2021

Planning Algorithms Under Uncertainty for a Team of a UAV and a UGV for Underground Exploration

Matteo De Petrillo
West Virginia University, madepetrillo@mix.wvu.edu

Planning Algorithms Under Uncertainty for a Team of a UAV and a UGV for Underground Exploration

Matteo De Petrillo

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Aerospace Engineering

Jason N. Gross, Ph.D., Chair
Guilherme A. S. Pereira, Ph.D.
Marcello M. Napolitano, Ph.D.
Natalia A. Schmid, Ph.D.
Yu Gu, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2021

Keywords: Localization, Unmanned Aerial Vehicle, Unmanned Ground Vehicle, Belief Space, Gaussian Process, Autonomy

Abstract

Planning Algorithms Under Uncertainty for a Team of a UAV and a UGV for Underground Exploration

Matteo De Petrillo

Robots' autonomy has been studied for decades in different environments, but only recently, thanks to the advance in technology and interests, robots for underground exploration gained more attention. Due to the many challenges that any robot must face in such harsh environments, this remains an challenging and complex problem to solve.

As technology became cheaper and more accessible, the use of robots for underground exploration increased. One of the main challenges is concerned with robot localization, which is not easily provided by any Global Navigation Services System (GNSS). Many developments have been achieved for indoor mobile ground robots, making them the easiest fit for subterranean exploration. With the commercialization of small drones, the potentials and benefits of aerial exploration increased along with challenges connected to their dynamics.

This dissertation presents two path planning algorithms for a team of robots composed of an Unmanned Ground Vehicle (UGV) and an Unmanned Aerial Vehicle (UAV) with the task of exploring a subterranean environment. First, the UAV's localization problem is addressed by fusing different sensors present on both robots in a centralized manner. Second, a path planning algorithm that minimizes the UAV's localization error is proposed. The algorithm propagates the UAV motion model in the Belief Space, evaluating for potential exploration routes that optimize the sensors' observations. Third, a new algorithm is presented, which results to be more robust to different environmental conditions that could affect the sensor's measurements. This last planning algorithm leverages the use of machine learning, in particular the Gaussian Process, to improve the algorithm's knowledge of the surrounding environment pointing out when sensors provide poor observations. The algorithm utilizes real sensor measurements to learn and predict the UAV's localization error.

Extensive results are presented for the first two parts regarding the UAV's localization and the path planning algorithm in the belief space. The localization algorithm is supported with real-world scenario experimental results, while the belief space planning algorithm has been extensively tested in a simulated environment. Finally, the last approach has also been tested in a simulated environment and showed its benefits compared to the first planning algorithm.

Contents

List of Figures

List of Tables

Contributing Papers

Part of the presented work and its relative results are a direct derivation of published or submitted articles. Some of them contribute to specific chapters as described below.

Ch. 2 Ryan Watson, Nicholas Ohi, Scott Harper, Cagri Kilic, Chizhao Yang, Jacob Hikes, **Matteo De Petrillo**, Jared Strader, Gabrielle Hedrick, Hayden Nichols, Emily Upton, Connor Kirk, Keelan Hendricks, Dylan Reynolds, JB Darr, Jonas A Bredu, Eric Lagnese, Yu Gu, Jason N Gross. "A Rover and Drone Team for Subterranean Environments: System Design Overview".

Ch. 3 Gross Jason, **De Petrillo Matteo**, Beard Jared, Nichols Hayden, Swiger Tommy, Watson Ryan, Kirk Connor, Kilic Cagri, Hikes Jacob, Upton Emily, Ross Derek, Russell Matt, Gu Yu, Griffin Christopher. "Field-Testing of a UAV-UGV Team for GNSS-Denied Navigation in Subterranean Environments". Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), Miami, Florida, September 2019, pp. 2112-2124.

Ch. 4 **M. De Petrillo**, J. Beard, Y. Gu and J. N. Gross. "Search Planning of a UAV/UGV Team With Localization Uncertainty in a Subterranean Environment", in IEEE Aerospace and Electronic Systems Magazine, vol. 36, no. 6, pp. 6-16, 1 June 2021, doi: 10.1109/MAES.2021.3065041.

Ch. 6 **M. De Petrillo**, D. Ross, and J.N. Gross, 2021, Nov. "Gaussian Process based navigation algorithm for a UGV-UAV team underground exploration". Submitted to AIAA Journal of Aerospace Information Systems, November 2021.

List of Acronyms

DARPA Defence Advanced Research Projects Agency 3

GNSS Global Navigation Satellite System 2

LiDAR Light Detection And Ranging 7

NED North East Down 12

BSP Belief Space Planning 9

UAV Unmanned Aerial Vehicle ii

UGV Unmanned Ground Vehicle ii

VIO Visual-Inertial Odometry 3

PRM Probabilistic Road Map 4

POMDP Partially Observable Markov Decision Process 5

GP Gaussian Process 7

DOF Degree of Freedom 12

DCM Direction Cosine Matrix 13

UWB Ultra Wide-Band 17

ROS Robot Operating System 17

LOS Line of Sight 8

Acknowledgments

The achievements described within this dissertation and all the moments that brought me to this point have been driven by the support of amazing friends and mentors. This page is for them.

There are not enough ways to thank **Dr. Jason Gross** to be an exceptional advisor and role model. This work would not have been possible without your guide. But most of all, I would like to thank you for not making me feel lost during this journey.

A special thanks go to my committee members. Thanks for your advice, availability, and patience. Also, a proper acknowledgment goes to Dr. Chris Griffin and his students for their availability and help in using the testing facility to collect important data.

Thanks to the close colleagues that dedicated some time to clear my ideas during this long process, and also thank to faraway friends for their priceless support.

And to conclude, the most important acknowledgment goes to my wife Maggie De Petrillo for her unconditional support and my parents that backed all my decisions since I can remember. This achievement is for all of you.

This research was supported in part by the Benjamin M. Statler Fellowship,and by an academic grant from the National Geospatial-Intelligence Agency (Award No. HM0476-18-1-2000, Project Title: Planning Algorithms Under Uncertainty for a UAV UGV Team for Underground Exploitation). Approved for public release, 22-146.

1

Introduction

1.1 Problem Overview

Nowadays, the use of robots in everyday life is becoming more usual for a variety of scenarios. The most commonly proposed applications include: maintenance services [1] [2], industrial facility inspection [3], and search and rescue [4]. For search and rescue scenarios in hazardous environments, there is a clear need to replace some of the first responders with mobile robots. In this

scenario, mobile robots, not only can save lives but could also perform better due to the variety of sensors available and the fact that they are less susceptible to conditions like dust or poor lighting conditions. However, many challenges must be addressed to have a fully autonomous robot or team of robots in these environments. Of the many challenges, an important difficulty arises from attempting to navigate in these environments due to no access to the Global Navigation Satellite System (GNSS) signals and unknown obstacles.

Extensive research has been done on the use of robots in subterranean environments as is discussed in [5]. With a specific focus on underground mines, Thrun at al. [6] presented a ground rover designed to explore and map these harsh environments. Nowadays, technology and the miniaturization of electronic components has allowed for the reduction of the size of UAVs, such as commercial drones, making it possible to leverage their ability to traverse a variety of environments and complete a large number of tasks. For example, Azhari et al. [7] used a UAV for underground mine mapping, taking advantage of its six degrees of freedom. To autonomously navigate in subterranean tunnels or mines, Papachristos et al. [8] provided an unmanned aerial system capable of performing Simultaneous Localization and Mapping (SLAM) fusing different sensors to provide valuable information in dark conditions, presence of dust, etc. The authors also provided an uncertainty-aware path planning strategy for autonomous micro UAV to balance exploration and mapping in real-time [9].

Due to the limited capability of a single UAV to carry a considerable amount of sensors, the use of multiple robots to achieve a common task is also well studied in the literature. One of the many benefits of using multiple robots consists of reducing the time to perform the task, such as exploration. In these cases, the planning algorithm is optimized to reduce the time to collect information [10] choosing different waypoints [11], especially if the same kind of robot and sensors are

used. For example, to diversify information and to perform both autonomous search and rescue, in previous works, we presented a UAV and UGV team for subterranean exploration [12]. The cooperation of multiple robots increases the ability to perform a wider variety of tasks. Li et al. [13] use a UAV to provide a higher point of view to improve the environment map for the UGV navigation.

When teaming robots for exploration, it is most common to assume that each agent is capable of estimating their pose, such that the team is primarily used to improve the efficiency of environment mapping and exploration. This is the approach adopted in [14]. Similarly, Steude et al. [15] used a UAV, capable of self-localization, to improve and expand the map exploration for a ground robot and to optimize the mission time. However, having the UAVs in the team maintaining accurate self-localization presents challenges. For example, as discussed in Baldini F. et al. [16], using Visual-Inertial Odometry (VIO) solutions on UAVs is prone to solution drift, and the level of drift is heavily affected by the light condition in which the UAV operates. For these reasons, [16] used a learning approach to reduce this drift. Employing SLAM directly on a UAV can also reduce the drift as it has been demonstrated in [17]. However, some operating environments (e.g., a long corridor or tunnel), may affect the reliability of loop-closures, crucial for SLAM.

In the last few years, the Defence Advanced Research Projects Agency (DARPA) subterranean (SubT) challenge received attention due to the unique problem and challenging environment. This represents the closest research done and sponsored from a different government agency to the work presented in this dissertation. As mentioned in the previous paragraph, these environments characterized by long and featureless tunnels are challenging the main localization technique (SLAM). To mitigate this issue LiDAR-only based SLAM has been developed [18]. Agha et al. [19] presented an uncertainty-aware framework in the planning algorithm to perform reasoning and decision making in the belief space. The DARPA subT challenge also pushed to a decentralized multi-

agent robots for the environment exploration and Hudson et al. [20] developed a similar team (ground and aerial robot) to produce a decentralized SLAM solution on each platform.

To address some of these challenges, the approach presented in this work proposes to use active sensors on the UGV to localize the UAV in a manner that is not prone to drift and with modalities that will work in a variety of lighting conditions. With this UAV/UGV teaming configuration, the UGV has the advantage of additional load capacity and longer endurance for easily carrying many perception sensors along with the significant computational power needed to use them in near real-time. In this scenario, the UGV is used to map the environment as well as for estimating the state of the UAV within the map, while the UAV's mobility is leveraged to perform the search task.

However, when adopting this approach, due to the nature of the UGV's perception sensors' uncertainty as well as the potential for non over-lapping sensor fields-of-view, the localization uncertainty of the UAV is affected by the chosen flight path. As such, part of this dissertation on developing a path planning algorithm for the UAV that takes into consideration both the exploration needs and the localization uncertainty of the UAV.

Considering the position uncertainty in the planning algorithm consists of finding a path in the belief space (Section 2.6) such that motion increases information from sensing to reduce pose uncertainty of the robot with respect to the environment [21]. Planning under uncertainty lends to the use of a dual-layer architecture; where the first layer predicts all the possible outcomes and the second layer determines the best action to take [22]. For mobile robot path planning, Prentice et al. [23] presented a revised Probabilistic Road Map (PRM) [24], called Belief Road Map, where they fuse the predicted estimated position uncertainty into the planning process. In this work, the trajectory was designed with respect to the location of pre-deployed ranging radio beacons in order to reduce the position uncertainty while traversing from a start to a goal location. A

belief space planning problem is often formulated as a Partially Observable Markov Decision Process (POMDP) [25], which is characterized by a high computational cost. Agha-mohammadi et al. proposed a feedback-based information road map (FIRM) [26] that generalizes the PRM to take into account motion and sensing uncertainties. Similarly to [26] the approach presented in this dissertation, used an Extended Kalman Filter (EKF) to approximate the robot belief dynamics and propagate the belief space $b \equiv (\hat{x}^+, P)$. The same authors also leveraged the low computational FIRM algorithm to cope with a dynamically changing environment including the kidnapped robot problem [27]. Differently, this dissertation focuses more on robust localization subject to environment understanding. This choice has been made due to the fact that a subterranean search and rescue scenario is not often subject to dynamic changes.

1.2 Research Objectives

It is clear, that robot localization in an unknown environment is crucial for any task or achievement the robot needs to perform. Planning in the belief space helps, from a decision-making standpoint, to pick the path with more information gathering, but it is still subject to a certain degree of uncertainty due to sensor model errors, noisy measurements, and a non-perfect model of the UAV kinematics. For this reason, this research focuses on the following objectives:

- develop a full autonomous system composed by a UGV and UAV for underground exploration
- develop a simulator for path planning and localization testing before they are deployed on the real system
- develop a path planning algorithm that propagates the belief state of the UAV to incorporate

uncertainties in the decision making process

- develop a machine learning based planning algorithm that is capable to predict how the environment conditions affects the sensor measurements

At the beginning of the project, the focus was on building a belief space planning algorithm using mainly the UGV resources and sensors. Since this dissertation aims to investigates how the algorithm can be affected by sensor's models and environment condition, a later addition of a stereo camera, for VIO purposes is considered and described in Section 3.4. As VIO guarantees a more consistent and available UAV full pose measurement update for a more robust localization, it can also be more affected by the environment condition since it travels with the UAV. The introduction of VIO substantially improves the UAV localization and as in [28], where they used a variation of an extended Kalman filter characterized by a sliding window of poses and fused probabilistic constraints through new observations. On the other hand they did not conduct any study regarding how environment conditions could affect the VIO.

1.3 Literature Review

Extensive research in visually degraded environments (dark) has been conducted by Alexis et al. using a perception system that included a near–infrared stereo camera system, flashing LEDs, inertial sensors, and a 3D depth sensor to compute visual–inertial odometry and dense mapping [8]. The same authors also adopted a Long Wave Infrared thermal vision which is unaffected by darkness and can penetrate most obstructions such as smoke, dust, fog [29, 30]. They opted to use the full radiometric information instead of the rescaled thermal camera data to improve the UAV's VIO.

On the other hand, when the environment is a mix of dark, dusty, or lighted areas, the previous approaches might not result in the best choice. This dissertation investigates how machine learning can help a path planning algorithm to recognize areas that affect the UAV's localization and at the same time to better estimate the sensors and dynamic models. As it is easy to understand when a visual camera could be affected by dark or dusty areas, a Light Detection And Ranging (LiDAR) sensor depends more on the wavelengths at which it operates. Since dust particles are normally characterized by a larger dimension than the LiDAR wavelength they could heavily affect the sensor from capturing its surroundings [31].

For the mentioned reasons, it is important to consider how the belief space planning could be affected by the sensor models and the environment. Most of the studies concerning the belief space often take into consideration the uncertainties related to the observations provided by sensors. To make the planning algorithm more robust, it is important to understand how the sensor measurements are affected by the environment conditions. For this reason, the proposed approach consists of investigating how a supervised learning, such as the Gaussian Process (GP) regression, can learn the input-output mappings from the empirical data [32]. As GP regression is assumed to use noise-free input, McHutchon et al. proposed a GP framework that can be trained with input and output measurements corrupted by noise [33]. They affirmed that corrupted input measurements will affect more a process with a rapidly variable output [33]. Similarly, this work will use the UAV's position error to estimate its relationship with the UAV's position in space. As mentioned, the UAV model approximation does not consider all the non-linearities in its dynamics and a local Gaussian process regression could help in learning these variable as presented in [34]. A more specific example, which could affect the localization, concerns the UAV detection by the UGV sensors (when the UAV exits their field of view) resulting in faulty results from the navigation filter.

For instance, the ultra wide-band ranging radio is a good localization tool that drastically degrades its performance when the UAV is not in the Line of Sight (LOS). As Yang at al. showed, a sparse pseudo-input Gaussian process can mitigate the bias of non-line-of-sight condition [35].

Specifically, this GP-based algorithm does not use approximation of sensors' model, which can often fail and lead to estimation inconsistency, instead it relies on learning the error models directly from sensor measurements, including the occurrence of outliers as well as missed expected measurement updates.

This supervised learning technique is used to improve the path planning robustness, providing a prediction of the UAV's localization error along predetermined paths. Adding this information to the algorithm helps to decide the path the minimize the UAV's position error. This technique, based on training data, provides a more accurate sensor model and its degradation due to environment but also mitigates false sensors readings that could easily drive the algorithm decision. This is possible thanks to the ability of GP to estimate sensor models [36]. Different from a Belief Space Planning (BSP) approach, which lacks in modeling the stochastic dynamics of the sensor's observations [37], this approach offers the contribution of directly learning environmental factors including measurement outliers and missing measurements that are not accounted for in the BSP. A downside of this approach results in the necessity to provide enough training data in the environment such that the GP is capable to estimate the error model. For this reason, a contribution of this work consists of demonstrating of how a subset of more reliable sensors are uses to train the error models of the remaining sensors to expand the potential of this approach.

1.4 Dissertation Contributions

The main contributions of this dissertation are as follows:

Contribution 1: The design and development of several localization filters to provide relative localization between the specific set-up system of a UAV and a UGV that cooperatively share information in a centralized manner. Also, the filters where tested and tuned to provide a position estimate of the UAV for close loop feedback for the flight controller. The experimental assessment allowed the realization of the hardware stack (UAV and UGV) as the test-bed of the selected localization filter.

Contribution 2: The development of a high fidelity simulation environment that was used to design, develop and test planning algorithms.

Contribution 3: Design and implementation of a Belief Space Planning (Belief Space Planning (BSP)) algorithm to introduce uncertainty information during the decision making process of the UGV in providing a path to follow to the UAV.

Contribution 4: Design of a machine learning planning algorithm, based on the Gaussian Process regression problem, to create a robust decision in case of sensor failure or degradation due to the environment.

1.5 Dissertation Outline

This dissertation has been organized as follows: Chapter 2 presents a brief overview of the methods and basic tools that are used to develop the UAV planning algorithm; Chapter 3 describes the different versions of the localization filter that was developed for the UAV pose estimation with

field testing results; Chapter 5 presents the autonomous flight using the localization filter solution. Chapter 4 shows the Belief Space path planning algorithm with simulation results and Chapter 6 will present the new GP based path planning algorithm. The last chapter summarize and proposes new ideas to extend the work presented in this dissertation.

2

Background

This chapter presents some basic tools and methods that are used in this dissertation. Starting from simple UAV dynamic equations to a generic explanation of the Gaussian process and Kalman filters. It also describes the hardware used for field testing and the simulation environment developed for the algorithm testing.

2.1 Quadrotor Kinematics Equations

A quadrotor is generally considered an underactuated system, described with highly non-linear equations of motion. Most of the time it can be approximated using the motion of a six Degree of Freedom (DOF) rigid body that moves in space.
This work mainly focuses on localization and path planning, relying on the low-level control built inside the flight controller. For simplicity the Earth rotation and the Coriolis effect are neglected, therefore two coordinate systems are used to define the space where the Quadrotor will operate:

- $\mathbb{O}_b$, or Body Frame, describes the coordinate system centered in the center of mass of the Quadrotor (linked with the body).
- $\mathbb{O}_n$, or Navigation Frame, represents the absolute fixed coordinate system with the origin centered in an arbitrary place. This frame is oriented as Forward, Left and Up.

To describe the Quadrotor motion in space the kinematic equations take into account the velocity and position of the body. The state vector is defined as

$$x_{NED}(k) = [v_N(k), v_E(k), v_D(k), r_N(k), r_E(k), r_D(k)]^T \tag{2.1}$$

where the subscript stands for North East Down (NED), which is a common way to describe airframes in space. The linearized drone kinematics in the discrete time is shown in eq. 2.2 where ΔT

is the system's sampling time.

$$x_{NED} = \begin{bmatrix} v_N(k+1) \\ v_E(k+1) \\ v_D(k+1) \\ r_N(k+1) \\ r_E(k+1) \\ r_D(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \Delta T & 0 & 0 & 1 & 0 & 0 \\ 0 & \Delta T & 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta T & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_N(k) \\ v_E(k) \\ v_D(k) \\ r_N(k) \\ r_E(k) \\ r_D(k) \end{bmatrix} \tag{2.2}$$

To fully describe the state in the $\mathbb{O}_n$ coordinate system, a static rotation is required. Most important, to transform the state from $\mathbb{O}_b$ to $\mathbb{O}_n$ a transformation matrix, known as Direction Cosine Matrix (DCM), is needed

$$\mathbf{DCM} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\varphi\sin\psi + \sin\varphi\sin\theta\cos\psi & \sin\varphi\sin\psi + \cos\varphi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\varphi\cos\psi + \sin\varphi\sin\theta\sin\psi & -\sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi \\ -\sin\theta & \sin\varphi\cos\theta & \cos\varphi\cos\theta \end{bmatrix} \tag{2.3}$$

where φ,θ and ψ are the measured angle of roll pitch and yaw respectively. In this work, the DCM will often be referred to as $\mathbf{C}_b^n$ to determine the rotation from the body to the navigation frame.

2.2 Kalman Filter

One of the most used and well-known state observers is the Kalman Filter (KF) [38]. Being an optimal observer it is a very powerful tool able to estimate past, present and future states of a non-precisely known model [39]. Considering a linear discrete-time system subject to the differential

equation

$$x_{k+1} = A_k x_k + Bu_k + w_k \tag{2.4}$$

where w_k describes the process noise and $x \in \mathbb{R}^n$ is the state of the model, which is subject to a measurement $z \in \mathbb{R}^m$ of the form

$$z_k = H_k x_k + v_k \tag{2.5}$$

with v_k being the measurement noise. These noise variables, to guarantee optimality, are assumed to be white noises, independent and with normal probability distribution [39]

$$p(w) \sim \mathcal{N}(0, Q), \tag{2.6}$$

$$p(v) \sim \mathcal{N}(0, R). \tag{2.7}$$

The state estimation process uses two fundamental steps, a *prediction* where the filter propagates the current state and the error covariance in the next time-step creating a so called *a priori* estimate, and a *measurement update* where information from a measurement is used to refine the prediction, having the *a posteriori* estimate. The first step or prediction is described by the following equation, where the state is propagated to the next step using the only knowledge of the system's model

$$\begin{cases} \hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + Bu_k + w_k \\ \hat{y}_k = H_k \hat{x}_{k|k} \end{cases} \tag{2.8}$$

$$P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k \tag{2.9}$$

The second step uses the measurement update to re-evaluate the a priori estimate and to produce the a posteriori and more precise state estimate.

$$K_k = P_{k+1|k} H_k^T (H_k P_{k+1|k} H_k^T + R_k)^{-1} \tag{2.10}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k(z_k - \hat{y}_k) \tag{2.11}$$

$$P_{k+1|k+1} = (I - K_k H_k) P_{k+1|k} \tag{2.12}$$

During the a posteriori step, the K_k represent the optimal Kalman Gain, the $\hat{x}_{k+1|k+1}$ is the a posteriori estimation and $P_{k+1|k+1}$ is the updated error covariance.

2.3 Extended Kalman Filter

A vast majority of the dynamic systems are described by non-linear differential equations and to achieve state estimation the Kalman filter needs to be modified to guarantee similar results than with linear systems. The EKF is a Kalman filter that linearizes around the current mean and covariance [39]. A non-linear system is described as

$$\begin{cases} x_{k+1} = f(x_k, u_k, w_k) \\ y_k = h(x_k, v_k) \end{cases} \tag{2.13}$$

and a measurement $z_k \in \mathbb{R}^m$.

As shown in the previous section the EKF, similarly to the KF, consists in two steps. The first, or

prediction

$$\begin{cases} \hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \\ \hat{y}_k = h(\hat{x}_{k|k}) \end{cases} \tag{2.14}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \tag{2.15}$$

where F_k being the linearization around the equilibrium point of the system 2.13 described as

$$F_k = \left. \frac{\partial f}{\partial x} \right|_{\hat{x}_{k|k}, u_k} \tag{2.16}$$

The second step, or update is similar to the KF

$$K_k = P_{k+1|k} H_k^T (H_k P_{k+1|k} H_k^T + R_k)^{-1} \tag{2.17}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k(z_k - \hat{y}_k) \tag{2.18}$$

$$P_{k+1|k+1} = (I - K_k H_k) P_{k+1|k} \tag{2.19}$$

where H_k is the linearization of the system's output function

$$H_k = \left. \frac{\partial h}{\partial x} \right|_{\hat{x}_{k|k}} \tag{2.20}$$

The extended Kalman filter will be extensively used int the following chapters for the UAV position localization.

2.4 Real World Scenario System Set-Up

As previously mentioned, the UGV is entitled to carry most of the heavy sensors and computing power therefore, the drive chassis Husky by Clearpath Robotics was chosen [40]. The main sensors, carried by the UGV for cooperative localization, consist of a Velodyne VLS-128 channels 3D LiDAR with a Field of View (FOV) of -25 +15 degree and a range of 300 meters. The 3D LiDAR, which is needed to provide a map of the environment and for SLAM purposes, is tilted 15 degrees up to increase the UAV detection zone. To measure the UGV-UAV distance an Ultra Wide-Band (UWB) (DWM-1001) ranging radio is used, which provides a measurement at 10 Hz. Finally, a FLIR camera with a fisheye lens is mounted upward to detect the UAV. The UGV, equipped with heavier sensors and higher capacity batteries, has the task of fusing the sensor data, including the one sent by the UAV, and provide the drone pose along with the planning algorithm. Figure 2.4.1 shows the Husky chassis with a list of the onboard sensors.

The UAV is built using a custom frame and it is equipped with the Holybro PixHawk4 (PX4) flight controller and a modified version of the PX4 Firmware [41] that supported additional sensors such as a higher quality IMU and a laser altimeter. This open-source firmware allows a wide range of implementation and testing. The flight controller is linked with an Nvidia Jetson TX1 as a companion computer, which has the task of running the Ubuntu 16.04 operating system and the Robot Operating System (ROS) [42] to send and receive data from the UGV through the Mavlink protocol. The UAV's IMU has been improved with the addition of an Analog Devices ADIS 16495-2 IMU. The UAV is also capable of a reliable altitude measurement thanks to a LidarLite laser altimeter mounted on its bottom. To conclude an Intel RealSense T265 depth camera is positioned in front

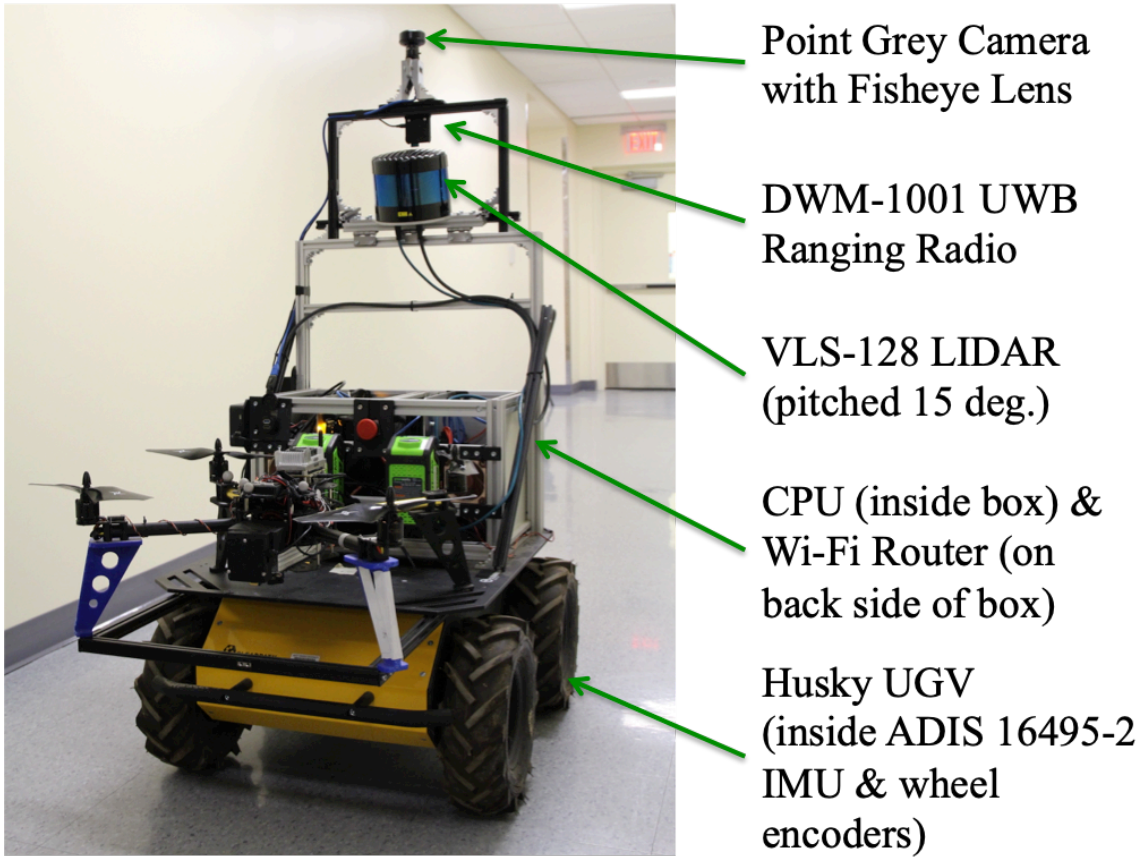


Figure 2.4.1: Description of the UGV onboard sensors

of the UAV for VIO purposes as shown in Figure 2.4.2. Developing and assembling the hardware always results a slow process. The many components and factors that need to work together make this process challenging. A key factor for the team of robots consisted of developing the network and communication protocol that guaranteed the exchange of information between the two agents. Also, when dealing with a multi-agent setup, it is important to make sure that all the machine are time synchronized. Regarding the assembling of a custom UAV, the weight and the balance of the system heavily impacts the flight performance.

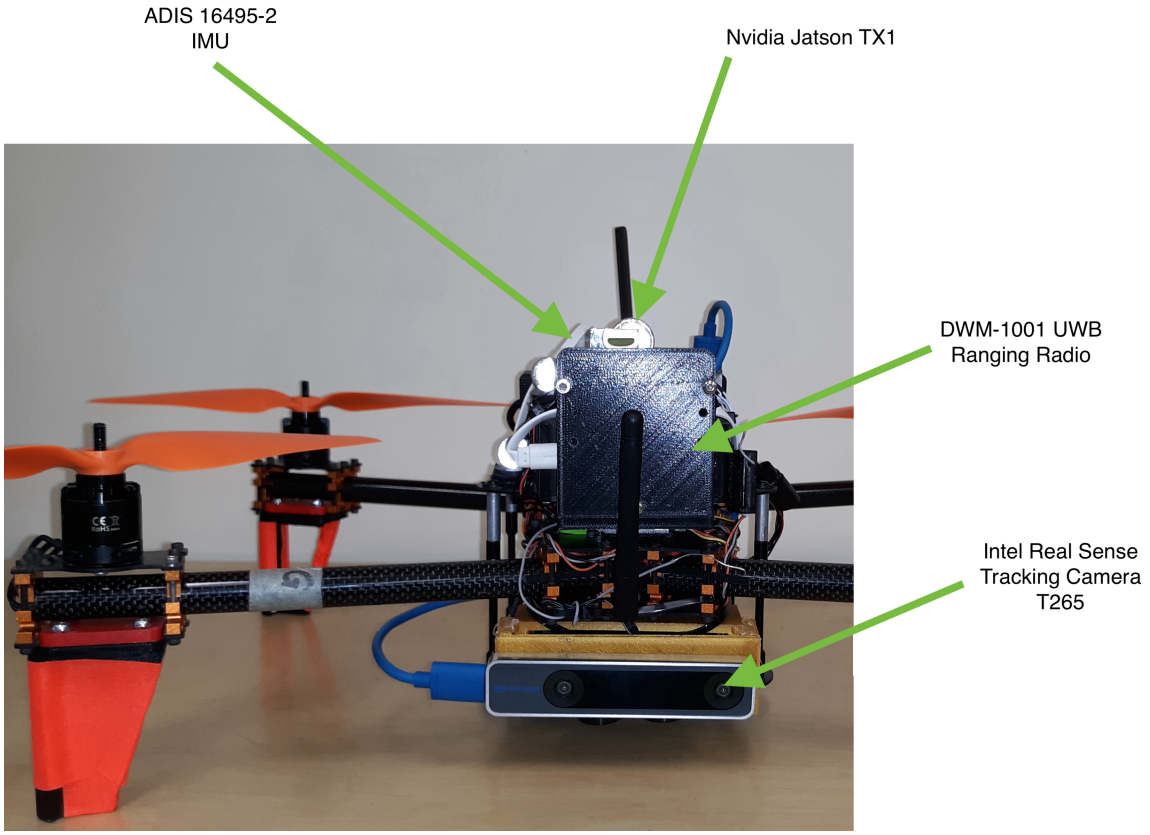


Figure 2.4.2: Description of the UAV's onboard sensors

2.5 Simulation Scenario System Set-Up

Experimental setup for research purposes quite often requires to use of new and not certified algorithms or actions that do not always behave how expected. To reduce failures a simulation environment is the best option to perform preliminary tests. Also, the flexibility of a simulated environment, not only reduces risks and costs but allows changes in the scenario (e.g. light conditions, visibility, obstacles, etc.) in a very easy manner. Using the PX4 stack [41] and ROS [42] in both the real world scenario and the simulator, allows an easy transition of any software developed from the

simulator to the hardware. This is possible thanks to the PX4 Firmware which includes a Software in the Loop (SITL) version that "mirrors" the real flight controller.

The simulated scenario is developed in Gazebo [43] and it similarly replicates the real test envi-

Figure 2.5.1: Simulation environment developed in the Gazebo software.

ronment where the team UGV-UAV will be deployed, shown in Figure 2.5.1. Following a list of the main packages used in the simulator:

- **MAVROS:** translate the Mavlink communication protocol in ROS language [44]
- **PX4 Firmware:** simulate a quadrotor using the same firmware on board of the real flight controller
- **Velodyne simulator:** a package that simulates the Velodyne Lidar products in Gazebo and

ROS giving access to their point cloud

- **Husky simulator:** allows to simulate the Husky stack in a Gazebo world
- **Octomap:** creates an occupancy map, of the environment using voxels [45].
- **VIO package:** performs visual inertial odometry using a sequence of stereo images.

These "off the shelf" packages have been modified to match the research needs, specifically the Velodyne simulator was updated to provide 128 channels and it was connected to the husky simulator to resemble the hardware configuration. Specific packages, such us the graph generator that provides the possible paths were also developed for planning algorithm and it will be presented in the next chapters.

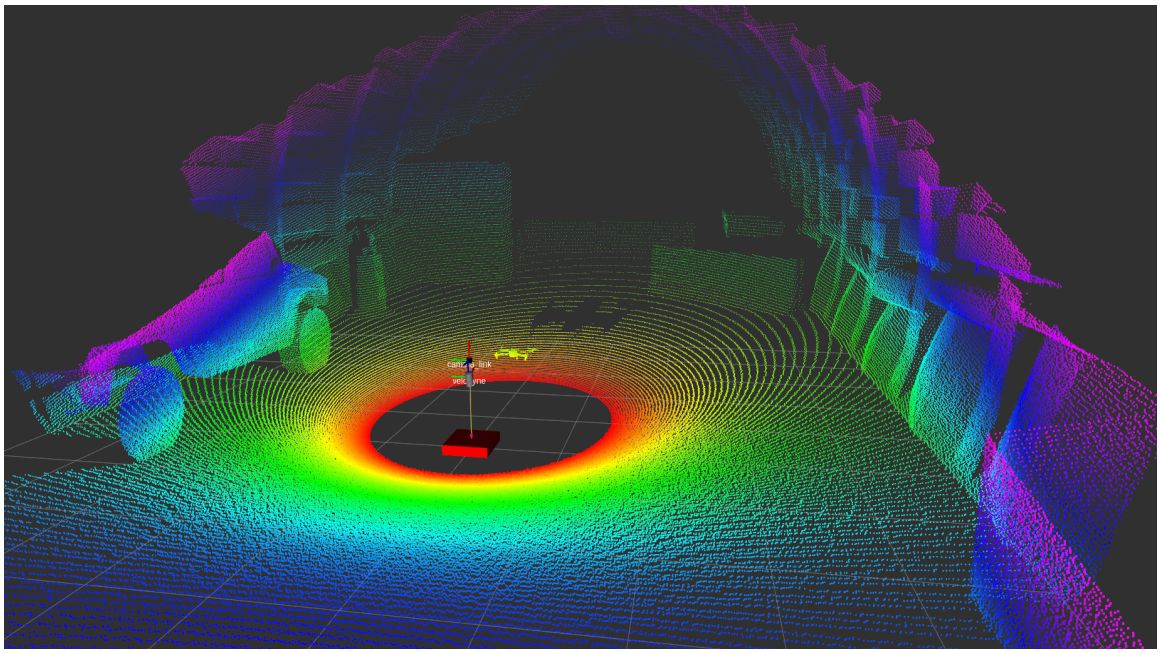


Figure 2.5.2: Point cloud visualization provided by the Velodyne simulator

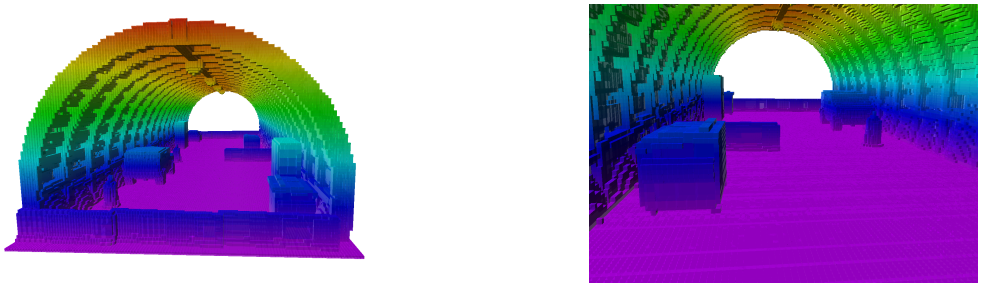


Figure 2.5.3: Octomap of the environment with the 3D voxels for the occupied space

Figure 2.5.2 represents the point cloud generated by the Velodyne LiDAR, which is the input to the ROS Octomap [45] package. This generates a 3D occupancy volume (i.e., an octree) of the environment as shown in Figure 2.5.3. To better represent how the simulation and the software share the data the diagram in Figure 2.5.4 show how each part of the simulation provides and accesses data.

2.6 Belief Space

In this section, a general idea of belief space is described. To allow a robot to navigate autonomously, it has to deal with uncertainties coming from an unknown or changing environment, unmodeled dynamics, unpredicted external forces, and especially noisy sensor measurements which are the key tools to understand the surroundings [46]. Since information gathering for an autonomous robot is crucial for its autonomy, the act of considering uncertainties during the planning process means to reduce, at best, the loss of that information exploring the state space where it is maximized [46]. For the robot to optimize its information loss, it has to estimate its state over a set of possible states, called belief states, that lies inside the Belief Space [46]. The belief space considers noisy

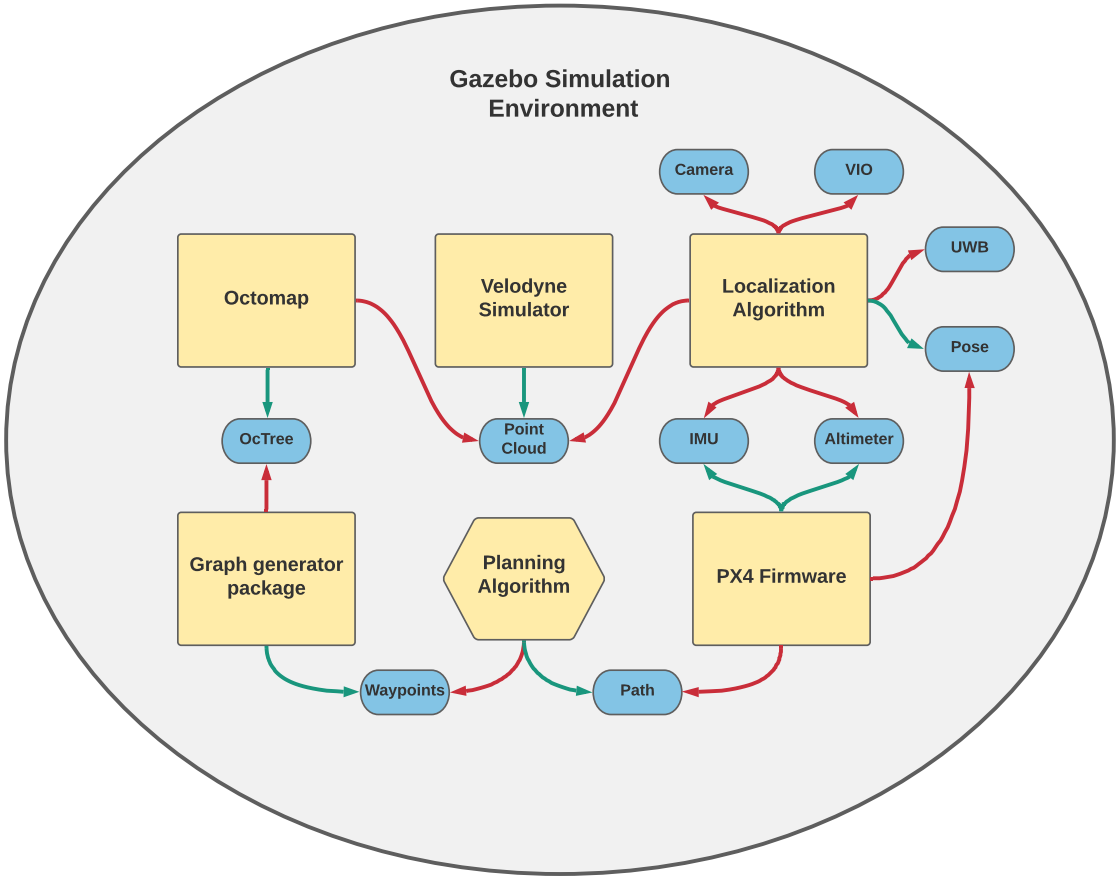


Figure 2.5.4: Diagram representation of the information flow within the simulation. Each block provides data through the green arrows and access information from the red arrows.

observations of the non-linear system at the time t of the form

$$z_t = g(x_t) + \omega \tag{2.21}$$

where x_t is the system state and ω is the zero-mean Gaussian noise with state covariance W_t.

The belief state results as the Probability Density Function (PDF) over the robot's state $P(x)$ and

it is generally updated using Bayesian filtering as

$$P(x_{t+1}) = P(z_{t+1}|x_{t+1}) \int_x P(x_{t+1}|x, u_t)P(x)dx. \tag{2.22}$$

For Gaussian belief state [37] an 'EKF is used for the belief update with the covariance matrix Σ_t update as

$$\Sigma_{t+1} = \Gamma - \Gamma C_t^T (C_t \Gamma C_t^T + W_t)^{-1} C_t \Gamma \tag{2.23}$$

where $\Gamma_t = A_t \Sigma_t A_t^T$ and A_t, C_t being the Jacobian matrices of the system dynamics linearized around the mean $\bar{x}_t$ updated as

$$\bar{x}_{t+1} = f(\bar{x}_t, u_t) + \Gamma_t C_t^T (C_t \Gamma C_t^T + W_t)^{-1} (\hat{z}_t - z_t) \tag{2.24}$$

An advantage of this approach consists of better options for decision making and planning, while the disadvantage concerns the computation complexity of the algorithm.

2.7 Gaussian Process Regression Problem

For this research work, a supervised learning regression problem will be formulated. Differently from a classification problem, a regression concerns the prediction of continuous quantities [32]. A Gaussian Process can be described as a distribution over functions [32].

Definition 1 *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [32].*

A GP is fully described by a mean and a covariance function.

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] \tag{2.25}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right] \tag{2.26}$$

where $m(\mathbf{x})$ represents the mean function and $k(\mathbf{x}, \mathbf{x}')$ the covariance function of a real process $f(\mathbf{x})$ [32]. The Gaussian Process is described as

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{2.27}$$

A training set $\mathcal{D}$ of $\boldsymbol{n}$ observations is defined as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, \ldots, n\}$, where $\mathbf{x}$ is the input vector and $\mathbf{y}$ denotes a scalar output. In the regression problem the outputs are real values and the goal is to make inference about the input-output relationship given the set $\mathcal{D}$.
Definition 1 implies that for any given set of inputs $\{x_1, \ldots, x_n\}$, the corresponding random variables $\{f(x_1), \ldots, f(x_n)\}$ have a n-dimensional normal distribution

$$p(f(x_1), \ldots, f(x_n)|x_1, \ldots, x_n) = \mathcal{N}(m, k) \tag{2.28}$$

where m and k are expressed in eq. 2.25 and in eq. 2.26 respectively [47]. More details about Gaussian Process regression can be found in [32].

2.8 Concluding Remarks

This chapter introduced some of the tools used to develop the localization filter and the planning algorithm presented in the next chapters. This general introduction is the foundation to better understand the process and the ideas involved in this dissertation.

3

UAV-UGV Coopertative Localization

This chapter is based on the cooperative localization algorithm described in “Field-Testing of a UAV-UGV Team for GNSS-Denied Navigation in Subterranean Environments” [12] which will be addressed as of version 1.0. Also two more modifications of it will be presented in this chapter.

3.1 Chapter Motivation

Robot localization is fundamental for its autonomy. A robot needs to know its location with respect to its surroundings to be able to take actions to reach its goal, accomplish tasks, avoid obstacles, etc. As pointed in chapter 1 when dealing with challenging environments like an underground tunnel, a GNSS-denied approach need to be taken into consideration. For this reason, the team UGV-UAV described in Section 2.4 cooperates to determine the UAV position with respect to the still UGV, and actuate the planning algorithm presented in the next chapter.

3.2 Localization Algorithm Version 1.0

3.2.1 Filter description

The main task of the UGV is to produce a preliminary map of the environment and to contribute to the drone localization once it is deployed. For simplicity, this first step is assumed to be already performed and the UAV has a full environment map available for autonomous flight. In this first step, where the environment is fully explored, the UAV contribution consists in offering an "eye in the sky" prospective for search and rescue purposes [12]. This localization filter design of used for estimating the UAV's position, relative to the UGV, is available for the entire desired flight with the goal of using it as position feedback for the flight controller. Therefore, to ensure a reliable localization, a navigation planner is required and it needs to take into account sensor uncertainties and sensor fields of view. Figure 3.2.1 shows an overview of the GNSS-denied navigation approach adopted.

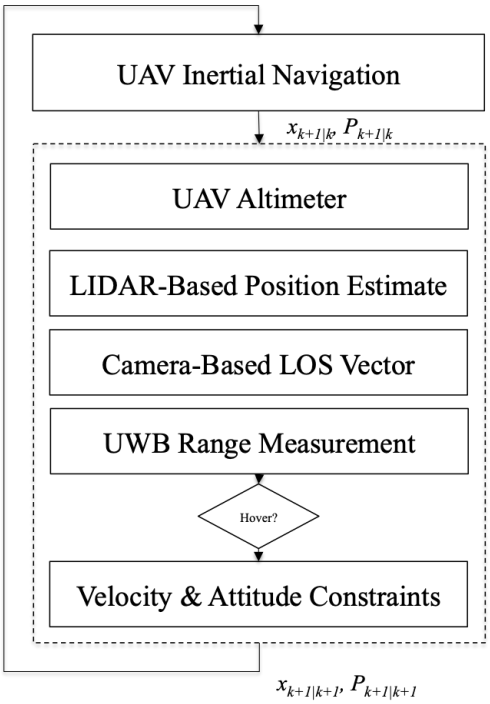


Figure 3.2.1: Localization filter logic diagram

Error-state Extended Kalman Filter

In this first version, all the sensor measurements are fused using an error-state EKF. Among the six sensors, the UAV's onboard inertial navigation is used to predict the navigation state with a 50 Hz update rate. The other five sensors, which are asynchronous, provide updates when they are available and at different rates. For instance, the LiDAR and the upward-facing camera's field of views are almost complementary with a small intersection between the two, and for this reason the two sensors do not always provide a measurement at the same time. As such, the EKF must be able to process any sub-set of measurement updates as they are available [12]. The UAV state vector consists of its attitude, position, and velocity in the navigation frame (NED), whose origin

is centered at the UGV and it is described in eq. 3.1.

$$\hat{x} = [\varphi_b^n, \theta_b^n, \psi_b^n, v_N, v_E, v_D, r_N, r_E, r_D]^T \tag{3.1}$$

However, because the inertial navigation is used as the navigation model the error-state EKF needs to estimate also the small deviations (δ) of the UAV IMU sensor accelerometer biases (b_a) and rate gyroscope biases (b_g), as listed in eq. 3.2,

$$\delta\hat{x} = [\gamma(1)_b^n, \gamma(2)_b^n, \gamma(3)_b^n, \delta v_N, \delta v_E, \delta v_D, \delta r_N, \delta r_E, \delta r_D, b_{ax}, b_{ay}, b_{az}, b_{gx}, b_{gy}, b_{gz}]^T \tag{3.2}$$

where γ_b^n represents the small-angle attitude error. The sensors that are not constrained by FOV and "always" provides updates according to their frequencies are

- UWB ranging measurement between the UAV and UGV at 10 Hz
- laser altimeter onboard the UAV at 10 Hz

and the ones that depend on the UAV being in the respective field of view

- LiDAR measurement at 5 Hz
- Fish-eye camera at 10 Hz

One more constraint on the UAV motion is added to the EKF when it is believed to be hovering triggering a zero-velocity update.

Details of the EKF

The inertial navigation is based upon the error-state formulation within the North-East-Down navigation frame that is outlined in Groves [48], but for simplicity the contributions of the Earth's

rotation and craft-rate terms within the Inertial Navigation Systems (INS) mechanization were ignored. Therefore, the attitude update is given as

$$\hat{C}^n_{b,k+1|k} = \hat{C}^n_{b,k|k}(I_{3\times 3} + \Omega^b_i \delta t) + w_{3\times 1,att} \tag{3.3}$$

where C^b_n is the rotation matrix between the navigation frame and body frame, Ω^b_i is the skew-symmetric matrix with terms defined by the IMU rate gyros (p - roll rate, q -pitch rate, and r - yaw rate) as shown in Eq. 3.4 [48], $w_{3\times 1,att}$ is attitude integration process noise, and δt is the IMU sampling rate.

$$\Omega^b_i = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{3.4}$$

As previously mentioned in 2.1 the Coriolis terms were neglected and the velocity update transforms the IMU's specific force measurements from the body-frame to the navigation frame, accounts for the acceleration due to gravity, and integrates over time, as shown in Eq. 3.5,

$$\begin{bmatrix} \hat{v}_N \\ \hat{v}_E \\ \hat{v}_D \end{bmatrix}_{k+1|k} = \begin{bmatrix} \hat{v}_N \\ \hat{v}_E \\ \hat{v}_D \end{bmatrix}_{k|k} + \left[\hat{C}^n_{b,k+1|k} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right] \delta t + w_{3\times 1,vel} \tag{3.5}$$

where $w_{3\times1,vel}$ is the velocity integration process noise. The position estimates are updated by simply integrating the velocity estimates using trapezoidal integration [12],

$$\begin{bmatrix} \hat{r}_N \\ \hat{r}_E \\ \hat{r}_D \end{bmatrix}_{k+1|k} = \begin{bmatrix} \hat{r}_N \\ \hat{r}_E \\ \hat{r}_D \end{bmatrix}_{k|k} + \left[\begin{bmatrix} \hat{v}_N \\ \hat{v}_E \\ \hat{v}_D \end{bmatrix}_{k|k} + \begin{bmatrix} \hat{v}_N \\ \hat{v}_E \\ \hat{v}_D \end{bmatrix}_{k+1|k} \right] \frac{\delta t}{2} + w_{3\times1,pos} \tag{3.6}$$

where $w_{3\times1,pos}$ is the assumed position integration process noise. Finally, all of the IMU sensor biases, are updated assuming random-walk as in equation 3.7.

$$b_{k+1|k} = b_{k|k} + w_{3\times1,bias} \tag{3.7}$$

The linearized error-state dynamics are given by Eq. 3.8,

$$F_{INS} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \hat{C}^n_{bk+1|k} \\ 0_{3\times3} & \left[\wedge - \hat{C}^n_b \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \right] & 0_{3\times3} & \hat{C}^n_{bk+1|k} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{3.8}$$

where $\hat{}$ indicates the skew-symmetric matrix of a vector. The error-state transition matrix, expanded to first order, is given in Eq. 3.9.

$$\Phi_{INS} = I_{15\times15} + F_{INS}\delta t \tag{3.9}$$

The laser altimeter transmits the altitude measurement to the UGV through UDP protocol and it is assumed to measure with respect to a flat ground. Because the UAV is meant to slowly patrol the environment, small angles of pitch and roll are also assumed, therefore this measurement, taking into consideration the UAV's orientation, is shown in Eq. 3.21 whose observation model is given in Eq. 3.11.

$$\hat{z}_{Alt} = \frac{-\hat{r_D}}{cos(\theta)cos(\varphi)} + v_{alt} \tag{3.10}$$

$$H_{alt} = \begin{bmatrix} 0_{1\times8} & -\frac{1}{cos(\theta)cos(\varphi)} & 0_{1\times6} \end{bmatrix} \tag{3.11}$$

The measurement error-covariance of the altimeter measurement was assumed to be $v_{alt} \sim \mathcal{N}(0, 10cm)$, given the an error of $\pm 10cm$ [12].

The 3D LiDAR produces a point cloud that represents all the surfaces hit by a beam of light, and each point corresponds to the position in space, with respect to the sensor, of the surface's part hit. With this same process the relative position of the UAV can be determined by the cluster of points that reflects its frame. To determine this cluster a technique developed by Bogoslavsyi and Stanchness [49, 50] is adopted, and when the UAV is found its position corresponds to the

cluster's centroid. This is used as a measurement updated in the error-state EKF (Eq. 3.35) with the linearized observation sensitivity matrix as given in Eq. 3.13.

$$\hat{z}_{LiDAR} = \begin{bmatrix} \hat{r_N} \\ \hat{r_E} \\ \hat{r_D} \end{bmatrix} + v_{3\times1,LiDAR} \tag{3.12}$$

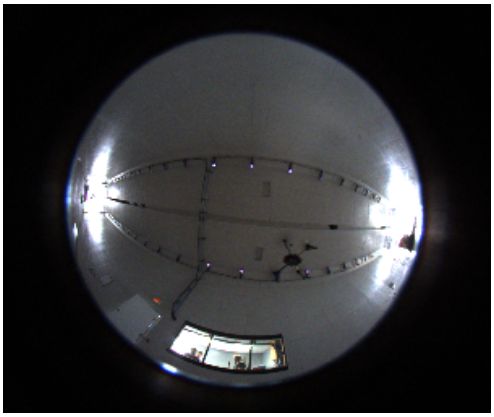
$$H_{LiDAR} = \begin{bmatrix} 0_{3\times6} & -I_{3\times3} & 0_{3\times6} \end{bmatrix} \tag{3.13}$$

The assumed measurement error-covariance for the LiDAR position updates is $v_{LiDAR} \sim \mathcal{N}(0_{3\times3}, I_{3\times3}10cm)$ [12].

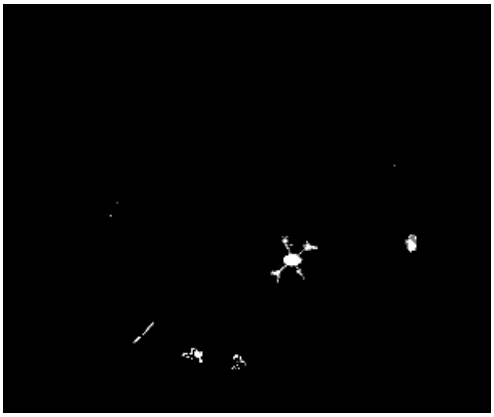
The fish-eye camera tracks the UAV using the OpenCV library [51, 52] and the center of the bounding box is considered to be the UAV's position estimate in the image pixel frame as shown in Figure 3.2.2.

From the pixel frame, the UAV location is transform in a LOS unit vector according to the Scaramuzza fish-eye camera calibration model [53]. The error-state EKF is then updated with the unit vector observation model described in Eq. 3.14,

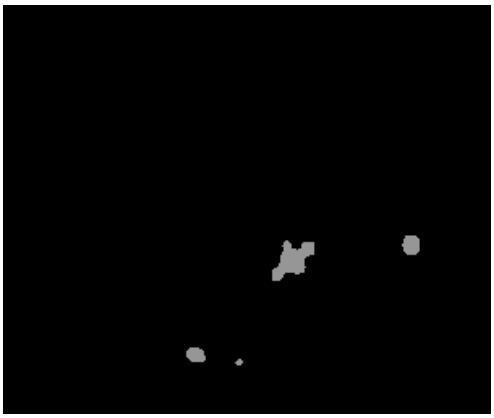
$$\hat{z}_{cam} = \begin{bmatrix} \frac{\hat{r_N}}{\hat{\rho}^{UAV}_{UGV}} \\ \frac{\hat{r_E}}{\hat{\rho}^{UAV}_{UGV}} \\ \frac{\hat{r_D}}{\hat{\rho}^{UAV}_{UGV}} \end{bmatrix} + v_{3\times1,cam} \tag{3.14}$$



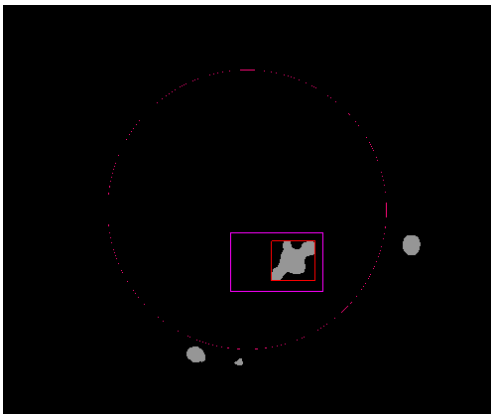
(a) Raw camera image



(b) Image after background subtraction.



(c) Image after binary threshold is applied.



(d) Image with perceived UAV (red box), UAV prediction (purple box), and edge segmentation (red ring).

Figure 3.2.2: Sample of the UAV camera tracking process.

where $\hat{\rho}_{UGV}^{UAV} = \sqrt{\hat{r_N}^2 + \hat{r_E}^2 + \hat{r_D}}$ is the estimated range of the UAV from the frame origin centered

at the UGV [12]. The LOS vector's observation sensitivity matrix, H_{cam}, is given by Eq. 3.15.

$$H_{cam} = \begin{bmatrix} 0_{1\times6} & \frac{-(r_E^2+r_D^2)}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{r_N r_E}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{r_N r_D}{(\hat{\rho}_{UGV}^{UAV})^3} & 0_{1\times6} \\ 0_{1\times6} & \frac{(r_N r_E)}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{-(r_N^2+r_D^2)}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{r_E r_D}{(\hat{\rho}_{UGV}^{UAV})^3} & 0_{1\times6} \\ 0_{1\times6} & \frac{(r_D r_N)}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{r_D r_E}{(\hat{\rho}_{UGV}^{UAV})^3} & \frac{-(r_E^2+r_N^2)}{(\hat{\rho}_{UGV}^{UAV})^3} & 0_{1\times6} \end{bmatrix} \tag{3.15}$$

The measurement error-covariance for the LOS unit vector reported by the camera tracking algorithm was $v_{cam} \sim \mathcal{N}(0_{3\times3}, I_{3\times3}0.1)$ [12].

The distance between the UGV and the UAV is available through the UWB. This ranging measurement it is also used in the LiDAR cluster algorithm to reduce the possible candidate cluster for the UAV localization. Being the range measurement also an observation for the error-state EKF, it is modeled as in Eq. 3.25,

$$\hat{z}_{UWB} = \hat{\rho}_{UGV}^{UAV} + v_{uwb} \tag{3.16}$$

and the observation sensitivity matrix is given as in Eq. 3.17.

$$H_{UWB} = \begin{bmatrix} 0_{1\times6} & -\frac{1}{\hat{\rho}_{UGV}^{UAV}} & -\frac{1}{\hat{\rho}_{UGV}^{UAV}} & -\frac{1}{\hat{\rho}_{UGV}^{UAV}} & 0_{1\times6} \end{bmatrix} \tag{3.17}$$

The measurement error-covariance of the UWB measurement was assumed to be $v_{uwb} \sim \mathcal{N}(0, 10cm)$.

3.2.2 Version 1.0 results

The localization algorithm described in section 3.2 has been experimentally tested off-line while the UAV was remotely piloted. This test was conducted inside a wind tunnel facility approximately

5-m x 5-m square and a length of 36 meters, with a motion capture system (VICON) mounted close to the ceiling that was used as ground truth to compare the results. A visual example of the LiDAR clustering result compared with the full error-state EKF output is shown in Figure 3.2.3.

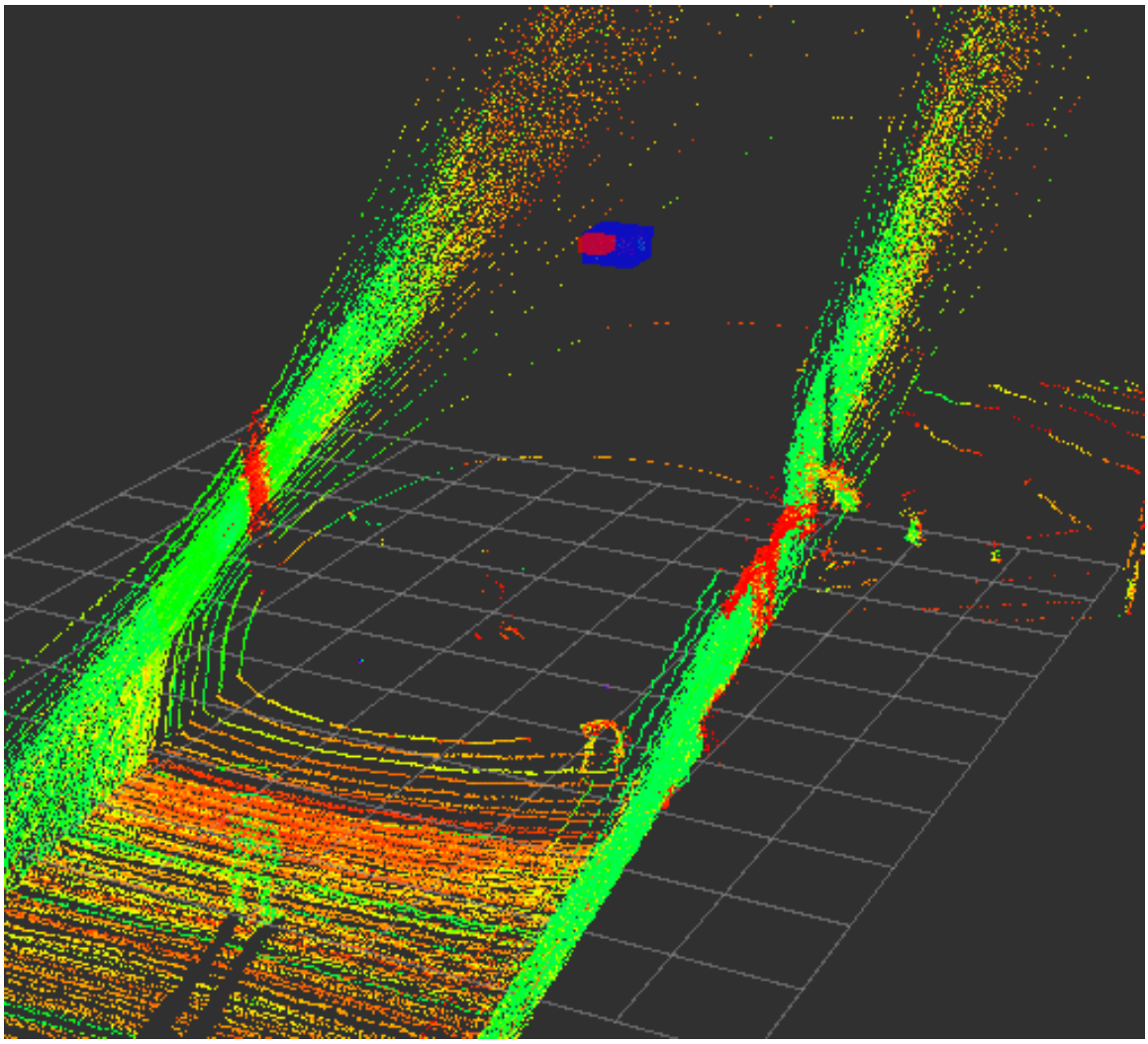


Figure 3.2.3: Example UAV position estimate at a single epoch (blue bounding box, LiDAR estimate, red bounding box EKF estimate) shown alongside the LiDAR pointcloud in a wind-tunnel facility of dimensions 5 m wide, 5 meter height and 36 m long.

Multiple flights have been conducted and Figure 3.2.4 shows the navigation filter performances of Flight 2 compared with the ground truth solution.

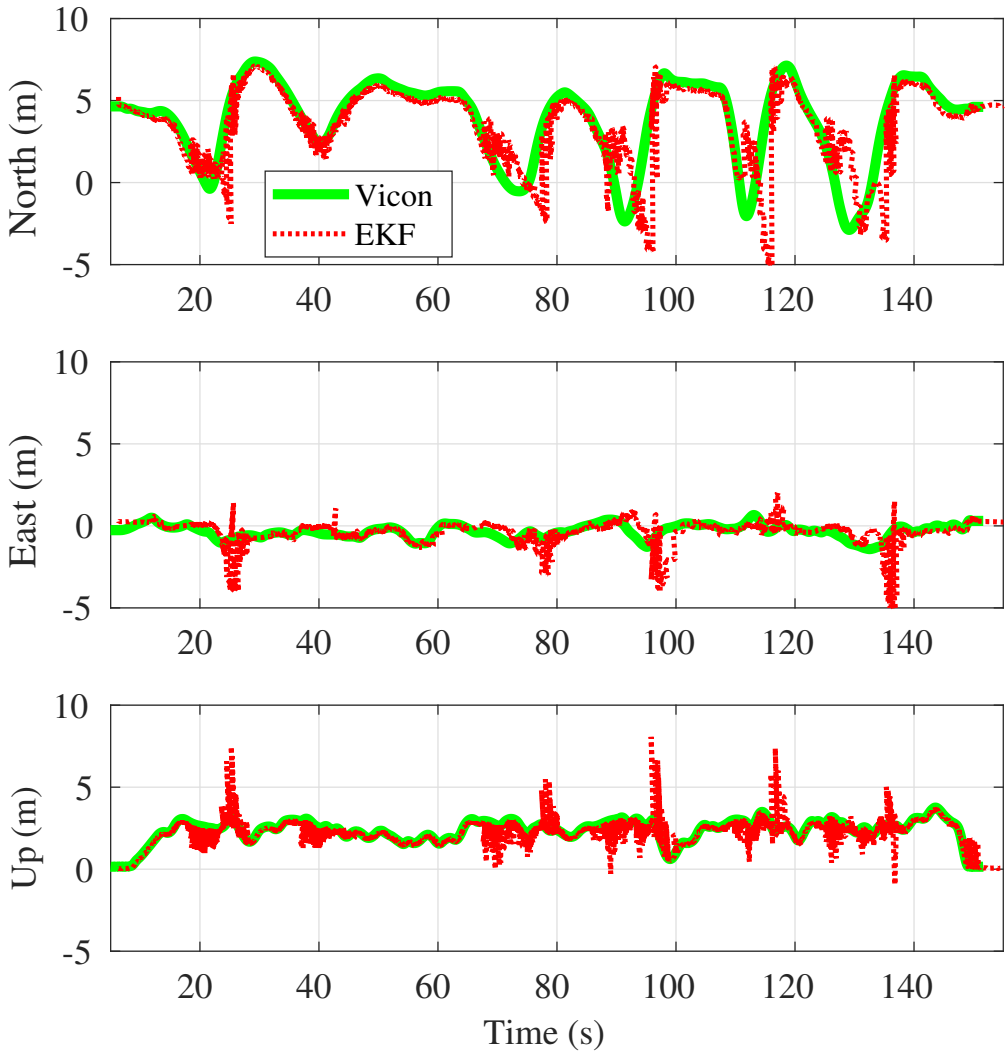


Figure 3.2.4: Example UAV position estimation performance against VICON reference solution for Flight 2.

The position error was estimated for each flight between the localization filter results and the ground truth according to the Eq. 3.18,

$$error_{3D} = \sqrt{(\hat{r}_E - r_{E,Vicon})^2 + (\hat{r}_N - r_{N,Vicon})^2 + (\hat{r}_D - r_{D,Vicon})^2} \tag{3.18}$$

Table 3.2.1 shows the navigation filter performance for each flight which have similar duration and

Table 3.2.1: EKF 3D positioning error statistics for the 3 flight tests.

	Duration (s)	RMS (m)	μ (m)	σ (m)	Median (m)	Max. (m)	# LiDAR Upd.	# Camera Upd.
Flight 1	158.65	2.60	1.59	2.05	0.91	15.83	250	144
Flight 2	139.86	2.16	1.37	1.67	0.65	10.75	251	201
Flight 3	159.9	2.35	1.20	2.02	0.61	17.77	122	182

kind of maneuvers. Given the size of the testing facility, the results obtained with this first version are too large to be considered as a position feedback solution for the flight controller.

3.3 Localization Algorithm Version 2.0

3.3.1 Filter description

The first version of the localization filter laid the basis for a more reliable and accurate version that is referred to as version 2.0. As the structure of the EKF is mainly unchanged from version 1.0, the UAV state vector is reduced to the drone kinematic. The state vector consists of the UAV velocity and position in a local North-East-Down, navigation frame, whose origin is centered at the UGV

position.

$$\hat{x} = [v_N, v_E, v_D, r_N, r_E, r_D]^T \tag{3.19}$$

The error-state of the Extended Kalman Filter, estimates small perturbations (δ) from the unknown true state vector (3.19) as listed in equation (3.20),

$$\hat{\delta x} = [\delta v_N, \delta v_E, \delta v_D, \delta r_N, \delta r_E, \delta r_D]^T \tag{3.20}$$

For the altimeter, an absolute distance to the ground is measured from a sensor that is mounted in a fixed orientation on the UAV's body. This yields a measurement model that depends upon the UAV's altitude.

$$\hat{z}_{alt} = \frac{-\hat{r_D}}{cos(\theta)cos(\varphi)} + v_{alt}, \tag{3.21}$$

For this measurement, the covariance matrix is assumed to be $R_{alt} = \sigma^2_{alt} = 0.01\ [m^2]$ and the Jacobian of the observation model is given as:

$$H_{alt} = \begin{bmatrix} 0_{1,5} & -\frac{1}{cos(\theta)cos(\varphi)}. \end{bmatrix} \tag{3.22}$$

In equation (3.22) the UAV's pitch φ and roll θ are estimated from the UAV's IMU during the belief-state propagation process. The resulting altimeter error covariance matrix update $P_{k|k}$ is given as

$$P_{k|k} = (I_6 - KH_{alt})P^{-1}_{k|k-1}\left(I_6 - KH_{alt}\right)^T + KR_{alt}K^T \tag{3.23}$$

where $K = \left(P_{k|k-1}H\right)^T \left(R_{alt} + H_{alt}P_{k|k-1}H_{alt}^T\right)^{-1}$ is the Kalman gain and I_6 is the identity matrix with the subscript dimension. Consequently the error-state update is

$$\hat{\delta x}_{k|k} = \hat{\delta x}_{k|k-1} + K(\hat{z}_{alt} - \hat{r}_D). \tag{3.24}$$

The UWB ranging radio measurement is modeled as

$$\hat{z}_{UWB} = \hat{d} + v_{uwb} \tag{3.25}$$

with the following observation model

$$H_{uwb} = \begin{bmatrix} 0_{1,3} & -\frac{\hat{r}_{Nk|k-1}}{\hat{d}} & -\frac{\hat{r}_{Ek|k-1}}{\hat{d}} & -\frac{\hat{r}_{Dk|k-1}}{\hat{d}} \end{bmatrix} \tag{3.26}$$

where $\hat{d}$ is the estimated distance between the UAV and UGV.

$$\hat{d} = \left\| \begin{bmatrix} \hat{r}_{Nk|k-1} & \hat{r}_{Ek|k-1} & \hat{r}_{Dk|k-1} \end{bmatrix} \right\|_2 \tag{3.27}$$

This yields the following error-covariance update.

$$P_{k|k} = (I_6 - KH_{uwb})P_{k|k-1}^{-1}\left(I_6 - KH_{uwb}\right)^T + KR_{uwb}K^T \tag{3.28}$$

For the state transition model, under the assumptions of constant velocity dynamics, the state transition matrix Φ is modeled, using simple kinematics, as

$$\Phi = \begin{bmatrix} \mathbf{I}_3 & 0_{3,3} \\ \mathbf{I}_3 \cdot T_s & \mathbf{I}_3 \end{bmatrix} \tag{3.29}$$

and the error covariance matrix update as

$$P_{k|k} = \Phi P_{k|k-1} \Phi^T + Q \tag{3.30}$$

where Q is the assumed covariance matrix for process noise and T_s is the update rate (sampling time) of $50Hz$.

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

The previous updates are considered to always occur to the exact update rate. As we mentioned before, for the LiDAR and the fish-eye camera updates will occur only if the UAV is in their FOV. The measurement update provides an estimate of the unit vector that points to the UAV with respect to the UGV

$$\hat{z}_{cam} = \begin{bmatrix} \frac{\hat{r}_N}{d} \\ \frac{\hat{r}_E}{d} \\ \frac{\hat{r}_D}{d} \end{bmatrix} + \nu_{3\times 1,cam} \tag{3.32}$$

and the camera update's observation model is given as,

$$H_{cam} = \begin{bmatrix} 0_{1,3} & \frac{-(\hat{r}_E^2+\hat{r}_D^2)}{\hat{d}^3} & \frac{\hat{r}_N\hat{r}_E}{\hat{d}^3} & \frac{\hat{r}_N\hat{r}_D}{\hat{d}^3} \\ 0_{1,3} & \frac{\hat{r}_N\hat{r}_E}{\hat{d}^3} & \frac{-(\hat{r}_N^2+\hat{r}_D^2)}{\hat{d}^3} & \frac{\hat{r}_E\hat{r}_D}{\hat{d}^3} \\ 0_{1,3} & \frac{\hat{r}_N\hat{r}_D}{\hat{d}^3} & \frac{\hat{r}_E\hat{r}_D}{\hat{d}^3} & \frac{-(\hat{r}_N^2+\hat{r}_E^2)}{\hat{d}^3} \end{bmatrix} \tag{3.33}$$

where $\hat{d}$ is as equation (3.27) and the covariance matrix update of the form

$$P_{k|k} = (I_6 - KH_{cam})\, P_{k|k-1}^{-1}\, (I_6 - KH_{cam})^T + K(\delta R_{cam})K^T \tag{3.34}$$

with the scale factor $\delta = 1/sin(\alpha)$, where $\alpha = \tan^{-1}(-\frac{z}{||z||})$.

The LiDAR measurement update provides a position in the 3D space, and as in the version 1.0 results

$$\hat{z}_{LiDAR} = \begin{bmatrix} \hat{r}_N \\ \hat{r}_E \\ \hat{r}_D \end{bmatrix} + v_{3\times1,LiDAR} \tag{3.35}$$

This yields an observation model matrix of

$$H_{_{LiDAR}} = \begin{bmatrix} 0_{3,3} & -\mathbf{I}_3 \end{bmatrix}, \tag{3.36}$$

and covariance matrix update of

$$P_{k|k} = (I_6 - KH_{_{LiDAR}})\, P_{k|k-1}^{-1}\, (I_6 - KH_{_{LiDAR}})^T + K(\gamma R_{_{LiDAR}})K^T \tag{3.37}$$

where γ is a scale factor that is based up to the number of LiDAR points that are within the determined UAV bounding box. This version also updated the clustering algorithm for the LiDAR

measurement updates. As the measurements are provided at a lower frequency than its previous version, not the Point Cloud Library (PCL) [54] provides a more reliable result.
This localization filter resulted to be more accurate, in simulation, in providing the UAV position estimate as it is shown in the result section.

3.3.2 Version 2.0 results

This version of the localization filter has been tested in simulation and also on the hardware setup with the VICON system as ground truth for comparisons. Different form version 1.0, version 2.0 was providing the UAV localization estimate in real time. Also, the UAV was flying autonomously a set of waypoints using the Vicon measurements as the position feedback for the flight controller, while in version 1.0 each flight was manned.
The addition of autonomous flight and online localization capabilities allowed a more precise testing and future development described in Chapter 4.1. A sample of the performances this version of the localization filter provided, are shown in Figure 5.4.3.

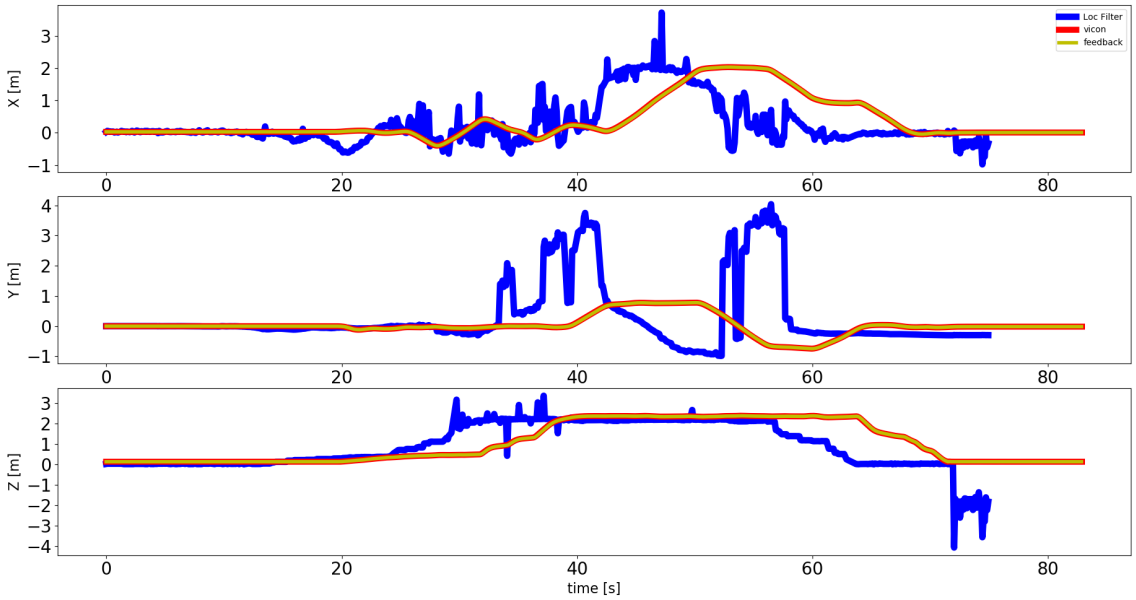


Figure 3.3.1: Localization filter version 2.0 results during field testing. In red the ground truth, also sent to the flight controller as the feedback position (yellow). In blue the localization solution with different outliers due to the camera and lidar clustering algorithm.

Table 3.3.1: EKF 3D positioning error statistics for the localization filter version 2.0 during flight tests.

	Duration (s)	RMS (m)	μ (m)	σ (m)	Median (m)	Max. (m)
X_{err}	82.89	0.4	0.17	0.36	0.06	1.71
Y_{err}	82.89	1.08	0.30	1.041	0.01	4.04
Z_{err}	82.89	0.48	0.12	0.47	0.04	4.09
$3D_{err}$	82.89	1.25	0.68	1.05	0.24	4.10

As shown in Table 3.3.1, the localization filter version 2.0 perform better than its predecessor. It can be noticed that the Y axis is heavily affected by the outliers measurements but the overall solution was improved compared to the version 1.0. During field testing with the real-world setup, it was noticed that the upward facing camera cluster algorithm would mostly fail and provide several

false positive detection of the drone and for this reason it was removed in the versrion 3.0.

3.4 Localization Algorithm Version 3.0

3.4.1 Filter description

The version 3.0 of the localization filter was developed to address some limitations of the Belief Space Planning algorithm that will be addressed in the next chapter. This version only differs from its predecessor for the introduction of a stereo camera on board of the drone for VIO purposes and the removal of the upward facing fish-eye camera. This upgrade was made either in simulation and in real-world scenario to improve the overall UAV localization performances [55]. The UAV has been equipped with an *"Intel® RealSense™ Tracking Camera T265"*, which is characterized by fisheye stereo vision and internal IMU. This tool provides a continuous velocity update of the UAV to the navigation filter described in Chapter 5 resulting in a more accurate and precise localization. For simulation purposes, the VIO is performed using [56], while in the real-world scenario the *"Intel® RealSense™ T265* provides a full pose and velocity of the camera.

As the navigation filter operates the same as in the previous chapter, a new measurement update function is added. VIO measurement estimate results as in Eq. 3.38

$$\hat{z}_{vio} = C_b^n \begin{bmatrix} v_x^{vio} \\ v_y^{vio} \\ v_z^{vio} \end{bmatrix} \tag{3.38}$$

where C_b^n is the rotation matrix of the form of eq. 2.3 and v^{vio} is the velocity vector provided by the

VIO. The observation model matrix is of the form

$$H_{vio} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{3.39}$$

and the measurement noise $\sigma = 0.12$.

3.4.2 Version 3.0 results

In Figure 3.4.1 is shown the results of the navigation filter version 3.0 in a simulation environment and the benefits of the VIO addition. VIO capabilities are adopted for mainly two reasons: first, as

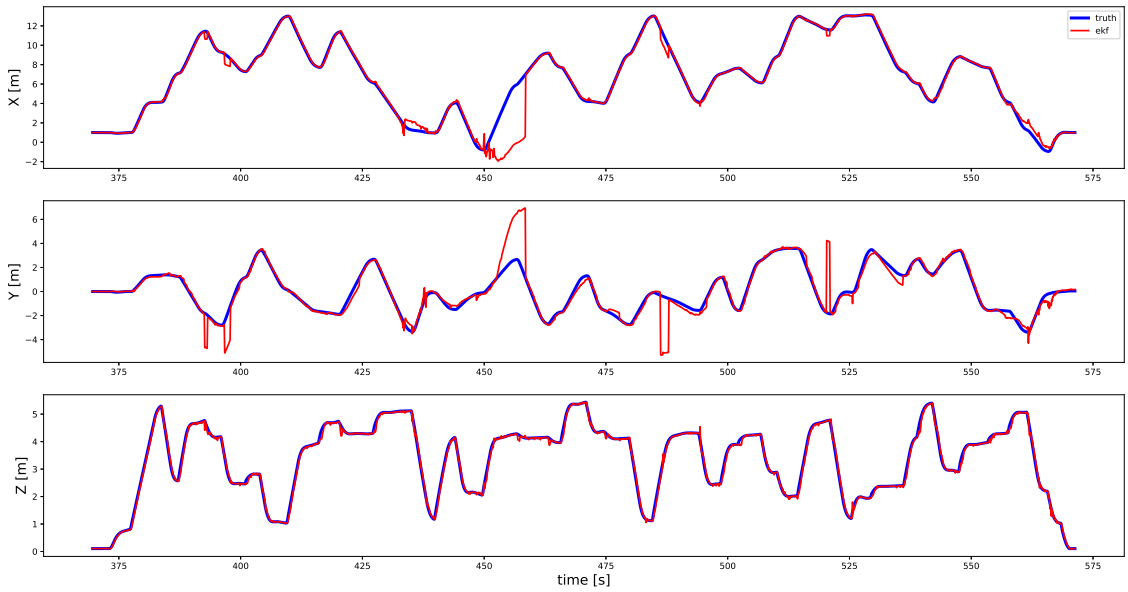


Figure 3.4.1: Localization filter version 3.0 solution in a simulation environment. In red the filter solution compared with the ground truth in blue.

discussed, it provides a more reliable UAV localization, and second, a sensor that is affected by the environmental conditions, such as lighting sources, is added. This addition provides the basis and reasons for the algorithm presented in chapter 6.

It needs to be considered that in simulation some sensor measurements are derived from the absolute UAV position. This helps the localization filter to provide very good performances. On the real hardware and during field testing, results were proved to be less precise as shown in Figure 3.4.2.

Table 3.4.1: EKF 3D positioning error statistics for the localization filter version 3.0 during flight tests.

	Duration (s)	RMS (m)	μ (m)	σ (m)	Median (m)	Max. (m)
X_{err}	109.86	0.71	0.43	0.57	0.37	2.03
Y_{err}	109.86	0.42	0.17	0.38	0.04	1.15
Z_{err}	109.86	0.35	0.11	0.33	0.09	2.25
$3D_{err}$	109.86	0.9	0.73	0.52	0.69	2.40

This discrepancy is mostly due to missed updates or outliers measurements during the field tests. For example, it was noticed that the LiDAR clustering algorithm provided false positive picking the operators. The results shown in Table 3.4.1 highlight the better performance of version 3.0 from the previous two. The results for the 3D position error of the three versions of the localization filter are summarized in Table 3.4.2. The previous table shows how each localization filter

Table 3.4.2: 3D positioning error statistics for the three versions of the localization filter.

	Duration (s)	RMS (m)	μ (m)	σ (m)	Median (m)	Max. (m)
v1.0 - $3D_{err}$	139.86	2.16	1.37	1.67	0.65	10.75
v2.0 - $3D_{err}$	82.89	1.25	0.68	1.05	0.24	4.10
v3.0 - $3D_{err}$	109.86	0.35	0.11	0.33	0.09	2.25

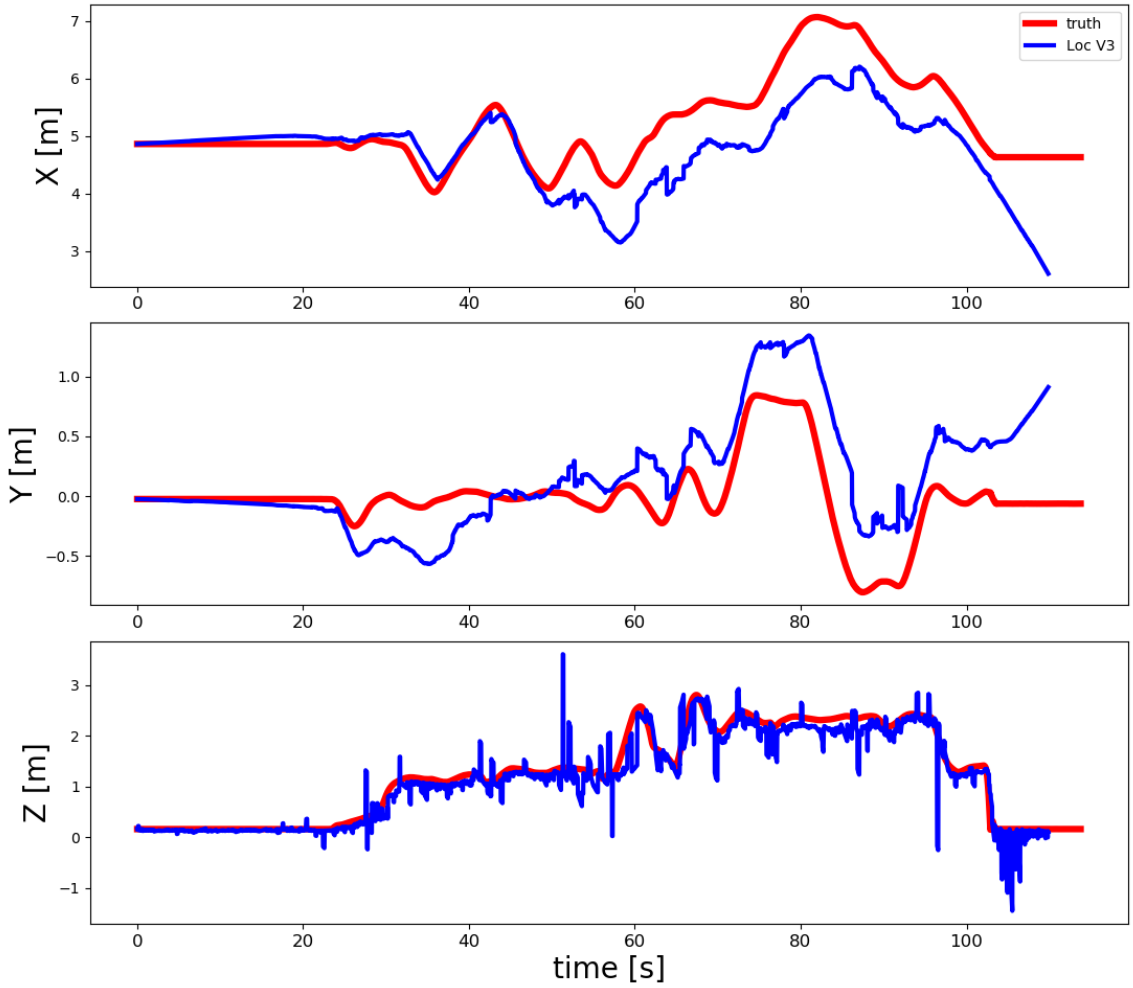


Figure 3.4.2: Localizaion filter version 3.0 results field testing. In red the ground truth result form the Vicon system, in yellow the position estimate used as the feedback for the flight controller, and in blue the localization filter position estimate.

was improved at each version iteration. As the overall results are promising, the maximum error is still high enough to consider these results to be used as a feedback solution for the flight controller. Since the environment is to be considered cluttered with obstacle, the overall error must reduce in order to have a safe flight using the localization filter solution.

3.5 Concluding Remarks

This chapter presented the evolution of the localization filter with its three versions. This is the foundation of this work and a necessary step for the implementation of the two planning algorithms with uncertainty discussed in the next chapters.

4

Autonomous flight

This chapter presents how the localization filter version 3.0 was used to provide the UAV's pose estimate to send as feedback to the flight controller such that complete autonomous flight could be achieved.

4.1 Localization Solution as a Feedback in the Loop for the UAV Position Control

As described in chapter 2.2, part of this research focused on having an autonomous UAV-UGV team to be able to test the planning algorithms on a real-world scenario. To achieve this high level of autonomy the UAV must have reliable pose feedback to send to the flight controller to perform waypoint navigation. Since the UAV flies in a GNSS-denied environment, the position feedback, first, is provided by a VICON motion capture system. This was helpful to test and tune all three versions of the localization filter. Most of the results discussed in this work assume the UGV to be still in its position.

The UAV/UGV team is considered a multi-agent subset where the UGV acts as the ground station providing the UAV the position feedback. Later the precise VICON solution was substituted by the localization filter version 3.0. Figure 4.1.1 shows how the navigation filter is used in feedback in the loop for the PX4 flight controller to perform waypoints navigation. Tests were conducted in a

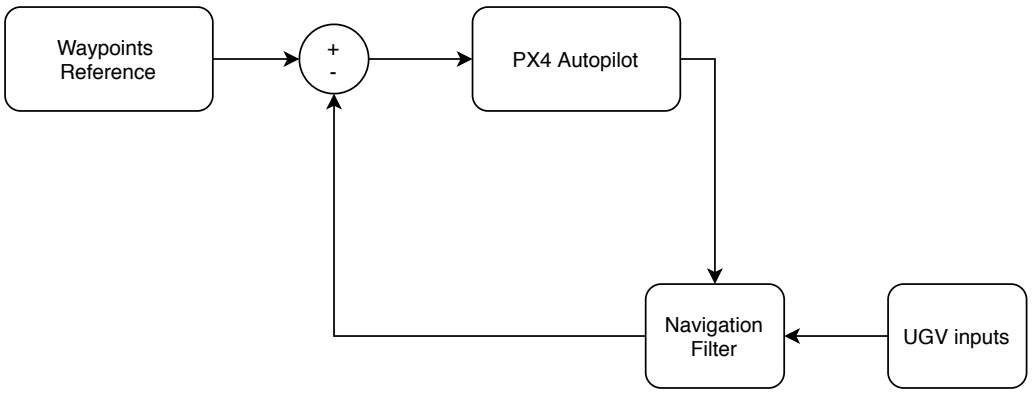


Figure 4.1.1: Control diagram concept with the localization filter in the feedback loop for the flight controller to perform waypoints navigation

WVU owned facility in Reedsville WV, which is characterized by a wind tunnel equipped with 40+ Vicon tracking cameras. This setup was used first to test the navigation filter, and second as ground truth for performance comparisons. Figure **??** shows the full system in its testing environment.

As mentioned, first the UAV uses a perfect pose from the Vicon system to perform waypoint navi-

Figure 4.1.2: UAV and UGV during the field testing in the WVU wind tunnel facility

gation and it is compared with the navigation filter solution in Figure 4.1.3. After extensive tuning,

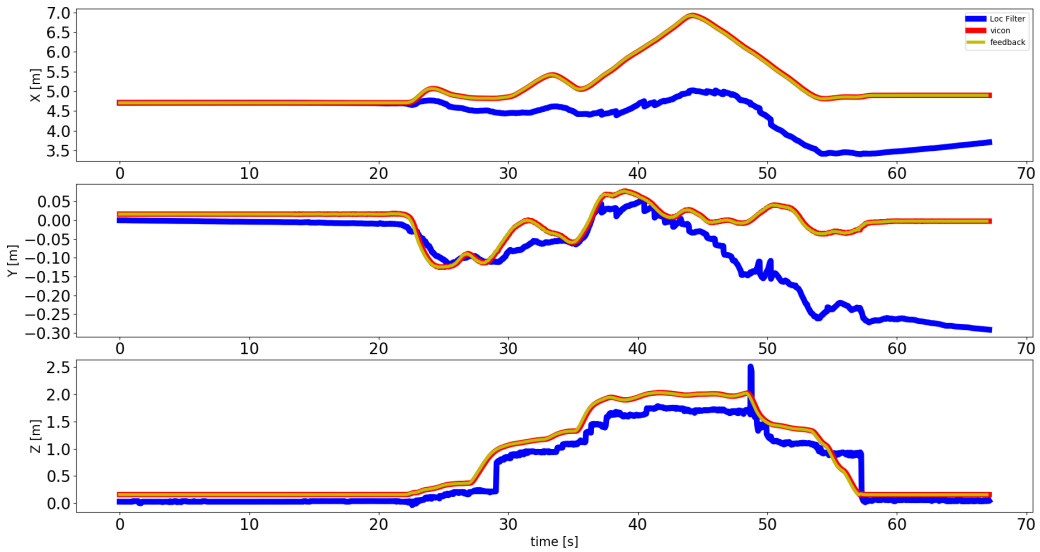


Figure 4.1.3: Localizaion filter version 3.0 results field testing. In red the ground truth result form the Vicon system, in yellow the position estimate used as the feedback for the flight controller, and in blue the localization filter position estimate.

the Vicon system is used as ground truth, while the UAV flight controller receives the navigation filter solution as a pose feedback. The results of this test are shown in Figure 4.1.4.

It is easy to address that the solution provided by the localization filter is not robust enough to have precise waypoints navigation. The nature of the testing environment allowed the experimentation of the localization filter as a source of position feedback with a successful and safe flight laying the basis for future tests and improvements. Also, the lack of obstacles and features in the environment heavily affected the localization filter estimation.

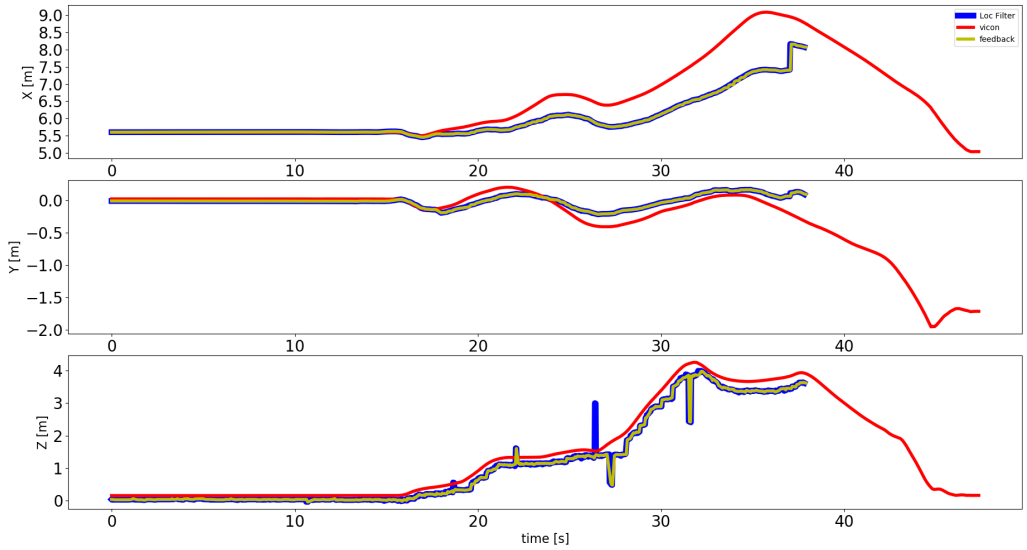


Figure 4.1.4: Localization filter solution used as a position feedback for the PX4 flight controller. In red the ground truth solution, in blue the localization filter version 3.0 solution and in yellow the position sent to the flight controller as the position feedback.

4.2 Concluding Remarks

This short chapter gave context to the development of the planning algorithm explained in chapter 6, and completed the full autonomy of the system for waypoint navigation. This milestone was a necessary step to be able to test the different planning algorithms in a real-world scenario.

5

Belief Space Planning Algorithm for space exploration

This chapter is focused on the autonomous navigation planning algorithm for the UAV to accomplish search and rescue tasks. The algorithm fuses sensor data from the UGV and the UAV and

consider their measurement uncertainty to decide which path provides better localization of the UAV. The work presented in this chapter is mainly based on [57].

5.1 Chapter Motivation

As it was explained in section 1.2 and with the use of the navigation filter version 2.0 described in section 3.3 the UAV's mobility is leveraged to perform the search task. Different from prior belief space planning research, the approach presented in this chapter has a required exploration goal and the UAV's localization uncertainty must be maintained to an acceptable level during the operation. The contribution of this chapter consists in a new path planning algorithm that fuses the capabilities of two robots (UAV/UGV) to balance between environment exploration tasks while reducing the localization error for the UAV along the selected trajectory.

5.2 Assumptions and Constraints

It is reasonable to assume that the UAV must be deployed from the UGV and returned to the UGV after a search, therefore, the planning algorithm is designed to provide a path that starts and ends at the UGV's location after exploring the map. Further, while the UAV performs its search mission, the UGV is assumed to be static to provide the UAV's localization estimate.

To support the belief state propagation during the planning, the algorithm simulates the sensor updates by assuming that the UAV moves with zero acceleration and at a constant velocity $v = 0.5$ m/s with pitch and roll of nearly zero. This simplification allowed to simplify the process of determining observation models considering the UAV flying in straight lines between waypoints. During the belief state propagation process, the algorithm takes into account an image degradation,

for the fish-eye camera, discarding any update whenever the UAV is further than 6 *meters* along either coordinate axes. This threshold was determined via testing the camera tracking performance.

The number of waypoints generated by the planning algorithm is constrained, such that the UAV's flight time is limited

$$T_{flight} < \rho \tag{5.1}$$

where ρ is a specific threshold in seconds determined by the Quad-rotor characteristics and payload. At the same time, to enlarge the covered area a random sampling-based algorithm is chosen to define waypoints. To make sure the maximum coverage all the connections between waypoints needs to be traveled guaranteeing that each path has the same entropy value.
While the UAV flight controller is set to follow the waypoint trajectory, the planner must also minimize the localization uncertainty

$$p_e = \min\left(\sum_{i}^{n} P_i^{pos}\right) \tag{5.2}$$

with P_i^{pos} being the covariance matrix P relative to the UAV's position at each time step.

5.3 Path Planning Algorithm

The algorithm must provide a path for the UAV to follow that meets the requirements and constraints described in the previous section 5.2. As mentioned in the assumptions, the planning algorithm can access a full map of the environment, that was provided by the UGV through SLAM, and offline produces the selected path for the UAV to fly. The algorithm consists of two phases and decouples the exploration task from the criterion to reduce the localization uncertainty [57]. First, using the PRM algorithm [24], a graph is built on 12 randomly generated nodes lying in the UAV

collision-free space favoring the map in front of the UGV where its LiDAR has maximum coverage. The number of nodes is heuristically selected as a trade off between exploration and computation. The algorithm connects each node using a 5-Nearest Neighbor (NN) approach that leads to a total of 32 edges in the free space, G(N,E) = G(12,32). Then, with the edges of this graph, the Route Inspection Problem (or Chinese Postman Problem (CPP)) [58] is solved. Since CPP requires the graph to be eulerian, in case a node has an odd number of edges, a new one will be created making sure it lies in the free space.

In the second phase, for each of the possible trajectories, the belief-state is propagated and used to select a path that has suitable position error uncertainty. The overall approach is shown in Fig. 5.3.1.

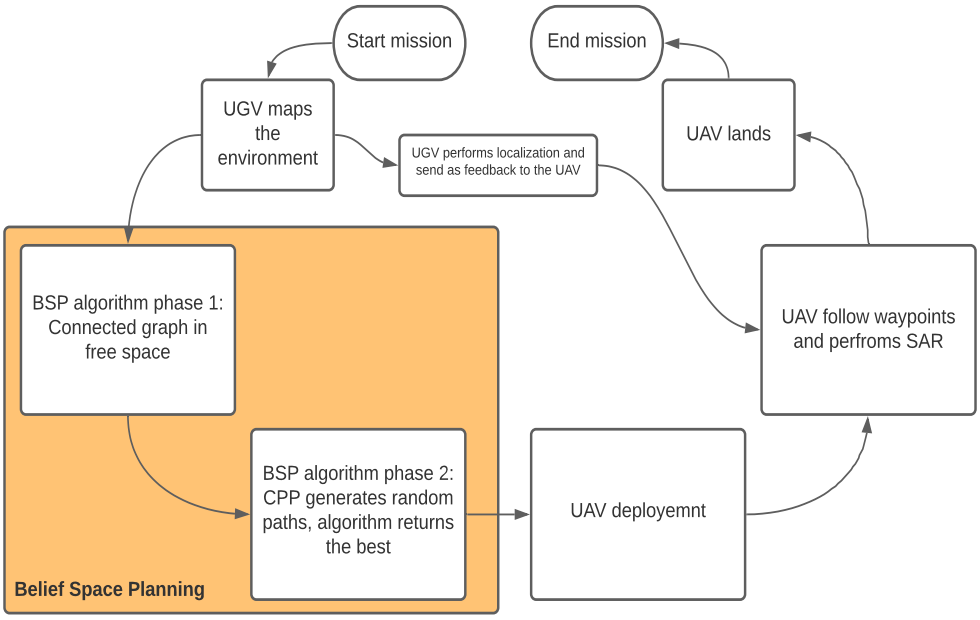


Figure 5.3.1: System block diagram for an hypothetical mission of the UAV/UGV team

5.3.1 Graph construction phase

The connected graph, generated by the algorithm in its first stage, is composed of nodes and edges that lie in the free space and without intersecting any obstacle as shown in Figure 5.3.2. This graph is used to generate a path starting at the *source* node and ending back to the same *source*. The sequence of edges that the UAV will follow is generated using the CPP [58]. It guarantees that each edge of the graph is traveled only once which is necessary for the UAV to explore more space without overlaps.

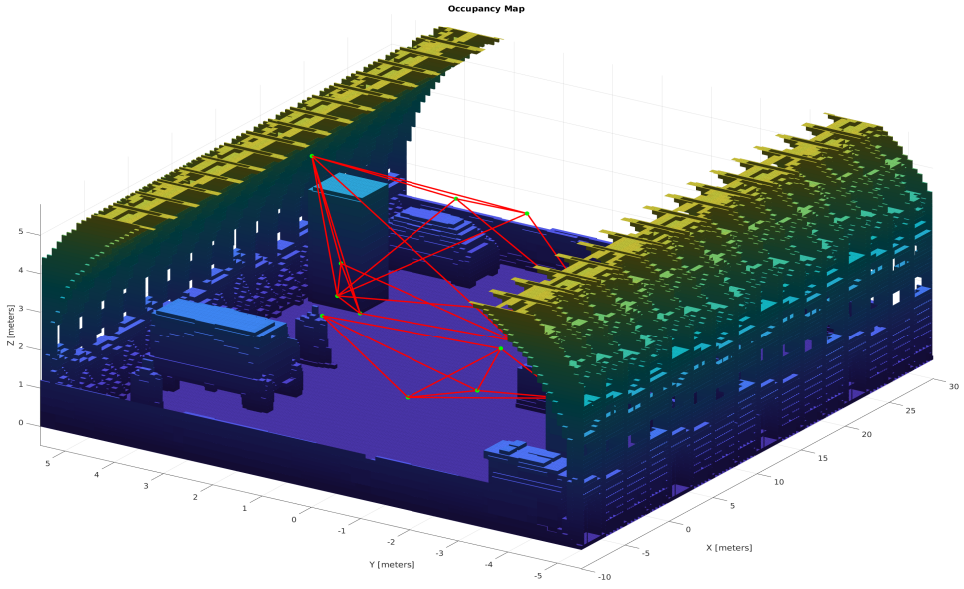


Figure 5.3.2: Octomap environment representation with the randomly selected nodes and edges provided by the first phase of the planning algorithm

A downside of having the source and the goal of the path being the same, along with with the goal of exploring the environment, consists in a single set of waypoints and edges that can be traversed

in different orders. To increase the variety of the solutions while keeping the same nodes, the CPP solver is run 80 times so that for each run all edges are always visited once but in a different order. This will change the belief propagation penalizing the paths that tend to be far from the camera and LiDAR field of view for a longer period of time. Also the order of waypoints followed by the UAV affects its localization uncertainty.

5.3.2 Belief state propagation

The belief state is propagated using the version 2.0 of the localization filter and the equations described in section 3.3. In this section, the EKF's error covariance updates are used to propagate the belief state for planning and the EKF implementation is used to evaluate the estimation performance. For each path generated by the first phase of the algorithm, the UAV's estimated error covariance is predicted taking into account all the sensors and their associated update rates along the path. As mentioned in Chapter 3.3 the main modification from version 1.0 consists of using a kinematic model of the UAV to propagate its error covariance between measurements update. The second phase of the algorithm selects the path with the lowest $\mathcal{L}_2$ norm of the sum of the position error covariance estimates after the belief state of each path is being processed.

5.3.3 Trajectory generation

The first phase of the planning algorithm selects waypoints and paths between them that do not collide with obstacles, generating a graph that lies in the free workspace of the 3D occupancy volume of the environment that was previously created by the UGV (Figure 2.5.3). In the second phase, the planning algorithm evaluates the 80 different trajectories, chosen heuristically to increase variability, to find the one that reduces the UAV localization uncertainty. Therefore the 3D

error covariance is calculated at each time step i as

$$PEC_i = ||P^i_{pos}||. \tag{5.3}$$

Next, the sum of this value was computed over the entire path in order to provide an indication of the quality of the error covariance of the flight j

$$PEC_{flight_j} = \sum_{i=1}^{t} PEC_i \tag{5.4}$$

where t is the number of time steps in the flight j. Finally, the best and worst trajectories were selected by finding the minimum and maximum of these values respectively among all propagated paths.

$$BEST_{path} = \min_{j=1,2,\ldots,80}(PEC_{flight_j}), \tag{5.5}$$

$$WORST_{path} = \max_{j=1,2,\ldots,80}(PEC_{flight_j}). \tag{5.6}$$

5.4 Belief Space Planning Algorithm Results

This section compares several tests designed to evaluate the performance of Beliefe space planning algorithm.

5.4.1 BSP performances

Figure 5.4.1 shows the value of each flight as in eq. 5.4 and in red the worst, green the best, black the second-worst, and yellow the second-best paths selected by the planning algorithm.

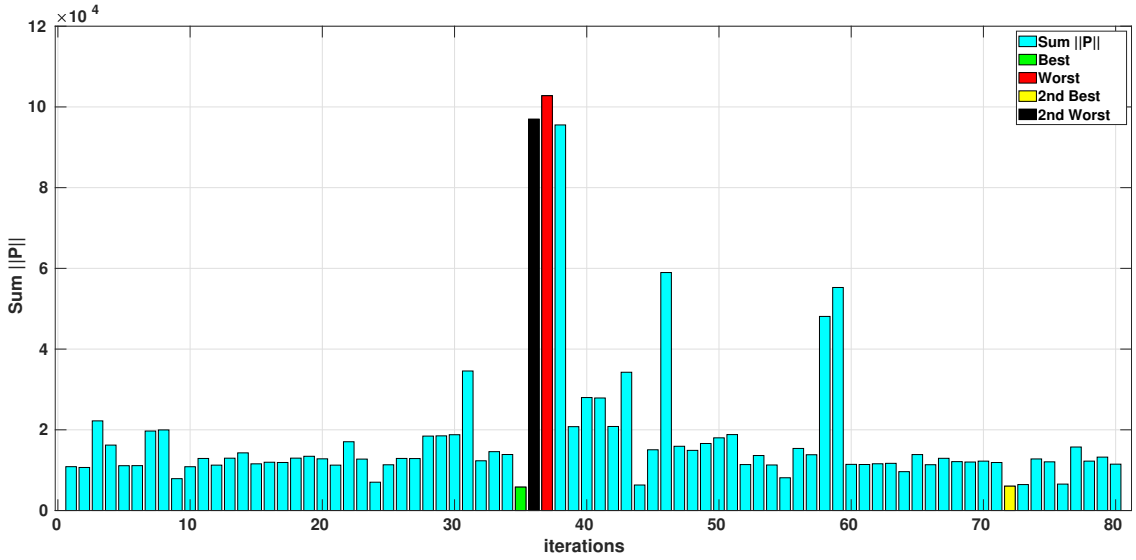


Figure 5.4.1: Sum of the $\mathcal{L}_2$ norm of the Position Error Covariance for each trajectory, in red the one with the maximum sum position error covariance and in green the one with the minimum. Also in black the path with the second maximum sum position error covariance and in yellow the respective second minum.

A side-by-side of the estimated uncertainty of the four trajectories selected are also shown for comparison in Table 5.4.1.

Table 5.4.1: Belief Space Planning algorithm results for the best and worst solutions.

	Best Path	Worst Path	2^{nd} Best Path	2^{nd} Worst Path
3D RMS Pos. Err. Cov. (m^2)	120.97	2527	121.1	2518.1
3D Max. Pos. Err. Cov. (m^2)	5796.4	1.03e+5	6026.8	96947
σ (m^2)	120.9	2526.7	121.1	2518
μ (m^2)	4.55	80.7	4.7	76.1
Median (m^2)	1.0006	1.0005	1.0005	1.0005

After the planning algorithm selected the best and worst trajectories, they were evaluated in the simulation environment described in Chapter 2.2 and compared to show the benefits of the proposed approach. For these experiments, all trajectories, picked by the algorithm and flown by the UAV, were simulated using the ground truth as pose feedback for the flight controller. The error-state EKF solution was not used as position feedback to facilitate the evaluation of the planner's ability to determine a trajectory without having to consider the implications of poor localization feedback in the execution of the path.

An error-state EKF was run "online" to estimate the UAV's position with respect to the UGV. Ten simulations of each set of waypoints were executed to better understand the trends of the planning algorithm. During the evaluation of the algorithm, it was determined that the simulation always returned different solutions even when the same list of waypoints was provided. This was due to the flight controller using an internal estimator to provide the UAV's local position during the waypoints navigation. For example, to show the differences between each simulation that was provided the same set of waypoints, the UAV's ground truth of each run is represented in Fig. 5.4.2. Because the planning algorithm was the main focus of this study, no major changes have been conducted to the UAV's autopilot estimator. For this reason, the average trends of multiple simulations, of the same set of waypoints, were considered.

Figure 5.4.3 shows the UAV's EKF estimated position compared to the ground truth when executing the best trajectory. In addition, the results of the UAV position estimation for the worst trajectory are presented in Figure 5.4.4, where it is evident how the EKF performs worse when comparing the two cases. Table 5.4.2 summarises the best and worst trajectory results. The standard deviation

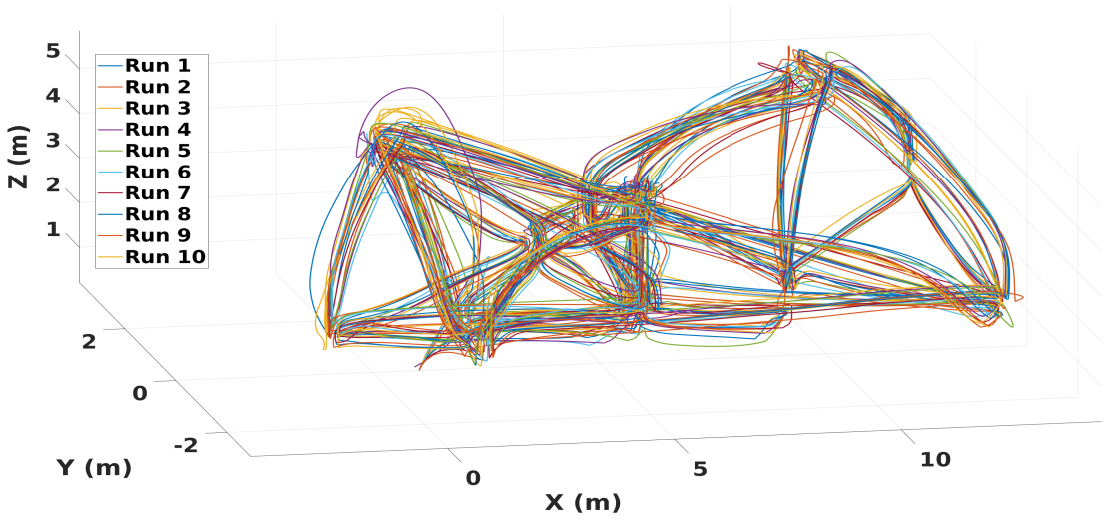


Figure 5.4.2: 3D representation of the ground truth for each run. This is meant to show how the simulator affected the results

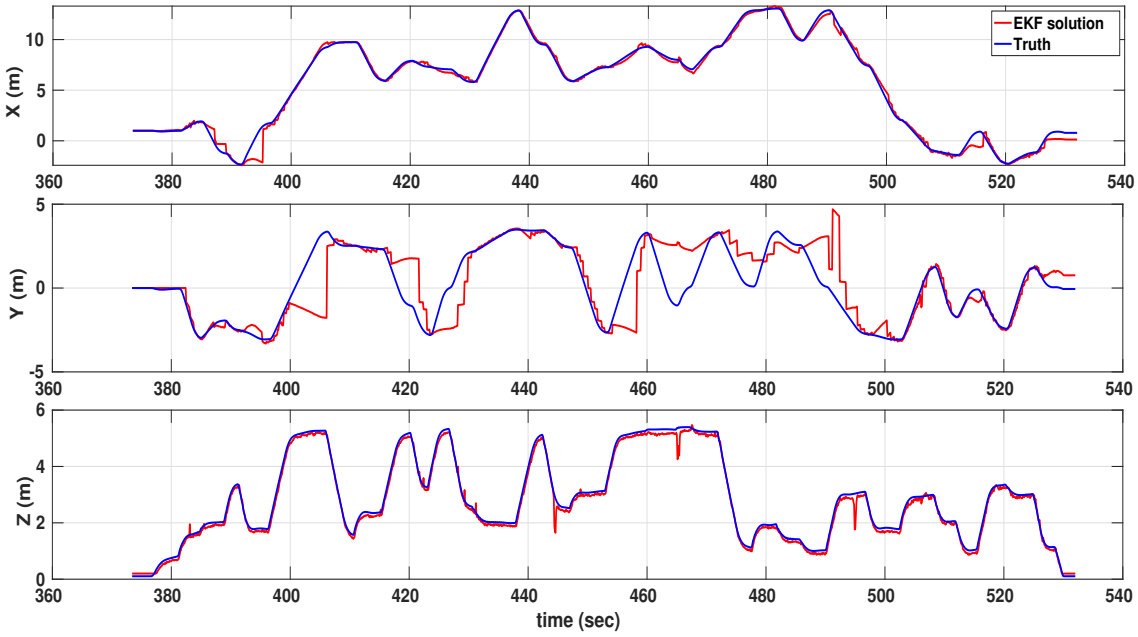


Figure 5.4.3: In red the EKF position estimation of the **best** trajectory, while in blue the ground truth.

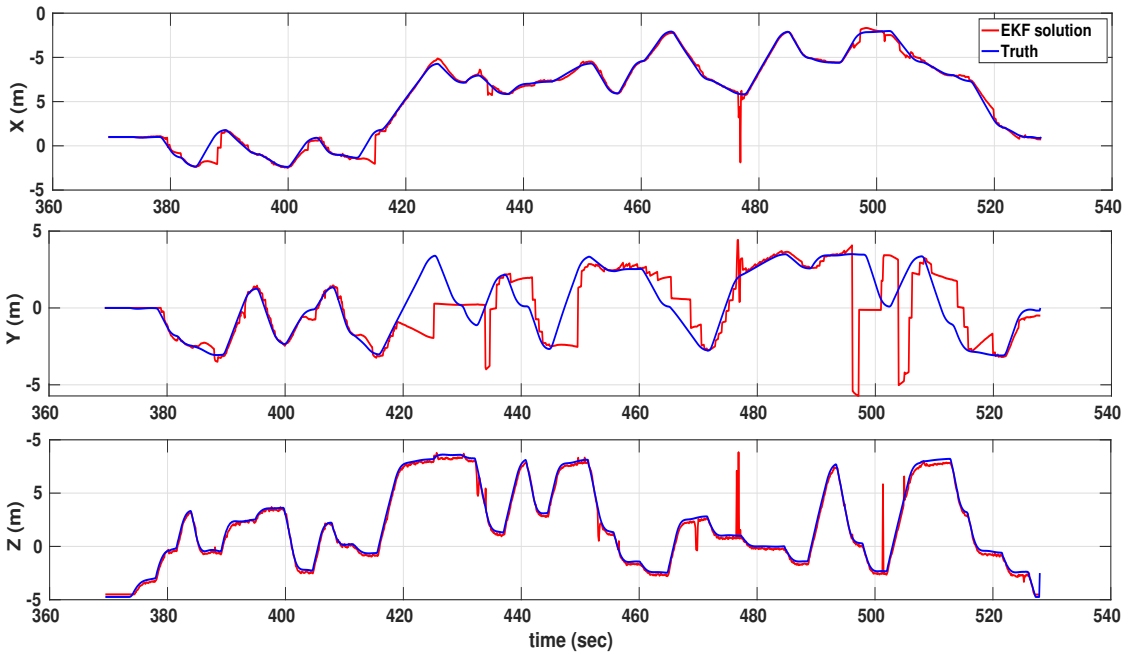


Figure 5.4.4: In red the EKF position estimation of the **worst** trajectory, while in blue the ground truth which is also the solution provided to the flight controller.

σ and the mean μ of the 3D positioning error are also reported in the above mentioned table. It can be shown that there is almost a 0.3 meters 3D RMS improvement between the two paths.

The results in Table 5.4.2 shows how the 3D RMS position error is generally smaller for the best trajectory as compared to the worst but mostly, how the maximum position error is reduced by almost a half. The results just presented did not provide a clear benefit of the algorithm. This is due to issues related to outliers and missing observations that are not modeled in the belief propagation.

5.4.2 Algorithm validation results

Due to the non readability of the realized paths, as shown in Fig. 5.4.2, the need to conduct multiple tests to present consistent results by the planning algorithm is essential [57]. Table 5.4.3 shows the

Table 5.4.2: BSP algorithm solution. Best and worst path compared.

	Best Path	Worst Path
Flight time (s)	158.70	158.52
X_{RMS} (m)	0.25	0.54
Y_{RMS} (m)	1.38	1.68
Z_{RMS} (m)	0.15	0.19
RMS 3D Pos. Err. (m)	1.46	1.77
σ (m)	1.1	1.42
μ (m)	0.95	1.07
Median (m)	0.46	0.47
Max Pos. (m)	5.24	9.24
# LiDAR Updates	91	97
# Camera Updates	162	161

results of ten runs of each best and worst trajectories.

To summarize the data in the previous tables, Table 5.4.4 shows the mean and median for the set of ten flights of the best and the worst trajectories. As shown in Table 5.4.3, the use of the best trajectory does not always guarantee a better UAV localization in absolute value, but multiple runs demonstrate that the planning approach is beneficial for reducing UAV position error. The same study, discussed in the last section, is conducted on what the planning algorithm considers to be the "second" best and "second" worst trajectories to guarantee repeatability performances. As before, Table 5.4.5 and Table 5.4.6 show the results for this second set of trajectories. Compared to the first set of simulations (Best vs. Worst) the second set does not quite behave as expected. This shows that the proposed algorithm could fail when there is a consistent presence of outlier measurements that are not taken into consideration by the planning algorithm. Also, some missed measurement

Table 5.4.3: Comparison of the best and worst path after being executed ten times to test repeatability.

Path	Run	Mean	Median	RMS (Pos. Err.)	MPE
Best Path	1	1.19	0.55	1.76	9.53
	2	1.02	0.80	1.38	5.03
	3	1.15	0.52	1.90	10.09
	4	0.99	0.46	1.51	5.52
	5	1.03	0.49	1.61	7.81
	6	0.97	0.46	1.57	7.91
	7	0.95	0.46	1.46	5.24
	8	1.05	0.49	1.58	5.28
	9	1.01	0.40	1.66	6.71
	10	1.01	0.50	1.54	7.36
Worst Path	1	1.16	0.43	2.16	15.68
	2	1.54	0.54	3.56	24.63
	3	1.22	0.53	2.08	8.19
	4	0.92	0.49	1.37	5.33
	5	1.07	0.47	1.77	9.24
	6	2.12	0.56	4.55	26.23
	7	0.98	0.44	1.60	8.21
	8	0.98	0.43	1.58	6.62
	9	0.90	0.40	1.42	6.39
	10	1.86	0.70	3.64	25.12

Table 5.4.4: Mean and Median of multiple runs for best and worst trajectories.

Path		Mean	Median	RMS (Pos. Err.)	MPE
Best Path	Mean	1.04	0.52	1.60	7.05
	Median	1.02	0.49	1.57	7.03
Worst Path	Mean	1.27	0.50	2.37	13.56
	Median	1.11	0.48	1.93	8.73

Table 5.4.5: Comparison of the second best and second worst path after being executed ten times to test repeatability.

Path	Run	Mean	Median	RMS (Pos. Err.)	MPE
2^{nd} Best Path	1	1.02	0.50	1.57	6.29
	2	1.12	0.49	1.76	6.48
	3	1.48	0.61	3.12	26.96
	4	1.05	0.45	1.68	6.47
	5	1.27	0.58	2.03	8.20
	6	1.05	0.49	1.67	7.47
	7	1.20	0.52	1.84	6.87
	8	1.11	0.61	1.68	7.44
	9	0.98	0.46	1.55	6.83
	10	1.09	0.51	1.66	7.74
2^{nd} Worst Path	1	1.32	0.47	2.32	9.56
	2	0.97	0.42	1.52	6.53
	3	1.00	0.43	1.68	8.09
	4	1.13	0.54	1.75	8.43
	5	0.90	0.42	1.44	6.08
	6	1.01	0.53	1.47	5.61
	7	1.07	0.53	1.59	5.79
	8	0.98	0.41	1.62	7.19
	9	1.08	0.47	1.67	6.19
	10	2.75	0.59	5.89	25.06

Table 5.4.6: Mean and Median of multiple runs of the second best and second worst trajectories.

Path		Median	Mean	RMS (Pos. Err.)	MPE
2^{nd} Best Path	Mean	1.14	0.52	1.86	9.08
	Median	1.10	0.50	1.68	7.16
2^{nd} Worst Path	Mean	1.22	0.48	2.09	8.85
	Median	1.04	0.47	1.65	6.86

might occur even if the UAV is in the sensors' FOV and it affects algorithm decision process since it does not consider the back-end algorithms used to provide information from the sensors' raw data. These reasons are the foundation of chapter 6 and this work, to investigate a new solution to reduce the discrepancy between the planning algorithm decision and the result obtained, making it more robust to outliers measurements and missing sensor observations.

5.5 Concluding Remarks

This chapter described an offline planning algorithm that finds a path that minimize the UAV's position uncertainty between the ones that maximize the covered area. In a simulation environment, the approach was shown to offer promise for selecting the waypoints' order to reduce the UAV's position uncertainty. In particular, the planning algorithm showed to be able to choose the path where the UAV is more favorable to be localized by the discussed EKF estimator.

6

Gaussian Process Based Navigation Algorithm for a UGV-UAV Team

This chapter describes new developments on both sides, hardware, and software, to improve the UAV's localization. Also, a new approach intended to improve the algorithm discussed in Chapter

5 is presented. In this chapter, the effort is focused on how the environment can affect the planning algorithm and its use of the belief. In Chapter 5 the planning algorithm was focused on determining the path that minimized the UAV's localization error taking into account only the sensors' model. Now instead, a new approach is proposed to also take into account the surrounding environment and the way it can affect the path planning algorithm using machine learning techniques trained on real sensor data. Because a subterranean environment could easily be affected by different light conditions, dust, or smoke, each sensor would perform differently, most likely compromising the path planning algorithm in its decision. The proposed idea is to use machine learning, in particular supervised learning, to improve the path planning algorithm in recognizing sensors' faults due to environmental conditions. The supervised learning considered in this chapter is the GP and some preliminary investigations are conducted to make sure that this tool can improve the planning algorithm in considering also the environment condition and not only the sensor models. As shown throughout this dissertation, "building blocks" have been assembled to proceed in the further investigation to solve a well-defined problem that affects robots' autonomy. Since the UGV/UAV team in underground navigation is a fairly new research field, which had many successes among the science community, the problem of robust localization and navigation is still a wide-open topic.

6.1 Gaussian Process for the UAV Position Estimate

6.1.1 Effects of the environment on the planning algorithm

The algorithm described in Chapter 5 assumes perfect environmental conditions. As it is reasonable to assume that a search and rescue scenario does not have moving obstacles, it is a simplification not to consider how the environment is affected by different lighting conditions, areas with

heavy smoke, dust rose by the UAV or even some Radio Frequency (RF) interference. These non-perfect conditions could affect directly the localization filter and consequently the planning algorithm. As the first effect is clear, since each sensor could provide faulty observations if the environmental conditions are not ideal, the second regards the fact that the planning algorithm does not consider possible sensor failure during the belief space planning.

6.1.2 Paths generation

As mentioned at the beginning of this chapter, the GP regression problem is used to improve the planning algorithm in its decision making. In this case, GP is able to predict the UAV's position error taking into account the sensors' models and the environmental conditions.

Similar to the BSP algorithm described in chapter 5, this approach also uses paths that are generated from randomly selected waypoints. Each path is a set of waypoints chosen within the free space after an accurate map of the environment is provided by the UGV. A heuristic number of random waypoints **N** are chosen to explore the environment from an elevated point of view. Similarly to the BSP algorithm described in chapter 5, the map is transformed into a 3D occupancy map to guarantee that each waypoint and its connections do not lay or cross any obstacle respectively.

The path planning algorithm interprets the waypoints as nodes of the graph and connects them with edges to create a fully connected graph where it is possible unless any of the edges cross obstacles, where in that case they must be eliminated. To increase the map exploration the UAV flies each edge at least once, however, the graph created $G(N, E)$ is not guaranteed to be fully connected and Eulerian and for this reason, the UAV may navigate some edges multiple times.

As in the BSP algorithm, the UAV is required to take off and land from the same location, therefore each path must start and end from the same node. To generate the sequence of edges of the graph

G, that the UAV has to fly, the algorithm randomly selects an edge that is connected to the node according to the adjacent matrix. Next, that selected edge is removed from the matrix of "available edges" and recursively all edges are selected until each edge is visited at least once. The algorithm for the path generation is described in 1, where it is highlighted when it is necessary for the UAV to fly over already visited segments. An overview of the information flow is shown in Figure 6.1.1.

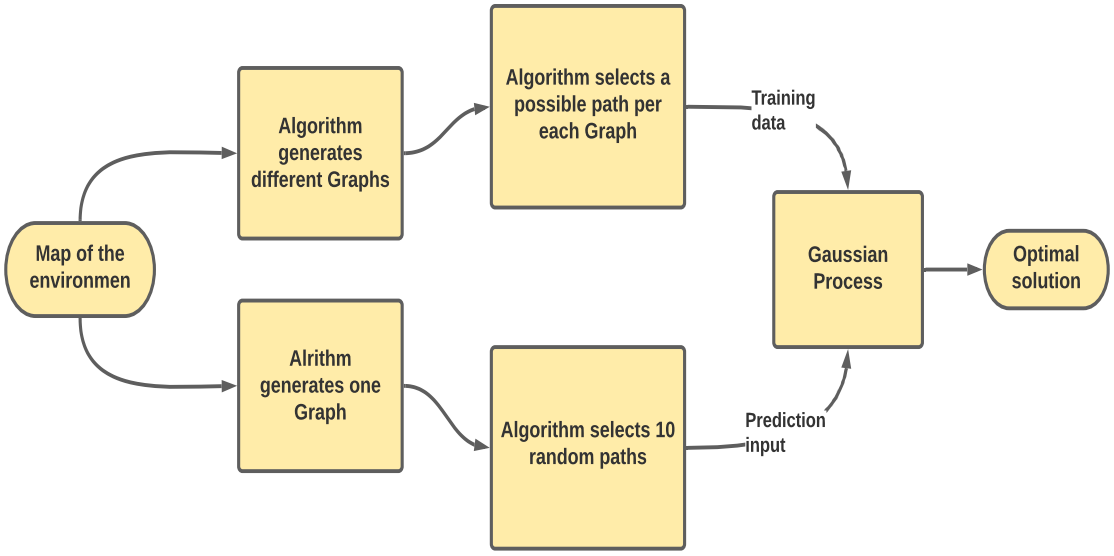


Figure 6.1.1: Concept diagram and information flow of the GP planning algorithm. On the top row the graph generator provides the training data for the GP algorithm to approximate the function that describes the relationship between inputs and output. The bottom row the graph generator provides the different sequences of waypoints for the GP algorithm to use to predict the localization error of the UAV.

Algorithm 1: Path generation algorithm

```
Result: Graph generation
while number of nodes do
    generate random node in space;
    if node is on obstacle then
        generate new;
    else
        node list = new node;
    end
end
while node list size do
    Adjunct matrix(node list(i), node list(i+1)) = distance between nodes;
    i = i+1;
    if Adjunct matrix(i,j) intersect obstacle then
        Adjunct matrix(i,j) = 0;
    end
end
Result: Path selection
Adj = Adjunct matrix;
while !path completed do
    for the number of nodes do
        if element of the raw of Adj are != 0 then
            create list of available nodes L;
        end
    end
    from L select random node to add to the path;
    switch to 0 the correspondent element of Adj;
    if Adj raw is 0 then
        find all available nodes from current node using Adjunct matrix;
        add the node that has more not yet travelled connection to path;
        if sum of Adj == 0 && path(last) == first node then
            path is completed
        end
        if sum of Adj == 0 && path(last) != first node then
            if Adj(last,1)!= 0 then
                path(last + 1)=first node;
                path is completed;
            end
        end
    end
end
```

6.1.3 Path selection

Given the fixed set of waypoints, multiple paths are generated according to the algorithm 1. As edges are selected randomly, these paths will mainly differ in the order the waypoints are reached. This differentiation wants to address how the order of waypoints affects the UAV error localization. The best path, within this fixed number of options, is the result of an evaluation of the Gaussian Process prediction for which order of waypoints will reduce the error position estimate of the UAV. To leverage the GP, first a set of four random different paths are used to explore different parts of the map and they compose the training set of the GP regression problem. The number of paths used for the GP training phase was chosen empirically as a trade off between computing time and data representation. For this study, a multi-input and single output GP has been chosen to predict the UAV's position error in each axis. The GP is used to predict the relationship between the UAV position estimate (inputs) in the Cartesian coordinates and each axis position error (outputs). This relationship is meant to map a 3D position in the environment and the UAV's localization error associated with it. Equations 6.1 and 6.2 show the GP inputs and outputs respectively.

Inputs:

$$\mathbf{X}_{in} = \begin{bmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \end{bmatrix} \tag{6.1}$$

Outputs:

$$\begin{aligned} \mathbf{Y}_{x_{err}} &= |\mathbf{x} - \hat{\mathbf{x}}| \\ \mathbf{Y}_{y_{err}} &= |\mathbf{y} - \hat{\mathbf{y}}| \\ \mathbf{Y}_{z_{err}} &= |\mathbf{z} - \hat{\mathbf{z}}| \end{aligned} \tag{6.2}$$

A different set of waypoints from the training ones is chosen to determine which order of nodes reduces the UAV position error estimate. After the GP is trained, the new set of inputs are used to predict the estimated UAV position error in the 3D space. The GP will determine the function that best approximate the input-output relationship

$$\begin{aligned} \mathbf{Y}_{x_{err}} &\sim f(\mathbf{X}_{in}) \\ \mathbf{Y}_{y_{err}} &\sim f(\mathbf{X}_{in}) \\ \mathbf{Y}_{z_{err}} &\sim f(\mathbf{X}_{in}) \end{aligned} \tag{6.3}$$

where the function $f(\cdot)$ is approximated by the GP as in eq. 2.27.

For the prediction phase, the same waypoints are used to randomly generate ten different paths. Due to the small experimental environment, also in this case, the number of paths was chosen as a trade off between variability and computational time for the GP to predict the one that minimizes the UAV position error. This way the algorithm returns information on how each sensor provides updates according to the UAV's current position in the map.

The GPy library [59] is used to train the GP regression problem and to predict the output using the validation data. A combination of two Kernels is chosen to better approximate the input-output relationship. Given the input 6.1 and the output 6.2 the sum of the radial basis function kernel (eq. 6.4), which is the most common, and the Matern3/2 Kernel (eq. 6.5), that instead helps to better represent some of the sparse data, is used to optimize the GP.

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right) \tag{6.4}$$

where σ^2 and l are the variance and length-scale respectively. The second kernel function considered is the Matern 3/2 described as

$$\begin{aligned} k_{Mat32}(r) =& \sigma^2(1+\sqrt{3}\sqrt{\sum_{i=1}^{in-dim}\frac{(x_i-x_i')^2}{l_i^2}}) \\ & \exp(-\sqrt{3}\sqrt{\sum_{i=1}^{in-dim}\frac{(x_i-x_i')^2}{l_i^2}}) \end{aligned} \tag{6.5}$$

Consequently, the GP uses the new set of waypoints and their ten randomly selected paths, generated as described in subsection 6.1, to predict the UAV's position error. Similar to the BSP, the GP algorithm chooses the path that minimizes the UAV's position error prediction. Figure 6.1.2 represents the GP results and the prediction for one of the ten candidate paths. The blue solid line represents the function $f(\cdot)$, with its confidence (lighter blue shade), that describes the relationship expressed in eq. 6.3. Figure 6.1.2 shows every single output affected by the three-dimensional input composed of the 3D position of the drone. For visual purposes, each subplot is a "slice" of the output compared to one input assuming the other two fixed.

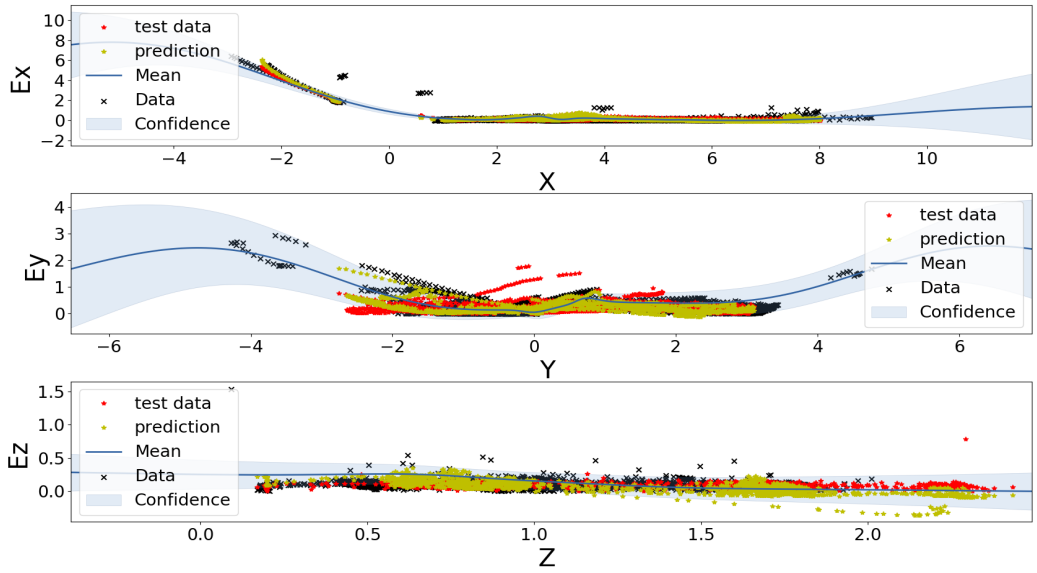


Figure 6.1.2: GP results with the black $\times$ representing training data, the red $\star$ the validation data, the blue line being the GP mean with its confidence and the yellow $\star$ representing the GP prediction.

To choose the best path, the GP algorithm evaluates the sum of the 3D predicted error distance of the UAV and it weights it according to the length of the path as in eq. 6.6.

$$L^{j}_{3D_{err}} = \frac{\sum_{i=1}^{n} \sqrt{e^2_{x_i} + e^2_{y_i} + e^2_{z_i}}}{n} \tag{6.6}$$

Each path j is represented by the result of eq. 6.6 where n represents the number of measurements. This is necessary since each path could be different in length if some edges of the graph were removed because of obstacles intersection. To select the best, the GP algorithm simply picks the minimum 3D UAV's predicted position error between the ten flights as in eq. 6.7.

$$BEST_{path} = \min_{j=1,\dots,10} L^{j}_{3D_{err}} \tag{6.7}$$

The result section 6.2 describes in details the differences between the BSP and the GP algorithm and shows how the latter better predicts the UAV's position error, using a more accurate representation of the sensors' model and the environment description.

6.2 Gaussian Process Based Algorithm Results

To show the validity of the method described in the previous section, different tests have been conducted in a simulated environment with different scenarios, or"toy examples" that were designed to address a limitation of the BSP approach that was implemented and described in chapter 5. The GP algorithm is compared to the BSP to show the benefits and improvements of this new contribution.

The first test (subsection 6.2.1) is conducted in a simulated room with artificial light and the GP-based planning approach is compared to BSP. The second test (subsection 6.2.2), instead, will use the same environment but some areas are poorly lit such that the vision system measurements and performances are degraded. In this case, differently from the GP, the BSP has no knowledge of the dark areas of the environment. The third (subsection 6.2.3) and fourth (subsection 6.2.4) experiments are similar to the previous with the addition of an unmodeled obstacle, which will be a source of missing information for both algorithms. In these situations, the BSP algorithm is unable to consider the obstacle since it was not present in the prior map. On the other hand, the GP algorithm leverages the training data to predict the missed information due to the obstacle obstruction.

6.2.1 Gaussian Process planning algorithm vs Belief Space Planning in a well-lit environment

To determine which order of waypoints better reduce the UAV's localization error in the simulation environment the GP algorithm is compared with the results obtained using the BSP approach. The GP training data set consists of four different random sets of waypoints that explore different areas of the environment. As previously mentioned, the number of paths for the GP training was chosen heuristically as a trade-off between computation and acceptable prediction results. Each set is composed of ten randomly placed nodes connected at most with the 6-nearest neighbors after checking that none of them collides or lays on obstacles.

These different sets of waypoints are flown by the UAV in the simulator and the collected data is used to train the GP. A different set of nodes and edges, which are not used during the training process, are randomly explored by the UAV. The algorithm guarantees that each waypoint is reached minimizing the number of times the edges of the graph are visited multiple times. Within these paths, the algorithm will be able to differentiate which area of the map provides worse position measurements update, therefore it will choose the one that flies those degraded areas as last to reduce the position error propagation.

The simulated environment is shown in Figure 6.2.4 and it is fully lighted to compare the performances between the GP prediction planning algorithm and the Belief space algorithm. Since the BSP does not take into consideration environmental conditions that could lead to a sensors' missed or faulty measurements, it would return always the same result for the bast path and its results are shown in Figure 6.2.1 and Table 6.2.1.

The GP-based planning algorithm predicts the UAV's localization error on the same ten paths

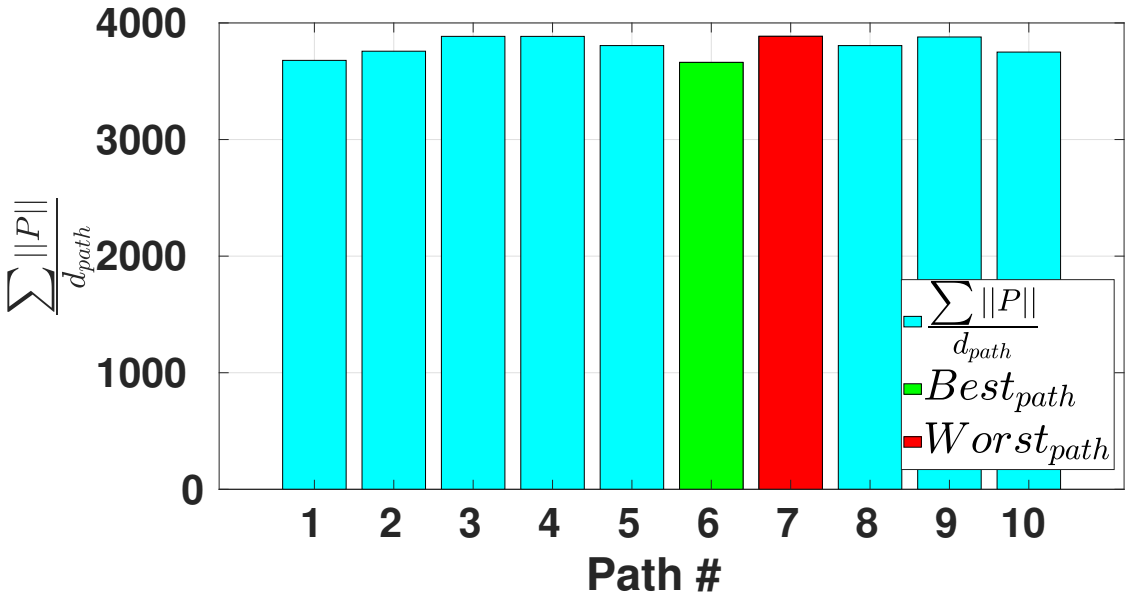


Figure 6.2.1: Belief Space Planning algorithm results for the well-lit environment. In green the best path that minimize the UAV's localization error.

Table 6.2.1: BSP results of the best path compared to the worst path.

Path	Median	Mean	RMS (Pos. Err.)	MPE
Best Path	0.227	0.227	0.227	3661.6
Worst Path	0.241	0.241	0.241	3885.3

and pick the one that minimizes it. The results are shown in Table 6.2.2 and in Figure 6.2.2.

As shown in Figure 6.2.2, the GP-based planning algorithm predicts that path number 7 provides a better UAV's localization than the number 6 selected by the BSP algorithm. The two best choices are compared and the results are shown in Table 6.2.3.

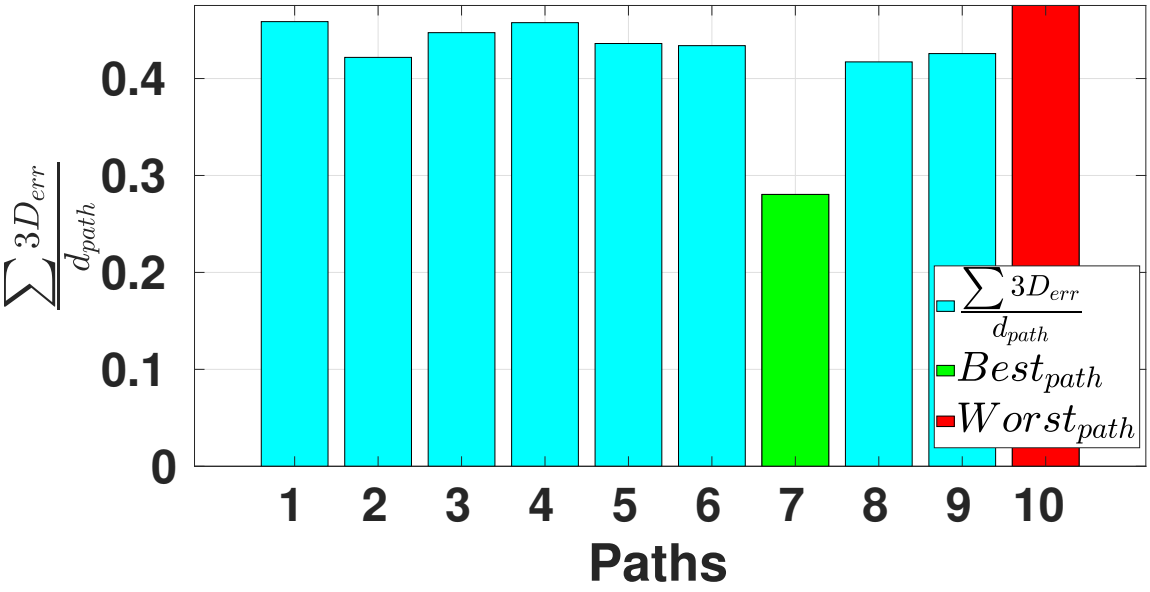


Figure 6.2.2: Gaussian Process prediction results in the well-lit environment. In green the best path that minimize the UAV's localization error.

Table 6.2.3: Comparison between the simulation of the best path according to the GP-based planner and the best according to BSP for a well lit environment.

	GP Best Path	BSP Best Path
Flight time (s)	163.6	170.2
RMS 3D Pos. Err. (m)	0.43	0.79
σ (m)	0.28	0.60
μ (m)	0.32	0.50
Median (m)	0.25	0.32
Max Pos. (m)	2.24	3.94

Fig. 6.2.3 shows the BSP solution compared to the EKF estimation error along with the more

Table 6.2.2: GP planning algorithm results in a well lit environment for the prediction of the 10 paths. In red the one that performs worse compared with the one in green that which minimizes the localization error.

Path	Median	Mean	RMS (Pos. Err.)	Sum. Err
1	0.26	0.46	0.81	0.46
2	0.25	0.42	0.77	0.42
3	0.29	0.45	0.76	0.45
4	0.29	0.46	0.83	0.46
5	0.28	0.44	0.77	0.44
6	0.29	0.43	0.71	0.43
7	0.22	0.28	0.34	0.28
8	0.25	0.42	0.72	0.42
9	0.29	0.43	0.73	0.43
10	0.29	0.48	0.89	0.48

accurate and GP planned solution. Previous results show how the GP-based prediction is more accurate in selecting the correct best path. This is due to the fact that the GP algorithm has access to actual sensors measurements and truth data for the training and the prediction can take into account outliers or missing updates.

6.2.2 Gaussian Process planning algorithm vs Belief Space Planning in a dark environment

The comparison discussed in the previous part it is now repeated simulating an unevenly lit environment with dark areas that can affect VIO as shown in Figure 6.2.4. Since BSP does not take into consideration the light change its result will be the same. The GP-based planner, on the other hand,

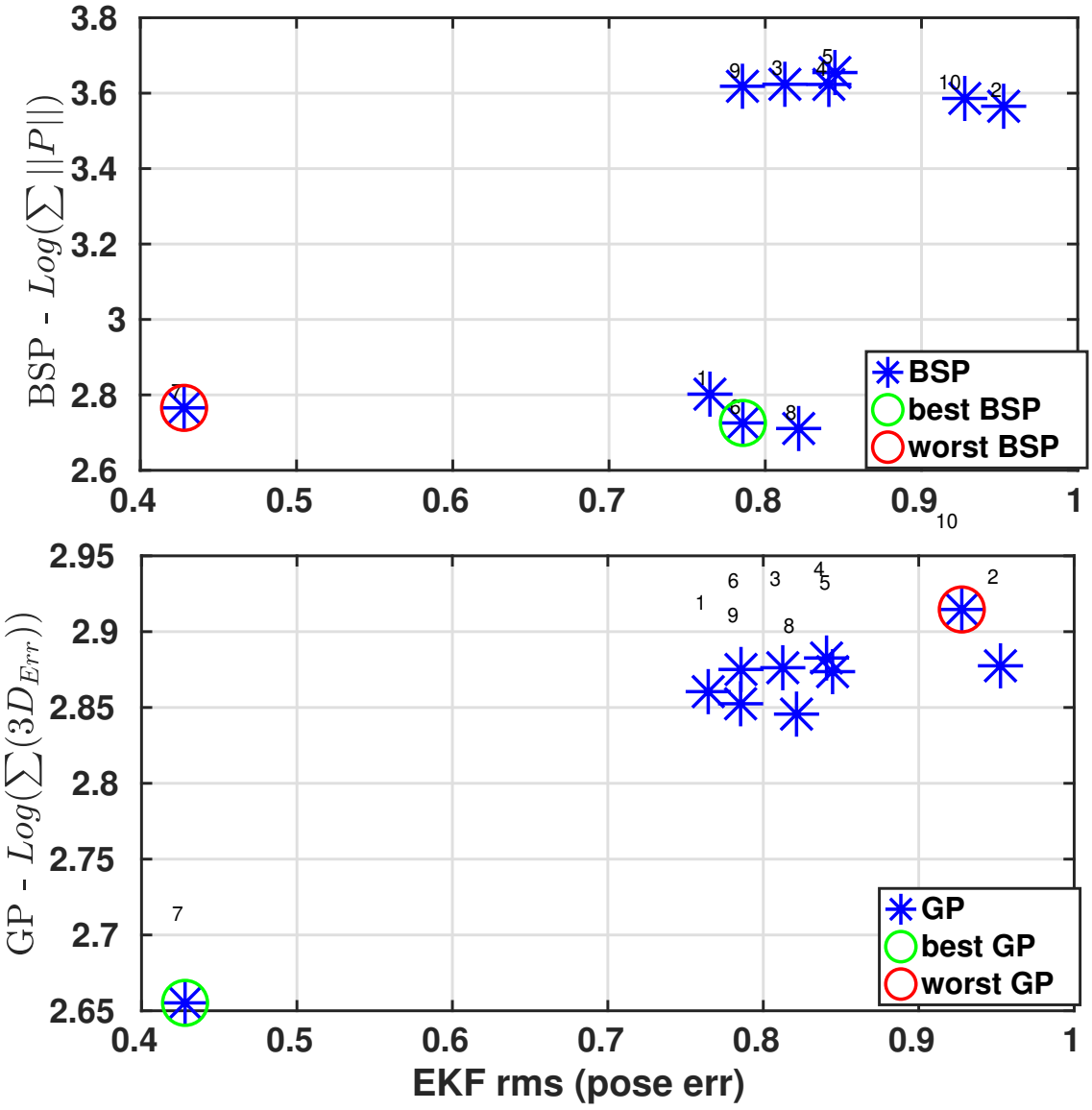


Figure 6.2.3: The BSP sum of the norm of the position error covariance matrix, related to the localization filter position error estimate, are compared with the GP planning algorithm sum of the prediction error related to the EKF position error estimate for a well-lit environment test.

uses training data collected in the dark environment. In Table 6.2.4 are shown the GP predicted

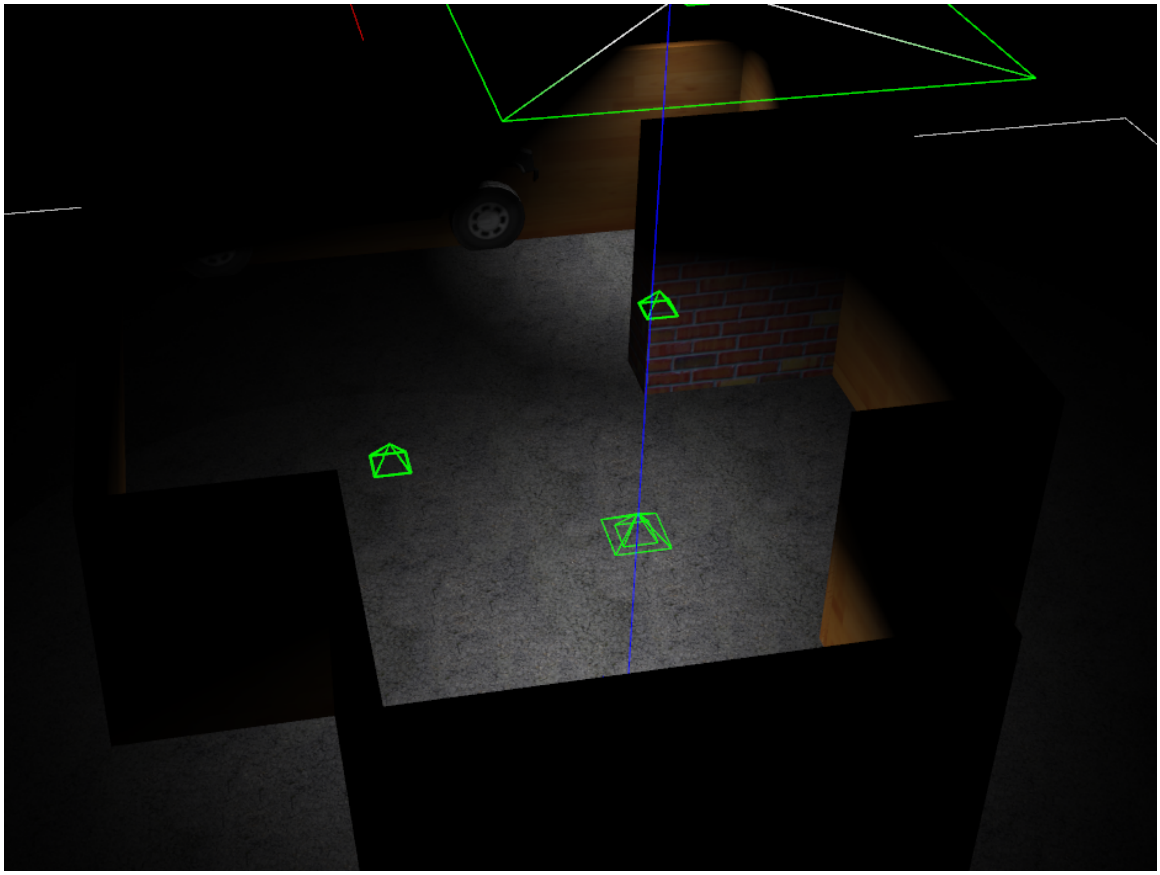


Figure 6.2.4: Test environment with different lit or dark areas.

performance for a dark environment, while instead, Table 6.2.4 shows how a dark environment increase the overall localization error and how the GP-based planner is able to find the one the path that significantly reduces it compared to the others.

Figure 6.2.5 shows how in a dark environment the GP-based planning algorithm is capable to predict the path that will minimize the UAV's localization error compared to the BSP algorithm. that

Table 6.2.4: GP algorithm results in a dark environment for the prediction of the 10 paths. In red the one that performs worse compared with the one in green that which minimizes the localization error.

Path	Median	Mean	RMS (Pos. Err.)	Sum. Err
1	0.40	0.74	1.15	0.74
2	0.35	0.64	0.98	0.64
3	0.39	0.71	1.19	0.71
4	0.34	0.47	0.63	0.47
5	0.38	0.64	0.98	0.64
6	0.40	0.65	0.92	0.65
7	0.37	0.63	0.98	0.63
8	0.35	0.61	0.95	0.61
9	0.37	0.62	0.99	0.62
10	0.36	0.64	1.02	0.64

Table 6.2.5: Comparison between the simulation of the best path according to GP and the best according to BSP in a dark environment.

	GP Best Path	BSP Best Path
Flight time (s)	159.2	170.9
RMS 3D Pos. Err. (m)	0.43	0.73
σ (m)	0.25	0.57
μ (m)	0.35	0.46
Median (m)	0.29	0.28
Max Pos. (m)	2.26	3.89

performs worse as shown in Table 6.2.5.

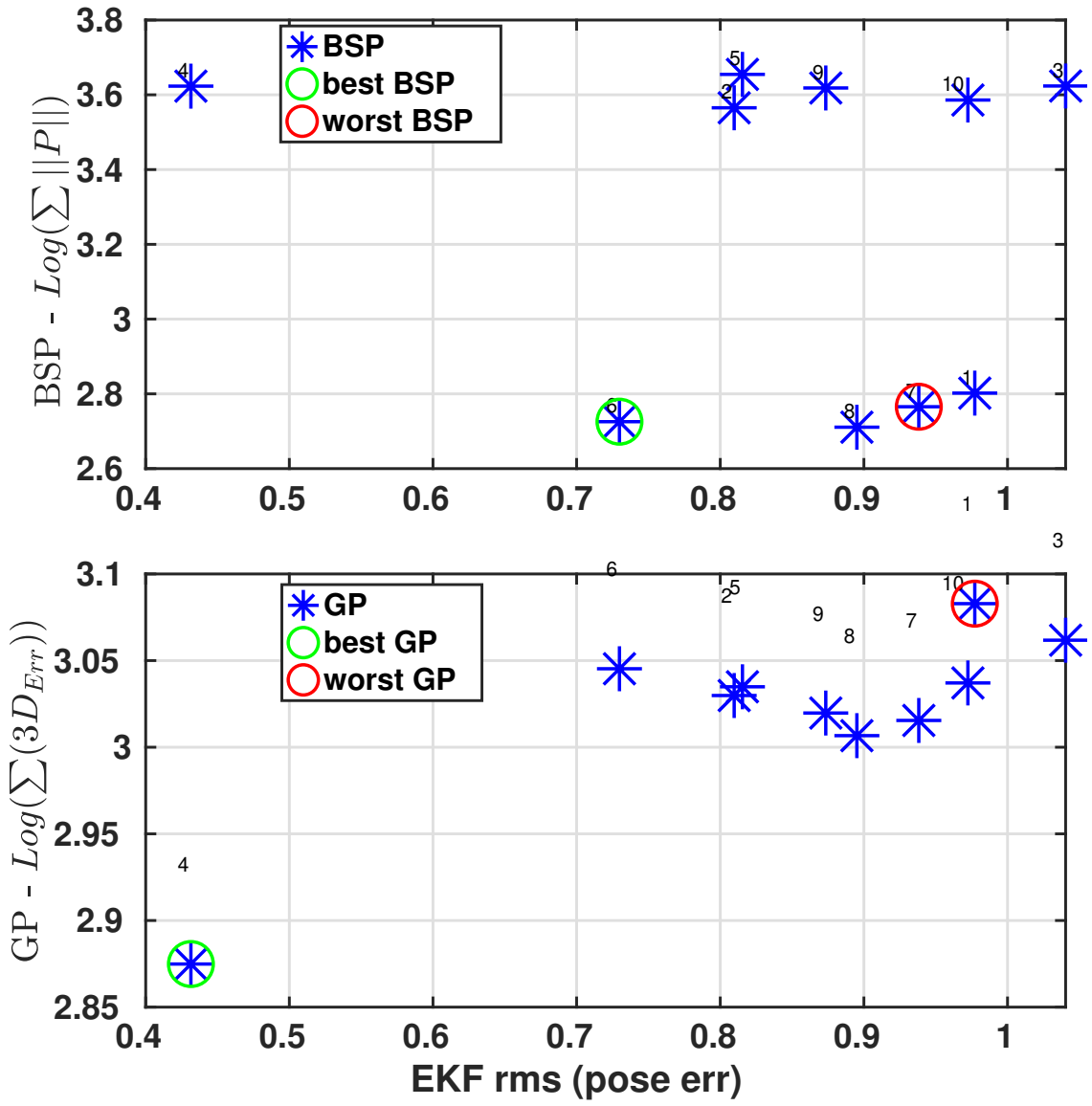


Figure 6.2.5: The BSP sum of the norm of the position error covariance matrix, related to the localization filter position error estimate, are compared with the GP planning algorithm sum of the prediction error related to the EKF position error estimate for the dark environment test.

6.2.3 Gaussian Process planning algorithm vs Belief Space Planning with unmodeled obstacles in a well-lit environment

One downside factor of the BSP is the possibility of wrong or not precise modeling of the environment or sensors. The GP, instead, uses data to be trained and then predicts the UAV's localization error. For this reason, a similar study is conducted with the addition of an obstacle in the environment. While GP is able to include the occlusion provided by the added obstacle, the BSP will have no knowledge of it making it generate the same results as in the previous subsection. Figure 6.2.6 shows the updated environment with the newly added obstacle.

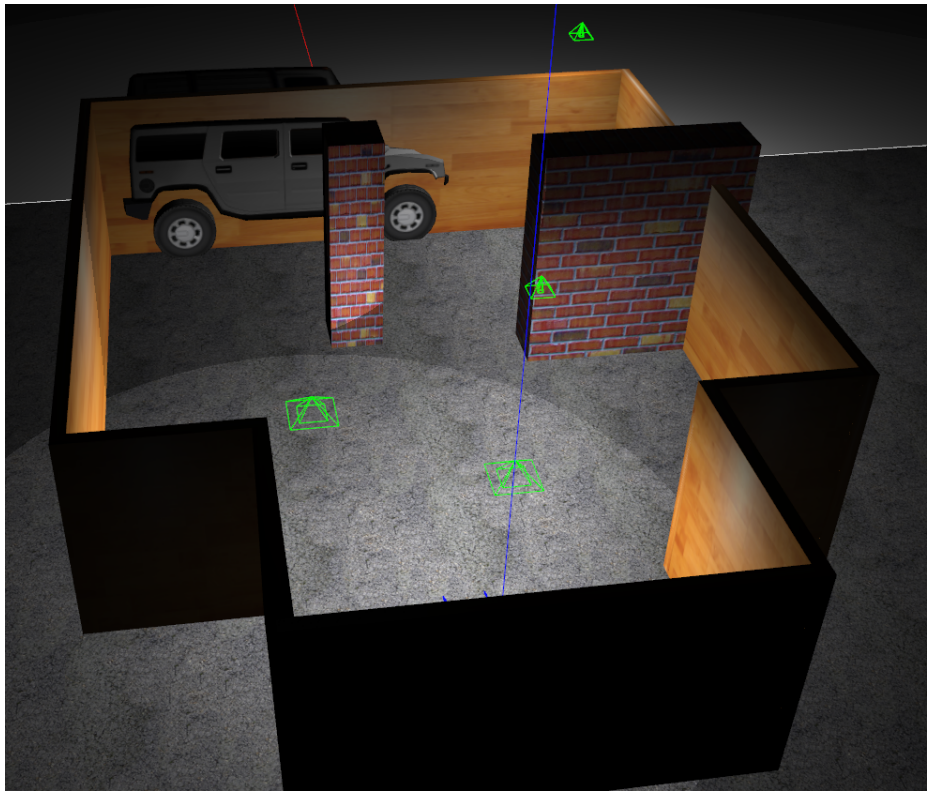


Figure 6.2.6: Environment with the addition of a column as obstacle.

As mentioned in the previous paragraph the BSP solution will not change due to its lack of knowledge of the new obstacles, while the GP-based planner leads to the following results shown in Figure 6.2.7 and Table 6.2.6.

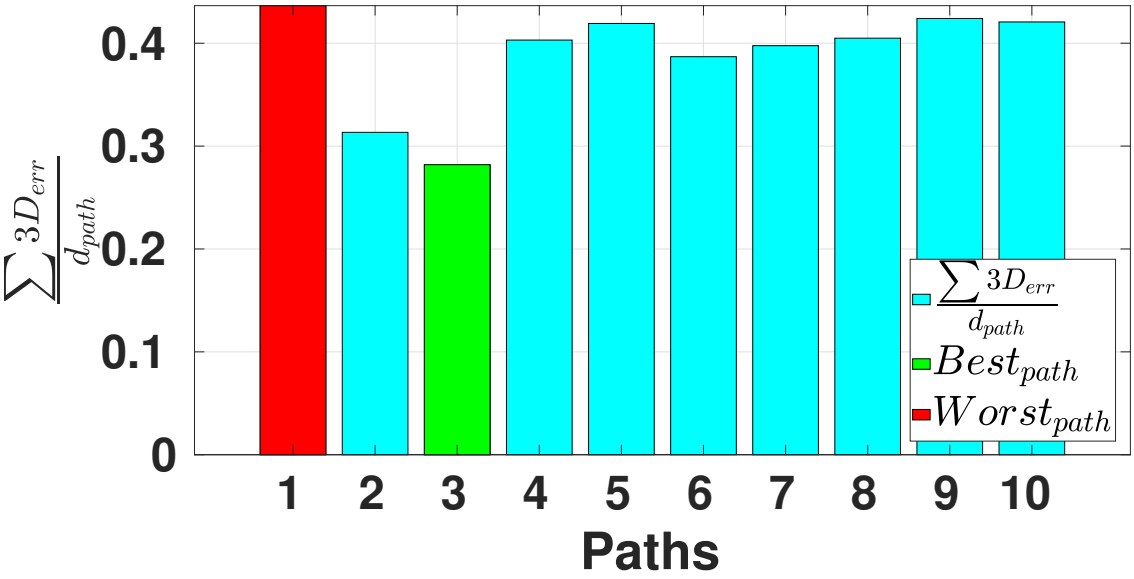


Figure 6.2.7: GP-based planning algorithm choice of the best path, compared to the worst for a well-lit environment and an unmodeled obstacle.

To understand how the GP-based planning algorithm performs compared to the BSP, their results are plotted with the UAV's EKF estimated position error in Figure 6.2.8.

As shown the GP is able to predict the missing measurements due to the obstacle occlusion and still choose the path that results in the lowest localization error of the UAV. The comparison between the two algorithms is summarized in Table 6.2.7.

Table 6.2.6: GP-based planning algorithm results in a lit environment for the prediction of the 10 paths with the addition of an obstacle. In red the one that performs worse compared with the one in green that which minimizes the localization error.

Path	Median	Mean	RMS (Pos. Err.)	Sum. Err
1	0.25	0.44	0.75	0.44
2	0.24	0.31	0.43	0.31
3	0.25	0.28	0.32	0.28
4	0.24	0.40	0.67	0.40
5	0.25	0.42	0.74	0.42
6	0.25	0.39	0.61	0.39
7	0.24	0.40	0.70	0.40
8	0.24	0.40	0.71	0.40
9	0.25	0.42	0.76	0.42
10	0.25	0.42	0.71	0.42

Table 6.2.7: Comparison between the simulation of the best path according to the GP-based planner and the best according to BSP in a lit environment and unmodeled obstacle.

	GP Best Path	BSP Best Path
Flight time (s)	163.1	171.6
RMS 3D Pos. Err. (m)	0.52	0.70
σ (m)	0.41	0.54
μ (m)	0.32	0.45
Median (m)	0.24	0.28
Max Pos. (m)	7.10	3.76

6.2.4 Gaussian Process planning algorithm vs Belief Space Planning with unmodeled obstacles in a dark environment

In this last scenario, both algorithms face a degraded scene with dark areas where sensors' performances are affected. Also, the presence of an unmodeled obstacle will guarantee some missing

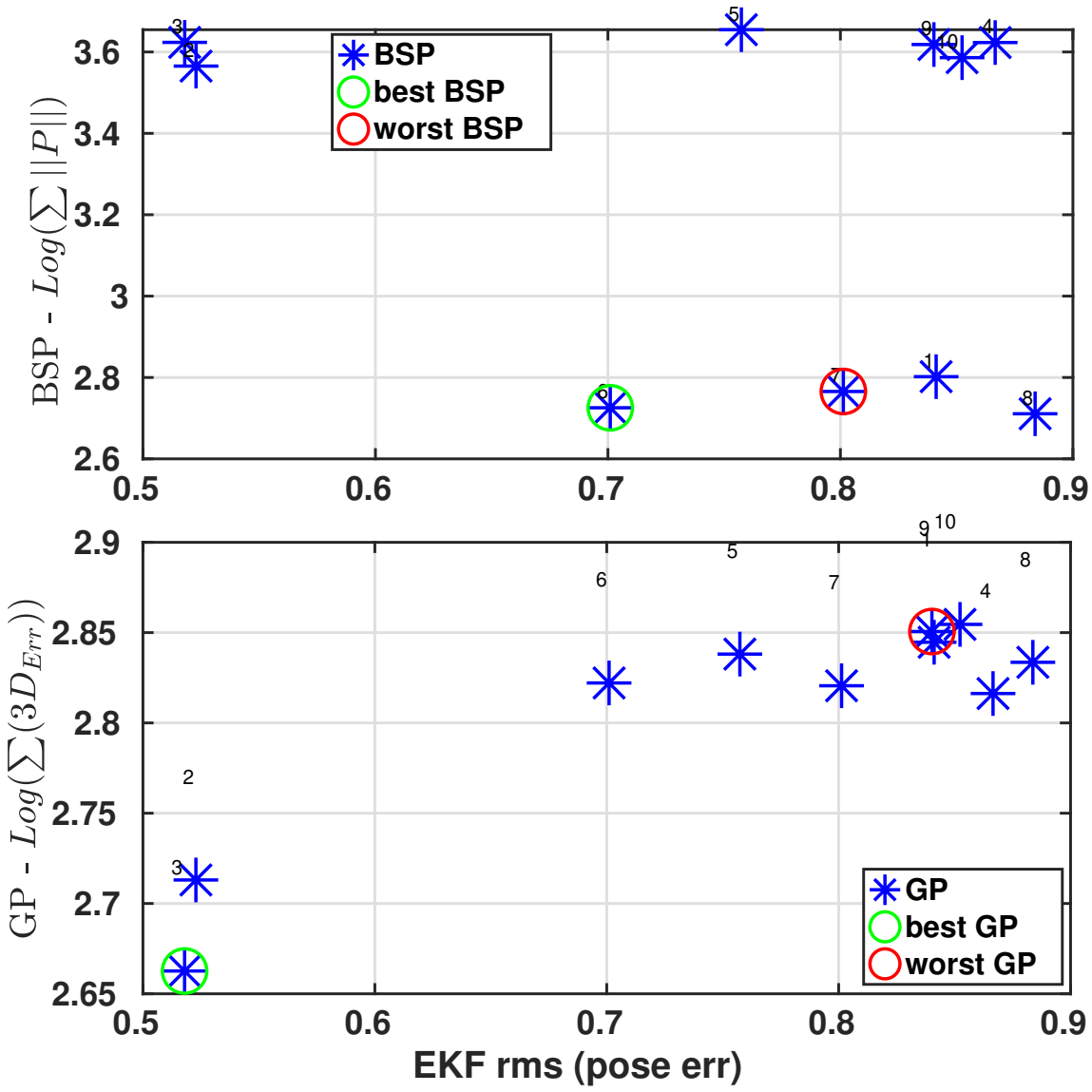


Figure 6.2.8: The BSP sum of the norm of the position error covariance matrix, related to the localization filter position error estimate, are compared with the GP planning algorithm sum of the prediction error related to the EKF position error estimate for the well-lit environment test and the added non-mapped obstacle.

measurements from some sensors. This will help to show the difference between the two algorithms in handling missing information. The following results show that the prediction better handles the dark areas and the added obstacle compared to the BSP. The algorithm's best path is shown in Figure 6.2.9 and results are described in Table 6.2.8.

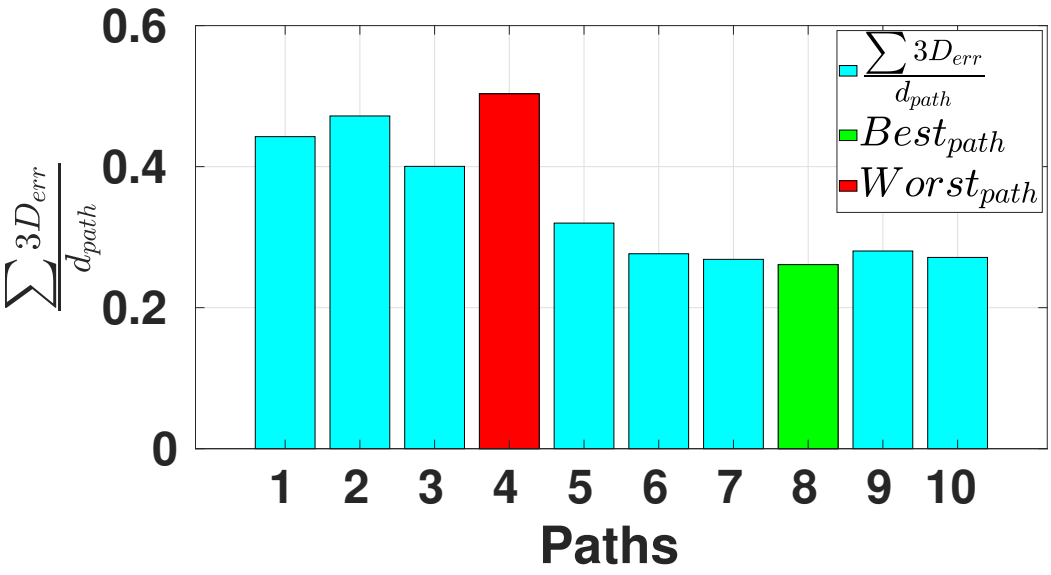


Figure 6.2.9: GP best and worst path for the dark environment with the unmodeled obstacle.

Figure 6.2.10, along with Table 6.2.9, show the comparison between the BSP and the GP related to the EKF localization error. In this case, the GP prediction does not return the best path, but the second-best being both very close to each other with less than one cm in RMS difference.

Table 6.2.8: GP algorithm results in lit environment for the prediction of the 10 paths with the addition of an obstacle. In red the one that performs worse compared with the one in green that which minimizes the localization error.

Path	Median	Mean	RMS (Pos. Err.)	Sum. Err
1	0.24	0.44	0.93	0.44
2	0.25	0.47	0.83	0.47
3	0.24	0.40	0.72	0.40
4	0.24	0.50	0.97	0.50
5	0.24	0.32	0.46	0.32
6	0.24	0.28	0.35	0.27
7	0.24	0.27	0.34	0.27
8	0.23	0.26	0.33	0.26
9	0.24	0.28	0.36	0.28
10	0.22	0.27	0.37	0.27

Table 6.2.9: Comparison between the simulation of the best path according to GP-based planner and the best according to BSP in a dark environment and unmodeled obstacle.

	GP Best Path	BSP Best Path
Flight time (s)	164.4	169.4
RMS 3D Pos. Err. (m)	0.42	0.58
σ (m)	0.25	0.40
μ (m)	0.34	0.41
Median (m)	0.27	0.31
Max Pos. (m)	1.43	2.35

6.3 GP Training Using LiDAR Solution

The previous section showed the potential of the GP algorithm in choosing the path that best localizes the UAV when given a true source to train the models. Because it was using the absolute

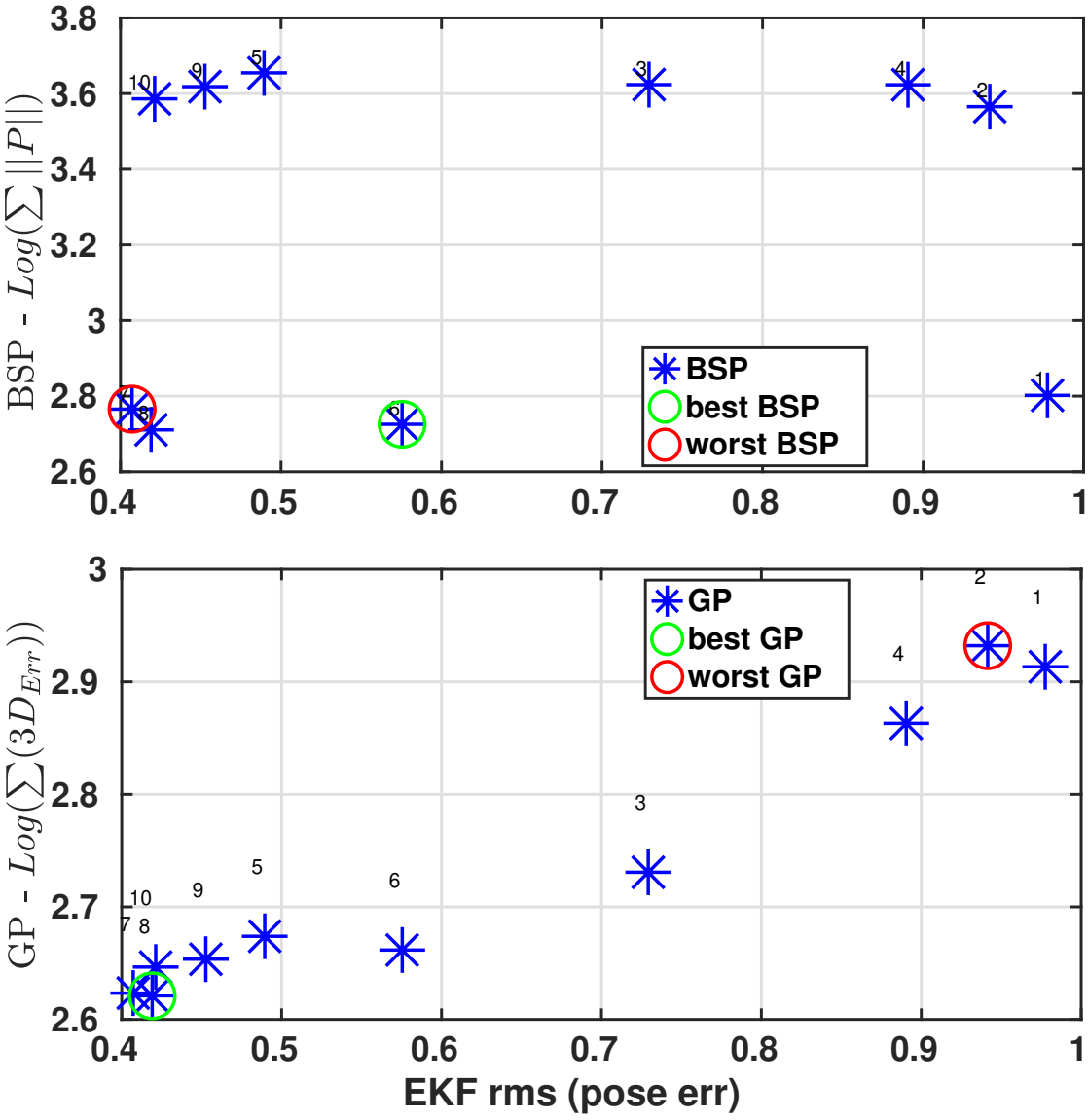


Figure 6.2.10: The BSP sum of the norm of the position error covariance matrix, related to the localization filter position error estimate, are compared with the GP planning algorithm sum of the prediction error related to the EKF position error estimate for the dark environment test and the added non-mapped obstacle.

ground truth (e.g. a motion capture system) as one of the data for the GP training process, the practicality of applying this approach in any kind of environment is reduced.

The previous results were conducted as an illustrative example to show the potential of the GP algorithm. This section shows how this kind of approach can be implemented in any environment with some grade of accuracy without being linked to a motion capture system. Such an environment could be represented by an underground scenario (or a warehouse) where the ground robot is used to provide enough data for the UAV to be able to better localize itself in future missions.

The main goal is to use the reliable position measurement from the LIDAR as the reference to compare the EKF position estimation which is driven by sensors that are prone to environmental degradation (e.g., VIO due to lighting). This could be realistically implemented by having the UAV manually flown in the environment to collect data, to then be used for the GP training, before the optimal path is chosen by the planning algorithm.

During this test, the EKF does not fuse the LIDAR measurements during the UAV position estimation and the results are shown in Table 6.3.1 and Figure 6.3.1.

Table 6.3.1: GP algorithm results in dark environment for the prediction of the 10 paths using LIDAR solution as ground truth. In red the one that performs worse compared with the one in green that which minimizes the localization error.

Path	Median	Mean	RMS (Pos. Err.)	Sum. Err
1	3.32	3.40	3.77	3.40
2	7.22	7.94	8.27	7.94
3	2.46	2.82	3.32	2.82
a 4	7.23	8.09	8.53	8.09
5	7.59	8.14	8.59	8.14
6	6.85	7.32	7.64	7.32
7	2.46	2.69	2.95	2.69
8	6.63	7.04	7.52	7.04
9	6.81	7.57	8.16	7.57
10	3.46	3.67	3.98	3.67

As show in the previous table, the overall UAV's position error increases but the GP-based planning algorithm is capable to pick the path that minimizes it.

6.4 Concluding Remarks

This chapter introduced a different and more robust path planning algorithm that leverages real sensors data to discard those pre-selected paths that could poorly affect the UAV's localization. Compared to the BSP algorithm, this algorithm better perform overall in choosing the path that minimize the UAV's localization error, but especially in those scenarios where sensors would provide less reliable measurements.

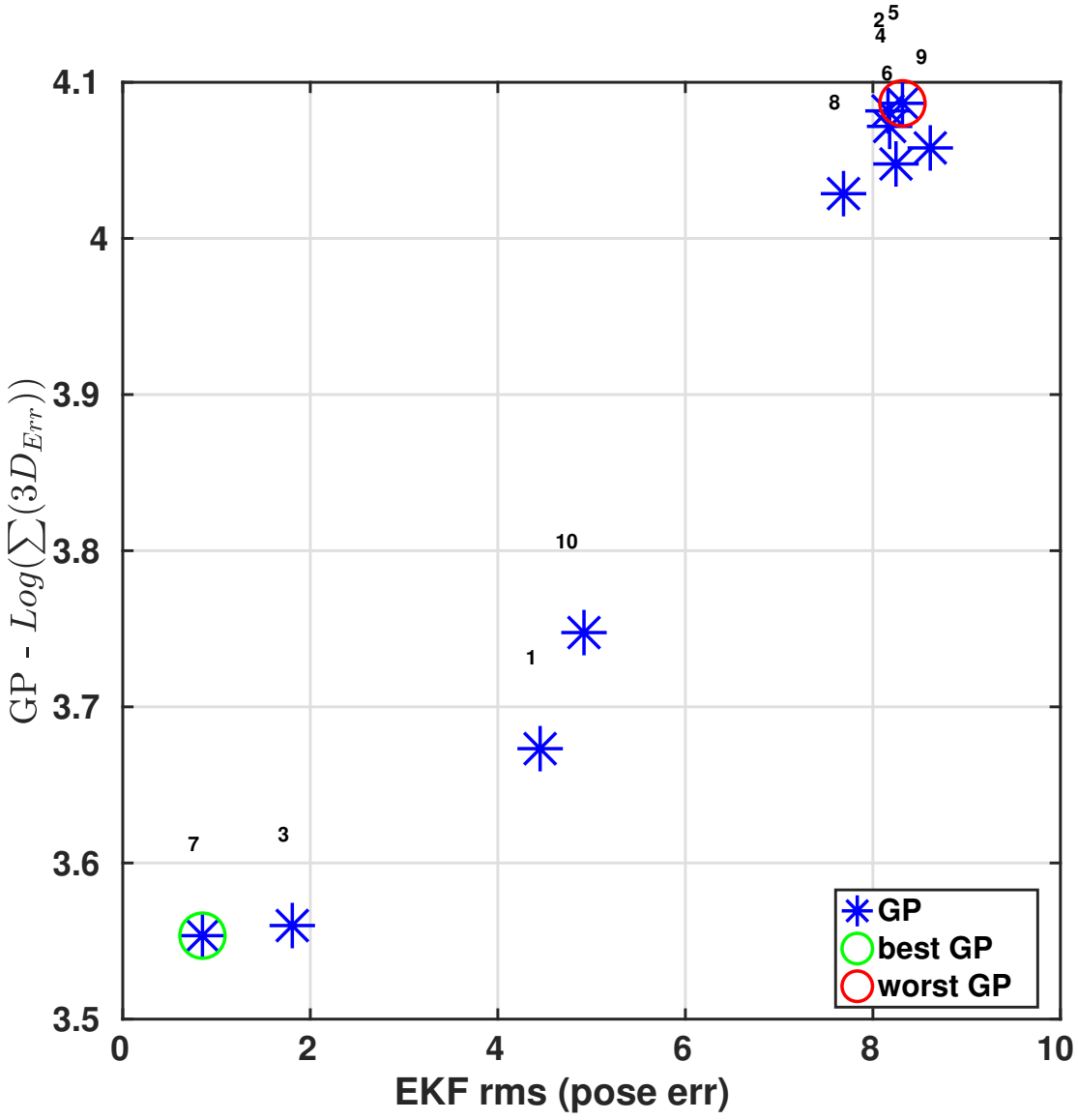


Figure 6.3.1: GP planning algorithm sum of the prediction error when the LiDAR solution is used as "ground truth" during the training process. It is related to the EKF position error estimate for the dark environment test.

7

Concluding Remarks

7.1 Discussion

This work presented the problem of cooperative navigation between two robots in a harsh environment. Different tools have been developed to improve the robot's autonomy and a complete stack was assembled to collect extensive data for the development of the navigation filter described in Chapters 3 and 4. This process delivered a team of two robots cooperating, where the main

computing power is processed on the ground rover and it also provides the UAV pose feedback necessary for the flight controller to perform waypoints navigation. Among many challenges, the main lesson learned consists in how important is the interpretation of the sensor measurements, their processing, and consequently their communication such that the UAV is able to benefit from them.

Both a hardware realization and a simulator were developed to test the UAV behavior in the real world scenario and especially to test the new algorithms described in chapters 5 and 6. This simulator was also extensively used to improve the navigation filter to perform waypoint navigation.

With the BSP algorithm, each path is associated with the total uncertainty that the drone would face flaying that specific order of waypoints. Since the waypoints are randomly chosen, the order they are traveled can substantially impact the drone localization error. As mentioned in chapter 5, some of the sensors, used to localize the UAV, provide less reliable measurements than others. For this reason, when the drone starts the path flying in the field of view of a less reliable sensor, the estimated position is affected by the propagation of a larger error. As this algorithm is able to recognize which is the order of waypoints that minimize the localization error of the UAV, it is also fully dependent on how accurate each sensor model is for the calculation of the error propagation. This being the major downside of this algorithm, there are a few others that are important to mention. First, the algorithm is developed to work offline, and second even a very accurate sensor model would not consider the environment condition. For this reason, the GP algorithm was developed to incorporate the sensor behavior, in a specific environment, in the decision-making process.

The GP planning algorithm results to be more robust than the BSP since it uses the real sensors data to predict their model. This allows the algorithm to associate a more realistic error prediction

to each position of the pre-selected paths. This way the algorithm can predict the sensor performances, and its degradation due to the environment. Like the previous algorithm, this also has some downsides that are important to mention. The main issue consists in the collection of the data necessary to train the GP. Also, at this moment the algorithm is dependent on the same environment where the training data are collected. However, the LiDAR training version shows the potential of using a subset of sensors to train the error model of others.

7.2 Future work

Like in any research work, the one presented in this dissertation can be extended and further improved. Future work can focus on two main fronts, first, improving the presented planning algorithms to be more generic and applicable in different scenarios, second improving the motion of the ground rover to better assist the UAV.

Since the presented algorithms are designed to work offline, a major improvement consists in making them capable of re-planning the route online or even adding more waypoints to the route. If a sub-optimal solution is accepted, each algorithm could predict the best solution on a shorter horizon before moving to the next.

More specifically for the GP algorithm, instead, the training process could be performed on a smaller set of data, such that the prediction would involve just a subset of the waypoints. This approach could transform the presented algorithm into an "almost online" version.

Further improvements consist of adjusting the navigation filter to take into account UGV movements while providing the UAV localization. This way the UGV movement could improve the environment map or it could position itself to keep the UAV always in the sensor's field of view.

These are just a few of the multiple routes that the work in this dissertation could be the base of.

Bibliography

[1] Jaka Katrasnik, Franjo Pernus, and Bostjan Likar. A survey of mobile robots for distribution power line inspection. *IEEE Transactions on power delivery*, 25(1):485–493, 2009.

[2] RP Preston and J Roy. Use of unmanned aerial vehicles to supplement conventional investigation methods for underground open void stability and mitigation. *Underground Mining Technology*, pages 609–616, 2017.

[3] Janosch Nikolic, Michael Burri, Joern Rehder, Stefan Leutenegger, Christoph Huerzeler, and Roland Siegwart. A uav system for inspection of industrial facilities. In *2013 IEEE Aerospace Conference*, pages 1–8. IEEE, 2013.

[4] Carlos Sampedro, Alejandro Rodriguez-Ramos, Hriday Bavle, Adrian Carrio, Paloma de la Puente, and Pascual Campoy. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *Journal of Intelligent & Robotic Systems*, 95(2):601–627, 2019.

[5] Aaron Morris, Dave Ferguson, Zachary Omohundro, David Bradley, David Silver, Chris Baker, Scott Thayer, Chuck Whittaker, and William Whittaker. Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57, 2006.

[6] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, D Montemerlo, Aaron Morris, Zachary Omohundro, et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91, 2004.

[7] Faris Azhari, S Kiely, C Sennersten, C Lindley, M Matuszak, and S Hogwood. A comparison of sensors for underground void mapping by unmanned aerial vehicles. *M. Hudyma und Y. Potvin (Hg.): Underground mining technology. Sudbury, Australia: Australian Centre for Geomechanics. Online verfügbar unter https://papers. acg. uwa. edu. au/d/1710_33_ Sennersten/33_Sennersten. pdf, zuletzt geprüft am*, 4(10):2018, 2017.

[8] Christos Papachristos, Shehryar Khattak, Frank Mascarich, and Kostas Alexis. Autonomous navigation and mapping in underground mines using aerial robots. In *2019 IEEE Aerospace Conference*, pages 1–8. IEEE, 2019.

[9] Christos Papachristos, Frank Mascarich, Shehryar Khattak, Tung Dang, and Kostas Alexis. Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning. *Autonomous Robots*, pages 1–31, 2019.

[10] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. Coordination for multi-robot exploration and mapping. In *Aaai/Iaai*, pages 852–858, 2000.

[11] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *ICRA*, pages 476–481, 2000.

[12] Jason Gross, Matteo De Petrillo, Jared Beard, Hayden Nichols, Tommy Swiger, Ryan Watson, Connor Kirk, Cagri Kilic, Jacob Hikes, Emily Upton, Derek Ross, Matt Russell, Yu Gu, and Christopher Griffin. Field-testing of a uav-ugv team for gnss-denied navigation in subterranean environments. In *Proceedings of the 32nd Annual International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+)*, pages 2112–2124. ION GNSS+, 2019.

[13] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan, and Z. Ming. A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, Dec 2016.

[14] Hailong Qin, Zehui Meng, Wei Meng, Xudong Chen, Hao Sun, Feng Lin, and Marcelo H Ang. Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments. *IEEE Transactions on Vehicular Technology*, 68(2):1339–1350, 2019.

[15] Jeffrey Delmerico, Elias Mueggler, Julia Nitsch, and Davide Scaramuzza. Active autonomous aerial exploration for ground robot path planning. *IEEE Robotics and Automation Letters*, 2(2):664–671, 2017.

[16] Francesca Baldini, Animashree Anandkumar, and Richard M Murray. Learning pose estimation for uav autonomous navigation andlanding using visual-inertial sensor data. *arXiv preprint arXiv:1912.04527*, 2019.

[17] Roberto G Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Ström, and Jizhong Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 5233–5238. IEEE, 2014.

[18] Kamak Ebadi, Yun Chang, Matteo Palieri, Alex Stephens, Alex Hatteland, Eric Heiden, Abhishek Thakur, Nobuhiro Funabiki, Benjamin Morrell, Sally Wood, Luca Carlone, and Aliakbar Agha-mohammadi. Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 80–86, 2020.

[19] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021.

[20] Nicolas Hudson, Fletcher Talbot, Mark Cox, Jason Williams, Thomas Hines, Alex Pitt, Brett Wood, Dennis Frousheger, Katrina Lo Surdo, Thomas Molnar, et al. Heterogeneous ground and air platforms, homogeneous sensing: Team csiro data61's approach to the darpa subterranean challenge. *arXiv preprint arXiv:2104.09053*, 2021.

[21] Sachin Patil, Gregory Kahn, Michael Laskey, John Schulman, Ken Goldberg, and Pieter Abbeel. Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Algorithmic foundations of robotics XI*, pages 515–533. Springer, 2015.

[22] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *The International Journal of Robotics Research*, 34(7):849–882, 2015.

[23] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

[24] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on robotics and automation*, 14(1):166–171, 1998.

[25] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[26] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014.

[27] Ali-Akbar Agha-Mohammadi, Saurav Agarwal, Aditya Mahadevan, Suman Chakravorty, Daniel Tomkins, Jory Denny, and Nancy M Amato. Robust online belief space planning in changing environments: Application to physical mobile robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 149–156. IEEE, 2014.

[28] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[29] S. Khattak, C. Papachristos, and K. Alexis. Keyframe-based direct thermal–inertial odometry. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3563–3569, 2019.

[30] Shehryar Khattak, Frank Mascarich, Tung Dang, Christos Papachristos, and Kostas Alexis. Robust thermal-inertial localization for aerial robots: A case for direct methods. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1061–1068. IEEE, 2019.

[31] Tyson Govan Phillips, Nicky Guenther, and Peter Ross McAree. When the dust settles: The four behaviors of lidar in the presence of fine airborne particulates. *Journal of field robotics*, 34(5):985–1009, 2017.

[32] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[33] Andrew McHutchon and Carl Rasmussen. Gaussian process training with input noise. *Advances in Neural Information Processing Systems*, 24:1341–1349, 2011.

[34] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[35] Xiaofeng Yang. Nlos mitigation for uwb localization based on sparse pseudo-input gaussian process. *IEEE Sensors Journal*, 18(10):4311–4316, 2018.

[36] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[37] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. 2010.

[38] R.E. Kalman. Contributions to the theory of optimal control, 1960.

[39] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, 1995.

[40] Ryan Watson, Nicholas Ohi, Scott Harper, Cagri Kilic, Chizhao Yang, Jacob Hikes, Matteo De Petrillo, Jared Strader, Gabrielle Hedrick, Hayden Nichols, et al. A rover and drone team for subterranean environments: System design overview.

[41] Px4, "px4/firmware," github, 30-sep-2019. [online]. available: https://github.com/px4/firmware.

[42] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[43] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, 12(2):494–506, April 2015.

[44] Mavlink, "mavlink/mavros," github. [online]. available: https://github.com/mavlink/mavros.

[45] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.

[46] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[47] Agathe Girard. *Approximate methods for propagation of uncertainty with Gaussian process models*. PhD thesis, Citeseer, 2004.

[48] Paul D Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[49] I. Bogoslavskyi and C. Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[50] I. Bogoslavskyi and C. Stachniss. Efficient online segmentation for sparse 3d laser scans. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, pages 1–12, 2017.

[51] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. volume 2, pages 28 – 31 Vol.2, 09 2004.

[52] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, May 2006.

[53] Davide Scaramuzza. *Omnidirectional Vision: from Calibration to Robot Motion Estimation*. PhD thesis, ETH Zurich, 2008.

[54] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[55] Derek Ross, Matteo De Petrillo, Jared Strader, and Jason N Gross. Uncertainty estimation for stereo visual odometry. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pages 3263–3284, 2021.

[56] Jared Strader. wvu-vo-ros, github repository, https://github.com/wvu-navlab/src2-vo-ros/tree/dev-drone, 2020.

[57] Matteo De Petrillo, Jared Beard, Yu Gu, and Jason N. Gross. Search planning of a uav/ugv team with localization uncertainty in a subterranean environment. *IEEE Aerospace and Electronic Systems Magazine*, 36(6):6–16, 2021.

[58] William R. Stewart. *Chinese Postman Problem (CPP)Chinese postman problem*, pages 84–87. Springer US, Boston, MA, 2001.

[59] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.