does not lend to reliable state estimation due to multi-modal nature of the problem. Lastly, in the previous chapter, we used the same sensor suite for local mapping and localization as in this approach. However, the segments were geometrically independent from each other, hence the optimizer was virtually free to converge at a state unconformable with the tunnel geometry. The approach of this chapter models a tunnel as a set of temporally and spatially smooth, connected segments.

We can list the fundamental limitations of our previous approaches that we overcome in this work as : *First*, we model the uniaxial, axisymmetric tunnel as a parametric, deformable *piecewise-smooth-generalized-cylinder* . This imposes constraints on the measurement model and the Kalman filter preventing the state from diverging even if the PSGC assumption fails. *Second*, we use the Watson distribution, a statistical tool that models axially symmetric distributions on $\mathbb{S}^p$, which has not been exploited by the robotics community to our knowledge. We use this tool to perform outlier elimination on $\mathbb{S}^2$ and to preserve consistency between local map segment orientations. The effect of this tool is more prominent especially when the cylindricality assumption does not hold. *Third*, we integrate sensory information for robot pose and local map estimation in a constrained UKF running on an appropriate

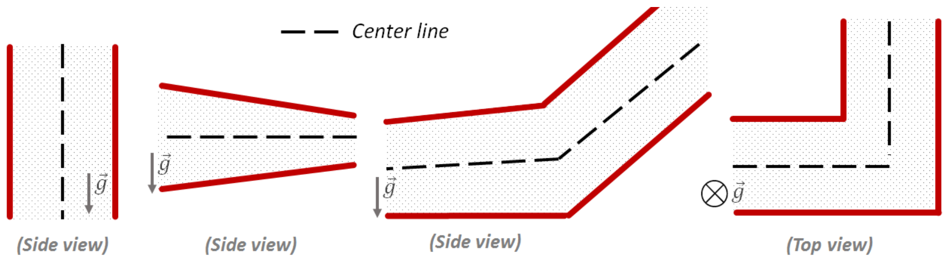


Figure 7.1: Illustrations for various piecewise-smooth-generalized-cylinder (PSGC) shaped environments. The center line curves of the right two topologies are piece-wise functions.

manifold.

Although we present results from a penstock, the proposed principles should not be thought to be peculiar to this infrastructure. They can be used in other environments which exhibit PSGC structure such as mine shafts, caves and sinkholes, highway tunnels and building corridors (Fig. 7.1).

7.1 Point Cloud Processing

Accurate surface normal estimation plays an important role in the proposed method since segment fitting process is formulated as an optimization problem with its cost function dependent on surface normals and curvatures. We use the method explained in Sec. 6.3.1 for surface normal and curvature estimation. This method calculates a scatter matrix of the k-nearest neighbors of a given point and takes the eigenvector corresponding to the smallest eigenvalue of this matrix as its surface normal. The ratio of the smallest eigenvalue to the summation of all three is taken as the curvature at the point of interest [76], [130]. Nearest neighbor search is sped up using a Kd-Tree. We also apply a voxel filter on the raw point cloud to reduce the point count and save CPU time.

7.2 Local Map as a Generalized Cylinder

A penstock has a well-defined geometry rarely encountered in indoor robotics applications. The simple geometry, however, brings challenges in mapping and estimation with. The representation of the map must be compatible with the inherent uncertainty in the robot axial position, and offer flexibility to conform to changes in the tunnel profile as the robot flies through it. For these reasons, we have chosen to model penstocks, as a generalization of an ordinary cylinder where the center axis can be any continuous spine and its radius may change along the axis (Fig. 7.1). The robot pose is represented with respect to the centerline. Since the centerline is a 1D manifold and can be parametrized with a single variable, it lends to modeling the axial position uncertainty. Binford [3] was the first to introduce the notion of a *generalized cylinder* in 1973. This became popular in robotics after Brooks' work in 1983 [6].

Due to the discrete nature of the lidar point cloud, we formulate the problem of estimating the *continuous* spine and radius functions in two stages. In the first stage, these two functions are estimated at equally spaced points along the tunnel axis. We refer to these points as *knots*. In the second phase, we use a Bézier interpolation to obtain the continuous local map. Since the mapping from the discrete map to the continuous map is unique, we represent the local map, without any loss, as an ordered set of knots, *i.e.* $L := \{K\}$. The number of knots of the local map may change throughout a flight depending on the visible penstock volume. Knot estimation and local map construction algorithms are explained in Sec. 7.4.

7.2.1 Map Knots

A knot is defined as

$$K := \left\{\mathbf{r}, \hat{\mathbf{t}}, \rho, P, \mathcal{W}\right\} \tag{7.1}$$

where $\mathbf{r} \in \mathbb{R}^3$ is its position, $\hat{\mathbf{t}} \in \mathbb{S}^2$ and $\rho \in \mathbb{R}^+$ are the spine tangent and the radius of the generalized cylinder at $\mathbf{r}$ respectively. P is the subset of the raw point cloud used to estimate the knot. As well as the Euclidean positions of its points, P also includes other features associated with each point such as surface normals and curvatures. Lastly, $\mathcal{W}$ is a Watson distribution fitted to a knot's associated point cloud which is used for filtering purposes as will be explained in the subsequent sections. Details of Watson distribution and how to fit a Watson distribution to a point cloud are explained in Sec. 7.3

While a discrete local map would suffice for state estimation and navigation, in the filtering context, a discrete map falls short for defining a distance metric. More specifically, the local map estimated at a given time step is updated by a UKF using the knot measurements at the following time step. The innovation step (measurement update) of a Kalman filter requires a proper distance metric defined between two measurements as well as the ability to uniquely associate them. Since the shape of the local map may change as the robot flies through the sections of the tunnel with different bending profiles and diameters, knots tracked by the filter and those given by the measurement model may not overlap *perfectly* which results in the data association to become ill conditioned. For this reason, the spine and radius functions are interpolated using a Bézier approximation with knots of the discrete local map being its control points. As will be discussed in the next section, an unambiguous data association and a valid distance metric on a Bézier curve can be defined under certain assumptions.

7.2.2 Bézier Interpolation

In the previous section we provided a *discrete* representation of the environment. However, given that a penstock is smooth, we can interpolate the sections between knots using a Bézier spline. This has several benefits such as the total curvature of the tunnel centerline provides a metric for the fitness of the local map, a 3D mesh of the tunnel can be reconstructed which then may be used for path planning etc. More importantly, two maps can be quantitatively compared using their corresponding Bézier splines to determine how *similar*

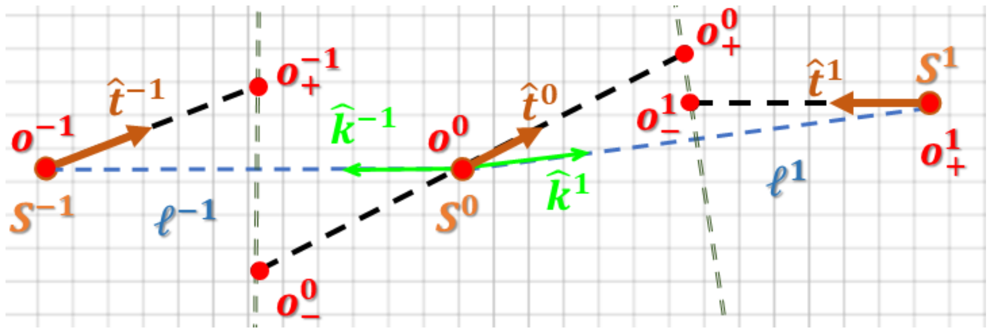


Figure 7.2: Generation of Bézier knots from tunnel axis tangents illustrated on a sample three segment local map.

or *close* they are to each other.

A Bézier spline is defined by an ordered set of knots, $\{K\}$. In our case, only the positions of the knots could have been used as the control points, $\{\mathbf{o}\}$ with $\mathbf{o}_i = \mathbf{r}_i$, which would give an acceptable approximation. However, we would like also to incorporate the axis tangent information for a more accurate interpolation. For this, we add two more control points to the opposite sides of each knot (only one to the terminal knots) which are obtained as

$$\mathbf{o}_{i^-} = \mathbf{r}_i + sign\,(i)\,\frac{\ell_i}{2}\,\hat{\mathbf{t}}_i\left(\hat{\mathbf{t}}_i^\top\hat{\mathbf{r}}_i\right)^{-1} \tag{7.2}$$

$$\mathbf{o}_{i^+} = \mathbf{r}_i + sign\,(i)\,\frac{\ell_{i^+}}{2}\,\hat{\mathbf{t}}_i\left(\hat{\mathbf{t}}_i^\top\hat{\mathbf{r}}_{i^+}\right)^{-1} \tag{7.3}$$

where ℓ and $\hat{\mathbf{r}}$ are defined as

$$\ell_i = ||\mathbf{r}_i - \mathbf{r}_{i^-}||_2 \tag{7.4}$$

$$\hat{\mathbf{r}}_i = \frac{\mathbf{r}_i - \mathbf{r}_{i^-}}{\ell_i}. \tag{7.5}$$

We adopt the $.^+$ and $.^-$ indexing operations previously defined in Sec. 6.1. With these additional control points, a local map consisting of n knots is approximated with a Bézier curve of $m = 3n - 2$ control points. The corresponding Bézier spline for the local map, $\acute{\mathcal{B}}_L$, is obtained as

$$\acute{\mathcal{B}}_L(t) = \sum_{v=0}^{m-1} \mathbf{o}_v \; B_{v,m}(t) \tag{7.6}$$

$$B_{v,m}(t) = \left(\begin{array}{c} m \\ \\ v \end{array} \right) t^v (1-t)^{m-v} \tag{7.7}$$

where $t \in [0,1]$, the second equation is the v^{th} Bernstein basis polynomial of degree m. Fig. 7.2 shows the control points on a sample map. Lastly, $\{\mathbf{o}\}$ is the ordered array of control points given as

$$\{\mathbf{o}\} = \left\{ \cdots, \mathbf{o}_{(-1)^+}, \mathbf{o}_{(-1)}, \mathbf{o}_{(-1)^-}, \mathbf{o}_{0^-}, \mathbf{o}_0, \mathbf{o}_{0^+}, \mathbf{o}_{1^-}, \mathbf{o}_1, \mathbf{o}_{1^+}, \cdots \right\}. \tag{7.8}$$

We implicitly assume that there is a mapping between the indices used in Equ. 7.6 and the control points.

We can define a distance metric between two knots as a function of their positions and axis tangents as

$$< K_i, K_j > = \alpha_p \left|\left| \mathbf{r}_i - \mathbf{r}_j \right|\right|_2 + \alpha_o \left(1 - \left| \hat{\mathbf{t}}_i^\top \hat{\mathbf{t}}_j \right| \right) \tag{7.9}$$

where α_p and α_o are weights assigned to position and orientation difference. Based on this metric we can define an operation which gives the closest point to a segment on a spline as

$$\acute{\mathcal{B}}(K) = \left\{ \acute{\mathcal{B}}(t^*) \;\middle|\; t^* = \underset{t}{\operatorname{argmin}} < \acute{\mathcal{B}}(t), K >, t \in [0,1] \right\}. \tag{7.10}$$

It should be noted that we used a point on the spline, *i.e.* $\acute{\mathcal{B}}(t)$, rather than a knot in the distance calculation. But this is a valid operation, since a point on a spline, in addition to a position, has a tangent which equals

$$\hat{\acute{B}}(t) = \frac{d\acute{\mathcal{B}}(t)}{dt} \left|\left| \frac{d\acute{\mathcal{B}}(t)}{dt} \right|\right|_2^{-1} \tag{7.11}$$

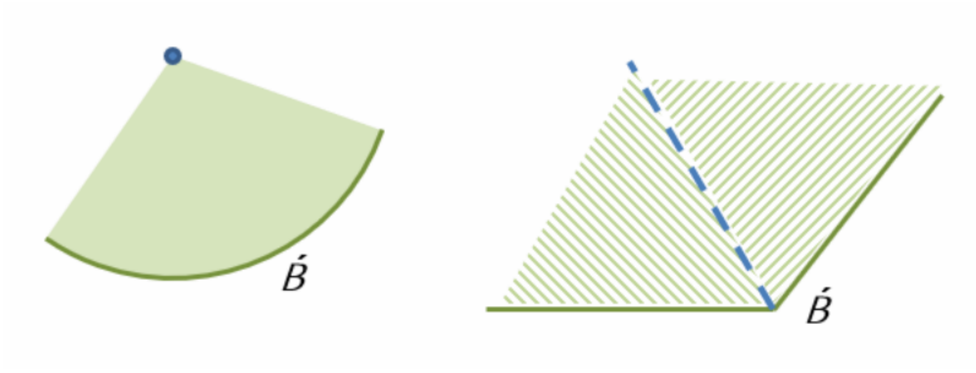


Figure 7.3: These illustrations show two cases where the closest point on a spline (dark green) is not unique. *(Left)* The set of closest points to the blue query point located at the center of the circular arc is the whole arc. *(Right)* There are two closest points to all query points along the dashed blue line one on each linear section of the curve.

We will use this operation during the calculation of innovation in the measurement update step of our UKF.

A careful reader would notice that the operation defined in Equ. 7.10 might not yield a unique solution for certain spline shapes and query points. For example consider the case that the spline is a circular arc and the query point is the center of the corresponding circle. Then the result of this operation becomes a *range* of points. In general, if the query point is on the boundary surface (curve, if 2D) of the generalized Voronoi diagram of the Bézier spline, the closest point on the spline is not unique. Sample cases are illustrated in Fig. 7.3.

Within a filtering context, however, the range of points can be reduced to a single point using prior information. The optimization process explained in Sec. 7.4.1 estimates knots in a specific order and one at a time. Corresponding points of these knots along the Bézier spline that represents the most recent local map is found using the operation defined in Equ. 7.10 which are then used in the measurement update of our UKF as discussed in Sec. 7.5.3. Since the knot measurements are provided in a certain order, their correspondences along the spline must follow the same order. The ordering of the correspondences is defined by their t coordinates (Equ. 7.6). In most cases, multiple solutions can be reduced to a single one exploiting this constraint as shown in an example on Fig. 7.4. When disambiguation is not possible, we simply discard that knot measurement and perform the Kalman measurement

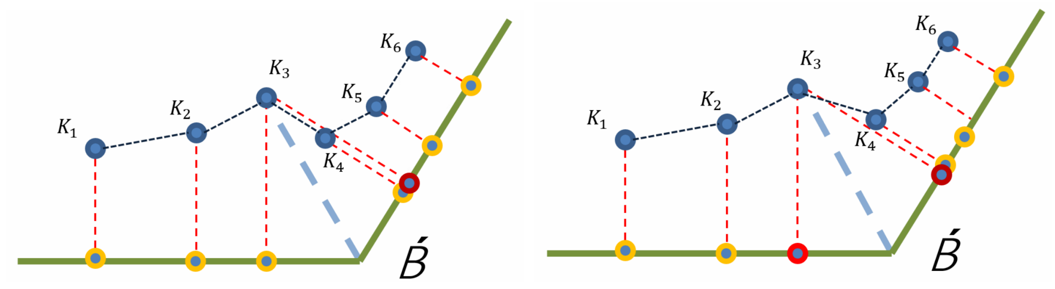


Figure 7.4: These illustrations show two cases where the closest point on a given spline to a knot is not unique. In the case depicted on the left, by exploiting the order of knot measurements $\{K\}$, one of the two closest points (red) to K_3 is eliminated. On the right illustration, since the relative position of K_3 and K_4 changes, the ordering of measurements does not suffice to disambiguate the closest point to K_3.

update with the rest of the measurements. We can liken this to an image feature being matched to multiple features across frames in a visual odometry application. The wisest strategy would be to discard such image features so as to prevent the camera state from diverging.

7.3 Distributions on $\mathbb{S}^{d-1}$

The statistical tool of choice in regression and estimation problems is *normal distribution* which is originally designed for Euclidean space. Under certain circumstances and with appropriate assumptions, random variables on manifolds can be analyzed with this tool. However we believe that the tools from the directional statistics literature specifically designed for spherical data should be used for data on spherical manifolds. In particular, unit vectors with widely spread uncertainties are often analyzed using normal distributions with self-wrapping overlooked. Because of this, we use Watson distribution on $\mathbb{S}^2$ (Section 9.4 in [17]) to describe the set of possible tunnel axes as well as to perform outlier elimination.

Watson [2] and Bingham [4] distributions are the two mostly used spherical distributions in the literature. Watson distribution (WD) is a special case of Bingham distribution (BD) where the spread is symmetric with respect to the distribution mode. Due to its formulation (Sec. 7.3.2) WD can represent *only* axisymmetric distributions limiting its use cases. On the other hand numerical calculations involving WDs are easier to work with compared to the more complex BDs.

7.3.1 Literature Review on Spherical Distributions

Scenarios which include unit vectors are the natural application areas of spherical distributions such as surface normals in point cloud processing, image feature direction in computer vision and orientation vectors in control and estimation. In a robotic manipulation scenario [46] uses WD to represent orientations that the robot hand can grasp an object. In a supervised learning scheme, the researchers collect feasible hand orientations of a human teacher which are then encoded as a girdle distribution. The controller is designed to exclude the *don't-care* orientations which is naturally encoded by the axisymmetric equatorial distribution.

Although designing spherical distributions is practically straightforward, the normalization factor makes their use prohibitively difficult due to hard to compute special functions and numerically unstable integrals. In a study concerned about such fundamental aspects of multivariate high dimensional WD, [104] proposes novel, numerically accurate, easy to compute approximations to maximum-likelihood estimates. This work offers a theoretical contribution to diametrical clustering used for gene-expression analysis.

BD is a generalization of WD which can represent anisotropic spreads on spheres [4]. Due to this capability, it found more use cases such as in computer vision [24], object detection [97], orientation estimation [89], [99], [109], [126] and pose estimation [108], [132]

In another robotics manipulation application, [89] uses BD to analyze point cloud for object orientation estimation. In this work, the researchers fit a Bingham mixture model to spherical surface orientation data extracted from a given raw point cloud data. This is then used for sampling orientations from a hypersphere which are tested against a given model for object pose estimation.

In a computer vision application, [24] uses BD for representing image feature point coordinates and uncertainties in projective space which are then used during feature matching and camera relative pose estimation. [97] proposes a new system for object detection from RGB-D camera data. This work uses a BD for scoring possible model orientation hypotheses to prune high number of possibilities to speed up their detection pipeline.

Orientation tracking is a fundamental problem in robotics. Most existing work uses traditional methods such as Kalman Filter without considering the periodic nature of rotations which results in poor performance. [99] proposes a recursive estimator for tracking 2D orientation based on BD. The authors later improve their work and extend it to work for 3D rotations using a UKF [126]. The authors compare their results particularly for high noise scenarios against classical UKF with a Gaussian noise model and Particle Filters (PF) with various particle counts. Due to its ability to handle periodicity in orientation, the new approach using BD outperforms all other methods.

In their work Gilitschenski *et al.* [108] uses BD to represent rigid body transformations on $SE(2)$. This work assumes that the rotation and translation random variables of a robot pose are sampled from an exponential distribution (definition 1 in the paper). The proposed distribution relates the position uncertainty of a robot to the parameters of a BD that represents its orientation uncertainty. In doing this, [108] uses dual quaternions to handle problems arising due to equivalence of antipodal points under BD.

The robotics literature uses the two spherical distributions for various purposes such as orientation tracking, pose estimation and accurate uncertainty representation. Our work contributes to the literature for similar considerations such as model fitting and noise filtering as explain in the subsequent sections.

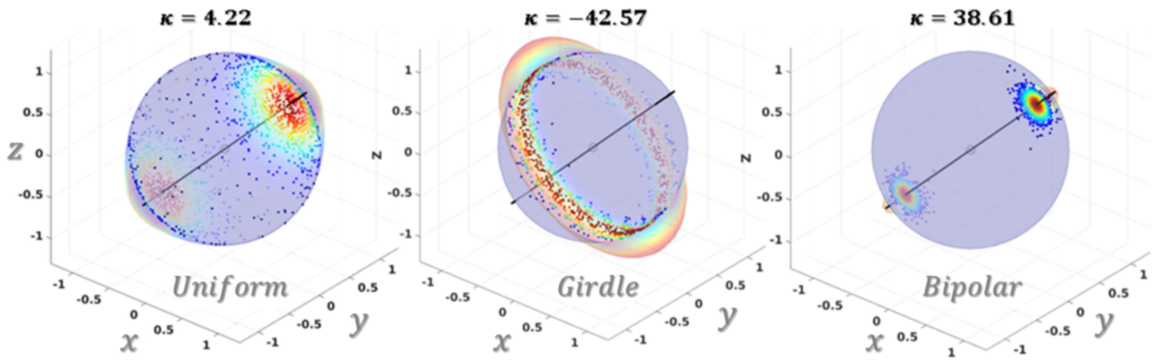


Figure 7.5: Synthetic data on $\mathbb{S}^2$ sampled from uniform, girdle and bipolar distributions.

7.3.2 Watson Distribution Formulation

The density of a Watson distribution which gives the probability density over the unit vectors $\hat{\mathbf{x}}$ is written as

$$\mathcal{W}\left(\pm\hat{\mathbf{x}}, \mu, \varsigma\right) = M\left(\frac{1}{2}, \frac{d}{2}, \varsigma\right)^{-1} exp\left(\varsigma\left(\mu^{\top}\hat{\mathbf{x}}\right)^2\right) \tag{7.12}$$

where μ is the mode, ς is the concentration parameter and $M\left(1/2, d/2, \varsigma\right)$ is the Kummer function. This function is defined as

$$M\left(\frac{1}{2}, \frac{d}{2}, \varsigma\right) = \beta\left(\frac{d-1}{2}, \frac{1}{2}\right)^{-1} \int_{-1}^{1} e^{\varsigma t^2}(1-t^2)^{\frac{(d-3)}{2}} dx. \tag{7.13}$$

where $d = 3$ for $\mathbb{S}^2$ and β is the beta function. The Watson distribution takes the form of a bipolar distribution for $\varsigma > 0$ and becomes a girdle distribution for $\varsigma < 0$ [17]. Distributions for different synthetic data are shown in Fig. 7.5.

As will be discusses in the following sections, the segment fitting process is designed as an optimization problem which minimizes a combination of various cost functions. One of these cost functions is the likelihood of a given direction vector being the *mode* of the WD fitted to the surface normals. It should be noted that the likelihood of a unit vector with respect to a WD and the likelihood of a unit vector being the mode of a WD are not necessarily the same. When $\varsigma \geq 0$, the likelihood of a point, $\hat{\mathbf{x}}$, being the mode is the same as the distribution itself, *i.e.*

$$\mathcal{W}_m\left(\hat{\mathbf{x}}, \mu, \varsigma\right) = \mathcal{W}\left(\hat{\mathbf{x}}, \mu, \varsigma\right) \tag{7.14}$$

However when $\varsigma < 0$, the mode likelihood becomes (omitting the parameters of the Kummer function for brevity)

$$\mathcal{W}_m\left(\hat{\mathbf{x}}, \mu, \varsigma\right) = \frac{2}{M} \int_0^{\pi} \exp\left\{\varsigma\left(1 - \left[\mu^\top \hat{\mathbf{x}}\right]^2\right) cos\left(\theta\right)^2\right\} d\theta \tag{7.15}$$

which is an integral over the great circle perpendicular to $\hat{\mathbf{x}}$. Since these integrals are prohibitively time demanding, we use lookup tables in our implementation.

7.3.3 Watson Distribution Fitting

An important novelty of this work is the use of spherical distributions to filter outliers and suppress their effect in the state estimation. Details of these will be discussed in the subsequent sections. In this section, we explain the Watson fitting process to a set of unit vectors, which are the surface normals in this particular application. In addition to fitting a WD using only data points, we also present how to incorporate priors in the fitting process where the priors are WDs as well.

The WD corresponding to a set of data points, $\{\hat{\mathbf{x}}\}$, in the maximum likelihood sense, can be found through an eigenvalue analysis on the scatter matrix of the data points. The scatter matrix is obtained as

$$\mathbf{S} = \sum_i \hat{\mathbf{x}}_i^\top \hat{\mathbf{x}}_i. \tag{7.16}$$

According to the relative values of the eigenvalues of $\mathbf{S}$, the shape of the distribution can take three forms. These cases can be listed as (Table 10.1 [17])

- $e_1 \simeq e_2 \simeq e_3$: uniform

- $e_1 \gg e_2 \simeq e_3$: bipolar
- $e_1 \simeq e_2 \gg e_3$: girdle.

The mode of the distribution is determined according to its shape. If the distribution is uniform, either of the eigenvectors of $\mathbf{S}$ can be chosen as the mode. For the bipolar case, the eigenvector corresponding to the largest eigenvalue, and for the girdle case the eigenvector corresponding to the smallest eigenvalue is chosen as the distribution mode.

We can say very little about the spread of the distribution by only knowing its shape and mode. On the other hand the concentration parameter, ς, explains the spread of the distribution as well as its shape. $\varsigma > 0$ for the bipolar case, $\varsigma < 0$ for the girdle case and $|\varsigma| < 1$ for a uniform shaped WD. As can be noticed, only for very large or very small concentration values the distinction is obvious leaving the other regions gray. For this reason, we exploit the prior knowledge about the data points and constrain the analysis only to the girdle case.

The data points in our case are the surface normals estimated from the raw point cloud data. Due to the shape of the tunnel, we expect the corresponding WD to always have a girdle shape with a large negative concentration value. This expectation fails when the robot flies around the tunnel sections that do not have a generalized cylindrical shape such as at distal ends or near the scaffolding. Whether this prior assumption holds can be checked at the end of the fitting process. In the positive case, mode likelihood, $\mathcal{W}_m$, and the concentration parameter, ς, will take large and small values respectively.

Determining the concentration parameter is much more difficult than estimating the mode since the concentration parameter appears in the Kummer function which includes special functions (Equ. 7.12). Since solving for these functions is prohibitively time consuming for real-time performance, we fit a polynomial to the equation below, originally given in [17], which describes the relation between the concentration parameter and the eigenvalue chosen as explained above. This equations is

$$
\begin{aligned}
D_p(\varsigma) &= e_3^\top \mathbf{S} e_3 \\
&= \frac{\int_0^1 t^2 exp\left(\varsigma t^2\right) dt}{\int_0^1 exp\left(\varsigma t^2\right) dt}
\end{aligned} \tag{7.17}
$$

(Equ. 10.3.31/32 in [17] with $p = 3$). The polynomial is fitted to the inverse of $D_p(\varsigma)$.

A careful reader would notice that the concentration parameter is a function of the data points and the distribution mode. Hence the MLE solution of a WD can be written as

$$p(\mu, \varsigma|P) = p(\mu|P) \tag{7.18}$$

reducing the complexity of the problem value of which will be obvious in the rest of the section.

The WD fitted to a given point cloud summarizes the shape of the environment with just two parameters. This is a very powerful tool to compress large point clouds and save from compute power and memory. However, a WD fitted using only data collected at a single time frame is temporally and spatially isolated which, in case the MLE estimate is inaccurate, may fail the downstream processes in the estimation pipeline. To remedy this, we would want to feed the fitter with priors which would collectively be less likely to give an estimate off by large. These priors might be past measurements from the same region of the tunnel or others from the neighboring regions along the tunnel. In this case Equ. 7.18 should be modified as

$$p(\mu, \varsigma|P, \{\mathcal{W}\}) = p(\mu|P, \{\mathcal{W}\}) \tag{7.19}$$

where $\{\mathcal{W}\}$ is the set of prior WDs.

In order to find the MLE WD in Equ. 7.19, we use a fixed, large number of hypotheses (i.e. normals) uniformly placed on a spherical polyhedron of unit radius obtained by recursively subdividing an icosphere as shown in Fig. 7.6. The hypotheses are the vertices of the

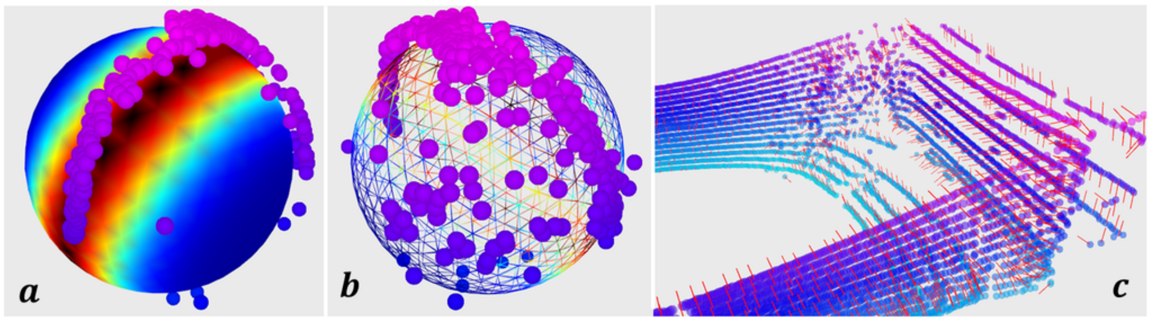


Figure 7.6: Sample surface normal data from Center Hill Dam experiments and the corresponding girdle distribution, i.e. $\varsigma < 0$, fit with the method explained in Sec. 7.4. *(a)* Likelihoods are color-coded with blue and black corresponding to low and high values respectively. *(b)* Vertices of the mesh are used as hypothesis. *(c)* Point cloud from which the distribution is obtained.

triangular faces. A hypothesis, h, is scored as

$$-\sum \left\{ \kappa_i^2 + \left(h^\top \hat{\mathbf{n}}_i \right)^2 \right\} + \sum \gamma_j \, log \left(\mathcal{W}_{m,j}(h) \right) \tag{7.20}$$

where the first summation runs over all of the surface normals, $\hat{\mathbf{n}}_i$, and curvatures, κ_i, of the specified knot, and the second summation runs over all of the prior WDs. As will be explained in Sec. 7.4, the priors are the WDs of all the knots of the most recent local map. The multiplicative factor, γ_j, is a decreasing function of the distance between the knots of concern and its neighboring knots. We can liken this to averaging a 1D signal.

The hypothesis with the highest score is chosen as the mode, μ_*, of the corresponding distribution. The effective eigenvalue, λ, and the weight of a given point, w_i are calculated as

$$\lambda = \frac{\sum w_i |\hat{\mathbf{n}}_i^\top \mu_*|}{\sum w_i} \tag{7.21}$$

$$w_i = exp\left(-\kappa_i^2 - \left(\hat{\mathbf{n}}_i^\top \mu_* \right)^2 + \sum \gamma_j \, log \left(\mathcal{W}_j(\hat{\mathbf{n}}_i) \right) \right). \tag{7.22}$$

where the summation runs over all of the knots in the local map. Finally we find the concentration parameter, ς_*, by plugging λ into the inverse of Equ. 7.17. If $-5 < \varsigma$ we conclude that the surface normal data is extremely noisy and terminate the process. This corresponds to either a uniform distribution if $|\varsigma|$ is small, or a bipolar distribution if $\varsigma > 0$.

The exponential function is a time consuming operation which may render the real-time performance impossible. For this reason, we use a polynomial approximation of the form

$$exp\,(x) \approx (a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + x \tag{7.23}$$

$$(a_5 + x(a_6 + x(a_7 + x(a_8 + x)))))))))a_8 \tag{7.24}$$

which is valid for $x \in [0, 1]$. Numbers outside this limit are factored into precomputed integer powers of e and a number in this range.

It can be argued that the mode of a girdle distribution could be directly used as the local centerline tangent. However, we think that a single distribution, such as Watson, compresses the point cloud information into two parameters, μ and ς, which blurs most of the details in the raw data. Hence, we use the Watson distribution as one of the factors that define a cost function and estimate the most likely centerline tangent through a more comprehensive optimization procedure which we present in Sec. 7.4.1.

7.4 Knot Estimation

This section describes the knot estimation process formulated as a constrained optimization on $\mathbb{S}^2 \times \mathbb{S}^2$. The input to this process is a preprocessed point cloud from the lidar. The preprocessing includes downsampling, surface normal and curvature estimation as explained in Sec. 7.1. The output of this process is an ordered set of knots which are then fed as measurements to a UKF.

Similar to the point cloud processing and segmenting process in Chap. 5, the raw point cloud data is first segmented into clusters. Every knot is estimated based on their associated point cloud clusters. The point-to-cluster distance is simply the point-to-plane distance where the plane passes through the knot position, $\mathbf{r}$, and is perpendicular to the knot tangent, $\hat{\mathbf{t}}$. The point cloud, P_i, associated with a given knot, K_i, is obtained as

$$P_i = \left\{ p_j \in P \mid \left| \hat{\mathbf{t}}_i^\top \left(p_j - \mathbf{r}_i \right) \right| \leq \ell_i \right\} \tag{7.25}$$

where P is the input point cloud, and ℓ_i is the chord length between the knots with indices i and i^- as explained in Sec. 7.2.2. The resultant point cloud associated with the given knot includes both the points and other features associated with them such as normals and curvatures.

7.4.1 Optimizer Definition

The optimizer minimizes a cost function of the point coordinates, their surface normals and curvatures. Its output is position and tangent estimates of a given knot and their uncertainty estimates. We apply different weights to points according to their various properties to robustify the optimizer. Since the walls of a penstock are expected to be smooth, points with high curvature are penalized. Such points are usually due to scaffolding, gates and doors at the terminals of a penstock, or other obstacles such as equipment and human operators. Each point is also weighed by comparing its normal against the Watson distribution fitted in the previous time step of the knot being optimized for as well as WDs of its neighboring knots.

The cost function is defined as

$$C = C_{\mathbf{r}} + C_{\hat{\mathbf{t}}} + C_{\mathcal{W}} \tag{7.26}$$

where the terms are

$$C_{\mathbf{r}} = \frac{1}{\sum w_i} \sum w_i \left| \bar{\rho} - \left|\left| \Delta p_i \right|\right|_2 \right| \tag{7.27}$$

$$C_{\hat{\mathbf{t}}} = \frac{1}{\sum w_i} \sum w_i \left(\hat{\mathbf{n}}_i^\top \hat{\mathbf{t}} \right)^2 \tag{7.28}$$

$$C_{\mathcal{W}} = -\log \left(\mathcal{W}_m \left(\hat{\mathbf{t}}, \mu, \varsigma \right) \right) \tag{7.29}$$

with all the vectors written in the lidar frame, $\mathcal{V}$, and summations running over all the points associated with the knot being optimized for. For clarity we omit segment indices, but the reader should be aware that C is written for a particular knot, K_i, using its corresponding point cloud $P_i \subset P$ and spherical distribution, $\mathcal{W}_{m,i}$.

The terms in the preceding equations penalize respectively *(1)* the discrepancy between each point and the knot axis in the point-to-line distance sense and its mean value (*i.e.* mean radius), *(2)* incompatibility between the surface normals and the knot axis, and *(3)* the difference between the knot axis and the mode of the corresponding segment's Watson distribution. The knot axis, $\hat{\mathbf{t}}$, gives an estimate of the tunnel centerline tangent at the knot position, $\mathbf{r}$. Lastly, point weights, w_i, are calculated as in Equ. 7.22.

The mean radius $\bar{\rho}$ is calculated as

$$\bar{\rho} = \frac{1}{\sum w_i} \sum w_i \left|\left|\Delta p_i\right|\right|_2 \tag{7.30}$$

where summations run over all the points of the knot optimized, Δp_i is the shortest vector from point p_i to the approximate tunnel centerline, $\mathbf{r} + \alpha\hat{\mathbf{t}}$ for $\alpha \in \mathbb{R}$, and is given by

$$\Delta p_i = (\mathbf{I} - \hat{\mathbf{t}}\ \hat{\mathbf{t}}^\top)(p_i - \mathbf{r}). \tag{7.31}$$

The solution to the fitting problem can be written as

$$\left\{\mathbf{r}_*, \hat{\mathbf{t}}_*\right\} = \underset{\mathbf{r},\hat{\mathbf{t}}}{\operatorname{argmin}}\ C. \tag{7.32}$$

However this choice of free parameters does not impose any constraints between the adjacent knot positions, hence may undesirably cause $\mathbf{r}$ end up at a far point from its neighboring knots. For this, we rewrite this parameter as

$$\mathbf{r}_i = \mathbf{r}_{i^-} + \ell_i \hat{\mathbf{r}}_i \tag{7.33}$$

where ℓ_i and $\hat{\mathbf{r}}_i$ are the chord length between adjacent knots and its direction vector as given in Sec. 7.2.2 respectively. Fig. 7.2 shows these two variables on a sample map. We can rewrite the solution as

$$\left\{\hat{\mathbf{r}}_*, \hat{\mathbf{t}}_*\right\} = \underset{\hat{\mathbf{r}},\hat{\mathbf{t}}}{\operatorname{argmin}}\, C. \tag{7.34}$$

Since ℓ is a constant, the knot position estimation problem is transformed into finding the best knot direction. This way, both free parameters are constrained to $\mathbb{S}^2$ preventing the knot position from ending up in an undesirable state. In a real-life scenario, a penstock does not bend sharply. Another advantage of this particular parameter choice that this type of a constraint can be easily imposed.

This reparametrization is applicable only to knots which have an anterior knot, *i.e.* K_{i^-} is defined. Thus, for the root knot, K_0, the optimizer with its original form given in Equ. 7.32 is applied. In order to ensure that the closest point along the centerline to the body origin is the knot position, we apply the constraint $\mathbf{r}_0^\top \hat{\mathbf{t}}_0 = 0$.

The optimizer uses the Levenberg-Marquardt method with adaptive step size to minimize C. The update is written as

$$\begin{bmatrix} \Delta\hat{\mathbf{r}} \\ \Delta\hat{\mathbf{t}} \end{bmatrix} = \left(\mathbf{J}\,\mathbf{J}^\top + \lambda\, diag\left(\mathbf{J}\,\mathbf{J}^\top\right)\right)^{-1}\,\mathbf{J}C \tag{7.35}$$

where $\mathbf{J} = \nabla C$ is the gradient of the cost function and $\lambda > 0$ is the damping parameter that determines the step size. Each update is first projected onto the tangent space at the most recent state as

$$\hat{\mathbf{r}}_{new} = \hat{\mathbf{r}} + \left(\mathbf{I} - \hat{\mathbf{r}}\,\hat{\mathbf{r}}^\top\right)\,\Delta\hat{\mathbf{r}} \tag{7.36}$$

$$\hat{\mathbf{t}}_{new} = \hat{\mathbf{t}} + \left(\mathbf{I} - \hat{\mathbf{t}}\,\hat{\mathbf{t}}^\top\right)\,\Delta\hat{\mathbf{t}} \tag{7.37}$$

followed by projection onto the unit sphere (not shown). The radius is not included in the optimization since it can be directly calculated from the knot position and tangent estimates as given in Equ. 7.30 once the optimization converges.

The covariance of the a knot is estimated based on the results of the optimization. The approximate Hessian scaled with the residual error, which equals the cost evaluated at the point of convergence, is taken as the uncertainty in the knot position and tangent. This can be written as

$$\mathbf{H} = \mathbf{J}\,\mathbf{J}^{\top} + \lambda\, diag\left(\mathbf{J}\,\mathbf{J}^{\top}\right) \tag{7.38}$$

$$\boldsymbol{\Sigma}_{\mathbf{r},\hat{\mathbf{t}}} = \mathbf{H}^{-1}\, C \tag{7.39}$$

Since the knot radius is not included in the optimization, its uncertainty needs to be calculated separately. The full covariance of a knot after incorporating the radius uncertainty becomes

$$\mathbf{H}_{full} = \begin{bmatrix} \mathbf{H} & \mathbf{J} \\ \mathbf{J}^{\top} & 1+\lambda \end{bmatrix} \tag{7.40}$$

$$\boldsymbol{\Sigma}_{K} = \mathbf{H}_{full}^{-1}\, C \tag{7.41}$$

where λ is the damping coefficient in Equ. 7.35 calculated at the last iteration of the optimization. However, this does not reflect the effect of normalization of the knot chord directions and tangents. This effect can be obtained by multiplying the full Hessian matrix by the following projection matrix

$$\boldsymbol{\mathcal{T}} = \begin{bmatrix} \frac{\partial \eta(\hat{\mathbf{r}}_{new})}{\partial \hat{\mathbf{r}}}\ell & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \eta(\hat{\mathbf{t}}_{new})}{\partial \hat{\mathbf{t}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \tag{7.42}$$

where $\eta\left(x\right) = \frac{x}{||x||_2}$ is the normalization function, ℓ is the chord length of the knot. For the root knot, K_0, this matrix becomes

$$\boldsymbol{\mathcal{T}} = \begin{bmatrix} \ell\,\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \eta(\hat{\mathbf{t}}_{new})}{\partial \hat{\mathbf{t}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix}. \tag{7.43}$$

The gradient of the normalization function is

$$\frac{\partial \eta\left(\mathbf{x}\right)}{\partial \mathbf{x}} = \frac{1}{\left|\left|\mathbf{x}\right|\right|_2^3}\left(\mathbf{I}\left(\mathbf{x}^\top \mathbf{x}\right) - \mathbf{x}\mathbf{x}^\top\right) \tag{7.44}$$

Finally the Hessian after correction becomes

$$\mathbf{H}_* = \boldsymbol{\mathcal{T}}\mathbf{H}\boldsymbol{\mathcal{T}}^\top. \tag{7.45}$$

The knot estimation process starts with the root knot, K_0, and continues both in forward and backward directions. The estimation in the two directions are independent of each other and one can terminate earlier than the other depending on three criteria. *First*, if the Watson distribution fitted to a knot's point cloud has a concentration value of greater than $\tau_\varsigma = -5$, the estimation stops. This occurs when the robot is flying close to an end of the tunnel or to an obstacle blocking the tunnel. For example, the surface normals at the gate point at random directions (gate is not a planar) which are mostly inconsistent with the rest of the normals. *Secondly*, if the mode likelihood of the knot tangent is less than a certain threshold, the estimator terminates. *Lastly*, if the current and the previous knot tangents are significantly different, then we again terminate the process. These conditions are checked separately for forward and backward map construction. Hence the robot may stop building a local map in forward direction and continue building a longer local map in the reverse direction which increases the robustness of the proposed method.

7.5 Filter Design

In this section we explain the details of a constrained UKF that tracks a body-centered local map of the environment along with the gravity vector and sensor biases. The robot state which is defined with respect to the centerline of the local map is also tracked *indirectly.* The proposed UKF implements two measurement updates which are for gravity vector correction and local map knot estimation. Based on the tunnel volume visible to the lidar, the state vector expands and shrinks by addition and removal of knots.

Nonlinearities due to components of the state vector prohibit the use of a linear Kalman Filter. These are the unit vectors in the state which are the gravity vector, knot chord directions and tangents. Although Extended Kalman Filter (EKF) offers a solution for nonlinear systems, linearization of the process and the measurement models can get very complicated and prohibitively time consuming for real-time performance. Hence, we prefer using a UKF rather than many other alternatives of the Kalman Filter family due to its superior ability to handle nonlinearities. The seminal work [137] explains that a UKF can handle nonlinearities up to third order.

7.5.1 State Vector

The system state vector writes as

$$\mathbf{x} = \left[\hat{\mathbf{g}}, \mathbf{b}, \mathbf{v}, \mathbf{r}_0, \hat{\mathbf{t}}_0, \rho_0, ..., \mathbf{r}_i, \hat{\mathbf{t}}_i, \rho_i, ...\right] \tag{7.46}$$

where $\hat{\mathbf{g}} \in \mathbb{S}^2$ is the gravity vector direction, $\mathbf{b} = [\mathbf{b}_{\mathbf{a}},\ \mathbf{b}_{\boldsymbol{\omega}}]$ where $\mathbf{b}_{\mathbf{a}}, \mathbf{b}_{\boldsymbol{\omega}} \in \mathbb{R}^3$ are the accelerometer and gyroscope biases, $\mathbf{v} \in \mathbb{R}^2$ is the translational velocity of the local map along the local tunnel cross-section. Knot positions and tangents are defined in the lidar frame, $\mathcal{V}$. Gravity and IMU biases are defined in the body frame, $\mathcal{B}$, which we define to be coincident with the IMU frame, $\mathcal{I}$. Since these two components are direction vectors and the only offset between the lidar and body frames is pure translational, we can assume that all components are defined in the lidar frame, $\mathcal{V}$. Lastly, the velocity vector, $\mathbf{v}$, is defined in the local map frame, $\mathcal{L}$, and is constrained to the $\hat{\mathbf{y}} - \hat{\mathbf{z}}$ plane.

The size of $\mathbf{x}$ changes depending on the visible range of the tunnel by addition and removal of knots. The *unconventional* state vector $\mathbf{x}$ has the *minimal* set of variables that *fully* represents the robot state and the local map in this unique environment. Although the robot state is not explicitly included in $\mathbf{x}$, it can be inferred from the local map as explained in Sec. 7.5.5. Finally the system state covariance is denoted as $\mathbf{\Sigma}_{\mathbf{x}}$.

7.5.2 Process Model

The process model, f, predicts the evolution of the system state with the control input, $\mathbf{u}$, and the process noise, $\mathbf{c}$. The process model is of the form

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{c}_t) \tag{7.47}$$

where $\mathbf{u}$ is the control input comprising of acceleration and rotational velocities from the IMU which writes as

$$\mathbf{u} = [\mathbf{a},\ \boldsymbol{\omega}] \tag{7.48}$$

and $\mathbf{c} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\mathbf{c})$ is the process noise consisting of noise in acceleration, angular velocity, acceleration bias, gyroscope bias and $\boldsymbol{\Sigma}_\mathbf{c}$ is the process noise covariance.

The process model f is defined as

$$\begin{bmatrix} \hat{\mathbf{g}} \\ \mathbf{b} \\ \mathbf{v} \\ \mathbf{r}_i \\ \hat{\mathbf{t}}_i \\ \rho_i \end{bmatrix}_{k+1} = \begin{bmatrix} exp\left(\tilde{\boldsymbol{\omega}}_\times\ \Delta t\right)\hat{\mathbf{g}} \\ \mathbf{b} \\ \mathbf{v} + \mathbf{P}^{\mathcal{L}}\mathbf{R}_{\mathcal{B}}\tilde{\mathbf{a}}\ \Delta t \\ exp\left(\tilde{\boldsymbol{\omega}}_\times\ \Delta t\right)\left(\mathbf{r}_i + \mathbf{v}\Delta t + \frac{1}{2}\tilde{\mathbf{a}}\Delta t^2\right) \\ exp\left(\tilde{\boldsymbol{\omega}}_\times\ \Delta t\right)\hat{\mathbf{t}}_i \\ \rho_i \end{bmatrix}_k \tag{7.49}$$

where $\tilde{\mathbf{a}} = \mathbf{a} + g\hat{\mathbf{g}} - \mathbf{b}_\mathbf{a}$, g is the gravitational acceleration, $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} - \mathbf{b}_{\boldsymbol{\omega}}$. $\omega_\times$ gives the skew-symmetric matrix of its vector argument as

$$\omega_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{7.50}$$

The permutation matrix $\mathbf{P}$ is

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{0}_{1\times2} \\ \mathbf{0}_{2\times1} & \mathbf{I}_{2\times2} \end{bmatrix} \tag{7.51}$$

7.5.3 Measurement Model

Our UKF executes two types of measurement updates which are for gravity vector and local map corrections. The input to the first model is the attitude estimated by the onboard autopilot. The latter model is fed with an array of *knot measurements*, $\left\{\tilde{K}\right\}$, estimated using the method explained in Sec. 7.4.

The orientation measurement model is a linear operation which writes

$$\mathbf{z}_t^{\mathcal{I}} = \mathbf{H}^{\mathcal{I}}\,\mathbf{x}_t + \mathbf{m}_t^{\mathcal{I}}. \tag{7.52}$$

where $\mathbf{m}^{\mathcal{I}}$ is the additive noise modeled as a zero-mean normal distribution. This measurement is obtained from the orientation estimate of the onboard controller as

$$\mathbf{z}_t^{\mathcal{I}} = -\mathbf{R}(\mathbf{q}_t^{\mathcal{I}})\,[0,\ 0,\ 1]^{\top} \tag{7.53}$$

where $\mathbf{R}$ gives the rotation matrix corresponding to the given quaternion and $\mathbf{q}^{\mathcal{I}}$ is the orientation estimate of the onboard autopilot. The IMU measurement model is defined as

$$\mathbf{H}^{\mathcal{I}} = [\mathbf{I}_{3\times3},\ \mathbf{0}_{3\times8},\ \mathbf{0}_{3\times7},\ ...,\ \mathbf{0}_{3\times7},\ ...]\,. \tag{7.54}$$

The knot measurement model for a single knot is linear in the state vector too, and writes as

$$\mathbf{z}_t^{K_i} = \mathbf{H}^{K_i}\,\mathbf{x}_t + \mathbf{m}_t^{K_i} \tag{7.55}$$

where $\mathbf{m}^{K}$ is the additive noise modeled as a zero-mean normal distribution. i is the index of the measured knot and its corresponding model is written as

$$\mathbf{H}^{K_i} = \left[\mathbf{0}_{3\times 11},\ ...,\ \mathbf{I}_{3\times 7},\ ...\right]. \tag{7.56}$$

There is an implicit mapping between the index i to the corresponding knot's position in the state vector.

Measurement error calculation cannot be performed through normal subtraction since the knot tangents and the gravity vector direction vectors are normalized. For this, we introduce *pure quaternions* to represent and rotate points in 3D, and *rotation vectors* to represent rotational error and uncertainty. The pure quaternion corresponding to a 3D point $\mathbf{p}$ is

$$\mathbf{q}_p\left(\mathbf{p}\right) = \left[0,\ \mathbf{p}\right]. \tag{7.57}$$

The rotation quaternion for a rotation vector $\boldsymbol{\omega}$ is

$$\mathbf{q}_r\left(\boldsymbol{\omega}\right) = \left[cos\left(\frac{||\boldsymbol{\omega}||_2}{2}\right),\ sin\left(\frac{||\boldsymbol{\omega}||_2}{2}\right)\frac{\boldsymbol{\omega}}{||\boldsymbol{\omega}||_2}\right]. \tag{7.58}$$

The rotation of a 3D point, represented with a pure quaternion as above, can be realized by evaluating the conjugation of this point with a rotation quaternion as

$$\mathbf{q}_p\left(\mathbf{p}\right)' \leftarrow \mathbf{q}_r\left(\boldsymbol{\omega}\right)\mathbf{q}_p\left(\mathbf{p}\right)\mathbf{q}_r\left(\boldsymbol{\omega}\right)^{-1} \tag{7.59}$$

using the Hamilton product. The difference between two vectors is the quaternion that transforms the subtrahend to the minuend vector. This can be written as

$$\hat{\mathbf{u}} - \hat{\mathbf{v}} = \mathbf{q}\ \mid\ \hat{\mathbf{u}} = \mathbf{q}\,\hat{\mathbf{v}}\,\mathbf{q}^{-1}. \tag{7.60}$$

As in our case, if both vectors are of unit norm, the resultant quaternion is of unit norm too. In this particular case, we can circumvent solving for the preceding tedious equation through a geometric reasoning and write that

$$\mathbf{q} = \mathbf{q}_r\left(\mathbf{w}\right) \tag{7.61}$$

$$\mathbf{w} = \hat{\mathbf{v}} \times \hat{\mathbf{u}} \tag{7.62}$$

where $\cdot \times \cdot$ is the cross product. The measurement error in knot position and radius can be calculated using the normal subtraction.

The knot measurement corresponding to a given knot, $K_i \in L$ is the closest point, in sense of metric given in Equ. 7.9, on the Bézier approximation to the knot measurements obtained from the point cloud, $\{\mathbf{o}\}$ (Equ. 7.8), through an optimization process explained in Sec. 7.4. This can be written as

$$\mathbf{z}^{K_i} = \left[\acute{\mathcal{B}}(t_*),\ \hat{\acute{B}}(t_*),\ \acute{\mathcal{B}}_\rho(t_*)\right] \tag{7.63}$$

$$t_* = \underset{t}{\operatorname{argmin}} < \acute{\mathcal{B}}(t), K >, t \in [0,1]. \tag{7.64}$$

If the coordinate of the closest point, t_*, is not inside the $[0,1]$ range, then K_i is marked for removal from the local map.

The measurement uncertainty is obtained through a similar interpolation approach. The position uncertainty of a point on the Bézier curve is the inverse of the weighted average of the Fisher information of each control point. The weights are simply the Bernstein polynomials (Equ. 7.7) evaluated as the corresponding point coordinate (*i.e.* t_*). As explained in Sec. 7.2.2 control points are generated from knot positions and knot axes. The information for the two types of control points are respectively obtained as

$$\boldsymbol{\mathcal{F}}_{\mathbf{o}_i} = \frac{1}{C}\mathbf{P}_{\mathbf{r}}\mathbf{H}_{*,i}\mathbf{P}_{\mathbf{r}}^\top \tag{7.65}$$

$$\boldsymbol{\mathcal{F}}_{\mathbf{o}_{(i-)}} = \frac{\ell_i}{C}\mathbf{P}_{\hat{\mathbf{t}}}\mathbf{H}_{*,i}\mathbf{P}_{\hat{\mathbf{t}}}^\top \tag{7.66}$$

$$\boldsymbol{\mathcal{F}}_{\mathbf{o}_{(i+)}} = \frac{\ell_{i+}}{C}\mathbf{P}_{\hat{\mathbf{t}}}\mathbf{H}_{*,i}\mathbf{P}_{\hat{\mathbf{t}}}^\top \tag{7.67}$$

where $\mathbf{H}_{*,i}$ is the corrected Hessian matrix of K_i, given in Equ. 7.45, of the cost function,

C (Equ. 7.26), evaluated both at the point of convergence. The variation in the knot-axes-borne informations is due to the different multiplicative factors in the construction of their corresponding control points. The permutation matrices are

$$\mathbf{P}_{\mathbf{r}} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times4} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times4} \end{bmatrix} \tag{7.68}$$

$$\mathbf{P}_{\hat{\mathbf{t}}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 \end{bmatrix} \tag{7.69}$$

The Fisher information of a point on the centerline, $\acute{\mathcal{B}}(t)$, is approximated as

$$\boldsymbol{\mathcal{F}}_{\acute{\mathcal{B}}(t)} = \sum_{v=0}^{m-1} \boldsymbol{\mathcal{F}}_{\mathbf{o}_v}\ B_{v,m}(t) \tag{7.70}$$

where $\boldsymbol{\mathcal{F}}_{\mathbf{o}_v}$ is the information matrix of the appropriate type.

The Fisher information of the tangent can be approximated in a similar way as

$$\boldsymbol{\mathcal{F}}_{\hat{\acute{\mathcal{B}}}(t)} = \left(\frac{d\eta\left(\acute{\mathcal{B}}'(\tau)\right)}{d\tau} \right) \left(\frac{d\boldsymbol{\mathcal{F}}_{\acute{\mathcal{B}}(\tau)}}{d\tau} \right) \left(\frac{d\eta\left(\acute{\mathcal{B}}'(\tau)\right)}{d\tau} \right)^{\top} \Bigg|_{\tau=t} \tag{7.71}$$

$$\acute{\mathcal{B}}'(t) = \frac{d\acute{\mathcal{B}}(\tau)}{d\tau}\Bigg|_{\tau=t} \tag{7.72}$$

where $\eta(\cdot)$ is the vector normalization function. Lastly, the information of radius can be written as

$$\boldsymbol{\mathcal{F}}_{\acute{\mathcal{B}}_\rho(t)} = \sum_{v=0}^{m-1} \boldsymbol{\mathcal{F}}_{\rho_{v^*}}\ B_{v,m}(t) \tag{7.73}$$

where $\boldsymbol{\mathcal{F}}_{\rho_v}$ is the control point radius information estimate. Radii of the control points at i, i^-, i^+ indices are the same as the source knot's radius estimate. With the abuse of notation, we assume that there is an implicit mapping from the index v in the preceding equation to the source knot's index which we denote as v^*. Finally,

$$\boldsymbol{\mathcal{F}}_{\rho_{i^*}} = \frac{1}{C}\mathbf{P}_\rho \mathbf{H}_{*,i}\mathbf{P}_\rho^\top \tag{7.74}$$

$$\mathbf{P}_\rho = \begin{bmatrix} \mathbf{0}_{6\times 6} & \mathbf{0}_{6\times 1} \\ \mathbf{0}_{1\times 6} & 1 \end{bmatrix} \tag{7.75}$$

7.5.4 A Constrained UKF on a Nonlinear Manifold

The two sources of nonlinearities are the unit vectors of the state vector and the constant chord lengths which prohibit the use of a vanilla Kalman filter. For this, we use a UKF since it neither requires process or measurement models to be linear nor linearized as in the case of an EKF. In the UKF framework, the state uncertainty is approximated using a carefully selected set of *sigma points.* These sigma points fully capture the state mean and covariance when modeled as a Gaussian random variable. Through application of process and measurement models on the sigma points, one can attain significantly better results compared to alternative KF variations. We leave the details of UKF to the seminal paper [12] and explain issues specific to our choice of state definition.

The state covariance, denoted as $\mathbf{\Sigma_x}$, is a $11 + 7n_s$ square, positive definite matrix where n_s is the number of segments. Sigma points are obtained as $\mathcal{X}_i = \mathbf{x} \pm \mathcal{W}_i$ where $\{\mathcal{W}_i\}$ are the columns of $\sqrt{\bar{\mathbf{\Sigma}}_\mathbf{x}}$ obtained through *Cholesky* decomposition. $\bar{\mathbf{\Sigma}}_\mathbf{x}$ is the extended covariance matrix which is obtained as

$$\bar{\mathbf{\Sigma}}_\mathbf{x} = \begin{bmatrix} \mathbf{\Sigma_x} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma_c} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \bar{\mathbf{\Sigma}}_\mathbf{z} \end{bmatrix} \tag{7.76}$$

where the diagonal block matrices are state, process and measurement covariances from left to right. $\bar{\mathbf{\Sigma}}_\mathbf{z}$ is a square matrix of size $7n_s$ and includes the measurement noise uncertainties for each knot in the state vector.

The blocks of $\bar{\mathbf{\Sigma}}_{\mathbf{x}}$ corresponding to $\mathbf{b}, \mathbf{v}, \mathbf{r}_i$ and ρ_i hold their corresponding uncertainties which are in the Euclidean space. However, the uncertainties of $\hat{\mathbf{g}}$ and $\hat{\mathbf{t}}_i$ are represented as 3D rotation vectors [28]. This choice of representation prevents rank deficiency of the covariance matrix. In sigma point generation, mean and covariance calculation steps, these elements have to be handled properly. The summation and difference operations for unit vectors is explained in the previous section (Equ. 7.60) The mean of a set of unit vectors, $\{\hat{\mathbf{a}}\}$, is [28]

$$\bar{\hat{\mathbf{a}}} = \underset{\hat{\boldsymbol{\alpha}}}{\operatorname{argmin}} \sum_{\{\hat{\mathbf{a}}\}} \hat{\mathbf{a}} \times \hat{\boldsymbol{\alpha}}. \tag{7.77}$$

The innovation of UKF does not respect the constant distance constraints, ℓ_i, between adjacent knots. We also want to keep the the position of the root segment, $\mathbf{r}_0$, fixed with respect to the robot frame along the tunnel axis direction. This can written as $\mathbf{r}_0 \cdot \hat{\mathbf{x}}^{\mathcal{M}} = 0$. Equality constraints can be realized in two different ways which are *pseudo-observation* and *projection* [58]. The former family of constraints generates a fictitious measurement from the equality constraint with its uncertainty always $\mathbf{0}$. The projection approach applies the constraint directly on the state estimate thus, unlike the former approach, the constraint is guaranteed to hold. For this reason we prefer latter approach. Furthermore, [58] explains in detail that, in the UKF context, a nonlinear constraint must be applied to both sigma points after propagation and the *a prior* estimate to guarantee that the constraints are satisfied.

The projection function for constant chord length is

$$\mathbf{p}_\ell\left(\mathbf{r}_i\right) = \mathbf{r}_i + \acute{\mathcal{B}}(t_* + \Delta t_*) - \acute{\mathcal{B}}(t_*) \tag{7.78}$$

where

$$t_* = \underset{t}{\operatorname{argmin}} < \acute{\mathcal{B}}(t), K_i > \tag{7.79}$$

$$\Delta t_* = \underset{\Delta t}{\operatorname{argmin}} \left|\left| \mathbf{r}_i - \mathbf{r}_{i^-} + \acute{\mathcal{B}}(t_* + \Delta t) - \acute{\mathcal{B}}(t_*) \right|\right|_2 - \ell_i. \tag{7.80}$$

Here, the projection function translates the knot position parallel to the centerline (at each t) such that its Euclidean distance from the preceding knot equals the constant chord distance. Starting from the knots adjacent to the root knot, each knot is slid this way one at a time. It should be noted that there are infinitely many translations that will bring a knot to the required distance. This particular approach constrains the direction towards which each knot must be translated respecting the shape of the centerline. Furthermore, the Euclidean distance of the slid knot from the centerline does not change, preventing artificially increasing the state uncertainty.

7.5.5 Robot State

Although we do not explicitly include the robot state in the UKF, it is indirectly estimated along with the local map. The position of the root segment, which is also the origin of local map frame, is constrained by the UKF as explained in the previous section. Hence $-\mathbf{r}^0$ is the robot position in the Velodyne frame. Furthermore, roll and pitch angles can be inferred from $\hat{\mathbf{g}}$ and yaw from the difference between $\hat{\mathbf{x}}^{\mathcal{B}}, \hat{\mathbf{x}}^{\mathcal{L}}$ pair. Finally, corresponding uncertainties of these variables can be extracted from the state covariance to obtain robot state uncertainty.

7.6 Experimental Results

In this section, we report on the experiments that we performed inside a penstock in Center Hill Dam, TN. We compare this approach with our work explained in the previous chapter to demonstrate how the new formulation and filtering mechanisms improve robustness around critical regions such at the top of the penstock. This penstock has a diameter of ~ 5 meters and is about 70 meters long. We could experiment only along certain parts of the penstock due to an on-going construction on the site. Because of this we could not collect data around the turbine area which is the lower part of a penstock. The data we present is from the horizontal and inclined sections of the tunnel.

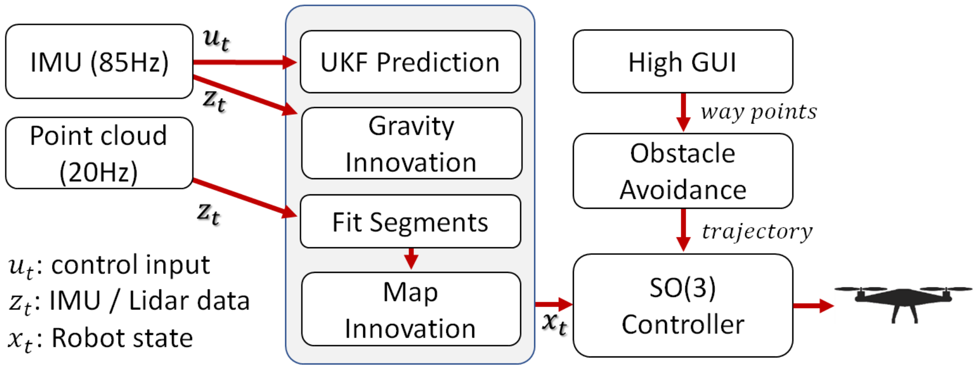


Figure 7.7: The system diagram of the estimator. In this chapter we explaine only the components directly related to the proposed improvements.

We used our custom-designed platform, which is explained in Sec. 3.2.4 equipped with a 3D lidar and an IMU. Fig. 7.7 shows the system diagram some components of which are explained in the previous sections. Our algorithms are implemented in ROS, optimized for real-time performance and uses less than a single core. Since our tests were conducted in confined space, we do not have ground truth data for comparison. A tunnel reconstruction that shows PSGC geometry also means that the state is estimated accurately since the robot pose is a obtained through inverting map parameters.

Our previous work follows a similar approach and reconstructs the tunnel by fitting segments. However, it does not impose any constraints between segments and also only uses the point curvature to weight the reliability of point cloud data. This often results in segments either far from their neighboring segment hence does not preserve continuity. Also when the robot flies in non-cylindrical regions, segments may end up at angles off by 90 degrees compared to neighbor segments. Two cases where the previous approach fails are shown in Fig. 7.8. In these figures, the robot is close to the gate where the tunnel cross-section is not cylindrical.

We compare also our results with the filtering enabled and disabled. Fig. 7.9 shows two cases where the current method solely using the optimizer with filtering and outlier elimination are both disabled. In this case the results are almost the same as our previous work. However as in Fig. 7.10, after enabling the filtering, the current method performs successfully both when the PSGC assumption is violated or holds.

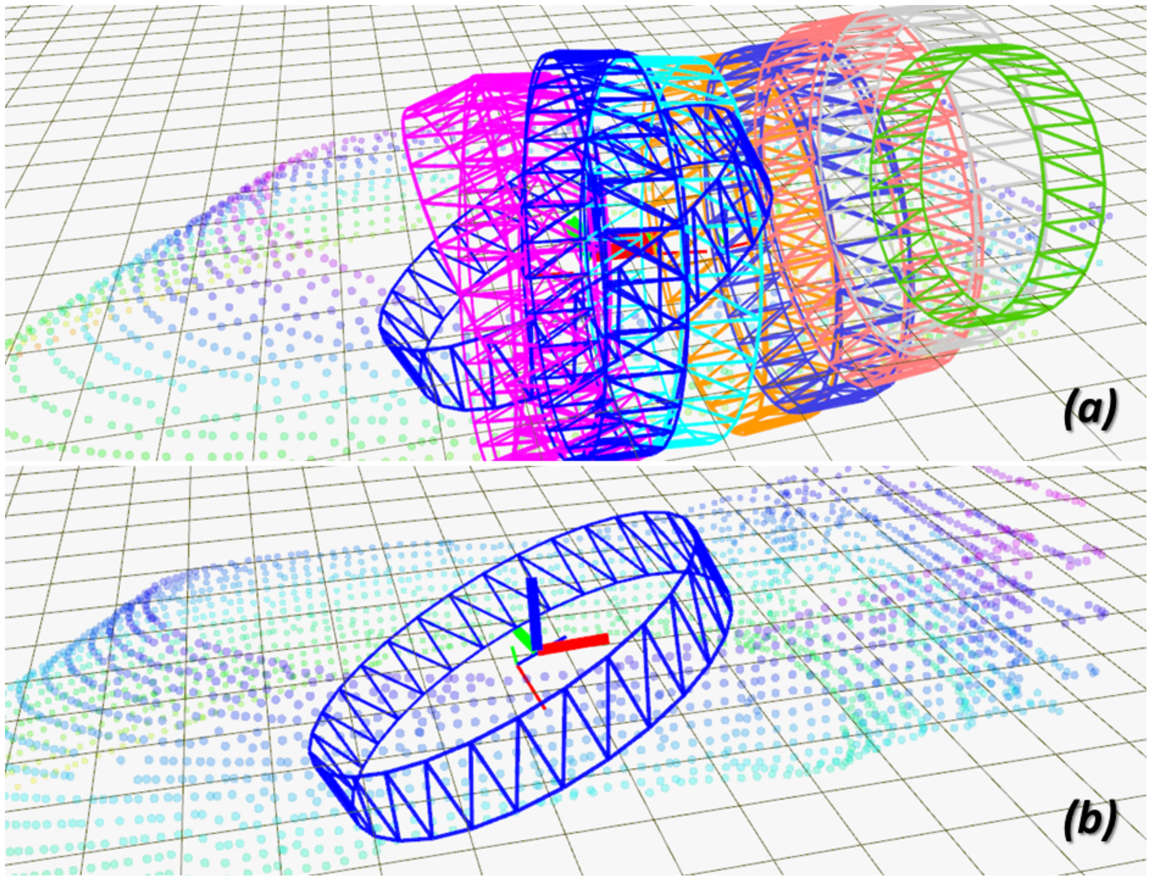


Figure 7.8: These RViz snapshots show failure of the approach in 6 failing while hovering close to the gate where the circular cross-section assumption does not hold. *(a)* Orientation of some segments are estimated wrong and the spacing between segments are not uniform. *(b)* Since the first segment is estimated wrong, the fitting process is early terminated.

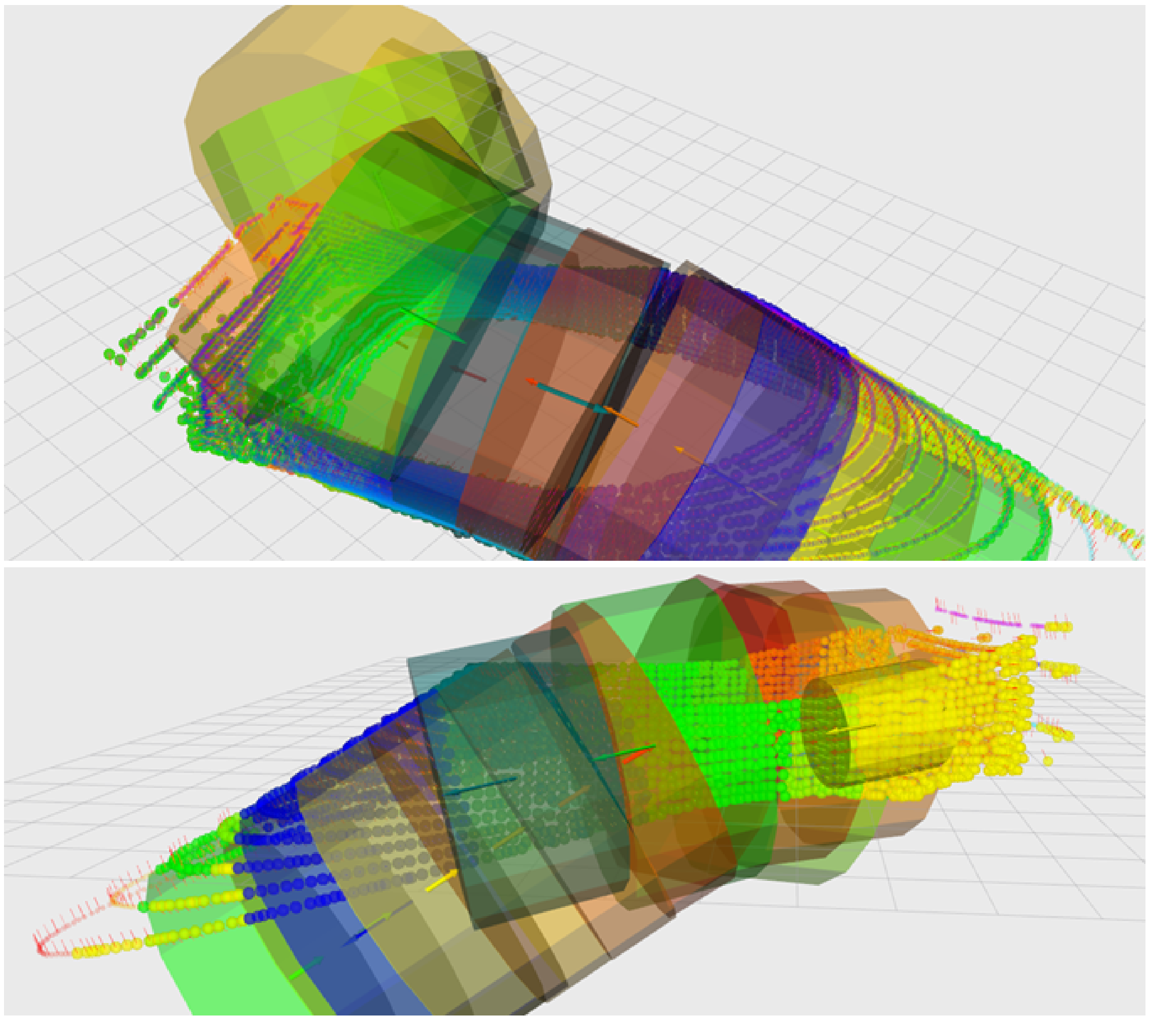


Figure 7.9: These RViz snapshots show failure of the current approach while the robot is hovering at the same region as in Fig. 7.8. Here, we excluded Watson distributions from the optimizer and disabled outlier elimination to demonstrate their effects.

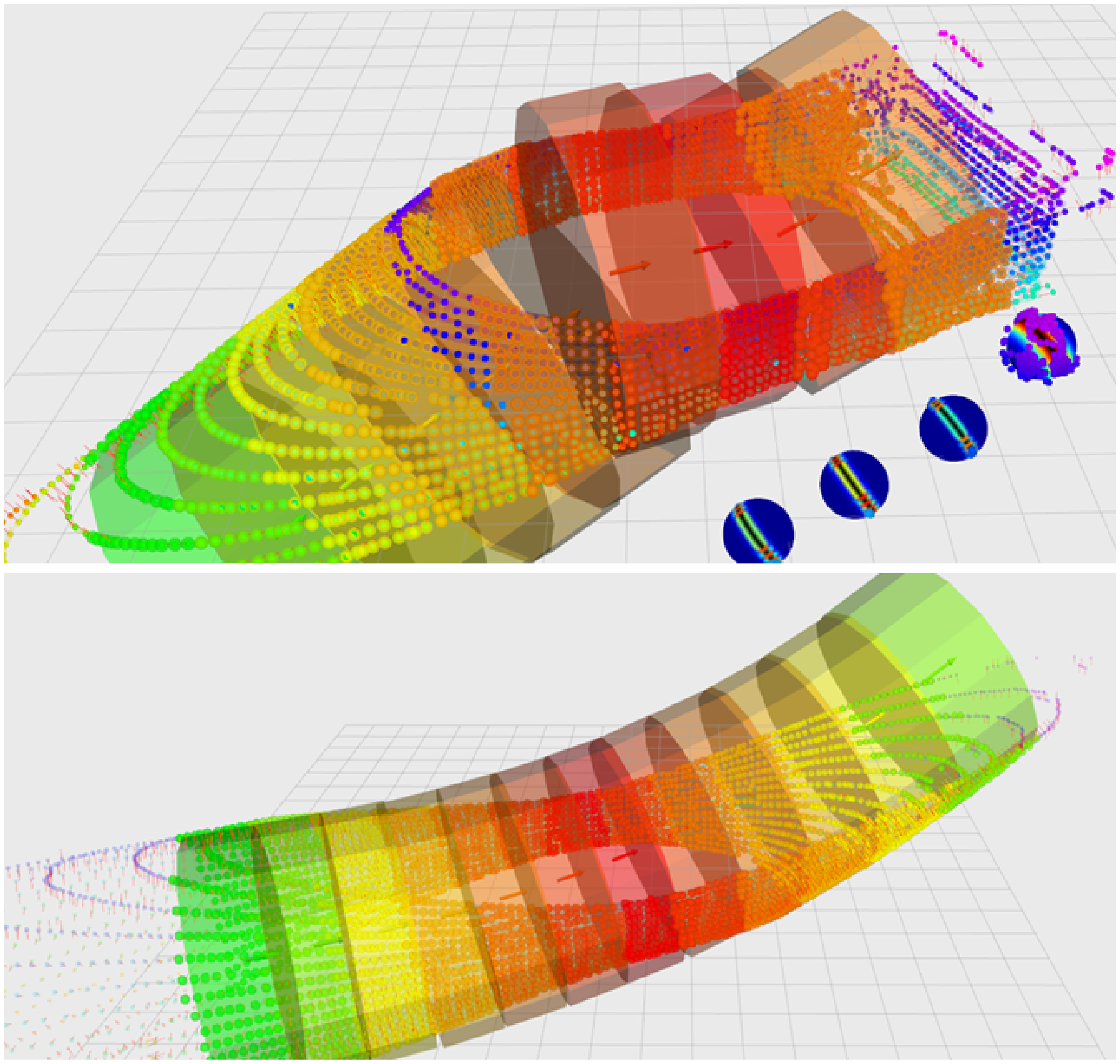


Figure 7.10: Two cases showing that the current method performs successfully when the robot is hovering close to the gate where the previous method and this method with the filtering disabled fails. The second snapshot also show the map along the inclined section of the tunnel.

Fig. 7.11-7.12 show the estimated tunnel radii as interpolated using a Bernstein polynomial with its knots being the individual segment radii. The first figure plots only a subset of centerline radii for clarity (at 0.7 seconds intervals) while the second figure shows the complete radii trajectory of the centerline curve with shading. The three axes of these figures are the time, signed centerline line integral and radius. The second term is obtained as

$$\acute{\mathcal{B}}_L(0 \to 1) - \acute{\mathcal{B}}_L(0 \to t^*) \tag{7.81}$$

where

$$\acute{\mathcal{B}}_L(a \to b) = \int_a^b \left\| \left. \frac{d\acute{\mathcal{B}}_L(t)}{dt} \right|_{t=\tau} \right\| d\tau. \tag{7.82}$$

and t^* is the index of the point along the centerline curve closest to body frame origin, i.e. $\acute{\mathcal{B}}_L(\mathcal{O}_\mathcal{B})$ (Equ. 7.9). This offset centers the plots around the body frame center. The plots show the radius estimates for ± 10 meters.

The radius estimate is very smooth from the beginning of the flight until 70^{th} second and after 120^{th} second until the end of the flight. In between, the robot is very close to the gate where the piecewise-smooth-generalized-cylinder (PSGC) assumption fails resulting in very noisy segment estimates. However, it should be noted that the noisy estimates are constrained to less than a third of the whole centerline closer to the gate (around +10 meters). The radii of the remaining sections of the local map are almost not affected at all. This is basically due to the smoothing effect of Bézier curve. This is very important, because the robot is located at *Curve Length = 0* in these plots where the centerline (which also means the local map) estimate is not adversely affected neither the robot state. The period where the radius estimate is noisy *approximately* corresponds to the period in between the 75^{th} and 105^{th} seconds in the video attachment.

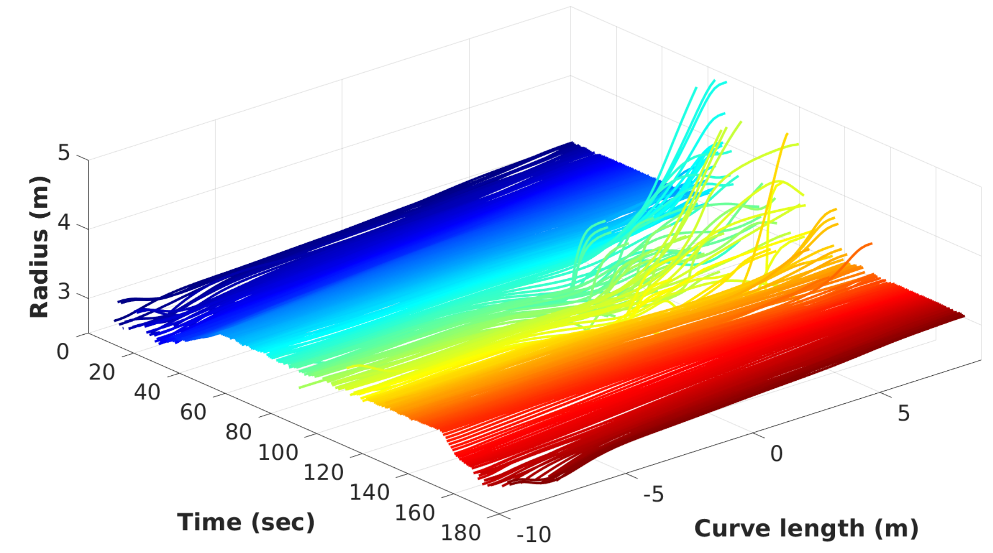
The estimated radii is close to 3 meters which is the actual diameter of the tunnel as we learned from the facility engineers. At the beginning and end of the flight, some regions are shown to have a smaller radius. These sections correspond to the region around the scroll case which has a converging conical shape. Since we do not have the engineering drawings of the whole structure, we cannot make a ground truth comparison.

In Fig. 7.13 we show the uncertainty of the results presented in the previous figures. Rather than variance, we plot the standard deviation along the z-axes since the variances are very small making visualization harder. The regions of high uncertainty are shown with peaks corresponding to the same regions where the radius estimate diverges from the ground truth value. These results are encouraging since by simply checking the radius estimate uncertainty, the robot can decide whether the tunnel shape agrees with the PSGC assumption.

In Fig. 7.14, we show the inclination estimate over time and along the local map axis. There are two dominant regions which correspond to the horizontal and inclined sections of the tunnel. These are marked with blue and green colors respectively while some regions are omitted since the inclination estimate is very noisy. The latter regions correspond to the sections of the penstock where the PSGC assumption does not hold. The ground truth values for these two regions are 0 and 30 degrees. Our estimates are in close match with these values for most of the flight. It should be noted that the inclination and radius estimates both fail over the same regions as can be compared with the other plots.

The slightly slant transition lines at 50^{th} and 130^{th} seconds are due to the gradual inclination change of the penstock. From these lines, it is possible to infer whether the robot is completely in the horizontal or inclined section of a tunnel. The blue region between 70^{th} and 110^{th} seconds at $CenterLine = 0$ corresponds to when the robot is hovering close to the gate at the upstream terminus of the penstock. This section of the tunnel is approximately a rectangular prism with its inclination close to zero . Although our inclination estimate is oscillatory, it performs acceptably well.

Finally, Fig. 7.15 shows the positional uncertainty of the centerline. This figure is generated by interpolating the information at a set of points along the centerline using the weights of the underlying Bernstein polynomial. The uncertainty ellipses represent the point covariances estimated as the inverse of weighted informations. It can be seen that, close to the region



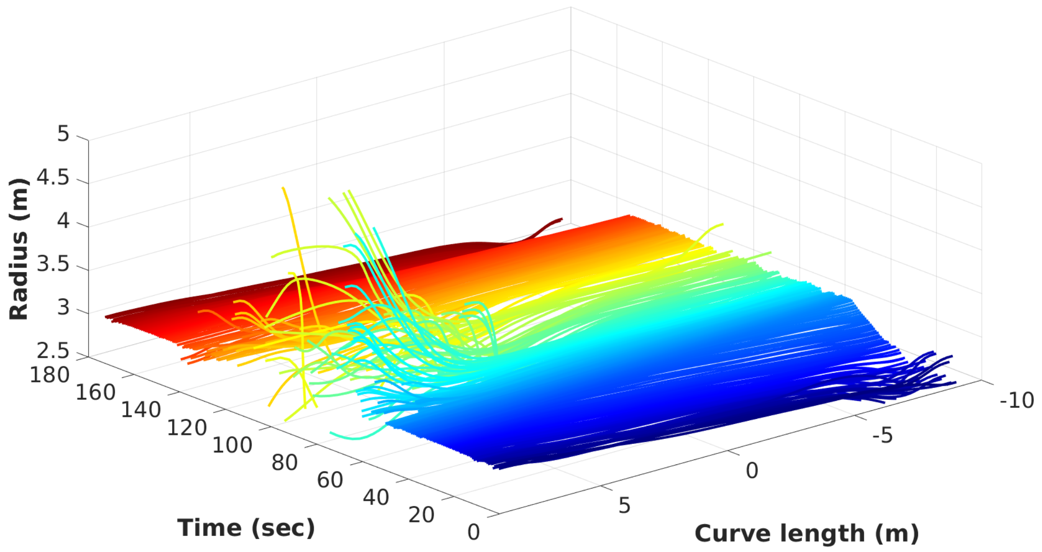
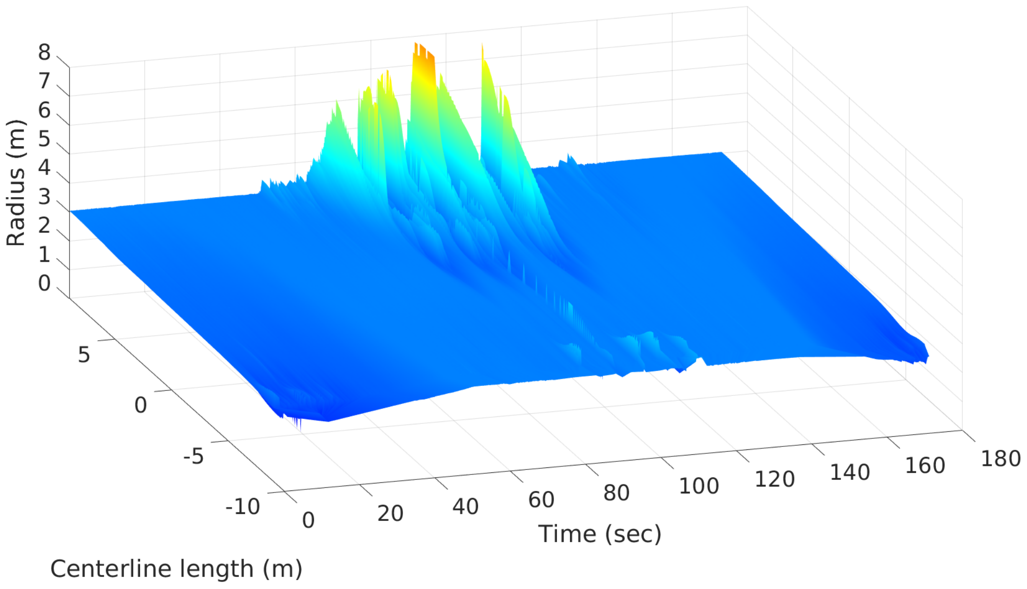


Figure 7.11: These plots show the radius estimate along the centerline estimate of the tunnel as a function of time and line integral of centerline estimate. The time spacing between each curve in the plot is 0.7 seconds deliberately kept large for clarity.



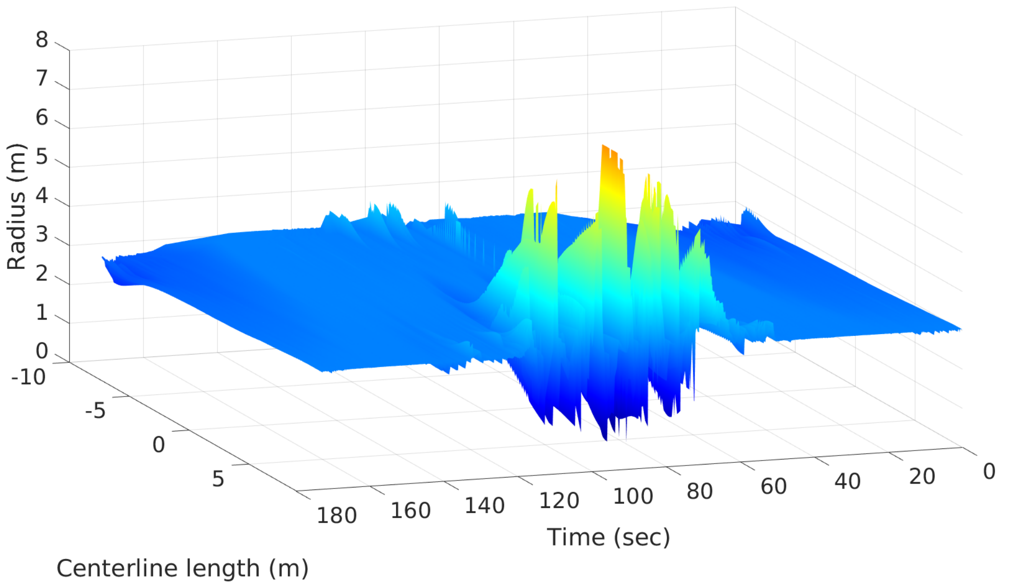


Figure 7.12: These plots show the radius estimate along the centerline estimate of the tunnel as a function of time and line integral of centerline estimate color coded as a function of estimated tunnel radius. This plots shows the *complete* radius estimate history.

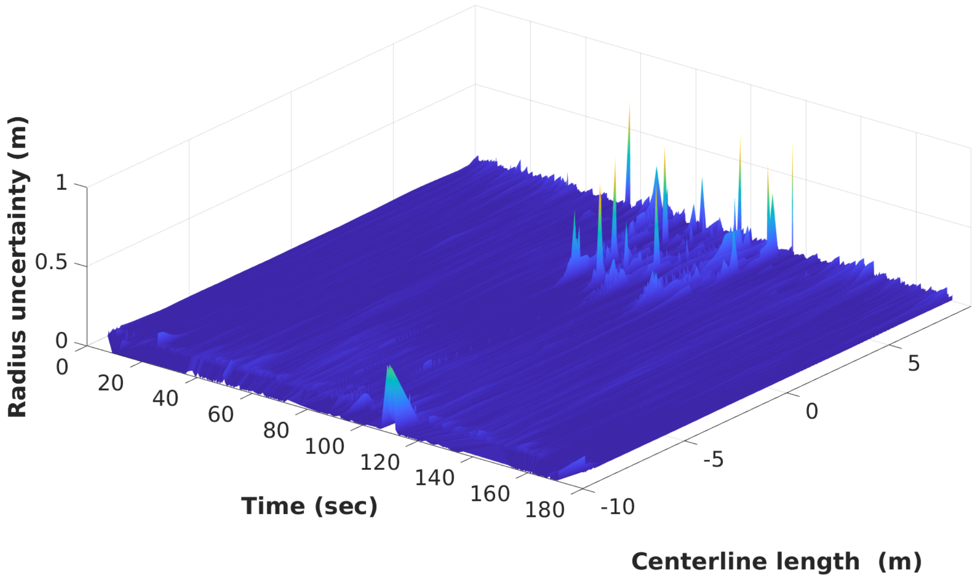


Figure 7.13: These plots show the uncertainty in the radius estimates along the centerline of the tunnel as a function of time and line integral of centerline estimate color coded as a function of estimator uncertainty. The regions where the uncertainty peaks match with the regions where the radius estimate diverges from the ground truth value as shown in the other figures.

with clutter (standing experiment crew), the uncertainty is higher. Also, the uncertainty along the lateral direction is much less compared to the vertical direction since the onboard lidar takes more measurements from the sides.

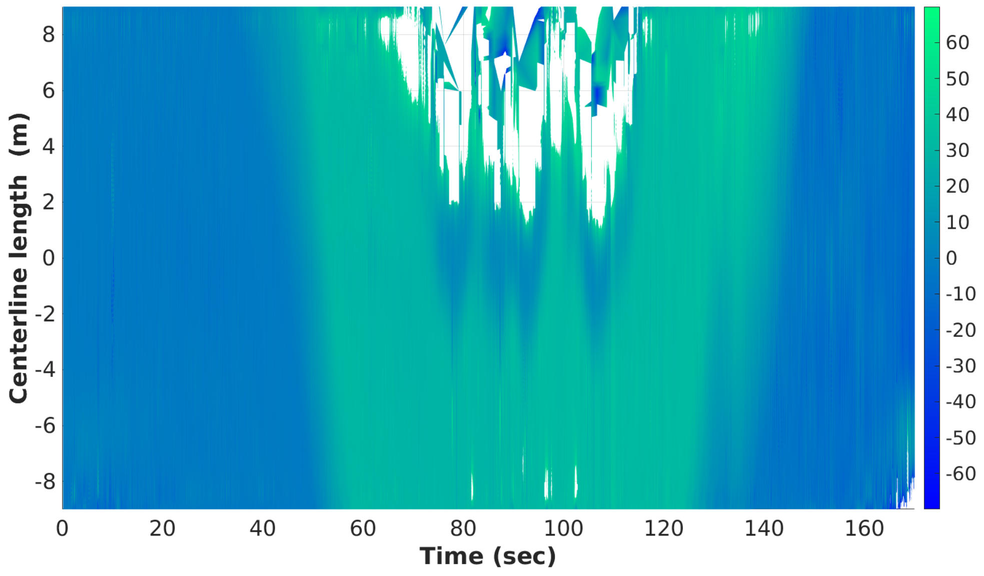


Figure 7.14: These plots show the inclination of the local map tunnel axis over the course of a flight. The vertical axis is the s coordinate along the centerline. Color shows the angle between the centerline tanget at a given s coordinate and the ground plane (*i.e.* plane normal to gravity vector). Light blue that covers almost haft the plot corresponds to ~ 0 degrees and green corresponds to ~ 30 degrees. The blue and green plateaus correspond to the horizontal and inclined sections of the tunnel with ground truth inclination angles of 0 and 30 degrees. Empty regions are where the inclination estimate is inaccurate and, hence, are not plotted.

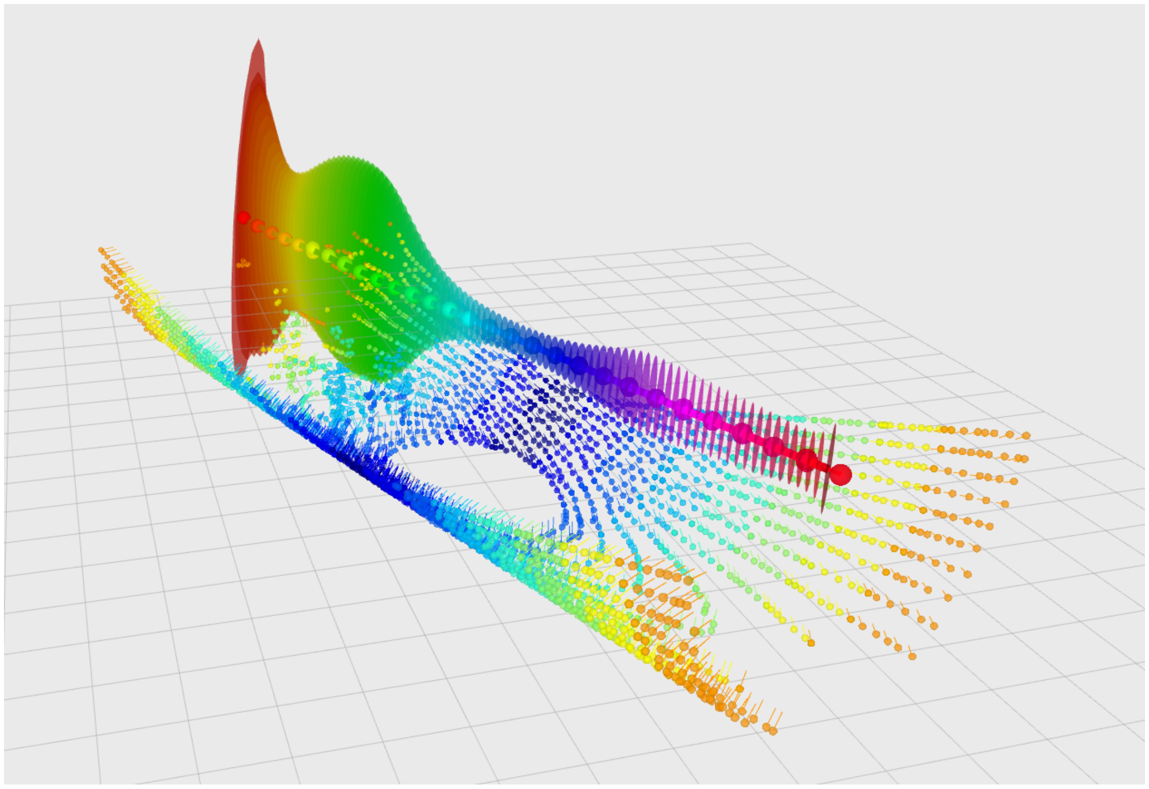


Figure 7.15: Positional uncertainties at a sparse set of points on the centerline.

Chapter 8

A Benchmark Comparison of Estimators

Flying robots employed in real-life scenarios require accuracy in order to achieve stable and robust flight. This requirement gains further importance when human operators are involved as in the case of infrastructure inspection. However, challenging environmental conditions such as mist, dust kicked up due to propeller downwash, reflective wet tunnel surfaces, and lack of sufficient illumination limits the choice of sensors for indoor inspection scenarios. Lidars and IMUs are the two sensors that suffer the least from these adverse conditions so range-based odometry algorithms are popular choices for state estimation in similar scenarios. Furthermore, equipped with these sensors, a MAV can build detailed maps of the environment while localizing itself without the requirement of external localization from motion capture or GPS.

Due to the highly unstable dynamics of multi-robot aerial vehicles, a failure in the state estimation can abruptly destabilize the robot causing it to crash. Hence, the performance of state estimation can be assessed *qualitatively* through successful autonomous flight of an MAV. On the other hand, it is not clear which range-based estimator proposed in the previous chapters perform better. Furthermore, solely comparing two range-based estimators might not necessarily reveal inaccuracies common to range-based approaches due to certain fundamental limitations of this class of estimators. For this reason, we also provide state estimates using a vision-only approach.

This chapter presents range-visual-inertial datasets collected onboard an MAV flying inside a penstock at Center Hill Dam, TN and performance comparisons of the two range-based estimators presented in Chap. 6 and Chap. 7, and a vision-based estimator details of which are explained in the subsequent sections. The datasets contain 3D point clouds, synchronized images from four color cameras and IMU measurements. The only source of illumination is the onboard LEDs that are placed around each camera. In order to provide artificial landmarks to the vision-based state estimation algorithm, we placed about a hundred AprilTag [81] fiducial markers along the tunnel. These tags are easily recognizable features that also guarantee loop closure once a tag is successfully recognized. The datasets include state estimates obtained using the TagSLAM library [135] as a comparate for benchmarking the proposed range-based estimators. TagSLAM builds a sparse map of AprilTags and provides 6 DoF state estimates using cameras only.

8.1 Dataset Collection

We collected 5 datasets by flying the DJI platform semi-autonomously along one of the penstocks at Center Hill Dam, TN. The horizontal and inclined sections of this penstock are both longer than 40 meters and 3 meters in diameter. The inclination is 30 degrees. Along the horizontal section, we placed about a hundred AprilTags for ground-truth. Due to the steep inclination it was unsafe to climb along the inclination and place tags. Because of this, we could place tags only along the horizontal section.

There are various AprilTag families which differ by the number of *pixels* used to encode tag ids. Among 4×4, 5×5 and 6×6 tag families (Fig. 8.1), the latter with 1 pixel border width is recommended by the author of [81] since the tag detector software performs significantly better with this tag family from different viewing angles and farther distances.

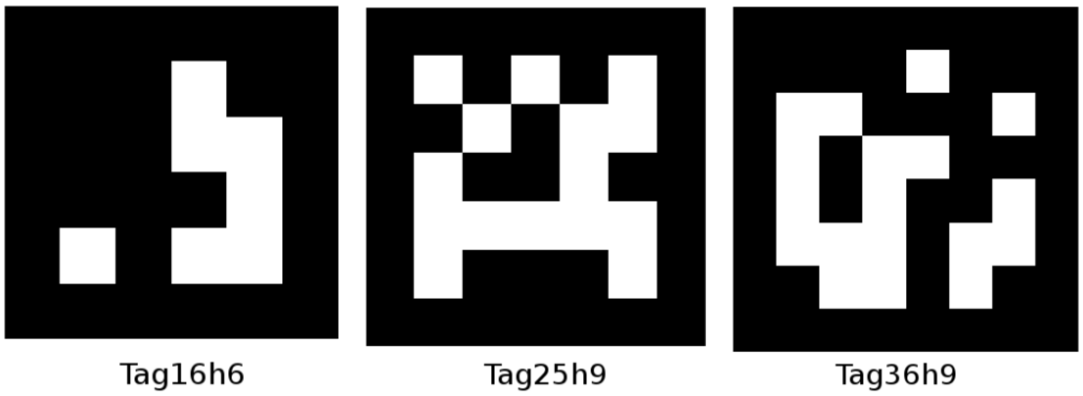


Figure 8.1: Sample AprilTag fiducial markers of sizes 4×4, 5×5 and 6×6 with 1 pixels border thickness.

We printed $16.5 \times 16.5\ cm^2$ AprilTags on water resistant, matte, transparent, sticky papers. The material choice for the printed tags is of great importance due to the wet experiment environment. Also, a glossy surface finish would cause specular reflection of light from the onboard power LEDs and cause the AprilTag detector to fail while a camera sees a tag at a high incident angle. These tags are sticked onto white $0.25''$ thick ABS sheets in order to ensure that the printed tags do not bend. In order to facilitate placing the tags, we screwed permanent neodymium magnets on all four corners of the ABS sheets. This way, tags snap on the metallic tunnel walls and their positions can be easily adjusted.

Before collecting the datasets, we calibrate the four onboard cameras and the IMU using the multi-camera calibration toolbox [1] of KumarRobotics and TagSLAM. The calibration is performed for both intrinsics and extrinsic parameters. Intrinsic calibration of each camera is performed using the multi-camera calibration toolbox, and the relative poses of each camera is estimated using TagSLAM.

Since we use wide angle lenses ($\sim 135^o$), we prefer the equidistant distortion model [47] which models fish-eye lenses more accurately than the radial-tangential distortion model. In particular, the OpenCV implementation of the latter model almost never converges for the lenses we use. The viewing cones of adjacent cameras intersect with sufficient overlap required for calibration at a distance longer than a few meters since adjacent cameras are

[1]`https://github.com/KumarRobotics/multicam_calibration`

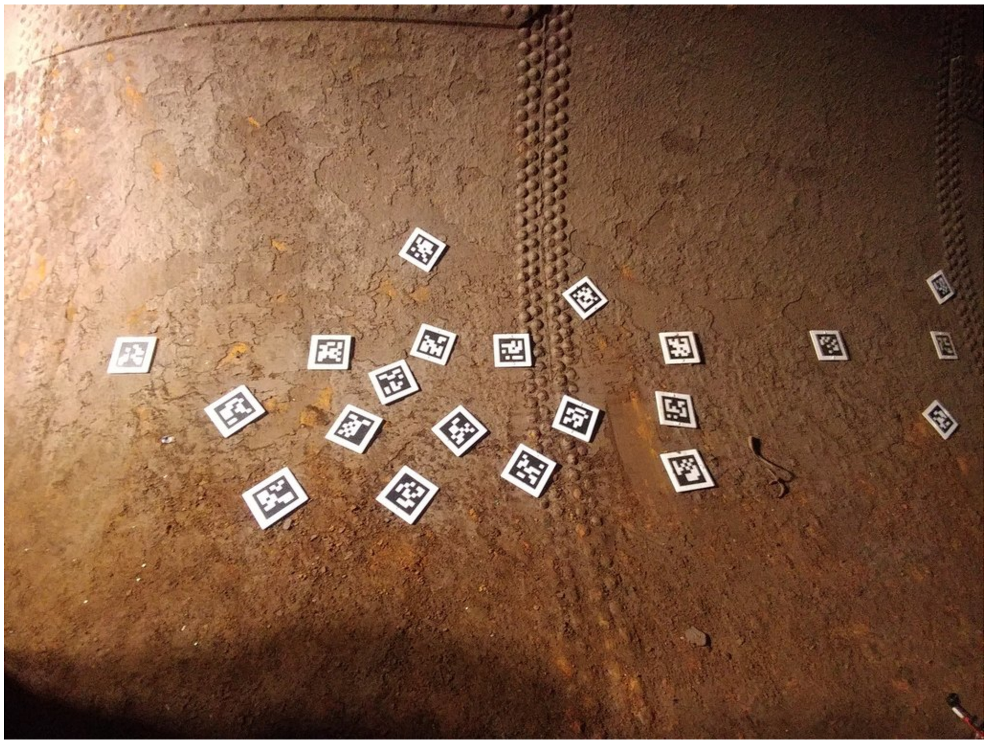


Figure 8.2: We use more than a dozen AprilTags randomly sprinkled on one side of the tunnel as the camera calibration pattern.

approximately oriented relatively at a right angle. Hence, in order to perform reliable calibration, a calibration plate larger than that can be passed through the small access hatch of the penstock would be required. As a solution to this, we use more than a dozen AptilTags randomly sprinkled on one side of the tunnel in such that adjacent cameras always see multiple tags at a certain distance. A sample photo of the *calibration pattern* is shown in Fig. 8.2.

On two sides of the tunnel, we placed two groups of tags that are visible from the onboard cameras while flying. The first group of tags are placed carefully along the curve that we marked using a laser level. A sample photo of these tags is shown in Fig. 8.3. The laser level projects a beam across the tunnel to establish a common level which we use to align this group of tags as can be seen in Fig. 8.4. Furthermore, adjacent tags are placed at a fixed distance with the help of a measure stick. These constraints are implemented as pose-graph factors in TagSLAM to improve the accuracy of the final map. Through setting the plane normal of the former constraint anti-parallel to the gravity vector, we

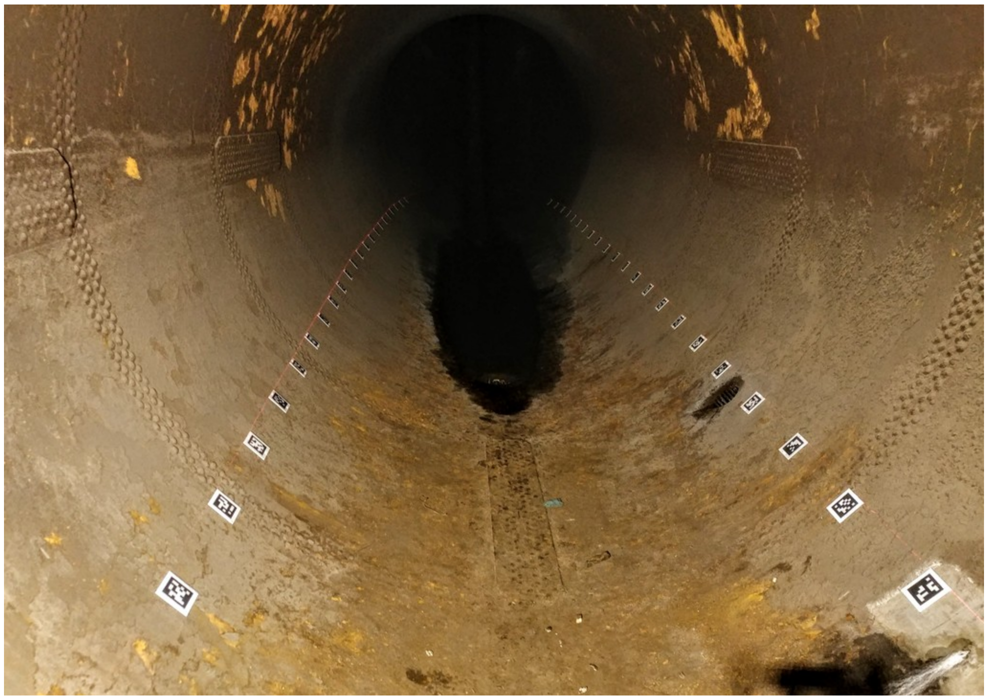
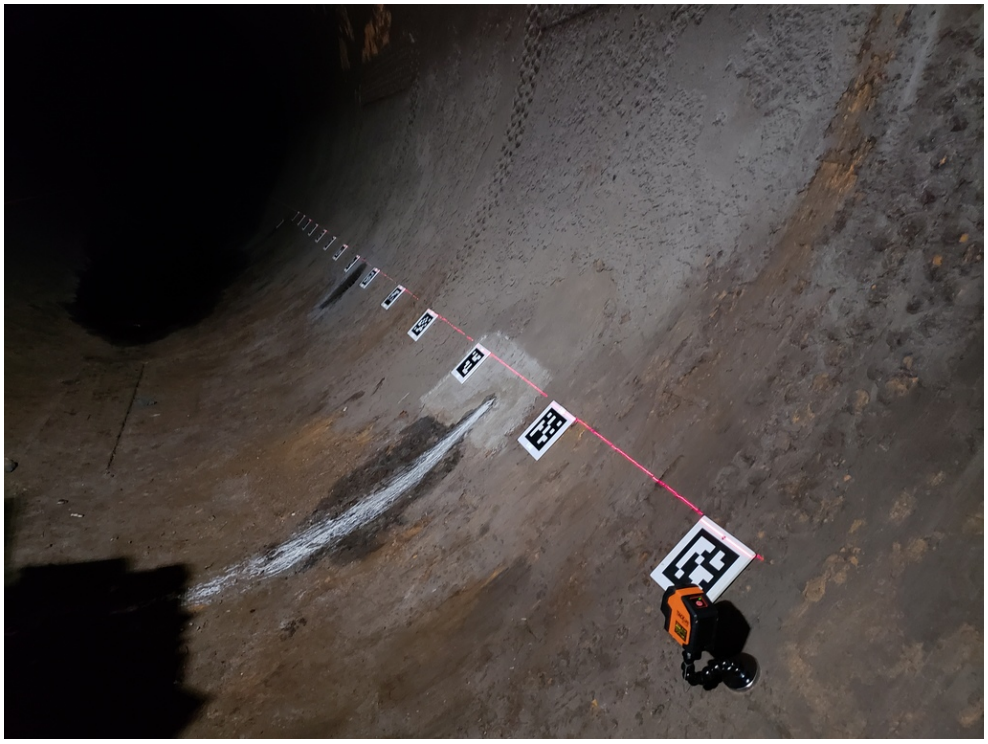


Figure 8.3: First group of tags are placed along a curve marked with the help of self-leveling laser level. Also, adjacent tags are placed as a fixed distance. These constraints are implemented as pose-graph factors in TagSLAM.

also estimate the gravity vector direction in the map frame. As will be explained in the subsequent sections, this greatly facilitates aligning coordinate frames of vision and range based estimators. The performance of TagSLAM with only tags aligned along this curve is not sufficient for estimator benchmarking purpose. For this, we randomly placed a second group of tags around the former group of tags. A photo of the tunnel with all the tags placed is shown in Fig. 8.5.

8.2 Map Reconstruction and Localization using AprilTags

TagSLAM requires tags and cameras to be attached to rigid bodies. Depending on the application, the relative poses of tags and cameras with respect to the bodies that they are attached to can be *static* or *dynamic*. Static bodies are not included in the optimization process other than providing constraints to it. A good example to this is the onboard cameras that are externally calibrated and attached to the abstract robot rigid body. Bodies can also



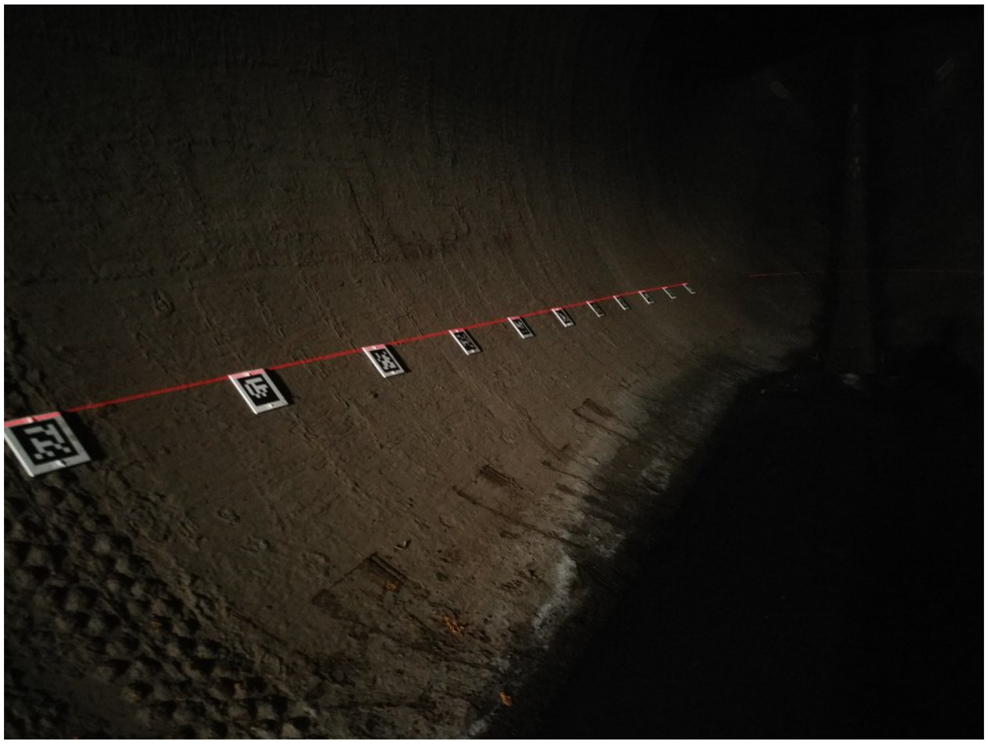


Figure 8.4: *(Top)* The self-leveling laser level is attached at the tunnel surface from its magnetic mount. *(Bottom)* The top edges of the first group of tags are aligned with the laser beam.

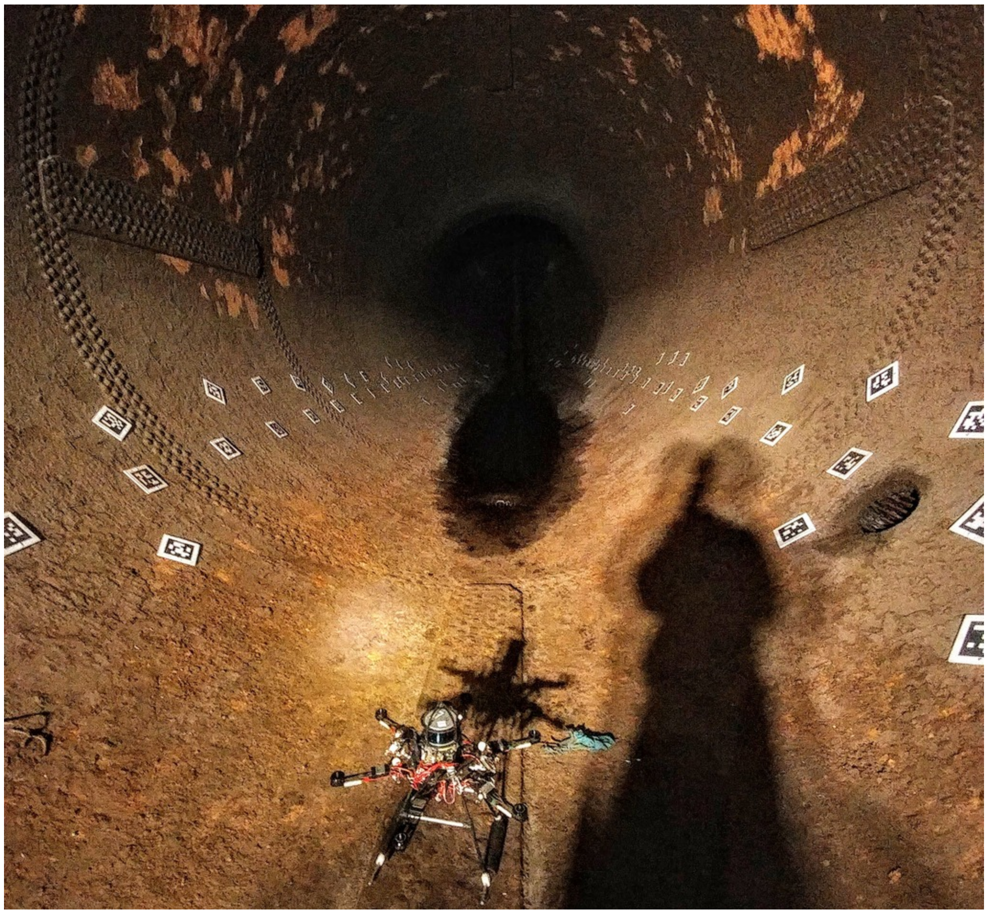


Figure 8.5: A photo of the penstock with both the aligned and randomly placed tags.

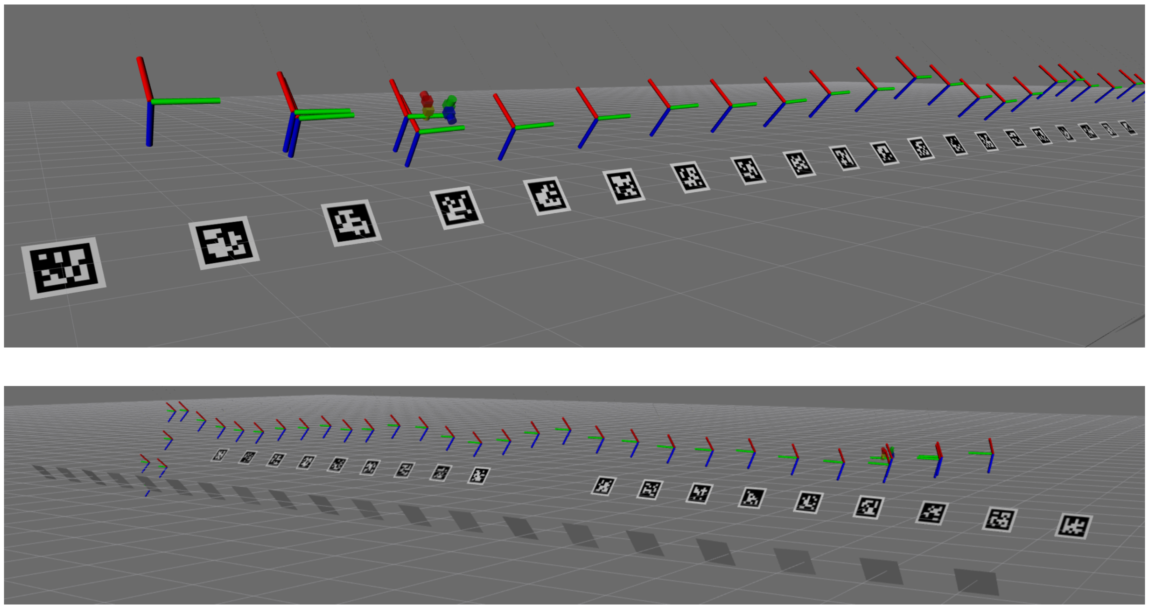


Figure 8.6: Snapshots from the RViz visualization tool that show the mapping results. This figure includes only the tags that are planed along a plane with the help of laser level.

be static or dynamic. Pose of a dynamic object is estimated on every frame with the most recent pose being used as the initialization point by the optimizer. For example, the robot body together with all the cameras attached to it is a single dynamic object. TagSLAM estimates a pose series for such objects. On the other hand, poses of static objects such as map landmarks are refined every time they are observed.

A nice feature of TagSLAM is that static objects that are defined previously can be imported. These static objects might be manually generated or obtained as a result of another run. We exploit this nice feature of TagSLAM by first running it on the datasets to generate a map of tags. In the second run, the optimized tag poses are fed to TagSLAM. Since tags are already localized with low uncertainties in the previous run, in the second run, TagSLAM only estimates the poses of dynamic objects with higher accuracy. We call these two different types of runs as *mapping* and *localization* modes. Fig. 8.6 and Fig. 8.7 are snapshots from RViz visualization tool showing the mapping results. The former figure shows only the tags placed along the plane marked with a laser level, and the latter figure shows all the tags.

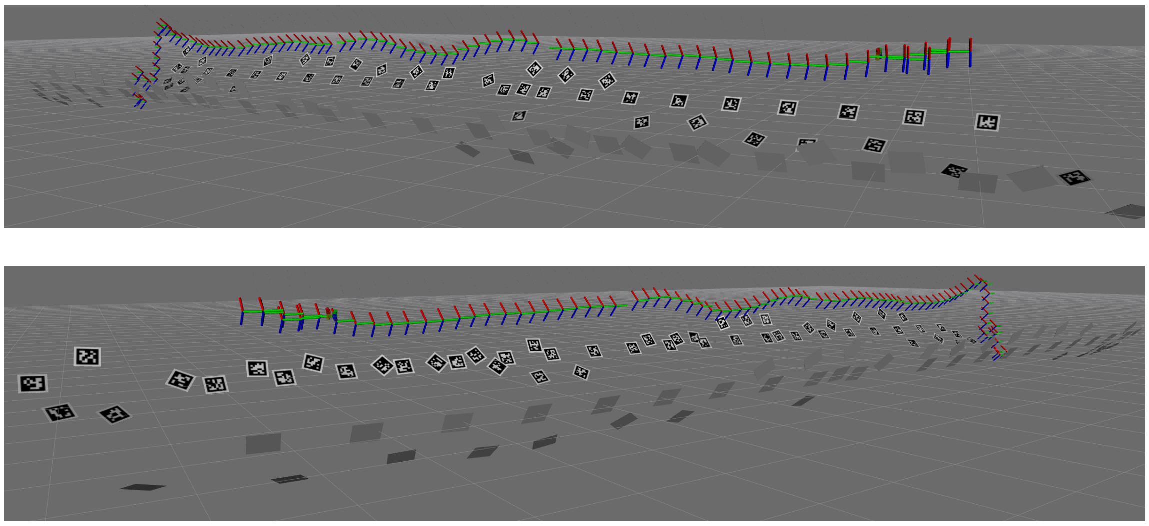


Figure 8.7: Snapshots from the RViz visualization tool that show the mapping results from different views.

8.2.1 Map Quality

In order to assess the quality of the map reconstructed by TagSLAM we use various metrics. These are the pixel projections errors at each frame, the pairwise distance between adjacent tags and the distance of each tags to the plane marked using a laser level. The former error is simply the distance in pixels between the corner of a detected tag and projection of its estimated position onto image plane. In Fig. 8.8 we present pixel projection errors for different maps. The average error is ~ 0.25 pixels which we believe is a demonstrative evidence of the map quality.

Tags placed along a plane are also placed at a specific distance from their adjacent tags with the help of a measure stick. In Fig. 8.9 we show tag-to-tag distance for different maps to be a little over 1 m. Although we intended to place tags with 1 meter of separation, we later noticed that the measure stick was longer by a few centimeters. However, as can be seen from these plots, TagSLAM, consistently, estimates very close values for each map which is a strong evidence for accuracy of the map. Secondly, we present in Fig. 8.10 the distance of each tag from the most-likely plane fit to them. The mean and maximum tag-to-plane distances are less than 1.5 cm and 6 cm respectively. It should be noted that a slight error

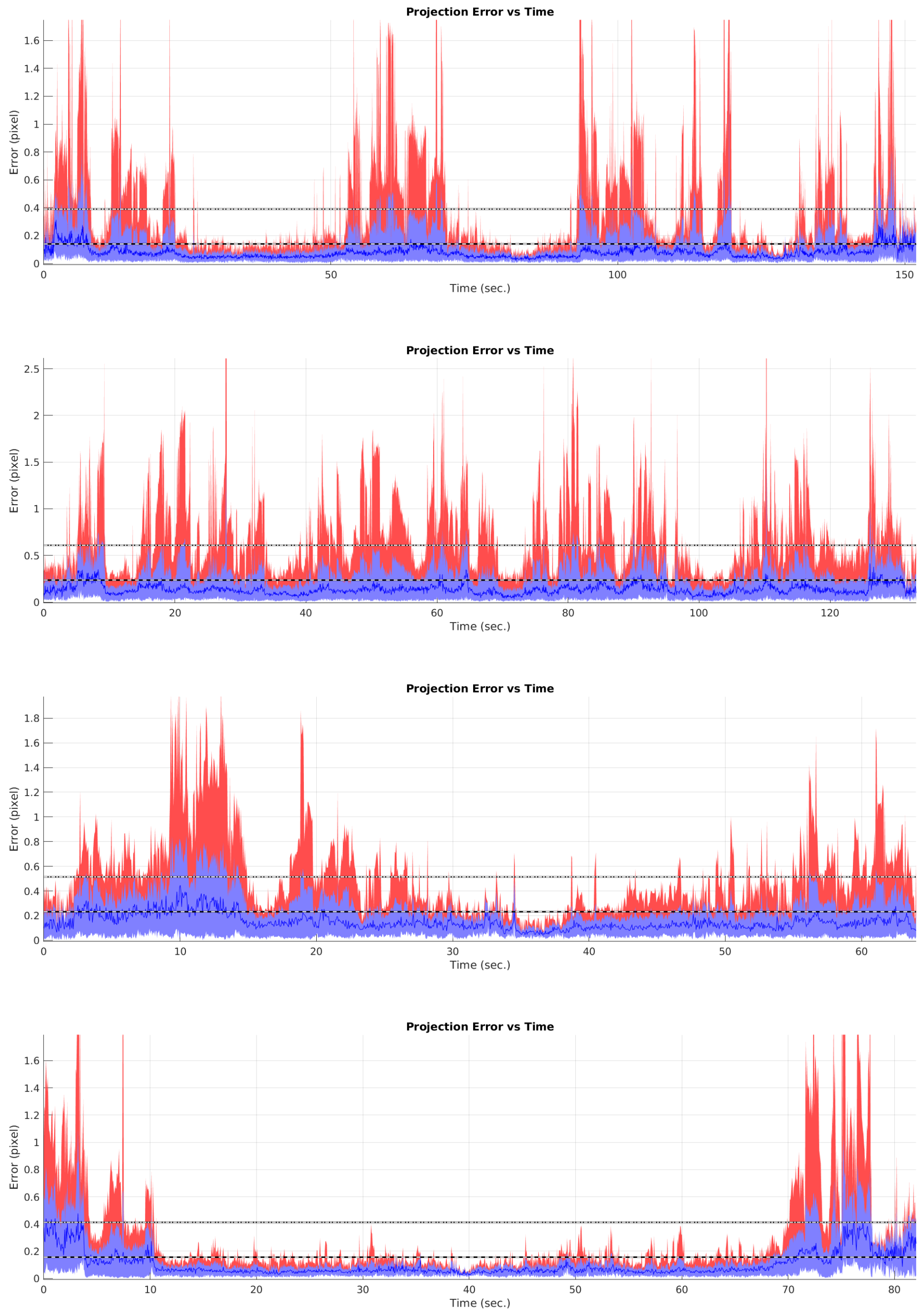


Figure 8.8: These plots show the pixel projection errors for maps reconstructed using four different datasets. The dark green curves are the median values. The red shades show the minimum and maximum projection errors, and the green shades show the standard deviation of the error. The two dashed lines show the overall mean error and the standard deviation of the total error.

in the plane normal would cause a great displacement at distal points since the length of the plane is more than 20 meters (*e.g.* 1^o error in orientation causes 17.5 cm distance at 10 m).

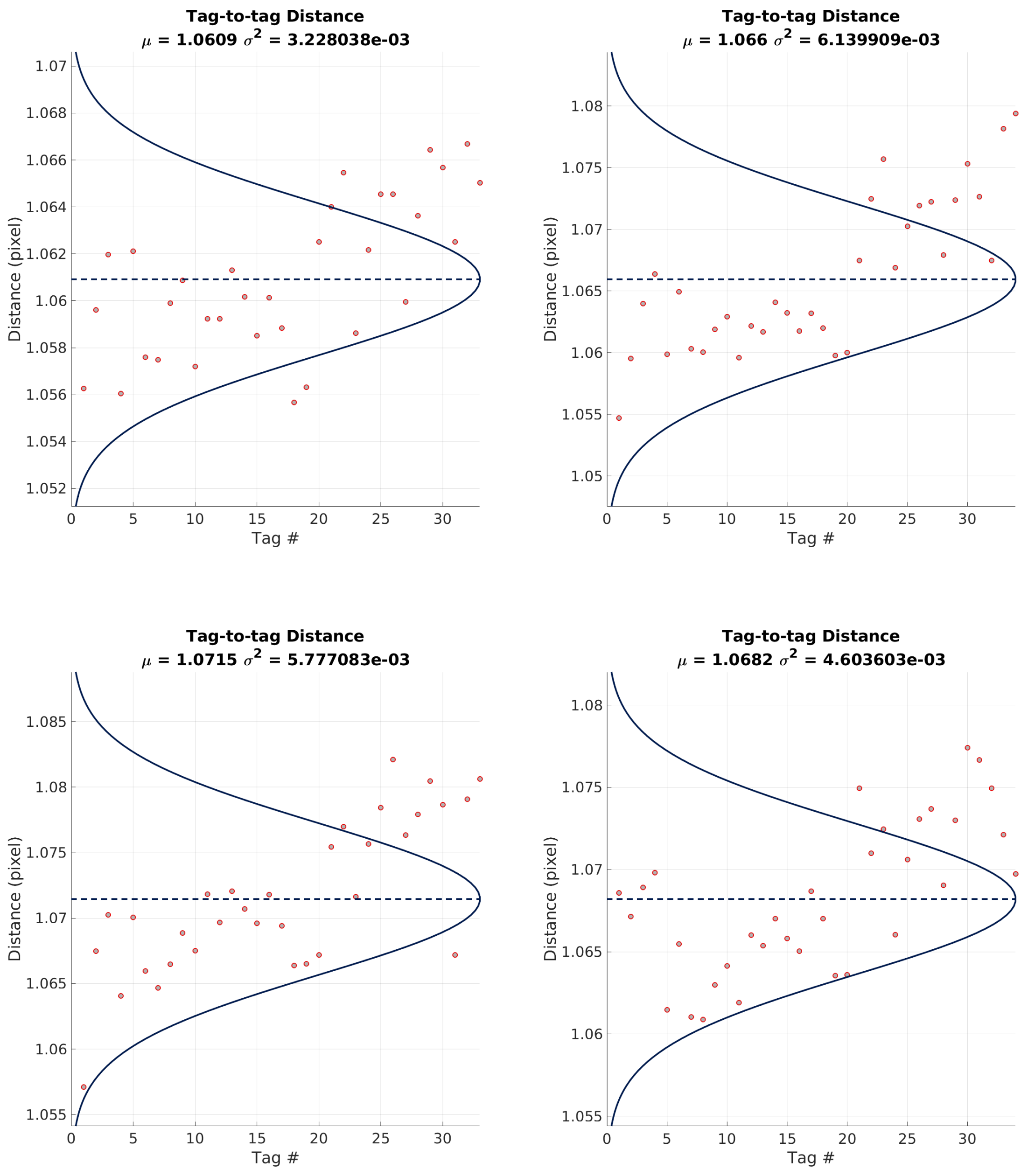


Figure 8.9: Distances between adjacent tags placed along a plane for different maps reconstructed by TagSLAM.

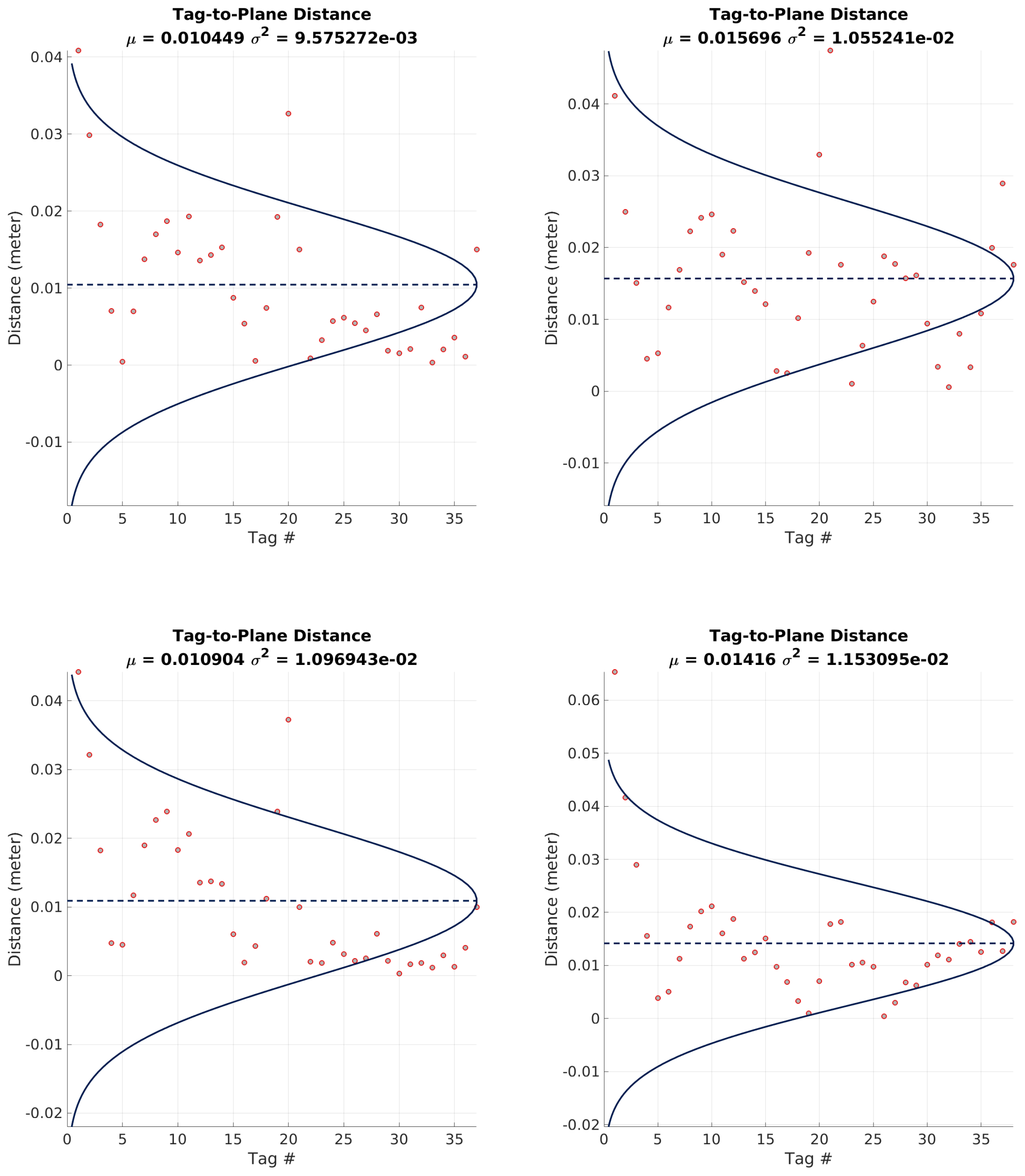


Figure 8.10: Distances between tags placed along a plane and the plane fit to these tags for different maps reconstructed by TagSLAM.

8.3 Coordinate Frame Transformations

The range-based estimators estimate the robot pose with respect to a local coordinate frame that varies over time. The origin of this coordinate frame is the point along the centerline closest to the 3D robot position and the frame triad is calculated based on the centerline tangent at its origin. On the other hand, TagSLAM estimates the robot pose as well as all the tag poses with respect to a specific tag defined as the origin. In order to compare the results, robot poses estimated by both type of estimators are transformed into a common coordinate frame. In particular, we transform the TagSLAM pose estimates into their corresponding local frames. This requires construction of the centerline from the tag poses.

TagSLAM estimates the orientations of the AprilTags along with their positions. Since the tag slabs abut against the tunnel walls at all their four corners thanks to the magnets, surface normals of tags can be used to estimate the centerline. The set of points calculated as the intersection of the normals of opposite tag pairs in the least-squares sense can then be used as the control points of a 3D Bernstein polynomial. The side of each tag is determined manually (*i.e.* right/left side of the tunnel). Opposite of a given tag is simply the closest tag in the opposite side of the tunnel in the Euclidean sense. The least-squares solution, $\tilde{\mathbf{p}}$, is calculated as

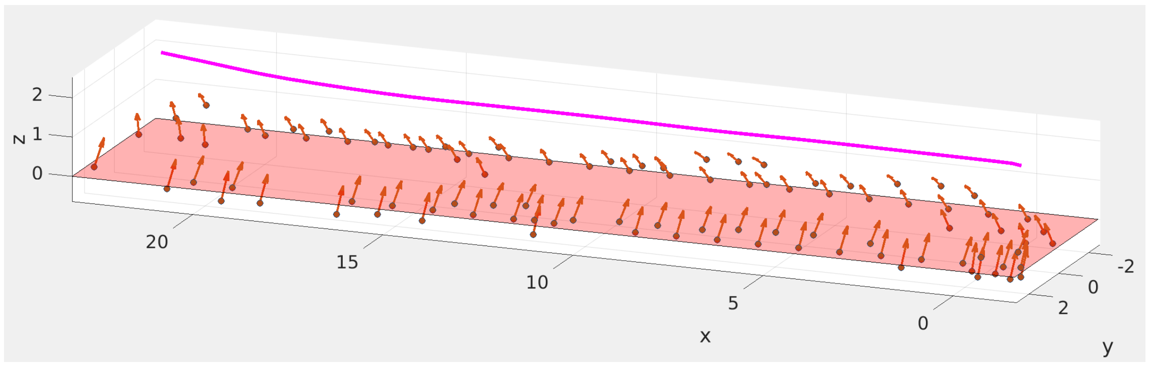
$$\tilde{\mathbf{p}} = \mathbf{S}^{-1}\mathbf{C} \tag{8.1}$$

where

$$\mathbf{S} = \sum_i \hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^\top - \mathbf{I} \tag{8.2}$$

$$\mathbf{C} = \sum_i \left(\hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^\top - \mathbf{I} \right) \mathbf{p}_i \tag{8.3}$$

where the summations run over all the points, $\mathbf{p}_i$, and their surface normals, $\hat{\mathbf{n}}_i$, included in the solution. Fig. 8.11 shows the centerlines approximated as 3D Bernstein polynomials with their control points estimated using the least-squares method.



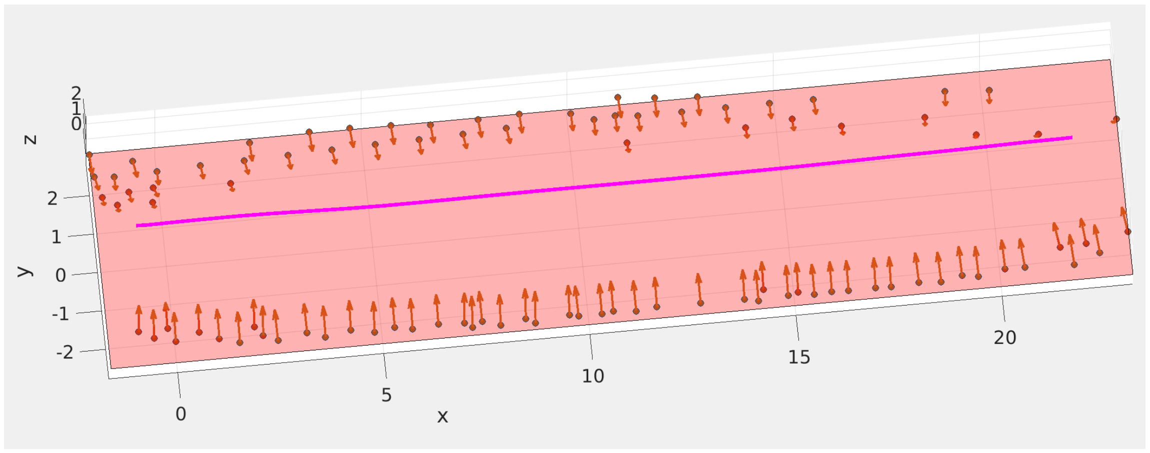


Figure 8.11: Centerline approximated as a 3D Bernstein polynomial. Its control points are the intersection of opposite tag normals.

Although it is not directly related to pose estimation, in Fig. 8.12 we show the estimated radii along the centerline. Similar to the centerline, radii is also interpolated using a Bernstein polynomial of degree 2. The first component is the coordinate of the projection of tags onto the approximated centerline. The coordinate of a point along the centerline is defined as the line integral from the beginning of the curve up to the point. The second component is the distance of a given tag from its projection onto the centerline, *i.e.* the local radius. As can be seen in these plots, the radius of the tunnel is not constant. It gets smaller closer to the scroll case around the turbine blades and increase towards the inclination. Although we do not have ground truth data, we can say that this is consistent with the actual tunnel shape. The radius estimates fluctuate after the 20^{th} meter around where the tunnel starts to include upwards. We believe this is due to the sparsity of tags.

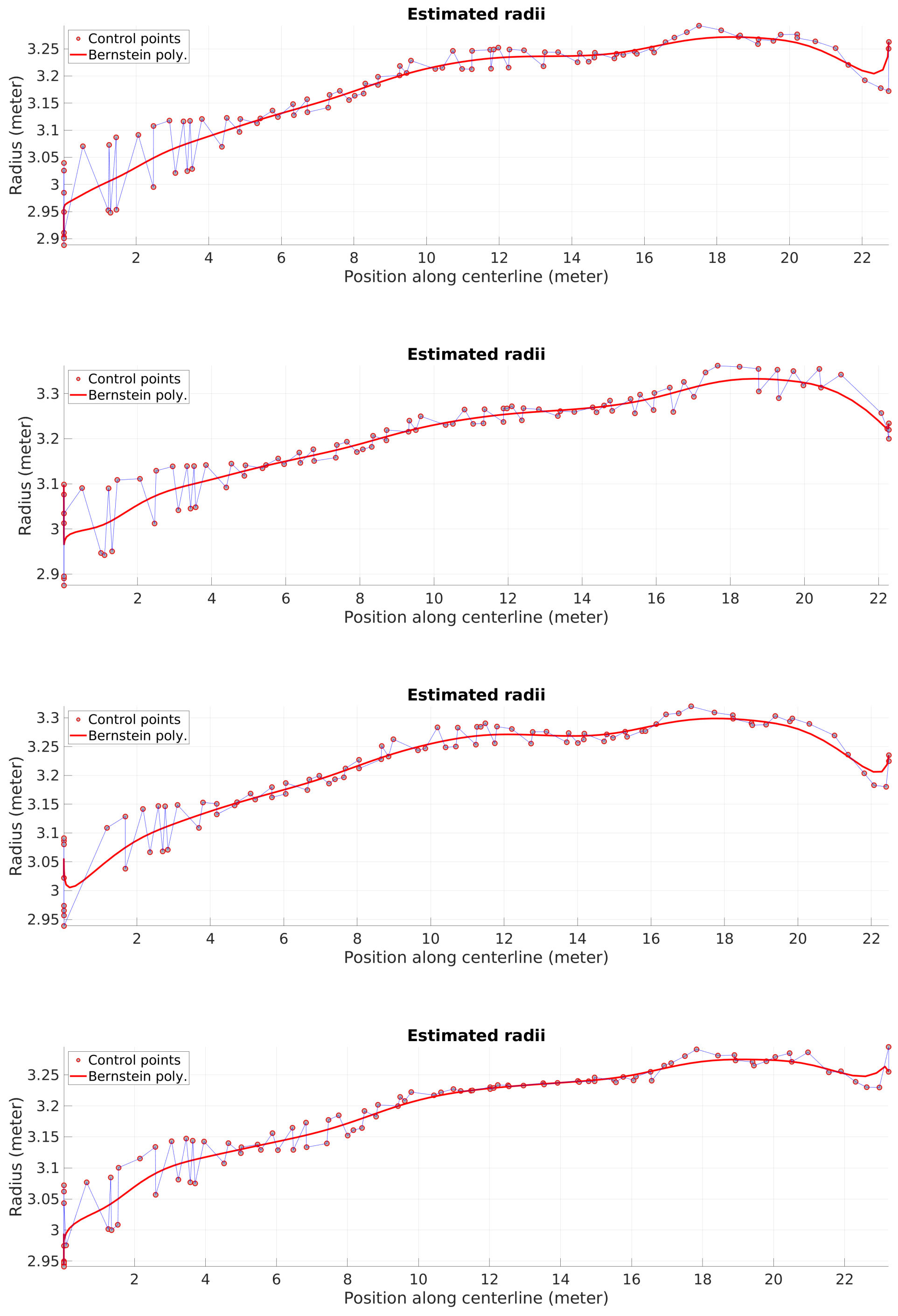


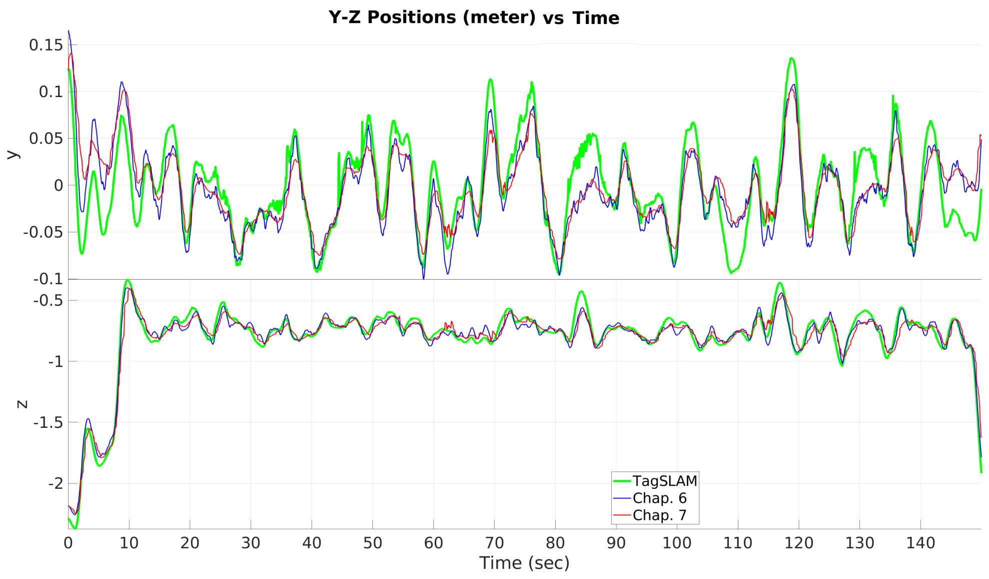
Figure 8.12: These plots show the distance of each map tag to the centerline approximated as a Bernstein polynomial, *i.e.* local radius. The red line is the continuous approximation of the tunnel radius with the individual radius estimates as its control points.

8.4 Comparison of Pose Estimates

In this section we present results for the three estimators which are the range-based methods presented in Chap. 6, Chap. 7 and the vision-based TagSLAM. The lateral and vertical positions as well as the full orientations are compared for 4 different datasets in Fig. 8.13-8.14-8.15-8.16. Range-based and the vision-based estimators are respectively filtering and optimization based approaches. The effect of this fundamental difference can be observed in the results. Especially, the difference in position estimates of two groups of approaches at peak points highlights the smoothing effect of the filtering. Due to the slow response of the filter, the discrepancy between results increase at most of the peak points. On the other hand, TagSLAM suffers from glitches since it does not inherently implement a smoothing mechanism. Except for these effects, it can be seen from the plots that the different in position is less than 10 cm on average.

Taking results of TagSLAM as the reference point, the two range-based approaches have comparable performance as seen in Fig. 8.13-8.14-8.15-8.16. The approach of Chap. 7 (Fig. 8.13), however, performs the poorest in yaw estimation with errors as large as 4 degrees. Such large errors occur mostly while to robot is close to the ground (taking off or landing). Due to the narrow field of view of the lidar, the central segment is estimated erroneously with a large translational offset from its neighboring segments causing the centerline twist and bend abruptly. This in turn affects the local coordinate frame, $\mathcal{L}(s = 0)$, which is used to infer the robot orientation.

Although the performance of the range-based approaches are similar in the plots, the method of Chap. 7 exhibits better performance when the tunnel circularity assumption fails. Its superiority is not only in the state estimation but it also constructs a more accurate local map. We refer to the experimental results of Chap. 7 for the details. The significance of this would be appreciated if map quality requirement for the obstacle avoidance is also considered. In conclusion, it can be argued that the method of Chap. 7 is a higher performing estimator and a better choice for the confined penstock setting.



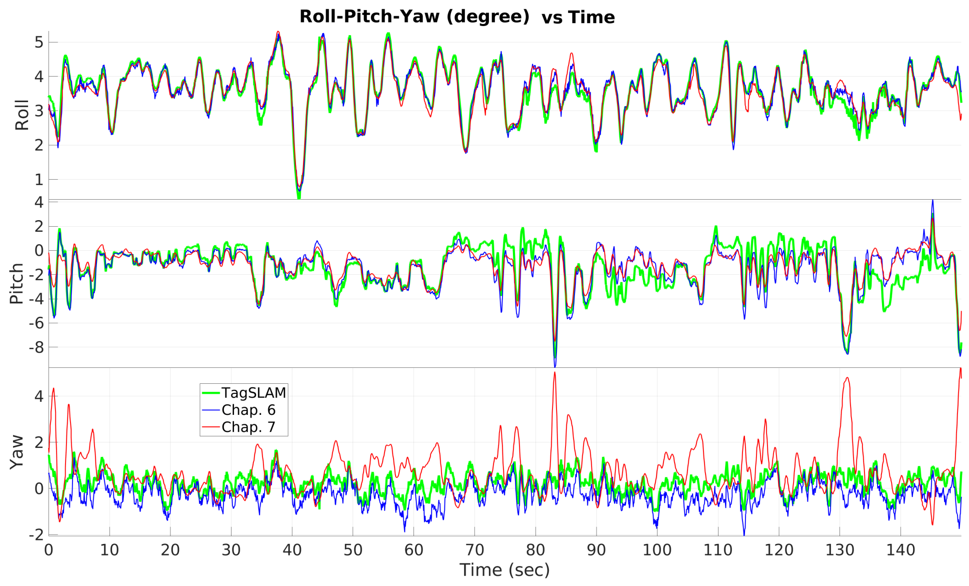
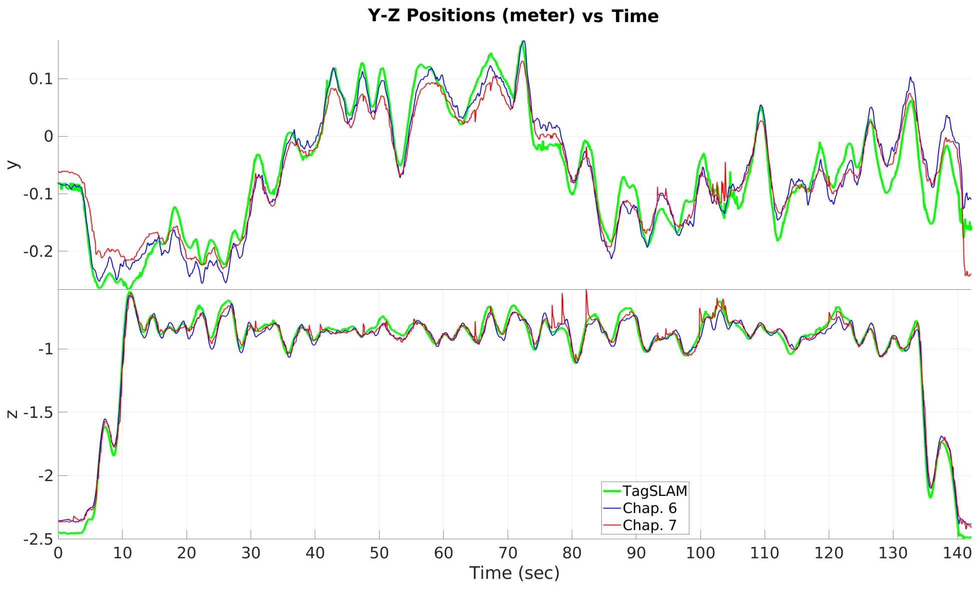


Figure 8.13: Comparison of the range-based approaches and TagSLAM on dataset #1.



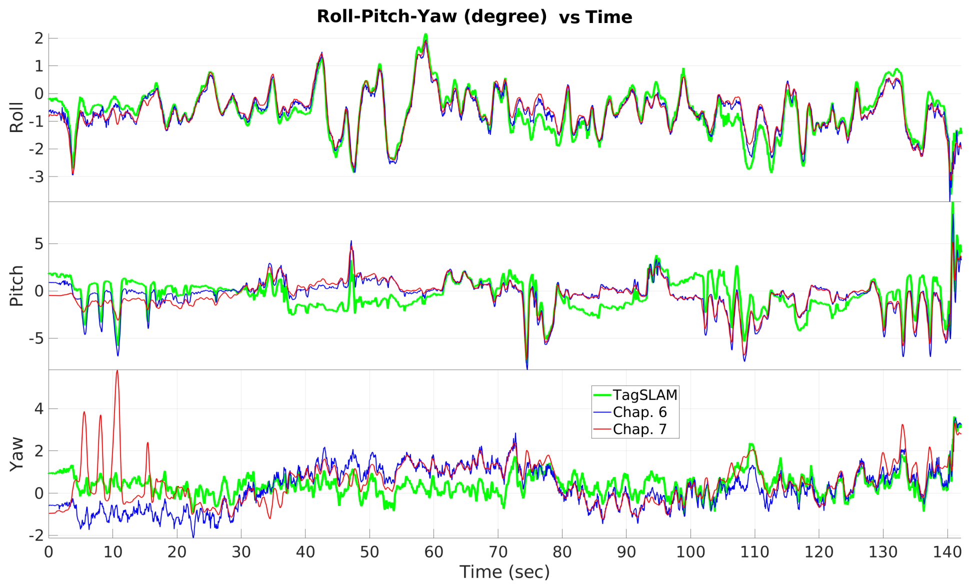
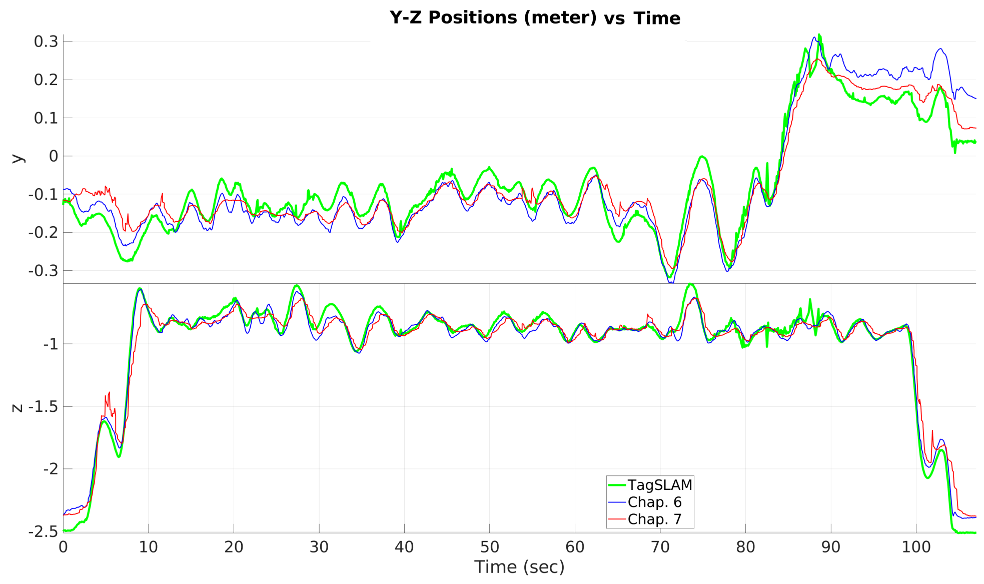


Figure 8.14: Comparison of the range-based approaches and TagSLAM on dataset #2.



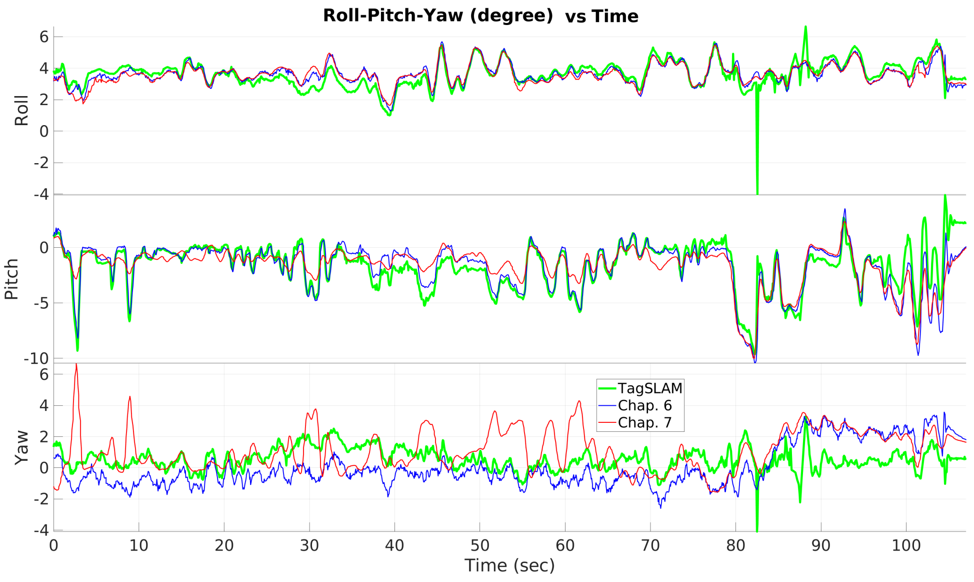
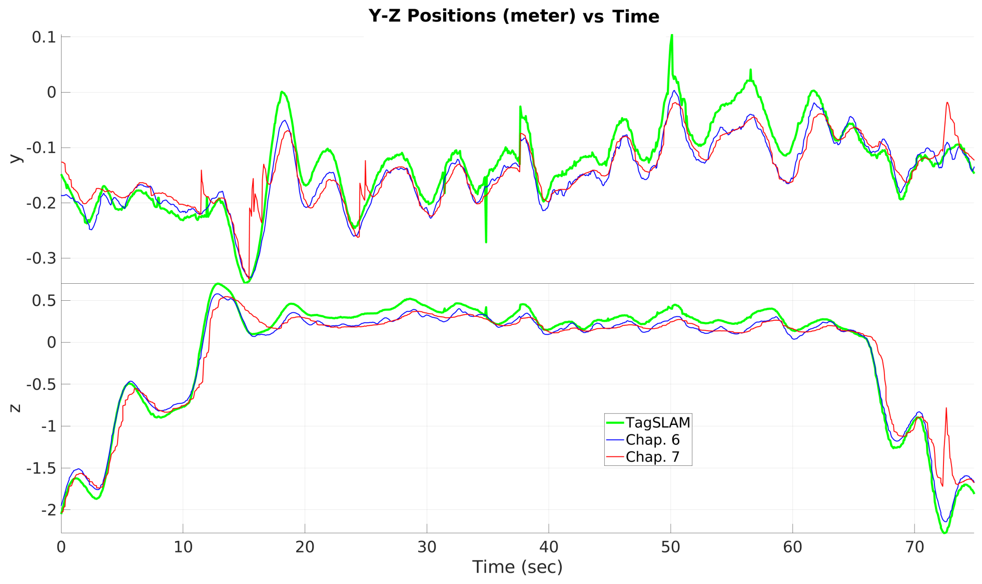


Figure 8.15: Comparison of the range-based approaches and TagSLAM on dataset #3.



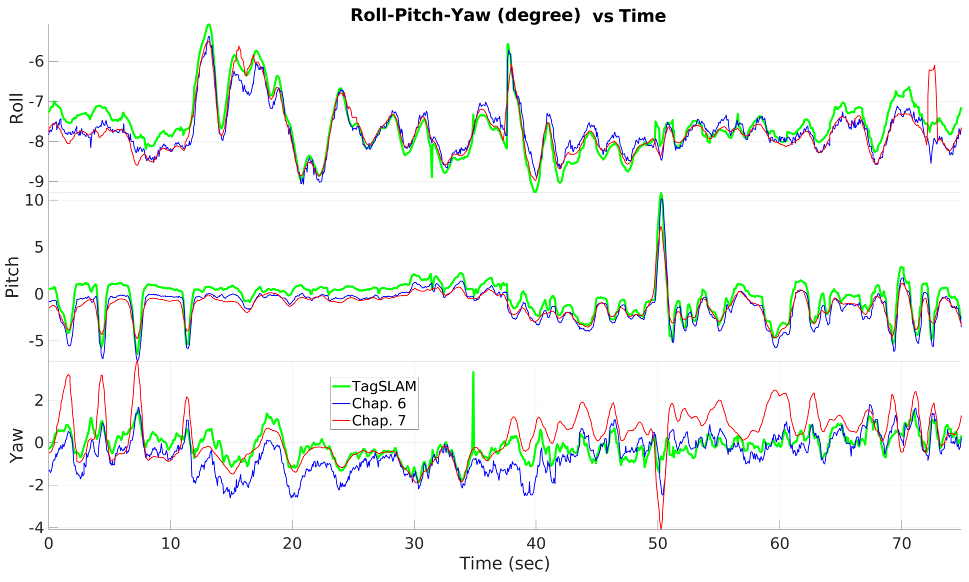


Figure 8.16: Comparison of the range-based approaches and TagSLAM on dataset #4.

Chapter 9

Conclusion

9.1 Key Contributions

In this this thesis, we study the state estimation and autonomous navigation of MAVs inside penstocks and tunnel-like, long, axisymmetric indoor environments. To our knowledge, ours is the only study in the field robotics literature that studies MAV autonomy in such environments.

The main contributions of the thesis are in Chap. 4-8. The first semi-autonomous flight inside a penstock is realized in Chap. 4 using an off-the-shelf MAV equipped with a minimal set of sensors including a single 2D laser scanner, an IMU and a low-end single-core processor. State estimation is performed using a Rao-Blackwellized particle filter which models the robot position along the long tunnel axis using particles. In Chap. 5, we present an extension to this work where a sensor fusion algorithm that uses multiple 2D laser scanners and cameras are utilized. The proposed fusion algorithm integrates optical flow information from the cameras to estimate the robot position along the tunnel axis. This chapter also proposes a 3D, offline visual inspection technique. In Chap. 6, we present an alternative state estimation and local mapping system that uses an onboard 3D lidar with a detailed measurement model. Unlike the first two approaches, this system does not require a prior map of the tunnel. Chap. 7 improves the approach of the previous chapter by imposing roto-translational constraints on the local map segments. Local map which, in Chap. 6, is constructed from *independently* estimated segments, are estimated using a holistic approach that uses *all* the segments improving the estimator robustness. Lastly, Chap. 8 provides datasets and benchmarks the performance of the last two approaches against a vision-based state estimator that relies on AprilTags. We note that the although we present results from only penstocks and university corridors, the techniques developed in this work can be extended to work in other tunnel-like environments such as mine shafts, highway tunnels. Furthermore, the proposed methods can be also be easily deployed on other robotics platforms such as wheeled or tracked vehicles with minor modifications.

9.2 Limitations

The techniques developed in the thesis assume a particular shape for the testing environment. Since the major concern of this work, in particular, is to fly inside penstocks, the environment is modeled as a cylindrical tube with variations in its radius, different bending profiles and smooth surfaces. In Chap. 7 we refer to such an environment as a *piecewise-smooth-generalized-cylinder* . Although we claim that the proposed methods can be used in other similar environments such as mine shafts, highway tunnels, building corridors etc., direct application of these methods as is might not perform as well as in penstocks. The proposed estimators can cope with slight deviations in the geometry of the environment from the assumed model. While the state definitions and the Kalman filter formulations can be directly used, point cloud processing algorithms and measurement models might require significant modifications. For example, the methods in Chap. 6-7 fit circular segments which would fail when flying inside building corridors with rectangular cross-sections. Furthermore, the same group of methods might fail inside mine shafts with irregular surfaces since the surface normal estimates will be much different than their expected directions failing the smooth surface assumption. Also environments with multiple branches would violate the PSGC assumption and will require a more sophisticated tunnel model.

9.3 Future Work

The approach detailed in Chap. 5 of this dissertation requires a prior map although, theoretically, the map can be estimated using the point cloud measurements from the two onboard 2D laser scanners. The proposed Kalman filter can be reformulated to also estimate the cylinder parameters where the parameters are inferred from the raw point cloud using the 5-point cylinder estimation approach which is referred to in Sec. 5.1. Since 2D laser scanners are cheaper and lighter compared to 3D lidars such as the one used in Chap. 6-7, inspection of penstocks dimensions of which are not available would be possible with smaller and cheaper platforms.

The optical flow-based axial velocity estimation method explained Chap. 5 can be used along with the method of Chap. 7 to estimate the full-state of the platform. However, the flow-based approach relies on the accuracy of the range-based position and yaw estimation which might in certain cases be inaccurate. Appending the state vector of our Kalman filter with image feature locations, and updating these and the local map in a tightly-coupled filtering framework would improve the robustness.

As stated in the previous section, the assumptions about the environment might be restrictive for application of the proposed methods in different settings. Among possible improvements, amending the segment estimation algorithms to handle non-circular cross-sections might not be as valuable since this would only require the formulation and a robust implementation of a model fitting algorithm. On the other hand, it would be more valuable and interesting to improve the centerline model to handle tunnels with branching off and merging. Also parametric approximate reconstruction of rough surfaces such as in mine shafts and a centerline model that can handle non-symmetric cross-sections would be a significant contribution.

Since this is primarily a field robotics research, testing different sensor modalities especially for estimating the axial position of the platform would be interesting. For example, Ultra-Wideband (UWB) sensors can be used in combination with cameras for this purpose. When the camera field of view is obscured by dust, mist particles, the onboard UWB sensors can still provide accurate distance measurement from a base station. Finally, surfaces of well-maintained sections of penstocks almost do not have any visual texture prohibiting optical flow calculation. The platform can be equipped with a squirt gun filled with non-corrosive dye to artificially paint visual texture on the tunnel surface.

Bibliography

[1] R. E. Kalman, "New Approach to Linear Filtering and Prediction Problems", *J. Basic Eng.*, vol. 82, no. 1, 35–45 (1960) (11 pages), 1960.

[2] G. Watson, "Equatorial distributions on a sphere", *Biometrika*, vol. 52, no. 1/2, pp. 193–201, 1965.

[3] G. J. Agin and T. O. Binford, "Computer description of curved objects", in *Proceedings of the 3rd international joint conference on Artificial intelligence*, Morgan Kaufmann Publishers Inc., 1973, pp. 629–640.

[4] C. Bingham, "An antipodally symmetric distribution on the sphere", *The Annals of Statistics*, pp. 1201–1225, 1974.

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

[6] R. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space", *IEEE Trans. Syst., Man, Cybern.*, vol. 13, no. 3, pp. 190–197, 1983.

[7] J. R. Magnus, "On Differentiating Eigenvalues and Eigenvectors", *Econom. Theory*, vol. 1, no. 02, pp. 179–191, Aug. 1985.

[8] K. Shoemake, "Animating rotation with quaternion curves", *ACM SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, 1985.

[9] C. Harris and M. Stephens, "A Combined Corner and Edge Detector", in *Procedings of the Alvey Vision Conference 1988*, 1988, pp. 23.1–23.6.

[10] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation", *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[11] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry", *Artificial Intelligence*, vol. 78, no. 1, pp. 87–119, 1995, Special Volume on Computer Vision.

[12] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems", in *Signal processing, sensor fusion, and target recognition VI*, International Society for Optics and Photonics, vol. 3068, 1997, pp. 182–193.

[13] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping", *Auton. Robots*, vol. 4, no. 4, pp. 333–349, Oct. 1997.

[14] P. Torr and A. Zisserman, "Robust computation and parametrization of multiple view relations", in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan. 1998, pp. 727–732.

[15] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review", *International journal of computer vision*, vol. 27, no. 2, pp. 161–195, 1998.

[16] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 476–480, May 1999.

[17] K. V. Mardia and P. E. Jupp, *Directional Statistics [Hardcover]*. J. Wiley, 1999, p. 350, ISBN: 0471953334.

[18] S. Thrun, D. Fox, W. Burgard, F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots", *Proc. 1999 IEEE Int. Conf. Robot. Autom. (Cat. No.99CH36288C)*, vol. 2, no. May, pp. 1322–1328, 1999.

[19] A. Doucet, N. D. Freitas, K. Murphy, and Stuart Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks", *UAI'00 Proc. Sixt. Conf. Uncertain. Artif. Intell.*, pp. 176–183, 2000.

[20] J. Y. Bouguet, “Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm”, Tech. Rep., 2001. [Online]. Available: `http://robots.stanford.edu/cs223b04/algo%7B%5C_%7Daffine%7B%5C_%7Dtracking.pdf`.

[21] T. Chaperon and F. Goulette, “Extracting cylinders in full 3d data using a random sampling method and the gaussian image”, 2001.

[22] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (SLAM) problem”, *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

[23] S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm”, English, in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, IEEE Comput. Soc, 2001, pp. 145–152.

[24] M. Antone and S. Teller, “Scalable extrinsic calibration of omni-directional image networks”, *International Journal of Computer Vision*, vol. 49, no. 2-3, pp. 143–174, 2002.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem”, in *Proceeding Eighteenth Natl. Conf. Artif. Intell.*, vol. 68, 2002, pp. 593–598.

[26] S. Thrun, “Robotic Mapping: A Survey”, *Science (80-.).*, vol. 298, no. February, pp. 1–35, 2002.

[27] A. Ansar and K. Daniilidis, “Linear pose estimation from points or lines”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578–589, 2003.

[28] E. Kraft, “A quaternion-based unscented Kalman filter for orientation tracking”, in *Proc. 6th Int. Conf. Inf. Fusion, FUSION 2003*, vol. 1, 2003, pp. 47–54.

[29] M. A. Paskin, “Thin Junction Tree Filters for Simultaneous Localization and Mapping”, *Int. Jt. Conf. Artif. Intell.*, 2003.

[30] K. Low, "Linear Least-squares Optimization for Point-to-plane ICP Surface Registration", low.report04, Tech. Rep. February, 2004, pp. 2–4. [Online]. Available: `https://www.iscs.nus.edu.sg/%7B%5C%%7D7B%7B~%7D%7B%5C%%7D7Dlowkl/publications/lowk%7B%5C_%7Dpoint-to-plane%7B%5C_%7Dicp%7B%5C_%7Dtechrep.pdf`.

[31] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[32] D. Nistér, "An efficient solution to the five-point relative pose problem", *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 0756–777, 2004.

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

[34] T. Rabbani and F. Van Den Heuvel, "3d industrial reconstruction by fitting csg models to a combination of images and point clouds", *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. 35, no. B5, p. 2, 2004.

[35] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters", in *Int. J. Rob. Res.*, vol. 23, 2004, pp. 693–716.

[36] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments", *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[37] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping", *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 196–207, 2005.

[38] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots", *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 364–375, Jun. 2005.

[39] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* MIT press, 2005.

[40] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part I”, *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, Jun. 2006.

[41] ——, “Simultaneous localization and mapping (SLAM): part II”, *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.

[42] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features”, *Eur. Conf. Comput. Vis.*, vol. 3951, pp. 404–417, 2006.

[43] D. M. Cole and P. M. Newman, “Using laser range data for 3D SLAM in outdoor environments”, *IEEE Int. Conf. Robot. Autom.*, no. May, pp. 1556–1563, 2006.

[44] F. Dellaert and M. Kaess, “Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing”, *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[45] U. Frese, “Treemap: An O(log) Algorithm for Simultaneous Localization and Mapping”, *Auton. Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[46] C. de Granville, J. Southerland, and A. H. Fagg, “Learning grasp affordances through human demonstration”, in *Proceedings of the International Conference on Development and Learning (ICDL’06)*, 2006.

[47] J. Kannala and S. S. Brandt, “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[48] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006, pp. 1–826, ISBN: 9780511546877. DOI: 10.1017/CBO9780511546877.

[49] D. Lichtblau, “Cylinders through five points: Complex and real enumerative geometry”, in *International Workshop on Automated Deduction in Geometry*, Springer, 2006, pp. 80–97.

[50] E. Olson, J. Leonard, and S. Teller, “Fast iterative alignment of pose graphs with poor initial estimates”, in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, 2006, pp. 2262–2269.

[51] E. Rosten and T. Drummond, “Machine Learning for High Speed Corner Detection”, *Computer Vision – ECCV 2006*, vol. 1, pp. 430–443, 2006.

[52] A. Censi, “An accurate closed-form estimate of ICP’s covariance”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, IEEE, Apr. 2007, pp. 3167–3172.

[53] ——, “On achievable accuracy for range-finder localization”, *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 4170–4175, Apr. 2007.

[54] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós, “Mapping Large Loops with a Single Hand-Held Camera”, in *Robot. Sci. Syst.*, 2007.

[55] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: real-time single camera SLAM.”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[56] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters”, *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[57] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, “A tree parameterization for efficiently computing maximum likelihood maps using gradient descent”, *Rss*, p. 8, 2007.

[58] S. J. Julier and J. J. LaViola, “On kalman filtering with nonlinear equality constraints”, *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2774–2784, 2007.

[59] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces”, in *2007 6th IEEE ACM Int. Symp. Mix. Augment. Reality, ISMAR*, 2007.

[60] M. Magnusson, A. Lilienthal, and T. Duckett, “Scan registration for autonomous mining vehicles using 3D-NDT”, *J. F. Robot.*, vol. 24, no. 10, pp. 803–827, 2007.

[61] M. Montemerlo and S. Thrun, “FastSLAM 2.0”, *Springer Tracts Adv. Robot.*, vol. 27, pp. 63–90, 2007.

[62] T. Tuytelaars and K. Mikolajczyk, “Local Invariant Feature Detectors: A Survey”, *Found. Trends®Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, 2007.

[63] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM", *International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[64] A. Censi, "An ICP variant using a point-to-line metric", in *Proc. - IEEE Int. Conf. Robot. Autom.*, IEEE, May 2008, pp. 19–25.

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

[66] L. M. Paz, J. D. Tardos, and J. Neira, "Divide and Conquer: EKF SLAM in O(n)", *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1107–1120, 2008.

[67] T. Tuytelaars, K. Mikolajczyk, *et al.*, "Local invariant feature detectors: A survey", *Foundations and trends® in computer graphics and vision*, vol. 3, no. 3, pp. 177–280, 2008.

[68] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying", in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2878–2883.

[69] E. B. E. B. E. B. Olson, "Real-time correlative scan matching", in *2009 IEEE Int. Conf. Robot. Autom.*, IEEE, May 2009, pp. 4387–4393.

[70] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM", *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.

[71] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping", in *IEEE International Conference on Robotics and Automation*, 2010, pp. 273–278.

[72] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, http://eigen.tuxfamily.org, 2010.

[73] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme", in *2010 ieee intelligent vehicles symposium*, IEEE, 2010, pp. 486–492.

[74] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed", *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 56–65, Sep. 2010.

[75] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, 2010.

[76] R. B. Rusu, “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments”, PhD thesis, TECHNISCHE UNIVERSITÄT MÜNCHEN, Nov. 2010. [Online]. Available: `http://link.springer.com/10.1007/s13218-010-0059-6%20http://link.springer.com/article/10.1007/s13218-010-0059-6`.

[77] P. Hansen, H. Alismail, B. Browning, and P. Rander, “Stereo visual odometry for pipe mapping”, in *IEEE Int. Conf. Intell. Robot. Syst.*, 2011, pp. 4020–4025.

[78] P. Hansen, H. Alismail, P. Rander, and B. Browning, “Monocular visual odometry for robot localization in LNG pipes”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3111–3116.

[79] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization”, English, in *2011 IEEE International Conference on Robotics and Automation*, IEEE, May 2011, pp. 3607–3613.

[80] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.

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

[82] R. B. Rusu and S. Cousins, “3D is here: point cloud library”, *IEEE Int. Conf. Robot. Autom.*, pp. 1–4, May 2011.

[83] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial]”, *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[84] S. Shen, N. Michael, and V. Kumar, “Autonomous multi-floor indoor navigation with a computationally constrained MAV”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, IEEE, May 2011, pp. 20–25.

[85] S. Weiss, D. Scaramuzza, and R. Siegwart, “Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments”, *J. F. Robot.*, vol. 28, no. 6, pp. 854–874, Nov. 2011.

[86] a. Bachrach, S. Prentice, R. He, P. Henry, a. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments”, *Int. J. Rob. Res.*, vol. 31, no. 11, pp. 1320–1343, 2012.

[87] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, C. Hurzeler, G. Caprari, R. Siegwart, C. Hürzeler, G. Caprari, C. Hurzeler, G. Caprari, and R. Siegwart, “Aerial service robots for visual inspection of thermal power plant boiler systems”, *Appl. Robot. Power Ind. (CARPI), 2012 2nd Int. Conf.*, pp. 70–75, 2012.

[88] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”, *Comput. Vis. Pattern Recognition)*, pp. 3354–3361, 2012.

[89] J. Glover, G. Bradski, and R. B. Rusu, “Monte carlo pose estimation with quaternion kernels and the bingham distribution”, in *Robotics: science and systems*, vol. 7, 2012, p. 97.

[90] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, “Real-Time Plane Segmentation Using RGB-D Cameras”, in *Robot Soccer World Cup XV. Rob. 2011*, vol. 7416 LNCS, 2012, pp. 306–317.

[91] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “ISAM2: Incremental smoothing and mapping using the Bayes tree”, *Int. J. Rob. Res.*, vol. 31, no. 2, pp. 216–235, 2012.

[92] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Visual SLAM: Why filter?”, *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, 2012.

[93] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems”, in *IEEE Int. Conf. Intell. Robot. Syst.*, 2012, pp. 573–580.

[94] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, “Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue”, *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 46–56, Sep. 2012.

[95] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen, “Planar surface SLAM with 3D and 2D sensors”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3041–3048.

[96] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset”, *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[97] J. Glover and S. Popovic, “Bingham procrustean alignment for object detection in clutter”, in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 2158–2165.

[98] K. Kawashima, S. Kanai, and H. Date, “Automatic recognition of piping system from laser scanned point clouds using normal-based region growing”, *ISPRS Ann Photogramm Remote Sens Spatial Inf Sci, II-5 W*, vol. 2, pp. 121–126, 2013.

[99] G. Kurz, I. Gilitschenski, S. Julier, and U. D. Hanebeck, “Recursive estimation of orientation based on the bingham distribution”, in *Information Fusion (FUSION), 2013 16th International Conference on*, IEEE, 2013, pp. 1487–1494.

[100] T. Lee, M. Leok, and N. H. Mcclamroch, “Nonlinear robust tracking control of a quadrotor UAV on SE(3)”, *Asian J. Control*, vol. 15, no. 2, pp. 391–408, 2013.

[101] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, “A UAV system for inspection of industrial facilities”, in *IEEE Aerospace Conference Proceedings*, 2013.

[102] M. Pivtoraiko, D. Mellinger, and V. Kumar, “Incremental micro-UAV motion replanning for exploring unknown environments”, in *Proc. - IEEE Int. Conf. Robot. Autom.*, IEEE, May 2013, pp. 2452–2458.

[103] C. Richter, A. Bry, and N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments”, *Isrr*, no. Isrr, pp. 1–16, 2013.

[104] S. Sra and D. Karp, "The multivariate watson distribution: Maximum-likelihood estimation and other aspects", *Journal of Multivariate Analysis*, vol. 114, pp. 256–269, 2013.

[105] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-Plane SLAM for Hand-Held 3D Sensors", in *Robot. Autom. (ICRA); 2013 IEEE Int. Conf.*, 2013, pp. 5182–5189.

[106] J. Engel, T. Schöps, D. Cremers, T. Sch, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM", *Eccv*, vol. 8690 LNCS, no. PART 2, pp. 1–16, 2014.

[107] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry", in *2014 IEEE Int. Conf. Robot. Autom.*, IEEE, May 2014, pp. 15–22.

[108] I. Gilitschenski, G. Kurz, S. J. Julier, and U. D. Hanebeck, "A new probability distribution for simultaneous representation of uncertain position and orientation", in *Information Fusion (FUSION), 2014 17th International Conference on*, IEEE, 2014, pp. 1–7.

[109] J. Glover and L. P. Kaelbling, "Tracking the spin on a ping pong ball with the quaternion bingham filter", in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 4133–4140.

[110] A. Ortiz, F. Bonnin-Pascual, and E. Garcia-Fidalgo, "Vessel Inspection: A Micro-Aerial Vehicle-based Approach", *J. Intell. Robot. Syst.*, vol. 76, no. 1, pp. 151–167, Sep. 2014.

[111] T. Özaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Inspection of penstocks and featureless tunnel-like environments using micro UAVs", in *Springer Tracts Adv. Robot.*, vol. 105, 2014, pp. 123–136.

[112] A. K. Patil, S. S. Park, P. Holi, and Y. H. Chai, "Automatic pipeline generation by the sequential segmentation and skelton construction of point cloud", *Advanced Science and Technology Letters*, vol. 67, no. 2014, pp. 43–47, 2014.

[113] S. Shen, "Autonomous Navigation in Complex Indoor and Outdoor Environments With Micro Aerial Vehicles", PhD thesis, University of Pennsylvania, 2014, ISBN: 9780874216561. DOI: 10.1007/s13398-014-0173-7.2.

[114] R. Zlot and M. Bosse, "Efficient Large-Scale 3D Mobile Mapping and Surface Reconstruction of an Underground Mine", in, Springer, Berlin, Heidelberg, 2014, pp. 479–493. DOI: 10.1007/978-3-642-40686-7_32.

[115] ——, "Efficient Large-scale Three-dimensional Mobile Mapping for Underground Mines", *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, Sep. 2014.

[116] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture", *Autom. Sci. Eng. (CASE), 2015 IEEE Int. Conf.*, pp. 462–469, 2015.

[117] P. Gohl, M. Burri, S. Omari, J. Rehder, J. Nikolic, M. Achtelik, and R. Siegwart, "Towards autonomous mine inspection", in *Proc. 3rd Int. Conf. Appl. Robot. Power Ind. CARPI 2014*, 2015.

[118] P. Hansen, H. Alismail, P. Rander, and B. Browning, "Visual mapping for natural gas pipe inspection", *The International Journal of Robotics Research*, vol. 34(4-5), no. 4-5, pp. 532–558, Nov. 2015.

[119] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies", *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, May 2015.

[120] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of MAVs using RGB-D sensors", in *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2015-June, 2015, pp. 4021–4028.

[121] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", *IEEE TRANSACTIONS ON ROBOTICS*, vol. 31, no. 5, 2015.

[122] F. Pomerleau, F. Colas, and R. Siegwart, “A Review of Point Cloud Registration Algorithms for Mobile Robotics”, *Found. Trends Robot.*, vol. 4, no. 1, pp. 1–104, 2015.

[123] J. Zhang and S. Singh, “Visual-lidar odometry and mapping: Low-drift, robust, and fast”, in *Robot. Autom. (ICRA), 2015 . . .*, IEEE, May 2015, pp. 2174–2181.

[124] L. Busé, A. Galligo, and J. Zhang, “Extraction of cylinders and cones from minimal point sets”, *Graphical Models*, vol. 86, pp. 1–12, 2016.

[125] G. S. Chirikjian and A. B. Kyatkin, *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Courier Dover Publications, 2016.

[126] I. Gilitschenski, G. Kurz, S. J. Julier, and U. D. Hanebeck, “Unscented orientation estimation based on the bingham distribution”, *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 172–177, 2016.

[127] T. Özaslan, K. Mohta, J. Keller, Y. Mulgaonkar, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, “Towards fully autonomous visual inspection of dark featureless dam penstocks using MAVs”, in *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016-Novem, Oct. 2016, pp. 4998–5005.

[128] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, “Visual Servoing of Quadrotors for Perching by Hanging From Cylindrical Objects”, *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, 2016.

[129] H. Alismail, M. Kaess, B. Browning, and S. Lucey, “Direct Visual Odometry in Low Light Using Binary Descriptors”, *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 444–451, 2017.

[130] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, “Autonomous Navigation and Mapping for Inspection of Penstocks and Tunnels With MAVs”, *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1740–1747, Jul. 2017.

[131] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous Navigation and Mapping for Inspection of Penstocks and Tunnels With MAVs", *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1740–1747, Jul. 2017.

[132] R. A. Srivatsan, M. Xu, N. Zevallos, and H. Choset, "Bingham distribution-based linear filter for online pose estimation", in *Robotics: Science and Systems*, 2017.

[133] J. Zhang and S. Singh, "Enabling aggressive motion estimation at low-drift and accurate mapping in real-time", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, pp. 5051–5058.

[134] ——, "Low-drift and real-time lidar odometry and mapping", *Auton. Robots*, vol. 41, no. 2, pp. 401–416, Feb. 2017.

[135] B. Pfrommer and K. Daniilidis, "Tagslam: Robust slam with fiducial markers", *arXiv preprint arXiv:1910.00679*, 2019.

[136] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison, "Dense Planar SLAM", in *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on.*

[137] R. Van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation", in *2001 IEEE Int. Conf. Acoust. Speech, Signal Process. Proc. (Cat. No.01CH37221)*, vol. 6, IEEE, pp. 3461–3464.