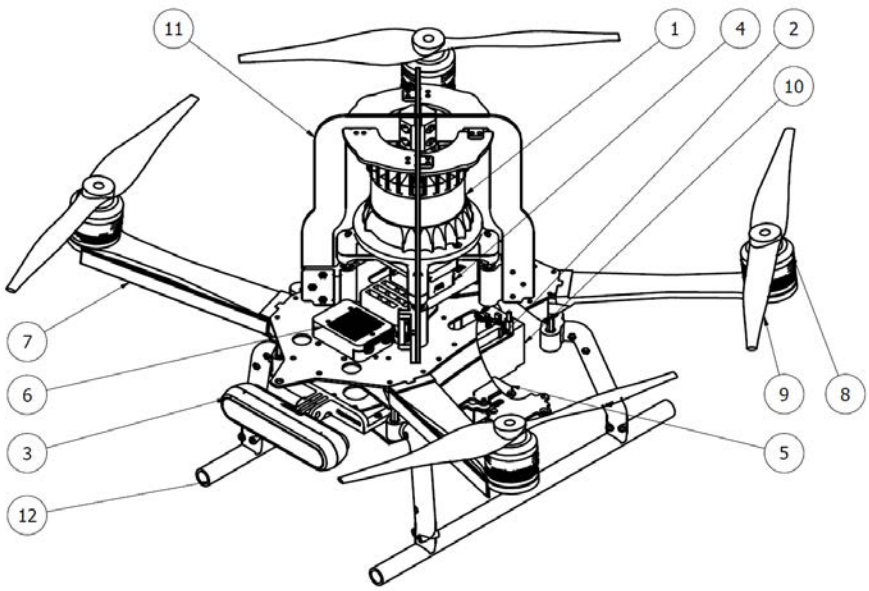
DOCTORAL THESIS

Reactive Navigation Methods for Autonomous Robots: Safety, Coordination, and Field Deployment



Björn Lindqvist

Robotics & AI

LULEÅ UNIVERSITY OF TECHNOLOGY

Reactive Navigation Methods for Autonomous Robots: Safety, Coordination, and Field Deployment

Björn Lindqvist

Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
Luleå, Sweden

Supervisor:

Prof. George Nikolakopoulos

To my friends, family, and to the curious reader

ABSTRACT

In the new era of intelligent systems, small-scale electronics and advanced sensors, the application use-cases and operational capabilities of autonomous robots are exponentially increasing. Through their ability to execute complex tasks, while relying only on onboard sensors and computation, autonomous field robots are showing promising results in inspection of infrastructures, search-and-rescue or surveillance, maintenance tasks, or in general operations in areas that prove to be dangerous or hazardous for human operators to operate, while also often increasing the efficiency of such tasks. But, to enable robots to autonomously execute their missions, the demands on onboard intelligence is increasing rapidly as well. As robot operations move into complex and dynamic environments, into mixed-traffic or multi-robot operational scenarios, or into missions that demand the exploration and navigation of completely unknown areas, a new paradigm of autonomous robot navigation and collision avoidance algorithms need to be developed as well. Towards achieving the vision of autonomous robots performing such tasks for the good of society, this new paradigm of navigation capabilities must first be extended to operate outside of simulation environments, and then to operations in realistic field conditions with all the challenges that comes with that.

This thesis presents the development of a series of navigation methods for autonomous robots, with a specific focus on Unmanned Aerial Vehicles (UAVs). Furthermore, the vision of this thesis is to enhance the application areas of completely autonomous robotic platforms by extending their navigation capabilities: towards avoiding obstacles in their environment both static and dynamic, towards the critical perception-actuation link for reactive navigation, towards exploring and planning dynamic paths through previously unknown areas, and towards the coordination and safety in multi-agent robotic systems. This thesis also has a significant focus in the area of field robotics, meaning the ability to robustify and extend the robots onboard intelligence to handle the harsh conditions of real operations. This thesis will specifically investigate the application of autonomous UAVs in search-and-rescue tasks in subterranean environments, as well as a variety of inspection tasks in underground mines. In these environments the robots must operate completely autonomously without any assisting communication, computation, or perception infrastructure. In all of these areas, a special focus has been placed on the real-life experimental validation of results and the required research to reach the readiness stage of such demonstrations, serving as the main motivator for the works presented in this manuscript.

CONTENTS

ACKNOWLEDGMENTS

First I would like to thank Prof. George Nikolakopoulos who has been my supervisor throughout my time as a PhD Student. George has a unique ability to push the readiness level of our frameworks through a significant focus on novel and difficult applications and field deployment, while also allowing large self-motivation and personal development through the adoption of responsibilities in progressing the work. I had a great time with George as my supervisor, and I look forward to any future collaborations.

Second I would like to extend a special ”thank you” to our research engineers Jakub Haluska and Ilias Tevetzidis. Focusing my work in both collision avoidance and field robotics, I can not count the number of times that they have had to repair broken quadrotor arms (or worse) after a crash, or have had to come up with quick fixes for various hardware problems and challenges in the field throughout the years.

Third, a huge ”thank you” to all the collaborators in the Robotics and AI Group at Luleå University of Technology. It was a lot of fun working with you all, especially in all the experiment and field trials that we travelled to and successfully executed together. We really did achieve great results together.

Finally, I extend a ”thank you” to my partner, Ellinor Emilsson, who has been very patient and understanding of my obsessions and long absent hours during the PhD process, as well as to my friends and family for their support.

CHAPTER 1

Introduction

1.1 Terminology

The following offers short explanations of some acronyms and commonly used terminology as to have a common ground with the reader.
Reactive Navigation - ”Navigating as you go” and constantly updating the trajectory - closely coupled with environment information, and executed at high frequencies. Similar concepts are sometimes denoted as ”Memory-less” navigation.
Autonomy - A systems ability to perform tasks without human input during execution. ”Fully autonomous” in this thesis mainly refers to a system not relying on external infrastructure for localization, computation, or perception.
Robot Platform - The baseline robot, without the sensors and algorithms.
Sensor Suite - The added sensors to the robot platform that allows it to sense its environment and enables autonomy.
Robot perception - Through its sensor suite a robot can sense its world around it. This term refers to the combination of the sensor data and the processing of that into useful information.
UAV - Unmanned Aerial Vehicle, often denoting flying robots in general.
MAV - Micro Aerial Vehicle, in this thesis used interchangeably with UAV but often denotes smaller scale craft.
Rotorcraft - An aerial vehicle that generates lift by vertically mounted propellers or rotors as opposed to fixed-wing aircraft that generates lift from its airfoil.
Quadrotor - a rotorcraft with four rotors. This is the platform of choice for the majority of this thesis, and when the term UAV is used it almost always refers specifically to a quadrotor. Four rotors provide four control surfaces which gives six degree-of-freedom control.
Wheeled Robot - Any classic car-like ground robot platform using differential or articulated steering.
Legged Robot - Ground robots using legs for locomotion. Bipedal and quadruped motion often mimicking biological systems. Ex. Boston Dynamics Spot, which is the legged platform of choice in this thesis.

Actuator - The controllable physical system that provides (in the robotics context) motion of the system. Servo-motors, propellers, powered wheels.
Controller - The mathematical entity that through automatic control theory drives a dynamical system to its desired state through some actuator.
Actuation - The input signals and resulting actuator action generated by the controller.
MPC - Model Predictive Control, a finite-horizon optimization that considers the predicted future states of a system, and often constraints, in its control objective.
`OpEn` - Short for the Optimization Engine, an open-source optimization software for nonlinear non-convex optimization, used heavily in this thesis for nonlinear MPC applications.
Optimizer - In this thesis referring to the piece of code and mathematics that solves (finds the minimum of) a specified problem.
Obstacle - A general term referring to an area the robot should not enter.
Occupancy Map - A pixel or voxel map where discrete areas or volumes (often boxes/squares) are set to free, occupied, or unknown based on the robots current knowledge of the area.
Robot Exploration - Navigating through an unknown environment often with the goal of maximizing new map information.
Information Gain - The discrete evaluation of new map information from executing some maneuver. If the map is voxelized the information gain can simply be the number of voxels set from an unknown to a known state.
APF - Artificial Potential Field, an old-school very effective approach to robot navigation where a repulsion to obstacles is combined with an attraction to a state reference.
ERRT - Exploration-RRT, a proposed tree-based method for robot exploration-planning fundamentally centered around RRT (Rapidly-Exploring Random Trees).
COMPRA - COMPact Reactive Autonomy, a proposed complete autonomy kit for rapid exploration mission execution in subterranean tunnel environments.
RIA - Routine Inspection Autonomy, a proposed complete autonomy kit for general multi-waypoint inspection missions in known environments.
Subterranean - Referring to any underground environment: caves, metro stations, underground mines, construction and infrastructure tunnels, and anything inbetween.
LiDAR - Light Detection And Ranging, a method of targeting a surface with a laser and measuring the distance/range to it by the time of the returning refracted light.
2D LiDAR - By rotating a laser, 2D coordinates of the laser hits can be estimated from the angle at which it was shot out and the range estimate.
3D LiDAR - A sensor very highly used in this thesis. Many lasers in a rotating array can provide 3D *x-y-z* coordinates of the surfaces hit by the laser beams. This can provide 3D perception and mapping for a 3D LiDAR equipped robot.
RGB-D Camera - Red-Green-Blue-Depth cameras provide a range/depth estimation of pixel coordinates in addition to color through stereo-vision.
IMU - Inertial Measurement Unit, a combination of various ”internal” sensing technologies such as accelerometers, gyroscopes, magnetometers, and pressure sensors.
SLAM - Simultaneous Localization And Mapping, an algorithm for estimating a robot position based on a procedurally generated map of the environment from onboard sensors.
LIO - LiDAR-Intertial Odometry, a sensor-fusion of slower LiDAR feature tracking and map-

based loop-closure with fast IMU predictions for robot pose estimation. This thesis heavily relies on LIO from 3D LiDAR SLAM.
FCU - Flight Control Unit, a micro-computer onboard an aerial vehicle responsible for attitude control, motor actuation, and often equipped with an IMU and GPS.
ROS - Robot Operating System, a middleware for message handling developed specifically for robotics applications. All presented frameworks in this thesis are utilizing ROS for inter-module communication and hardware-software integration.

1.2 Autonomous Field Robots and their Applications

The concept of constructing a machine that can autonomously execute a programmed task has occupied the human mind for thousands of years, even dating back to Homer. While historical claims of ancient actual constructed "robots" are often dubious, industrial autonomous machinery revolutionized manufacturing a long time ago now. Through the innovation of *automatic control* such systems could start to interact with the world through feedback control, as opposed to fully pre-configured execution. In the modern world, we have access to miniaturized hardware and smart electronics that have given rise to new robotic platforms, such as robotic manipulators, wheeled ground vehicles, aerial vehicles, and legged robots both quadruped and humanoid, all with revolutionary capabilities in their application domains. The last piece of the puzzle lies in the development of robotic sensing technologies, or *robot perception* capabilities, that allows a system to sense its environment. This is happening not only by just reading a temperature or pressure gauge, but sensing technologies that can be placed onboard a robot that allows it to build an understanding and reconstruction of the world around it in real time, such as cameras, LiDARs, and Radars, and their related perception-layer algorithms. The robots' perception can then be linked to the navigation and control systems, enabling the modern era robots to execute missions in unstructured or dynamic environments - and for more abstract missions, such as the exploration of previously unknown areas. Research into this type of modern, mobile, and agile robots, received very high interest from the academic world from a multidisciplinary perspective, combining computer science, mathematics, automatic control, mechatronics, technical design, and computer vision & perception, into the development of fully autonomous robots.

When discussing the meaning of an autonomous robot, often one will find notions of multiple levels of autonomy. But there is no doubt that the autonomous systems that are fully self-sufficient when it comes to perception and computation, and whose software stack allows them to execute complex missions without operator assistance and in the real world, and for the good of society, holds a special place. These *autonomous field robots* are now deployed for a variety of tasks in many environments. These range from the inspection of large-scale infrastructure [1–3], disaster management missions [4], search and rescue missions [5], including the delivering of first-aid devices in case of an accident [6, 7], monitoring of forest fires [8, 9], inspection of mining tunnels [10], exploration and 3D reconstruction of cave systems [11], crowd surveillance and monitoring [12, 13], ware-house logistics [14, 15] and with a large potential in the near future for example in the construction industry [16, 17]. There are common themes in all these types of missions that are of a high interest to the related industry, where either

the autonomous robots can perform the task with higher efficiency, repeatability, and lowest cost, or they can execute tasks in inherently hazardous environments and as such increase the safety of the operator. When it comes to the field application areas, the works in this thesis will have an overarching theme of deployment into subterranean environments. Here, autonomous robots have huge application possibilities for increasing the safety of operations, performing rescue missions, and entering unsafe areas or exploring inaccessible ones. Towards that direction, the research contributions in this thesis will try to extend the navigation capabilities of autonomous systems towards robust collision avoidance and safe navigation, and towards efficient exploration and deployment capabilities.

1.3 Robot Navigation

Robot navigation in general describes a set of autonomy modules that when working together allow the robot to move through its environment in a safe and efficient way, while also fulfilling its mission demands. It is very rare that the navigation system on a fully autonomous robot is handled by a single algorithm, simply due to the considerable complexity of the problem. An example of a generalized architecture for robot mission execution and navigation can be seen in Figure 1.1. There is of course no requirement that all of these modules are present for a robotic mission, and the exact autonomy architecture is highly dependant on what the mission and environment requires, but this example architecture can serve as an initial point for discussion.

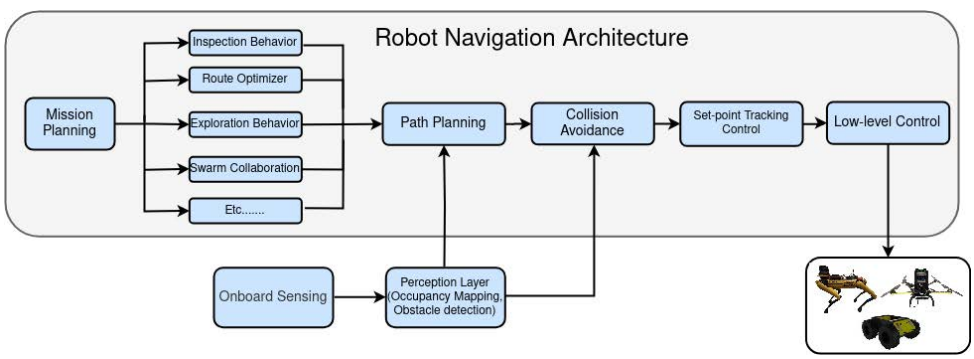


Figure 1.1: An example navigation architecture for a robotic mission, going from low level actuator control up to high level mission planning.

Starting from the lowest level (closest to the robot actuation) and working our way up, the lowest level generally relates to developing controllers that are tightly coupled with the dynamics and actuators of the system. Low level control is of course highly platform dependant, and while wheeled ground robot actuation can be relatively simple, quadruped [18] or bipedal [19] low level actuation control is very challenging and a highly active research subject, although

it will not be investigated in this thesis. We denote the next level as set-point tracking control, referring to controllers responsible for tracking position, orientation, and velocity set-points, or full trajectory tracking controllers [20] for agile transitioning between set-points. Set-point reference tracking is required if any robot platform is to be linked with the higher level navigation modules, and its performance greatly influences the performance of higher level modules. This thesis will particularly discuss the coupling and merging of set-point tracking control and collision avoidance for high performance maneuvering.
Robust collision avoidance is critical for any robot mission, and it is the main subject of interest in this thesis. In general, collision avoidance algorithms are responsible for maintaining a safe distance from obstacles, other vehicles or humans, and in general from an encountered dynamic environment. The demands on collision avoidance algorithms are mainly towards higher run-time requirements for fast reactions, a tight coupling with the set-point tracking controller, and the ability to ensure robot safety, as for many robot platforms (especially aerial) any interactions or collision with the environment can result in the end of the mission and a broken robot. Among the active problems in collision avoidance are the robustness and safety guarantees [21], and how to couple advanced collision avoidance algorithms with the perception systems of the robot. Robot path planning is a quite general subject, but in Figure 1.1 we are mainly talking about occupancy-based or grid-based path planners like A^* [22] or Rapidly-exploring Random Trees (RRT) [23], whose goal is to plan a path from the current estimated position of the robot to a desired goal in a known or partially known map of the area. The focus often lies on the speed of computation for finding the shortest or optimal path, and the traversability or robot safety along the generated path.
The next level is more diverse, and we can denote them as *mission specific modules*, as they are specific to the exact mission the robot is to execute. Some examples are provided in this Figure, such as generating inspection behavior and coverage around infrastructure, optimizing a route when visiting many way-points or task assignment for multi-agent systems, algorithms for exploring an unknown space, swarm behavior, and many more. All these components define what mission the robot is performing, and from all of them the major focus in this thesis will be on the exploration missions, but other topics will be briefly addressed as well. On the highest level we have the mission planning, responsible for higher level coordination (ex. behavior trees [24]) or for combining multiple mission-specific modules for more complex missions. A simple example of that could be to explore until a target is found, switching to an inspection behavior to completely inspect it with a sensor/payload, and then return-to-base behaviour with a grid-based path planner. The last chapter of this thesis will touch on this subject related to field work, such as multi-robot collaboration, inspection mission configuration, as well as return-to-base and guided landing after an exploration mission is completed.
Each autonomy architecture needs to be tailored to the mission and environment - e.g. a warehouse robot that is following pre-defined routes to deliver objects might only need a mission planner, simple collision avoidance, and controllers, while completing a collaborative search-and-rescue mission in an unknown harsh environment could demand high performance of all levels of navigation systems. In the introduction sections of the following related chapters, we will go into more details into the state-of-the-art and the challenges of each navigation sub-system that this thesis will touch upon.

1.4 Thesis Contents and Contributions

The contributions of the research contained in this thesis can be summarized as: the development of novel navigation methods for autonomous robots that focus on collision avoidance both inside multi-agent systems and of obstacles in the environment, the critical perception-actuation link for real-world navigation in unknown environments, as well as both reactive and sampling based exploration algorithms for robot exploration. Overall, this is achieved through the development of three separate frameworks, and a major field study: 1) a Nonlinear MPC framework based on the optimization engine [25] where set-exclusion constraints form the basis of collision avoidance and local path planning, 2) an artificial potential field (APF) implementation that focuses on complete reactivity by directly using 3D LiDAR pointclouds for local obstacle avoidance and navigation purposes, 3) the development of the ERRT algorithm where a tree-based method is developed for a complete solution to the combined exploration-planning problem in unknown and unstructured environments, 4) a significant field study in the deployment of autonomous robots in subterranean environments, such as underground mines - using the developed COMPRA (COMPact Reactive Autonomy) framework (also centered around the reactive APF and the fully reactive navigation concept), and the RIA (Routine Inspection Autonomy) framework that combines local APF navigation with a risk-aware path planner, route optimization, and global relocalization. These contributions and developed frameworks can be fit into the general architecture in Figure 1.1, and we highlight the contributions of the components of this thesis in an updated visualization in Figure 1.2.

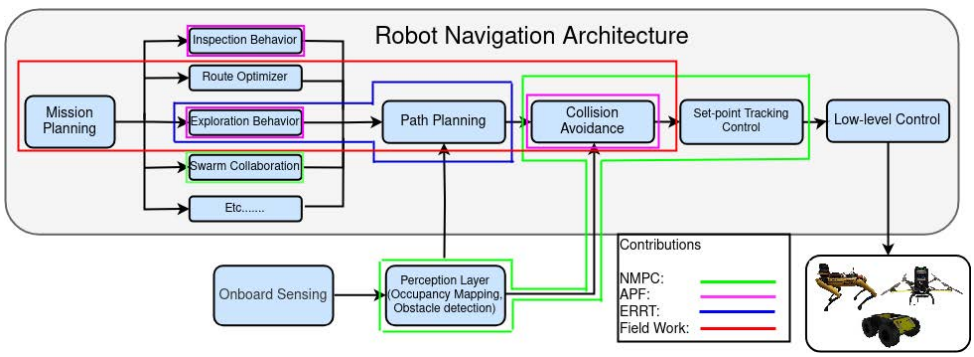


Figure 1.2: Highlighting the contributions of the thesis. Green highlights the works on NMPC for collision avoidance, magenta highlights the works on reactive APFs, blue highlights the works on exploration-planning with ERRT, and red highlights the field work on navigation and mission execution in subterranean environments.

The following bullets summarize the research contents of the thesis in the order they can be found in the thesis:

- A reactive NMPC-based navigation method where obstacles of different types are detected by a 2D LiDAR, and then translated into set-exclusion constraints on the available

position-space of the robot. As the collision avoidance is part of the control system, the result is a local avoidance planner that fully takes the dynamics of the system into consideration. Results prove real-time applications in laboratory environments using onboard hardware for object detection and computation.

- The application of NMPC to the avoidance of dynamic obstacles. Due to the already predictive nature (as the name implies) of MPC we can fit the consideration of dynamic obstacles into the optimization problem by providing a prediction of future obstacle positions that define set-exclusion constraints both in position and predicted time. The scheme is thoroughly evaluated in laboratory experiments on both UAVs and legged robots, where the legged robot case is demonstrated in the context of safe human-robot interaction in a "track & avoid" architecture.

- Research into using the predictive nature of NMPC for safe trajectory orchestration in multi-agent systems. This is demonstrated both in a centralized approach where one central computational agent at each execution step optimizes trajectories for the whole system, and in a distributed formulation where predicted trajectories are shared among agents in the system and those predicted trajectories are used to form avoidance constraints for the ego-agent along the prediction horizon. All these are demonstrated both in simulation and laboratory experiments for real-time applications of up to ten aerial agents.

- The development of a 3D LiDAR-based Artificial Potential Field, where fail-safe navigation is achieved by removing a reliance of a perception layer and instead repulsive forces are generated directly from the raw pointcloud. The artificial potential field was also combined with a NMPC, and a Lidar-Intertial SLAM framework [26], to enable a complete kit for local autonomy, and fit to a specific sensor and computation hardware. This framework was an enabler for research in multiple directions as an easy-to-use and fail-safe local layer of navigation that any other module could be placed on top of.

- Further investigations into reactive 3D LiDAR APFs, trying to push the perception-layer-free navigation into the use-cases of reactive exploration and infrastructure inspection. Additionally, the framework was evaluated also for collision avoidance of multiple UAVs operating in the same local space.

- The development of a 3D local exploration-planning algorithm: ERRT. The algorithm is based on a tree-based random search, where specific sampled trajectories are analyzed in terms of path length, information gain, and model-based actuation required to follow the trajectory (solved as a NMPC problem). The framework was coupled with a state-of-the-art 3D occupancy mapper [27] to enable hardware integration with 3D LiDAR pointclouds. The framework was evaluated in both large-scale simulations and in real-world experiments in subterranean environments.

- The COMPRA (COMpact Reactive Autonomy) Kit for subterranean search and rescue operations. COMPRA was developed to enable quick deployment, and fast as well as

reliable navigation through tunnel environments enabled by reactive modules for obstacle avoidance and reactive tunnel exploration (the depth-camera based Deepest-Point Heading Regulation method). The kit also includes a pipeline for object detection and localization. The COMPRA kit was evaluated extensively in real-world subterranean and mining environments.

- A multi-modality robot deployment framework where a legged robot is combined with an UAV, capable of complex mission execution where the modalities of both robots are leveraged to efficiently execute a search-and-rescue mission in a subterranean tunnel environment.

- A significant investigation into how autonomous UAVs can be used to execute a variety of inspection tasks in mining environments with the long-term and large-scale goal of removing human workers from hazardous inspection tasks such as gas monitoring and visual inspection after blasting, 3D re-mapping and inspection after a rockfall, or in general tedious and time-consuming routine inspection or mapping tasks for maintaining a digital twin of the mine and to ensure areas are safe for workers to enter.

We can very compactly define the goals of the thesis from the perspective of novel navigation algorithms, based on these contributions as: development and demonstration of 1) robust collision avoidance relying on onboard obstacle detection and in multi-agent systems using NMPC, 2) reactive collision avoidance relying only on raw sensor data with an artificial potential field, 3) a robot exploration algorithm, ERRT, for combined exploration-planning using occupancy maps, and 4) reactive exploration methodologies where direct sensor data generates exploration behavior in subterranean tunnels.

1.5 Summary of Published Works

The following thesis chapters are compilations and summaries of a series of published or submitted conference papers and scientific journals in the field of automatic control and robotics. In short, the manuscript is based around works published at the following conferences: The European Control Conference, The International Federation of Automatic Control World Congress (IFAC), The International Conference on Intelligent Robots and Systems (IROS) (3 papers), the Mediterranean Conference on Control and Automation (MED) (2 papers), and published work at the following scientific journals: The Robotics and Automation Letters (RA-L), The Transactions on Control Systems Technology (TCST), The Journal of Intelligent Robotic System (JINT), The Journal of Robotics and Autonomous Systems (JRAS) and IEEE Access. Additionally, submitted works currently under review to: The Transactions on Robotics (TRO) (under revision), and Expert Systems with Applications (ESWA). All published and submitted works that the thesis chapters are based on can be found in the list below:

- Lindqvist, Björn, Sina Sharif Mansouri, and George Nikolakopoulos. ”Non-linear mpc based navigation for micro aerial vehicles in constrained environments.” 2020 European Control Conference (ECC). IEEE, 2020.

- Lindqvist, Björn, et al. "Collision Free Path Planning based on Local 2D Point-Clouds for MAV Navigation." 2020 28th Mediterranean Conference on Control and Automation (MED). IEEE, 2020.

- Lindqvist, Björn, et al. "Collision avoidance for multiple micro aerial vehicles using fast centralized nonlinear model predictive control." IFAC-PapersOnLine 53.2 (2020): 9303-9309.

- Lindqvist, Björn, et al. "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles." IEEE robotics and automation letters 5.4 (2020): 6001-6008.

- Lindqvist, Björn, et al. "Reactive navigation of an unmanned aerial vehicle with perception-based obstacle avoidance constraints." IEEE Transactions on Control Systems Technology 30.5 (2021): 1847-1862.

- Lindqvist, Björn, Pantelis Sopasakis, and George Nikolakopoulos. "A scalable distributed collision avoidance scheme for multi-agent UAV systems." 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021.

- Lindqvist, Björn, Ali-Akbar Agha-Mohammadi, and George Nikolakopoulos. "Exploration-RRT: A multi-objective path planning and exploration framework for unknown and unstructured environments." 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021.

- Lindqvist, Björn, et al. "Compra: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations." Journal of Intelligent & Robotic Systems 105.3 (2022): 49.

- Lindqvist, Björn, et al. "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue." Robotics and Autonomous Systems 154 (2022): 104134.

- Lindqvist, Björn, et al. "An Adaptive 3D Artificial Potential Field for Fail-safe UAV Navigation." 2022 30th Mediterranean Conference on Control and Automation (MED). IEEE, 2022.

- Karlsson, Samuel, Björn Lindqvist, and George Nikolakopulos. "Ensuring Robot-Human Safety for the BD Spot Using Active Visual Tracking and NMPC With Velocity Obstacles." IEEE Access 10 (2022): 100224-100233.

- Lindqvist, Björn, et al. "Deployment of Autonomous Uavs in Underground Mines: Field Evaluations and Use-Case Demonstrations." Available at SSRN 4374895. (Submitted to ESWA))

- Lindqvist, Björn et al. "A Tree-based Next-best-trajectory Method for 3D UAV Exploration" (Submitted to TRO)

1.6 Thesis Chapters

The rest of the manuscript is structured around four chapters each with their own subject related to autonomous navigation methods, as well as a concluding summary. Chapter 2 will discuss research into NMPC with integrated constraint-based obstacle avoidance in a variety of use-cases from the perception-actuation link, avoidance of dynamic obstacles, and multi-agent coordination, Chapter 3 discusses the development and use-cases for an APF that directly uses raw LiDAR data for UAV navigation, Chapter 4 includes the formulation and implementation of a novel exploration-planning "next-best-trajectory" method, ERRT, for 3D UAV exploration. Chapter 5 will discuss research and development of autonomy for field deployment in two areas: 1) search-and-rescue in subterranean environments, and 2) inspection, mapping, and surveying in underground mines. Finally, a summary of the obtained results and concluding remarks are provided in Chapter 6.

CHAPTER 2

Nonlinear MPC for Obstacle Avoidance

2.1 Overview

No autonomous robot can operate without a robust and high performance control scheme, and no mission can be executed reliably without onboard obstacle avoidance systems keeping the robot safe. In the literature there exists a multitude of solutions to both those problems, but, in general, they are solved separately. That means that the obstacle avoidance system that generates the avoidance maneuver and the controller tasked to execute it are, in essence, not fundamentally synced. Problems can arise with, for example, the feasibility of executing the avoidance maneuver in time to avoid the collision, or with guaranteed safety when the movements are very rapid or the system dynamics are challenging. In this context Model Predictive Control (MPC) is one of the stand-out solutions as the central controller can enable very high performance model-based control, while the addition of constraints in the optimization can represent obstacles or other robots that are to be avoided. Additionally, MPC, as the name implies, generates a predicted trajectory solution that is within the constraints posed by the system dynamics, ensuring that the robot can actually follow the predicted maneuver. The result is a combination and merging of the control and obstacle avoidance systems that also acts as a local path planner for the robot, where the solutions are both optimal (based on the problem formulation) and within the described vehicle dynamics.
This chapter includes the development and implementation of a Nonlinear Model Predictive Control (NMPC) framework for autonomous robot control and collision avoidance. The main focus area is unmanned aerial vehicles (UAVs) but implementations and experiments for both legged robots and ground robots are included as well. The chapter will start by introducing the fundamental aspects of the framework, such as the dynamic prediction model, the cost function and constraints, as well as the utilized optimization (solver). These stay relatively similar throughout the presented works, and the main effort in this chapter will be on the various applications of the baseline developed framework, and the modifications and additions that en-

able those application areas. Three main areas of research are included: 1) Linking the NMPC module with onboard perception systems through connecting the NMPC parametric constraints with detected obstacles in the environment. 2) The ability for the constrained NMPC to handle dynamic obstacles. As MPC schemes are inherently predictive, they are the perfect fit for obstacle avoidance of moving obstacles. 3) Enabling safe multi-agent coordination through vehicle-vehicle collision avoidance using NMPC. The chapter includes both a centralized formulation, where a single computational agent orchestrates trajectories for the whole system, and a distributed formulation where each agent solves its own optimization problem, but the predicted trajectories are shared among agents to enable coordination. In all of these directions, extensive real-world experiments are provided in order to demonstrated the applicability of the framework, and as it will be shown, the developed NMPC can handle all scenarios, while providing smooth and stable collision avoidance maneuvers that maintain robot safety while also keeping computation times low enough for real-time missions.
It should be highlighted that the major focus in this chapter will be the obstacle avoidance and control performance, not the optimization itself. This thesis does not include novel ways to formulate a MPC problem with proved guaranteed feasibility, nor does it include any new method of nonlinear optimization developed by the author. Instead, the developed NMPC framework is an enabler for collision and obstacle avoidance in very challenging scenarios and contexts, and the novelty and contributions come in the form of the problem formulation that fits the NMPC into the proper context, the applications, implementations, the developed assisting modules, and in the significant real-world demonstrations of the framework. In general, the goal is to demonstrate how a constrained NMPC can provide very high performance control and collision avoidance, and how it can be applied in various contexts where collision avoidance is of paramount importance, such as for dynamic obstacles, human-robot scenarios, and in multi-agent systems. As such the following descriptions of system models and optimization in Sections 2.4 and 2.5, as well as the evaluations in Sections 2.6-2.8 directly provides specific examples as opposed to general descriptions, as the specifications need to be included anyway for the application use-cases.

2.2 Introduction

The general ability of a robot to navigate a constrained environment, while avoiding collisions with any obstacles encountered along the way, is one of the key ingredients that enables any fully autonomous mission scenario, and as such, robot path planning and obstacle avoidance is an extensively studied subject [28, 29]. To allow complex mission execution in a dynamic or unknown environments, map-based path planners, such as heuristic grid-search algorithms [30] or sampling based methods [31], provide the high-level paths that should be followed to reach the desired goal (area for inspection, next area to explore, etc.).

While map-based path planning can provide low-risk paths and can incorporate notions of risk-aware behavior [32], it is common to combine them with a layer of reactive collision avoidance, operating at the local level and at a high frequency, and preferably linked more directly to the onboard perception and control system for efficient maneuvering as map-based planners struggle with moving obstacles and fast path updates in case new obstacles are encountered.

For example, in [33] a RRT-based global planner is paired with an APF for UAV navigation.

Comprehensively listing and discussing the benefits and negatives of all popular and novel approaches to collision avoidance is not possible due to the multitude of completely different approaches to the problem, and what type of scenario it is to be used in (multi-agent safety, densely cluttered environment, dynamic obstacles etc.), but the following section can serve as an overview of what types of methods are commonly used, with the overarching theme of discussing how MPC specifically can be applied and provides a benefit in these contexts.

For the classical robot safety problem of maintaining a safe distance from static obstacles the previously mentioned APF [34] is still one of the most common approaches for reactive obstacle avoidance and has been used for the application use-cases for both mapping [35] and exploration tasks [36], as an extra reactive layer for obstacle avoidance, and is the main focus of the next chapter 3. Learning-based methods [37, 38] have also gained significantly in popularity due to the general improvements in neural network architectures, and methods such as the minimum time flight approach in [39] using an onboard LiDAR to detect potential collisions have been demonstrated for avoidance in dynamic scenes. There are also reactive, or local planning, approaches to occupancy-map based planning [40, 41], where by smart partitioning of the map, the computation time can be greatly reduced. Several methods that link the visual information from depth or monocular cameras to the optimization problems have also been attempted [42, 43]. While MPC schemes for high-performance optimal control have been around for some time [44], obstacle avoidance can be integrated into the MPC problem through state constraints that restrict the position-space of the robot [45], thus achieving both model-based optimal control and obstacle avoidance now solved for as one. This approach has been applied to both ground [46] and later also aerial vehicles [47] that naturally have higher run-time requirements for the controller, which ens up as a large limiting factor. In this chapter, one of the main target problem in this area is the link between real-time obstacle detection and classification using onboard sensors and fitting that data into NMPC constraints, all performed using only limited onboard computation, and demonstrated on real hardware in laboratory experiments.

The idea of incorporating moving obstacles explicitly in the local navigation problem has gained significant attention using a variety of methods [48–52]. This allows not only for a reactive but a proactive response to a future collision with a moving obstacle. It is perhaps here that the MPC have the most clear advantage over many other methods. The direct consideration of future states of the robot as compared to the future states of the obstacle through prediction models allows an early response to a future collision, while also allowing real-time solutions that take the robot model and limited actuation into account. Here one of the main questions is how to predict obstacle trajectories [53], and how to possibly guarantee no collision despite limited or uncertain information about the obstacle. There are modern methods for constructing and solving MPC problems that take model and obstacle uncertainty into account; Chance Constraints [54] or Robust Scenario MPC [55] for example. NMPC has been applied in this context also [56], and the critical perception link regarding velocity obstacles has also been studied [57] as part of a Chance-Constrained MPC. The work in this thesis on obstacle avoidance of dynamic obstacles focus on rapidly moving obstacles while keeping avoidance maneuvers smooth, real-time solutions, as well as a trajectory classification scheme. Extensive

laboratory experiments are also performed, even for multiple simultaneously moving obstacles. Additionally, the thesis includes experiments with a legged robot for human-robot safety scenarios where the robot should both track (with a camera) and avoid the quickly moving human obstacle, in a track-and-avoid scenario.

There are many different approaches to collision avoidance schemes for multi-agent robotic systems. Rule-based schemes, such as potential functions [58, 59] are popular here as well, or optimal control schemes [60]. A differential game approach is proposed in [61] with impressive results, but relies on many conditions. In [62] a mixed-integer quadratic program is proposed for centralized trajectory generation, but does not allow for real-time solutions meaning it would be hard to call it a reactive collision avoidance scheme. Other modern solutions showing promise are the barrier functions [63], or *safety barrier certificates* which were evaluated for mini quad-copters in laboratory experiments, and [64] that combined potential-like functions with adaptive control to achieve collision avoidance robust to model uncertainty. MPC-approaches also have an advantage in this context due to their predictive nature. Especially in a distributed scheme [65, 66] agents can share their intended predicted trajectories with other agents in the system that can be leveraged to orchestrate robust and consistent collision-free motion in dense swarms of robots. This thesis presents both a centralized scheme, where we analyze the scalability of the rapidly growing optimization problem, and a distributed scheme that attacks the problem of scalability through obstacle prioritization that uses the shared predicted trajectories. The centralized scheme is evaluated for up to nine agents in simulation, and for four agents in real experiments. The distributed scheme is evaluated for up to ten aerial agents in real-world experiments. The distributed solution is also applied as a safety-coordination layer in a task allocation mission context for wheeled robots.

All the works presented in this chapter are based on the Optimization Engine [25, 67] (for short `OpEn`),which is an open-source code generation software for embedded nonlinear optimization, that is fully ROS-integrated. It generates Rust code, which is very fast and provably memory safe. `OpEn` uses PANOC (proximal averaged Newton-type method for optimal control) [68, 69] combined with either a penalty method [70] or an augmented Lagrangian method [25, 71] to account for general non-convex constraints such as the ones that result from collision avoidance.

2.3 Contributions

This chapter will present a large number of different works but with an overarching theme: the applications of a NMPC framework towards reactive navigation and collision avoidance, with a heavy focus on experimental verification of the frameworks. The goal is to demonstrate how the constrained NMPC can provide an early response to obstacles, generate smooth and stable avoidance maneuvers without unnecessary maneuvering, maintain specified safety distances to obstacles, maintain collision-free motion in dense robot swarms and for rapidly moving obstacles, and that through `OpEn` keep computation times low even with a large prediction horizon. The specific contributions of the included works can be summarized in the following bullets:

- The overall NMPC framework that stays relatively similar throughout the chapter, that includes the prediction models, cost function, constraint formulations, and the optimization formulation as to fit it neatly into the `OpEn` framework. The main benefit of the overall formulation is the merging of model-based set-point reference tracking with the obstacle avoidance.

- Developing the seamless link between onboard perception systems and the parametric NMPC constraints. Using an onboard 2D LiDAR to detect and classify nearby static obstacles forming a framework that that could stand as a stable and properly functional navigation framework, using onboard sensing and computation. We use an outer penalty method combined with the PANOC algorithm to handle the nonconvex constraints that result from collision avoidance.

- Incorporating the concept of obstacle avoidance of dynamic obstacles. Following a similar optimization framework, we add a trajectory classification and prediction scheme and evaluate the NMPC in scenarios of rapidly moving obstacles, and multi-obstacle scenarios.

- Extending the moving obstacle framework to a legged robot that also includes a pipeline for detecting and tracking dynamic moving obstacles with a RGB-D camera in the human-robot safety scenarios. This is demonstrated in completely infrastructure-free experiments relying on onboard sensing, state-estimation, perception, and computation. Here we also apply constraints to keep the moving obstacle in view of an onboard camera to ensure that the robot can continuously see the moving obstacle.

- A centralized NMPC (C-NMPC) for multi-agent trajectory orchestration. The scheme is evaluated in simulations and experiments for dense robot swarms with the goal of maintaining safe and consistent maneuvering without oscillations in attitude, where one central computation agent computes trajectories for all agents.

- A distributed multi-agent scheme that is fully scalable via an obstacle prioritization scheme. Agents share their predicted trajectories at high frequencies with all other agents in the system, and an augmented Lagrangian method is used to solve for non-convex constraints.

2.4 Model and Cost function

A majority of the works presented in this chapter will be towards UAVs, specifically rotorcraft/quadrotors. Changes in the robot model or cost function in the sections that use legged robots or ground robots will be described in the related sections, but we can start by considering an UAV-system as it is used in most of the works included in this thesis. Assuming that the UAV is equipped with an attitude controller, as is often the case [72, 73], a common configuration of actuation commands sent to a rotorcraft are reference angles in roll and pitch, a yawrate command, and a thrust command. Yawing is decoupled from the rest of the actuation

while there is a tight connection between the roll, pitch, and thrust due to the acceleration of the system being produced by tilting the thrust vector via roll and pitch. As such, the UAV model used in this chapter does not include a heading or yaw state, and this state will be controlled by a separate simple PD-controller (Proportional-Derivative). A result of not considering the yaw angle as part of the system dynamics used for the NMPC is that the dynamics are described in yaw-compensated global frame of reference meaning a frame rotated about the heading angle of the UAV. As such, the quadrotor system has six degrees of freedom controlability but the NMPC model only considers five. The nonlinear UAV model is described as follows:

$$\dot{p}(t) = v(t) \tag{2.1a}$$

$$\dot{v}(t) = R(\phi,\theta)\begin{bmatrix}0\\0\\T\end{bmatrix} + \begin{bmatrix}0\\0\\-g\end{bmatrix} - \begin{bmatrix}A_x & 0 & 0\\0 & A_y & 0\\0 & 0 & A_z\end{bmatrix} v(t), \tag{2.1b}$$

$$\dot{\phi}(t) = {}^1\!/_{\tau_\phi}(K_\phi \phi_{\text{ref}}(t) - \phi(t)), \tag{2.1c}$$

$$\dot{\theta}(t) = {}^1\!/_{\tau_\theta}(K_\theta \theta_{\text{ref}}(t) - \theta(t)), \tag{2.1d}$$

where $p = [p_x, p_y, p_z]^\top$ are position states, $v = [v_x, v_y, v_z]^\top$ linear velocity states, both in the yaw-compensated global frame of reference, and ϕ and $\theta \in [-\pi, \pi]$ are the roll and pitch angle states. Moreover, $R(\phi(t), \theta(t)) \in \text{SO}(3)$ is a rotation matrix that describes, in Euler form, the rotation of the thrust vector with $\phi_{\text{ref}} \in \mathbb{R}$, $\theta_{\text{ref}} \in \mathbb{R}$ and $T_{\text{ref}} \geq 0$ to be the references in roll, pitch and total mass-less thrust generated by the four rotors. The above model assumes that the acceleration depends only on the magnitude and angle of the thrust vector, produced by the motors, as well as the linear damping terms $A_x, A_y, A_z \in \mathbb{R}$ and the gravitational acceleration g.

The attitude terms are modeled as a first-order system between the attitude (roll/pitch) and the references ϕ_{ref}, θ_{ref}, with gains $K_\phi, K_\theta \in \mathbb{R}$ and time constants $\tau_\phi, \tau_\theta \in \mathbb{R}$. The aforementioned terms model the closed-loop behavior of a low-level attitude controller tracking ϕ_{ref} and θ_{ref}.

Following that, let the state vector be denoted by $x = [p, v, \phi, \theta]^\top$, and the control action as $u = [T, \phi_{\text{ref}}, \theta_{\text{ref}}]^\top$. The system dynamics of the UAV are discretized with a sampling time T_s using the forward Euler method to obtain

$$x_{k+1} = \zeta(x_k, u_k). \tag{2.2}$$

This discrete model is used as the *prediction model* of the NMPC. This prediction is done with receding horizon e.g., the prediction considers a set number of steps into the future. We denote this as the *prediction horizon*, $N \in \mathbb{N}$, of the NMPC. Let us then directly denote $x_{k+j|k}$ as the predicted state at time step $k+j$, produced at the time step k. Also denote the control action as $u_{k+j|k}$. Let the full vectors of predicted states and inputs along N be denoted as $\boldsymbol{x}_k = (x_{k+j|k})_j$ and $\boldsymbol{u}_k = (u_{k+j|k})_j$. By associating a cost to a configuration of states and inputs, at the current time and in the prediction, a nonlinear optimizer can be tasked with finding an optimal set of control actions, defined by the cost minimum of this *cost function*. Putting aside the collision

avoidance constraints for now, the control objective is to reach a desired state reference x_{ref} while minimizing the actuation and minimizing the rate change in actuation from time step to time step in order to promote smooth control signals. Thus we formulate the cost function (or objective function) as:

$$J(\boldsymbol{x}_k, \boldsymbol{u}_k, u_{k-1|k}) = \sum_{j=0}^{N} \underbrace{\|x_{\text{ref}} - x_{k+j|k}\|^2_{Q_x}}_{\text{State penalty}} + \underbrace{\|u_{\text{ref}} - u_{k+j|k}\|^2_{Q_u}}_{\text{Input penalty}} + \underbrace{\|u_{k+j|k} - u_{k+j-1|k}\|^2_{Q_{\Delta u}}}_{\text{Input rate penalty}}, \quad (2.3)$$

where $Q_x \in \mathbb{R}^{8\times 8}, Q_u, Q_{\Delta u} \in \mathbb{R}^{3\times 3}$ are symmetric positive definite weight matrices for the states, inputs and input rates respectively. In (2.3), the first term denotes the *state penalty*, which penalizes deviating from a certain state reference x_{ref}. The second term denotes the *input penalty* that penalizes a deviation from a desired or baseline input vector, for the case of an UAV system this can be the steady-state input $u_{\text{ref}} = [g, 0, 0]$ i.e. the inputs that describe hovering in place. Finally, to enforce smooth control actions, a third term is added that penalizes changes in successive inputs, the *input change penalty*. It should be noted that for the first time step in the prediction, this cost depends on the previous control action $u_{k-1|k} = u_{k-1}$, which is provided as an additional input to the optimizer.

2.5 Constraint Definition and Optimization

Following the constraint formulation structure used in [47, 69] we use the function $[h]_+ = \max\{0, h\}$ as described by (2.4). This allows us to formulate the constraints as equality expressions such that $[h]_+ = 0$ implies that the constraint is satisfied, which also fits neatly into the `OpEn` framework. This form of constraint is used when we are solving the constraints with the penalty method [70], which is the case for all sections in this chapter except in section 2.8.2, where the augmented Lagrangian method [25] is used instead, and will be discussed specifically in the relevant section.

$$[h]_+ = \begin{cases} 0 & \text{if } h \leq 0 \\ h & \text{otherwise} \end{cases} \quad (2.4)$$

Equation (2.4) is used for describing a constrained area by choosing h as an expression that is positive while violating the constraint (inside the obstacle), and negative when the constraint holds (outside the obstacle). Additionally, by utilizing this constraint representation, we can define more complex obstacle shapes by taking the product of multiple such terms, as the product is zero when any of the terms are negative.

2.5.1 Obstacle Constraints of various types

Obviously a wide range of obstacles shapes and geometries can be encountered by a robot moving in a cluttered environment. This section will describe a set of useful constraint types,

which are also the ones applied in the various following application scenarios. To start, a circular expression can be used for any blocked area or general obstacle, where the radius of the circle envelops the area that is undesirable or blocked, forming an infinite cylinder in the 3D space. The circular equality constraint C_{circle} is defined in (2.5) with a specified radius and x-y position as:

$$C_{\text{circle}}(p, \xi^{\text{c}}) \coloneqq [h_{\text{circle}}]_+ = [r_{\text{c}}^2 - (p_x - p_x^{\text{c}})^2 - (p_y - p_y^{\text{c}})^2]_+ = 0, \tag{2.5}$$

where $\xi^{\text{c}} = [p_x^c, p_y^c, r_c]$ define the x,y coordinates of the center and the radius of the obstacle. Although we have obviously not described the whole NMPC problem or optimization process yet, a useful visualization of a circular or cylindrical obstacle and the simulated navigation around such an obstacle can be seen in Figure 2.1. The left image shows a cost map, where the expression in (2.5) has been mapped to the cost domain, and the right image shows the resulting constrained position space with multiple such obstacles and how the simulated UAV (in this case denoted as a MAV (Micro Aerial Vehicle) following the work it is from) does not enter the obstacles when moving around them.

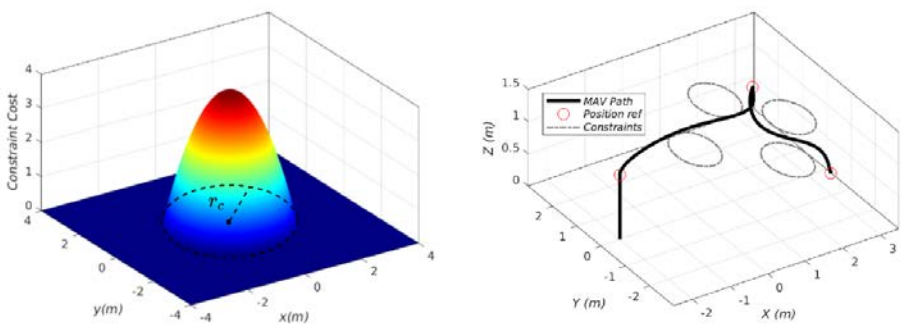


Figure 2.1: Cost map of a circular (or infinite cylinder) constraints with a radius of 2 m (left). Simulated navigation around circular obstacles (right).

The 3D version of this constraint denoted as C_{sphere}, that will be very useful when it comes to multi-robot systems and for keeping a general distance in 3D to a point-like obstacle, can be written as

$$C_{\text{sphere}}(p, \xi^{\text{s}}) \coloneqq [r_{\text{s}}^2 - (p_x - p_x^{\text{s}})^2 - (p_y - p_y^{\text{s}})^2 - (p_z - p_z^{\text{s}})^2]_+ = 0, \tag{2.6}$$

where $\xi^{\text{s}} = [r_{\text{s}}, p_x^{\text{s}}, p_y^{\text{s}}, p_z^{\text{s}}]$. Another useful constraint is a polytope surface, of which a simple and applicable version is that of a 2D rectangle as an intersection area of lines or a 3D intersection volume of planes. This type of constraint can represent any wall or flat object that fits poorly into a circular or ellipsoidal shape. The constraint for a single line takes the form of

(2.7),

$$C_{\text{line}}(p;\xi^1) \coloneqq [mp_x - p_y + b]_+ = 0, \tag{2.7}$$

where m, b are standard line constants. We want to extend this type of constraint to form a rectangle that envelops a certain area that contains the obstacle. For example, in the following section the perception system provides end-points of line segments as its output data that represents the location of a wall-like obstacle. This type of information can be turned into a rectangle of intersecting lines that envelops the line segment with a specified enlarging safety distance, visualized in Figure 2.2.

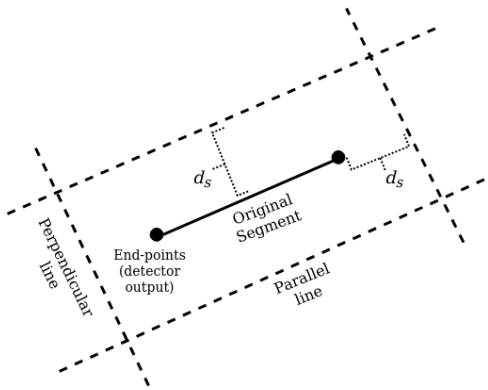


Figure 2.2: The obstacle detector outputs a line segment, which is turned into a constrained area with safety distance d_s.

To describe the 2D rectangular constraint we can take the product of four line terms as:

$$\begin{aligned} C_{\text{rec}}(p,\xi^{\text{rec}}) &\coloneqq [m_{\text{par}}p_x - p_y + b_{\text{par},1}]_+ \\ &[-(m_{\text{par}}p_x - p_y + b_{\text{par},2})]_+ [m_{\text{perp}}p_x - p_y + b_{\text{perp},1}]_+ [-(m_{\text{perp}}p_x - p_y + b_{\text{perp},2})]_+ = 0, \end{aligned} \tag{2.8}$$

where $\xi^{\text{rec}} \in \mathbb{R}^6$ includes line constants for all four lines. It should be clear that to form such a constraint, we will have two lines parallel (with constants $m_{\text{par}}, b_{\text{par},1}, b_{\text{par},2}$, and opposing signs) to the original line segment and two perpendicular (with constants $m_{\text{perp}}, b_{\text{perp},1}, b_{\text{perp},2}$). These line constants can easily be computed via simple algebraic operations to envelop a specific area, which produces a constraint as in Figure 2.3, also showing a simulated path of the UAV around the polytope surface obstacles.

To demonstrate the power and generality this constraint method offers, we can also form a *constrained passage obstacle* by combining the rectangle and the circle. For simplicity in notation and implementation, let's assume that the entrance to the passage, and the wall that the entrance is part of are along one of the coordinate axis of the robot. The expression for the entrance becomes

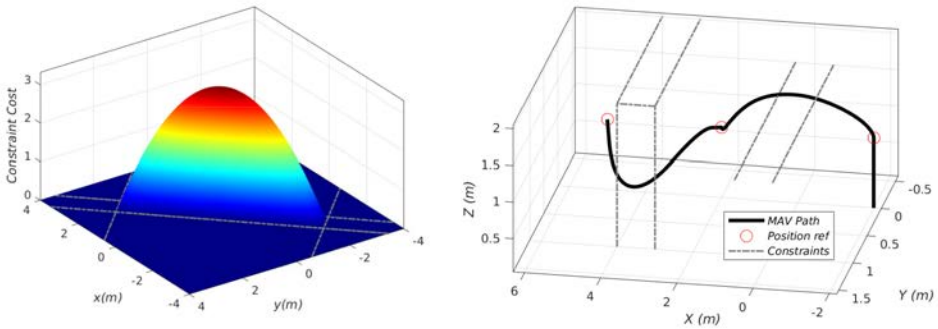


Figure 2.3: Cost map of a polytope or wall-like constraint as the intersection area of four lines (2D) (left). Simulated navigation around two wall-type obstacle (here as the 3D intersection of six planes) (right).

$$C_{\text{entrance}}(p, \xi^{\text{e}}) \coloneqq [-(r_{\text{e}}^2 - (p_y - p_y^{\text{e}})^2 - (p_z - p_z^{\text{e}})^2)]_+ = 0, \tag{2.9}$$

with $\xi^{\text{e}} = [r_{\text{e}}, p_y^{\text{e}}, p_z^{\text{e}}]$ e.g. using the circular expression with a negation such that the whole position space, except the infinite cylinder here in the y-z plane, is constrained and the constraint would only be satisfied inside the "entrance". The constrained passage representation becomes

$$C_{\text{passage}}(p, \xi^{\text{e}}, \xi^{\text{rec}}) \coloneqq [h_{\text{entrance}}]_+ [h_{\text{line},1}]_+ \ldots = 0, \tag{2.10}$$

creating a wall (rectangle) geometry with the product of four line expressions $h_{\text{line},1}, \ldots h_{\text{line},4}$ as in (2.8) with a hole in it from the entrance (negated circle) such that the constraint is satisfied ($= 0$) outside the rectangle, and inside the passage where any of the terms are zero. A similar visualisation of a simulated navigation scenario through such entrances is shown in Figure 2.4 where the robot has to navigate through the passages to reach the other side of the wall.

2.5.2 Input rate constraints

In all the works summarized in this chapter, we also impose a constraint on the successive differences of control actions for specific control variables. For the UAV case we impose such constraints on ϕ_{ref} and θ_{ref}, so as to directly prevent an overly aggressive behavior of the controller in-flight in addition to the input rate costs in the cost function, that is

$$|\phi_{\text{ref},k+j-1|k} - \phi_{\text{ref},k+j|k}| \leq \Delta\phi_{\max}, \tag{2.11a}$$

$$|\theta_{\text{ref},k+j-1|k} - \theta_{\text{ref},k+j|k}| \leq \Delta\theta_{\max}. \tag{2.11b}$$

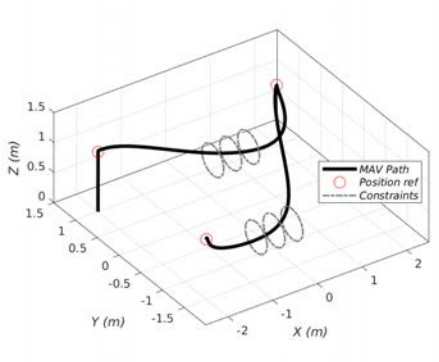


Figure 2.4: Simulated navigation through constrained passages, where the only path to the other side is through the passage visualized by the circles.

The above inequality constraints can be re-written as equality constraints as follows:

$$[\phi_{\text{ref},k+j-1|k} - \phi_{\text{ref},k+j|k} - \Delta\phi_{\max}]_+ = 0, \tag{2.12a}$$

$$[\phi_{\text{ref},k+j|k} - \phi_{\text{ref},k+j-1|k} - \Delta\phi_{\max}]_+ = 0, \tag{2.12b}$$

e.g. setting a lower and upper bound with the maximum allowed change in consecutive control action as $\Delta\phi_{\max}$. The exact same constraint is also formed for θ_{ref}, with the maximum change as $\Delta\theta_{\max}$. For the subsections using other robotic platforms, the same method of constraining the input rates are applied to the relevant control variables.

2.5.3 Input constraints

Finally, we also directly apply constraints on the control inputs. Since the NMPC is used with a real robotic systems, hard bounds on control inputs must be considered, as for example a low-level controller might only be able to stabilize the attitude within a certain range for a UAV, or due to the actuator restrictions of the platform itself for legged or wheeled robots. Thus, we define bounds on inputs as:

$$u_{\min} \leq u_{k+j|k} \leq u_{\max}. \tag{2.13}$$

2.5.4 Resulting NMPC problem and Optimization

The NMPC problem resulting from the discussed cost function and constraints is solved by the open-source Optimization Engine (OpEn) [25, 67] and its associated algorithm PANOC [68, 69], that solves nonlinear non-convex optimization problems. The Optimization Engine generates embedded-ready source code from a specified cost function and set of constraints, while it

can solve general parametric optimization problems on the form:

$$\mathbb{P}(x_{k|k}) : \text{Minimize}_{z\in Z} f(z,\rho) \tag{2.14a}$$

$$\text{subject to: } G(z,\rho) = 0, \tag{2.14b}$$

where f is a continuously differentiable function with Lipschitz-continous gradient function and G is a vector-valued mapping so that $\|G(z,\rho)\|^2$ is a continuously differentiable function with Lipschitz-continous gradient. The decision variable and additional input parameter are denoted by z and $\rho \in \mathbb{R}^{n_p}$ respectively. ρ is an input parameter to the NMPC module including the initial measured state of the UAV $\hat{x}_k = x_{k|k}$, references u_{ref} and x_{ref}, the previous control action u_{k-1} (for the first term in the input change cost), and importantly the obstacle data ξ^{obs} ($\xi^{\text{c}}, \xi^{\text{rec}}$ etc.), where n_p is the total number of such input parameters.

Based on the cost function and constraints outlined in the previous sections we can now formulate the NMPC problem. For simplifying the notation, let us collect all obstacle terms of interest for the specific use-case into a single notation as $\boldsymbol{C}_{\text{obs}}$ that could include multiple of the same obstacle, or a mix of different defined obstacle types. The number of such constraints considered in the NMPC problem is then N_{obs}. The NMPC prolem becomes:

$$\underset{\boldsymbol{u}_k, \boldsymbol{x}_k}{\text{Minimize}} J(\boldsymbol{x}_k, \boldsymbol{u}_k, u_{k-1|k}) \tag{2.15a}$$

$$\text{s. t.: } x_{k+j+1|k} = \zeta(x_{k+j|k}, u_{k+j|k}),$$

$$j = 0, \ldots, N-1, \tag{2.15b}$$

$$u_{\min} \leq u_{k+j|k} \leq u_{\max},\ j = 0, \ldots, N, \tag{2.15c}$$

$$C_{\text{obs}}(p_{k+j|k}, \xi_i^{\text{obs}}) = 0,\ j = 0, \ldots, N, \tag{2.15d}$$

$$i = 1, \ldots, N_{\text{obs}} \tag{2.15e}$$

$$\text{Rate constraints (2.12)}, j = 0, \ldots, N, \tag{2.15f}$$

$$x_{k|k} = \hat{x}_k. \tag{2.15g}$$

This can be fit into the OpEn framework by performing a *single-shooting* of the cost function via the decision variable $z = \boldsymbol{u}_k$ and define Z by the input constraints (2.13). We also define G to cast the previously discussed equality constraints. For the consideration of the constraints, a quadratic penalty method [70, 74] is applied, as it allows for the types of equality constraints proposed requiring only that the mapping of the constraint expression is continuously differentiable with Lipschitz-continous gradient. By formulating the problem as:

$$\text{Minimize}_{z\in Z} f(z,\rho) + q\|G(z,\rho)\|^2, \tag{2.16}$$

where $q \in \mathbb{R}_+$ is a positive penalty parameter, the PANOC algorithm [69] can be applied to the problem. When using a penalty method, an optimization problem where the constraints are mapped to the cost-domain is re-solved multiple times with an increasing penalty parameter q

associated to the constraints, while using the previous solution as the initial guess (hotstart). In very simplified terms, this method gradually moves the cost-minima of (2.16) by increasing q until non of the constraints are violated, or rather until a specified tolerance is met.

In Figure 2.5 the penalty method concept is displayed, where the solution from one to five penalty method iteration are shown for obstacle avoidance around a circular obstacle where $q^i = 10^i, i = 1,\ldots,5$. As the penalty parameter is increased, the optimized trajectory, $\boldsymbol{u}_k$ (here displayed in position coordinates via the prediction model), is moved out of the obstacle. The trajectory will lie completely outside the obstacle as q approaches infinity, and as such small constraint violations should be expected and compensated by enlarging the obstacle. In Figure 2.5 it is depicted that the first iterations are not significantly different, and in the following experimental evaluations we will be using penalties with $q^i = 10^3 \times (4^{i-1}), i = 1,\ldots,4$, based on the results from initial testing and simulations.

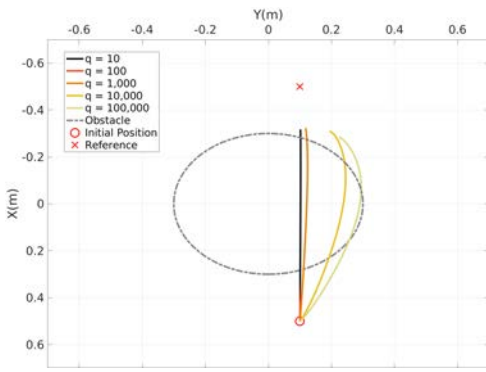


Figure 2.5: NMPC optimized trajectories using one to five penalty method iterations for a circular obstacle with radius $r = 0.3m$. The cost-minima is gradually pushed out as the cost associated with violating the constraint is increased.

2.6 Perception-based Reactive Navigation

In order for the described constrained NMPC to act as an independent and fully functional reactive local navigation scheme, the fundamental link between the NMPC constraints and perception information from onboard sensors must be investigated. This is one of the major contributions of this chapter, since a vast majority of previous works, to the authors best knowledge, rely either on simulations [75–78], or when experiments are performed rely on pre-defined obstacles [47, 69], or motion-capture system to track obstacles [79, 80]. This limits the real-world impact of NMPC-based obstacle avoidance in the robotics context, and we will show in multiple challenging laboratory experiments the application of an NMPC-based obstacle avoidance scheme in real-time, real-world scenarios for an Unmanned Aerial Vehi-

cle (UAV), relying only on on-board computation and perception using on-board 2D LiDAR, forming a complete navigation scheme for constrained environments. As will be experimentally demonstrated the novel proposed reactive scheme is computationally efficient and fast enough to satisfy the run-time requirements of the UAV platform, using limited computation power, while maintaining a safe distance from any obstacle and performing smooth and efficient obstacle avoidance maneuvers.

2.6.1 Directly using LiDAR data

A first thought in this direction could be to use the raw 2D LiDAR hits in close proximity to the UAV, and form a separate circular constraint for each LiDAR point sufficiently close to the UAV to be reachable within the prediction horizon. This method was attempted in simulations using the physics-based gazebo simulator that also includes realistic sensor data from an onboard 2D LiDAR, and the flight dynamics of the platform. Figure 2.6 shows the simulated mission set-up in a maze-like scenario. We can see that the constrained NMPC manages to navigate through the maze but the path is not optimal. Similarly, Figure 2.7 shows the safety distance from the environment (from the 2D LiDAR range measurements) where we can see that a specified 0.7 m safety distance is maintained throughout the scenario (e.g. $r_c = 0.7$), but the figure also shows that the computation time exceeds a desired 20 Hz (or 50 ms) sampling time/rate specified by previous hardware experiments for UAVs [47], and this was not even using limited onboard computation resources for the simulation. The straightforward reason is the very high number of constraints required if each LiDAR point forms a separate constraints along the horizon. Additionally, local minima can occur in between points/constraints and the result is either the robot getting stuck, or the suboptimal resulting maneuver shown in Figure 2.6. A conclusion is if we desire optimal behavior and real-time execution with limited resources the perception-constraint link has to be done in a more proper way.

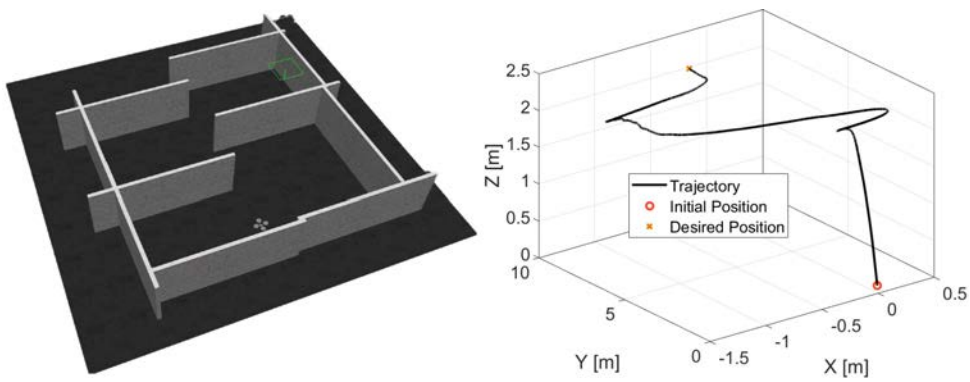


Figure 2.6: Gazebo environment setup and 2D-LiDAR equipped UAV (left). Path traversed through the environment when given a waypoint on the opposite side of the maze.

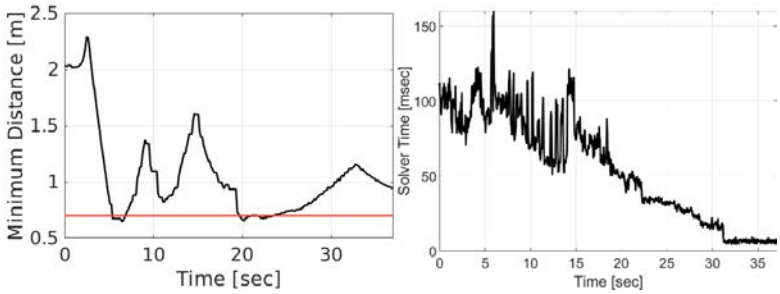


Figure 2.7: Minimum-range LiDAR measurement (as a measure of safety distance) (left), and the solver-time of the NMPC module throughout the simulation run (right).

2.6.2 Obstacle Detection and Classification

The solution for linking 2D LiDAR sensor information to the NMPC constraint came in the form of the Obstacle Detector package [81]. The package uses combined segmentation and merging of obstacles from 2D LiDAR data and it is a perfect fit for the required needs, providing geometric approximations of the surrounding environment in the form of line segments and circles, which, not incidentally, fits into the previously discussed obstacle types. Circular obstacles, c, are, just like in the constraint formulation, defined by a radius and the x-y coordinates of the center of the circle as;

$$c \triangleq \{r_c, p^c\} = \{r_c, (p_x^c, p_y^c)\}. \tag{2.17}$$

where r_c, p_x^c, p_y^c define the radius, and x,y-position of the center of the circle. The obstacle detector also supports an additional safety distance, d_s, such that $r_c = r_{\text{real}} + d_s$. This safety distance is required, since the constraints for obstacle avoidance assumes the position of the UAV to be expressed by a point. As such, d_s represents the size of the UAV, and also in practise an extra increase to compensate for inaccuracies in measurements, solver tolerances, and limited penalty method iterations. Line segments, l, are defined by extreme points of the line segment as:

$$l \triangleq \{p^{l,1}, p^{l,2}\} = \{(p_x^{l,1}, p_y^{l,1}), (p_x^{l,2}, p_y^{l,2})\}, \tag{2.18}$$

where $(p_x^{l,1}, p_y^{l,1})$ defines the x,y-position of the start-point of the line segment and $(p_x^{l,2}, p_y^{l,2})$ the end-point. From each such set of points, we can parameterize a rectangular constraint by computing the four line equations, as shown in Figure 2.2, to envelop the original line segment with a rectangle using the same extra safety distance d_s. As the detector is limited to outputting circles and line segments, obstacles are forced to be classified into either one which is obviously a limiting factor.

2.6.3 Hardware Experiments

Laboratory experiments were performed to validate the pairing between the NMPC and obstacle detection, and the ability for the `OpEn`-based NMPC to generate safe and consistent

maneuvers around obstacles. The UAV platform used can be seen in Figure 2.8, equipped with a rotating 2D LiDAR and using an Intel UpBoard as its computation resource. A motion capture system was used to estimate the UAV state $\hat{x}$, and the full sensing and autonomy architecture can be seen in Figure 2.9. The pipeline for object detection can be seen in Figure 2.10, where the Obstacle Detector turns the surrounding environment into a set of line segments and circles which can then be fit into the defined constraints, shown as cost maps. As a note, the prediction horizon was set to $N = 40$ in the following experiments implying a 2 s prediction with the controller running at 20 Hz.

Figure 2.8: UAV used in the following hardware experiments. Notably, the UAV is equipped with a rotating 2D LiDAR (for obstacle perception), an Intel UpBoard for onboard computation, and the ROSflight FCU.

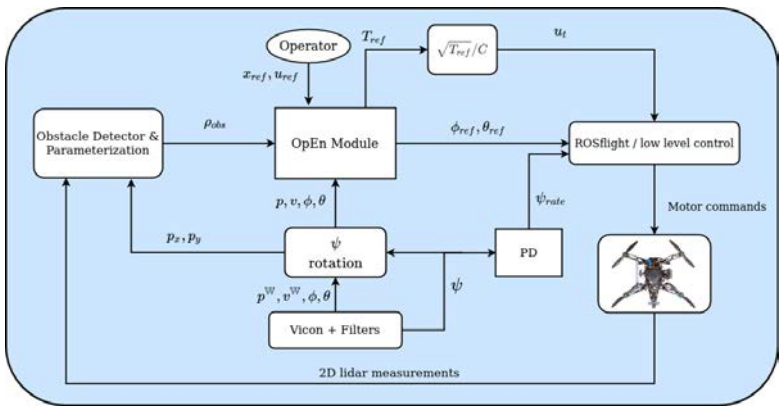


Figure 2.9: Complete control and information flow architecture during hardware experiments. State estimation is provided by a Vicon motion capture system in a global world frame $\mathcal{W}$, while obstacles are detected by 2D LiDAR and the Obstacle Detector. State and obstacle information is fed to the NMPC which generates control inputs to the ROSflight attitude controller.

The best way to visualize the experiment is by the following video link: `https://www.`

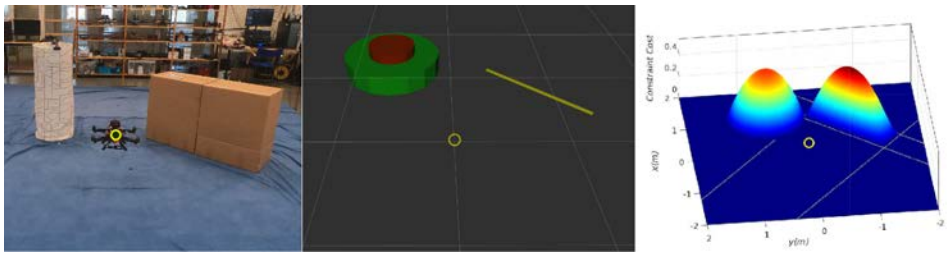


Figure 2.10: The pipeline for obstacle constraint generation. 2D LiDAR equipped UAV and obstacles (left), the output of the Obstacle Detector (middle), and resulting constraint cost maps (right).

youtube.com/watch?v=xl_YQuDjs1M&feature=youtu.be. Here the reader can see the laboratory scenarios set up for validation, the real-time behavior of the platform, and the generated prediction trajectories. The evaluations were designed such that after take-off, a single position reference was given to the UAV on the opposite side of the area. In between the take-off location and the provided position reference, obstacles were placed in such a way that the UAV is forced to maneuver around them. Three experiment scenarios were trialed: 1) A baseline generic cylindrical obstacle, 2) a multi-obstacle course of wall-like obstacles that the UAV has to move between, and 3) a scenario with a very narrow opening between obstacles where the constrained NMPC particularly can shine through guaranteeing hard bounds on safety distances. The first experiment is visualized in Figure 2.11 showing the full avoidance path as well as snap shot images from the experiment. Notably the third sub-plot shows the maintaining of a specified hard bound of 0.4 m safety distance (based on the size of the platform) from the detected obstacle surface.

The obstacle course scenario is visualized in Figure 2.12. Here the UAV must pass between two obstacles placed 1 m apart while maintaining the 0.4 m safety distance from each. The obstacles are correctly detected and a safe distance is maintained while navigating past them. The generated avoidance trajectory is efficient and smooth considering the scenario.

The last experiment with a narrow opening is visualized in Figure 2.13. The scenario is simple: the only way to get to the other side is through the narrow opening of 0.85 m (e.g. there is a total of 5 cm clearance based on the desired safety distance). The UAV executes an aggressive avoidance maneuver to perfectly fit in between the obstacles. Here the ability to perfectly sync the control behavior with the obstacle avoidance really shines through, as the maneuver is precise, quick, and totally within the ability for the platform to execute efficiently.

Figure 2.14 shows the minimum-range LiDAR measurements throughout the three scenarios where the maximum violation of safety distnaces was 3 cm, a very small violation considering the real-world experiment scenario with the narrow opening, and the fact that the obstacle perception data comes from onboard sensors and will as such not be perfect. Figure 2.14 also shows the solver times from the experiments. At a 20 Hz control loop rate, set the same as the discretization rate in the prediction model, the goal is to keep below a 50 ms solver time using only the limited onboard resources. To maintain flight stability the solver was set up to stop the optimization process after 50 ms in case it would get stuck, which is why the one spike in the second experiment, and the peak in the third experiment is maximally 50 ms. In general, the

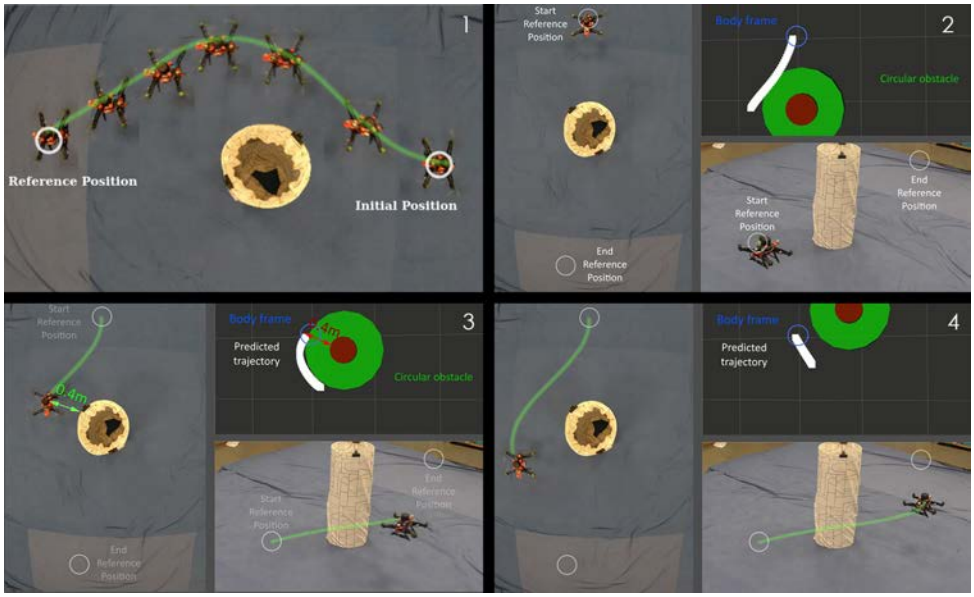


Figure 2.11: Path around a cylinder-like obstacle in a laboratory environment when given a position reference on the other side, using perception-based constraints. (1) complete path around obstacle, (2)-(4) showing snapshots during navigation highlighting the predicted trajectory and detected obstacles.

optimization time is kept well below, but there are critical moments where the solver does not converge within the desired time. This can cause significant problems if a control input that is not collision free is applied to the system but thankfully due to the very fast re-calculation rate (twenty times per second) any incorrectly generated control signals were quickly corrected before a collision or violation of safety distances.

Overall, the NMPC showed significant promise as a local navigation scheme in very challenging scenarios where maintaining a safe distance is critical, and the robot must navigating in between obstacles in a cluttered environment. A missing experiment for this direction is definitely navigation through a larger scale area cluttered with various obstacles of different shapes and sizes, but was not possible at the time due to the size of the testing arena.

2.7 Obstacle Avoidance of Dynamic Obstacles

While most autonomy framework are satisfied with safely navigating through a static environment, there are certain scenarios where the consideration of moving obstacles are absolutely critical. For example, if autonomous robots are to be used in close quarters with humans, vehicles, and other robots in urban environments, special attentions has to be placed on their ability to correctly coordinate maneuvers around obstacles that are also moving. The fundamental difference between the usual notion of collision avoidance and avoidance of moving obstacles is

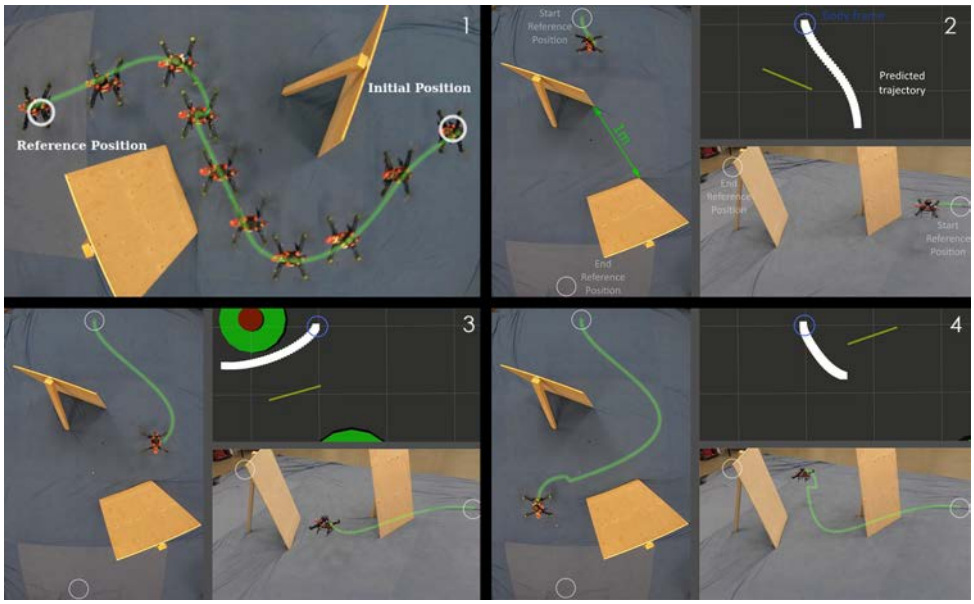


Figure 2.12: Path around two wall-like obstacles in a laboratory environment when given a position reference on the other side, using perception-based constraints. (1) complete path around obstacles, (2)-(4) showing snapshots during navigation highlighting the predicted trajectory and detected obstacles.

that for a rapidly moving obstacle, or a robot with slow dynamics, the avoidance maneuver has to start long before the obstacle enters the current unsafe zone or *area of influence* of the avoidance method or algorithm. To do so, an important component is the ability to predict where an obstacle will be in the future, and to connect that prediction to the avoidance algorithm, to enable an early response and an efficient avoidance maneuver. Using the quite clear example of the autonomous car, safety critical maneuvering in regards to pedestrians or other cars must include direct considerations of how those obstacles are predicted to move in the future so that the car can break in time to avoid the collision, which is why significant research effort in this direction were done the context of autonomous cars [82], since it is required to ensure collision-free paths in the urban environment. These methods use a wide array of stochastic prediction models [83] or hypothesis based models [84] for the consideration of moving obstacles. For smaller-scale robots the real-world scenarios include human-robot safety in cluttered environments, objects thrown or launched at for example a UAV, or in collaborative robotic swarms.

This section will discuss the relatively straight-forward addition of moving (or dynamic) obstacles into the developed constrained NMPC framework through two works. One focuses on UAVs and rapidly moving obstacles of various types where we will introduce the obstacle prediction and a trajectory classification scheme as add-ons to the baseline framework. The second work focuses on a legged robot (Boston Dynamics Spot), where we adapt the system

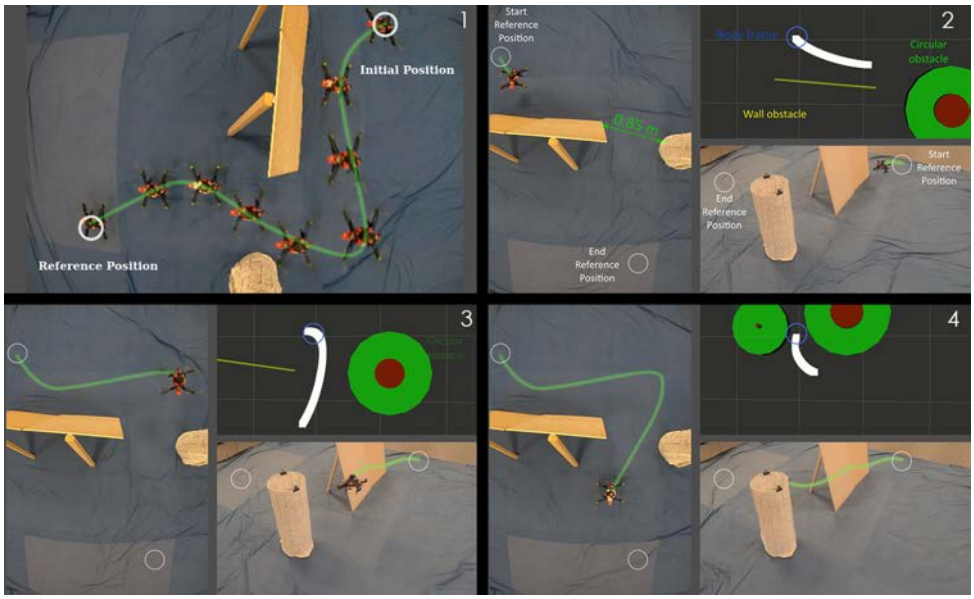


Figure 2.13: Path through a very narrow constrained opening in a laboratory environment when given a position reference on the other side, using perception-based constraints. (1) complete path through opening, (2)-(4) showing snapshots during navigation highlighting the predicted trajectory and detected obstacles.

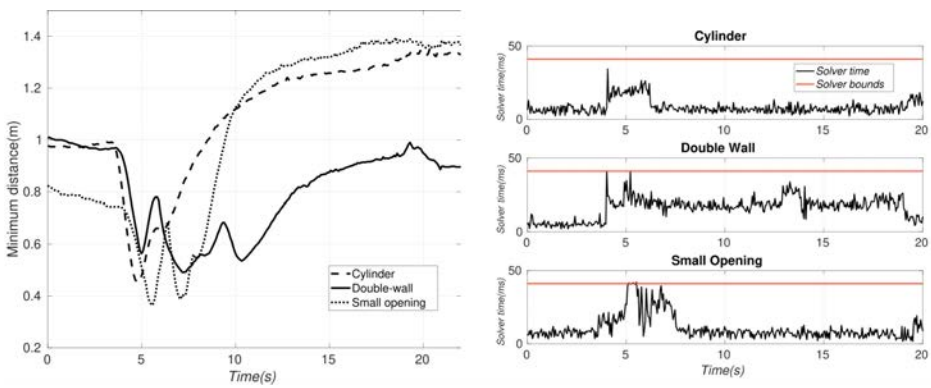


Figure 2.14: Minimum-range LiDAR measurement throughout the three experiments with a set safety distance $d_s = 0.4\,\text{m}$ (left) and solvertime on limited hardware for the NMPC module (right).

model to fit the new robot platform. This work considers the human-robot safety scenario where the legged robot operates in a fully autonomous mode, using onboard sensing to detect and track the human, as well as onboard state-estimation and computation.

2.7.1 Avoidance of various trajectory types with an UAV.

Consider a general obstacle that we would like the UAV to maintain a specified safety distance from. The spherical obstacle described in (2.6) fits our purpose well, where we can say that the radius of the sphere $r_s = r_{\text{uav}} + r_{\text{obs}} + d_s$, e.g. the size-radius of the robot plus the approximate size-radius of the obstacle, plus an extra safety distance. Fundamentally, in (2.15) we are forming a separate constraint for each predicted time step along the horizon and for each obstacle which we denoted $C_{\text{s}}(p_{k+j|k}, \xi_i^{\text{s}}) = 0,\ j = 0,\dots,N, i = 1,\dots,N_{\text{s}}$, with ξ_i^{s} in that case describing the relevant information (position, size, etc.) of the obstacle, as an input to the `OpEn` solver. Let us extend the notation to have a separate obstacle data input for each constraint along the horizon as $\xi_{i,k+j|k}^{\text{s}}$, where we can now feed a predicted obstacle trajectory (along the horizon, discretized at the same sampling rate as the system prediction model) as an input to the solver. This is formulated such that the obstacle trajectory is fully parametric and as such the prediction of future obstacle positions is external to the NMPC module, and it is agnostic to the method of obstacle prediction. Any discrete prediction model of the obstacle will have larger errors for predictions further into the future due to propagated model mismatch and estimation noise. Let us right away implement a compensating obstacle enlargement as a brute-force way to compensate for that error as $d_{s,k+j|k}$, and in this work we will consider a simple linearly increasing safety distance along the horizon, as that worked very well in practise. As an example, in the following evaluations this parameter is 0.4 m at $k|k$ and increases up to 0.6 m at $k+N|k$

The next step is computing the $\xi_{k+j|k}^{\text{s}}$ for all $j = 0,\dots N$, e.g. the predicted obstacle trajectory. In this work we will be using a trajectory classification and prediction scheme where a buffer of measurements of the obstacle position and velocity are used to classify its motion into one specific type from a pre-defined set of trajectory models. While many kinds of trajectories can be encountered and considered in the scheme, we are focusing on three types: obstacles moving with linear motion, obstacle moving with projectile motion, and static obstacles ($\dot{p}^s = 0$). Linear motion is described by:

$$\dot{p}^{\text{obs}}(t) = v^{obs}(t), \tag{2.19}$$

where v^{obs} are the velocities of the obstacle and no forces are acting on the obstacle. The projectile-motion trajectory is defined by:

$$\dot{p}^{\text{obs}}(t) = v^{obs}(t), \tag{2.20a}$$

$$\dot{v}^{\text{obs}}(t) = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \begin{bmatrix} B_x & 0 & 0 \\ 0 & B_y & 0 \\ 0 & 0 & B_z \end{bmatrix} v^{obs}(t), \tag{2.20b}$$

where B are linear aerodynamic damping terms. The buoyancy force of the obstacle is considered small enough to ignore. Equations (2.19), (2.20) are then discretized by the forward Euler method using the same sampling time as the controller/prediction model, T_s. In the discrete prediction model of the projectile-motion we also include a condition for bouncing, with a much-simplified collision interaction with a coefficient of restitution applied on the velocities (assuming the ground is flat) as the sphere hits the ground to model the energy-loss in the collision. The result is that from an initial measured state $[\hat{p}^{obs}, \hat{v}^{obs}]$ we can compute any $\xi_{k+j|k}^{\text{s}}$.

The classification is done by a simple heuristic that is comparing the M last measured position and velocity terms to a backwards prediction based on the above equations iterating from the current measured state. Using the same notation as for the prediction of the NMPC, let $p^{obs}_{k|k}$ denote the current measurement and $p^{obs}_{k|k-j}$ denote the predicted obstacle state j steps back in time based on the initial condition. Also denote the previous measurements in position and velocity of the obstacle as $p^{obs,prev}_j$ and $v^{obs,prev}_j$ respectively (a buffer of previously measured states). We measure the error, e^{traj}, as:

$$e^{traj} = \sum_{j=1}^{M} \mid p^{obs,prev}_j - p^{obs}_{k|k-j} \mid + \mid v^{obs,prev}_j - v^{obs}_{k|k-j} \mid , \tag{2.21}$$

e.g. comparing the measured previous states with the "backwards predicted" previous states, and summing the error for the M measurements to get the total. Equation (2.21) is evaluated for the three different classes of trajectories and the class that generates the lowest error is chosen for the trajectory prediction, as it is the best match from the measured obstacle state information. This is run for every new measurement of the obstacle and thus the trajectory of a single obstacle is allowed to change during the movement, and the predicted trajectory only depends on the current/lates measurement. It should be noted that the static condition and the linear are identical for an actual static obstacle, but generally this type of measurement will include noise and uncertainties and thus the static condition is included to filter for erroneous velocity and position measurements on a static obstacle. The outcome is that we can now evaluate the dynamic obstacle avoidance in different types of scenarios without having to re-define the obstacle prediction specifically for that scenario.

With this relatively straight-forward addition we are ready to evaluate the NMPC for dynamic obstacle avoidance scenarios. The UAV platform used for experiments is the Crazyflie Nano Quadcopter, depicted in Figure 2.15.

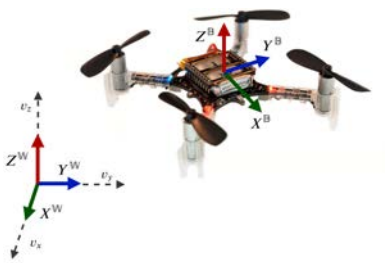


Figure 2.15: Platform used for experimental evaluation - The Crazyflie 2.0 Nano Quadcopter.

The smaller platform was chosen due to the risks related to throwing obstacles at the robot, and for operations in close proximity to humans. The downside is that no computation or perception can be run onboard the platform and as such the NMPC module is run on a laptop and the control signals $T, \theta_{ref}, \phi_{ref}, \dot{\psi}$ are sent to the Crazyflie over Bluetooth communication.

This does result in delays of approximately 0.1 s before a computed control signal actuates the vehicle. A Vicon motion capture system tracks both the UAV and the obstacle in the following experiment evaluations and provides both $\hat{x}$ and $[\hat{p}^{obs}, \hat{v}^{obs}]$. Four experiments are performed: 1) Avoidance of an obstacle thrown at the UAV (projectile motion) visualized in Figure 2.16, 2) A "pedestrian" walking towards the UAV visualized in Figure 2.17, 3) A bouncing ball scenario, shown in Figure 2.18 where we leverage the fact that we can provide a trajectory of arbitrary form and as such can easily include the bouncing dynamic, and 4) a scenario where another autonomous UAV moves through the ego agents' space while an obstacle is simultaneously thrown at it, which is demonstrated in Figure 2.19.

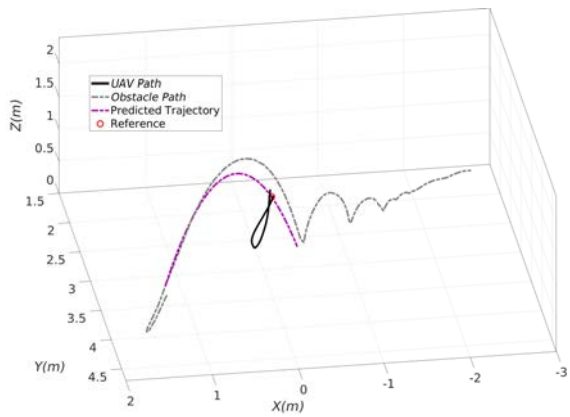


Figure 2.16: Experiment with a ball thrown at the UAV. Obstacle trajectory (gray dotted line), predicted trajectory at the first moment a collision is predicted (purple), and the resulting collision avoidance maneuver (black).

Visualizing multiple moving objects in static images is very difficult, and as such the reader is recommended to watch the following video (and it is really the only way to properly see the results) demonstrating the dynamic avoidance capabilities: `https://www.youtube.com/watch?v=vO3xjvMMNJ4`. The video also includes two comparisons using a basic Artificial Potential Field [34] and using the NMPC from the last section without the obstacle prediction. The safety distances between obstacle and UAV are shown in Figure 2.20, where we can see that the other methods fail and result in a crash, while the dynamic obstacle NMPC maintains the desired distances , with the closest obstacle-UAV distance in any experiment was 0.38 m, only a 2 cm violation of the desired safety distance. This occurred during the bouncing ball experiment, and in all other scenarios the desired safety distance was maintained. Figure 2.20 also shows the performance of the trajectory classification scheme. The plot shows the values of e^{traj} for each obstacle class during the initial moments in the experiments where the obstacle starts to move. It only took $0.1 - 0.2$ s from the initial movement (e.g. ball is thrown, pedestrian starts walking) for the correct trajectory class to be identified, the trajectory predicted, and for the controller to react to the moving obstacles.

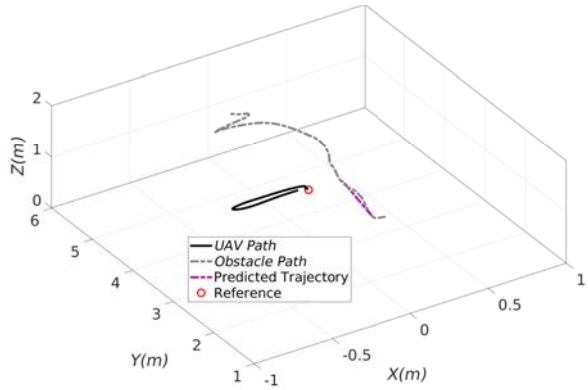


Figure 2.17: Experiment with a moving towards the UAV. Obstacle trajectory, predicted trajectory at the moment a collision is predicted, and the resulting collision avoidance maneuver.

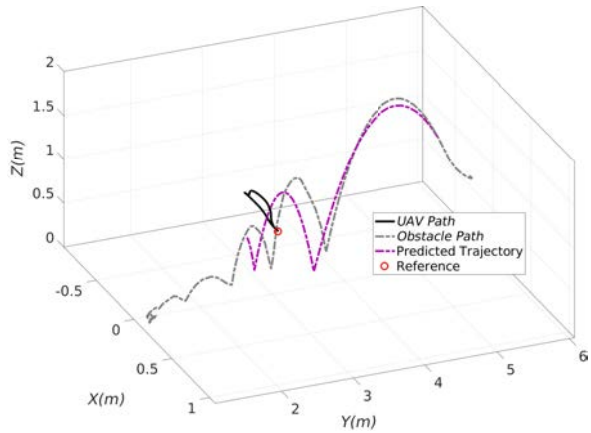


Figure 2.18: Experiment with a bouncing ball, where the obstacle would hit the UAV after the first bounce. Obstacle trajectory, predicted trajectory at the moment a collision is predicted, and the resulting collision avoidance maneuver.

2.7.2 Human-Robot Safety for legged robots

The same ideas were also demonstrated on a legged robot platform, namely the Boston Dynamics (BD) Spot. The aim of this section is to demonstrate collision avoidance of moving dynamic obstacles in the context of complete autonomy, meaning no motion-capture system or pre-defined obstacle, instead only using onboard sensing and computation. The NMPC is thus augmented with several other assisting autonomy modules and sensing capabilities, where the robot, full autonomy kit, sensors, and information flow is visualized in Figure 2.21.

This section will focus on the NMPC implementation and on the experimental results, while a more detailed description of the other components can be found in [85]. On a high level, the

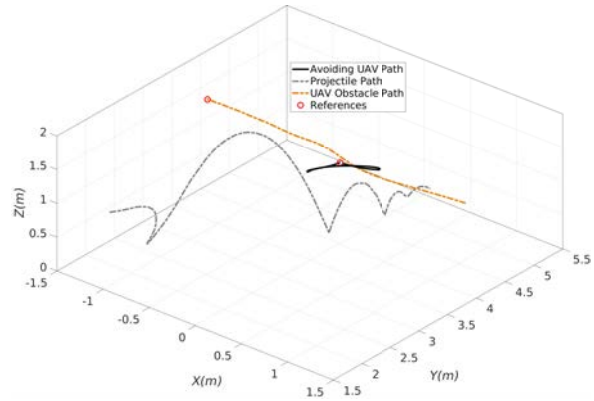


Figure 2.19: Experiment with multiple dynamic obstacles; one projectile and one linearly moving non-cooperative UAV. Obstacle trajectories and the resulting collision avoidance maneuver.

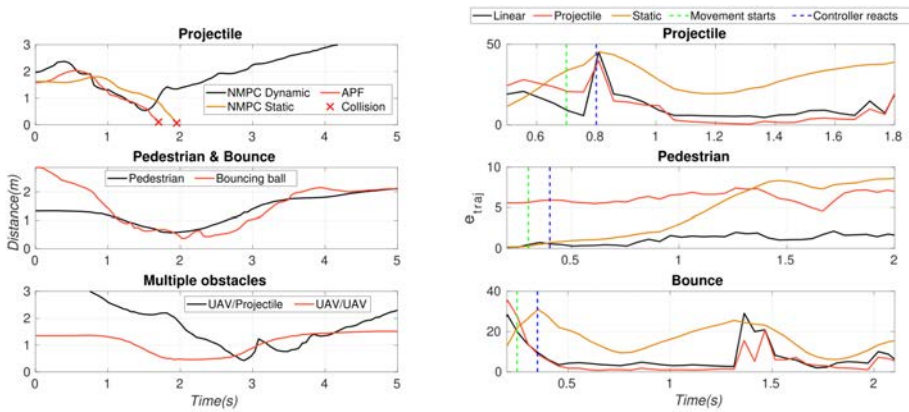


Figure 2.20: Relative distances as measured from the center of the obstacle to the center of the UAV (from Vicon Mo-cap) (left), and the trajectory classification scheme showing also the time of obstacle movement beginning and the controller initiating obstacle avoidance (right).

track-and-avoid architecture is divided into three steps: 1) CNN-based (Convoluted Neural Network) object detection using the YOLO [86] (You Only Look Once) framework trained on the Microsoft Coco dataset for human detection [87], 2) an object localization and state estimation pipeline for tracking human obstacles that is using depth-camera information and a Kalman filter [88] with a constant acceleration model to estimate the obstacle states $[\hat{p}^{obs}, \hat{v}^{obs}]$, and 3) the proposed NMPC to perform the active tracking and avoidance maneuvers. To supplement full autonomy alongside the proposed framework we use the Lidar-Inertial Odometry LIO-SAM [89] to estimate the robot state $\hat{x}$. While there are some differences in NMPC implementation, most key components stay the same: the constraint formulation, cost function,

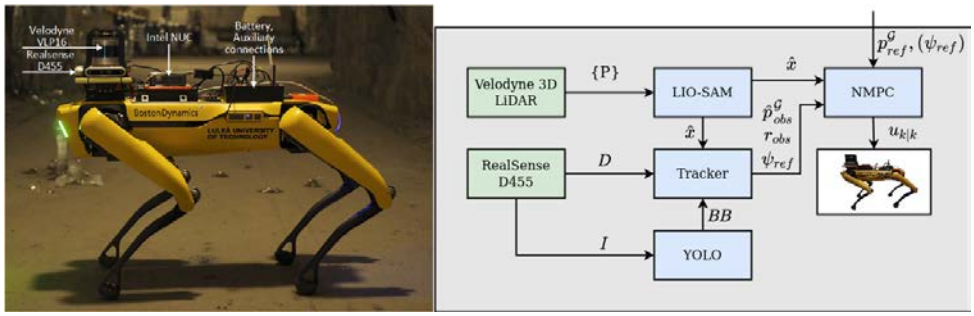


Figure 2.21: Boston Dynamics spot with custom sensor suite for full autonomy (left), and the Track & Avoid architecture used for experiments (right). The velodyne 3D Lidar generates the pointcloud $\{P\}$ used for state-estimation, while the Intel RealSense D455 generates RGB image I and depth image D which are used to detect and localize the obstacle.

solver etc. are almost identical, and this section will thus only discuss the differences from the baseline framework. As the motion of the legged robot is in 2D, we will only consider the linear motion obstacle model from the previous section, which is also more commonly denoted as *constant velocity obstacles*.

As the autonomy architecture relies on the built-in Spot walking controller, we can ignore the complex low-level dynamics of autonomous quadruped walking movements. Instead, we define a high level model with states $x = [p_x^{\mathcal{G}}, p_y^{\mathcal{G}}, v_x^{\mathcal{B}}, v_y^{\mathcal{B}}, \psi]^T$ and with control inputs $u = [u_{vx}^{\mathcal{B}}, u_{vy}^{\mathcal{B}}, u_\omega]^T$. Here, the denoted frame $\mathcal{G}$ is in a global frame of reference, and robot body frame velocities in $\mathcal{B}$ represent velocities with a rotation about the z-axis with the heading angle ψ. The proposed high level kinematic model for the legged robot, keeping the position coordinates in a global frame, can then be formulated as:

$$
\begin{aligned}
\dot{p}_x^{\mathcal{G}}(t) &= \cos\psi(t) v_x^{\mathcal{B}}(t) - \sin\psi(t) v_y^{\mathcal{B}}(t) \\
\dot{p}_y^{\mathcal{G}}(t) &= \sin\psi(t) v_x^{\mathcal{B}}(t) + \cos\psi(t) v_y^{\mathcal{B}}(t) \\
\dot{v}_x^{\mathcal{B}}(t) &= 1/\tau_{vx}(\kappa_{vx}(u_{vx}^{\mathcal{B}}(t) - v_x^{\mathcal{B}}(t)) \\
\dot{v}_y^{\mathcal{B}}(t) &= 1/\tau_{vy}(\kappa_{vy}(u_{vy}^{\mathcal{B}}(t) - v_y^{\mathcal{B}}(t)) \\
\dot{\psi}(t) &= \kappa_\omega u_\omega(t)
\end{aligned}
\tag{2.22}
$$

where the evolution of velocity states is modelled as a first-order system with gains $\kappa_{vx}, \kappa_{vy} \in \mathbb{R}$ and time constants $\tau_{vx}, \tau_{vy} \in \mathbb{R}$, representing the response of the low-level Spot controller when acting on control input u, while $\kappa_\omega \in \mathbb{R}$ is the gain related to the rotational movements. A fundamental difference here is that we want the NMPC to both track (keep within field of view of the camera) and avoid the obstacle, and as such, as opposed to the UAV model, we must directly include the heading state ψ into the problem. If we want to keep the global frame, it becomes a difficult to apply the same quadratic cost to the heading state as for the other terms in equation (2.3) as the heading state is 2π-modular. For example, if we assume that the state

estimation provides $\psi \in [-\pi, \pi]$, there is a clear discontinuity at $\psi \sim \pm\pi$ and an incorrect cost associated with $\psi_{ref} - \psi$ for a subset of initial conditions and references. There are ways around that by shifting the coordinate frames and such, but as the solver supports the nonlinear terms we could use a trigonometric function in the cost function specifically for the heading state cost for $\psi_{ref} - \psi$. A variety of expressions could be used but here we add the following term to (2.3): $(-\cos(\psi_{ref} - \psi) + 1)Q_{\psi}$. The expression is at its minimum for $(\psi_{ref} - \psi) \mod 2\pi = 0$ and positive real valued.

As opposed to the previous work on dynamic avoidance, we can not assume that the obstacle size is known. The robot moves in 2D, so we can use the circular constraint from (2.5) for the set-exclusion constraint that represents the obstacle. The object localizer provides the width of the detected bounding box with the human obstacle in it as W_{obs}. We keep the desired minimum safety distance to the obstacle as d_s. Additionally, let us use the co-variances of the measured obstacle state from the Kalman filter (Figure 2.21) as a way to enlarge the obstacle in the presence of uncertainty, where σ_p, σ_v are the Euclidean norms of co-variances of the tracking Kalman filter along each axis for the position and velocity states respectively. The idea is to increase the obstacle radius if the measurement (and thus also prediction) uncertainty of obstacle states are high. A similar idea was used in [56] using robot state uncertainties, while in our case the co-variances represent how uncertain the predicted obstacle positions are. The radius of the circular constraint becomes

$$r_c = W_{\text{obs}}/2 + d_s + c_{\sigma}(\sigma_p + \sigma_v) \tag{2.23}$$

with c_{σ} as a scaling constant. The legged robot is not enveloped well by a single circle due to its shape, and as such we impose two circular constraints which can be seen in Figure 2.22, one over the shoulder that overlaps with the estimated position coordinate, and another offset by the robot length l as to be situated over the rear.

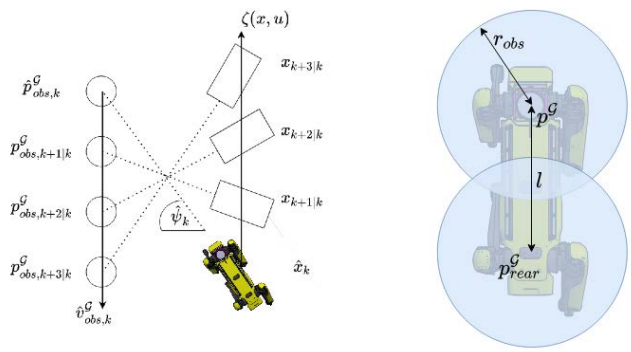


Figure 2.22: Active tracking of the obstacle based on predicted trajectories of both the robot and the obstacle (left), and the constraints used for obstacle avoidance (right).

Figure 2.22 also shows the concept of predictive tracking. We want to generate heading references along the prediction horizon that track the moving obstacle, such that the obstacle

never leaves the field of view of the onboard camera. We also impose this as a constraint as:

$$C_{\text{track}}(\psi, \psi_{\text{ref}}, \beta) \coloneqq [-\cos(\psi_{ref} - \psi) + 1 - \beta]_+ = 0, \tag{2.24}$$

using a similar formulation as the heading cost. Here β represents the allowed offset between the heading state ψ and the generated heading reference ψ_{ref} that is based on the field of view of the camera. The heading reference to look towards the obstacle is

$$\psi_{\text{ref}} = \arctan(p^{\mathscr{G}}_{obs,y} - p^{\mathscr{G}}_{y}, p^{\mathscr{G}}_{obs,x} - p^{\mathscr{G}}_{x}). \tag{2.25}$$

We can evaluate this expression based on future predicted robot states and predicted obstacle states via the discrete prediction model $x_{k+1} = \zeta(x_k, u_k)$. We use the previous solution $\boldsymbol{u}_{k-1}$, under the assumption that NMPC trajectories will not differ significantly from one time step to the next during fast run-time operations to generate $\psi_{\text{ref,j}}, j = 0 \ldots N$ along the horizon from predicted obstacle positions $\boldsymbol{p}^{\mathscr{G}}_{\text{obs}}$, estimated position $\hat{p}^{\mathscr{G}}$, and previous solution $\boldsymbol{u}_{k-1}$. We have thus extended the constrained avoidance NMPC into a predictive track-and-avoid framework, fully integrated with onboard perception systems. Notably, as the higher level dynamics of the legged robot are relatively slow, this NMPC is sampled at 10 Hz as opposed to the 20 Hz for the UAV system. That also implies that the prediction horizon of $N = 40$ results in a 4 s prediction of both the obstacle and robot states.

The framework is evaluated in laboratory experiments, where once again the optimal way to view the results is through the real-time behavior of the robot which can only be seen in video form at the following link: `https://www.youtube.com/watch?v=_NaRNfnSmks`. A snapshot moment from one of the experiments is visualized in Figure 2.23 showing the approaching human to the robot and the predicted avoidance maneuver (red arrows) around the estimated predicted obstacle trajectory (green line). The red arrows, whose base represent the predicted robot position and arrow angle represents the predicted heading state, both avoid the obstacle and align the predicted heading state towards the predicted obstacle position.

Several scenarios are presented: stress testing the predictive tracking, avoidance of a static obstacle, a scenario when a rapidly moving human runs towards the BD Spot robot such that it must execute an aggressive track-and-avoid maneuver, and finally a meeting scenario where the human and robot are moving in opposite directions. The framework demonstrated very high performance even for an obstacle moving at an estimated 2.5 m/s which is a significant result for perception-based avoidance of dynamic obstacles. The formulation for predictive tracking also maintained the human in the field of view of the camera for all experiments. We used a safety distance $d_s = 1$ m, and extracted the minimum range measurements from the depth camera from the bounding box with the human in it to evaluate if that distance was maintained. In all experiments the distance was kept under 1 m except in one instant with the most rapidly moving obstacle (momentary estimated velocity of over 4 m/s and average approach speed of 2.5 m/s), where that safety distance was violated by 0.1 m. This is still a very promising result as the dynamic obstacle suffers from estimation errors which can of course lead to small violations of the safety distances regardless of the performance of the NMPC, or simply that the human obstacle was moving so fast that a complete avoidance maneuver was not possible based on the initial conditions. In general, the framework performed very well and serves as a demonstration of dynamic obstacle avoidance and visual tracking in a fully autonomous scenario with zero external support for the robot.

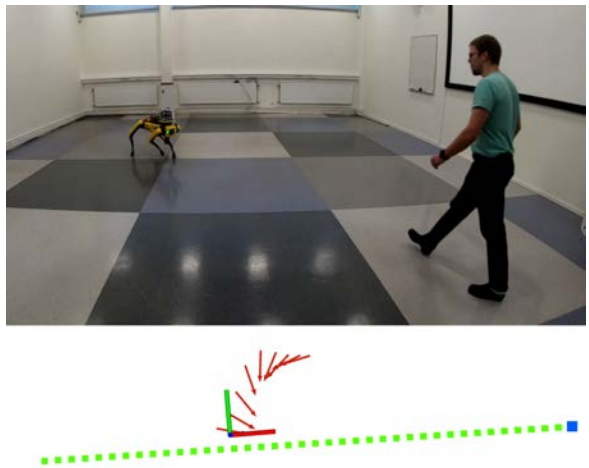


Figure 2.23: Example of a predicted track and avoid maneuver. The coordinate frame denotes the current robot pose, the red arrows denote the predicted robot poses that track and avoid the obstacle, while the blue dot and green line denote the current and predicted positions of the obstacle along the horizon.

2.8 Multi-agent Coordination

Coordination of multi-agent systems act as an enabler for increasing the efficiency of executing various tasks like inspection, exploration, or construction missions, as more agents can participate in the task. Coordination happens on multiple levels, where on the high level one might discuss the task assignment problem [90, 91] or swarm behavior [92], but to reach the stage of multiple agents operating in the same local space collaboratively the first step is agent-agent safety. This is often solved more as a more local problem, and in this section we will apply the constrained NMPC in this context. The fundamental difference to previous sections is the cooperativeness of the obstacles. In a multi-agent scenario we leverage the fact that we have control over all agents in the system (e.g. the obstacles), and also have information about how they are planning to move in the future. Two NMPC formulations will be presented: 1) A centralized NMPC (CNMPC) where a central computation agent orchestrates control actions for all agents in the system, where all agent states are augmented into one large system and coupled through the collision avoidance constraints between agents, and 2) a distributed formulation where each agent solves in principal the same NMPC problem as described in Section 2.7 with the difference that the obstacle trajectories are not predicted from a measured obstacle state, and are instead shared among the agents in the system. The distributed work will also address scalability through an obstacle prioritization scheme.

2.8.1 Centralized Collision Avoidance

The centralized scheme fundamentally only has two differences to the baseline NMPC framework. First, we collect the state and input vectors of all agents into one system of states as $x = [x^1, x^2, \ldots, x^{N_a}]$ and $u = [u^1, u^2, \ldots, u^{N_a}]$ where N_a denotes the number of agents in the system. The dynamic model for each agent stay the same as in (2.1). We can apply the same cost function as (2.3), penalizing deviations from state and input references as well as penalizing consecutive changes in inputs along the horizon, with the only difference being that each agent has a related state and input reference $x_{\text{ref}} = [x^1_{\text{ref}}, x^2_{\text{ref}}, \ldots, x^{N_a}_{\text{ref}}]$ etc. In the same way input constraints (2.13) and inpute rate constraints (2.12) are still applied to all agents in the centralized scheme. The second difference is that the agent-agent collision avoidance constraints, in this work using the circle-type constraint (2.5), are formulated between agent states, not between robot and an "external" obstacle. As such we can rewrite the constraint as

$$C_{l,i}(\boldsymbol{x}_k) \;:=\; \left[r_{\text{c}}^2 \;-\; (p^i_{x,k+j|k} \;-\; p^l_{x,k+j|k})^2 \;-\; (p^i_{y,k+j|k} \;-\; p^l_{y,k+j|k})^2\right]_+ \;=\; 0. \quad (2.26)$$

Here the superscripts i and l denote non-duplicate unordered pairs of agents (e.g. a constraint between i,l is equivalent to one between l,i so both do not need to be included) and $i \neq l$. As a note the total number of agent-agent safety constraints would then equal the (N_a-1)-*th* triangular number (1,3,6,10,15,21, etc.). In this work we used the 2D circle-type constraint (2.5) as opposed to a spherical one with the motivation that preferably in dense swarms agents should not fly on top of each other as the propeller wake might cause a destabilizing effect. We can still also apply and use the same obstacle constraints as previously discussed, but now formulated for each agent in the centralized scheme such that the agents must avoid the obstacle while maintaining a safe distance from all other agents.

This scheme was first evaluated in baseline simulations, and then in laboratory experiments using the Crazyflie Nano Quadcopter. For the simulations, the state update model is the same as the nonlinear discretized prediction model of the system with an addition of a Gaussian noise parameter. This noise represents a general uncertainty in state data, as well as an uncertainty in how the vehicles behave based on a certain input. Adding noise to the state update forces the optimizer to make realistic micro-adjustments to compensate. The noise parameter is generated with a normal Gaussian distribution with a specified mean, μ, and standard deviation, σ^2 as $\mathcal{N}(\mu, \sigma^2)$ [93]. The noise added to each state are the IID (independent and identically distributed) processes $\eta_p \sim \mathcal{N}(0, 0.01)$, $\eta_v \sim \mathcal{N}(0, 0.005)$ and $\eta_{\theta,\phi} \sim \mathcal{N}(0, 0.001)$, where η_p, η_v, and $\eta_{\theta,\phi}$ are the noise added to the position, velocity, and attitude terms respectively.

The first simulation scenario set-up can be seen in Figure 2.24.

Agents (here denoted as MAVs (Micro Aerial Vehicles) following the notation in the published work) must move past an obstacle while also keeping a safe distance from each other. This scenario can also be used to see how the `OpEn` solver scales with an increasing number of constraints and decision variables (control inputs) by adding more and more agents to the sides such that the innermost agents must avoid the obstacle and all outer agents must compensate in their trajectories to keep a safe distance. We analyze the up-scaling of the centralized scheme on the computation time of the optimizer and on how the constraint violations increase in the

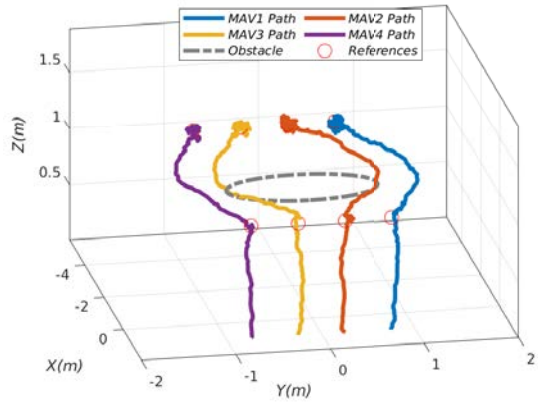


Figure 2.24: A multi-agent scenario used of simulation evaluations, combining obstacle and agent-agent avoidance. The scenario was also extended by adding more agents to each side of the obstacle.

set up scenario for more and more added agents. This data can be seen in Figure 2.25, where we ran the scenario for up to nine agents. The solver time shows both the average and maxi-

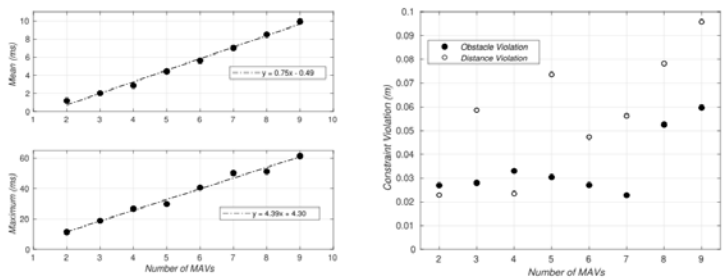


Figure 2.25: Computation times and constraint violations as the number of agents in the scenario is scaled up.

mum throughout the simulations, where the interesting parameter is really the maximum e.g. in the instants of intense avoidance where all penalty method iterations are applied. We can see that the scheme could maintain solver times under the 50 ms bound for up to seven agents with the eights just slightly above, using a prediction horizon of $N = 30$. An interesting result is that the `OpEn` based solver showed a linear relationship between computation times and number of agents, while a more standard SQP-based solver `fmincon` in another work showed an exponentially increasing solver time in a similar centralized scheme [94]. In terms of constraint violations, which are divided into the agent-agent distances and agent-obstacle distances, the scheme shows consistent results up to around seven agents where the violations start to in-

crease significantly, perhaps denoting the limit of `OpEn` in terms of the size of the optimization problem. As a note, for nine agents there would be 810 decision variables ($3 \times N \times N_a$), and 1080 agent-agent safety distance constraints ($36 \times N$) as we must form such constraints along the horizon.

A second scenario where four agents are commanded to simultaneously move through a center point such that all agents must avoid each other precisely, is visualized in Figure 2.26. The figure also shows the resulting computation time which peaks right at the start when the initial trajectories must be orchestrated and then rapidly drops off. We are using the previous solution $\boldsymbol{u}_{k-1}$ to hot-start the solver which greatly speeds up the convergence after the initial orchestration. The desired 0.4 m safety distances are very exactly maintained with very minor constraint violations due to the added simulation noise, and the figure shows how at the critical point in the simulation all agent almost perfectly align with the other agents at the desired 0.4 m safety distance.

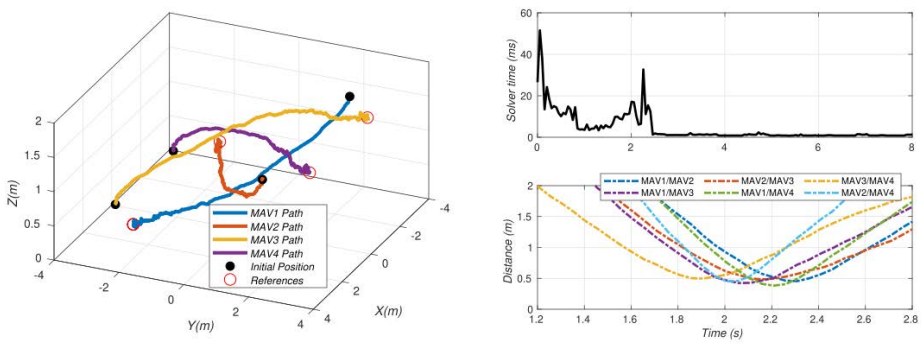


Figure 2.26: Simulation scenario with four agents all commanded to move through the center (left), and the resulting computation times and agent-agent safety distances for all agent pairs.

The centralized scheme was also evaluated in laboratory hardware experiments. Figure 2.27 shows the experiment architecture where the Vicon MoCap system provides the position and quaternion information for all agents, which trough a median filter for the velocities and proper coordinate transforms provides the state vector $\hat{x}$. The related references for all agents are here given by an operator, but could be given by any higher level mission module that coordinates the agents for inspection or other missions. Computation happens on a central laptop, and each agents receives its control signals through a Bluetooth transmitter, using the CrazyflieROS [95] client, based on the Robot Operating System (ROS) [96].

A series of experiments using the CNMPC can be found at: `https://www.youtube.com/watch?v=dJRe_ETvxx0`, where four agents fly in formation and periodically swap position with each other resulting in a stressful collision avoidance scenario. The framework demonstrated very efficient and consistent avoidance while delivering smooth control behavior. We increased the safety distance to 0.6 m in the experiments, and the agent-agent distances are

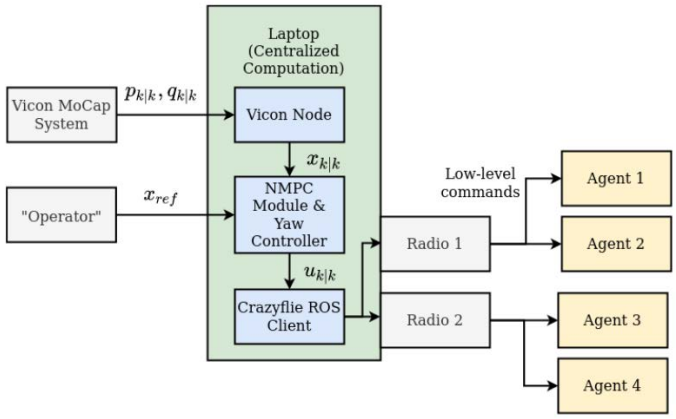


Figure 2.27: Centralized set-up for experiments with four Crazyflie Nano Quadcopters.

shown in Figure 2.28 where we can see how during the most critical moments all agent-agent distances almost perfectly maintain the desired 0.6 m distance, with a maximum violation of the safety distance of 5 cm.

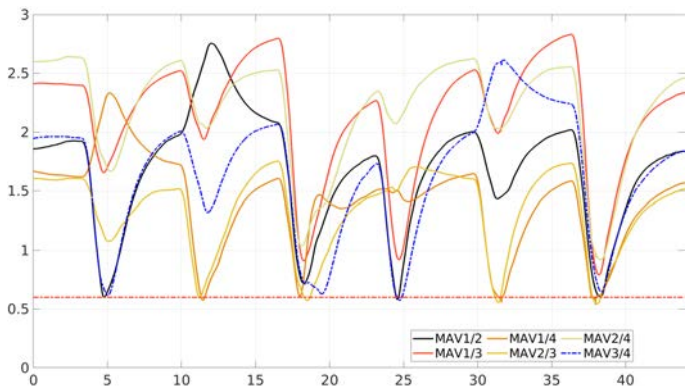


Figure 2.28: Agent-agent safety distances for all agent pairs during hardware experiment for centralized collision avoidance using four MAVs.

2.8.2 Distributed Collision Avoidance

The final NMPC use-case included in this research thesis is a distributed formulation of agent-agent collision avoidance. The underlying formulation from the NMPC side is close to identical to the dynamic obstacle avoidance NMPC discussed in Section 2.7 for the so-called ego agent

of interest, but instead of a obstacle prediction model the agents in the system share their predicted trajectories with each other in real time providing detailed information about where each agent plans to move in the future.

One of the central questions and contributions of this section is trying to solve the problem of scalability for a NMPC-based avoidance scheme. A common limitation of existing MPC approaches is that since the number of constraints (or potential-like terms) of the underlying optimization problem must remain constant during runtime, there can be situations where there are too many agents in proximity to the ego vehicle. Moreover, selecting the closest neighbors (as in [66]) may disregard those — potentially more remote — agents that are on a direct collision course with the ego agent. Assuming a system composed of N_a agents, ideally each agent should be able to form obstacle constraints with all other agents, such that the number of obstacles $N_{\text{obs}} = N_a - 1$. Due to limitation in computation power and the speed of optimization algorithms, this is not always possible for high numbers of agents. Instead, we need to choose $N_{\text{obs}} < N_a - 1$ necessitating some kind of obstacle prioritization where the ego agent takes into account a limited number of N_{obs} other agents. Assuming access to the predicted trajectories, $\boldsymbol{u}_{k-1}^{\text{obs},i}$ and measured states $\hat{x}_k^{\text{obs},i}$ of nearby agents, we want the prioritization scheme to be fully based on the motion intentions of each agent. For this purpose we propose Algorithm 1 as the obstacle prioritization scheme, which can be described as follows:

- Using the NMPC prediction model we can describe any $x_{k-1+j|k-1}$ from $\boldsymbol{u}_{k-1}$ and $\hat{x}_k$, and similarly using the shared NMPC solutions $\boldsymbol{u}_{k-1}^{\text{obs},i}$ and $\hat{x}_k^{\text{obs},i}$ to describe $x_{k-1+j|k-1}^{\text{obs},i}$ for $i = 1 \dots N_a - 1$ denoting which agent is considered.

- Calculate the predicted Euclidean distances between the ego agent and all other agents at the current and future time instants $j = 0, \dots, N$

- An agent is prioritized if the distance is below a threshold specified by $r^{\text{obs},i} + d_{in}$ where d_{in} represent the extra radius of influence of the prioritization scheme. This is done via the following weighted sum as a gauge for the prioritization

$$w_i = \sum_{j=0}^{N} \alpha(d_{i,j}, v_{k-1+j|k-1}^{\text{obs},i}) \beta(j),$$

  where $d_{i,j}$ denotes the distance to the *i-th* agent at time instant j. Let $\alpha(d_{i,j}, v_{k-1+j|k-1}^{\text{obs},i})$ and $\beta(j)$ be decreasing functions in $d_{i,j}$ and j respectively, and as such the scheme prioritizes agents that are at a closer predicted distance, and at fewer predicted time steps into the future.

- We also add an extra safety protocol by adding a large number K to w_i if the agents are closer than the obstacle radius $r^{\text{obs},i}$ at the current measured positions ($j = 0$) to always prioritize agents that directly violate the obstacle constraint at the current time instant.

- Obstacle trajectories $(\xi_j^{\text{obs},i})_{i,j}$ (the obstacle trajectory parameters used as an input to the NMPC along the horizon) are then sorted by the corresponding values in $\boldsymbol{w}$ in descending order, to prioritize the N_{obs} trajectories that produced the largest sums w_i.

For the weight functions we propose simple expressions with the desired functionality, and in this article we are using $\alpha(d,v) = (1 - \frac{d}{r^{\text{obs}}+d_{in}})^2\|v\|$ (velocity compensation since faster moving obstacles spend fewer time instants in $r^{\text{obs},i} + d_{in}$) and $\beta(j) = \frac{N}{(j+1)^a}$ with a being a tuning constant to describe the relative emphasis on closer versus more distant time instants.

Algorithm 1: Obstacle Prioritization

Inputs: $\hat{x}_k, \boldsymbol{u}_{k-1}, \hat{x}_k^{\text{obs},i}, \boldsymbol{u}_{k-1}^{\text{obs},i}, N_a, N_{\text{obs}}, \boldsymbol{r}^{\text{obs}}, d_{in}$

Result: From the shared trajectories and measured states decide which N_{obs} agents should be considered as obstacles

for $i = 1, N_a - 1$ **do**

  for $j = 0, N$ **do**

    Compute $p_{k-1+j|k-1}$, $p_{k-1+j|k-1}^{\text{obs},i}$ and $v_{k-1+j|k-1}^{\text{obs},i}$

    $d \leftarrow \|p_{k-1+j|k-1} - p_{k-1+j|k-1}^{\text{obs},i}\|$

    $v_{\text{m}} \leftarrow \|v_{k-1+j|k-1}^{\text{obs},i}\|$

    if $(d \leq r^{\text{obs},i})$ *and* $(j = 0)$ **then**

      $w_i \leftarrow w_i + K$

    else if $d \leq r^{\text{obs},i} + d_{in}$ **then**

      $w_i \leftarrow w_i + (1 - \frac{d}{r^{\text{obs}}+d_{in}})^2 v_{\text{m}} \frac{N}{(j+1)^a}$

Sort in descending order: $(\xi_j^{\text{obs},i})_{i,j}$ by corresponding element in w_i

$(\xi_{\text{prio},j}^{\text{obs},i})_{i,j} \leftarrow [(\xi_j^{\text{obs},1})_j, (\xi_j^{\text{obs},2})_j, \ldots, (\xi_j^{\text{obs},N_{\text{obs}}})_j]$

Output: $(\xi_{\text{prio},j}^{\text{obs},i})_{i,j}$

Thus in the optimization problem, only the prioritized obstacle trajectories described by $\xi_{\text{prio}}^{\text{obs}}$ are included in the optimization problem as collision avoidance constraints for dynamic obstacles. In the following evaluations, we are using a low number of obstacle constraints $N_{obs} = 3$ while using up to ten agents $N_a = 10$ to validate the proposed prioritization scheme for dense robotic swarms, while keeping the computational complexity low.

In this work we also used a different and novel way to solve for the collision avoidance constraints implemented in `OpEn`, namely an outer Augmented Lagrangian Method (ALM) [25] that still uses PANOC to solve the inner problem. As the method is not a contribution of the author of this thesis, but of a co-author to the published paper [97], this section will not include a detailed description. Instead the reader is directed to the `OpEn` [25, 98] and the related work [97] for details. Fundamentally, we need to slightly re-define the spherical safety distance constraint to fit into the ALM as:

$$C_{\text{sphere}}(p, \xi^{\text{obs}}) = (r^{\text{obs}})^2 - (p_x - p_x^{\text{obs}})^2 - (p_y - p_y^{\text{obs}})^2 - (p_z - p_z^{\text{obs}})^2 \leq 0, \tag{2.27}$$

with $\xi^{\text{obs}} = [p^{\text{obs}}, r^{\text{obs}}]$. The obstacle avoidance requirement is equivalent to $C_{\text{sphere}} \leq 0$, that

ego agent position p is required to lie completely outside of the sphere defined by ξ^{obs}. And the resulting optimization problem $\mathbb{P}$ sovled by `OpEn` as:

$$\mathbb{P}(x_{k|k}): \underset{\boldsymbol{u}_k \in Z}{\text{Minimize}}\, f(\boldsymbol{u}_k; x_{k|k}) \tag{2.28a}$$

$$\text{subject to: } G(\boldsymbol{u}_k; x_{k|k}) \leq 0, \tag{2.28b}$$

where $f(\,\cdot\,; x_{k|k}) : \mathbb{R}^{3N} \to \mathbb{R}$ is a continuously differentiable function with Lipschitz gradient defined by the cost function and $G(\,\cdot\,; x_{k|k}) : \mathbb{R}^{3N} \to \mathbb{R}^{NN_{\text{obs}}}$ is a differentiable mapping with Lipschitz-continuous Jacobian, that represents the collision avoidance constraints. Still following the *sequential* or *single-shooting* approach as before where the state sequence is eliminated. Another minor addition is to add a terminal state cost to the cost function in (2.3) as $\|x_{\text{ref}} - x_{k+N|k}\|^2_{Q_t}$, which adds an extra cost to the state penalty but only at $j = N$.

The final addition to the general NMPC problem is an adaptive weights scheme for the cost matrix Q_x. Let us first of all define the vector of Lagrange multipliers as $\boldsymbol{y}_k$, described in detail in [97]. On a general level, the multipliers correspond to the penalty parameter q previously discussed and visualized in Figure 2.5. The Lagrange multipliers can be thought of in a similar way, where the optimal Lagrange multipliers, $\boldsymbol{y}^\star_k$ (the multiplier once the optimizer has converged), can be thought as indicators of how much the optimal trajectories need to "bend" to avoid the obstacles. The idea is to use $\boldsymbol{y}^\star_k$ to update the reference tracking weights in real time so that obstacle avoidance is prioritized over set point tracking for a smoother maneuver that does not as aggressively try to reach the set-point reference during the avoidance. Let Q_p define the first three diagonal elements of positive-definite weight matrix Q_x. We introduce a scaling factor that adapts Q_p from $Q_{p,\min}$ to $Q_{p,\max}$ based on Lagrange multiplier $\boldsymbol{y}^\star_k$ as follows:

$$Q_p = Q_{p,\min} + \frac{Q_{p,\max} - Q_{p,\min}}{\sum_{l=0}^{N_{\text{obs}}N} W_l y^*_{k,l} + 1} \tag{2.29}$$

where W_l is some weight that is decreasing with respect to elements in $y^*_{k,l}$ (l just denoting the length-index of the multiplier) that represents constraints at more distant future time instants, which in our formulation results in $W_l = b(1 - \frac{l \mod N}{N})$ where b is a tuning constant. This heuristic seems to work well in practise. All elements in Q_p are scaled by the same factor, but it is enough for reducing the general emphasis on reference tracking. Note that the terminal state penalty still promotes the UAV to be as close to its end goal as possible but only at $j = N$.

The distributed scheme is evaluated in laboratory experiments. As should now be standard for the reader, the best way to view the results is through the experiment video at: `https://www.youtube.com/watch?v=3kyiL6MZaag`. Three scenarios are included 1) formation flight of nine agents that move one-by-one shown in Figure 2.29, 2) a scenario with ten agents that all move at once creating a very stressful avoidance scenario shown in Figure 2.30, and 3) a scenario where a non-cooperative velocity obstacle in included into the scheme (and its velocity obstacle trajectory is included in the prioritization algorithm) using the concepts from Section 2.7, where eight agents must avoid the non-cooperative obstacle while also maintaining a safe distance to each other. The usual and most crucial parameter for evaluations, the minimum agent-agent safety distances for all experiments, is shown in Figure 2.32.

For the first two experiments, the 0.4 m desired safety distance is maintained with a maximum constraint violation of 3 cm, while for the non-cooperative obstacle scenario, that necessarily will have lower performance due to the trajectory information not being as accurate (velocity obstacle vs. shared trajectories) for the non-cooperative obstacle, there was a violation of 7 cm. This was still not enough to cause a collision among the agents, which in Figure 2.32 is denoted as the safety-critical distance. In general, the obstacle prioritization scheme always correctly assigns the prioritized obstacles to be the agents on the most dangerous collision course with the ego agent despite the high number of agents and low number of constraints, demonstrating the scalability of the scheme. The distributed formulation demonstrated high-performance collision avoidance in dense aerial swarms, always generating consistent maneuvering without agents getting stuck or moving back-and-forth.

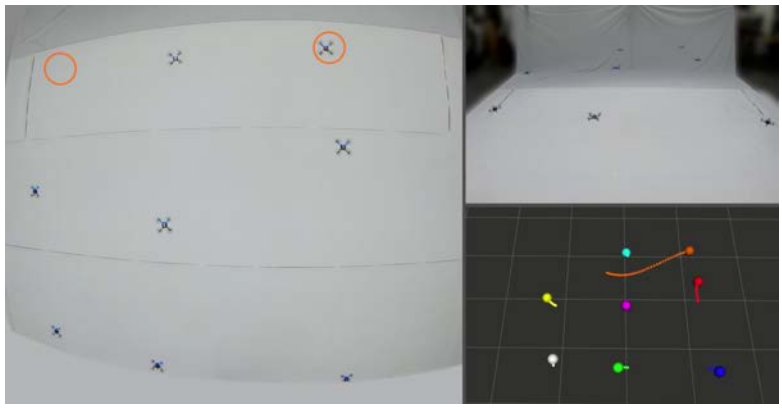


Figure 2.29: Experiment where agents move one-by-one while maintaining a formation.

As a final demonstration, the distributed NMPC was also deployed as part of a full mission scenario, on ground robot platforms. Here the higher level mission is related to reactive real-time task assignment [91] for multi-agent systems, and the distributed NMPC acts as an enabler for the experimental evaluation as a local agent-agent safety layer (while also acting as the regular set-point tracking controller when no collisions are imminent). The exact same NMPC problem is solved but for a very simple kinematic model of a ground robot. Ground robots commonly accept high-level actuation commands as $u = [u_v, u_\omega]$, where u_v is a forward/backward velocity command and u_ω is an angular velocity command. As such, we can use a simple nonlinear kinematic model of the robot as:

$$\begin{aligned} \dot{p}_x(t) &= \cos\psi(t) u_v(t) \\ \dot{p}_y(t) &= \sin\psi(t) u_v(t) \\ \dot{\psi}(t) &= u_\omega(t) \end{aligned} \tag{2.30}$$

with p denoting the global-frame position coordinate and ψ denoting the heading angle state

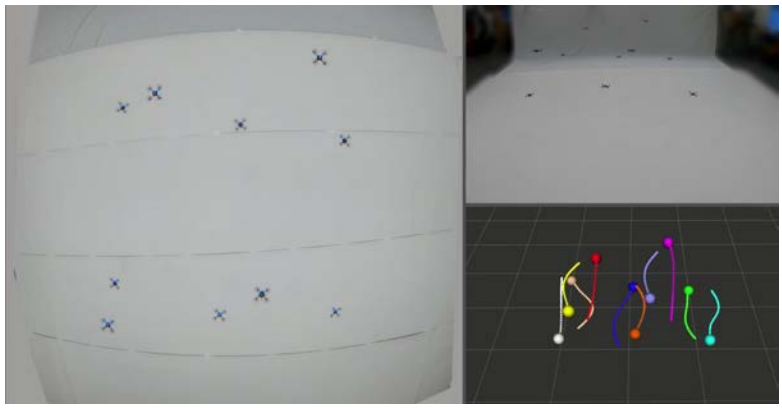


Figure 2.30: Two teams of five agents swap positions - stressing the prioritization scheme.

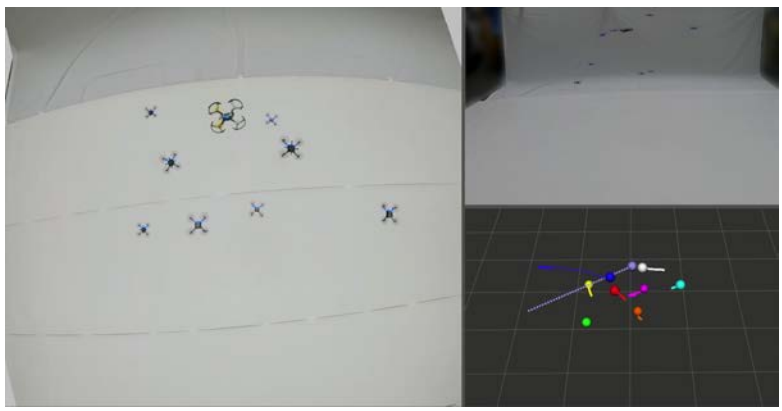


Figure 2.31: Introduction of a non-cooperative agent - now agents must avoid it and other agents in the collaborative system.

e.g. the robot states are $x = [p_x, p_y, \psi]$. We note that the robot velocity actuation u_v is in its body-frame. The task assignment framework is not part of this thesis, but the following video link: `https://www.youtube.com/watch?v=ZdEkoOOlB2g&feature=youtu.be` demonstrates a real use-case where the distributed NMPC is the perfect fit for reactive collision avoidance in tight spaces for multi-agent systems. The experiment set-up scenario of task assignment in a maze-like laboratory environment is shown in Figure 2.33 with four ground robot agents that are tasked with "pick up and deliver" tasks that are randomly generated in the maze. Figure 2.34 also shows how the desired safety distances are maintain through the experiment run, showing the minimum agent-agent distances at all times during the mission.

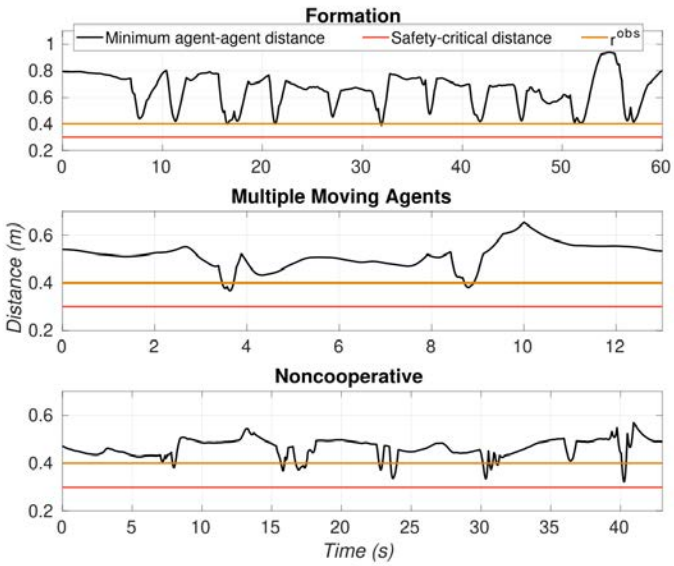


Figure 2.32: Safety distances for the three performed experiments, showing the momentary smallest agent-agent distance throughout the experiments.

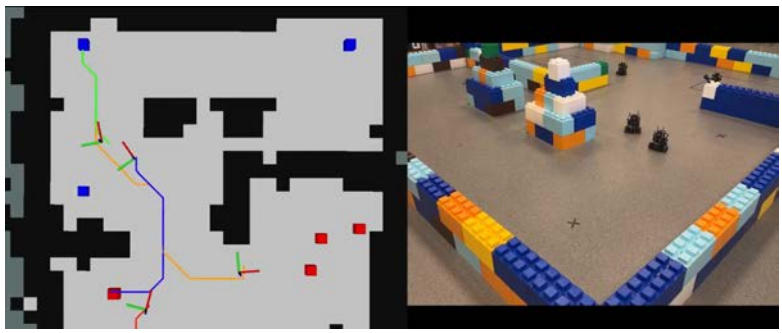


Figure 2.33: Mission set-up for task assignment: four ground robots are assigned ”pick and place tasks” (red/blue boxes as available tasks and drop-off locations) at random points in the maze. While following their planned paths (colored lines) the distributed NMPC scheme is used to avoid agent-agent collisions during the high-level mission execution.

2.9 Concluding Remarks

This chapter has conclusively demonstrated through a variety of use-cases that a constrained NMPC, where set-exclusion constraints represent obstacles in the environment, can deliver high performance collision avoidance, control, and local predictive path planning for aerial,

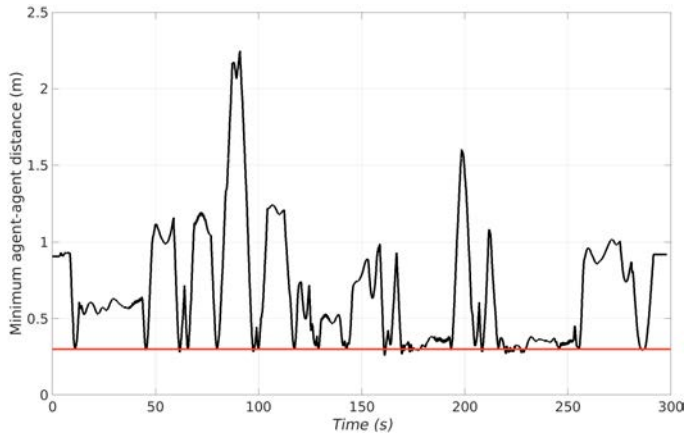


Figure 2.34: Minimum agent-agent safety distances for ground robots during the task assignment mission.

legged, and ground robots. Importantly, the chapter has also addressed the seamless integration of the NMPC framework with fully autonomous robotic systems where obstacle information is provided by onboard perception systems. The perception-actuation link was demonstrated for both static obstacles detected by 2D LiDAR, and dynamic moving obstacles in the human-robot safety context with an RGB-D camera. The NMPC, based on the Optimization Engine, also showed very promising results for multi-agent system safety, showing scalability and applicability to dense robotic swarms. All these results are validated with real hardware experiments with a focus on the NMPCs capability to maintain safety guarantees, not only in theory but also in practise, through maintaining specified safe distances to obstacles and other agents in the local environment. Although promising, a fundamental challenge for the use of this type of framework for field deployment in real mission scenarios, is the reliance on good obstacle information from the environment. Further research needs to be made on detection and integration of 3D obstacles that can represent the surrounding environment geometry, or linking the very general and commonly used occupancy map [99] to the obstacle avoidance constraints, which would allow the NMPC framework to be deployed as a local path planner in any environment. This main limitation segways well into Chapter 3, that focuses on fully reactive obstacle avoidance and navigation behavior without the need for a dedicated perception layer through an Artificial Potential Field (APF)-like formulation, which despite perhaps having lower performance than the works in this chapter, enables safe deployment in real field conditions.

CHAPTER 3

Navigation based on Fully Reactive Artificial Potential Fields

3.1 Overview

This chapter will present the development and use-cases of a reactive artificial potential field (APF) for collision avoidance and integrated navigation behaviors. The main attraction of this framework, as opposed to the NMPC discussed in the previous chapter and to most other navigation methods in the literature, is that it is formulated to operate directly on the raw sensor data (3D LiDAR) therefore removing the perception layer completely from the navigation problem. Doing so removes the largest error factor and the largest challenge when it comes to robot navigation; unreliable or momentarily failing perception whether that is occupancy mapping, object detection, or a momentarily poor scene recognition or unfamiliarity in a learning-based framework, that might then lead to the robot colliding with its environment. Especially in UAV missions, where one single interaction with the environment will lead to a crash and an end of the mission, the method described in this chapter proved highly effective as a fail-safe collision avoidance layer. The framework can be paired with any higher level module for inspection, exploration or other missions, and became an enabler for research into other navigation methods in real-world experimentation, as it could ensure that there would be no collisions with the environment during testing allowing faster prototyping. This was the approach taken in the later Chapters 4 and 5 where the framework described in this chapter was used as the local autonomy and navigation stack. In addition to avoiding collisions with the environment, this chapter will also investigate using this kind of fully reactive APF to manage multi-agent and mixed traffic (human-robot) scenarios, as well as integrating both reactive exploration of subterranean tunnels and infrastructure inspection behavior into the ”perception-free” navigation. This enables the robot to execute relatively complex behaviors using one simple algorithm that does not rely on assisting perception modules (occupancy mapping etc.). Additionally, due to the fast nonlinear dynamics of UAV (rotor) flight, those systems are also particularly susceptible to oscillating or twitching flight behavior while performing, for example, obstacle avoidance

maneuvers. As such, the developed reactive APF and controller framework was developed to facilitate smooth and stable flight behavior during maneuvering close to walls and obstacles, while also keeping computation times low which is a must for a fail-safe collision avoidance layer.

3.2 Introduction

For any remote operation or fully autonomous robotic mission, one of the most critical components has to be the ability for the robot to use onboard sensor data to avoid collisions with the environment and to execute mission-completing navigation behavior, and as such obstacle avoidance algorithms, reactive or local path planning, and any perception-based actuation have been studied extensively [100]. There exist a wide range of solutions to the general robot safety problem while some of the legacy approaches include the Dynamic Window approach [101] and the classic Artificial Potential Fields [34], which is still the basis for continued research today [102, 103], and of course also the basis for the method described in this chapter. In the last years though, the research is gravitating more towards modern state-of-the-art methods such as barrier functions [104], MPC [105] or deep-learning [106]. In this context, it might be the case that these methods, including the NMPC discussed in Chapter 2, theoretically outperform the legacy methods such as the classic Artificial Potential Fields. But, a more complex method can also require more or better perception information, and in harsh field environments where robustness and resilience to degraded perception information is critical, simplicity can often be undervalued.

For real robot deployment where one can not ”hack” the perception by Motion-Capture systems or similar, one of the main underlying problems in safe robot navigation is linking smart obstacle avoidance algorithms with onboard perception information, as for example: visual [107], LiDAR [108], learning methods using camera information [109], or novel fast local map-based planning [110]. Many modern solutions perform exceptionally well, but for many of them the question remains on how to reliably extract the necessary environment information from the raw sensor data in order to make those methods work optimally, and what happens if that extraction process (bounding box detection, obstacle classification, occupancy mapping) momentarily fails in a critical moment. Similarly, the multi-agent safety algorithms both centralized [111] and distributed [97] presented in the previous chapter require a constant line of communication between agents in the system and knowledge of their relative pose. Just as momentarily poor object detection can cause an environment interaction, a momentary drop in communication (pose and trajectory sharing) could cause a collision among agents.

In the same direction, almost all algorithms for robot exploration (navigating through unknown territory) are based on occupancy maps of the environment where popular 3D version are the OctoMap [99] and VoxBlox [112] frameworks. Robot exploration will be discussed in more detail in Chapter 4, but in short the most commonly used methods are the frontier exploration [113] and various sampling-based solutions [114]. These methods rely on classifying areas in the map as high in information gain (unknown territory) and planning safe paths based on a procedurally generated occupancy map of the area to the best such location (based on some predefined heuristic). In this chapter we will present a different approach to robot explo-

ration in subterranean environments, similar to the approaches described in [115] (which were the main motivators for extending the APF to exploration behavior), that does not use a map of the environment but instead generates continuous forward waypoints while the robot heading is aligned to the open areas. In this chapter that behavior is achieved by generating repulsive forces on the heading state from LiDAR hits in front of the robot. Especially in subterranean tunnel areas this method proved highly effective (but less general than the frontier- or sampling methods) while again doing so only based on the raw LiDAR pointcloud.

Finally, inspection of infrastructure is one of the main applications of autonomous robots. Here the challenge is generating safe inspection trajectories around the structure. Doing so in an offline approach assuming previous knowledge of the structure is very common, and has seen use in field applications [116] of inspection of wind turbines. But this is not always possible, and inspection of unknown and geometrically fractured objects (e.g. not simple smooth surfaces) can be very challenging. This can be achieved in a similar manner to exploration of unknown areas, for example by generating frontiers with the goal of complete 3D coverage of the structure [117, 118]. Other novel more reactive methods utilize the generation of safe visual viewpoint poses to follow a surface [119], which is more in line with the desired behavior presented in this chapter. Here we will utilize a rotational potential field to generate smooth and continuous safe waypoints that can inspect a fractured uneven structure.

3.3 Contributions

Towards providing fully reactive solutions to the above described scenarios, the contributions of the work described in this chapters are:

- A generalized artificial potential field formulation that can work directly with raw pointcloud data. This decouples the problem of robot safety from the reliance on any map, object detection, or similar software. We also apply a series of practical additions to the generated forces to further enforce smooth and stable navigation without introducing oscillations (or similar) to the UAV flight behavior.

- We introduce an adaptive weight scheme that adapts the flight behavior based on the generated repulsive forces, as the UAV should, in general, move more carefully when in the presence of obstacles, humans, or other robotic agents. We also include an adaptive safety radius that scales the radius of influence based on the velocity of the UAV. This both allows the UAV to safely move with large velocities by increasing the radius of influence, and at the same time helps the UAV to fit in-between obstacles without getting stuck.

- The general APF solution is developed towards a series of use-cases. 1) Using the reactive APF as an inner fail-safe collision avoidance layer, 2) demonstrating the use of the fully reactive APF in human-robot scenarios and multi-robot scenarios, 3) An extension of the general APF solution for reactive exploration of tunnel areas, 4) An extension using a rotational APF for following surfaces to inspect infrastructure.

- Finally, we evaluate the scheme in a variety of laboratory experiments and field trials in order to demonstrate how this APF formulation can provide fail-safe navigation in critical obstacle avoidance situations as an inner reactive safety layer for any higher-level mission scenario, and towards mixed-traffic, exploration, and inspection missions. To enable field deployment this chapter will also discuss a complete platform-sensor-autonomy kit that will the the baseline both in this chapter and in following chapters for field deployment.

3.4 Fully Reactive Collision Avoidance

3.4.1 Force Field Formulation

This work will focus on a similar force/potential formulation as in the fundamental works on Artificial Potential Fields [34, 120], with the difference of not considering an obstacle as a single point or surface, and instead considers that each point detected within a specified radius of influence r_F should generate a weak repulsive force, and then summing all such forces to get the total. Assuming a pointcloud input, in this paper from a 3D LiDAR, as $\{\boldsymbol{P}\}$ consisting of points relative to the LiDAR frame as $\rho = [\rho_x, \rho_y, \rho_z]^T$. The subset of such points inside r_F is $\boldsymbol{\rho}_F \in \{\boldsymbol{P}\}$ where $\| \rho_F^i \| \leq r_F$ and $i = 0, 1, \ldots, N_{\rho_F}$ are the points that we want to use to generate a repulsive force. The fundamental idea is to generate repulsive forces in the opposite direction to the detected point inside r_F, and we can denote this as the *linear* component to the APF, always pushing the UAV away from obstacles/walls/etc. The expression for the linear repulsive force is:

$$F^{r,lin} = \sum_{i=1}^{N_{\rho F}} L^r (1 - \frac{\| \rho_F^i \|}{r_F})^2 \frac{-\rho_F^i}{\| \rho_F^i \|}, \tag{3.1}$$

with $F^{r,lin} = [F_x^{r,lin}, F_y^{r,lin}, F_z^{r,lin}]^T$ denoting the linear repulsive force and $L^r = \text{diag}(L_x^r, L_y^r, L_z^r)$ denoting a diagonal matrix of repulsive gains (someone experienced in APFs will see that this term represents a collection of model-based terms that can be included in the problem, that are not of interest here). From (3.1) it is also clear that values in L^r represents the maximum force-per-point and that the repulsive force increases as the relative distance between the detected point and the LiDAR decreases, while it is zero at the boundary of $\| \rho_F^i \| = r_F$ ensuring a smooth transition into the radius of influence.

Next, we add an inner set of points defined by critical safety radius r_c, such that $\boldsymbol{\rho}_c \in \{\boldsymbol{P}\}$ where $\| \rho_c^i \| \leq r_c$ and $i = 0, 1, \ldots, N_{\rho_c}$. The idea is to impose a large static force to any point, again in the opposite direction, within the critical radius imposing a direct boundary on safety distances. The sum of such static forces are defined as $F^{r,c}$, e.g. the *critical* repulsive force, as:

$$F^{r,c} = \sum_{i=1}^{N_{\rho_c}} L^c \frac{-\rho_c^i}{\| \rho_c^i \|}, \tag{3.2}$$

with diagonal matrix L^c as the critical static force-per-point inside r_c. Clearly, $F^{r,c}$ is an absolute necessity for guaranteeing safety from smaller obstacles whose linear force response

would not be sufficiently large to result in a proper avoidance maneuver (ex. hanging wires or thin metal beams). Similarly, enforcing an inner safety radius is critical in interactions with other robots or humans.

Only using linear components to the APF works very well for maintaining safe distances to obstacles, but only being pushed *away* and not *around* obstacles decreases the ability for the APF to on its own avoid and move past encountered obstacles. The idea is to add only a weak rotational component in the *xy*-directions to help the UAV better move around obstacles, with the magnitude being a fraction of the linear force in the other direction (*x*-*y*), and the sign of the rotational force is determined by the sign of the linear force in the same direction. The result is:

$$F_x^{r,rot} = L^{rot}\,\mathrm{sgn}(F_x^{r,lin})\|F_y^{r,lin}\|, \tag{3.3a}$$

$$F_y^{r,rot} = L^{rot}\,\mathrm{sgn}(F_y^{r,lin})\|F_x^{r,lin}\|, \tag{3.3b}$$

$$F_z^{r,rot} = 0. \tag{3.3c}$$

One should mention that if the scenario only calls for the APF to act as a safety layer and map-based path planning takes care of moving through the constrained environment, L^{rot} can be set to zero.

To get the resulting repulsive force F^r from all these components, they are summed as:

$$F^r = F^{r,lin} + F^{r,c} + F^{r,rot}. \tag{3.4}$$

Assuming the goal is to reach the next way-point $wp = [wp_x, wp_y, wp_z]^T$, generated by some higher level navigation module, we define the attractive force $F^a = [F_x^a, F_y^a, F_z^a]^T$ such that $F^a = wp^{\mathscr{B}} - \hat{p}^{\mathscr{B}}$, with $wp^{\mathscr{B}}$ and $\hat{p}^{\mathscr{B}}$ denoting a yaw-compensated coordinate frame measurement of the robot position $\hat{p}$ and the next way-point goal wp. From an intuitive point of view this can be seen as the attractive force F^a being the vector from the current position to the next given way-point with an unitary gain, while the repulsive force F^r is the shift in the next way-point required to avoid obstacles.

In order to promote smooth flight behavior and to make the scheme more resilient to state-estimation jumps or incorrect way-point inputs we propose Algorithm 1 that places saturation limits on the repulsive force, rate of change on the repulsive force, and normalizes the resulting attractive force and summed total force $F = F^a + F^r$. This also means that the APF will work just fine for maintaining a safe distance to obstacles even if wp is given very far away, inside an impassable wall, etc. further promoting the fail-safe aspect of the APF.

Algorithm 2: Force calculation

Inputs: F^a, F_k^r, F_{k-1}^r
if $|| F_k^r ||> F_{max}$ **then**
  $F_k^r \leftarrow sgn(F_k^r) F_{max}$
if $|| F_k - F_{k-1} ||> \Delta F_{max}$ **then**
  $F_k^r \leftarrow F_{k-1}^r + sgn(F_k - F_{k-1}) \Delta F_{max}$
if $|| F^a ||> 1$ **then**
  $F^a \leftarrow \frac{F^a}{||F^a||}$
$F \leftarrow F_k^r + F^a$
if $|| F ||> 1$ **then**
  $F \leftarrow \frac{F}{||F||}$
Output: F

A problem discovered in [121] with this type of APF (where it was used as a comparison method) is the difficulty in moving in-between obstacles. APFs generally work better if the radius of influence can be chosen relatively large so that the avoidance maneuver can start in time to avoid the obstacle, without the need to impose large repulsive gains. The problem is that choosing r_F large means that the UAV will never be able to pass in-beween obstacles or through narrow corridors, tunnels, etc. Towards solving this problem we propose an adaptive radius of influence r_F, and critical safety radius r_c based on the estimated velocity $\hat{v}$ of the UAV. Equation (3.5) shows the proposed adaptive radii.

$$r_F = r_{F0} + c_1 \log(|| v || +1) \tag{3.5a}$$

$$r_c = r_{c0} + c_2 \log(|| v || +1) \tag{3.5b}$$

r_{F0} and r_{c0} denote the smallest but still safe radii of influence (based on the size of the UAV and selection of repulsive gains), and as the velocity increases, the radii increase by the logarithm of the magnitude of the velocity as well as tuning parameters c_1, c_2.

3.4.2 APF-Controller Pairing

The artificial potential field framework is paired with a NMPC full-state controller fundamentally with the exact same formulation as in the previous Chapter 2, but without any of the collision avoidance constraints. As such, it will not be described in detail in this chapter as well, and in the following text we will use the same notation as in the previous chapter to describe aspects of the cost function/states etc. The only contribution to the controller pairing is a method for adapting the weights in the cost function to achieve smoother and more careful maneuvering around obstacles e.g. we want the control behavior to be slower when the repulsive forces are higher.

In general, the desired flight behavior when flying in open areas should be to move as quickly as possible, or at a specified desired velocity. But, when in the presence of obstacles, the UAV should slow down and move much more carefully as to avoid "bouncing", or back-and-forth behavior when in an obstacle-dense area. First, let us denote the state weight matrix as $Q_x = \text{diag}(Q_{p,x}, Q_{p,y}, Q_{p,z}, Q_{v,x}, Q_{v,y}, Q_{v,z}, Q_\theta, Q_\phi)$, or specifically elements related to the

position error as Q_p and the velocity error as Q_v. The idea is to adapt Q_p and Q_v based on the magnitude of the repulsive force between a minimum and maximum value. This can be seen as a "switching" controller but with a grey area between the two modes of obstacle-free or obstacle-rich areas. The selected expression for adapting Q_p and Q_v can be found in (3.6).

$$Q_p = Q_{p,min} + \frac{Q_{p,max} - Q_{p,min}}{1 + c_3 \mid\mid F_r \mid\mid}, \tag{3.6a}$$

$$Q_v = Q_{v,max} - \frac{Q_{v,max} - Q_{v,min}}{1 + c_4 \mid\mid F_r \mid\mid}, \tag{3.6b}$$

With only these terms, the system could become too slow in performing critical maneuvers when in the presence of obstacles. As such we added a similar term, shown in (3.7), to adapt the input change weight matrix $Q_{\Delta u}$, resulting in faster optimal change of direction, but still maintaining low velocity and careful movements. For all of these expressions c_3, c_4, c_5 are simply tuning constants in order to match the adaptiveness to the desired flight behavior change.

$$Q_{\Delta u} = Q_{\Delta u,min} + \frac{Q_{\Delta u,max} - Q_{\Delta u,min}}{1 + c_5 \mid\mid F_r \mid\mid}. \tag{3.7}$$

The more careful movement in the proximity of obstacles, combined with the adaptive radius of influence, greats assist the UAV in being able to smoothly navigate through narrow areas such as tunnels, openings, or in-between encountered obstacles.

3.4.3 Local Autonomy Kit

As the point of the developed framework is its real-world utilization of reactive navigation, we want to use a platform that can fulfill the onboard requirements for full autonomy, both in terms of the sensor payload and onboard computation power. The utilized UAV is a custom built quadrotor designed and built at Luleå University of Technology for full autonomous mission execution can be found in Figure 3.1, together with the complete hardware-software architecture used for fully autonomous experiments. The UAV has a maximal size radius of around 0.4 m. The UAV is equipped with a 3D LiDAR (Velodyne Puck Lite VLP16 or Ouster OS1-32) and a baseline Intertial Measurement Unit (IMU). Onboard computation is done via an Intel NUC - NUC10i5FNKPA. For state-estimation (SLAM) in GPS-denied environments we are using the tightly-coupled LiDAR-inertial odometry LIO-SAM [26] from which the framework utilizes the state vector $[p_x, p_y, p_z, v_x, v_y, v_z, \psi]$, while roll and pitch angles $[\phi, \theta]$ are provided by the onboard IMU. A downward-facing single-beam LiDAR measures the clearance R below the UAV. After the proposed local APF-NMPC reactive framework, low-level commands are fed to a Pixhawk Flight Control Unit (FCU) that generates motor commands for the UAV. The utilized implementation of the APF is in Python and ROS [96]. The denoted waypoint wp could come from any higher level module for exploration, path planning or inspection but in the following experiments it is simply provided by an operator giving the robot a specific waypoint goal to reach (with obstacles in the way).

Figure 3.1: 3D-LiDAR equipped UAV, and the complete LiDAR SLAM and local navigation architecture used during experiments.

3.4.4 Results

This section will detail the evaluation of the reactive APF for collision avoidance and safety. All experiments are performed in fully autonomous mode with computation, sensing, state estimation etc. running onboard the robot. First, we test in a laboratory setting with the goal of displaying and visualizing how each component introduced and added to the APF assists in allowing smooth and efficient avoidance maneuvers. The task is moving past a simple obstacle where the UAV is given a position reference waypoint on the other side of the obstacle. The experiment was repeated three times: 1) using only linear components e.g. the classic APF (Figure 3.2A), 2) with added force normalization, saturation, and rotational component (Figure 3.2B), 3) with adaptive weights and adaptive radii (Figure 3.2C). One can easily see the progression from the classic APF where we get a sudden jump and twitch, to the smoother and more rotational maneuver of the enhanced version. Adding the adaptive radii and adaptive weights in the third run allows for a faster maneuver where the UAV can still break early, but then move closer to the obstacle, as well as significantly speed up after the obstacle has been passed.

Additionally, to showcase how the adaptive radii assists in moving in-between obstacles Figure 3.3 shows an example experiment where the UAV is commanded to move through an opening of 1.2 m between two obstacles, first without (A) and then with the adaptive radii (B). In the first run, the radius is selected as low as possible while still capable of maintaining reasonable avoidance behavior ($r_{F0} = 1.1\,\text{m}$, $r_{c0} = 0.7\,\text{m}$), while in the second run we could reduce r_{F0} to be as low as 0.8 m and r_{c0} to as low as 0.55 m, allowing the UAV to pass through the 1.2 m opening while still avoiding collisions.

We also evaluate the APF in the field, in this case in a subterranean tunnel environment, that will be discussed to a greater extent in Chapter 5. In short, flying UAVs in subterranean areas has gained an interest both in the context of search-and-rescue due to the DARPA Subterranean Challenge [122, 123] but also for a variety of application in the mining sector [36]. Subterranean environments can be difficult to navigate due to narrow areas, and the constant proximity to walls and obstacles. The following tests are performed in realistic GPS-denied tunnel environment below Mjölkuddsberget, Luleå, Sweden.

These trials investigate two things: 1) a general obstacle course scenario in a real appli-

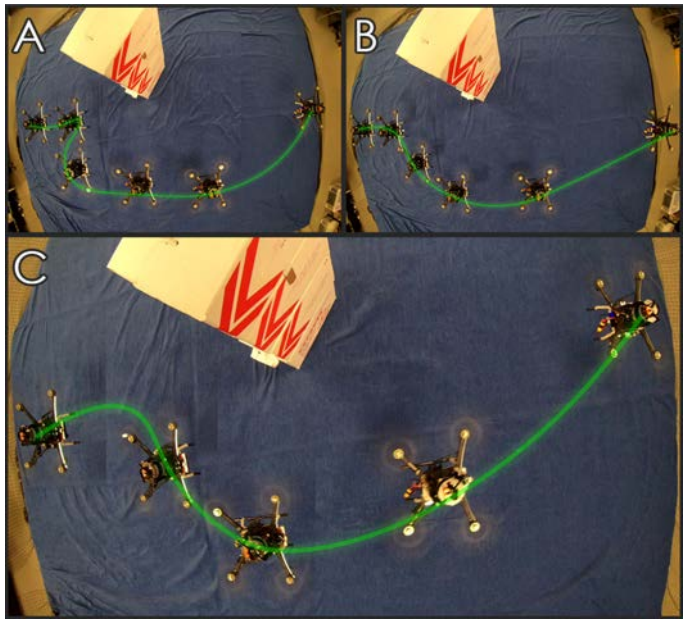


Figure 3.2: Improvements in avoidance paths around a simple obstacle for a) legacy APF, b) rotational and normalized, c) adaptive weights and radii.

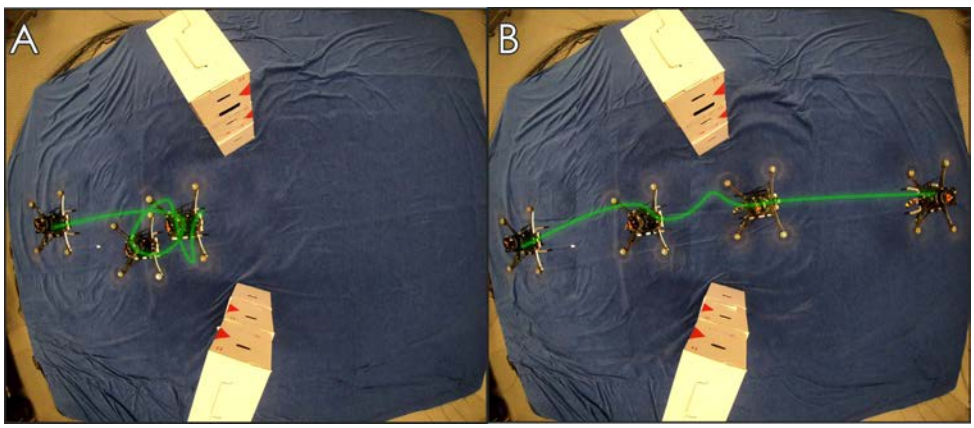


Figure 3.3: Moving through a narrow entrance (ca. 20cm clearance) first without (left) and then with (right) adaptive radii. Safety distances are selected to the minimal safe radius of influence to avoid collision in both cases.

cation environment and its associated challenges, and 2) the ability to withstand failure and incorrect waypoint generation from higher level modules exemplified by an experiment where

we give random waypoints very close to or inside the walls in a narrow subterranean tunnel area. These experiment runs can be seen in Figure 3.4, where the minimum measured distance from the environment was 0.88 m. For the obstacle course, the adaptive APF provided safe and effective navigation when given a single goal waypoint on the other side of the obstacles. In the random waypoint experiment, the UAV was given sampled *xyz*-position and yaw waypoints in a box of size $6 \times 6 \times 2$ m every 10 seconds for 70 seconds while tuned to move quickly/aggressively, in a 3.5 m wide subterranean tunnel area (as to generate waypoints inside or too close to the walls). The combination of force rate saturation and adaptive weights keeps the UAV stable despite the impossible-to-reach waypoint command. This is an underrated evaluation scenario in the literature, and in the field context when other modules are added on top of the APF to execute complex missions, as it is also the most likely scenario where the reactive avoidance layer has to save the robot from a crash.

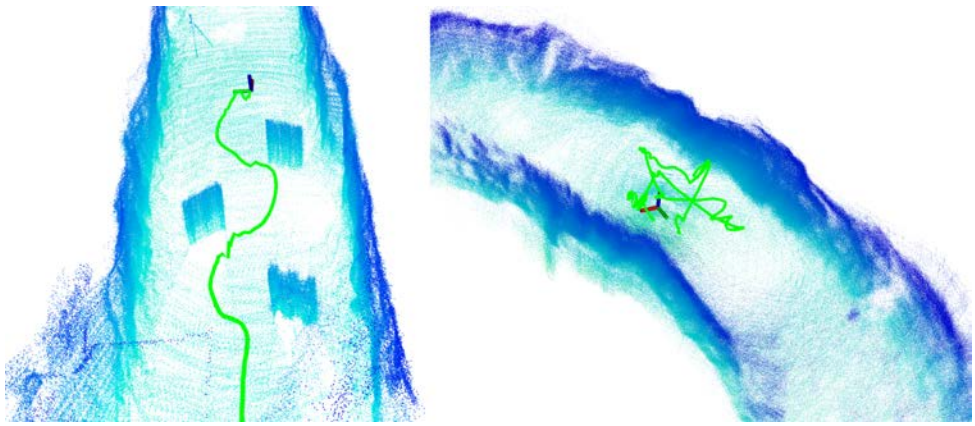


Figure 3.4: Field evaluations in a subterranean environment of APF-based reactive avoidance. Through an obstacle-filled tunnel (left), and in a scenario where random waypoints are generated in the tunnel too close or inside the walls (right) and a safe distance must be maintained.

Finally, the reactive collision avoidance framework is evaluated in scenarios of mixed traffic in two separate experiments: 1) a scenario for human-robot safety where a human "worker" is sharing the same operational space as the autonomous robot and is actively entering its local space, and 2) a scenario with two fully autonomous UAVs that are tasked to swap positions in a constrained laboratory environment. In general, the APF formulation is identical to the previous experiments but due to safety concerns the radius of influence has been increased, and in the multi-agent scenario the rotational gain L^{rot} was increased to facillitate a smoother avoidance maneuver as the two agents approach each other. It should again be noted that the two agents share no data and are in two completely separated global frames of reference, and the avoidance maneuver is generated solely from the 3D LiDAR repulsive forces as a result of LiDAR hits on the other agent. These two experiments are relatively hard to visualize in still

images as there are multiple moving parts, and as such the reader should place higher emphasis on the real-time behavior shown in the following video links for 1) human-robot interactions which can be viewed at `https://www.youtube.com/watch?v=3ViSPCbW9Rg`, and 2) for the scenario with two autonomous vehicles in the same operating space at `https://www.youtube.com/watch?v=cY6ISNW4dcw`. The experiment set-up can be seen in Figure 3.5. In both scenarios the APF generates safe collision avoidance maneuvers for the moving obstacles. We note that in the multi-agent scenario the two powerful UAVs are flying very close to each other and to the ground which creates air disturbances on them both, which can be seen as the slightly more unstable flight behavior in the video.

Figure 3.5: Mixed-traffic scenario evaluations. A human worker walks into the operating space of the UAV and avoidance maneuvers are executed (left). Set-up for multi-agent experiment: two large-scale 3D LiDAR equipped UAVs are flying in close proximity and are tasked to swap positions in a confined space (right).

In addition to the experiments discussed in this chapter, a large contribution to this fail-safe avoidance layer is its use in a variety of other publications and works. The framework can be seen as an enabler to more quickly go to the experiment stage for other navigation methods as it can provide a fail-safe in case the other ”higher level” module or its assiciated mapping/detection layer malfunctions, while also providing an efficient state-estimation and controller framework that can be used for almost any mission. The local autonomy kit has been used to enable path planning evaluations [124], frontier exploration of unknown areas [125–127], subterranean search-and-rescue [128, 129], and inspection of infrastructure [130]. It is also the core kit used in the upcoming Chapters 4 and 5.

3.5 Integrated Exploration & Inspection Behavior

This section will detail an extension of the collision avoidance methodology of using raw LiDAR data to generate repulsive forces into two other navigation behaviors namely reactive exploration of subterranean tunnels and inspection of infrastructure, meaning the ability navigate around the structure by following its surface. It should be highlighted right away that it is not likely that these methods can outperform the state-of-the-art exploration or inspection plan-

ners due to the simple reason that there is no higher level planning layer nor the use of a map of the area. The point is to provide alternative efficient *local* navigation methods that are very robust, simple, and only utilize the raw instantaneous sensor data to navigate. These could then be paired with higher level modules of mission planning for a more complete mission execution framework.

The integrated exploration behavior will utilize a "carrot chasing" approach where the UAV is provided continuous forward position references as an attractive force while the heading state is aligned to move towards the open areas (in this case the tunnel direction). Combining that with the previously discussed collision avoidance generates the motion primitive of "explore forwards towards the most open area while avoiding obstacles along the way".

We can consider the attractive force on the position states as $F^{a,exp} = [p_x^{\mathscr{B}} + L^{a,exp}, p_y^{\mathscr{B}}, p_z^m - p_z^l]$, simply a giving body-frame forward attraction, with L^a_{exp} having an appropriate magnitude to match the desired forward velocity. In this case $L^a_{exp} = 1$ for simplicity, since we still apply the force normalization of Alg 2. p_z^m denotes the desired mission height above the ground, and p_z^l denotes the local z-coordinate as the clearance height provided by the single-beam LiDAR measurement R.

The real addition to the method comes from applying a repulsive force on the heading state ψ from LiDAR hits in a conical shape in front of the LiDAR.

The subset of $\{\boldsymbol{P}\}$ of interest is $\boldsymbol{\rho}_\psi \in \{\boldsymbol{P}\}$ where $\arccos(\frac{\rho^i_{\psi,x}}{||\rho^i_\psi||}) \leq \theta_{co}$ and $|| \rho^i_\psi || \leq d_\psi$. Here θ_{co} represents the cut-off angle for the width of the cone in front of the UAV, and d_ψ is simply the maximum distance from the LiDAR we are interested in. Representing the number of points that pass both conditions $i = 0, 1, \ldots, N_{\rho_\psi}$. Utilizing a very similar force field function as before

$$F^{r,\psi} = \sum_{i=1}^{N_{\rho\psi}} L^\psi (1 - \frac{|| \rho^i_\psi ||}{d_\psi})^2 \operatorname{sgn}(-\rho_{\psi,y}), \tag{3.8}$$

with L^ψ as the repulsive gain. The direction of the force is in the opposite direction of the sign of the y-coordinate of ρ^i_ψ e.g. the heading state is repulsed to "look away" from obstacles or walls in front of the robot.

There is no specific attraction force for the heading state, and as such the heading references passed on to the controller becomes $\psi_{ref} = \hat{\psi} + F^{r,\psi}$.

Two examples are given in the Figure 3.6. The left image shows the UAV being repelled in its position by the wall (red) while the heading regulation (white) aligns its heading state with the tunnel direction. The red arrow denotes the pose reference (position and heading) sent to the controller. The right image shows how the heading state is repulsed from the structures ahead of the UAV towards the more open area.

The exploration-APF was evaluated in the same subterranean tunnel area as the experiments in Figure 3.4. Here we provide no operator waypoint wp but instead let the forward attraction and heading repulsion combined with the normal collision avoidance APF guide the UAV through the tunnel. The resulting exploration path can be seen in Figure 3.7 where we explored a curving tunnel that also included an area with obstacles along the walls and an open void-like area. The resulting exploration path was around 140 meters. We performed another shorter

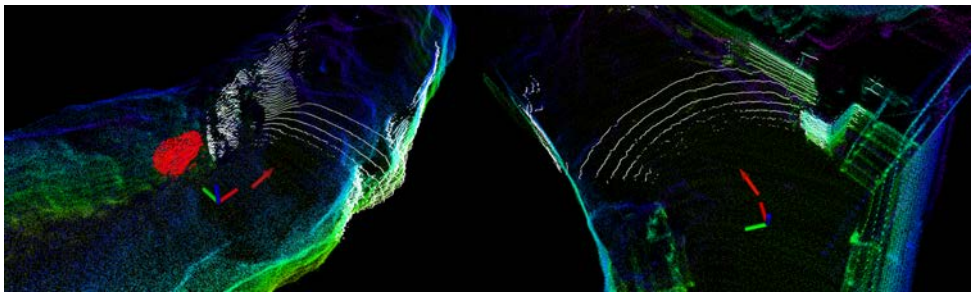


Figure 3.6: Examples of combined avoidance plus heading regulation. The red pointcloud denotes points $\boldsymbol{\rho}_F$ in the repulsive avoidance force, while the white points $\boldsymbol{\rho}_\psi$ denote the points included in the repulsive heading force.

run in a more narrow upward sloping tunnel, where the UAV also starts in a junction and has to enter the narrower area. This can be seen in Figure 3.8. In both instances the complete APF keeps the UAV safe in the middle of the tunnel and away from walls and obstacles while the heading repulsion technique aligns it towards the more open areas. A video demonstrating the real-time flight behavior can be found at `https://www.youtube.com/watch?v=c5knu3asy-c`.

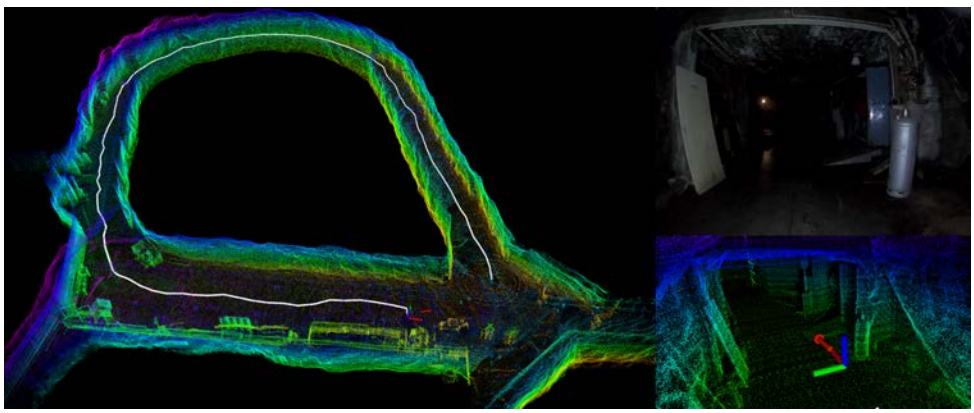


Figure 3.7: Exploration path and generated pointcloud map from APF-based exploratory navigation in a subterranean tunnel environment (left). Snapshot images (right) from the mission showing an obstacle-rich area, where the red arrow denotes the current pose reference (p_{ref}) and ψ_{ref}) sent to the controller.

We can apply similar ideas to achieve a surface following behavior, or inspection behavior. Assuming an unknown structure that we would like to inspect both through 3D LiDAR reconstruction and visual inspection with a camera, we would like to formulate an attractive force field that guides the robot around or along the structure while aligning its heading to-

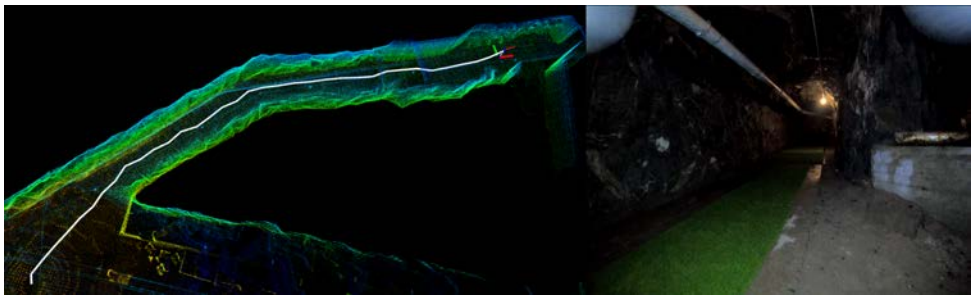


Figure 3.8: APF exploration into a more narrow and upward sloping tunnel. Exploration path (left) and a snapshot from the onboard camera showing the UAV enter into the tunnel.

wards the structure. At the same time, the previously described avoidance repulsion maintains a safe distance from the environment. Towards that, we first need to describe an attraction in both position and heading states that pulls the robot towards the structure while aligning its heading towards it as well. We apply the same conical shape for the subset of points: $\boldsymbol{\rho}_a \in \{\boldsymbol{P}\}$ where $\arccos(\frac{\rho^i_{a,x}}{||\rho^i_a||}) \leq \theta_{co}$ and $||\ \rho^i_a\ || \leq d_a$. Let N_{ρ_a} again denote the number of points inside that cone. The strategy will be to use a static attraction per point linearly towards the structure, whose magnitude does not increase or decrease in distance, to compete with the repulsion from the collision avoidance, resulting in an equilibrium between them at the desired inspection distance from the surface:

$$F^{a,lin} = \sum_{i=1}^{N_{\rho a}} L^a \frac{\rho^i_a}{||\ \rho^i_a\ ||} \tag{3.9}$$

The attractive gain-per-point L^a is selected based on L^r to achieve the desired inspection distance from the structure under the assumption that once the robot is close to the structure a majority of points in $\boldsymbol{\rho}_a$ will also be in $\boldsymbol{\rho}_r$, and $F^{r,lin}$ will increase as the robot gets closer but $F^{a,lin}$ will remain close to static. We could define an attractive heading force the same way as the repulsive one defined in (3.8) but since the attractive inspection force $F^{a,in}$ already encodes the direction towards the structure as the mean direction to all points in $\boldsymbol{\rho}_a$, a more straightforward way is that the yaw reference sent to the controller should be $\psi_{ref} = \hat{\psi} + \arctan\frac{F^{a,in}_y}{F^{a,in}_x}$. Finally, we deploy a similar "carrot chaing" approach as for exploration behavior to the rotational surface-following component as $F^{a,rot} = [p^{\mathscr{B}}_x, p^{\mathscr{B}}_y + L^{a,rot}, p^m_z - p^l_z]$, and $F^a = F^{a,lin} + F^{a,rot}$. We define again a set distance to the ground by p^m_z for the inspection to be executed at.

In total, the waypoint reference is a combination of attraction and repulsion towards the surface resulting in an equilibrium distance, while the heading is aligned also to the surface (e.g. the body X-axis is aligned to the surface). At the same time the robot is guided perpendicular to the surface by $F^{a,rot}$.

The inspection-APF was evaluated in a laboratory environment, where the UAV was tasked

to inspect a fractured structure with an uneven shape. As the APF has no concept of needing a discrete surface type to follow or a map of the structure, this is a good example of where this inspection navigation can shine as it is only using the raw LiDAR data to generate the desired behavior. As the inspection-APF is not designed to ”find” the structure first, the mission is initialized with the UAV facing the structure. For a more complex mission, obviously the APF would need to be paired with a higher level mission planner, as it is only executing reactive navigation behavior. The resulting inspection trajectory around the structure can be found in Figure 3.9, where the robot maintains a safe distance from the structure throughout the mission. The figure also shows the generated current reference pose (red arrows) at two instances during the mission. The inspection trajectory is relatively smooth with minimal unnecessary maneuvering despite the fractured shape of the structure and the constrained space the mission was executed in. The real-time flight behavior can be found at: `https://www.youtube.com/watch?v=13dW_zgrM4A`.

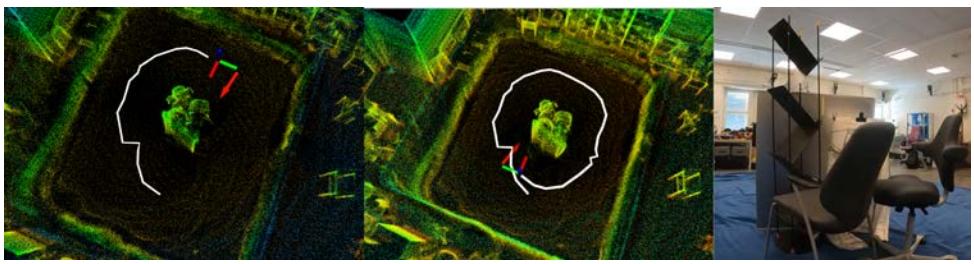


Figure 3.9: APF-based inspection trajectory in a laboratory environment, showing the current UAV pose (coordinate frame) and the current pose reference generated for inspection behavior (red arrow). The right image shows the geometrically fractured and not-smooth object to be inspected.

3.6 Concluding Remarks

The chapter has introduced and evaluated a fully reactive APF, with the motivation to generate robust navigation behavior in a series of scenarios while only relying on the direct sensor (3D LiDAR) data to navigate. As such these behaviors are executed without the classical perception or mapping layer, which are often a source of unreliability and error when it comes to autonomous navigation. Similarly, in safety-critical scenarios such as human-robot and multi-agent operational situations, any moment of poor perception or communication loss could lead to a crash or injury. As the APF is fully reactive it does not have the ability to on its own execute more complex missions, but in the authors opinion the evaluations listed in this chapter provide an argument for combining reactive local navigation with higher level directives from a mission planner that can utilize a list of reactive behaviors to execute complex tasks, as opposed to always relying on occupancy maps and/or global path planning.

The reactive APF was deployed for fail-safe collision avoidance maneuvers for both static

obstacle and mixed-traffic, and extensions were made to demonstrate the exploration of subterranean tunnels, and the inspection of structures. The APF framework was evaluated both in a laboratory setting, and in realistic field environments, and it could generate efficient and reliable navigation behaviors in all scenarios.

CHAPTER 4

Combined Exploration-Planning in 3D Environments

4.1 Overview

This chapter will discuss the development and evaluations of a tree-based combined exploration-planning algorithm: Exploration-RRT (ERRT). Exploring previously unknown areas is a critical use-case for autonomous robots, for search-and-rescue purposes or for autonomously mapping unknown unstructured areas that have not yet been mapped. In short: while navigation in known environments has its own set of challenges, exploring and navigating through unknown areas places very high demands on the onboard intelligence of the robot. The robot has to take decisions not only on how to get to the desired waypoint in a trade-off between a short and safe path, but also on the more abstract question of which area in the unknown environment it should go to in order to explore it. In this chapter we will discuss a method of combining those two questions of "where to go next" and "how to get there" through the use of Rapidly-Exploring Random Trees (RRT). The idea is to generate many goal waypoints where if the robot was to navigate there, it would increase its known space, then leveraging the RRT to find robot-safe paths to all those goals. Those paths can then be evaluated on the predicted explored volume along the path, the length of that path, and the model-based actuation ("effort/energy") required of the robot to get there. The outcome is a general algorithm that when applied to a robotic platform will guide it to explore any 3D environment autonomously, where the exploration and path planning problem are unified. This chapter will initially describe the problem formulation of the combined exploration-planning problem, then go through the algorithmic implementation, and finally offer significant simulation and in-the-field experiments to verify its efficiency. As will be demonstrated ERRT has the ability to efficiently and intuitively explore vast unstructured 3D environments in simulations, and can be run on a real robotic (UAV) platform using only onboard sensing and computation resources.

4.2 Introduction

As a general principle, robotic exploration is fundamentally based around guiding the robot towards unknown areas, while continually and procedurally building a map of explored or known territory around it through onboard sensors. The most common conceptualization for achieving such a behavior is the discretization of the space around the robot into an occupancy map [99,112] that encodes information about if a certain 2D pixel or 3D voxel is free, occupied, or unknown. The legacy works then coined the boundary between the free and the unknown space as *frontier points* [131–133]. As opposed to the tunnel following exploration APF from Chapter 3 that is simply attracted to open areas whether they have already been explored or not, the frontier provides a discrete measure for an area that is of interest for the robotic explorer. Another way to state that, is that by travelling through the free space to the frontier point, the robot will increase its *information gain*, as new unknown areas will come into sensor view and will update the state of affected voxels from unknown to free or occupied. The process of exploration then becomes straight forward: from the current state of the occupancy map and the robot estimated position - evaluate the best frontier point to go to, and once it is reached (or at a set update rate) re-evaluate the best frontier to go to next based on the now updated map of the environment. Assuming that the ”perception problem” of deciding the occupancy state of the map is working sufficiently well (which is its own separate research question), the remaining problem is then to decide which frontier is best. Commonly, this is done by a heuristic function that evaluates each frontier on some pre-defined criteria like the number of other frontier points around it and the euclidean distance from the robots current estimated position to that frontier. In this way the problem of ”where to go” and ”how to get there” are solved separately. This line of thinking continues to be expanded in the state-of-the-art in modern works: The method in [134] formulated a Neural Network that could classify the frontier point with the highest probability of information gain and use a A* planner [135] to generate the path to that frontier. The work in [113] selects frontiers not from a perspective of maximizing information gain, but from a perspective of fuel efficiency and a continued forward exploration, while a risk-aware grid-search algorithm [136] plans the path to both local and global frontiers. The work in [137] generates frontiers and plans a global route as a travelling salesman problem, while local kinodynamic paths are planned via B-Spline planning [138]. The work in [139] also include concepts related to semantic environment representations in classifying frontier points for a more ”human-like” exploration process.

A separate line of methods in the state-of-the-art does not directly try to evaluate which frontier point to go to, but instead samples many poses around the robot and based on a model of the robots onboard sensor(s), evaluates which of these randomly sampled poses it should go to in order to maximize its information gain. These methods are often denoted as *Next Best View* (NBV) methods [140] [141], and were used to both explore areas and for coverage planning e.g. generating the navigation behavior as to sense/measure/reconstruct an unknown structure of interest. One of the foundational works on NBV in modern times, found in [118], utilizes a RRT structure to build a search tree in the free space around the robot. Each node in the generated tree is evaluated for its predicted information gain and the summed path length of that particular branch, and the the search tree is expanded until a sufficiently good branch is

found (which the authors denote as a receding horizon problem). This is relatively computationally heavy as each new node added to the tree has to be evaluated but the result is a solution to the combined exploration-planning problem for exploring and mapping an unknown area. This type of method gained popularity in the latest years, and the method that will be described in this chapter follows a similar direction. The method described in [110] tackled the problem of agile navigation in constrained spaces by sampling acceleration directly, and by considering the robot as a volumetric object as to generate robot-safe paths. The works in [114, 142–144] apply a similar methodology and generate graphs of sampled positions in the free (and safe) space around the robot and the edge nodes in the graph are evaluated for information gain, where then a graph-search algorithm can be used to find the shortest paths between nodes in the graph.

The method that will the described in this chapter, the ERRT algorithm, approaches the problem slightly differently. As opposed to evaluating each branch for information gain and other criteria, which can be very computationally heavy (and other methods sometimes simplify this process through averaging), ERRT samples specific goal positions under certain criteria (e.g. information gain greater than zero or above a set value) and only evaluates RRT-branches that eventually lead to those goals to reduce the computational effort and enabling more complex modules to be added on top (as the total number of paths to be evaluated is pre-defined). In ERRT, those modules are 1) path optimization through iterative environment collision checks to shorten the sampled paths, and 2) solving the predicted robot model-based actuation along the paths as a NMPC (nonlinear model predictive control) problem, from which we both generate dynamics-based paths and have the ability to include the robot actuation along the path in the path evaluation process. The rest of this chapter will detail the general problem formulation of the ERRT algorithm, and its implementation into an executable algorithm that also utilizes real-time sensor data. A large emphasis is also placed on evaluating the algorithm. This is done in the context of exploring large subterranean environments in realistic simulations, taken from the DARPA Subterranean Challenge [122], and also through field experiments in relevant subterranean tunnel environments.

4.3 Contributions

This chapter will present contributions to the general field of robot exploration methods in the form of the ERRT algorithm:

- The development of a novel combined exploration-planning method that is fully utilizing random trees (RRT). ERRT will take into account the predicted information gain, the total path length, and the model-based robot actuation, to evaluate the ”next best trajectory” for exploration purposes. The method samples candidate goal positions and uses a 3D LiDAR model to predict which unknown areas will come within sensor view by travelling along branches to those goals, as a measure of the predicted information gain.
- The algorithm is designed to on its own execute complete exploration of a completely unknown area, while also planning safe paths throughout the mission by considering the robot as a volumetric object.

- ERRT is coupled with the state-of-the-art occupancy mapper UFOMap [27] that maps voxel states not only to free or occupied but also explicitly models the unknown space as well. Through the UFOMap, ERRT can use real sensor data and a continuously updated 3D occupancy map of the environment to generate exploration trajectories.
- ERRT is evaluated extensively first in small-scale simulations that check for the cosistency of perforance, and then in large-scale simulation environments some of which were real scanned environments from the DARPA SubT Challenge Final Stage [122].
- Finally, ERRT is evaluated in the field using a real UAV platform equipped with a 3D LiDAR, using only onboard computation power. The field experiments are performed in narrow subterranean tunnel areas that are very challenging, especially for robot safety. As will be demonstrated ERRT can generate safe and dynamical paths that provide consistent exploratory behavior with minimal backtracking.

4.4 Exploration-RRT

4.4.1 Problem Formulation

The momentary combined exploration-planning problem can be seen as a maximization of information gain, and a minimization of total path length, the actuation required to follow the path, and the traversability or risk cost associated with following that trajectory. We can formulate this problem in the following way:

$$\underset{\text{trajectory } \boldsymbol{\chi}}{\text{Minimize}} \; C_d(\boldsymbol{\chi}) + C_r(\boldsymbol{\chi}) + C_u(\boldsymbol{u}(\boldsymbol{\chi})) - C_i(v(\boldsymbol{\chi})) \tag{4.1}$$

$$\text{subj. to: } \boldsymbol{\chi} \in V_{\text{free}}$$

$$v(\boldsymbol{\chi}) > 0$$

Here, $\boldsymbol{\chi}$ represents the state-trajectory to be solved for (the solution to the problem), while $C_d(\boldsymbol{\chi})$ is a cost associated with the path length, $C_r(\boldsymbol{\chi})$ a cost associated with the risk or *traversability* of $\boldsymbol{\chi}$, $C_u(\boldsymbol{\chi})$ is the cost related to the model-based actuation along $\boldsymbol{\chi}$, and $-C_i(v(\boldsymbol{\chi}))$ is the revenue or negative costs from the sensor-based information gain v along $\boldsymbol{\chi}$. $v(\boldsymbol{\chi}) > 0$ is the constraint that forces solutions to the minimization to generate exploration behavior. Here, V_{free} denotes the free space as a subset of V_{map} that describes an occupancy grid of the space around the robot divided into free,occupied, and unknown space. We can state this to be the ”real” exploration-planning problem whose optimal solution would be the best trajectory (based on the tuning) to follow to optimize the momentary robot exploration problem. Note: the over-time exploration process is often described as a reduction of map entropy, but we are more interested in describing the momentary exploration-planning minimization of ”what is the next best trajectory?”. Equation (4.1) is an extremely hard problem to formulate for a piece of optimization software, and as such we are forced to look for sampling-based solutions instead. In the ERRT algorithm, the ”actual” problem being solved is:

$$\boldsymbol{\chi}^* = \underset{(\boldsymbol{\chi}_j)_j}{\arg\min} \left(C_d(\boldsymbol{\chi}_j) + C_u(\boldsymbol{u}(\boldsymbol{\chi}_j) - C_i(v(\boldsymbol{\chi}_j))_j, \right. \tag{4.2}$$

$$j = 1, \ldots, n_{\text{traj}}$$

$$\text{where } \boldsymbol{\chi}_j \in V_{\text{safe}}$$

$$v(\boldsymbol{\chi}_j) > 0$$

where n_{traj} denotes the number of sampled candidate trajectories that satisfy $v(\boldsymbol{\chi}) > 0$. Based on the formulation of the problem in (4.1) and a high enough number of sampled trajectories n_{traj}, we denote the output as the approximate optimal next-best-trajectory $\boldsymbol{\chi}^*$. For an UAV that can move freely in 3D, the risk-cost $C_r(\boldsymbol{\chi})$ can be simplified into the statement that $\boldsymbol{\chi}$ should be in V_{safe} where $V_{\text{safe}} \subset V_{\text{free}}$ such that for all safe and free voxels $\{F_s\}$ in V_{safe}:

$$|| \{F_s\}_i - \{O\}_{\text{closest}} || > r_{\text{robot}} \tag{4.3a}$$

$$|| \{F_s\}_i - \{U\}_{\text{closest}} || > r_{\text{robot}} \tag{4.3b}$$

with $\{O\}_{\text{closest}}$ and $\{U\}_{\text{closest}}$ denoting the position of the closest occupied or unknown voxel to $\{F_s\}_i$ and r_{robot} denotes the size-radius of the robot (e.g. for all trajectories in V_{free} any non-free part of the environment should be a distance r_{robot} away from the robot).

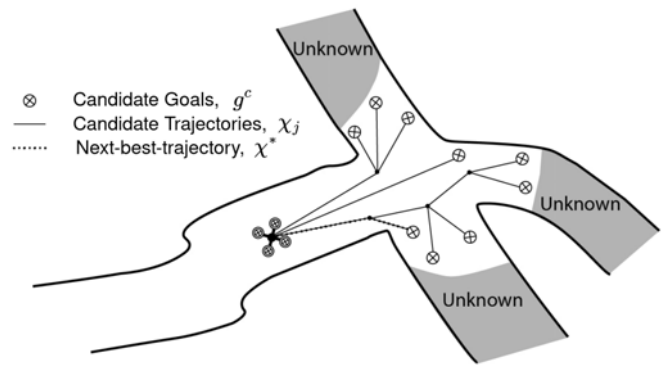


Figure 4.1: The ERRT concept - candidate goals are generated under certain conditions, candidate trajectories are generated to them, and the approximate next-best-trajectory is selected.

Equation (4.2) defines the desired outcome of the ERRT algorithm, as a simplified sampling solution to the real problem (4.1), and a concept figure of the ERRT process can be seen in Figure 4.1. The following section 4.4.2 will overview the code implementation and describe how ERRT solves that problem.

4.4.2 Implementation

Algorithm Overview

This section will serve as a high level description of ERRT. The implementation of ERRT follows a multi-stage approach where the various components in (4.2) are calculated in sequence and will be explained in more detail in the following Sections. The stages can be summarized in order as: (1) Candidate Goal Sampling, (2) Robot-safe RRT* tree expansion and candidate branch generation, (3) Computation of model-based actuation trajectories, and 4) Computation of information gain and trajectory evaluation. The process is visualized in the Figure 4.2.

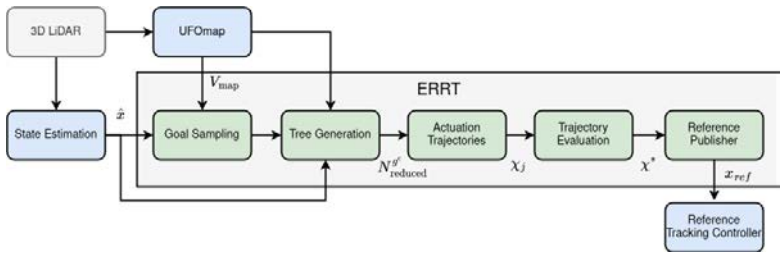


Figure 4.2: The ERRT architecture: candidate goal sampling, tree generation, calculation of actuation trajectories, and trajectory evaluation. ERRT is coupled with a 3D LiDAR as well as the UFOmap occupancy mapper, and generates state references to a controller.

We can leverage the ERRT concept of generating a set n_{traj} number of candidate branches to deploy more computationally heavy processes, in this case the iterative path improvement by volumetric collision checks, and also computing actuation trajectories as a NMPC problem both to "smooth out" the trajectories as to be described by how the robot it actually actuated and to be able to evaluate the actuation cost $C_u(\boldsymbol{u}(\boldsymbol{\chi}_j)$ along the trajectory. The tree-based exploration process is applied only on a local subset of the whole map space as RRT-based solutions lose their efficiency relatively quickly for larger areas. As such V_{map} is replaced with local map V^l_{map} in (4.1), which in the implementation implies extracting a subset of the occupancy map confined by bounding box of the desired sampling volume centered on the estimated robot position $\hat{p}$. As such, in this implementation, ERRT on its own should be considered a local exploration-planning framework, but as will be seen in Section 4.6 ERRT still enables the exploration of large-scale environments.

Candidate Goal Sampling

The ERRT algorithm starts by the generation of candidate goals. This process starts by generating candidate goal poses g^{c}, which based on (4.2) are n_{traj} in number, all have $v(g^{\text{c}}) > 0$, and all lie in the local safe space $g^{\text{c}} \in V^l_{\text{safe}}$. Sampled points P are randomly generated in V^l_{map} and checked for these conditions to see if it is a valid candidate goal g^{c}. To ensure an efficient and more even distribution of candidate goals we apply *Poisson disk sampling* such that all

candidate goals must be a set distance d_{g^c} from each other, and a set distance from the robot position $\hat{p}$. To compute $v(g^c) > 0$ a volumetric check is made on a sampled point P based on a 3D LiDAR field-of-view with sensor range S_r and vertical field-of-view S_θ such that $v(P)$ returns the number of unknown voxels $\{U\}$ that are in field-of-view and not blocked by any occupied voxels $\{O\}$. If all such conditions are met, the sampled point is added as a candidate goal g^c. Additionally, although $v(g^c) > 0$ is the condition for continued exploration and the code can exit the check when any unknowns are found, in the implementation we allow the user to select also a $v(g^c) > v_{\text{min}}$ but this comes at a cost of computational efficiency as total information gain must be calculated for each sampled goal that goes through the initial check $g^c \in V^l_{\text{safe}}$ (not just that it is non-zero).

Tree Expansion and Candidate Branch Generation

Rapidly-Exploring Random Trees have been around for a while now [23] as one of the many methods for robot occupancy-based path planning, and still sees research today towards developing the RRT tree-expansion itself [31, 145, 146]. In ERRT, we will be applying a relatively standard RRT* algorithm. As the tree-building closely follows the legacy method it will not be described in great detail here. In short (and generalizing), RRT algorithms are based on sampling points in the occupancy-space around the robot, checking if the sampled point can be attached to the current configuration of the tree without the straight line between them entering into any occupied cells, and then discarding the point or attaching it to the tree. The process then rapidly repeats, expanding the tree to cover the free space in the occupancy map. We can then ask if a certain "goal" in the map can be attached to the tree, or if it is already part of it. The branch that the goal is attached to can then be traced back to the root of the tree (at the robot position), and a path between the robot position and the goal has been found.

First, initialize the tree $\boldsymbol{N}$ with a root node at the robot current position $\hat{p} = [p_x, p_y, p_z]$. In our implementation, random points $P = [P_x, P_y, P_z]$ are generated in the local sampling space V^l_{map}. P is checked against its occupancy state in V^l_{map} to ensure that it is in V^l_{safe} by performing a volumetric spherical collision check with robot radius r_{robot} centered on P. Then, we search through $\boldsymbol{N}$ to find the closest node $N_{\text{closest}} \in \boldsymbol{N}$ in the tree, and check if the straight line $P \rightarrow N_{\text{closest}} \in V^l_{\text{safe}}$ by collision-checking the cylindrical volume between N_{closest} and P with radius r_{robot}. If all checks pass, P is added as a child node to N_{closest} in the complete tree $\boldsymbol{N}$, and the process is repeated. We found that adding P to $\boldsymbol{N}$ directly as opposed to the more usual "step" from N_{closest} towards P was more efficient for searching a smaller local space with fewer iterations. After a set number nodes have been added to $\boldsymbol{N}$ we compute which branches in $\boldsymbol{N}$ can be extended by the candidate goals g^c by checking all nodes and goals for the conditions $N \rightarrow g^c \in V^l_{\text{safe}}$, and $\mid N - g^c \mid < d_{\text{extend}}$ (as to limit the number of nodes to test for). We then extract the shortest such branches to each goal node and we can denote them as $\boldsymbol{N}^{g^c}_j$ where $j = 1, 2....n_{\text{traj}}$. As such the total search tree $\boldsymbol{N}$ has been reduced to a set of n_{traj} candidate branches that have guaranteed information gain in them since $v(g^c_j) > 0$ (and each branch leads to a separate goal).

The second step is attempting to optimize and shorten all $\boldsymbol{N}^{g^c}_j$ through iterative collision checks. This is classically not used in a RRT* set-up but in ERRT the idea is to make a more

sparse tree that might not have optimally short branches, and instead try to optimize those $\boldsymbol{N}_j^{g^c}$. The program starts by checking if $N_{j,1}^{g^c} \rightarrow N_{j,n_j}^{g^c} \in V_{\text{safe}}$ e.g. if the straight line from the robot position $\hat{p}$ to the candidate goal g_j^c is collision-free and safe. If it is, as the straight line is the shortest possible path, all other nodes are redundant and are removed. If not, the process continues with checking if $N_{j,2}^{g^c} \rightarrow N_{j,n_j}^{g^c} \in V_{\text{safe}}^l$ etc. If a path improvement is found between index i and the end node, the process is restarted with checking $N_{j,1}^{g^c} \rightarrow N_{j,i}^{g^c} \in V_{\text{safe}}^l$ and so on. We can denote the resulting tree as $\boldsymbol{N}_{\text{reduced}}^{g^c}$. The path shortening step is then repeated but with added nodes at a set distance from each other in the resulting $\boldsymbol{N}_{\text{reduced}}^{g^c}$, as an additional possibility at connecting the nodes and making branches with a smaller total path length. This last step is to make smoother paths, and to shorten paths around corners or obstacles. The end-result are the n_{traj} optimized (shortened) candidate graphs/branches, with a specified distance between nodes, where each branch has its end-node at the candidate goal position. For the sake of notation in the following Section 4.4.2 let us denote them as the reference position trajectories $\boldsymbol{X}_j^{\text{ref}} = [X_{j,1}^{\text{ref}}, X_{j,2}^{\text{ref}} ... X_{j,n_j}^{\text{ref}}]$.

Actuation Trajectories

We are interested in computing the model-based actuation of the robot required to follow a specific candidate trajectory. There are three main reasons how doing so can assist in generating the desired exploration behavior. First, as posed by (4.1), the actuation cost as a gauge for the energy/effort required to follow a trajectory is of great interest. A trajectory with a higher actuation requirement will require more effort from the robot to execute, and, in the context of an UAV, rapid direction changes or aggressive maneuvering makes the trajectories harder to follow and can result in unwanted flight behavior. Second, the resulting actuation-trajectory will be within the dynamic constraints of the platform and reflect how the robot can move based on its available mode of actuation. Third, penalizing high-effort direction changes also penalizes backtracking and changing the direction of exploration, which will lead to favoring trajectories that not only minimize path length but also minimize unwanted maneuvering for a continued forward exploration. In other works on robot exploration this behavior is hard-coded as part of frontier selection [113], is part of the acceleration sampling [110], or due to "carrot-chasing" approaches to robot exploration [128] (also the case in the exploration in Chapter 3). In ERRT, the actuation trajectories are solved for as a NMPC problem (following the approach in previous works [47, 69, 80]), that fully considers the nonlinear model of the UAV and its dynamical constraints. This almost exactly follows the approach described in Chapter 2, and as such we will not describe it in detail here as well. The only difference in the following formulation is that we replace the state reference x_{ref} with our state reference trajectory $\boldsymbol{X}_j^{\text{ref}}$. For the sake of notation, let us define that $k+l \mid k$ implies a prediction l time steps forward produced at time step k, where the rest of the notation follows the previous Chapter 2. We now formulate the objective function $J(\boldsymbol{\chi}_k, \boldsymbol{u}_k)$ as:

$$J(\boldsymbol{\chi}_k, \boldsymbol{u}_k) \;=\; \sum_{l=0}^{N} \|X_l^{\text{ref}} \;-\; \chi_{k+l|k}\|_{Q_X}^2 \;+\; \|u_{\text{ref}} \;-\; u_{k+l|k}\|_{Q_u}^2 \;+\; \|u_{k+l|k} \;-\; u_{k+l-1|k}\|_{Q_{\Delta u}}^2. \quad (4.4)$$

Let us also right away define constraints on the control inputs $u_{\min} \leq u_{k+l|k} \leq u_{\max}$ to realistically restrict the thrust and attitude commands based on the utilized UAV platform's dynamical constraints.

We feed in the optimized RRT-branches $\boldsymbol{X}^{\text{ref}}$ as state references along the horizon (with velocity and angle references set to zero) and output the predicted and dynamically constrained state trajectory $\boldsymbol{\chi}$ and the actuation vector $\boldsymbol{u}$. The resulting optimization problem can be written as:

$$\underset{\boldsymbol{u}_k, \boldsymbol{\chi}_k}{\text{Minimize}} J(\boldsymbol{\chi}_k, \boldsymbol{u}_k) \quad (4.5a)$$

$$\text{s. t.: } \chi_{k+l+1|k} = \zeta(\chi_{k+l|k}, u_{k+l|k}),$$

$$l = 0, \ldots, N, \quad (4.5b)$$

$$u_{\min} \leq u_{k+l|k} \leq u_{\max}, l = 0, \ldots, N, \quad (4.5c)$$

$$\chi_{k|k} = \hat{x}_k. \quad (4.5d)$$

where the initial state $\chi_{k|k}$ is the full UAV model state $\hat{x} = [\hat{p}, \hat{v}, \hat{\theta}, \hat{\phi}]$ estimated by onboard state-estimation. The problem is solved by the nonlinear nonconvex parametric optimization software Optimization Engine [25] and the PANOC [68] algorithm. The NMPC problem is solved for all reference trajectories $\boldsymbol{X}_j^{\text{ref}}$ to generate the optimally actuated (based on objective function (4.4)) trajectories $\boldsymbol{\chi}_j$ and actuation vectors $\boldsymbol{u}_j$ that track $\boldsymbol{\chi}_j$. The result is the full state trajectory $\boldsymbol{\chi}_j$. It should be noted that a clear limitation is that maximally N steps in the original trajectory $\boldsymbol{X}_j^{\text{ref}}$ can be computed in this way, but for a reasonable local sampling space and desired velocity (e.g. based on sampling time in the NMPC problem and distance between nodes in $\boldsymbol{X}_j^{\text{ref}}$) it is rarely a limitation. For example, using $\delta_t = 0.4\,\text{s}$, and a branch step size of $0.4\,\text{m}$ (e.g. we generate trajectories with an average exploration speed of $1\,\text{m/s}$), and a horizon $N = 50$, the trajectories can cover $20\,\text{m}$. The added computation time from the NMPC optimization is around $3\,\text{ms}$ per trajectory.

Trajectory Evaluation

The final step of the ERRT algorithm is evaluating the generated candidate trajectories/branches according to the minimization in (4.2) in order to select the next-best-trajectory $\boldsymbol{\chi}^*$. To calculate the information gain along $\boldsymbol{\chi}_j$ we require a sensor model. In this work, a simple 3D LiDAR model is used with only two parameters: the sensor range S_{r} and the vertical field-of-view angle S_θ. Through the UFOmap code library one can extract all unknown voxels $\{U\}$ in a "frustrum" (camera-like field-of-view) from the robot position or from any point in the

trajectory. That subset of unknown voxels $\{U\}_{\text{fov}}$ in V^l_{map} can then be analyzed if they are in line-of-sight of the sensor e.g. that $\boldsymbol{\chi}_{j,i} \rightarrow \{U\}_{\text{fov}} \in V^l_{\text{free}}$ for each $\{U\}_{\text{fov}}$. We can perform four such checks in each cardinal direction (*x*,-*x*,*y*,-*y*), with $\pi/2$rad horizontal field-of-view, S_θ vertical field-of-view, and range S_{r}, with the center point at a certain position in the trajectory, to mimic the field-of-view of a 3D LiDAR and to extract all unknown voxels within that field-of-view. Here, ERRT supports two modes: 1) calculating the information gain at the candidate goal g^c_j which is the most common approach in the literature, or 2) leveraging the fact that we are only working with a set relatively low number of candidate trajectories (not the whole graph/tree), and evaluating information gain along the trajectories at a higher computational cost. Computing information gain at each equidistant point in $\boldsymbol{\chi}_j$ is too difficult and has significant overlap. Instead the information gain can be calculated at each node in $\boldsymbol{N}^{g^c}_{\text{reduced}}$ that exceed a specified distance apart, and any duplicate/overlapping unknown (seen from multiple nodes) is removed. This generates a better approximation of the total information gain along the trajectory as opposed to only evaluating at the end-point. We then denote the resulting total non-duplicate number of $\{U\}$ along $\boldsymbol{N}^{g^c}_{\text{reduced,j}}$ as the information gain for that trajectory assuming that $v(\boldsymbol{N}^{g^c}_{\text{reduced,j}}) \sim v(\boldsymbol{\chi}_j)$. The information gain revenue is calculated as:

$$C_{\text{i}}(v(\boldsymbol{\chi}_j)) = K_{\text{i}} v(\boldsymbol{\chi}_j), \tag{4.6}$$

where K_{i} denotes a gain related to the relative emphasis on the information revenue parameter. As such $C_{\text{i}}(v(\boldsymbol{\chi}_j))$ represents the predicted sensor-based information gain along the trajectory.

Evaluating the distance cost $C_{\text{d}}(\boldsymbol{\chi}_j)$ is done by simply summing the distance between consecutive positions in the actuation-based trajectories $\boldsymbol{\chi}_j$ as:

$$C_{\text{d}}(\boldsymbol{\chi}_j) = K_{\text{d}} \sum_{i=2}^{n_j} || \, \chi_{j,i} - \chi_{j,i-1} \, || \tag{4.7}$$

with K_{d} denoting the relative emphasis on the length of the trajectory. Finally, computing the actuation cost is done by feeding the actuation vector $\boldsymbol{u}_j$ back into (4.4) as:

$$C_{\text{u}}(\boldsymbol{u}(\boldsymbol{\chi}_j)) = K_{\text{u}} \sum_{i=0}^{N} \|u_{\text{ref}} - u_{j,i}\|^2_{Q_u} + \|u_{j,i} - u_{j,i-1}\|^2_{Q_{\Delta u}}. \tag{4.8}$$

The resulting $C_{\text{u}}(\boldsymbol{u}(\boldsymbol{\chi}_j))$ represents the (nonlinear) model-based actuation cost along each full trajectory $\boldsymbol{\chi}_j$. With all trajectories generated in V^l_{safe} and all costs computed, the ERRT program can now evaluate the expression in (4.2) to find the approximate next-best-trajectory $\boldsymbol{\chi}^*$.

4.5 Initial small-scale simulation

The first stage of evaluations are performed in a simple simulation environment where ERRT is evaluated separately from the supportive modules of the UFOmap and onboard sensors. Due to the simplicity of the simulation, we are also not considering the robot size in the problem and as such $V_{\text{safe}} = V_{\text{free}}$. The goal is to assess if ERRT can consistently and efficiently fully explore

a small fully unknown 3D environment, without getting stuck or generating undesirable exploration behavior. The environment as well as snapshots of candidate branch generation from a simulation run can be seen in Figure 4.3, but the best visualisation of the exploration behavior can be found in the following video link: `https://drive.google.com/file/d/1v3vg3Z9iB2DR-Oec3MxWUg_39d3lIj1F/view`, where the viewer can see the real-time exploration of the environment. ERRT efficiently explores the 3D unstructured environment by generating trajectories that have unknown (green) voxels in sensor view. An important criteria for a purely sampling based algorithm is the ability to consistently generate good performance without significant outliers. The ERRT algorithm was run in the same environment from the same starting position for ten runs with two different tunings (one more greedy prioritising information gain, and one conservative prioritising low actuation paths). Figures 4.4 and 4.5 show the time and total exploration path length for 90% coverage and full exploration of the environment, showing that ERRT was consistent over many runs when it comes to exploration efficiency in this limited simulation study.

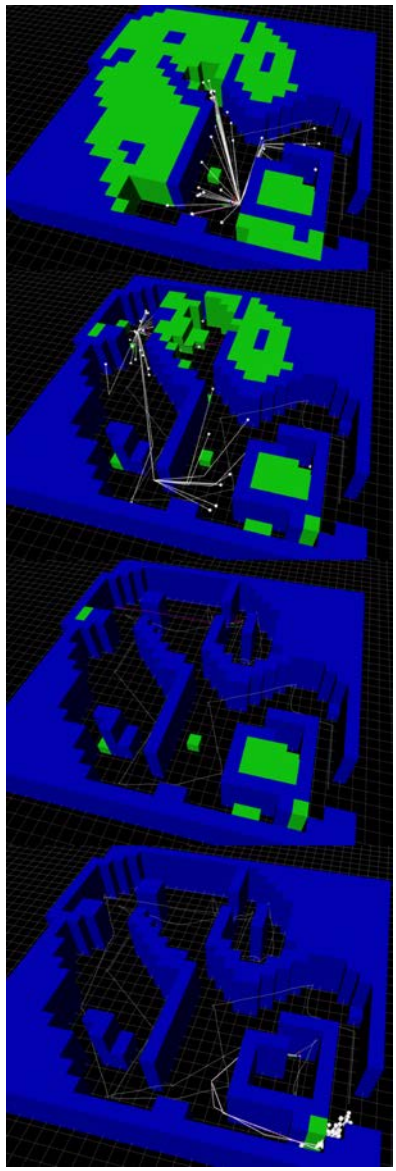


Figure 4.3: Environment for initial small-scale simulations, and an example of an exploration progression in that environment. Green are unknown but free voxels while the blue represents the occupied areas.

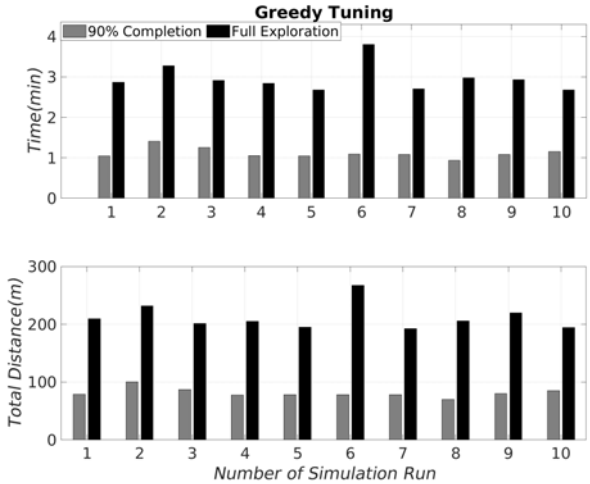


Figure 4.4: Testing the consistency of the algorithm. Showing the total time and distance travelled to explore 90% and fully exploring the small-scale environment over ten runs with a greedy tuning.

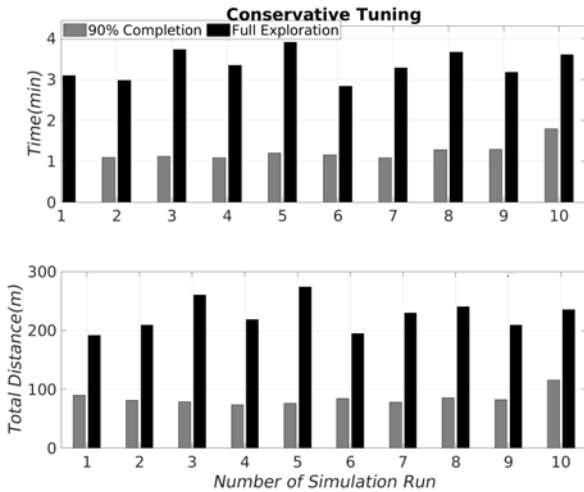


Figure 4.5: Testing the consistency of the algorithm. Showing the total time and distance travelled to explore 90% and fully exploring the small-scale environment over ten runs with a more ”conservative” tuning.

4.6 Realistic Simulations in large-scale Subterranean Environments

The next step was deploying ERRT in a more realistic and large-scale simulation. Here we are using the Gazebo simulator, where both the UAV size (and environment interactions), dynamics, and the onboard sensors are included in the simulation. The UAV platform as well as the dynamics and attitude controller are from the RotorS package [147]. The selected simulations worlds are from the DARPA Subterranean Challenge [122], where we are using both worlds from the simulation challenge, as well as a scanned map from the final stage competition. These environments include complex tunnels and junctions, open caves, narrow entrances, as well as tight and obstacle-rich areas, presenting a significant challenges to onboard navigation and exploration behaviors. In the simulations, ERRT is coupled with a full state reference tracking controller, namely the NMPC described in Chapter 2. The simulated UAV is equipped with a 32° vertical field-of-view velodyne VLP16 3D LiDAR.

As ERRT is focusing on efficient information-gain maximizing behavior, is designed to be a local exploration module, and since real UAVs have limited flight time, we set up simulation runs in each world where we let the UAV explore for a number of minutes at realistic navigation speeds of $0.5-1\,\mathrm{m/s}$. As such full coverage of the area will never be achieved as the worlds are huge, but instead the main goal should be efficient local exploration behavior that continually guides the UAV into information-rich areas.

In Figure 4.6 the over-time exploration progress is visualized for a run in the DARPA final stage world, and a video visualisation of that simulation run can be found at `https://www.youtube.com/watch?v=EAXsn-KjW-k`.

Figure 4.7 shows the general ERRT concept and program; 3D robot-safe tree expansion in the local safe space V^{l}_{safe}, the computation of improved (shortened) actuation trajectories $\boldsymbol{\chi}_j$ extended to the sampled g^{c} with $v(g^{\mathrm{c}}) > 0$, and finally the selected $\boldsymbol{\chi}^*$ with a visualisation of the predicted explored volume (red markers highlighting which unknown voxels will be in sensor view) that will be achieved by following that trajectory.

Figure 4.8 shows two more exploration runs in the final stage simulator, going into other areas of the map.

These simulations highlight ERRTs capabilities in very narrow and constrained areas, where robot safe navigation is of high importance, and generating continued forward exploration through narrow areas that have limited sensor visibility (for information gain calculations). Figure 4.9 also shows a visualisation of the tree expansion in a warehouse-like area of the final stage map, with significant cluttering and obstacles. We can see the robot-safe tree completely cover the volume and find paths to all of the local g^{c}.

As opposed to the very narrow obstacle filled tunnels and junctions of Figure 4.6 the following simulation runs focus more on open voids and caves as well as large-scale tunnels, from the DARPA Virtual Competition and from the Cave Circuit. The ERRT exploration missions can be seen in Figure 4.10 and Figure 4.11.

These are massive simulation worlds, with many junctions and branching areas (not very visible in the maps). The goal is a continued greedy forward exploration without getting stuck fully exploring the massive voids and caves which can be a waste of time in a time-limited

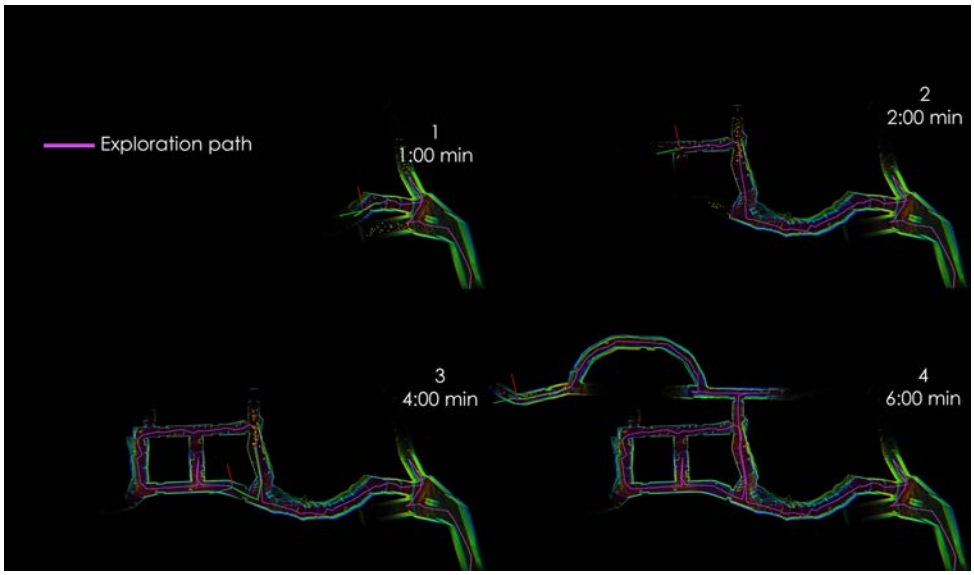


Figure 4.6: Local Exploration over 6 minutes in the DARPA Final Stage gazebo world - mimicking the conditions of the real competition. Figures highlighting the progress of exploration at set times. Total exploration path length was around 250 m.

mission, and is a very challenging problem for many similar frameworks as ERRT. A video of those two simulation runs can be found at: `https://www.youtube.com/watch?v=RKV-sbq79OU&`. ERRT selects paths with efficient information-gain maximizing behavior and 3D exploration in the wide interconnected cave areas, while maintaining safe navigation, efficiently moving from cave to cave without significant unnecessary maneuvering or backtracking.

While the ERRT framework can be tuned to fit a desired computational effort; number of nodes in the tree, number of sampled goals, sensor range (more voxels to iterate through), voxel size, using $v(g^c) > v_{\min}$ etc. all of these experiments were run with configurations resulting in a computation time of 0.8 s up to peaks of 2 s. While 2 s might sound significant, the vast majority of time is spent following trajectories, not calculating, so the average exploration speed is not significantly affected.

4.7 Field Trials

To demonstrate ERRTs ability to run on onboard hardware for computation and sensing, field experiments were performed with a custom built UAV platform in a subterranean environment, namely the same platform (and the same subterranean environment) as in the previous Chapter 3. In fact, we are completely using the local autonomy kit developed and described in Chapter 3, where ERRT simply sends state references (position, velocity) to the APF-NMPC pairing,

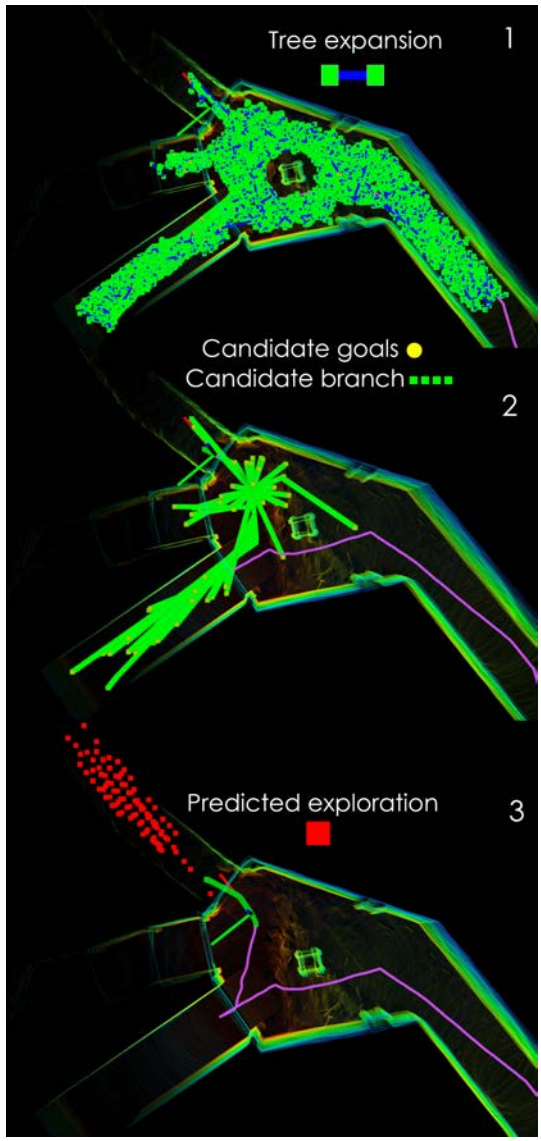


Figure 4.7: The ERRT process - 1) local robot-safe Tree Expansion filling V^{l}_{safe}, 2) pseudo-random goal sampling g^{c} (yellow dots) and improved actuation-paths $\boldsymbol{\chi}_j$ (green), and 3) selected path $\boldsymbol{\chi}^{*}$(green) with marked unknown voxels that will be discovered along the "next-best-trajectory" (red) (bottom).

and where state estimation is enabled through the LiDAR-Inertial Odometry (LIO) package LIO-SAM [26]. Two experiments are performed, where samples of the environments can be

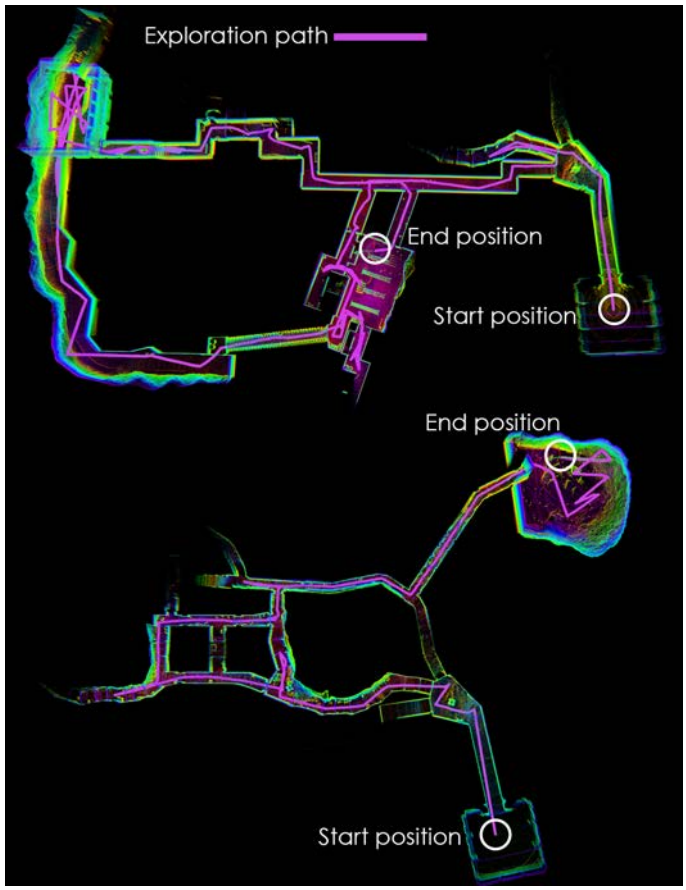


Figure 4.8: Two more ERRT exploration runs into different parts of the area, with varying kinds of environments from urban warehouse like areas to tunnels and caves.

found in Figure 4.12; one in a curving tunnel that also has a cluttered void, and one in a very narrow and constrained tunnel.

Figure 4.13 shows the exploration run in the curving tunnel, where ERRT consistently provides information-generating behavior without any backtracking or poor side-to-side maneuvering.

Another example of ERRT trajectory generation and selection can be found in Figure 4.14 as the UAV is entering the junction area from a narrower tunnel.

The next trajectory is selected deep into the middle of the junction to maximize information gain per distance travelled. Figure 4.15 shows two momentary trajectory selection in the cluttered void area where robot-safe, efficient, and dynamical paths are generated around obstacles in the environment.

Figure 4.9: Robot-safe 3D Tree expansion in an obstacle-rich and complex warehouse-like area (top), and selected exploration ”next-best-trajectory” (green line) (bottom)

The second experiment is visualized in Figure 4.16, where ERRT guides the UAV in a very narrow and constrained tunnel. Here the most challenging aspect is the limited visibility due to the narrow tunnel and the fact that only a very limited part of the tunnel could be considered robot-safe. ERRT could still manage to explore the tunnel, keeping the UAV in the middle of the tunnel and exploring forward until the junction. At the junction the UAV first went into one tunnel but then found a more information-rich path. The mission had to be terminated at that point due to the limited testing flight area.

4.8 Concluding Remarks

This chapter has introduced and described the ERRT algorithm for the combined exploration-planning problem. The goal of the algorithm is to find the ”next best trajectory” for exploring a fully unknown and unstructured 3D environment in a trade-off between maximizing infor-

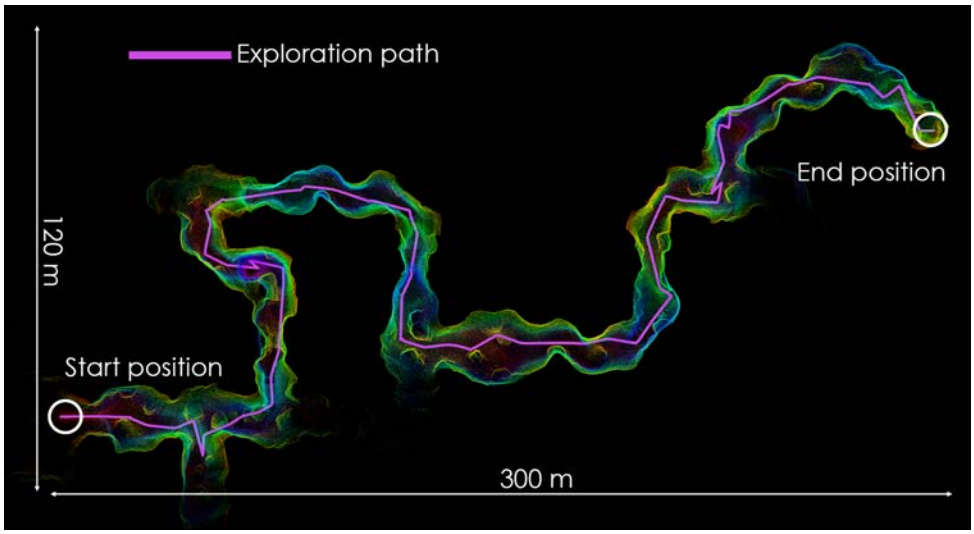


Figure 4.10: Explored area after 16 minutes in the DARPA Cave World - approx. 10-15m wide tunnels.

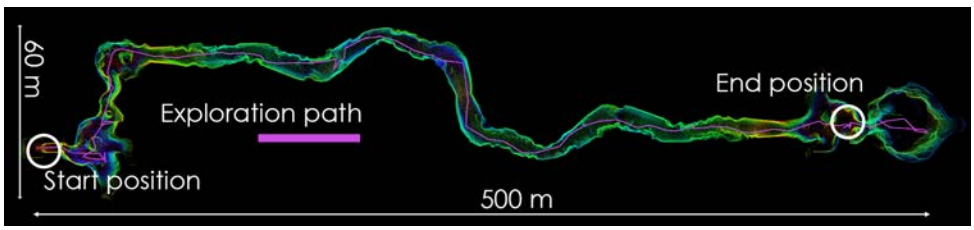


Figure 4.11: Explored area after 16 minutes in the DARPA Virtual Competition World - long cave-like tunnels.

mation gain, while minimizing the distance travelled and the robot actuation required to follow the trajectory. ERRT follows a sampling and tree-based approach where a RRT is used to find branches to candidate goals for exploration. A trajectory shortening method is applied, and the shortest branch to each candidate goal is extracted and evaluated on the aforementioned criteria. The ERRT algorithm was designed and evaluated mainly for subterranean environments where robot-safe paths are paramount as well as maintaining efficient continuous exploration behavior without backtracking. This is extensively demonstrated first in a simple simulator, then in realistic large-scale simulations with integrated sensors, and finally in real field environment on a custom built UAV platform. ERRT demonstrated efficient and safe navigation in all evaluation scenarios, from narrow and cluttered tunnels to open cave-like voids. The presented results show a promising direction for combining the exploration-planning problem but there are both limitations and future works. ERRT should be extended with a globalisation strategy that could use the already constructed tree of accessible areas and information-rich

Figure 4.12: Field Evaluation Environments for the two performed experiments. Narrow constrained tunnel (left), open but cluttered area with connected tunnels (right).

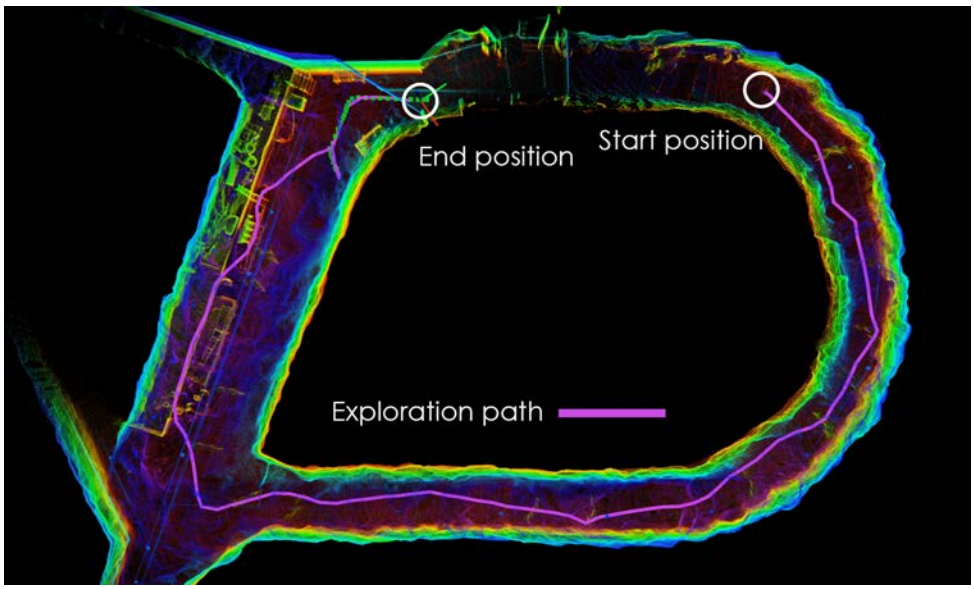


Figure 4.13: Exploration path from local exploration in the field, in curving tunnel area with a larger void. Exploration path length is around 160 m.

candidate goals. The fact that this stage was not reached limits the possible evaluation scenarios as ERRT can not handle a dead-end that is far enough from an unexplored area such that it is not included in the local V^l_{map}. But it should be noted that the local sampling space can, and was, selected relatively large as compared to other local planners, and as such the ERRT program rests somewhere in between a purely local and a fully globalised program. Also, following the approaches in Chapter 2 the NMPC module could be used to solve for

robot-safe trajectories (including obstacle avoidance in the NMPC problem) although it is not straight-forward to go from an occupancy map representation to something that the NMPC can utilize for obstacle constraints. Another future work is implementing a more advanced sampling algorithm both for the goals and for generating the tree. With a uniform sampling in a simple bounding box, many many random points are discarded or of poor quality, which wasted computational effort. Something like the work in [145] where the authors use a neural network to perform non-uniform sampling for more efficient tree expansion which could see a great application in an ERRT-like program.

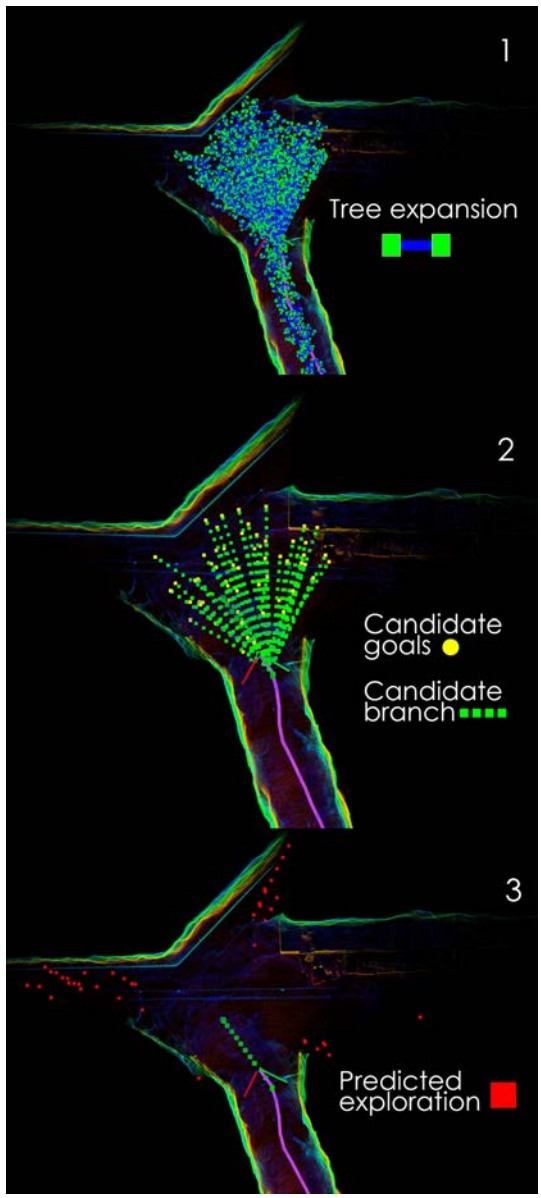


Figure 4.14: The ERRT stages in a field environment with a junction. 1) Tree expansion, 2) candidate goals and candidate trajectories, and 3) selected exploration trajectory with expected explored volume (red).

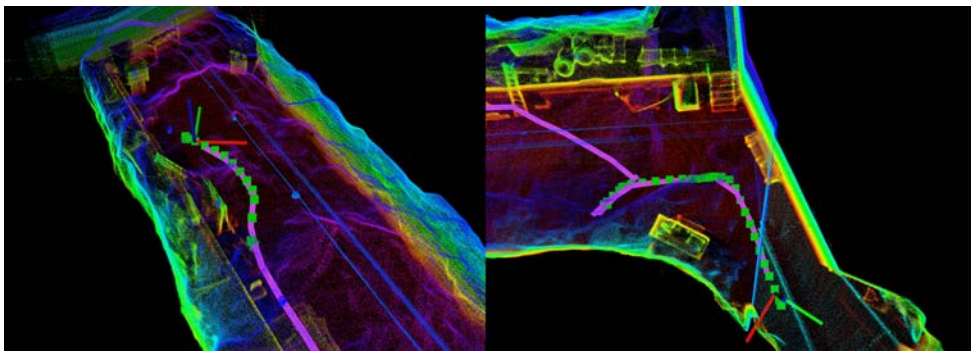


Figure 4.15: ERRT dynamic and robot-safe trajectory generation around small/complex obstacles in obstacle-rich and narrow environments in the field.

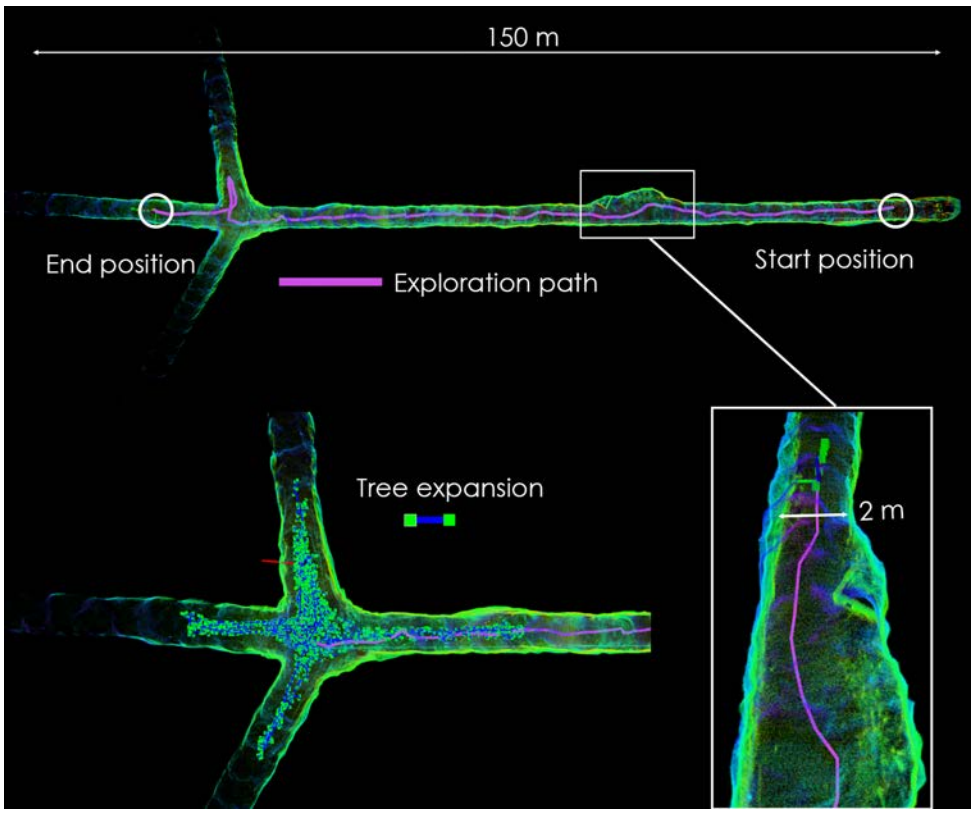


Figure 4.16: ERRT exploration path in a very constrained and narrow field environment with a total exploration path of 170 m (top), and samples of tree expansion in the juction and path selection in an extra narrow section during the exploration run (bottom)

CHAPTER 5

Field Deployment in Subterranean Environments

5.1 Overview

This Chapter will detail the developments and efforts towards the use of autonomous robots in real applications in subterranean environments. For the reader this should not come as a surprise, as underground applications are a mainstay throughout this thesis, where the works both in Chaper 3 and Chapter 4 are evaluated in subterranean environments. The significant difference is that this chapter will not focus on one singular developed component to be tested, but the goal of the following presented works is instead the application outcome and the research into advanced field capabilities of autonomous robots as a whole. The chapter will delve into two use-cases, namely autonomous search-and-rescue missions inspired by the DARPA (SubT) Robotics Challenge, as well as inspection and surveying missions in underground mines. The work on underground mine inspection is the outcome of multiple national and European projects together with significant partners in the mining industry, which enabled evaluation not only in realistic environments, but in actual underground mines where some tests were executed at over 1200 m under ground. In general, subterranean caves, mining areas, and other tunneling infrastructure (trains, construction etc.) present massive use-case opportunities for autonomous robots when it comes to safety for workers performing dangerous tasks, and for efficiently performing routine inspection and maintenance. This chapter will also describe the deployment frameworks, hardware and sensors, and autonomy architectures that were used to achieve fully autonomous mission execution. Most missions are centered around the COMPRA (Compact Reactive Autonomy) framework, that uses a fully reactive autonomy stack for tunnel navigation. The COMPRA-enabled UAV is also extended and deployed from both an integrated base-station, and from a legged robot (Boston Dynamics Spot) forming a multi-modality combined robot framework for complex mission execution. This chapter will also introduce an operator guided waypoint inspection framework, centered around a risk-aware gridsearch algorithm, that was deployed for visual safety inspection after blasting and gas con-

centration measurement missions. The goal and long-term aim of the hardware- and platform-supported autonomy architectures, and the demonstration experiments themselves, is to further the state-of-the-art and raise the industry interest in the area and use of autonomous inspections in underground mines. Finally, it should be clear to the reader that the works presented in this Chapter are collaborations where the author of this thesis did not on their own develop all the presented components. From both the hardware and software side there is a combination of in-house developments together with the use of industry and open-source components, all with the end-goal of achieving the best mission capabilities and demonstrations.

5.2 The Subterranean Environment

5.2.1 Introduction and Applications

Due to their large size, inaccessibility, and inherent risk to humans from cave-ins, the use of autonomous robots in subterranean environments has been a hot topic for some time. With the introduction of smaller-scale sensors and electronics, and now with novel smart algorithms for localization and navigation, the industry is more ready for the adoption of these new technologies. As developed on the academic side, the application areas range from mapping and inspecting caves [11, 148], inspection and safety in the mining industry [10, 36], monitoring for natural disasters [149] or even in the search for extraterrestrial life in subterranean systems on other planets or asteroids [150, 151]. There are many reasons for full autonomy when it comes to subterranean areas, but the main one is the almost impossibility of maintaining connectivity to a teleoperated robot outside of certain areas. Once the robot is out of line of sight, a comprehensive installed infrastructure is needed to enable teleoperation. The solution is of course to make the robots fully autonomous, such that they do not need any operator input after the mission is started. Seismic events, or the result of man-made excavation can do severe damage to such types of environments, as depicted in Figure 5.1. That possibility may block human workers from exiting dangerous areas, and makes it dangerous for human teams to carry out rescue operations in case of an accident. It is also not uncommon for persons to get stuck or go missing in natural caves. Both finding them and performing rescue operations presents a significant risk to any personnel involved in the operation. The same is true in case of natural disasters or war, where subterranean infrastructure like metros or road tunnels are highly risky to enter in case they are damaged. In conclusion, autonomous robots can reduce the risk to humans workers that perform a variety of tasks, and the rest of this Chapter will focus on two different types of tasks; robotized routine inspection missions, as well as missions in the context of search-and-rescue scenarios.

5.2.2 Navigation in Subterranean Field Environments

The harsh underground environment presents many challenges for deployment of autonomous robots, but it also presents several opportunities in what those robots can accomplish. From a robotics perspective, these environments heavily challenge the onboard perception systems. Complete darkness limit the use of any visual sensors, as even with onboard illumination their

Figure 5.1: Examples of rockfalls after seismic events in the underground mines [152]

performance is significantly degraded. The high levels of dust can also disrupt most sensor devices such as LiDAR systems or cameras (and LiDAR dust filtering and de-noising is a very active topic [153, 154]). Additionally, tunnel environments are very self-similar which can cause issues for onboard SLAM systems when performing map-based loop closure. Exploratory navigation in narrow or curving tunnels is also challenging due to the limited field-of-view of, for example, 3D LiDARs. Floor and ceiling planes will have limited visibility as most LiDAR beams will be hitting the tunnel walls, and completely mapping them for a complete reconstruction might necessitate dangerous maneuvering in the narrow space (e.g. by flying close to the ceiling or walls). For aerial robots, propeller downwash can kick up excessive amounts of dust if the robot navigation system takes it too close to the ground, and for ground robots many areas will be inaccessible due to the ground terrain traversability (water, mud, sand-like rock dust). On the same note, natural caves or damaged man-made mines can have very narrow and unstructured openings that require careful 3D navigation to pass through. Some of these environment challenges are exemplified in Figure 5.2.

Figure 5.2: Navigation Challenges in subterranean caves and mining environments; high levels of dust and water combined with narrow constrained passages.

As such, significant effort has to be placed on the selection of sensors and the tuning of SLAM algorithms when deploying robots in large-scale subterranean areas. Onboard navigation systems are also heavily stressed by the constrained tunnel areas as there is limited space for collision-free motion, and a collision is always only a couple of meters away in case there is a failure in any perception or localization system (which is likely to occur at some point due to the above listed perception challenges) and as such robust collision avoidance is heavily stressed (which was a motivator for the development of the APF in Chapter 3). In man-made underground tunnels, and mines specifically, irregular wind gusts are also a constant presence

due to the ventilation systems that are needed for humans to be able to access these massive underground tunneling areas, and onboard control systems have to be resilient to disturbances and tuned accordingly.

5.2.3 The DARPA Subterranean Challenge

The area of autonomous subterranean search-and-rescue has grown significantly in the last years due to the fact that it was the focus and primary selected application area of the DARPA Robotics Challenge [122]. The goal of the challenge was to deploy teams of robots, both tele-operated and autonomous, into tree different types of fully unknown subterranean areas. The three target areas were: tunnels (mines), urban areas (metro), as well as natural caves. In those selected challenge "circuits", specific *artefacts* of interest had been placed or hidden inside, mimicking a person to be saved, a lost piece of equipment, or a gas leak. Scoring for the competing teams was done based on how many artefacts they could find and locate within a desired localization accuracy, highly stressing the onboard SLAM (Simultaneous Localization and Mapping) modules as well as object detection in highly sensor-degraded environments. During the challenge, the two critical robotics problems of onboard SLAM [155–157] and exploratory navigation [129, 142, 158] saw multiple novel directions and extensions, and multiple teams have published works overviewing their complete deployment frameworks [123, 159, 160]. The presented results in this Chapter in Section 5.5 on the topic of search-and-rescue can be seen as part of the NeBula autonomy framework [123, 155, 158] and the Team CoSTAR [161], although they were not deployed as part of the competition itself. But, as the reader will see, the mission set-up, specifications, and the desired outcome of the demonstrations are heavily influenced by the SubT Challenge objectives.

5.2.4 Underground Mines

Although obviously similar in its challenges, the use of autonomous robots in underground mines must serve a specific purpose that benefits the mine. Due to the above listed challenges for subterranean navigation, one might assume that this environment is not suited for autonomous robot missions, and autonomous UAVs specifically, due to the challenges when it comes to dust, darkness, and constrained navigation. But, as reviewed in the following survey [162], UAVs have significant applications in the mining industry as well. Manually operated UAVs have been used for gas concentration measurements [163], visual and thermal inspection of rock mass characteristics [164], inspection of the distribution of rock fragmentation (or muck piles) after blasting [165], and for providing visual inspection of unreachable areas [166]. And as the reader will see, this Chapter will provide demonstration experiments of several of these use-cases as well, executed via fully autonomous missions. Interestingly, the survey lists two of the main challenges for UAVs in underground mines as; communication and connectivity for out of line of sight operations [167, 168] and the inability for drone pilots to access nearby safe areas to operate from. Both of these challenges can be answered by autonomous deployment as the operator is fully removed and autonomous vehicles can operate completely without communication links after the mission is initiated. In conclusion; the sig-

nificant advantages and applications of autonomous UAVs make up for the related challenges, and drive the development of systems robust to such harsh environments.

5.3 Contributions

The contributions presented in this chapter consist of the development of two autonomy architectures, and their very extensive evaluations in search-and-rescue scenarios, and in the autonomous inspection of mines. The contributions are as follows:

- The development of the COMPRA framework. A reactive navigation stack specifically designed to efficiently explore and navigate through mining tunnel areas. The framework is designed around rapid deployment and fast navigation using only local reactive components and is completely free of occupancy-map based path planning. The framework is made complete by the addition of necessary components such as LiDAR-Intertial SLAM and a simple return-to-base capability.

- Towards the SubT Challenge concept, COMPRA is also extended with a pipeline for object detection, localization, and validation, with the goal of detecting and localizing objects of interest in a global frame of reference during the exploration process.

- COMPRA is also extended to be deployed from a legged Boston Dynamics Spot robot, for complex mission execution where the legged robot acts as a carrier for the aerial platform in order to reach previously unreachable (for the legged robot) areas. Similarly, COMPRA is deployed from an in-house designed base-station platform towards having UAVs as part of the mining infrastructure. Here, COMPRA is also extended with a vision-based guided landing sequence to land back into the platform after mission completion.

- The COMPRA stack is extensively evaluated first in search-and-rescue scenarios in realistic tunnel environments, and then towards the routine inspection and mapping of mining areas which is demonstrated in real operational mines. COMPRA is also demonstrated to navigate and inspect areas not safe for human workers after a simulated rockfall.

- The second field deployment framework is centered around the risk-aware DSP path planner [136], and will be denoted as the RIA (Routine Inspection Autonomy) framework. This framework was designed as an operator guided general inspection framework of previously mapped environments. Here, an operator provides a set of waypoints to be inspected, which are optimized via a travelling salesman problem. The waypoints are then navigated to safetly by a combination of DSP and the local control and avoidance stack from Chapter 3.

- The inspection framework is deployed for the use-cases of visual safety inspection after blasting, and for autonomous gas concentration missions, which are two application missions of high interest from the industry as they present significant risk to human workers.

In general, the use-case demonstrations of autonomous robots executing application-driven full missions in real field environments contributes to both the general academic research in field robotics, but through the related collaborations and research projects also increases the trust from the industry in these novel technologies and can serve as part of a road-map towards full industry integration of autonomous inspection robots.

5.4 The COMPRA Framework

5.4.1 Framework Overview & Motivation

The COMPRA (Compact Reactive Autonomy) was put together as a kit with the capability of executing a complete inspection mission into a previously unmapped area. The idea is simple; leverage the environment to your advantage and simplify the tunnel navigation problem into the motion directive of "follow the tunnel centroid while avoiding obstacles", the same idea as presented in Section 3.5 where repulsive forces regulate the position and heading states of the vehicle to do just that. After a certain time has passed, follow the travelled breadcrumb path back to where the mission was initiated. Generalizing, the COMPRA kit is composed of the local autonomy (control & avoidance) framework described in Chappter 3 and the same source of state-estimation [26], with some additional modules added on top. The local kit is combined with a tunnel following algorithm denoted as the Deepest-Point Heading Regulation (DPHR) technique, a pipeline for detecting objects of interest, as well mission executions behaviors. The UAV platform used can be seen in Figure 5.3 with all its sensors (as a note: The Ouster OS1 was swapped with a Velodyne Puck Lite in some of the missions), onboard computer, and other hardware. The complete COMPRA architecture can be found in Figure 5.4, showing all the autonomy components and the mission behavior design. Notably, the two operator inputs to the system are: the time duration of the exploration mission until return is triggered T_{explore} and the desired mission height from the ground the UAV should fly at p_z^m (the z-reference so-to-say) based on the local measurements $p_z^{\mathcal{L}}$ from an onboard single-beam LiDAR.

It should be highlighted that the COMPRA framework is fully focused on rapid deployment and quick navigation, not full coverage of the area. As will be demonstrated the simple mission structure of COMPRA enables very consistent and robot-safe mission execution in tunnel environments. The operator also has control of which initial tunnel/direction the UAV should follow as opposed to the UAV freely selecting that based on, for example, an optimal frontier. The fully reactive nature of COMPRA also means that navigation speed can be significantly pushed, which will be demonstrated in the following experiments, as the next waypoint generation happens in real-time and is independent on transitioning from one generated trajectory to the next, which means consistent smooth navigation. In general the COMPRA kit proved to be very efficient and consistent when deployed in critical scenarios, where specific capabilities (ex. inspecting an area after a rockfall) was to be demonstrated.

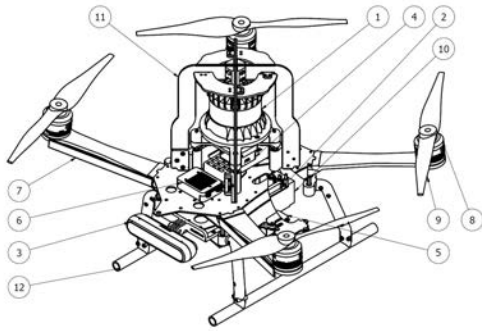


Figure 5.3: The utilized custom-built UAV for field missions. 1 - Ouster OS1 3D LiDAR, 2 - Intel NUC, 3 - Intel Realsense D455, 4 - Pixhawk Cube Flight Controller, 5 - Garmin singlebeam LiDAR, 6- Telemetry module, 7 - LED strips, 8- T-motor MN3508 kV700, 9- 12.5in Properllers, 10 - Battery, 11 - Roll cage, 12- Landing gear

Frame Notation

For the sake of ease and clarity in notation in the following sections about COMPRAs additional modules, let us define specifically the frames of interest. The previous parts of the thesis has not particularly been detailed on this, but as these modules work on camera data it is a necessity. The world frame $\mathscr{W}$ is fixed with the unit vectors $\{x^{\mathscr{W}}, y^{\mathscr{W}}, z^{\mathscr{W}}\}$ following the East-North-Up (ENU) frame convention. The body frame of the aerial vehicle $\mathscr{B}$ is attached on its base with the unit vectors $\{x^{\mathscr{B}}, y^{\mathscr{B}}, z^{\mathscr{B}}\}$, representing the rotated global coordinates $\mathscr{W}$ in along the z-axis. The $z^{\mathscr{B}}$ is antiparallel to the gravity vector, $x^{\mathscr{B}}$ is looking forward the platform's base and $y^{\mathscr{B}}$ is in the ENU convention. The onboard camera frame $\mathscr{C}$ has unit vectors $\{x^{\mathscr{C}}, y^{\mathscr{C}}, z^{\mathscr{C}}\}$. Furthermore, $y^{\mathscr{C}}$ is parallel to the gravity vector and $z^{\mathscr{C}}$ points in front of the camera. The image plane is defined as $\mathscr{I}$ with unit vectors $[x^{\mathscr{I}}, y^{\mathscr{I}}]$. Figure 5.5 depicts the utilized main coordinate frames of the aerial platform.

5.4.2 Depth-based Reactive Exploration

The exploration behaviour in this work considers the generation of local reference waypoints $p^{\mathscr{B}}_{ref}$ in 3D, as well as local yaw references ψ_{ref}.

The exploration waypoints follow the heuristic concept of constant value "carrot chasing" in the bodyframe x-axis of the platform, while we utilize a fully reactive heading regulation technique to align the MAV body x-axis with the tunnel direction. More specifically, the generated way-points $p^{\mathscr{B}}_{ref}$ always add a constant value ahead of the x-axis, while the motion in the y-axis in $\mathscr{B}$ frame depends only on the potential fields input. The waypoints are defined as $wp^{\mathscr{B}} = [p^{\mathscr{B}}_x + 1, p^{\mathscr{B}}_y, p^m_z]$, and are then fed to the potential field to generate the obstacle-free $p^{\mathscr{B}}_{ref}$. The local altitude reference p^m_z (or the mission altitude) is kept constant and selected be-

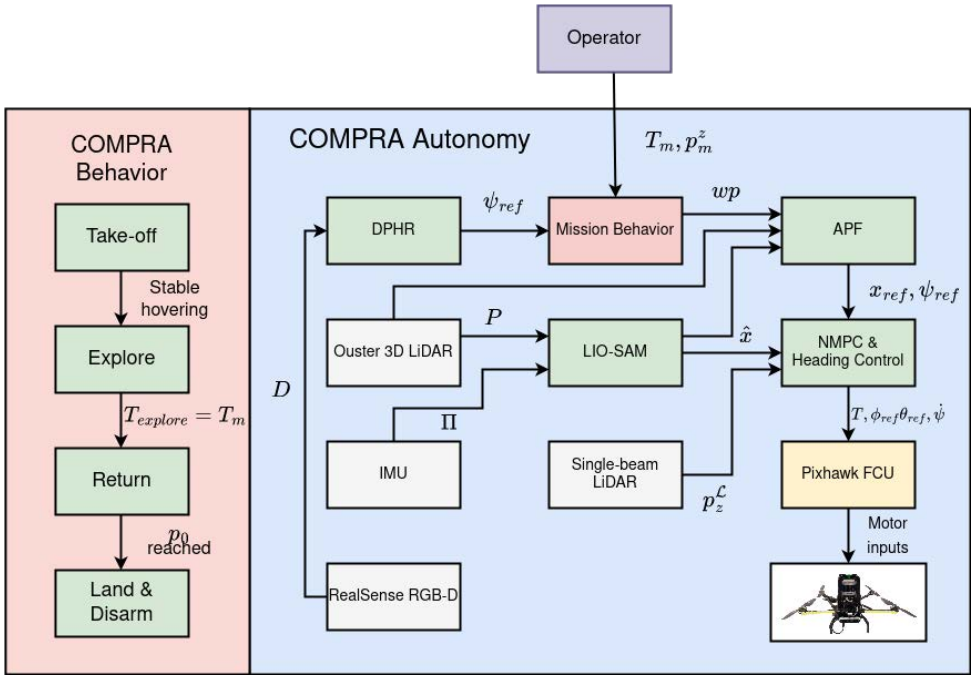


Figure 5.4: The COMPRA Autonomy Architecture and mission behavior. The COMPRA mission is initialized with the desired T_m and p_m^z, and explores the tunnel area until the exploration time $T_{explore} = T_m$, at which point a return to base is triggered and the UAV backtracks back to the initial deployment point p_0. From LiDAR pointcloud P and IMU data Π (gyroscope, accelerometer) Lidar-Ineratial Odometry generates the robot state $\hat{x}$ (position, velocity, Euler angles), while a single-beam LiDAR measures the distance to the ground $p_z^{\mathbb{L}}$ (local z-coordinate). DPHR generates heading references ψ_{ref} from a depth-image D for continuous forward exploration following the waypoint wp. The APF generates collision-free state references to the control system x_{ref}, that are translated to optimal control inputs in thrust, roll, pitch, and yawrate signal as $u = [T, \theta_{ref}, \phi_{ref}, \psi_{ref}]$ to the onboard flight control unit (FCU).

fore the mission starts, and the measured local z-position is defined by the range measurements, R_{sbl}, from the single-beam LiDAR as $p_z^{\mathbb{L}} = R_{sbl}\cos\theta\cos\phi$, e.g. the distance to the ground.

The sensor information used in the DPHR method is the instantaneous depth image $D^{\mathscr{C}}(x^{\mathscr{I}}, y^{\mathscr{I}})$ provided by the onboard RGB-D sensor, where $x^{\mathscr{I}}$ and $y^{\mathscr{I}}$ denote the pixel coordinates. At each iteration the pixel information from $D^{\mathscr{C}}$ is processed and converted into yaw rate command (rad/s) without keeping past sensor information. $D^{\mathscr{C}}$ is initially preprocessed into $\tilde{D}^{\mathscr{C}}$ with a morphological closing operation [169] using a rectangular kernel structuring element of size 20. This operation is considered a filtering step used to remove noise caused from hardware sensor imperfections or environmental conditions and enhance the pixel intensities of the most distant areas in the tunnel. After the filtering step the $\tilde{D}$ is segmented into a discrete number of regions denoted as $N_{clusters} \in \mathbb{Z}$ that are populated with pixels with high similarities.

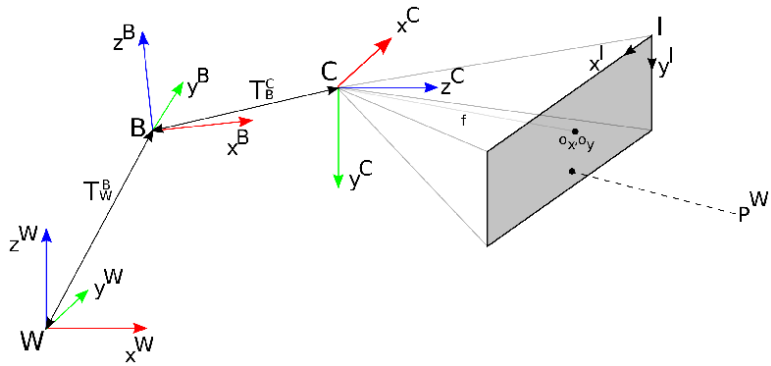


Figure 5.5: Coordinate frames, where $^{\mathscr{W}}$, $^{\mathscr{B}}$, $^{\mathscr{L}}$, $^{\mathscr{C}}$ and $^{\mathscr{I}}$ denote world, body, LiDAR, camera and image coordinate frames respectively.

The multiple clusters are generated using a k-means method in an environment related number of cluster centers $C_i, [i = 1,2,3,\ldots,N_{clusters}] \in \mathbb{R}^2$, where COMPRA defaults to $N_{clusters} = 10$. Each C_i includes pixels with similar intensity levels, which are used to calculate its mean intensity value. The cluster selection part is based on the mean intensity value of each cluster, converting the grayscale image to binary image $\tilde{D^{\mathscr{C}}}_{binary}(x^{\mathscr{I}}, y^{\mathscr{I}})$ by thresholding C_i with the maximum mean intensity. The $x^{\mathscr{I}}$ pixel coordinate of the cluster centroid is then considered since the rate of change of $s_x^{\mathscr{I}}$ is mainly affected when the camera is undergoing a yaw motion ψ. Finally, the cluster centroid (s_x, s_y) is calculated based on the binary image moments M_{pq} as shown in Equation(5.1)

$$M_{pq} = \sum_{x^{\mathscr{I}}} \sum_{y^{\mathscr{I}}} x^{\mathscr{I}p} y^{\mathscr{I}q} \tilde{D^{\mathscr{C}}}_{binary}(x^{\mathscr{I}}, y^{\mathscr{I}}) \tag{5.1}$$

using $s_x = \frac{M_{10}}{M_{00}}$ where M_{10} is calculated with $p = 1$ and $q = 0$ and M_{00} is calculated with $p = 0$ and $q = 0$. Finally, s_x is normalized and transformed with respect to the image principal point $\bar{s_x}$ and converted to a yaw rate reference $\dot{\psi}_{ref} \in$ [min max], using $\dot{\psi}_{ref} = \bar{s_x} * l$, where l maps linearly the yaw rate to min and max values. Figure 5.6 showcases a snapshot from the implemented deepest point extraction process.

5.4.3 Object Detection & Localization

For the mission as defined by the DARPA subterranean challenge, and in general for a mission centered around finding objects of interest such as a search-and-rescue mission, a critical capability is detecting and localizing objects. This section will present a pipeline for object detection, localization, and finally validation, that can be seen in Figure 5.7. The framework is utilizing the visual RGB data I, the depth image stream D, as well as the estimated robot states $\hat{x}$, where specifically $\hat{p}$ is in the world frame coordinates.

The object detection part is based on the tiny and Intel hardware optimized version [170]

Figure 5.6: snapshots of the DPHR methodology depicting: on the left the extracted centroid (marked black circle) of the open tunnel area and on the right the binarized image of the segmented area with the higher depth values.

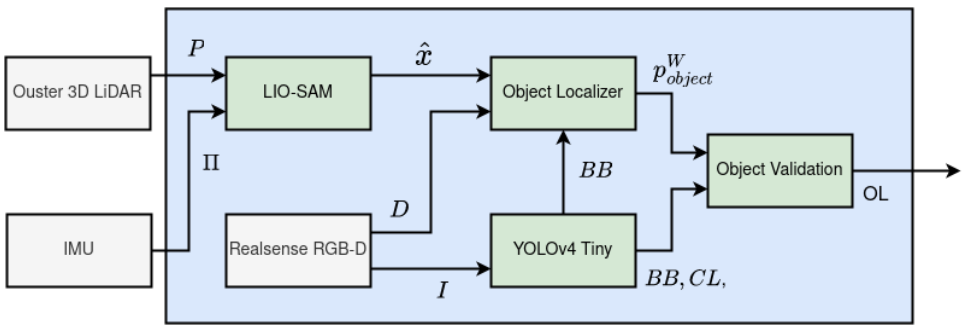


Figure 5.7: The pipeline for detecting, localizing, and validating objects of interest or "artefacts" in the environment during the COMPRA mission.

of the state of the art CNN object detector Yolo V4 [86]. We trained the network to detect and classify 6 classes $CL \in [1, \cdots, 6]$ defined by the SubT competition using a custom dataset consisting of approximately 700 images for each class. The input size of the images is 416 $\times$ 416 and the output of the algorithm are the detected bounding boxes and the class probability $Pr_{CL} \in [0,1]$. The other component of the pipeline is the object localizer, which receives the bounding box $BB = (x^{\mathscr{I}}_{min}, y^{\mathscr{I}}_{min}, Wd^{\mathscr{I}}, Ht^{\mathscr{I}})$ measurements of one of the predefined object classes from the RGB image stream I, where $x^{\mathscr{I}}_{min}$ and $y^{\mathscr{I}}_{min}$ denote the minimum $x^{\mathscr{I}}$ and $y^{\mathscr{I}}$ axis pixel coordinates and $Wd^{\mathscr{I}}, Ht^{\mathscr{I}}$ denote the width and height in pixels. The localizer transfers the identified bounding box in the aligned depth image stream D and extracts the relative position of the object in the camera frame $\mathscr{C}$, defined as $p^{\mathscr{C}}_{object} = [p^{\mathscr{C}}_x, p^{\mathscr{C}}_y, p^{\mathscr{C}}_z]$. Frequently, the extracted bounding boxes include part of the background with the object of interest, thus to avoid this issue we calculate the object position considering only a 3$\times$3 window around the centroid of the bounding box. The main assumption is that the centroid always is projected to the object of interest. Finally the object location is converted in the global world frame

$\mathscr{W}$, defined as $p^{\mathscr{W}}_{object}$, using the transformation $p^{\mathscr{W}}_{object} = {}^{\mathscr{W}}T_{\mathscr{C}}\ p^{\mathscr{C}}_{object}$, where ${}^{\mathscr{W}}T_{\mathscr{C}}$ denotes the transformation matrix from $\mathscr{C}$ to $\mathscr{W}$, defined as ${}^{\mathscr{W}}T_{\mathscr{C}} = [R|t]$. The final component is the object validation, structured around two subcomponents, i) the buffer of measurements and ii) the processor of buffered measurements. The buffer stacks artifact positions using a two-step outlier rejection process. Initially, it accepts only bounding boxes with class probability above a specified threshold ($Pr_{CL} \in [Pr_{threshold}, 1]$) and are located inside a sphere with radius of 5 meters (based on the depth camera range) and secondly, removes detections when their metric bounding box width bounds are outside a fixed width interval for each known object $Wd^{\mathscr{W}} \in [Width^{CL}_{min}, Width^{CL}_{max}]$, to address false positive inputs with high class probabilities but where the size of the object does not match an assumed already known size interval. Afterwards, the processor of buffered measurements is initiated once a specified time window from the last observation in the buffer has passed. During this process, the buffered values for each class are clustered based on Euclidean distance and the mean value of the positions of each cluster is calculated. Additionally, the current clusters are compared against already localized objects using Euclidean distance to deduce whether it will belongs to previously seen object or it will be reported as a new observation. Once this step is finished the buffer is cleaned. This architecture can handle multiple observations of the same class at different locations in the same buffer session. The localizer returns a list $\mathbb{OL} = \{\vec{0}_{3\times1}, p^{\mathscr{W}}_{object1}, p^{\mathscr{W}}_{object2}, \ldots, p^{\mathscr{W}}_{object,n}\}$, where n is the number of detected objects. Figure 5.8 presents the overall architecture on the strategy related to the object detection and localization. Some examples of false-positive detection hits can be seen in Figure 5.9, where the object validation rejects them based on the size of the object and class probability, which was very common in the dark, and in this case also graffiti-filled, subterranean tunnels.

5.4.4 Mission Behavior & Return-to-base

The final subcomponents of the COMPRA mission are the baseline switching mission behaviors, and the return-to-base after the mission is completed. In a COMPRA mission, these are not complex systems, and rely mainly on the mission duration as compared to the desired exploration time T_{explore}. The COMPRA return to base relies on following a breadcrumb trail of waypoints back to the place where the mission was initiated while the potential field still provides collision-free motion. Obviously there can be scenarios where the optimal return path is not the same as the exploration path, but using this simple method we ensure that the robot returns by navigating through areas that are already visited and confirmed safe, while also preventing state-estimation drift by only traversing already known areas for map-based loop closure. We can also keep the COMPRA navigation occupancy-map free where we do not risk any issues related to traditional path planners, removing a possible source of mission failure. The return-to-base records a list of breadcrumb waypoints at a certain time interval as $\mathscr{BW} = [\mathscr{BW}_1, \mathscr{BW}_2, \ldots]$ with each waypoint consisting of position and heading (yaw) references as $[\hat{p_x}^{\mathscr{W}}, \hat{p_y}^{\mathscr{W}}, \hat{p_z}^{\mathscr{W}}, \hat{\psi} + \pi]$, e.g. following the previously travelled path but with the heading in the opposite direction as to increase the chance of detecting a previously obscured artefact on the way back.

The mission behavior of COMPRA can be found in Figure 5.4 and consists of four stages:

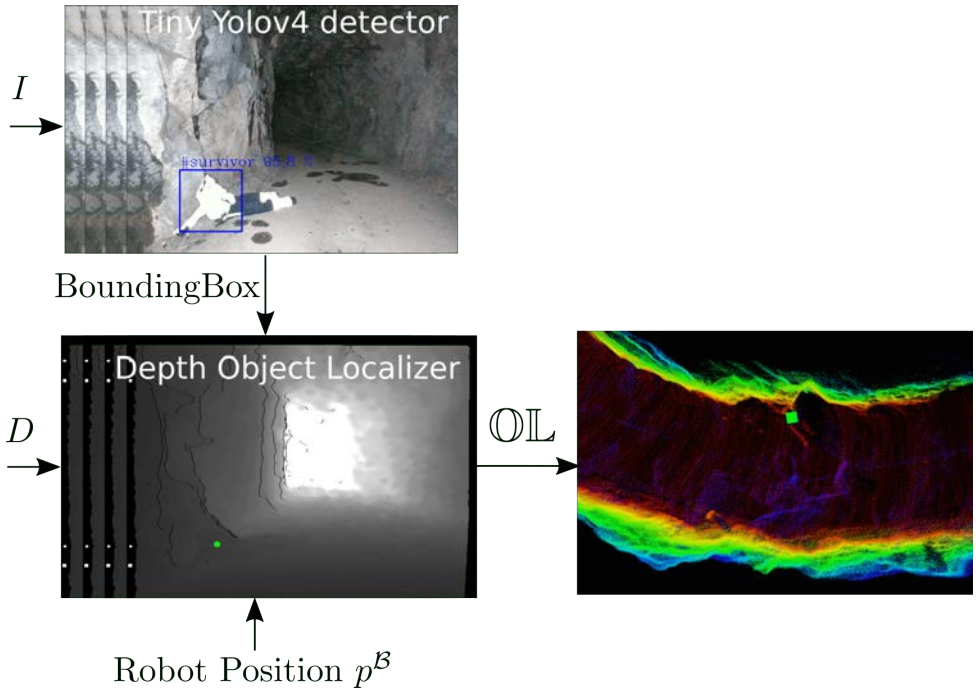


Figure 5.8: Visualization of the object detection pipeline: The YOLO modules provides bounding boxes with artefacts and the localizer combines depth camera images and the global-frame robot pose estimate to localize artefacts in the generated map.

Figure 5.9: Examples of false positive detections during COMPRA missions, which are the reasons for the object validator. The validator rejects these detection hits on the basis of size of the object and the detection confidence.

1) Initialization and take-off. The UAV arms itself (goes into autonomous mode) and initiates a take-off sequence based around not aggressively tracking the position and heading states to limit excessive initial maneuvering as the risk of state-estimation drift is largest during the initial upward acceleration if movevents are rapid. 2) Once stable in the air, the APF and DPHR modules take over and the exploration stage is executed by reactively following the main tunnel

branch by aligning the heading to the deepest visible area. 3) The mission timer T_m is equal to the desired exploration duration $T_m = T_{\text{explore}}$ the return-to-base is initiated. Breadcrumb waypoints are followed while keeping the APF active for local navigation. 4) When the UAV has reached the end of its return trajectory a landing command is triggered and the UAV lands and disarms itself, ready to be collected or deployed again by an operator. This stage will also be extended in Section 5.6.2 by a visual servoing based guided landing, for a precise landing maneuver to land back into a deployment base station.

5.5 Search-and-Rescue

The following section will detail the experimental evaluation of COMPRA missions in the search-and-rescue context. The missions are performed at an underground tunnel facility below Mjölkuddsberget, Luleå, Sweden, as well as at a mining facility at the Epiroc Test Mine near Örebro, Sweden. In general, these mission are constructed in the context of shorter directed missions where artefacts of interest are placed further into the unknown environment, and the goal is to navigate through the tunnels while mapping the area and detecting the artefacts. The experiments include narrow tunnels, wide tunnels, voids ,dust & darkness, and in general present a variety of different navigation conditions. This section also includes the combination of COMPRA with another autonomy framework [136, 171] for a legged robot, where the legged robot acts as a carrier of the aerial robot, and the aerial robot provides the capability to enter into unknown areas that could be blocked from entry by a ground-based platform. Together they form what we denote as a multimodality robot framework, that can combine the maneuverability of the UAV with the long battery life of the legged robot, for more complex mission execution. This section will shortly summarize the multimodality framework and the motivation behind it, before its field evaluation.

5.5.1 COMPRA Deployment

The COMPRA stack is deployed in fully autonomous complete missions into unknown subterranean tunnel environments. The following video link provides the real-time navigation and detection behavior for the following experiments to be presented (and a few more): `https://www.youtube.com/watch?v=xHmeX7a8A3g`. Figures 5.10-5.13 show the generated pointcloud maps of the environment with the global-frame artefact detection hits, and the green line highlights the exploration (and return) paths of the robot. The figures also show snapshot images from the onboard camera for critical moments during the mission such as artefact hits or moments of stressed collision avoidance.

Figure 5.10 shows a mission in a curving narrow tunnel, where many artefacts were placed along the tunnel. Figure 5.11 shows a mission in a narrower tunnel, with multiple junctions where the UAV stays in the main branch. That area also had difficult obstacles along the way such as a metallic scaffolding (see snapshots in figure) where the reactive APF saves the robot and maintains a safe distance. Figure 5.12 shows a shorter mission into a very narrow (less than 2 m in places) and upward sloping tunnel, where the COMPRA reactive system keeps the UAV comfortably in the center of the tunnel despite the very difficult terrain. Figure 5.13

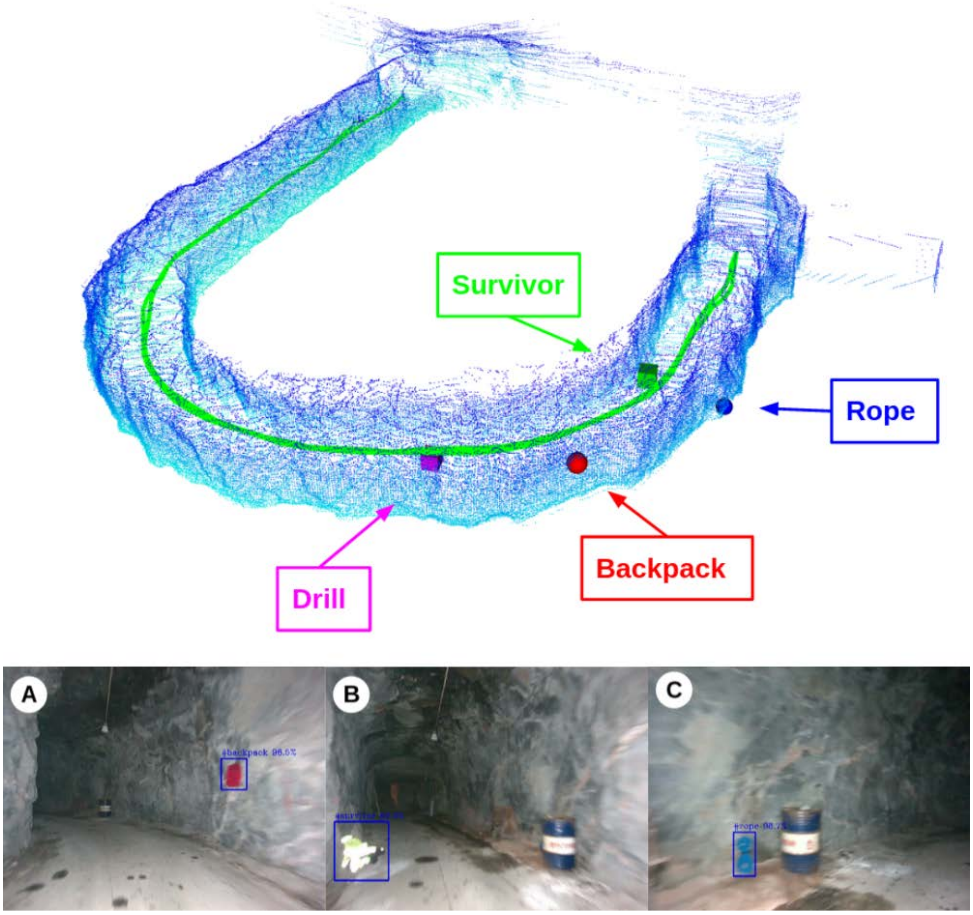


Figure 5.10: COMPRA mission in a curving tunnel environment approximately 3 meters wide. The figure shows the generated map with detected artefacts (top) and snapshots from the onboard camera during the mission (bottom).

shows a mission from the Epiroc Test Mine, in significantly wider and more open tunnels of around $10-12\,\mathrm{m}$. The video also shows two more missions in an open void area as well as at the Callio Pyhäsalmi Mine, Pyhäjärvi, Finland. In all these missions the average exploration speed was around $1\,\mathrm{m/s}$.

In general, 15 out of 17 artefacts (all missions in the video) were detected and found, and there was zero false-positive detections that passed the object validation process. As can be seen in the Figures there were also no duplicate detections of the same artefact. There was no ground truth to validate the precision of the localization and object detection system though.

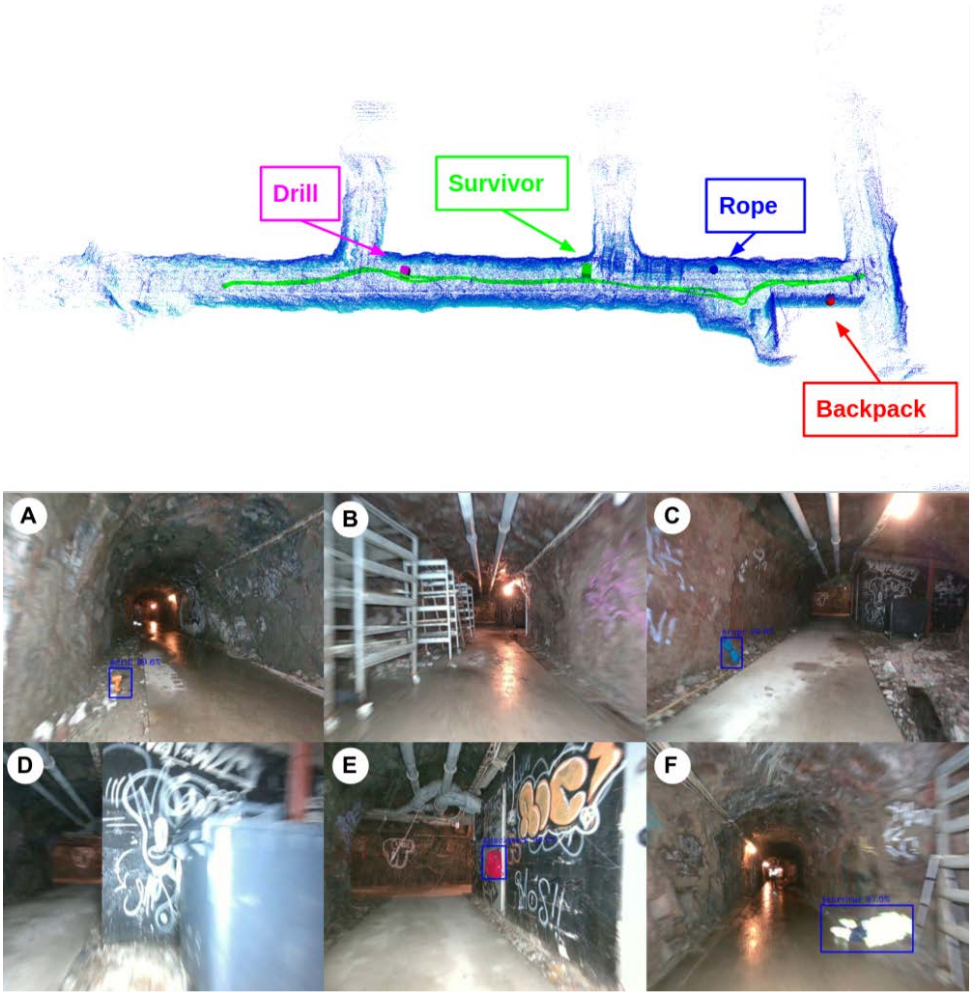


Figure 5.11: COMPRA mission in a obstacle filled tunnel section and critical moments during the mission. (A), (C), (E), (F) shows artefact detection hits while (B) and (D) show collision avoidance situations during the mission.

Throughout all missions, the COMPRA navigation stack provided safe navigation where the UAV is kept at the center of the tunnel, and provides smooth continuously forward exploration. The closest the UAV ever got to an obstacle or wall was 0.54 m, based on the LiDAR range measurements. COMPRA is demonstrated to work well in a variety of tunnel environments and has the capability to enter into and navigate through narrow constrained tunnels.

We performed an extra shorter experiment without artifacts in the curving tunnel area (the area in Figure 5.10) trying to push the navigation speed. We could easily reach an average of

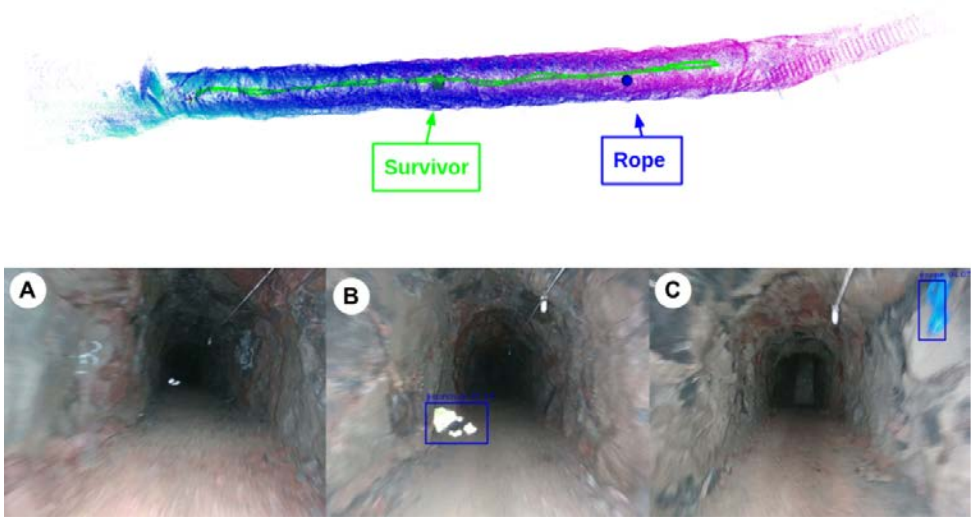


Figure 5.12: Shorter COMPRA mission in a narrow cave of less than 2 meters width, with an upward slope.

2.3 m/s after the initial acceleration, and the velocity (magnitude) from that mission can be seen in Figure 5.14. In the COMPRA framework, the mission velocity is not constrained by computation time, map update rates, or similar navigation-related limitations due to its purely reactive nature, and as such to increase the velocity we simply increase the weights related to position reference tracking in the NMPC for a more aggressive following of waypoints. As seen, the velocity can be kept relatively consistent throughout the mission as there are no "transition points" from one trajectory to the next since the waypoint is continually updated by the reactive exploration scheme. The dips seen in Figure 5.14 simply represent instants where the adaptive weights scheme reduced the speed due to the proximity of an obstacle, and the velocity never dropped below 1.6 m/s. The smoothness of the navigation even at high velocities in constrained environments (curving mining tunnel) is a major outcome of the COMPRA framework.

High-speed navigation in narrow or constrained subterranean environments is a very difficult problem, and other related works are evaluated (in real-life constrained environments) up to: 0.1 m/s [10], $0.4-0.5$ m/s [172], 0.5 m/s [173], 0.5 m/s [174], 0.75 m/s [114], as compared to the $0.9-1$ m/s full mission speed and the 2.3 m/s of the high-velocity COMPRA mission. As far as the author knows, the fastest field deployment subterranean flight outside of COMPRA was 1.8 m/s [110]. Although not a perfectly fair comparison as the COMPRA navigation stack is specifically designed to reactively explore and navigate tunnels and can not do much more than that, the comparison can highlight effectiveness of the reactive navigation method.

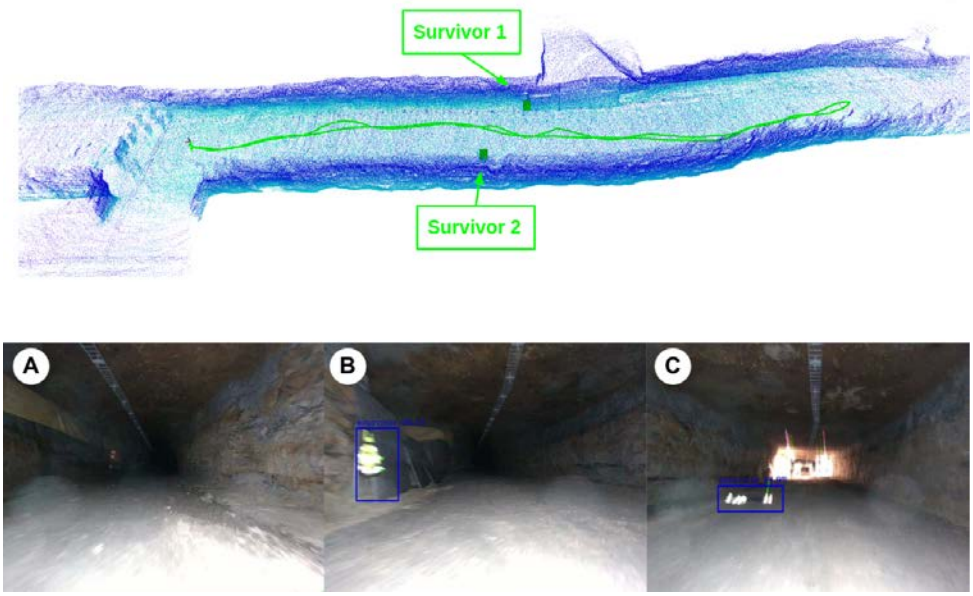


Figure 5.13: COMPRA mission in a wider tunnel at the Epiroc test site. Total mission length of around 200 meters. Detected reflective jackets were placed to mimic ”survivor” artefacts.

5.5.2 Multimodality Robotic Missions

Framework Overview

A separate line of research towards the DARPA SubT challenge objectives and robotic seach-and-rescue focused on the combination and collaboration of different robotic platforms. Different platform types can bring different capabilities to the table such as payload, operational duration (battery), terrain traversability, and of course the 3D maneuverability only aerial platform bring to the table.

In contrary to ground robots, aerial robots have very limited battery endurance when equipped with extra payload that incorporates a full sensor suite with onboard computation power. This payload is required to allow them to execute a fully autonomous exploration or search-and-rescue task and limits their maximum mission duration e.g. there is a sharp trade-off between autonomy level and flight time [123]. However, for aerial robots the traversability level of ground terrain is irrelevant, and access to blocked passages is only limited by the size of the aerial robot. As discussed, subterranean terrain, especially after a seismic event or other damage, can offer extreme challenges when it comes to traversability of the terrain (rocks, loose gravel), and certain passages or tunnels that contain survivors or other objects of interest might

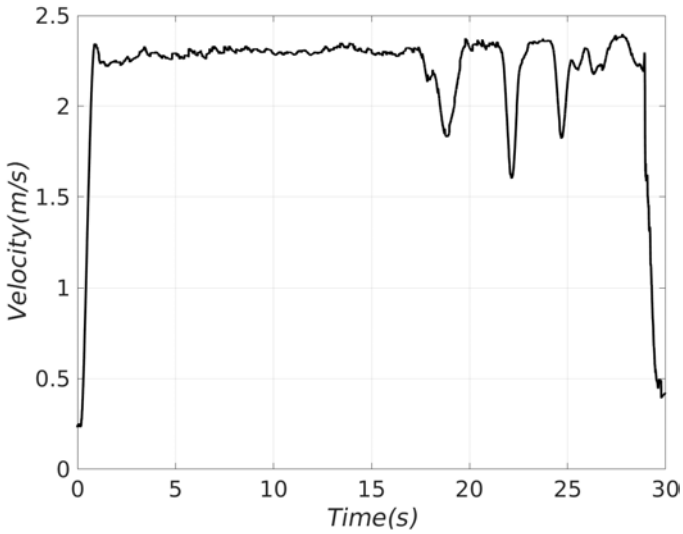


Figure 5.14: Navigation velocity magnitude for a shorter mission in the curving environment shown in Figure 5.10. COMPRA maintains a high velocity throughout the mission, and averages around

$$2.3\,\mathrm{m/s}$$

.

be blocked off completely from ground robots.

Due to these kinds of challenges, hybrid locomotion and multimodal robots have shown promising results in a variety of tasks where extending the terrain traversability of the robot is critical, such as; wheeled-legged robots for inspection tasks in difficult terrain [175], or for combining ground and aerial capabilities, robots such as the Drivocopter [176] have been attempted as well.

This section of the thesis will present a different direction, where a legged ground robot acts as the carrier of an aerial agent forming a multimodality combined robot system, seen in their full sensor suits in Figure 5.15, thus mitigating the expended flight time to reach the desired exploration location (e.g. the blocked passage, or hard-to-reach area) without sacrificing mobility. As pictured, the legged robot is also equipped with a custom-built landing platform that locks the UAV in place when it is not operational. The legged robot carries the same sensor and computation kit as the aerial robot.

This thesis will not go into detail on the legged robot autonomy kit. Instead the reader can find more information in the related works [136, 171] and in the published work on multimodality robots [129], but a high-level description of the mission behavior execution can be seen in Figure 5.16. In short, assuming a partially known map of the area an operator defines a deployment point p_{dep} that the legged-aerial system should navigate to, using the risk-aware path planner DSP [136] in combination with the Cartographer localization framework [177].

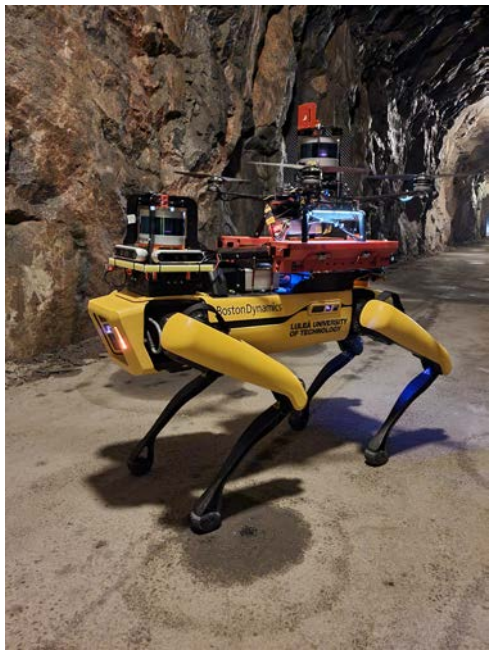


Figure 5.15: Multimodality robotic system for field deployment. The Boston dynamics Spot is combined with a custom-built quadcopter.

The operator defines the tunnel direction that COMPRA should be deployed in as well as the mission parameters T_{explore} and p_z^m. Once p_{dep} is reached, the aerial autonomy is initiated and the COMPRA mission can begin. The legged robot, the UAV, and the landing platform communicate with simple commands and flags that define when a certain action is completed but otherwise do not otherwise coordinate between them during mission execution.

Field Evaluations

The multimodality mission is evaluated in the same subterranean compound as the COMPRA-only missions previously discussed. Two missions are included in this thesis that prioritize different components of the combined mission: 1) A longer mission that simulates as well as possible the DARPA SubT Challenge scenario. The robots start at the entrance to the compound, and the deployment point is selected around 120 m into the area in a junction. The COMPRA framework is then directed to explore an unknown section of the tunnel where multiple artefacts have been placed and return to the deployment point. At the time of performing these missions the capability to land back on the platform with high precision in the low-light conditions was not yet developed, but a similar scenario is considered later in Section 5.6.2. 2) The second mission is a shorter mission where we target the blocked passage scenario. Here an entry to a tunnel is blocked and not accessible to the legged robot. Instead, the UAV can be

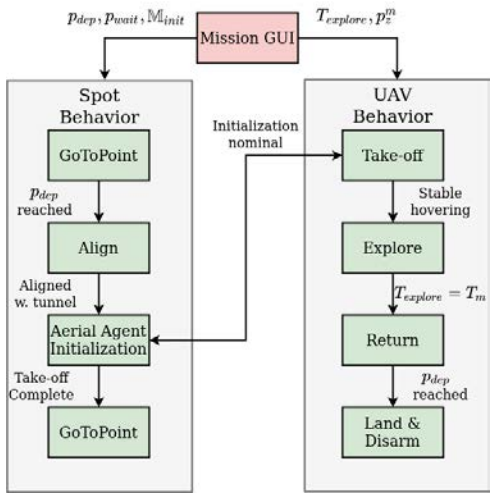


Figure 5.16: Simple mission execution workflow for the combined legged-aerial mission that was used in the following field evaluation.

deployed over the barrier to find the survivor placed in the blocked tunnel.

For the longer mission, the reader should watch the real-time mission execution at the following video link: `https://www.youtube.com/watch?v=0p56NkUD_Q8`. Figure 5.17 show mission snapshots during critical moments during the combined mission, while the complete generated pointcloud map, that includes the traversed path and the detected artefact can be found in Figure 5.18. Figure 5.19 shows the detected and localized artefacts discovered during the mission.

The blocked passage mission can similarly be seen in Figure 5.20, where we have simulated a blockage of one of the passages. The UAV maneuvers over the blockage and can reached the constrained tunnel where the legged robot could not. The survivor artefact is located in a place that could not be reached by a single robot, highlighting the unlocked capabilities of the combined framework. Figure 5.21 shows the map generated by the robots, and the total volumetric gain as the explored volume by both robots, again highlighting the extra area reached by the deployed aerial agent.

In general: utilizing the presented legged-aerial quick deployment framework, the mission was easily repeatable without inconsistencies in mission behavior and without any submodule failures such as the robots getting stuck, environment interactions, or large state estimation drifts or jumps.

Figure 5.17: Snap shots from the full mission. Spot-UAV system approaching deployment point (top left), deployment point reached (top right), aerial agent take-off (middle left), hard hat detection from onboard camera during COMPRA mission (middle right), survivor detection (bottom left), and aerial agent return to deployment point (bottom right).

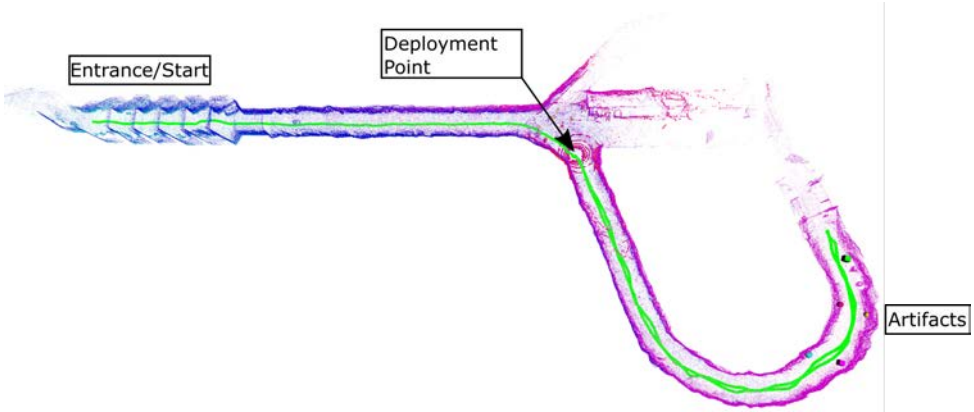


Figure 5.18: Generated pointcloud map and traversed path for the multimodality framework during the mission, highlighting the start, deployment point, and location of artefacts.

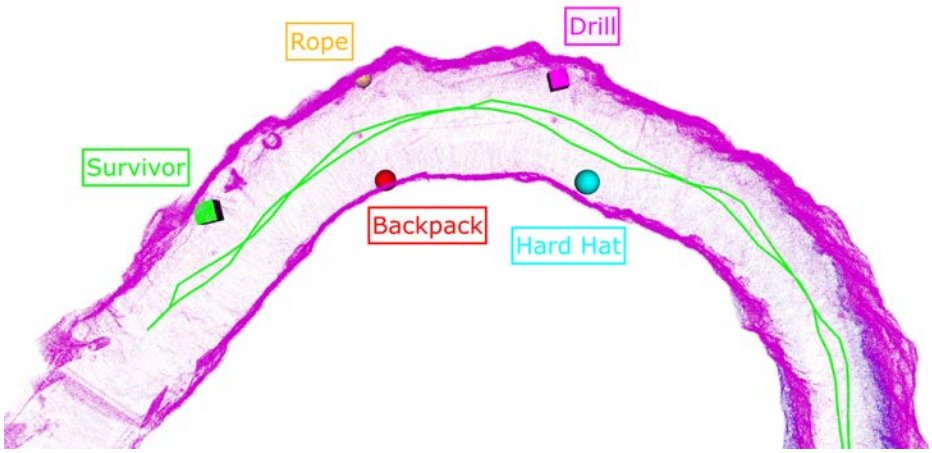


Figure 5.19: Artefacts detected and localized during the mission.

5.6 Towards Routine Robotized Inspection of Underground Mines

As has been discussed previously, underground mines present very challenging conditions for autonomy, but due to the inherent risk to human workers performing routine inspection tasks after blasting, or surveying a closed-off area after a seismic event, they also offer significant industry-driven use-cases for autonomous field robots. This section will continue to present the deployment of the COMPRA framework but in this case not for search-and-rescue mission but as a tunnel inspection and mapping framework. COMPRA will also be extended by

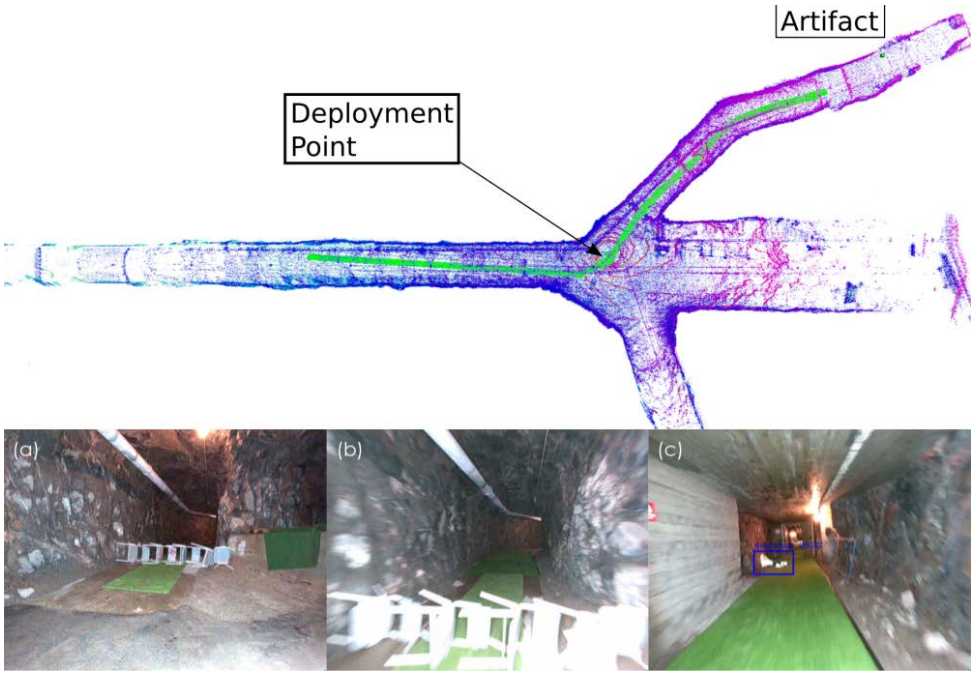


Figure 5.20: Simple mission execution workflow for the combined legged-aerial mission that was used in the following field evaluation.

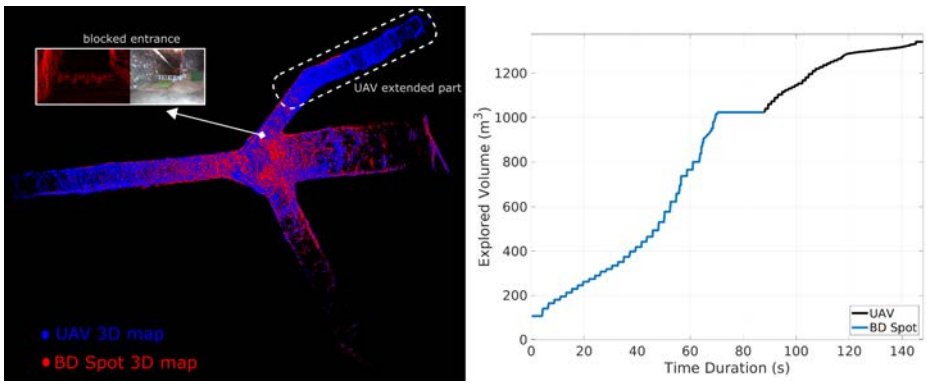


Figure 5.21: Simple mission execution workflow for the combined legged-aerial mission that was used in the following field evaluation.

the addition of a high-precision guided landing using a downward facing camera, such that is can be deployed and land back into an integrated base-station. Finally, this section will also overview another autonomy framework focusing on a more generalized approach to in-

spection missions where an operator selects a series of waypoints to be inspected through a GUI in a known map of the area. The thesis includes two evaluation scenarios for that framework: visual safety inspection after blasting in a salt mine, and the autonomous monitoring of gas concentration through an onboard sensor. As the following demonstration are much more related to direct industry collaborations and research project, the author would like to thank our collaborators in the mining industry for the rare opportunity in evaluating our autonomy frameworks in realistic mining conditions; Epiroc Rock Drills AB, K+S Group, and Luossavaara-Kiirunavaara AB (LKAB). Additionally, the research projects that enable such a collaboration; illuMINEation [178], Next Generation Carbon Neutral Pilots for Smart Intelligent Mining Systems (NEXGEN SIMS) [179], and the Sustainable Underground Mining (SUM) Academy Programme [180].

5.6.1 Inspection and mapping of tunnels with COMPRA

Tunnel areas in underground mines can benefit from routine inspections, whether that is remapping an area after a blast and drilling operation, creating the local map of the tunnel area if it was not mapped beforehand, or surveying a damaged unsafe area. These inspection tasks can be tedious, time consuming, and expensive when performed by workers carrying camera and LiDAR sensors around, in addition to the potential danger when entering certain areas. This section will demonstrate how the COMPRA framework can be deployed for that purpose. Two environments are used for the evaluations: The Epiroc Test Mine near Örebro, Sweden, and in the largest underground iron mine in the world, operated by LKAB in Kiruna, Sweden, where tests were performed at over 1200 m depth under the ground.

For an overview of three missions performed at the Epiroc Test Mine, the reader should view the experiment video at: `https://www.youtube.com/watch?v=4aXwCiOEfx4`. The three experiments were as follows: 1) A scenario where various rock piles and other obstacles were added to a tunnel area that COMPRA was to navigate through that can be seen in Figure 5.22, 2) A longer mission (without return) in a huge multi-branching tunnel area that can be seen in Figure 5.23, and finally 3) A scenario that simulates a rockfall or tunnel collapse. Here a large almost 2 m high rock pile was added to block off the tunnel, and the COMPRA-enabled UAV was sent to investigate and survey the area behind the simulated rockfall, where no human worker would be allowed to enter. That mission is visualized in Figure 5.24. In all three scenarios the reactive COMPRA framework provides smooth and consistent navigation behavior through the tunnels, and provides a 3D reconstruction of the area combined with visual inspection data from the onboard camera. The UAV could easily navigate over and past the blocked area and provided critical inspection data as to the state of the tunnel on the other side. As the mission is easily directed towards the tunnel area to be explored and inspected, and the flight relatively fast while always keeping in the center of the tunnel, the missions could very quickly and rapidly be deployed to inspect the critical unsafe or unmapped areas.

Finally, we evaluate the COMPRA framework for its efficient and quick exploratory navigation in an LKAB tunnel at over 1200 meters depth. The mission is to explore the tunnel for as long as possible without returning to base, in order to evaluate the framework on a very large scale. Due to COMPRA's reactive nature, both its navigation speed and scalability are

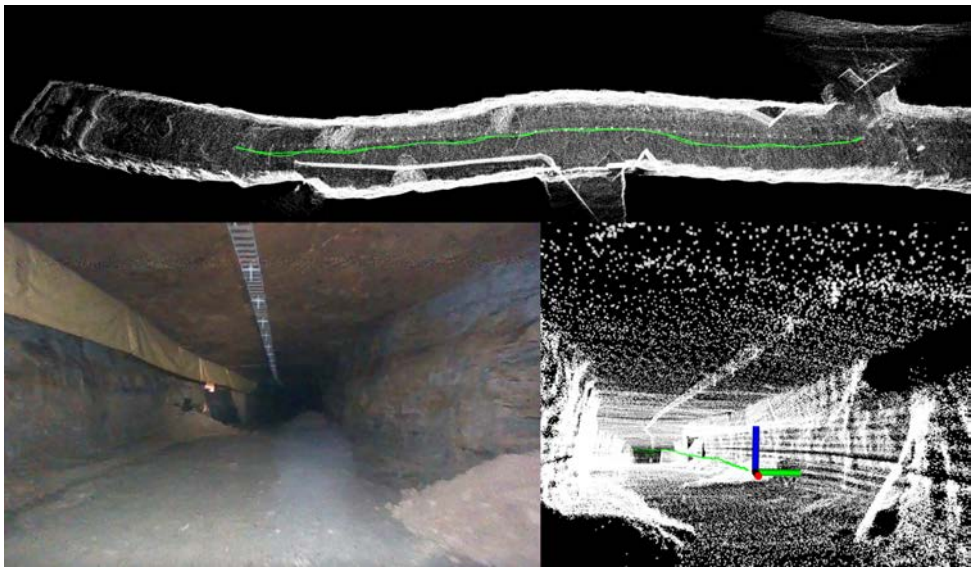


Figure 5.22: Resulting pointcloud reconstruction from fully autonomous exploration mission, and sample of visual inspection data from onboard camera. Exploration path for COMPRA autonomy (green). Around 200m total flight path.

unrelated to computation times from grid-search path planning or similar. Figure 5.25 shows the resulting surveyed tunnel area where the COMPRA stack explored over 800 m of tunnel fully autonomously, at a rapid 1.6 m/s exploration speed. To the best of the authors knowledge, this would constitute the largest subterranean area explored by a single UAV mission in the literature and was executed in less than 10 minutes. The LIO-SAM [26] framework also showed great scalability and resilience to self-similarity during this large-scale tunnel mission.

5.6.2 Deployment from integrated base-stations

A challenge when it comes to inspection of hazardous areas or as a disaster response is that due to the massive size of the mines, an autonomous robot can not always reasonably be deployed by an operator on the surface or from a distant safe area in the mine in order to perform the safety inspection tasks. Instead, autonomous robots can become part of the infrastructure in the mine, being deployed from integrated base-stations that offer protection from the environment outside of operations while allowing faster access to the area of interest, charging, and data transfer. This concept is already a reality and sees partial use in the mining industry [181]. This section will extend the COMPRA framework to be deployed from a custom-built base-station through a visual servoing based guided landing for high precision landing maneuvers. This can also be seen as an addition to the previously discussed multimodality framework, where landing back on the legged robot could become a possibility with this extension. Figure 5.26 shows a custom built and 3D printed demo-version of such a base-station capsule, specifically

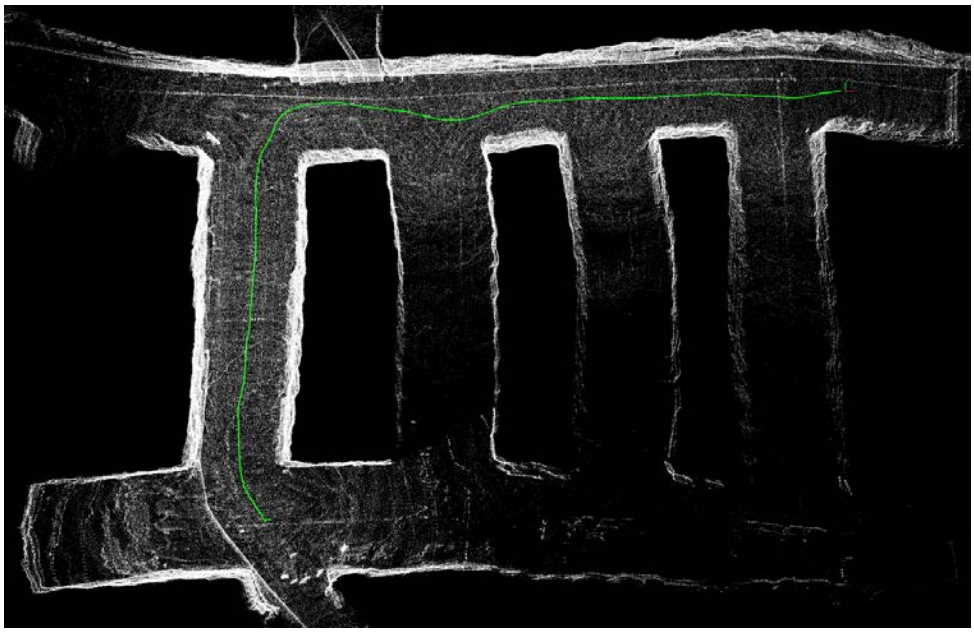


Figure 5.23: Resulting pointcloud reconstruction from a larger-scale COMPRA mission in a tunnel environment. Exploration path for COMPRA (green).

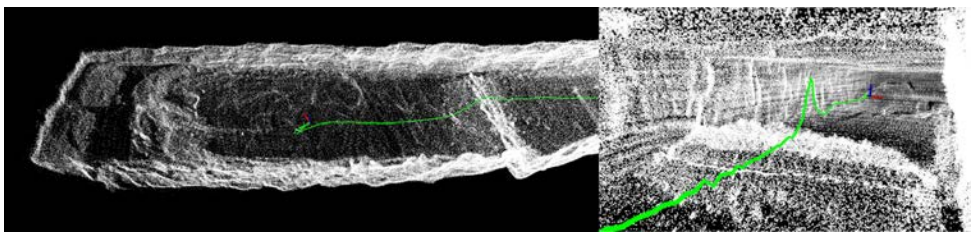


Figure 5.24: Navigation over in a rockfall scenario (approx. 2m high rock pile placed to fully block the tunnel), and resulting pointcloud reconstruction of the other side.

designed for the custom built quadcopter (Figure 5.3) that has been used throughout the thesis.

The landing system is using a downward-facing monocular camera, and a QR-code placed on the landing platform, where we used the tracker in [182]. Additionally, simple commands confirming task-completion are sent between the UAV and the capsule's computer. The reason guided landing is needed is that very slight drifts or errors in state-estimation during the mission could lead to the origin pose no longer being perfectly aligned to the center of the landing platform. Only a few centimeters of error could lead to an unsuccessful landing, especially when compounded with the general state-estimation noise and controller imperfections, and as such loop-closure with a tracked target is needed. For adding guided landing to the

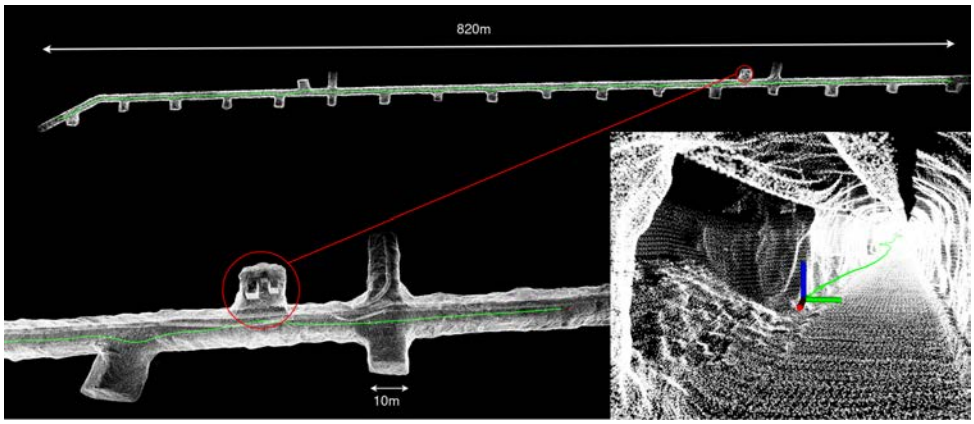


Figure 5.25: COMPRA exploration of a large-scale mining tunnel. The complete map (top), navigation path (in green) and reconstructed infrastructure (bottom left), and navigation past and reconstruction of a rockpile (bottom right).

COMPRA mission, the visual servoing module is handled through the same NMPC used during the mission, where weak integrators are added to the roll, pitch, and yawrate commands to slowly eliminate model-mismatch and steady-state errors such that the UAV over time is perfectly aligned with the tracked QR code pose. The strategy to achieve high accuracy in landing precision was slow movements as there is both state-estimation and tracking measurement noise during the guided landing, exacerbated by low-light conditions. The landing is triggered only if the average values of a buffer of distance and heading angle measurements of the relative QR-code pose are below the desired accuracy bounds (here 4 cm, and $0.05 rad$, to ensure a successful landing and ”lock in”). The landing system was tested in an area the reader should be familiar with at this point, that being a curving tunnel area at Mjölkuddsberget, Luleå, Sweden, consisting of around 3.5 m wide tunnels where the base-station capsule is placed at a central juction area. The mission execution can be found in the following video link: `https://www.youtube.com/watch?v=fCht4aQfP2M`, while figure 5.27 display critical moments during the mission. Through the guided landing maneuver, the UAV smoothly lands back with only a few centimeters drift despite low light, estimation noise, and the downwash effect from the rotors. This demonstration and addition to the COMPRA framework shows the potential to add autonomous UAVs as part of the mining infrastructure to enable inspection missions without any operators entering the mine.

Towards a similar direction, the guided-landing enhanced COMPRA mission was also deployed from the mining machine itself. As the machines are already operating in the mine, they can act as the carriers of the UAV platforms for their missions, or rather the UAVs can act as the eyes of the machine to perform precise scouting och inspection tasks around the machine. For example the autonomous planning of extraction operations after blasting could utilize the UAV that is deployed from the mining machine to generate a reconstruction of the area before the machine is sent in. COMPRA was deployed from an Epiroc machine at the

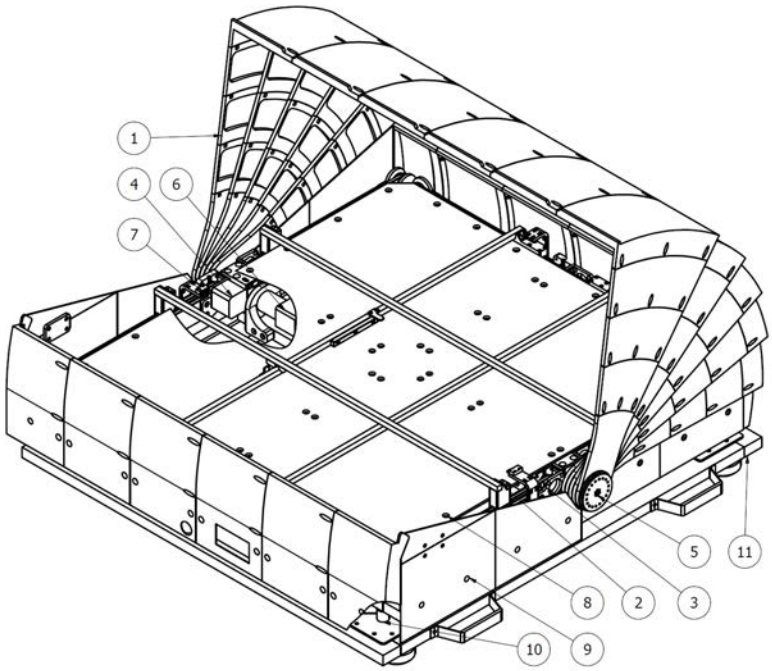


Figure 5.26: Demo version of a base-station capsule for drone mission integration. The capsule opens as the COMPRA mission is initiated, and closes as the guided landing sequence is completed. In case the landing is imperfect, the design allows the UAV to be mechanically re-aligned to the start position. 1 - Canopy element, 2 - Push bar 3 - Push bar linear actuator, 4 - Push bar stepper motor, 5 - Canopy pivot, 6 - Canopy stepper motor, 7 - Limit switch (end stop), 8 - Landing plate, 9 - Side cover, 10 - Vibration dampener, 11 - Main frame

Epiroc Test Mine and we used the guided landing to land back on a landing platform on top of the machine, landing back with an accuracy of a few centimeters. The mission can be seen at: `https://www.youtube.com/watch?v=6KZTQVDWuQY`.

5.6.3 A General Operator-guided Inspection Framework for Mining Areas

Framework Overview

This framework relates to performing inspection missions in a known and already mapped area, which we will denote as the Routine Inspection Autonomy (RIA) framework. The problem formulation that RIA is to solve can be seen as: from a set of operator provided waypoints or view points in the known map, plan the safest and shortest route to visit all inspection points fully autonomously while re-mapping the area and again providing inspection information from

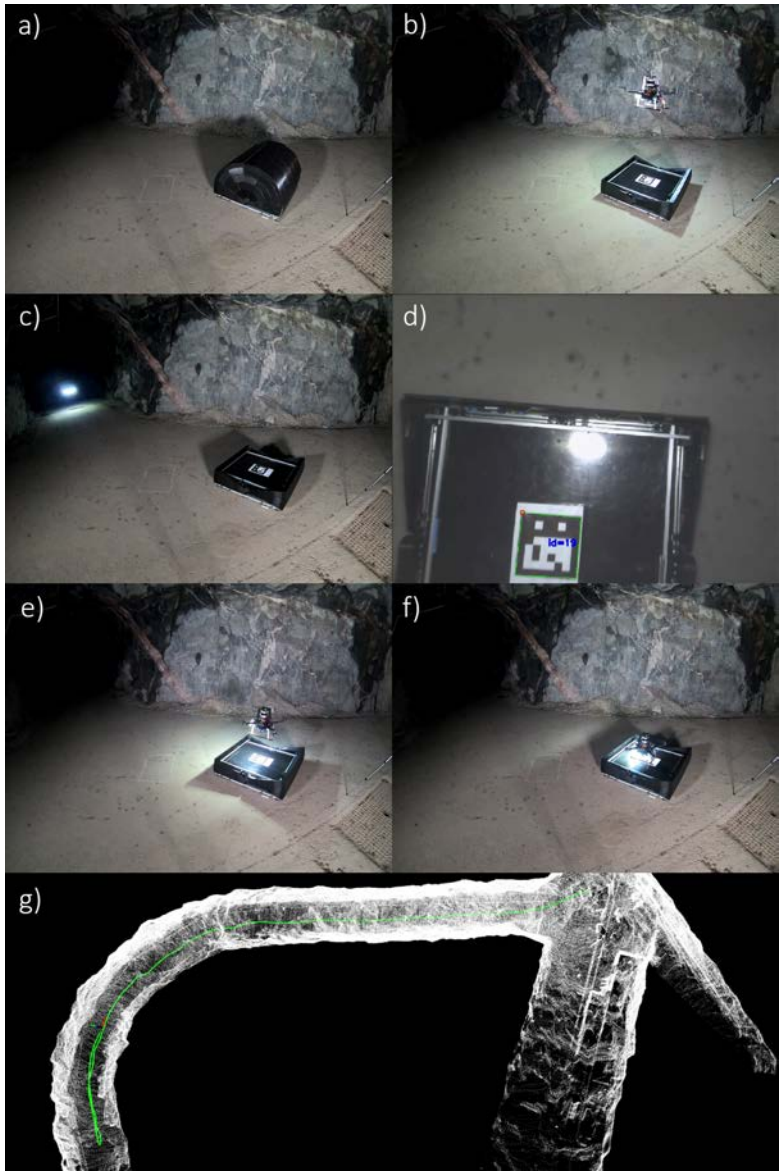


Figure 5.27: Simple mission execution workflow for the combined legged-aerial mission that was used in the following field evaluation.

whatever sensor payload is onboard the UAV. This framework assumes that there exists an occupancy map [99] of the area that is used for path planning, and a pointcloud map that is

used for localization and for clicking the waypoints to be inspected. The complete framework can be seen in Figure 5.28, with a descriptive caption, where the inputs to the framework is a list of waypoints to be visited, as well as the occupancy and pointcloud maps of the area. In this framework, the DSP [136] 3D gridsearch path planner is combined with a travelling salesman problem (TSP) route optimization [90] and relocalization in pointcloud maps via the 3DEG framework [183]. Supportive autonomy modules is the Octomap [99] occupancy mapper, and the local autonomy presented in [184] (Chapter 3), with the exception that FAST-LIO [185] is used for localization as it supports re-localization on a provided known map. Autonomous UAVs have the potential perform inspection missions of active mining areas in order to, for example, secure that the integrity of the mining face and hanging walls are maintained or that gas concentrations are below dangerous levels, without the need for human workers to enter the area and subject themselves to the potential risk. Additionally, routine inspections in tunneling areas can provide operators with information regarding rockfalls, misplaced or broken equipment, faulty pipes, water leakage, etc. in the inspected section of the mine, that could then be used to update mine plans for more efficient operations.

Field Evaluations

Field trials were performed for two critical use-case scenarios, that being: 1) Visual and LiDAR inspection after a blasting operation to make sure that the hanging walls, mining face, and the tunnel itself are safe for the mining machines and workers to enter. This scenario was performed at the K+S Group Werra salt mine in Germany, in a large-scale mission scenario where the UAV visits multiple drifts and mining faces. And 2) a scenario where the UAV has been equipped with a gas measurement sensor (Dräger X-AM 5000), where we have simulated a scenario with an increased carbon monoxide (CO) concentration to be located (one of the critical gasses to be weary of after blasting in the mine) by running the diesel engine of a large mining truck at the Epiroc Test Mine. Central to the demonstration is the ability to link the real-time gas measurements with the onboard localization system such that each measurement is associated with a position coordinate in the mine. The mission setup for the after blasting mission can be seen in Figure 5.29 that shows the operator-provided inspection waypoints as well as the 3D pointcloud map used for re-localization. The full mission can be seen in the following video: `https://www.youtube.com/watch?v=6QtFWSjXWf0`, that highlights the extreme levels of dust in this specific mine which was a major challenge. Figure 5.30 shows the resulting optimized and safe inspection route from combining the DSP algorithm with the travelling salesman route optimization, while Figure 5.31 shows some snapshots from the mission execution and the resulting visual inspection.

The autonomous UAV successfully navigates through the area and visits the desired inspection points with a total mission length of over 350 m at around 0.9 m/s navigation speed, while demonstrating robot-safe and efficient navigation throughout the mission via the DSP path planner and the local autonomy kit. The constrained NMPC handled the wind gusts from the ventilation system, and the APF collision avoidance system could handle the large amounts of dust with some tuning efforts. The UAV enters areas that were deemed unsafe (as the ceiling had not been secured by rock bolts yet in some areas of the drifts), and provides fully autonomous inspection data to operators that could be located on the surface or kilometers

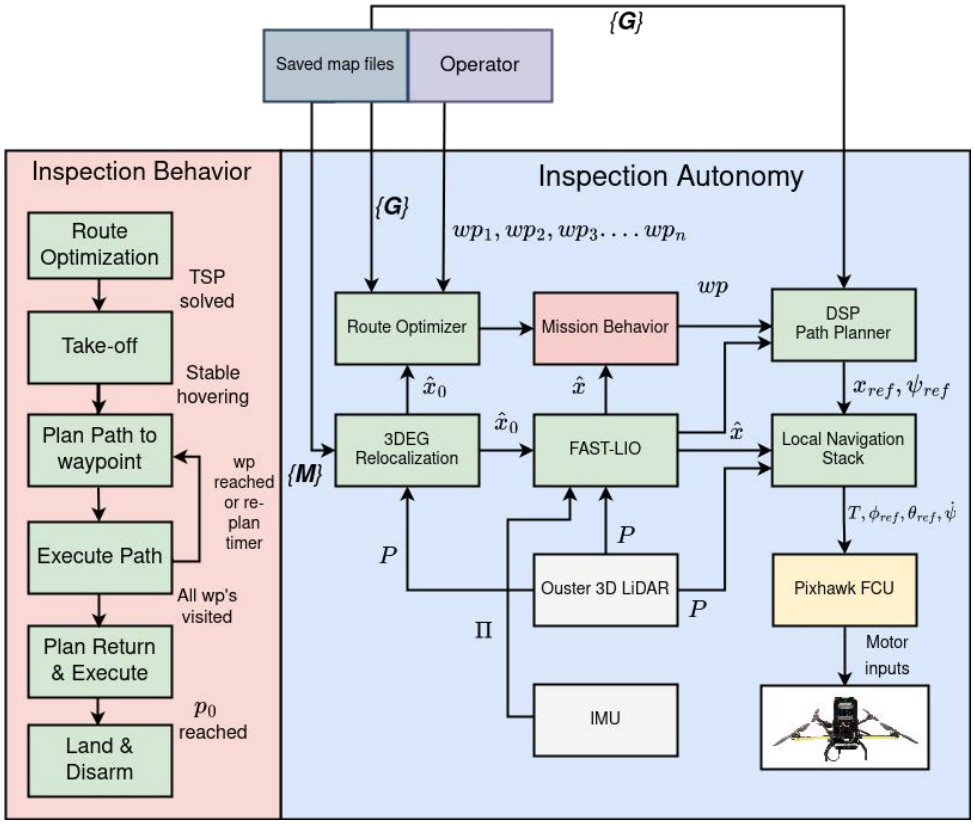


Figure 5.28: The Routine Inspection Autonomy (RIA) and mission behavior. Assuming a known point-cloud map $\{\boldsymbol{M}\}$ and occupancy gridmap $\{\boldsymbol{G}\}$ of the environment, the 3DEG relocalization framework estimated the initial robot state $\hat{x}_0$, which is given as an initial guess to FAST-LIO. After an operator provides a series of desired inspection waypoints $wp_1, wp_2, wp_3 \ldots wp_n$, a route optimization generates the optimal order of visiting waypoints, and after that the mission is initiated. The DSP path planner plans robot-safe paths to each waypoint in order, and the local autonomy stack executes the path. After all waypoints are visited, DSP plans the path back to the initial robot position p_0.

away if. This type of mission, if fully integrated to the mining workflow, would enable the mine managers and operators to much more quickly and efficiently orchestrate the extraction process after blasting, and without having to send any human workers into potentially unsafe areas. The gas concentration monitoring mission can be seen in Figure 5.32. This was a shorter mission, designed more as a proof of concept of autonomous UAVs operating in underground mining areas while equipped with gas sensors. Notably despite the propeller downwash the higher gas concentrations were located with relatively high accuracy around the exhaust of the mining vehicle, and despite low overall concentrations the sensor could easily pick up the dif-

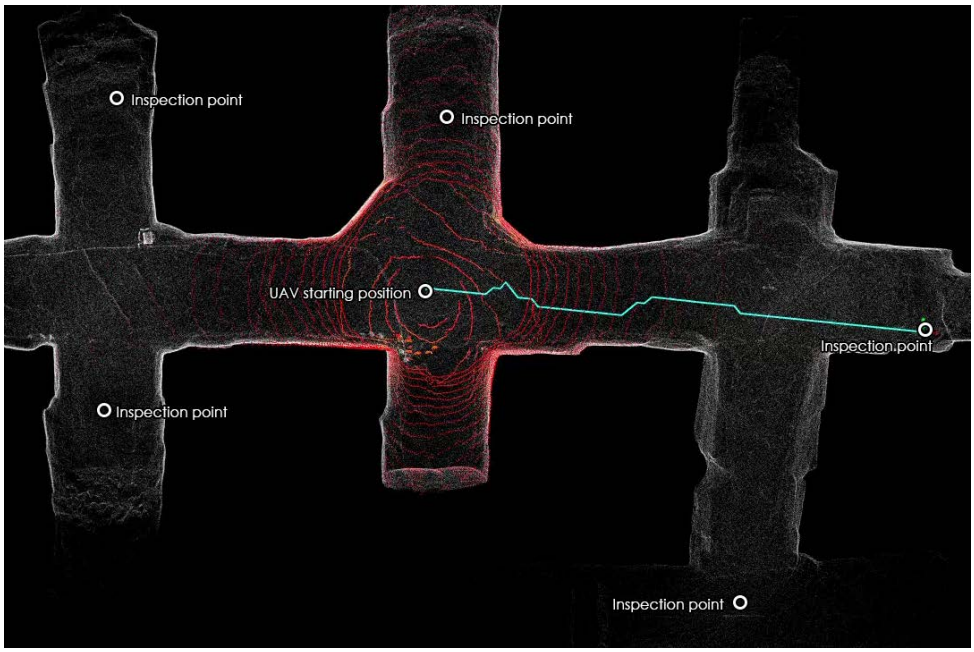


Figure 5.29: Pointcloud map used for planning and localization, and the desired inspection waypoints. The teal line shows the planned path to the first waypoint. The momentary raw pointcloud scan is shown in red.

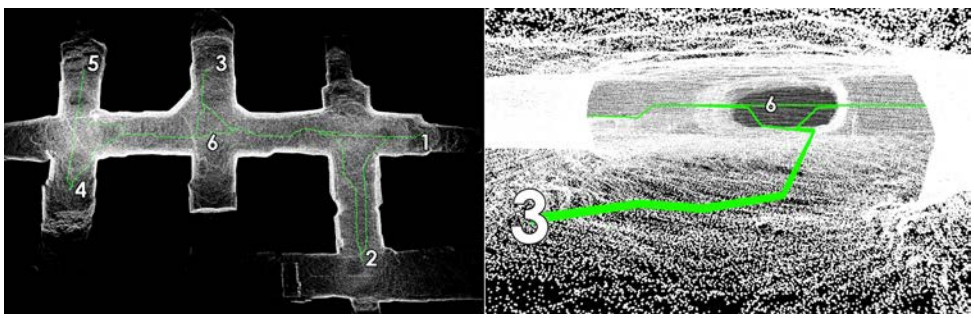


Figure 5.30: The full optimized inspection route (green) solved for using a Travelling Salesman Problem in combination with the DSP algorithm.

ference. This mission should be seen as an enabler for the measurement and localization of high gas concentrations, for example after blasting, without the need for any human operator to enter the mining area where the blast took place and expose themselves to potentially dangerous gas levels, which has potentially massive impact for the efficiency and safety of mining operations.

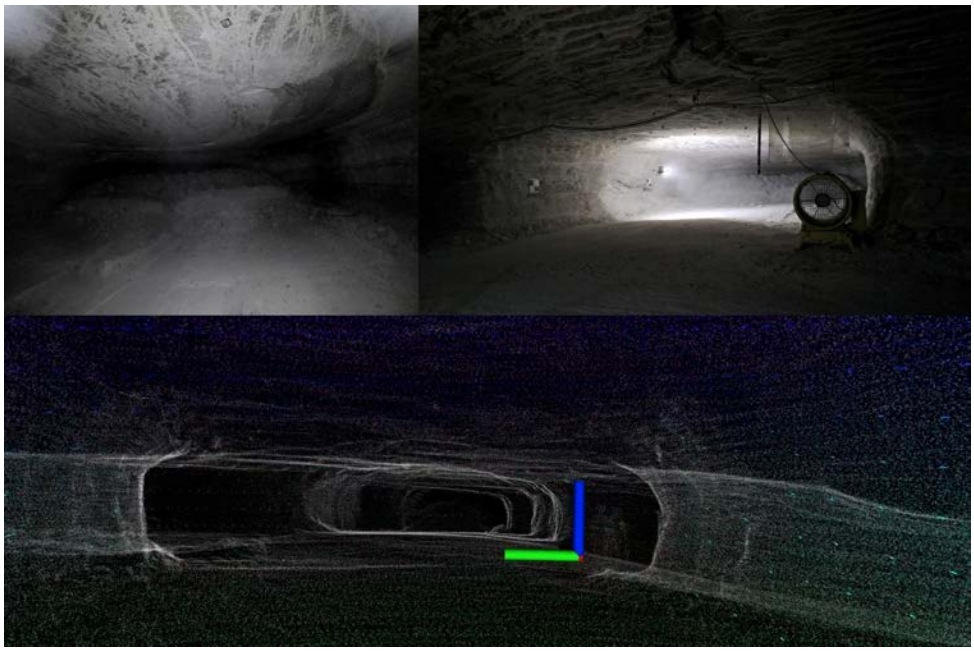


Figure 5.31: Visual inspection of a mining drift with a muckpile from onboard camera (top left), UAV during the autonomous inspection mission when entering a drift (top right), sample of pointcloud reconstruction of the area from onboard 3D LiDAR (bottom).

5.7 Concluding Remarks

This chapter has demonstrated the potential that autonomous robots have in the context of deployment in subterranean environments. These demonstrations were made possible through the developed COMPRA framework, a fully reactive navigation and perception stack that was deployed for a variety of missions and extensions, as well as the RIA framework used for an operator-guided inspection of mining areas. The COMPRA mission was deployed for search-and-rescue missions, with a focus on the context of the DARPA subterranean challenge, where the UAV successfully could navigate the subterranean tunnel areas, as well as detect and localize objects of interest. The COMPRA mission was then also deployed for routine tunnel exploration missions in real field mining environments. Additionally, the fully reactive navigation style of COMPRA allowed both very fast navigation (and as a result also very large-scale missions), as well as navigation in narrow and constrained areas. Successful demonstrations of extensions to the COMPRA framework were also included where the COMPRA-enabled UAV was deployed first from a legged robot in the search-and-rescue context, and then from an integrated base-station capsule where a visual-servoing guided landing could provide a successful landing back into the base-station capsule. The RIA framework was deployed in very harsh real mining conditions to perform visual inspection and remapping after blasting in a

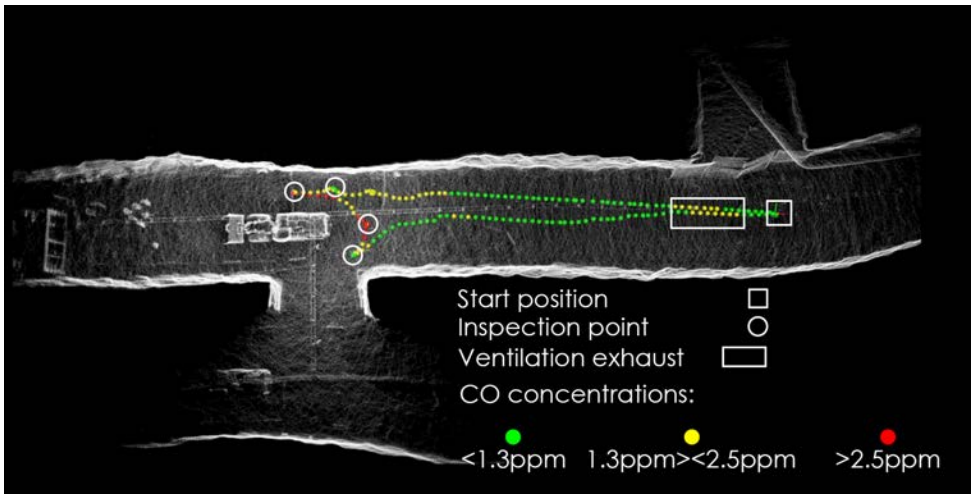


Figure 5.32: Simple mission execution workflow for the combined legged-aerial mission that was used in the following field evaluation.

salt mine, as well as the monitoring of gas concentrations. In all instances the RIA navigation stack provided robot-safe paths, and the UAV could enter unsafe areas to give operators a view of the state of the mining face. While the academic contribution of this chapter also includes the research into the autonomy modules that enable such missions, the major outcome can be said to be the field work itself. It is quite rare in the literature for such extensive and realistic evaluations to be performed, especially in the real field environments. The continued pursuit of demonstrating use-case oriented missions, in collaboration with the mines, has massive potential in presenting this novel technology to the industry. And through the adoption of this technology, many safety risks can be reduced in routine inspection tasks in the mining industry, as well as in the monitoring of subterranean areas after an accident for both surveying and for assisting rescue teams in finding survivors.

CHAPTER 6

Conclusions

6.1 Summary of Obtained Results

This section will summarize the results and finding in Chapters 2-5, related to the stated goals and contributions discussed in the introduction to this thesis. The stated vision of this thesis was to "further the application areas of completely autonomous robotic platforms by extending their navigation capabilities: towards avoiding obstacles in their environment both static and dynamic, towards the critical perception-actuation link for reactive navigation, towards exploring and planning dynamic paths through previously unknown areas, and towards the coordination and safety in multi-agent robotic systems."

Towards this overarching goal, the thesis has presented two collision avoidance frameworks, one based on constrained NMPC and the other on Artificial Potential Field concepts. Together, they have demonstrated collision avoidance of dynamic obstacles, safe human-robot interactions, and safe navigation in field environments. Two different directions for the perception link to the avoidance systems are considered, where the NMPC used obstacle detection via 2D Lidar and RGB-D cameras to form set-exclusion constraints, while the APF was centered around using the raw 3D LiDAR pointclouds to form repulsive forces. The APF was briefly evaluated for multi-agent systems, and the NMPC much more conclusively where ten agents demonstrated safe navigation in dense aerial swarms using a distributed trajectory sharing approach. Out of the two, the NMPC demonstrated significant performance in generating smooth proactive avoidance maneuvers in challenging scenarios, while maintaining the safety-guarantees of the platform. At the same time, the APF saw significant use in field applications in the later chapters due to the fail-safe nature of using only raw 3D LiDAR data, and it was used in a variety of other works as a local autonomy stack as well.

On the topic of exploring unknown areas, this thesis has presented three different navigation methods. First, two reactive frameworks are demonstrated in subterranean field conditions. One based on the reactive 3D LiDAR APF, and the other based on depth camera information which we denoted as the Deepest-Point Heading Regulation Technique (DPHR). Reactive exploration ideas dominated the field work in chapter 5 in the context of subterranean search

and rescue and inspection missions in underground mines through the COMPRA framework. The reactive methodology to underground navigation enables very fast deployment, rapid navigation, and most importantly very consistent navigation in the specific environment it was developed for. The third method was more closely aligned with other state-of-the-art methods, ERRT, using a tree-based method for "next best view"-style exploration. ERRT was demonstrated for large-scale simulations in the DARPA SubT Challenge worlds, in complex environments with narrow tunnels, junctions, and voids. The framework was also deployed on real hardware in a subterranean field environment, there combining ERRT with the APF and local autonomy kit developed in chapter 3, to form a complete navigation stack from the controller to the exploration module that was developed completely in the works of this thesis.

All considered frameworks and algorithms have been evaluated on real hardware experiments, which was another large focus of the thesis and its included works. While a large part of the experiments in chapter 2 (except section 2.7.2) were demonstrated on smaller platforms and with assistance frommotion-capture systems, all the works in chapters 3-5 were not only evaluated in hardware experiments, but also in field environments. The robot and the related algorithms operated in complete autonomy, not relying on operator input during run-time, nor external sensing or computation. Instead, the robots are fully self-reliant in their autonomy approach, furthering the stated goal of advancing the navigation capabilities for autonomous robots in field application scenarios.

6.2 Limitations & Future Works

The limitations and future works are best presented chapter by chapter. The following bullets summarize ideas, limitations, and extensions of the presented works:

- Chapter 2 - Nonlinear MPC for Obstacle Avoidance: A clear direction for extending the frameworks is looking at different solvers and different MPC formulations such as the chance-constrained MPC [186] or Tube MPC [187] which could offer advantages in the context of obstacle and state uncertainty. Considering the additions to the baseline framework, a better and more general coupling of perception systems to the obstacle avoidance constraints needs to be considered, for example forming set-exclusion constraints from a local subset of an occupancy map, especially for 3D obstacles. Similarly, towards trajectory prediction, boot-strapping methods [53] or potentially learning methods could generalize trajectory prediction for dynamic obstacle scenarios. Both of these extensions attack the one limiting factor of the presented works: the difficulty in deploying the constraint MPC in real field environment which require reliable and generalized perception solutions.

- Chapter 3 - Navigation based on Fully Reactive Artificial Potential Fields: The concepts displayed in this chapter were, according to the author, very interesting and only investigated on a surface level. The fundamental limitation was that the reactive navigation concepts were never coupled with a higher level mission planner. Two direction are of interest here: 1) combining a variety of fully reactive behaviors into more complex missions

for example an explore-inspect mission [188], where the robot would perform a challenging mission type but only involve reactive components. 2) The combination of reactive behavior to semantic understanding, scene graphs [189], semantic mapping [190] etc. for a more human-like navigation process. We do not plan global exact routes when moving from place to place, instead we use picture-memories and semantics maps of our environment combined with locally avoiding obstacles and traversing terrain. This type of navigation style could have great potential for robotic missions, and is a step in the opposite direction of gridsearch-, graph-, or sampling-based path planning on global occupancy maps.

- Chapter 4 - Combined Exploration-Planning in 3D Environments: The ERRT algorithm has multiple direction for future works. An obvious one is integrating a globalization strategy based on ”remembering” high information goals from previous tree-expansions, coupled with utilizing a backtracking approach to reach those locations for new local exploration, again avoiding the need for global path planning. The general ERRT structure is also very fitting for an extension into multi-agent or multi-modality exploration using robots with different actuation/traversability or equipped with different sensors, which could be reflected in their revenue functions for the sampled goals.

- Chapter 5 - Field Deployment in Subterranean Environments: As this chapter does not evaluate some specific component, it is harder to state direct directions of future works. The multi-modality framework could be extensively expanded in the ability for the robots to collaborate. The prospects of using autonomous robots in mine rescue scenarios is very promising, and further investigations into the ability to fly in smoke, the ability to traverse collapsed tunnels, and sharper field demonstrations with a closer collaboration with mine rescue personnel is desirable. The same for general mine monitoring and inspection, where the technology is closing in on being able to achieve the real needs of the industry. Other industries of interest are forestry and construction, with both similar and different challenges, but where many concepts on increasing safety and efficiency of inspection tasks remain the same. Field robotics is in an incredibly interesting stage at the moment, with endless possibilities, but also with realistic visions for the near future.

REFERENCES

[1] S. S. Mansouri, C. Kanellakis, D. Wuthier, E. Fresk, and G. Nikolakopoulos, "Cooperative aerial coverage path planning for visual inspection of complex infrastructures," *arXiv preprint arXiv:1611.05196*, 2016.

[2] C. Kanellakis, E. Fresk, S. S. Mansouri, D. Kominiak, and G. Nikolakopoulos, "Towards visual inspection of wind turbines: A case of visual data acquisition using autonomous aerial robots," *IEEE Access*, vol. 8, pp. 181 650–181 661, 2020.

[3] K. Máthé and L. Buşoniu, "Vision and control for uavs: A survey of general methods and of inexpensive platforms for infrastructure inspection," *Sensors*, vol. 15, no. 7, pp. 14 887–14 916, 2015.

[4] D. Thakur, G. Loianno, W. Liu, and V. Kumar, "Nuclear environments inspection with micro aerial vehicles: Algorithms and experiments," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 191–200.

[5] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[6] J. K. Zègre-Hemsey, B. Bogle, C. J. Cunningham, K. Snyder, and W. Rosamond, "Delivery of automated external defibrillators (aed) by drones: Implications for emergency cardiac care," *Current cardiovascular risk reports*, vol. 12, no. 11, p. 25, 2018.

[7] S. I. Khan, Z. Qadir, H. S. Munawar, S. R. Nayak, A. K. Budati, K. D. Verma, and D. Prakash, "Uavs path planning architecture for effective medical emergency response in future networks," *Physical Communication*, vol. 47, p. 101337, 2021.

[8] S. Sudhakar, V. Vijayakumar, C. S. Kumar, V. Priya, L. Ravi, and V. Subramaniyaswamy, "Unmanned aerial vehicle (uav) based forest fire detection and monitoring for reducing false alarms in forest-fires," *Computer Communications*, vol. 149, pp. 1–16, 2020.

[9] A. Al-Kaff, Á. Madridano, S. Campos, F. García, D. Martín, and A. de la Escalera, "Emergency support unmanned aerial vehicle for forest fire surveillance," *Electronics*, vol. 9, no. 2, p. 260, 2020.

[10] S. S. Mansouri, C. Kanellakis, D. Kominiak, and G. Nikolakopoulos, "Deploying mavs for autonomous navigation in dark underground mine environments," *Robotics and Autonomous Systems*, vol. 126, p. 103472, 2020.

[11] P. Petráček, V. Krátkỳ, M. Petrlík, T. Báča, R. Kratochvíl, and M. Saska, "Large-scale exploration of cave environments by unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7596–7603, 2021.

[12] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[13] S. Minaeian, J. Liu, and Y.-J. Son, "Vision-based target detection and localization via a team of cooperative uav and ugvs," *IEEE Transactions on systems, man, and cybernetics: systems*, vol. 46, no. 7, pp. 1005–1016, 2015.

[14] N. V. Kumar and C. S. Kumar, "Development of collision free path planning algorithm for warehouse mobile robot," *Procedia computer science*, vol. 133, pp. 456–463, 2018.

[15] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Industrial Robot: An International Journal*, vol. 43, no. 6, pp. 583–587, 2016.

[16] G. Carra, A. Argiolas, A. Bellissima, M. Niccolini, and M. Ragaglia, "Robotics in the construction industry: State of the art and future opportunities," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 35. IAARC Publications, 2018, pp. 1–8.

[17] P. Pradhananga, M. ElZomor, and G. Santi Kasabdji, "Identifying the challenges to adopting robotics in the us construction industry," *Journal of Construction Engineering and Management*, vol. 147, no. 5, p. 05021003, 2021.

[18] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[19] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[20] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot operating system (ROS)*. Springer, 2017, pp. 3–39.

[21] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[22] C. W. Warren, "Fast path planning using modified a* method," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 662–667.

[23] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[24] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[25] P. Sopasakis, E. Fresk, and P. Patrinos, "Open: Code generation for embedded nonconvex optimization," *International Federation of Automatic Control*, 2020.

[26] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[27] D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6411–6418, 2020.

[28] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[29] K. Cai, C. Wang, J. Cheng, C. W. De Silva, and M. Q.-H. Meng, "Mobile robot path planning in dynamic environments: a survey," *arXiv preprint arXiv:2006.14195*, 2020.

[30] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[31] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[32] S. Karlsson, A. Koval, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "D*+: A generic platform-agnostic and risk-aware path planing framework with an expandable grid," *arXiv preprint arXiv:2112.05563*, 2021.

[33] P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-d trajectory planning of aerial vehicles using rrt," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1116–1123, 2016.

[34] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[35] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, and S. Behnke, "Multi-layered mapping and navigation for autonomous micro aerial vehicles," *Journal of Field Robotics*, vol. 33, no. 4, pp. 451–475, 2016.

[36] C. Kanellakis, S. S. Mansouri, G. Georgoulas, and G. Nikolakopoulos, "Towards autonomous surveying of underground mine using mavs," in *International Conference on Robotics in Alpe-Adria Danube Region*. Springer, 2018, pp. 173–180.

[37] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[38] P. Fraga-Lamas, L. Ramos, V. Mondéjar-Guerra, and T. M. Fernández-Caramés, "A review on iot deep learning uav systems for autonomous obstacle detection and collision avoidance," *Remote Sensing*, vol. 11, no. 18, p. 2144, 2019.

[39] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[40] S. Hrabar, "Reactive obstacle avoidance for rotorcraft uavs," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4967–4974.

[41] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5332–5339.

[42] A. Schaub, D. Baumgartner, and D. Burschka, "Reactive obstacle avoidance for highly maneuverable vehicles based on a two-stage optical flow clustering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2137–2152, 2016.

[43] B. Ruf, S. Monka, M. Kollmann, and M. Grinberg, "Real-time on-board obstacle avoidance for uavs based on embedded stereo vision," *arXiv preprint arXiv:1807.06271*, 2018.

[44] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[45] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 4136–4141.

[46] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "An mpc algorithm with combined speed and steering control for obstacle avoidance in autonomous ground vehicles," in *Dynamic Systems and Control Conference*, vol. 57267. American Society of Mechanical Engineers, 2015, p. V003T44A003.

[47] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3556–3563.

[48] Y. Xinyi, Z. Yichen, L. Liang, and O. Linlin, "Dynamic window with virtual goal (dw-vg): A new reactive obstacle avoidance approach based on motion prediction," *Robotica*, vol. 37, no. 8, pp. 1438–1456, 2019.

[49] B. H. Lee, J. D. Jeon, and J. H. Oh, "Velocity obstacle based local collision avoidance for a holonomic elliptic robot," *Autonomous Robots*, vol. 41, no. 6, pp. 1347–1363, 2017.

[50] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1089–1096.

[51] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6057–6063.

[52] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[53] S. X. Wei, A. Dixit, S. Tomar, and J. W. Burdick, "Moving obstacle avoidance: A data-driven risk-aware approach," *IEEE Control Systems Letters*, vol. 7, pp. 289–294, 2022.

[54] X. Zhang, J. Ma, Z. Cheng, S. Huang, S. S. Ge, and T. H. Lee, "Trajectory generation by chance-constrained nonlinear mpc with probabilistic prediction," *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3616–3629, 2020.

[55] I. Batkovic, U. Rosolia, M. Zanon, and P. Falcone, "A robust scenario mpc approach for uncertain multi-modal obstacles," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 947–952, 2020.

[56] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 236–243.

[57] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2682–2688.

[58] A. Budiyanto, A. Cahyadi, T. B. Adji, and O. Wahyunggoro, "UAV obstacle avoidance using potential field under dynamic environment," in *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2015, pp. 187–192.

[59] A. Mondal, L. Behera, S. R. Sahoo, and A. Shukla, "A novel multi-agent formation control law with collision avoidance," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 558–568, 2017.

[60] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4357–4363.

[61] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, 2017.

[62] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 477–483.

[63] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[64] C. K. Verginis and D. V. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *arXiv preprint arXiv:2005.12599*, 2020.

[65] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[66] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[67] P. Sopasakis, E. Fresk, and P. Patrinos, "Optimization Engine," 2019. [Online]. Available: http://doc.optimization-engine.xyz/

[68] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[69] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1523–1528.

[70] B. Hermans, G. Pipeleers, and P. P. Patrinos, "A penalty method for nonlinear programs with set exclusion constraints," *Automatica*, vol. 127, p. 109500, 2021.

[71] E. G. Birgin and J. M. Martínez, *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014.

[72] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2992–2997.

[73] J. Jackson, G. Ellingson, and T. McLain, "ROSflight: A lightweight, inexpensive MAV research and development tool," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 758–762.

[74] B. Hermans, P. Patrinos, and G. Pipeleers, "A penalty method based approach for autonomous navigation using nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 234 – 240, 2018.

[75] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2016.

[76] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 811–817.

[77] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[78] S. S. Mansouri, C. Kanellakis, B. Lindqvist, F. Pourkamali-Anaraki, A.-A. Agha-Mohammadi, J. Burdick, and G. Nikolakopoulos, "A unified nmpc scheme for mavs navigation with 3d collision avoidance under position uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5740–5747, 2020.

[79] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Nonlinear model predictive control for multi-micro aerial vehicle robust collision avoidance," *arXiv preprint arXiv:1703.01164*, 2017.

[80] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[81] M. Przybyła, "Detection and tracking of 2d geometric obstacles from lrf data," in *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2017, pp. 135–141.

[82] T. Fraichard and A. Scheuer, "Car-like robots and moving obstacles," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 64–69.

[83] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[84] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 1149–1154.

[85] S. Karlsson, B. Lindqvist, and G. Nikolakopulos, "Ensuring robot-human safety for the bd spot using active visual tracking and nmpc with velocity obstacles," *IEEE Access*, vol. 10, pp. 100 224–100 233, 2022.

[86] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[87] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[88] S. Haykin, *Kalman filtering and neural networks*. John Wiley & Sons, 2004, vol. 47.

[89] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," 2020.

[90] Y. Bai, B. Lindqvist, S. Karlsson, C. Kanellakis, and G. Nikolakopoulos, "Multi-robot task allocation framework with integrated risk-aware 3d path planning," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 481–486.

[91] N. Dahlquist, B. Lindqvist, A. Saradagi, and G. Nikolakopoulos, "Reactive multi-agent coordination using auction-based task allocation and behavior trees," *arXiv preprint arXiv:2304.01976*, 2023.

[92] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, p. 36, 2020.

[93] P. Z. Peebles, *Probability, random variables, and random signal principles*. McGraw-Hill New York, 2001, vol. 3.

[94] S. S. Mansouri, G. Nikolakopoulos, and T. Gustafsson, "Distributed model predictive control for unmanned aerial vehicles," in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, 2015, pp. 152–161.

[95] W. Hönig and N. Ayanian, *Flying Multiple UAVs Using ROS*. Springer International Publishing, 2017, pp. 83–118.

[96] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[97] B. Lindqvist, P. Sopasakis, and G. Nikolakopoulos, "A scalable distributed collision avoidance scheme for multi-agent uav systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9212–9218.

[98] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," in *IFAC World Congress*, Berlin, 2020, software available at http://doc.optimization-engine.xyz/.

[99] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[100] J. N. Yasin, S. A. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches," *IEEE access*, vol. 8, pp. 105 139–105 155, 2020.

[101] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[102] H. M. Jayaweera and S. Hanoun, "A dynamic artificial potential field (d-apf) uav path planning technique for following ground moving targets," *IEEE Access*, vol. 8, pp. 192 760–192 776, 2020.

[103] Y. Du, X. Zhang, and Z. Nie, "A real-time collision avoidance strategy in dynamic airspace based on dynamic artificial potential field algorithm," *IEEE Access*, vol. 7, pp. 169 469–169 479, 2019.

[104] P. Glotfelter, I. Buckley, and M. Egerstedt, "Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1303–1310, 2019.

[105] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, "Simultaneous trajectory planning and tracking using an mpc method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4273–4283, 2018.

[106] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 2759–2764.

[107] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," in *Experimental Robotics*. Springer, 2016, pp. 195–209.

[108] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, "Lidar-based obstacle avoidance for the autonomous mobile robot," in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 2019, pp. 197–202.

[109] K. van Hecke, G. de Croon, L. van der Maaten, D. Hennes, and D. Izzo, "Persistent self-supervised learning: From stereo to monocular vision for obstacle avoidance," *International Journal of Micro Air Vehicles*, vol. 10, no. 2, pp. 186–206, 2018.

[110] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 179–185.

[111] B. Lindqvist, "Collision avoidance for multiple MAVs using fast centralized NMPC," in *International Federation of Control 2020*, 2020.

[112] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[113] A. Patel, B. Lindqvist, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Ref: A rapid exploration framework for deploying autonomous mavs in unknown environments," *arXiv preprint arXiv:2205.15670*, 2022.

[114] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3105–3112.

[115] C. Kanellakis, S. Sharif Mansouri, M. Castaño, P. Karvelis, D. Kominiak, and G. Nikolakopoulos, "Where to look: a collection of methods formav heading correction in underground tunnels," *IET Image Processing*, vol. 14, no. 10, pp. 2020–2027, 2020.

[116] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Nikolakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[117] M. Faria, A. S. Ferreira, H. Pérez-Leon, I. Maza, and A. Viguria, "Autonomous 3d exploration of large structures using an uav equipped with a 2d lidar," *Sensors*, vol. 19, no. 22, p. 4849, 2019.

[118] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[119] V. K. Viswanathan, S. G. Satpute, and G. Nikolakopoulos, "Flie: First-look enabled inspect-explore autonomy toward visual inspection of unknown distributed and discontinuous structures," *IEEE Access*, vol. 11, pp. 28 140–28 150, 2023.

[120] C. W. Warren, "Global path planning using artificial potential fields," in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 316–317.

[121] B. Lindqvist, S. S. Mansouri, J. Haluška, and G. Nikolakopoulos, "Reactive navigation of an unmanned aerial vehicle with perception-based obstacle avoidance constraints," *IEEE Transactions on Control Systems Technology*, 2021.

[122] DARPA. Subterranean challenge (SubT). [Online]. Available: https://www.subtchallenge.com/

[123] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[124] S. Karlsson, A. Koval, C. Kanellakis, and G. Nikolakopoulos, "D-star-plus: A risk aware platform agnostic heterogeneous path planner," *Expert systems with applications*, p. 119408, 2022.

[125] A. Patel, B. Lindqvist, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Ref: A rapid exploration framework for deploying autonomous mavs in unknown environments," *Journal of Intelligent & Robotic Systems*, vol. 108, no. 3, p. 35, 2023.

[126] A. Patel, B. Lindqvist, C. Kanellakis, and G. Nikolakopoulos, "Fast planner for mav navigation in unknown environments based on adaptive search of safe look-ahead poses," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 545–550.

[127] A. Patel, S. Karlsson, B. Lindqvist, C. Kanellakis, A.-A. Agha-Mohammadi, and G. Nikolakopoulos, "Towards energy efficient autonomous exploration of mars lava tube with a martian coaxial quadrotor," *Advances in Space Research*, vol. 71, no. 9, pp. 3837–3854, 2023.

[128] B. Lindqvist, C. Kanellakis, S. S. Mansouri, A. akbar Agha-mohammadi, and G. Nikolakopoulos, "Compra: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations," 2021.

[129] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluška, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue," *Robotics and Autonomous Systems*, vol. 154, p. 104134, 2022.

[130] V. K. Viswanathan, S. G. Satpute, B. Lindqvist, C. Kanellakis, and G. Nikolakopoulos, "Experimental evaluation of a geometry-aware aerial visual inspection framework in a constrained environment," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 468–474.

[131] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[132] ——, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.

[133] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 476–481.

[134] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[135] F. DuchoĖ, A. Babineca, M. Kajana, P. BeĖoa, M. Floreka, T. Ficoa, and L. Jurišicaa, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[136] S. Karlsson, A. Koval, C. Kanellakis, and G. Nikolakopoulos, "D*+: A risk aware platform agnostic heterogeneous path planner," *Available at SSRN 4137561*.

[137] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 779–786, 2021.

[138] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[139] J. Liu, Y. Lv, Y. Yuan, W. Chi, G. Chen, and L. Sun, "An efficient robot exploration method based on heuristics biased sampling," *IEEE Transactions on Industrial Electronics*, 2022.

[140] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[141] K.-L. Low and A. Lastra, "Efficient constraint evaluation algorithms for hierarchical next-best-view planning," in *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. IEEE, 2006, pp. 830–837.

[142] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[143] T. Dang, F. Mascarich, S. Khattak, H. Nguyen, H. Nguyen, S. Hirsh, R. Reinhart, C. Papachristos, and K. Alexis, "Autonomous search for underground mine rescue using aerial robots," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–8.

[144] T. Dang, F. Mascarich, S. Khattak, H. Nguyen, N. Khedekar, C. Papachristos, and K. Alexis, "Field-hardened robotic autonomy for subterranean exploration," *Field and Service Robotics (FSR)*, 2019.

[145] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[146] J. Qi, H. Yang, and H. Sun, "Mod-rrt*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7244–7251, 2020.

[147] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 595–625, 2016.

[148] M. Dharmadhikari, H. Nguyen, F. Mascarich, N. Khedekar, and K. Alexis, "Autonomous cave exploration using aerial robots," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 942–949.

[149] M. Sarkar, X. Yan, B. A. Erol, I. Raptis, and A. Homaifar, "A novel search and survey technique for unmanned aerial systems in detecting and estimating the area for wildfires," *Robotics and Autonomous Systems*, vol. 145, p. 103848, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889021001330

[150] J. Blank, B. Morrell, A. Bouman, T. Touma, M. Ginting, C. Patterson, and A. Aghamohammadi, "Autonomous mapping and characterization of terrestrial lava caves using quadruped robots: Preparing for a mission to a planetary cave," *LPI Contributions*, vol. 2595, p. 8122, 2021.

[151] T. N. Titus, J. J. Wynne, M. J. Malaska, A.-a. Agha-Mohammadi, P. B. Buhler, E. C. Alexander, J. W. Ashley, A. Azua-Bustos, P. J. Boston, D. L. Buczkowski *et al.*, "A roadmap for planetary caves science and exploration," *Nature Astronomy*, vol. 5, no. 6, pp. 524–525, 2021.

[152] Z.-X. Zhang, "Rock mechanics related to mining engineering," in *ISRM 3rd Nordic Rock Mechanics Symposium-NRMS 2017*. OnePetro, 2017.

[153] L. Zhou, G. Sun, Y. Li, W. Li, and Z. Su, "Point cloud denoising review: from classical to deep learning-based approaches," *Graphical Models*, vol. 121, p. 101140, 2022.

[154] A. Afzalaghaeinaeini, J. Seo, D. Lee, and H. Lee, "Design of dust-filtering algorithms for lidar sensors using intensity and range information in off-road vehicles," *Sensors*, vol. 22, no. 11, p. 4051, 2022.

[155] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, and A.-a. Agha-Mohammadi, "Locus: A multi-sensor

lidar-centric solution for high-precision odometry and 3d mapping in real-time," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 421–428, 2020.

[156] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[157] S. Khattak, H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1024–1029.

[158] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "Plgrim: Hierarchical value learning for large-scale exploration in unknown environments," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 652–662.

[159] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[160] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *International Conference on Modelling and Simulation for Autonomous Systesm*. Springer, Cham, 2019, pp. 274–290.

[161] TEAM COSTAR. NeBula autonomy. [Online]. Available: https://costar.jpl.nasa.gov/

[162] J. Shahmoradi, E. Talebi, P. Roghanchi, and M. Hassanalian, "A comprehensive review of applications of drone technology in the mining industry," *Drones*, vol. 4, no. 3, p. 34, 2020.

[163] L. Dunnington and M. Nakagawa, "Fast and safe gas detection from underground coal fire by drone fly over," *Environmental Pollution*, vol. 229, pp. 139–145, 2017.

[164] R. Turner, N. Bhagwat, L. Galayda, C. Knoll, E. Russell, and M. MacLaughlin, "Geotechnical characterization of underground mine excavations from uav-captured photogrammetric & thermal imagery," in *52nd US Rock Mechanics/Geomechanics Symposium*. OnePetro, 2018.

[165] T. Bamford, K. Esmaeili, and A. P. Schoellig, "Aerial rock fragmentation analysis in low-light condition using uav technology," *arXiv preprint arXiv:1708.06343*, 2017.

[166] G. Freire and R. Cota, "Capture of images in inaccessible areas in an underground mine using an unmanned aerial vehicle," in *UMT 2017: Proceedings of the First International Conference on Underground Mining Technology*. Australian Centre for Geomechanics, 2017.

[167] A. E. Forooshani, S. Bashir, D. G. Michelson, and S. Noghanian, "A survey of wireless communications and propagation modeling in underground mines," *IEEE Communications surveys & tutorials*, vol. 15, no. 4, pp. 1524–1545, 2013.

[168] A. Ranjan, H. Sahu, and H. Sahu, "Communications challenges in underground mines," *Communications*, vol. 5, no. 2, pp. 23–29, 2014.

[169] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2003.

[170] Intel. OpenVINO™ Toolkit. [Online]. Available: https://docs.openvinotoolkit.org/latest/index.html

[171] A. Koval, S. Karlsson, and G. Nikolakopoulos, "Experimental evaluation of autonomous map-based spot navigation in confined environments," *Biomimetic Intelligence and Robotics*, p. 100035, 2022.

[172] V. Krátkỳ, P. Petráček, T. Báča, and M. Saska, "An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments," *Journal of field robotics*, vol. 38, no. 8, pp. 1036–1058, 2021.

[173] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska, "A robust uav system for operations in a constrained environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2169–2176, 2020.

[174] T. Özaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1740–1747, 2017.

[175] P. Rea and E. Ottaviano, "Design and development of an inspection robotic system for indoor applications," *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 143–151, 2018.

[176] A. Kalantari, T. Touma, L. Kim, R. Jitosho, K. Strickland, B. T. Lopez, and A.-A. Agha-Mohammadi, "Drivocopter: A concept hybrid aerial/ground vehicle for long-endurance mobility," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–10.

[177] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[178] illuMINEation Horizon 2020. Bright concepts for a safe & sustainable digital mining future. [Online]. Available: https://www.illumineation-h2020.eu/

[179] N. S. H. 2020. Next generation carbon neutral pilots for smart intelligent mining systems. [Online]. Available: https://www.nexgensims.eu/

[180] LKAB. A new world standard for sustainable mining. [Online]. Available: https://lkab.com/en/what-we-do/our-transformation/a-new-world-standard-for-sustainable-mining/

[181] B. Dynamics. Lkab & ltu - going deeper. [Online]. Available: https://www.bostondynamics.com/resources/case-study/lkab-lulea-university-technology

[182] Amovlab, "Prometheus autonomous uav opensource project," https://github.com/amov-lab/Prometheus.

[183] N. Stathoulopoulos, A. Koval, and G. Nikolakopoulos, "3deg: Data-driven descriptor extraction for global re-localization in subterranean environments," *arXiv preprint arXiv:2210.07285*, 2022.

[184] B. Lindqvist, J. Haluska, C. Kanellakis, and G. Nikolakopoulos, "An adaptive 3d artificial potential field for fail-safe uav navigation," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 362–367.

[185] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[186] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[187] G. Garimella, M. Sheckells, J. L. Moore, and M. Kobilarov, "Robust obstacle avoidance using tube nmpc." in *Robotics: Science and Systems*, 2018.

[188] V. Kottayam Viswanathan, B. Lindqvist, S. G. Satpute, C. Kanellakis, and G. Nikolakopoulos, "Towards visual inspection of distributed and irregular structures: A unified autonomy approach," 2023.

[189] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone, "Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9272–9279.

[190] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

Department of System- och rymdteknik
Division of Signaler och System

ISSN 1402-1544
ISBN 978-91-8048-377-3 (print)
ISBN 978-91-8048-378-0 (pdf)

Luleå University of Technology 2023

LULEÅ
UNIVERSITY
OF TECHNOLOGY