Master's Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Multi-Robot Exploration Using a Heterogeneous UAV Team

Bc. Michaela Cihlářová

Supervisor: Ing. Václav Pritzl
Field of study: Cybernetics and Robotics
January 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Cihlářová Michaela**	Personal ID number:	**483482**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Multi-Robot Exploration Using a Heterogeneous UAV Team

Master’s thesis title in Czech:

Multirobotická explorace týmem heterogenních UAV

Guidelines:

The goal of the thesis is to design, implement, and experimentally verify an algorithm for multi-robot exploration of an environment with obstacles using a heterogeneous team of UAVs considering the specific capabilities of the different UAVs.
The specific tasks to be done are the following:
1) Get familiar with the current state of the art in multi-UAV exploration.
2) Design a multi-robot exploration algorithm for a team of heterogeneous UAVs in an environment with obstacles, taking into account their different capabilities, mainly their different physical sizes.
3) Implement the proposed approach in ROS, building upon the MRS UAV System [1].
4) Analyze the performance of the proposed algorithm in simulations.
5) If the necessary UAV hardware and a suitable environment are avaible, test the proposed approach in real-world experiments.

Bibliography / sources:

[1] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert and Martin Saska. “The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles”. Journal of Intelligent & Robotic Systems 102(26):1–28, May 2021.
[2] Best, Graeme, Rohit Garg, John Keller, Geoffrey A. Hollinger, and Sebastian Scherer. “Resilient Multi-Sensor Exploration of Multifarious Environments with a Team of Aerial Robots.” In Robotics: Science and Systems XVIII. Robotics: Science and Systems Foundation, 2022. https://doi.org/10.15607/RSS.2022.XVIII.004.
[3] Zhou, Boyu, Hao Xu, and Shaojie Shen. “RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System.” IEEE Transactions on Robotics 39, no. 3 (June 2023): 1816–35. https://doi.org/10.1109/TRO.2023.3236945.
[4] Kulkarni, Mihir, Mihir Dharmadhikari, Marco Tranzatto, Samuel Zimmermann, Victor Reijgwart, Paolo De Petris, Huan Nguyen, et al. “Autonomous Teamed Exploration of Subterranean Environments Using Legged and Aerial Robots.” In 2022 International Conference on Robotics and Automation (ICRA), 3306–13, 2022. https://doi.org/10.1109/ICRA46639.2022.9812401.
[5] Matej Petrlik, Pavel Petracek, Vit Kratky, Tomas Musil, Yurii Stasinchuk, Matous Vrba, Tomas Baca, Daniel Hert, Martin Pecka, Tomas Svoboda and Martin Saska. “UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT”. Field Robotics 3:1–68, January 2023.

Name and workplace of master’s thesis supervisor:

Ing. Václav Pritzl   Multi-robot Systems FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **08.09.2023** Deadline for master's thesis submission: **09.01.2024**

Assignment valid until: **16.02.2025**

Ing. Václav Pritzl	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to thank my supervisor Ing. Václav Pritzl, for his guidance during the work and his valuable feedback. Next, I would like to thank my family for their endless support over the course of my life and my studies. And finally, I would like to thank my friends and my partner for keeping my spirits up in those past few years.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

V Praze, 9. January 2024

Abstract

This master's thesis develops a novel method for exploring unknown spaces with a heterogeneous team of Unmanned Aerial Vehicles (UAVs) of different sizes and sensory equipment. It employs frontier-based exploration with two task allocation strategies: a greedy strategy that assigns Points of Interest (POIs) based on Euclidean distance and UAV priority and an optimization strategy that solves a minimum cost flow problem. It utilizes the SphereMap algorithm to assess the accessibility of the POIs and generate paths that account for obstacle distances. It also employs the MRS SubT planning library to implement collision avoidance maneuvers among UAVs. The method was validated through simulation testing and real-world experiments that evaluated the method's performance onboard the UAV. The results demonstrated the method's efficacy and applicability to real-world deployment while indicating potential areas for future improvement.

Keywords: multi-robot exploration, unmanned aerial vehicles, indoor exploration, frontier-based exploration, task allocation, planning

Supervisor: Ing. Václav Pritzl
ČVUT,
Resslova 307/9,
Praha 2

Abstrakt

Tato diplomová práce představuje novou metodu pro exploraci neznámých prostorů pomocí heterogenního týmu bezpilotních letounů (UAVs). Využívá explorace založené na navádění UAV k hranici mezi známým a neznámým prostředím s dvěma strategiemi pro rozdělování cílů: hladovým algoritmem, který vybírá body zájmu (POIs) na základě euklidovské vzdálenosti a priority UAV, a optimalizační strategií, která formuluje problém toku s minimální cenou. K posouzení dostupnosti bodů zájmu a generování tras, které zohledňují vzdálenosti od překážek, je využíván SphereMap algoritmus. K implementaci manévrů na vyhýbání se srážkám mezi UAVs pak slouží plánovací knihovna MRS SubT Planner. Metoda byla ověřena prostřednictvím simulačních testů a reálných experimentů, které zhodnotily výkon navržené metody na palubě UAV. Výsledky demonstrovaly funkčnost metody a odhalily potenciální oblasti pro zdokonalení.

Klíčová slova: multirobotická explorace, bezpilotní helikoptéry, explorace vnitřních prostor, průzkum podél hraničního prostoru, alokace úloh, plánování

Překlad názvu: Multirobotická explorace týmem heterogenních UAV

Contents

Figures

Tables

Chapter 1

Introduction

Robotic exploration is the frontier of scientific discovery and innovation. Environments deemed too dangerous, distant, or inaccessible for humans, such as deep oceans, volcanoes, or caves, can be explored and mapped [1], [2], [3]. Data gathering, experimentation, and the testing of technologies that can advance our knowledge are made possible through the use of robots.

Unmanned Aerial Vehicles (UAVs) have opened up new opportunities for exploration of a priori-unknown environment, surpassing the limitations of traditional ground robotic systems. They have become very popular in the last decade as they can be used for many real-world applications. Unlike ground robots, these aerial vehicles can move fast over different terrains to access many hard-to-reach spaces. However, navigating them through complex environments demands a balance between accurate obstacle perception and the weight and cost constraints imposed on UAVs. The integration of sophisticated sensors to ensure precise perception becomes a challenge when faced with the size, weight, and power limitations inherent in UAVs.

One possible solution is to use multi-UAV teams that can share information and cooperate with each other. By distributing the sensing and computation tasks among multiple UAVs, the individual requirements for each UAV can be reduced, allowing for smaller, lighter, and cheaper platforms. Moreover, multi-UAV teams can increase the robustness and reliability of the exploration mission, as the failure of one UAV can be compensated by the others [4].

Heterogeneous UAV teams consist of UAVs with different types, roles, and capabilities, each equipped with different sensors, actuators, and communication devices. The teams have the advantage of being able to handle diverse challenges by leveraging different capabilities. By combining UAVs with different sensory modes, endurance levels, and maneuvering skills, these teams can better adapt to complex and dynamic environments.

An example of this approach is mapping indoor areas with narrow passages and openings. A team consisting of a bigger UAV with precise sensors and high computing capabilities and one or multiple smaller dependent UAVs capable of fitting through narrow entrances at the cost of the quality of the sensors can be useful in complex environments. The smaller UAVs can be sent to scan places that are hard to reach, while the bigger UAV uses its hardware to create global maps, synchronize the fleet, and guide the exploration. This approach protects the more expensive UAV while the cheaper ones perform riskier tasks.

However, heterogeneity in UAV teams also comes with challenges. Coordinating different agents requires advanced algorithms and communication protocols to ensure smooth collaboration. Furthermore, incorporating various UAV platforms complicates system design, maintenance, and resource allocation. These challenges require careful attention to balance the benefits of heterogeneity with the overall effectiveness of the exploration mission.

This thesis aims to investigate the complexities of multi-UAV exploration, focusing on the interactions within heterogeneous teams. The intentional variation of UAV capabilities within a team adds a new dimension to exploration missions. Understanding the inherent advantages and disadvantages of such teams is essential for evaluating the impact on system efficiency, adaptability, and overall mission success.

1.1 Problem Statement

The primary challenge addressed in this thesis is an exploration of complex environments, such as an indoor office space, through cooperative deployment of multiple UAVs featuring different physical characteristics and sensory capabilities. In this context, two UAVs are considered for exploration: a larger primary Unmanned Aerial Vehicle (pUAV), equipped with high computation power and 3D Light Detection and Ranging (LiDAR) sensor with substantial range and high-precision data acquisition capabilities, and a comparatively smaller secondary Unmanned Aerial Vehicle (sUAV), intentionally designed to navigate through narrow spaces, although limited to a Red-Green-Blue-Depth (RGBD) camera with considerably reduced Field of View (FOV) and lower-quality mapping functionality. The UAVs are capable of mutual communication over a wireless network, and the algorithms are designed to run fully onboard the UAVs with no external computational resources utilized.

The exploration framework builds upon several already existing components which are not part of this work. These include the control algorithms that are responsible for the flight behavior, mapping on both UAVs to generate local maps, and the merging of these maps (conducted on pUAV), localization systems determining UAVs' positions, and the utilization of LiDAR data on the pUAV to establish the relative localization of the sUAV. The next chapter provides a more detailed overview of these algorithms.

The goal is to optimize the collaborative efforts of these two UAVs to explore and map a priori-unknown environment where physical constraints and differing sensory inputs impact their exploration capabilities. This can be divided into a few key components:

1. **Selecting Point of Interest (POI)** - the challenge of selecting relevant points within the environment to guide the UAVs' exploration efforts.
2. **Accessibility Problem** - the consideration of accessibility constraints, particularly indoors with confined spaces, ensuring that the selected points are reachable.
3. **Task Allocation** - the problem of distributing specific exploration tasks among the UAVs efficiently to maximize exploration coverage.
4. **Planning** - path planning strategies for the UAVs to navigate from their current location to the selected points of interest.
5. **Obstacle Avoidance** - the UAVs must safely navigate around static obstacles like walls and furniture, as well as dynamically moving objects (other UAVs).

1.2 State of the art

The exploration by multi-UAV systems has been a subject of significant research, focusing on developing efficient and effective strategies for UAVs to navigate and map unknown environments. In [5], a comprehensive overview of the applications and the next steps in using multiple UAVs is provided, pointing out the advancements and challenges in the field. This section will focus only on different approaches in exploration strategies and tasks using multi-UAV teams.

1.2.1 Exploration Strategies

In this part, a review of the existing methods and techniques for robotic exploration is provided, and their advantages and disadvantages are discussed. The focus is on two main categories of methods: sampling-based and frontier-based. Sampling-based methods use probabilistic models to generate and evaluate candidate viewpoints. In contrast, frontier-based methods work with geometric information to identify and reach the borders between known and unknown regions. We will also examine how various methods can combine or enhance these methods.

Sampling-based exploration methods have been widely used in robotic exploration. The idea is to sample candidate viewpoints and evaluate their information gain to advance the exploration. These approaches are linked with the Next-Best-View (NBV) approach [6] focused on selecting informative views based on the spatial context. In [7], the authors introduced the NBV idea in 3D space by using Rapidly Exploring Random Tree (RRT) and selecting the most informative branch. Subsequently, the research [8] improves the approach by implementing RRT*. Furthermore, the rising popularity of machine learning inspired [9], which proposes to learn an informed distribution of views based on the spatial context.

Identifying frontiers, the boundaries between known and unknown areas, is a common strategy. Robots aim to move toward these frontiers to gather information about unexplored regions. This approach was initially proposed in [10] for single-robot exploration and a year later in [11] for multi-robot scenarios. Even though this method depends on accurate mapping to detect the boundaries, it provides easy identification of unexplored areas and efficient information gathering. Popular approaches to searching for frontiers are the Wavefront Frontier Detector (WFD), a graph-search-based algorithm, and the Fast Frontier Detector (FFD), which process only the newly acquired laser data [12]. The main advantage of these algorithms is that they do not process the entire map data. In [13], both of these methods are compared and innovations are proposed to WFD that take advantage of the properties of FFD.

Another challenge of the frontier detection-based algorithm is selecting the goal from the set of frontiers that is the best for the mission objective. Common approaches include clustering frontiers based on distance and selecting relevant points from these clusters. In [14], K-means clustering is used to represent the found borders by only a few points, and the goal is selected from these points based on Euclidean distance and number of frontiers in the cluster. The authors of [15] have proposed a frontier-selection method designed specifically for high-speed flight. For that, the algorithm prioritizes POI in the UAV's FOV. In [16] and [17], computing information gain was proposed to evaluate the possible goals.

The combination of sampling-based and frontier-based approaches aims to reduce computational complexity by targeted sampling around the detected frontiers [18], [19], [20].

1.2.2 Task Allocation Strategies

A key challenge in multi-robot exploration is to assign tasks effectively so that each robot can explore different areas without interference. A common solution is using a central server that communicates with all robots and distributes tasks. Thanks to its low computational complexity, one of the basic methods popular to this day is the greedy approach [21], which selects goals for UAVs based on distance and information gain. However, this approach does not ensure effective coordination of the robot system. The segmentation method [22] addresses this issue by sending robots to different parts of the explored area using a Voronoi graph. Advanced methods like reinforcement learning can also help the system's coordination [23].

Task allocation can be seen as a combinatorial optimization problem. The authors of [24] propose multiple Traveling Salesman Problem (TSP) formulation for a predefined set of goals. For unknown space exploration, the goals are not usually known beforehand. Other methods from the combinatorial optimization field, such as the Hungarian method [25] or reduction to a flow problem [26], are used to solve this situation. These approaches depend on a central unit and reliable communication, which may cause problems when exploring large areas of complex environments where communication can be obstructed.

In contrast, decentralized multi-robot systems are composed of robots that can coordinate their actions without relying on a central controller or a global communication network. This makes them more robust and scalable than centralized systems, especially in dynamic and uncertain environments. The concept was introduced in [27], where robots share map information and navigate toward the closest frontiers. This, however, can result in multiple robots moving to the same area. Also, it heavily relies on communication. Various methods have been proposed to address these challenges, such as auction-based mechanisms [28], potential fields [29], and a decentralized Monte Carlo tree search method [30]. These methods aim to achieve a high level of coordination among robots while minimizing the communication overhead and the computational complexity.

1.2.3 Multi-robot Systems

The research of systems for exploration using legged, wheeled, and flying robots is supported by competitions, such as Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge (SubT Challenge) [3], [31], [32], [33], [34]. These papers focus on various multi-robot systems and their application in underground environments with very limited communication. In [31], an approach of a centralized multi-robot system that allows long-term autonomy of the individual robots is proposed. The research is centered around using various legged and flying robots, whereas in [34], the authors focus on the advantages of the use of both range and vision-sensing modalities.

Communication with a central controller can be problematic in complex environments. The research [35] proposes an exploration approach for a decentralized multi-UAV fleet. With space decomposition, it ensures simultaneous exploration of distinct regions, using only limited and asynchronous communication. Unreliable communication and limited battery life are the motivations in [4]. This work also proposes a decentralized approach but introduces the possibility of sacrificing part of the team for information gain.

This thesis focuses on a centralized heterogeneous multi-UAV team with different sizes and sensor quality. The decision-making and planning processes are executed on board one of the UAV with no available external computational device.

To the best of the author's knowledge, the implementation of a cooperative exploration approach that delegates motion planning to another UAV has not occurred within a UAV-only team, apart from previous work done by the Multi-robot Systems group (MRS group) [36] and [37]. Deploying such a strategy onboard UAVs poses challenges due to size, weight, and power limitations. However, it offers many advantages, such as distributing multiple sensors throughout the system and exploring diverse environments unsuitable for ground robots and where communication with external computational resources is unavailable.

1.3 Outline

Motivation for this thesis is described in Chapter 1, along with problem definition and overview of related work.

Chapter 2 describes the used multi-UAV system. Section 2.1 goes over the hardware specification of both UAVs. The software pipeline, used in simulation and real-life experiments, is outlined in Section 2.2.

In Chapter 3, the approach for unknown space exploration using the described system is proposed. Section 3.1 describes a frontier-based exploration strategy, including selecting POI. The solution for the accessibility problem, using a SphereMap representation, is explained in Section 3.1.2. Section 3.1.3 provides information about two implemented task allocation strategies. Path planning using various planners is described in Section 3.2, and Section 3.3 proposes a collision avoidance algorithm.

Chapter 4 contains verification and tests of the algorithms' performance in simulations, including indoor office space and a 3D model of a real-world warehouse. Section 4.2 provides data from real-world experiments executed in the warehouse.

Finally, Chapter 5 provides a summary of the algorithms proposed in this thesis and the results.

1.4 Mathematical Notation

Vectors are denoted with bold lowercase letters, matrices with bold uppercase italic letters, and frames of reference with uppercase upright letters. Sets and sequences are detonated by uppercase calligraphic letters. The transformation matrix describing the transition from frame A to frame B is represented as ${}^{\mathrm{B}}_{\mathrm{A}}\boldsymbol{T} \in SE(3)$. Let ${}^{\mathrm{A}}\boldsymbol{x} \in \mathbb{R}^3$ be a 3D position vector in frame A, and let ${}^{\mathrm{A}}\mathcal{P}_B$ be a sequence of UAV reference poses $({}^{\mathrm{A}}\boldsymbol{x}_i, {}^{\mathrm{A}}\phi_i)$, with position ${}^{\mathrm{A}}\boldsymbol{x}_i \in \mathbb{R}^3$ and heading/yaw orientation ${}^{\mathrm{A}}\phi_i \in [-\pi, \pi]$, for UAV B in reference frame A.

Chapter 2

Description of the System

This chapter provides an overview of the hardware and software used for the UAVs. It covers technological elements and setups essential for this work.

2.1 Hardware

This part goes through the main hardware specifications for pUAV and sUAV visualized in Figures 2.1a and 2.1b. The detailed description of the UAV platform is available in [38] and [39].

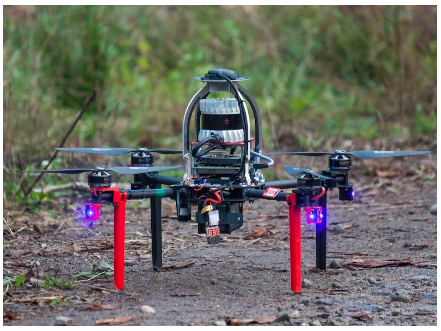
The low-level control of both UAVs is performed by the Pixhawk 4 Flight Controller, which includes a built-in Inertial Measurement Unit (IMU) with an accelerometer, a gyroscope, a magnetometer, and a barometer.

2.1.1 Primary UAV

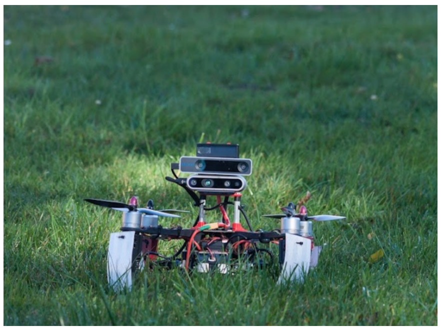
The primary UAV employed in this research project is built upon the Holybro X500 frame, offering structural integrity and support for the onboard hardware. With dimensions of 0.7 by 0.7 meters, including the propellers, it can still safely navigate through larger indoor spaces, such as open-space offices or warehouses. Its computational core features the Intel NUC 10iFNH onboard computer with the processing power of the Intel i7-10710U CPU with six cores and 16 GB of RAM. This computer is also equipped with a Wi-Fi module vital for data exchange.

The sensory capabilities are primarily provided by the Ouster OS0-128 Rev D 3D LiDAR. This sensor offers 360° horizontal and 90° vertical FOV, and with 1024x128 resolution and a maximum range of 50 meters, it enables precise data capture, detailed mapping, and obstacle detection. Operating at a rate of 10 Hz, this LiDAR provides a consistent and real-time data stream. Exact specifications of the LiDAR can be found at [1].

[1] `https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p4-os0.pdf` (accessed November 2023)



(a) : Primary UAV



(b) : Secondary UAV

Figure 2.1: Used UAV platforms

2.1.2 Secondary UAV

The smaller UAV is constructed around the DJI F330 frame, providing a lightweight structure. It has compact physical dimensions, including the propellers, measuring approximately 0.45 by 0.45 meters. This design ensures that the UAV is agile enough to easily navigate through standard doorways and confined indoor spaces. It is equipped with an Intel Core i7-10710U CPU, a capable processor with six cores, complemented by 16 GB of RAM. This hardware configuration provides adequate computing resources for real-time data processing.

Additionally, the UAV is equipped with a tracking camera, the RealSense T265[2], and a depth camera, the RealSense D435[3]. The RealSense T265 compact camera includes two fisheye lens sensors with 173° diagonal FOV, an IMU, and an Intel Movidius Myriad 2 VPU. It contains the sensors needed to track sUAV's location and orientation by Visual Inertial Odometry (VIO) algorithms. The RealSense D435 is a stereo-depth camera that uses stereo vision to calculate depth. It includes a pair of infrared (IR) sensors with $87° \times 58°$ FOV, a Red-Green-Blue (RGB) sensor with $69° \times 42°$ FOV, and an IR projector.

2.2 Software

This section provides an overview of a multi-UAV software pipeline, shown in Figure 2.2 as a high-level diagram.

The software operating on both UAVs is designed to offer a robust and adaptable framework for achieving the mission objective. The software stack is based on the Ubuntu 20.04 operating system with Robot Operating

[2]`https://www.intelrealsense.com/wp-content/uploads/2019/09/Intel_RealSense_Tracking_Camera_Datasheet_Rev004_release.pdf?_ga=2.242768065.257479683.1698752765-1326342669.1698335160` (accessed November 2023)

[3]`https://www.intelrealsense.com/depth-camera-d435/` (accessed November 2023)

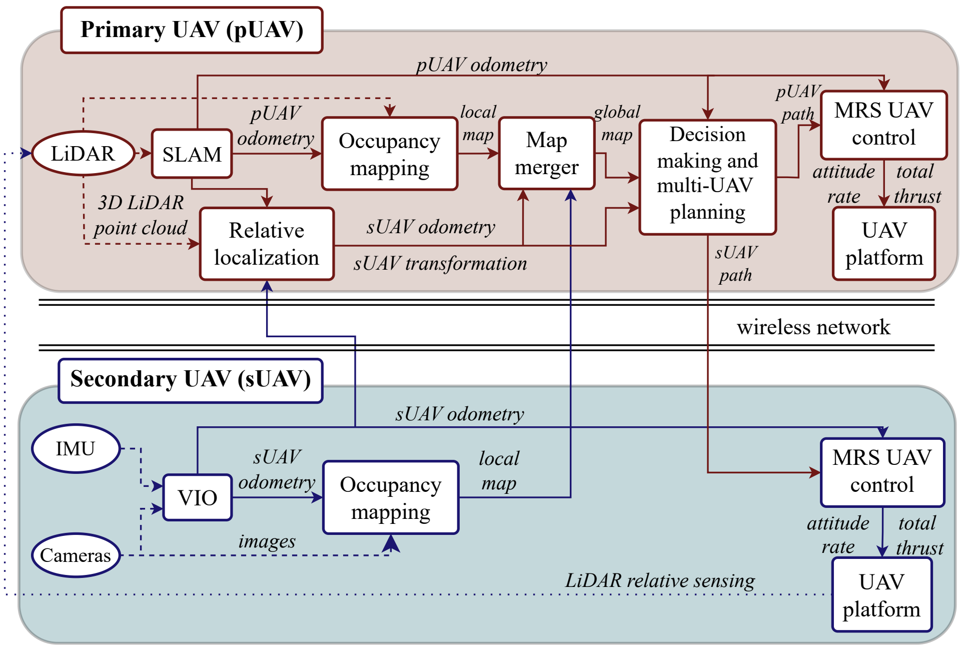


Figure 2.2: Diagram of the multi-UAV software pipeline

System (ROS) 1 Noetic[4] as a middleware for development. The UAVs run on Multi-robot Systems Group UAV system (MRS UAV system)[5], which can be effectively employed onboard the UAVs, and the system time of the onboard PCs of the UAVs is synchronized over the wireless network.

One of the main attributes of the MRS UAV system is the easy transition between simulation and real-world environments, enabling both safe experimentation and reliable real-world deployment. The system also integrates many functionalities, including control mechanisms, pose estimation, mapping, and planning. Detailed information can be found in [40].

Before deployment on the real UAVs, the Gazebo robotic simulator[6] was used. As an open-source 3D simulator, it enables testing and optimizing exploration strategies, sensor configurations, and algorithms in a controlled, risk-free environment.

2.2.1 Control Pipeline

The system shown in Figure 2.3 represents the control pipeline executed on board the UAVs. It was developed by the MRS group and is described in [40], [41] and [42] in detail. This work provides only a short overview of the used components.

The trajectory generation algorithm is based on polynomial trajectory planning described in [43] and [44], with some changes to fit the MRS UAV system.

[4]`http://wiki.ros.org/noetic` (accessed November 2023)
[5]`https://github.com/ctu-mrs/mrs_uav_system` (accessed November 2023)
[6]`https://gazebosim.org/home` (accessed November 2023)

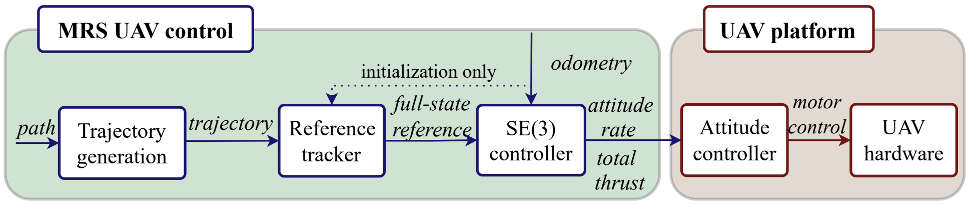


Figure 2.3: Diagram of the control pipeline

It takes the desired path defined by Cartesian coordinates $\boldsymbol{r} = [x, y, z]^\mathrm{T}$ and returns a time-parametrized trajectory that satisfies the existing dynamical constraints of the UAVs and minimizes the path's completion time.

Using the Model predictive control (MPC)-based tracker as the reference tracker, originally published in [42], the system takes in a desired trajectory, comprised of a series of waypoints with a fixed time step Δt. Each point is defined by its Cartesian coordinates $\boldsymbol{r}_D = [x_D, y_D, z_D]^\mathrm{T}$ and the preferred UAV yaw orientation ϕ_D. The tracker produces a smooth and feasible trajectory, maintaining a consistent sampling step. This trajectory includes velocities and accelerations at each point, considering the UAV's translational dynamics and the constraints imposed on both inputs and states.

The SE(3) controller, as described in [45], operates as a nonlinear state feedback controller. It takes in a trajectory defined by specific positions $\boldsymbol{x}_D$, yaw angles ϕ_D, linear velocities $\dot{\boldsymbol{x}}_D$, angular velocities $\dot{\phi}_D$, linear accelerations $\ddot{\boldsymbol{x}}_D$, and angular accelerations $\ddot{\phi}_D$, along with the current UAV state. The output comprises the desired UAV attitude and total thrust, subsequently transmitted to the attitude controller.

The Pixhawk 4 Flight Control unit is equipped with an embedded attitude controller that takes in the desired UAV orientation $\boldsymbol{R}$ and desired total thrust T_D. The controller then generates commands for the electronic speed controllers controlling the motors to gain the specified attitude.

The UAV hardware block represents configurations specified in Section 2.1.

2.2.2 Primary UAV

The pUAV's software pipeline, visualized in Figure 2.2, starts by odometry computation and self-localization using Simultaneous Localization And Mapping (SLAM) algorithm LiDAR Odometry and Mapping (LOAM) [46], specifically Advanced implementation of LOAM (A-LOAM)[7]. It takes LiDAR point cloud data and returns position ${}^\mathrm{P}\boldsymbol{x}_P$ and orientation ${}^\mathrm{P}\phi_P$ in pUAV SLAM frame P.

The pUAV's odometry, and LiDAR data are then processed by the MRS Octomap Server[8], which, using the OctoMap [47] approach, produces a local

[7]`https://github.com/ctu-mrs/aloam` (accessed December 2023)
[8]`https://github.com/ctu-mrs/mrs_octomap_server` (accessed December 2023)

occupancy map ${}^{\mathrm{P}}\mathcal{M}_P$. To optimize computational efficiency, the map ${}^{\mathrm{P}}\mathcal{M}_P$ focuses on a constrained region of the environment, centered at the current position of the pUAV.

The relative localization provides accurate transformation ${}^{\mathrm{S}}_{\mathrm{P}}\boldsymbol{T}$ from the frame P to the sUAV VIO frame S. It is composed of multiple actions. First, the sUAV is identified within LiDAR point cloud data in frame P, based on prior work on autonomous interception of intruder UAVs [48], [49]. Then, the LiDAR-based detections of the sUAV ${}^{\mathrm{P}}\boldsymbol{x}_S$ are fused with the sUAV VIO pose $({}^{\mathrm{S}}\boldsymbol{x}_S, {}^{\mathrm{S}}\phi_S)$ received over a wireless network [36]. Such an approach allows precise guidance of sUAV by more capable pUAV [37].

The map merger algorithm creates a global occupancy map ${}^{\mathrm{G}}\mathcal{M}$ in a common global frame G, using the transformations ${}^{\mathrm{S}}_{\mathrm{P}}\boldsymbol{T}$, ${}^{\mathrm{G}}_{\mathrm{P}}\boldsymbol{T}$, local occupancy map ${}^{\mathrm{P}}\mathcal{M}_P$ produced by pUAV's mapping algorithm, and local occupancy map ${}^{\mathrm{S}}\mathcal{M}_S$, received over a wireless network from sUAV. This map keeps all the information throughout the exploration and is used for high-level planning.

The high-level planning and decision-making algorithms, the main objects of this thesis, work in the global frame G. Without loss of generality, we set the global frame G to be equivalent to the pUAV SLAM frame P. This means that transformation ${}^{\mathrm{G}}_{\mathrm{P}}\boldsymbol{T}$ is an identity matrix, and $({}^{\mathrm{P}}\boldsymbol{x}_P, {}^{\mathrm{P}}\phi_P) = ({}^{\mathrm{G}}\boldsymbol{x}_P, {}^{\mathrm{G}}\phi_P)$ for the specific case of a single LiDAR-equipped pUAV cooperating with a single camera-equipped sUAV. For better clarity, the superscript G, denoting the global frame, is omitted in the description of the algorithms unless its usage is necessary to prevent ambiguity.

2.2.3 Secondary UAV

Mapping and localization of the sUAV is executed by processing data from the two RealSense sensors. The OpenVINS algorithm [50] offers a state-of-the-art, filter-based, VIO. It takes the camera images and IMU data provided by the RealSense T265 and computes pose in sUAV VIO frame S.

The RealSense D435 provides depth information, which is, with odometry from OpenVINS, sent to the occupancy mapping module, which creates local occupancy map ${}^{\mathrm{S}}\mathcal{M}_S$.

The map ${}^{\mathrm{S}}\mathcal{M}_S$, and pose $({}^{\mathrm{S}}\boldsymbol{x}_S, {}^{\mathrm{S}}\phi_S)$ are sent to the pUAV. The sUAV receives path ${}^{\mathrm{S}}\mathcal{P}_S$ for the control pipeline.

Chapter 3

Multi-UAV Exploration Framework

The exploration algorithm runs on board the pUAV and coordinates the behavior of the entire UAV team. The algorithm itself consists of the following submodules, each running in parallel in a separate thread:

1. **Selecting POIs** - The first thread focuses on identifying POIs. A frontier detection algorithm identifies unexplored regions. The system also assigns goals to individual UAVs, based on the goal's accessibility and distance, optimizing the distribution of exploration efforts across the fleet.

2. **Path planning** - The second thread handles the planning of paths to already selected POIs. It identifies UAVs completing their current task and requiring a new path. This ensures effective and continuous exploration of the unknown space.

3. **Collision avoidance** - The third thread addresses the crucial aspect of collision avoidance among UAVs. This algorithm continuously monitors the spatial dynamics of the two UAVs and employs safety measures if needed.

The implementation is written in C++, using the ROS. It offers a configuration file to allow prompt adjustments to environmental conditions and mission requirements. Through this, operators can fine-tune numerous parameters without program recompilation. It enables, for example, to specify UAVs' dimensions, define the exploration area, or set minimal safety distance between the UAVs.

It is assumed that all poses $(\boldsymbol{x}, \phi)$, paths $\mathcal{P}$, and other variables dependent on a specific frame without a prefix are in global frame G, for example, $\mathcal{P} = {}^{\mathrm{G}}\mathcal{P}$.

3.1 Selecting Points of Interest (POIs)

There are multiple ways to approach the exploration of unknown space. This work focuses on frontier-based exploration but implements various task allocation methods. They were selected from the state-of-the-art approaches with respect to time complexity and efficiency. Low computational cost is critical so the software can easily run onboard UAVs.

3.1.1 Frontier Detection

As mentioned in section 2.2.2, the explored space is represented by an OctoMap, enabling simple detection of frontiers. The nodes without any children, called leaf nodes, are the only candidates for frontiers. In Algorithm 1, let $\mathcal{L}$ represent the set of all the leaf nodes l from global OctoMap $\mathcal{M}$. The map $\mathcal{M}$ can be divided into two subsets: the set of free nodes $\mathcal{M}_{\text{free}}$ and the set of occupied nodes $\mathcal{M}_{\text{occ}}$, therefore

$$\mathcal{M} = \mathcal{M}_{\text{free}} \cup \mathcal{M}_{\text{occ}} \tag{3.1}$$

Let $\mathcal{M}_{\text{unk}}$ represent nodes whose state is currently unknown and are not part of the map $\mathbf{M}$.

The exploration can be limited to a particular area by creating a virtual border. It is defined in the configuration file and represents the maximum distance of UAVs from pUAV's initial position for each axis separately. The set $\mathcal{B}$ contains all the (x, y, z) coordinates that are inside the exploration area. If no border is defined, $\mathcal{B}$ is equal to the whole 3D space.

A leaf node l is declared frontier if it is not occupied, is inside B, is not inside a set of already visited points $\mathcal{V}$, and has at least one unknown neighbor. The frontiers are divided into clusters f based on their mutual Euclidean distance, meaning that frontiers close to each other are put in the same group to reduce computational requirements. The set of all the frontier clusters is defined by $\mathcal{F}$. Finally, the POIs are selected from these clusters f by computing the centroid $\boldsymbol{m}$ of individual groups and finding the closest frontier voxel $\boldsymbol{v} \in f$ to the cluster's centroid m.

In the case of a very complex environment, it is beneficial to draw random samples from the clusters to add POIs. Number of samples from each group depends on its size. The number of samples is calculated as

$$k = \alpha \cdot \textsc{length}(f) \tag{3.2}$$

where α is a parameter defined by the user.

3.1.2 Accessibility Problem

The accessibility problem in multi-robot exploration of unknown spaces is a crucial aspect of ensuring the safety of drone operations. In complex 3D envi-

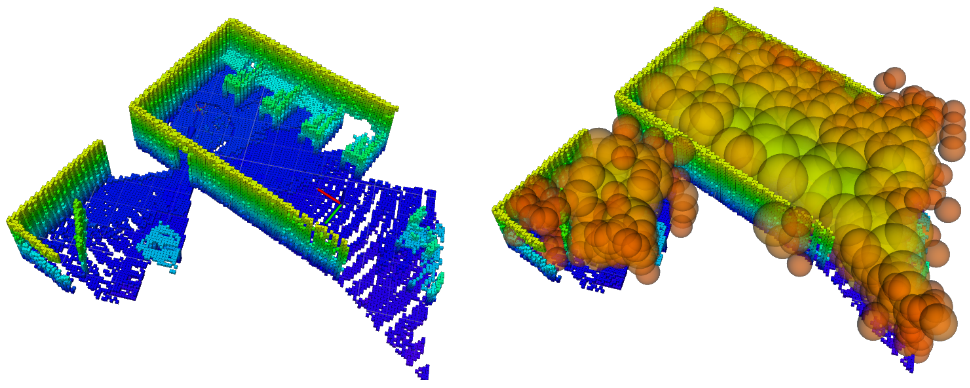
Algorithm 1: Selecting POIs

```
Input: set of leaf nodes $\mathcal{L}$, visited points $\mathcal{V}$, set of free map nodes
       $\mathcal{M}_{\text{free}}$, unknown space $\mathcal{M}_{\text{ukn}}$
Output: set of POIs $\mathcal{G}$
Parameters: exploration area $\mathcal{B}$, maximal Euclidean distance from
            cluster $d_f$
1  $\mathcal{F} \leftarrow \emptyset$                                  ▷ set of frontier clusters
2  for $\forall l \in \mathcal{L}$ do
3      if $l \in \mathcal{M}_{\text{free}}$ and $l \in \mathcal{B}$ and $l \notin \mathcal{V}$ then
4          $\mathcal{N} \leftarrow l$.NEIGHBORS
5          if $\mathcal{N} \cap \mathcal{M}_{\text{unk}} \neq \emptyset$ then
6              $d \leftarrow \min_{f \in \mathcal{F}} \|l - f\|_2$          ▷ from the closest point in the
                cluster $f$
7              $f_{\text{old}} \leftarrow \arg\min_{f \in \mathcal{F}} \|l - f\|_2$
8              if $d \leq d_f$ then
9                  $f_{\text{old}} \leftarrow f_{\text{old}} \cup \{l\}$
10             else
11                 $f_{\text{new}} \leftarrow \{l\}$
12                 $\mathcal{F} \leftarrow \mathcal{F} \cup f_{\text{new}}$
13             end
14         end
15     end
16 end
17 for $\forall f \in \mathcal{F}$ do
18     $n \leftarrow$ LENGTH$(f)$
19     $\boldsymbol{m} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_i$; $\boldsymbol{v} \in f$
20     $\boldsymbol{g} \leftarrow \arg\min_{\boldsymbol{v}_i \in f} \|\boldsymbol{m} - \boldsymbol{v}_i\|_2$
21     $\mathcal{G} \leftarrow \mathcal{G} \cup \{\boldsymbol{g}\}$
22 end
```

ronments, where the map is large, and obstacles are numerous, determining the accessibility of POIs poses a significant computational challenge. The method of obstacle inflation, commonly used in 2D maps, is not computationally efficient for 3D representation.

To address the challenge, this work proposes an approach that utilizes the SphereMap, described in [51], a technique designed to simplify the representation of 3D space. It fills free space using intersecting spheres with a predefined minimal radius and creates a graph connecting the centers of the intersecting spheres. An example of such a representation of free space can be seen in Figure 3.1. SphereMap effectively reduces the computational complexity of path planning and obstacle avoidance. It was developed by the MRS group and successfully employed for the DARPA SubT Challenge, demonstrating its effectiveness in single UAV exploration.

Even though the challenge in this thesis is more complex, not just by



(a) : OctoMap **(b) :** SphereMap with OctoMap

Figure 3.1: Map representations

working with multi-UAV system but also by having UAVs with different sizes, it is possible to use SphereMap. It only enforces the predefined distance when necessary, maintaining a safer distance otherwise, which means that safety is the main priority rather than the path's speed. Moreover, thanks to the spheres, detecting the closest obstacle to every point in the path is simple. So, it is possible to set the minimal sphere radius according to the smallest UAV's size, ensuring that areas that are not accessible by any UAV are excluded and check the obstacle distance for the rest.

To determine the accessibility of a POI, a path to the POI is planned using the SphereMap representation. If the path exists and the distance to the closest obstacle remains greater than the UAV's radius, the point is marked as accessible. This approach ensures that the path to the POI can be safely navigated by the UAV without colliding with any obstacles.

The SphereMap-based accessibility check is computationally efficient enough to operate in real time, even for a large number of POIs. However, in practice, the aim is to minimize the number of planned paths to reduce computational overhead. This is achieved by identifying the most interesting POIs by considering other criteria, as described in the following section.

3.1.3 Multi-Robot Task Allocation (MRTA)

Task allocation presents a complicated challenge in multi-robot systems, and its definition varies depending on the application - coverage, surveillance, distribution, etc. The problem, defined as NP-hard, becomes even more challenging in larger environments containing many robots and tasks. This work focuses on effectively exploring unknown spaces, minimizing traveled distance in a confined environment using only a few UAVs.

Exploration can be seen as a matching problem, one of the combinatorial optimization tasks, as it is possible to assign only one goal to one UAV at a time. A common approach to address this challenge is to represent the

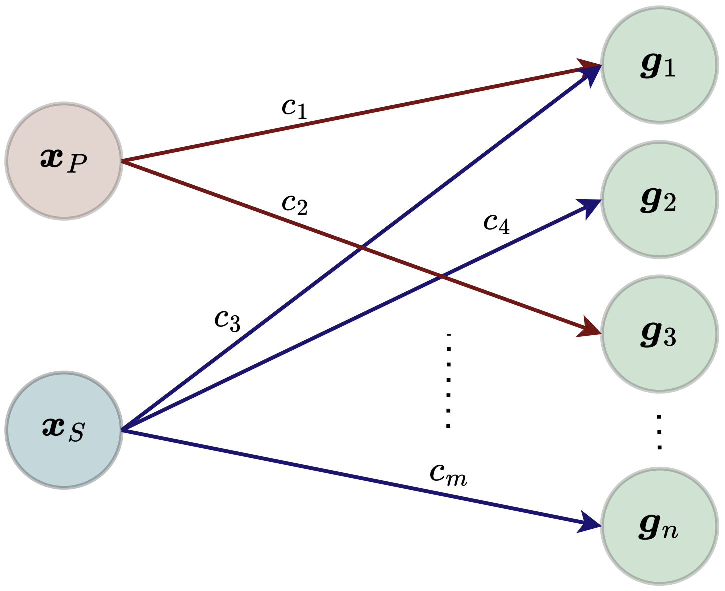


Figure 3.2: Task allocation represented as a bipartite graph

UAVs and the POIs as a bipartite graph $G = (U, V, E)$, where $\boldsymbol{x}_P, \boldsymbol{x}_S \in U$ are nodes represented by UAVs' position, $\boldsymbol{g}_i \in V$ is a node which stands for corresponding POI i, and $(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{c}) \in E$ are edges defined by three values: starting node, end node, and cost c (see Figure 3.2. For example, arc from pUAV node to POI $\boldsymbol{g}_1$ with cost c_1 would be described as $(\boldsymbol{x}_P, \boldsymbol{g}_1, c_1)$.

An edge between a UAV and a POI indicates the feasibility of the UAV visiting that specific point. The minimum weight matching problem can be employed to find an optimal task assignment within this bipartite graph representation. This identifies the UAV-POI pairings that minimize a defined cost function, such as the total distance traveled by the UAVs or the overall completion time of the tasks.

This formulation is referenced as an assignment problem in the following sections. From the definition, the assignment problem is a minimum weight perfect matching, meaning there should be the same number of UAVs and POIs. However, from the nature of unknown space exploration, the number of POIs is usually different than the number of UAVs, but the goal is the same: to assign some task (POI) to every worker (UAV). The only change is that the final solution can leave some unmatched POIs.

The primary constraint in choosing algorithms for this purpose is the necessity of real-time execution onboard one of the UAVs in a dynamic environment. This means that the goals for each UAV cannot be selected before the flight, but the assignment problem needs to be solved multiple times during the exploration (with every update of the global map). This work offers two approaches: a simple greedy algorithm based on Euclidean distance and a more complex algorithm that reduces the matching problem to a flow problem.

Greedy Approach

To verify the functionality of the system and provide a baseline for comparison, this work offers a greedy approach based on Euclidean distance d and heading θ. The heading computation is necessary for sUAV due to the depth camera's limited horizontal FOV. This creates the need to change sUAV's heading during the flight to explore new areas, which is time-consuming. To minimize the time required to change the heading of the sUAV, the greedy approach prefers points located in the same direction as the sUAV is facing. For pUAV, the heading change is not necessary, thanks to the 360° horizontal FOV, and due to the low update frequency of the global map. Therefore, the greedy approach selects the point with the lowest Euclidean distance for the pUAV. The costs c_{i_P} and c_{i_S} are computed for each POI $\boldsymbol{g}_i \in \mathcal{G}$ as

$$c_{i_P} = \|\boldsymbol{x}_P - \boldsymbol{g}_i\|_2 \tag{3.3}$$

$$c_{i_S} = \alpha\|\boldsymbol{x}_S - \boldsymbol{g}_i\|_2 + \beta|\phi_S - \theta_i| \tag{3.4}$$

where $\alpha, \beta \in \mathbb{R}$ are predefined parameters.

The UAVs are assigned priority in goal selection to consider the different UAV sizes. This means that the pUAV selects its destination first as it fits only in spacious areas. The Algorithm 2 follows a step-wise approach. First, it computes the Euclidean distance for each POI $\boldsymbol{g}_i \in \mathcal{G}$ for the pUAV and determines the distance and heading change for the sUAV. These values establish minimum priority queues $\mathcal{Q}_\mathcal{P}$ and $\mathcal{Q}_\mathcal{S}$. The algorithm then identifies the closest accessible point $\boldsymbol{g}_{P_n}$ for the pUAV by planning on SphereMap, as described in Section 3.1.2. Subsequently, it finds the first accessible point $\boldsymbol{g}_{S_n}$ for the sUAV using the same method, excluding the goal $\boldsymbol{g}_{P_n}$ chosen for the pUAV. If no accessible point is found, the UAV stays at its current position.

Algorithm 2: Task allocation - greedy approach

```
Input: pUAV pose ($\boldsymbol{x}_P, \phi_P$), sUAV pose ($\boldsymbol{x}_S, \phi_S$), the set of POIs $\mathcal{G}$
Output: pUAV next goal $\boldsymbol{g}_{P_n}$, sUAV next goal $\boldsymbol{g}_{S_n}$
Parameters: pUAV size $s_P$, sUAV size $s_S$
1 $\mathcal{Q}_P \leftarrow$ PRIORITY_QUEUE($\mathcal{G}, \boldsymbol{x}_P$)  ▷ based on distance from pUAV
2 $\mathcal{Q}_S \leftarrow$ PRIORITY_QUEUE($\mathcal{G}, \boldsymbol{x}_S, \phi_S$) ▷ based on distance and heading
    change from sUAV
3 $\boldsymbol{g}_{P_n} \leftarrow$ FIND_FIRST_ACCESSIBLE($\mathcal{Q}_P, \boldsymbol{x}_P, s_P$)
4 $\boldsymbol{g}_{S_n} \leftarrow$ FIND_FIRST_ACCESSIBLE($\mathcal{Q}_S \setminus \{\boldsymbol{g}_{P_n}\}, \boldsymbol{x}_S, s_S$)
```

The algorithm's advantages include speed, low computational complexity, simplicity, and memory efficiency. However, its drawbacks lie in treating the UAVs independently rather than as a unified system. Furthermore, it does not consider path length, which can lead to high travel time despite proximity to the selected point.

Minimum-cost flow (MCF) Algorithm

Thanks to the graph properties, the assignment problem can be reduced to a flow problem: a network optimization problem that involves finding the maximum flow from a source node to a sink node, subject to capacity constraints on the edges, and subsequently, be solved by an algorithm for Minimum-cost flow (MCF) problem, which has polynomial time complexity on a bipartite graph. To represent the assignment problem as a flow problem, the following terms need to be defined:

- **Source Node** S: The source node is a single node that represents all the UAVs.
- **Sink Node** T: The sink node is a single node representing all the POIs.
- **Arcs** $\mathcal{A}$: There is an arc from a UAV to a POI if the UAV can safely visit the POI. The cost c of an arc is the weight of the corresponding edge in the bipartite graph.
- **Capacity** u: The capacity of an arc from a UAV to a POI is one if the UAV can visit the POI. Otherwise, the capacity is 0 (in this case, these arcs are not created).
- **Balance** b: Balances refer to the conservation of flow at each node in the network. Each feasible solution must satisfy that the amount of flow leaving minus the amount of flow entering the node must be equal to the node's balance.

Let $\mathcal{N}$ be the set of all nodes, $f(a) \in \mathbb{R}_0^+$ the flow on arc $a \in \mathcal{A}$, $\delta^+(n)$ the set of arcs leaving node $n \in \mathcal{N}$, $\delta^-(n)$ the set of arcs entering node $n \in \mathcal{N}$, $c(a)$ the cost per unit of flow on arc $a \in \mathcal{A}$, $u(a)$ the capacity of arc $a \in \mathcal{A}$, and $b(n)$ the balance of node $n \in \mathcal{N}$. The MCF problem with zero lower capacity bound can be formulated as

$$\min_{a \in \mathcal{A}} c(a) \cdot f(a) \tag{3.5}$$

$$\text{s.t.} \sum_{\forall a \in \delta^+(n)} f(a) - \sum_{\forall a \in \delta^-(n)} f(a) = b(n) \qquad \forall n \in \mathcal{N} \tag{3.6}$$

$$0 \le f(a) \le u(a) \qquad \forall a \in \mathcal{A} \tag{3.7}$$

The cost from the source node to each UAV and from each POI to the sink node are set to 0. These arcs exist only to transfer a matching problem to a flow problem and are irrelevant to the final solution. The balance for each node except source S and sink T is set to 0. As an example, see Figure 3.3.

The Algorithm 3 creates priority queues, $\mathcal{Q}_P$ and $\mathcal{Q}_S$, to manage the POIs' order based on the distance from the pUAV and the combined metric of distance and heading change from the sUAV. These computations are identical to Equations 3.3 and 3.4. The assignment of the next goals involves

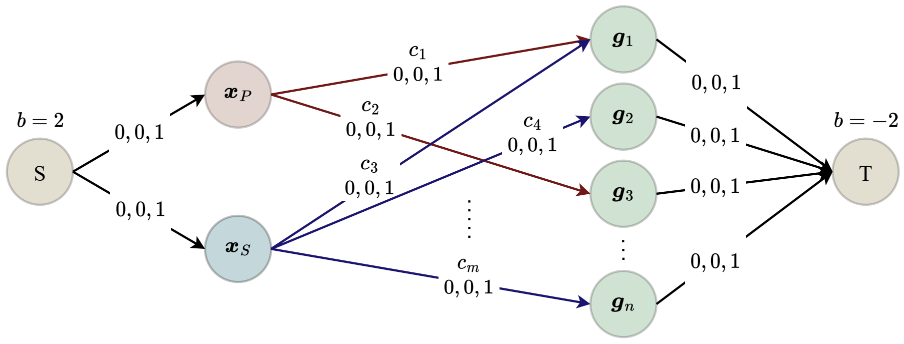


Figure 3.3: Representation of the assignment problem as a minimum cost flow problem, the notation of edge labels $0, 0, 1$ represent capacity's lower bound, current value, and upper bound u, respectively

iteratively selecting POIs $\boldsymbol{g}$ from these priority queues, computing paths using SphereMap planning, and assessing the feasibility of the paths. The planning incorporates considerations for the sizes of the UAVs, s_P and s_S, ensuring paths can be traversed safely.

This algorithm proceeds to construct arcs, $\mathcal{A}_P$ and $\mathcal{A}_S$, connecting the UAVs with accessible POIs, considering their associated costs and path length. To minimize the computational complexity, this work takes advantage of priority queues $\mathcal{Q}_P$ and $\mathcal{Q}_S$ and the fact that the POI are examined for accessibility in a specific order from the most interesting to the least. Parameter N is set, which indicates a sufficient number of arcs for each UAV, meaning that when there is already N accessible POIs found, no additional arcs for that UAV are created.

To ensure that the problem can be solved even if $\mathcal{A}_P$ or $\mathcal{A}_S$ is empty, special arcs $(\boldsymbol{x}_P, \boldsymbol{g}_{x_P}, c_{x_P})$ and $(\boldsymbol{x}_S, \boldsymbol{g}_{x_S}, c_{x_s})$, with high costs c_{x_P} and c_{x_S}, are created. Nodes $\boldsymbol{g}_{x_P}$ and $\boldsymbol{g}_{x_P}$, added to the POIs' set, represent current $\boldsymbol{x}_P$ and $\boldsymbol{x}_S$ positions. So, if no accessible POIs are found, the UAV does not move.

Then, the MCF problem is formulated. The nodes $\mathcal{N}$ include source S and sink T, the current positions of the UAVs $\boldsymbol{x}_P, \boldsymbol{x}_S$, the set of POIs $\mathcal{G}$, and the added points $\boldsymbol{g}_{x_P}$ and $\boldsymbol{g}_{x_P}$. The set of all arcs $\mathcal{A}$ contains arcs $\mathcal{A}_{source}$ from source S to each UAV, arcs $\mathcal{A}_{sink}$ from each POI $g \in \mathcal{G}$ to sink T and computed arcs $\mathcal{A}_P$ and $\mathcal{A}_S$.

To solve the MCF problem, C++ solver from Min-Cost-Flow-Class library[1] was used. The library offers multiple methods, described in [52], but this work incorporates only a solver based on the primal and dual revised network simplex algorithm. It was selected due to its open-source properties.

[1]`https://github.com/frangio68/Min-Cost-Flow-Class/tree/master` (Accessed December 2023)

Algorithm 3: Task allocation - assignment problem

```
Input: set of POIs $\mathcal{G}$, pUAV position $\boldsymbol{x}_P$, sUAV pose $(\boldsymbol{x}_S, \phi_S)$,
       source node $S$, sink node $T$, arcs from source node $\mathcal{A}_{source}$,
       arcs to sink node $\mathcal{A}_{sink}$
Output: pUAV next goal $\boldsymbol{g}_{P_n}$, sUAV next goal $\boldsymbol{g}_{S_n}$
Parameters: pUAV size $s_P$, sUAV size $s_S$, cost $c_{x_P}$, cost $c_{x_S}$,
            number of maximum arcs $N$ for each UAV

1  $\boldsymbol{g}_{x_P} \leftarrow \boldsymbol{x}_P$, $\boldsymbol{g}_{x_S} \leftarrow \boldsymbol{x}_S$
2  $\mathcal{Q}_P \leftarrow$ PRIORITY_QUEUE$(\mathcal{G}, \boldsymbol{x}_P)$        ▷ based on distance from pUAV
3  $\mathcal{Q}_S \leftarrow$ PRIORITY_QUEUE$(\mathcal{G}, \boldsymbol{x}_S, \phi_S)$ ▷ based on distance and heading
     change from sUAV
4  $\mathcal{A}_\mathcal{P} \leftarrow \emptyset$                              ▷ arcs from pUAV
5  $i \leftarrow 0$
6  while $i < N$ and $\mathcal{Q}_P \neq \emptyset$ do
7      $c, \boldsymbol{g} \leftarrow \mathcal{Q}_P$.POP()                     ▷ cost and POI
8      $\mathcal{P} \leftarrow$ FIND_PATH$(\boldsymbol{x}_P, \boldsymbol{g}, s_P)$            ▷ SphereMap planning
9      if $\mathcal{P} \neq \emptyset$ then
10         $c \leftarrow c+$LENGTH$(\mathcal{P})$                   ▷ number of waypoints
11         $\mathcal{A}_P \leftarrow \mathcal{A}_P \cup \{(\boldsymbol{x}_P, \boldsymbol{g}, c)\}$
12         $i \leftarrow i + 1$
13     end
14 end
15 $\mathcal{A}_P \leftarrow \mathcal{A}_P \cup \{(\boldsymbol{x}_P, \boldsymbol{g}_{x_P}, c_{x_P})\}$              ▷ to secure feasibility
16 repeat lines 4-15 for sUAV with $\mathcal{Q}_S, \boldsymbol{x}_S, s_S, \boldsymbol{g}_{x_S}, c_{x_S}$ and $\mathcal{A}_S$
17 $\mathcal{N} \leftarrow \{S, T, \boldsymbol{x}_P, \boldsymbol{x}_S\} \cup \mathcal{G} \cup \{\boldsymbol{g}_{x_P}, \boldsymbol{g}_{x_S}\}$
18 $\mathcal{A} \leftarrow \mathcal{A}_{source} \cup \mathcal{A}_{sink} \cup \mathcal{A}_P \cup \mathcal{A}_S$
19 $\boldsymbol{g}_{P_n}, \boldsymbol{g}_{S_n} \leftarrow$ SOLVE_MIN_COST_FLOW$(\mathcal{N}, \mathcal{A})$ ▷ dual revised network
     simplex algorithm
```

3.2 Path Planning

Safe and fast path planning is critical for successful robot exploration. Safe planning ensures that the robot navigates without compromising its integrity, while quick planning allows it to complete its mission efficiently. These two aspects are complementary, enabling robots to explore unknown environments with minimal risk and maximum efficiency.

The Algorithm 4 outlines a planning process for a cooperative exploration scenario. It operates in two states: MONITORING and PLANNING. In the monitoring state, the algorithm checks if either UAV $u \in \mathcal{U}$, where $\mathcal{U} = \{\text{pUAV, sUAV}\}$, has reached its current goal $\boldsymbol{g}_{u_c}$, creating a set of UAVs waiting for new path $\mathcal{W}$ and updating the set of already visited points $\mathcal{V}$ used in Algorithm 1. When at least one UAV is waiting for new action, the algorithm transitions to the planning state.

During the planning stage, the algorithm first waits for a map $\mathcal{M}$ update. This is necessary to ensure that newly explored areas are also searched for POIs. Then, it computes paths for each UAV $w \in \mathcal{W}$ using one of the two existing planners in the MRS UAV system incorporated in this work:

1. **Planning on SphereMap** - This representation is already described in Section 3.1.2 with its advantages and disadvantages.
2. **MRS SubT planning library** - An A* planner and path post-processing on OctoMap, described in [53], developed for the DARPA SubT Challenge. This library provides the shortest found path with respect to minimum obstacle distance d_O, but the computation is significantly slower than planning on SphereMap.

When planning the path to the selected POI, the first option (planning on SphereMap) was deployed, as it aligns with the priorities of fast computation and safety, as the sUAV is designed to fly through narrow spaces. However, in specific scenarios involving collision avoidance, where the objective is to plan the shortest path to increase mutual UAV distance, the MRS SubT planner is used. This is discussed in Section 3.3. However, it is possible to use the second option even for general planning by setting the corresponding parameter in the configuration file.

The computed paths are transformed into the UAVs' local frames. For pUAV, the transformation matrix ${}^{\mathrm{P}}_{\mathrm{G}}\boldsymbol{T}$ is identity, as the global frame G is initialized from pUAV SLAM frame P. For sUAV, the matrix ${}^{\mathrm{S}}_{\mathrm{G}}\boldsymbol{T}$ is continuously updated by the relative localization described in Section 2.2.2. In Algorithm 4, these two transformations are labeled as ${}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{T}$, where L substitutes the frame corresponding to UAV w.

Each UAV is then instructed to follow its respective path, and the algorithm returns to the monitoring state.

3.3 Obstacle Avoidance

Obstacle avoidance is essential for multi-robot exploration to ensure safe navigation, prevent collisions, and achieve mission goals. Effective algorithms enable robots to navigate complex environments, maintain safe separation, and accomplish their objectives in real-world applications.

The choice of obstacle avoidance method depends on the specific requirements of the multi-robot exploration task. Reactive approaches are well-suited for environments where obstacles are constantly moving or changing. Proactive strategies are more efficient for static or predictable environments, where the robots have more time to plan their movements.

In this thesis, it is assumed that the environment does not contain dynamic obstacles, so only static obstacles and the possible collision of the two UAVs

Algorithm 4: Planning

Input: Map $\mathcal{M}$, pUAV pose $(\boldsymbol{x}_P, \phi_P)$, sUAV pose $(\boldsymbol{x}_S, \phi_S)$, pUAV current goal $(\boldsymbol{g}_{P_c}, \theta_{P_c})$, sUAV current goal $(\boldsymbol{g}_{S_c}, \theta_{S_c})$, pUAV next goal $(\boldsymbol{g}_{P_n}, \theta_{P_n})$, sUAV next goal $(\boldsymbol{g}_{S_n}, \theta_{S_n})$, visited points $\mathcal{V}$, set $\mathcal{U}$ including both UAVs

Output: None or paths ${}^{\mathrm{P}}\mathcal{P}_P$ and ${}^{\mathrm{S}}\mathcal{P}_S$ sent to the UAVs' control pipelines

Parameters: pUAV minimum obstacle distance d_{O_P}, sUAV minimum obstacle distance d_{O_S}

```
switch state do
    case MONITORING do
        $\mathcal{W} \leftarrow \emptyset$
        for $\forall u \in \mathcal{U}$ do
            if $\|\boldsymbol{x}_u - \boldsymbol{g}_{u_c}\|_2 \leq \mathcal{M}.resolution$ then
                $\mathcal{W} \leftarrow \mathcal{W} \cup \{u\}$
                $\mathcal{V} \leftarrow \mathcal{V} \cup \{\boldsymbol{g}_u\}$
                state ← PLANNING
            end
        end
    end
    case PLANNING do
        wait for $\mathcal{M}$ update
        for $\forall w \in \mathcal{W}$ do
            $\mathcal{P}_w \leftarrow$ FIND_PATH$(\mathcal{M}, \boldsymbol{x}_w, \phi_w, \boldsymbol{g}_{w_n}, \theta_{w_n}, d_{O_w})$ ▷ SphereMap planning or MRS SubT planning library
            ${}^{\mathrm{L}}\mathcal{P}_w \leftarrow$ TRANSFORM_PATH$({}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{T}, \mathcal{P}_w)$
            $w$.FOLLOW$({}^{\mathrm{L}}\mathcal{P}_w)$
        end
        state ← MONITORING
    end
end
```

are considered. The static obstacles are handled by the planning process described in Section 3.2. It ensures a safe path in the already explored space.

The Algorithm 5 prevents collision of the two UAVs. It creates three safety zones based on the UAVs' distance. The first one, bounded by the critical distance d_C parameter (set in the configuration file), marks the area where the UAVs are too close to each other, and an avoidance maneuver is necessary. The maneuver takes advantage of the small size of the sUAV. The pUAV stops and is stationary the whole time. A new goal position $\boldsymbol{g}_S \in \mathbb{R}^3$ and heading $\theta_S \in [-\pi, \pi]$, computed by adding normalized direction $\boldsymbol{u} \in \mathbb{R}^3$ from pUAV to sUAV to sUAV's current position $\boldsymbol{x}_S$, and setting desired heading θ_S to sUAV current heading ϕ_S, is assigned to the sUAV (the value of the z-axis stays the same as the current altitude):

$$\boldsymbol{u} = \frac{\boldsymbol{x}_S - \boldsymbol{x}_P}{\|\boldsymbol{x}_S - \boldsymbol{x}_P\|_2} \tag{3.8}$$

$$\boldsymbol{g}_S = \boldsymbol{x}_S + \boldsymbol{u} \tag{3.9}$$

$$\boldsymbol{g}_{S_z} = \boldsymbol{x}_{S_z} \tag{3.10}$$

$$\theta_S = \phi_S \tag{3.11}$$

The path $\mathcal{P}_S$ to the goal $\boldsymbol{g}_S$ is planned by the MRS SubT planning library on global map $\mathcal{M}$, described in Section 3.2, ensuring that it is feasible, short, and safe. It is found even if the goal $\boldsymbol{g}_S$ is not accessible, but there is reachable point in its proximity. This path is then transformed to sUAV VIO frame S using transformation matrix ${}^{\mathrm{S}}_{\mathrm{G}}\boldsymbol{T} \in SE(3)$ and sent to the control pipeline. When the sUAV reaches its goal, the whole process is repeated until the distance between the UAVs is bigger than the critical distance d_C.

In the second zone, when the distance is smaller than the safety distance d_S parameter but bigger than d_C, the safety of pUAV is prioritized, as it is equipped with more precise and more expensive sensors. The pUAV stops its action and waits until the sUAV, which continues in its current action, is far enough.

The last zone, where the distance is bigger than the safety distance d_S, is marked as safe, and the UAVs explore the space as described in previous sections.

This approach is repeated at a high frequency, which can be set in the configuration file. It offers a fast and computationally inexpensive way to avoid collision. It has low time and computational complexity but suffers from several issues. It is not robust to odometry error; it assumes that the computed distance is always precise. It slows down the exploration efforts and, in confined environments, can lead to situations where the sUAV has no free space available and, therefore, is unable to move away from the pUAV. This creates a deadlock, which can be solved only by human intervention. Solving these issues is a subject of future work.

Algorithm 5: Safety check and avoidance

```
Input: Map $\mathcal{M}$, pUAV pose $(\boldsymbol{x}_P, \phi_P)$, sUAV pose $(\boldsymbol{x}_S, \phi_S)$,
Output: None or path ${}^{\mathrm{S}}\mathcal{P}_S$ sent to the sUAV's control pipeline
Parameters: minimum safety distance $d_S$, minimum critical
            distance $d_C$, sUAV minimum obstacle distance $d_O$
1  if $\|\boldsymbol{x}_S - \boldsymbol{x}_P\|_2 \leq d_C$ then
2      pUAV.STOP()
3      sUAV.STOP()
4      while $\|\boldsymbol{x}_S - \boldsymbol{x}_P\|_2 \leq d_C$ do
5          $\boldsymbol{u} \leftarrow \frac{\boldsymbol{x}_S - \boldsymbol{x}_P}{\|\boldsymbol{x}_S - \boldsymbol{x}_P\|_2}$
6          $\boldsymbol{g}_S \leftarrow \boldsymbol{x}_S + \boldsymbol{u}$
7          $\theta_S \leftarrow \phi_S$          ▷ Desired orientation is the same as current
            orientation
8          $\mathcal{P}_S \leftarrow$ FIND_PATH$(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{S}}, \phi_S, \boldsymbol{g}_S, \theta_S, d_O)$     ▷ Using A* MRS
            SubT planner
9          ${}^{\mathrm{S}}\mathcal{P}_S \leftarrow$ TRANSFORM_PATH$({}^{\mathrm{S}}_{\mathrm{G}}\boldsymbol{T}, \mathcal{P}_S)$
10         sUAV.FOLLOW$({}^{\mathrm{S}}\mathcal{P}_S)$
11         wait until $\boldsymbol{g}_S$ is reached
12         get updated UAVs' poses $(\boldsymbol{x}_P, \phi_P)$ and $(\boldsymbol{x}_S, \phi_S)$
13     end
14 else if $\|\boldsymbol{x}_S - \boldsymbol{x}_P\|_2 \leq d_S$ then
15     pUAV.STOP()
16     sUAV.CONTINUE()
17 else
18     pUAV.CONTINUE()
19     sUAV.CONTINUE()
20 end
```

Chapter 4

Experiments

This chapter provides an analysis of the proposed method, demonstrates its function in two simulation environments, and shows results from real-world experiments. Videos of the performed simulations and experiments are available on `https://mrs.felk.cvut.cz/cihlarova2024thesis`.

The performance of all the algorithms proposed in Chapter 3 is evaluated, and the greedy approach and the reduction to the flow problem for solving the task allocation problem are tested and compared. The desired result of an exploration mission is an occupancy map of the environment containing information from both UAVs.

Two different colors, red and blue, distinguish the parts of the environment explored by different UAV. Red is used for space, which was scanned by pUAV's LiDAR and possibly also seen by sUAV, but the LiDAR data are more precise, therefore information from pUAV's occupancy map is prioritized. The blue color marks space seen only by sUAV. The occupancy maps are shown from the top view, and the floor and ceiling are not visualized in the figures but are part of the results.

The speed, acceleration, jerk, and snap of the UAVs are limited. These values are listed in Table 4.1 and are identical for all the experiments described in this chapter.

	Speed [m/s]	Acceleration [m/s^2]	Jerk [m/s^3]	Snap [m/s^4]
Horizontal	0.5/1.0	1.0/1.0	20.0/20.0	20.0/20.0
Vertical (Asc)	0.5/1.0	1.0/1.0	20.0/20.0	20.0/20.0
Vertical (Des)	0.5/1.0	1.0/1.0	20.0/20.0	20.0/20.0
Heading	0.5/0.5	0.5/1.0	10.0/20.0	10.0/20.0

	Roll [deg/s]	Pitch [deg/s]	Yaw [deg/s]	
Angular Speed	10.0/60.0	10.0/60.0	10.0/60.0	

Table 4.1: pUAV/sUAV movement constraints

4.1 Simulation

Before executing real-world experiments, the proposed algorithms were extensively tested in the Gazebo robotic simulator.

This chapter goes over two simulation environments. The first one is an office space created in the Gazebo robotic simulator, and the second one, obtained by a 3D scanner, provides a model of a real-world warehouse near Prague. In both cases, the UAVs were spawned with almost exact hardware specifications as described in Section 2.1. The only difference is a substitution of Ouster OS0-128 for LiDAR with only 512×32 resolution to reduce the computational requirements of the simulation. The UAVs do not have any prior knowledge of the environments, and all the information is obtained during the exploration mission, which is started after initialization of the simulation and run without any user intervention.

To evaluate the performance of the proposed algorithms in isolation from localization errors and to reduce the computational demands of the simulation, ground-truth data were used for self-localization of both UAVs and relative localization between the UAVs.

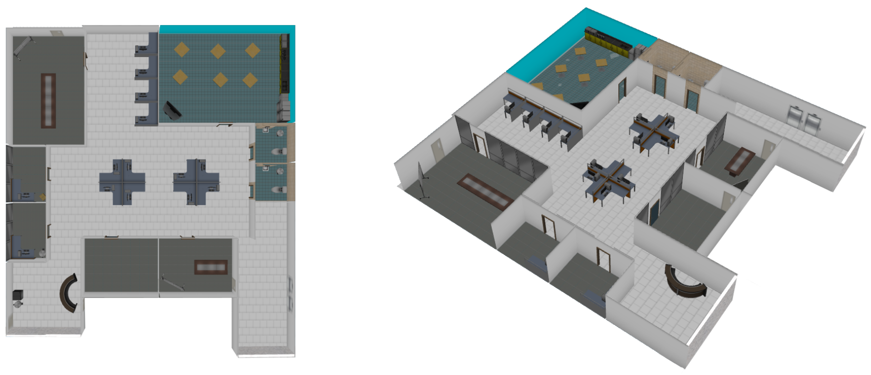
Table 4.2 provides a list of parameters used for the simulations. These parameters are defined in the configuration file and are identical for both environments.

Parameter	**Notation**	**Value**	**Unit**
pUAV size (radius)	s_P	0.45	m
sUAV size (radius)	s_S	0.25	m
minimum safety distance	d_S	2.5	m
minimum critical distance	d_C	2.0	m
minimum radius of spheres	r_{sph}	0.35	m
frontier detection rate	F_{front}	0.5	Hz
path planning rate	F_{path}	2	Hz
collision avoidance rate	F_{coll}	10	Hz

Table 4.2: Exploration parameters for the simulation experiments

4.1.1 Office

The first set of experiments was executed in an environment created in the Gazebo robotic simulator visualized in Figure 4.1. It simulates an office space, approximately $23 \times 23 \times 3$ meters, with a spacious common area and multiple smaller rooms accessible through open doors, around 0.75 meters wide, where only the sUAV can safely fit. This creates appropriate testing conditions, as the correct execution of proposed algorithms can be easily examined. The environment also includes multiple objects, such as tables, chairs, whiteboards, etc., to make it more complex.



(a) : Top view **(b) :** Side view

Figure 4.1: Gazebo model of an office environment used for the experiments

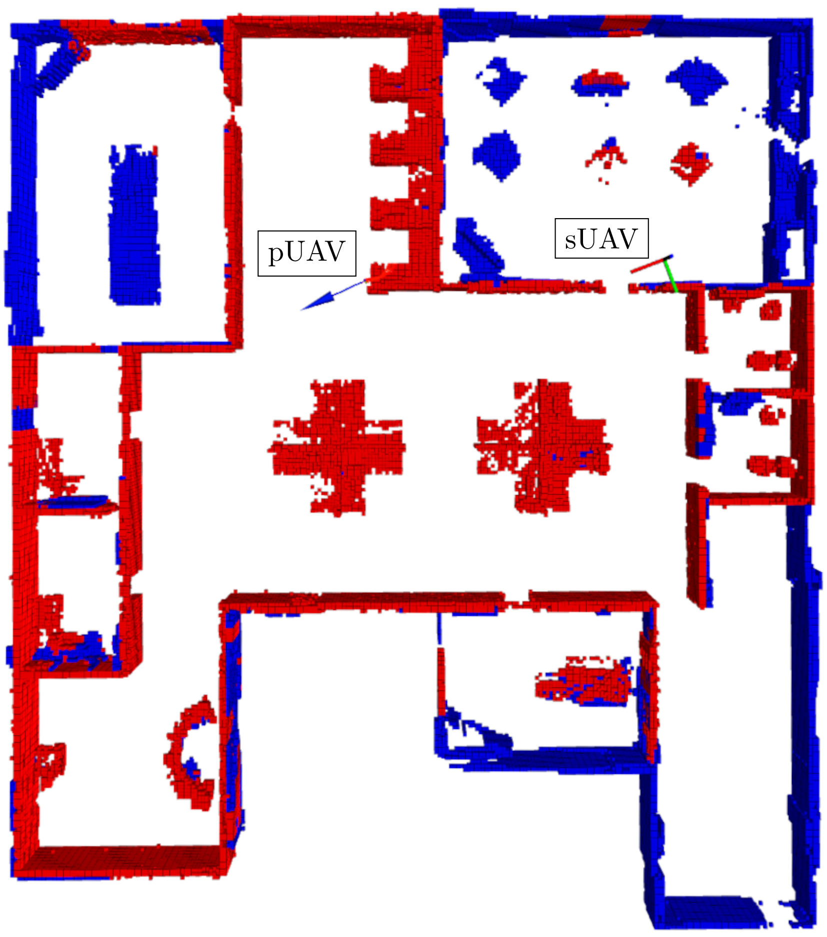
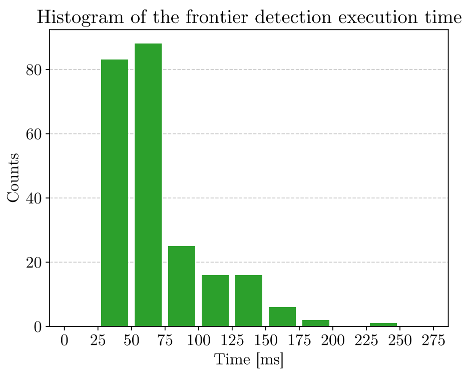


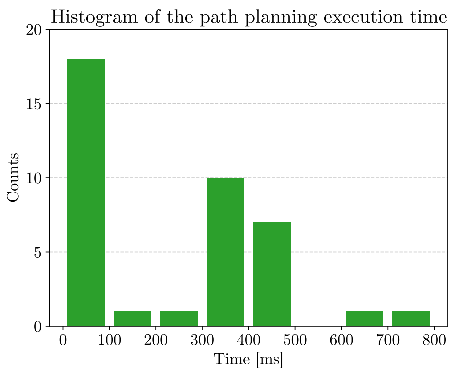
Figure 4.2: Occupancy map of the office space at the end of the 4-minute mission. The red color represents an area scanned by the pUAV using LiDAR, the blue color represents space explored only by sUAV.

Figure 4.2 shows the final occupancy map created by the map merger node described in Section 2.2.2. This map was obtained as the result of an exploration mission that lasted around 4 minutes using the greedy approach described in 3.1.3 to solve the allocation challenge. It can be observed that the common, spacious area is explored by the pUAV, whereas the smaller rooms, accessible only through the door, are explored only by the sUAV. The space's floor and ceiling are not visualized in the figure but are part of the result.

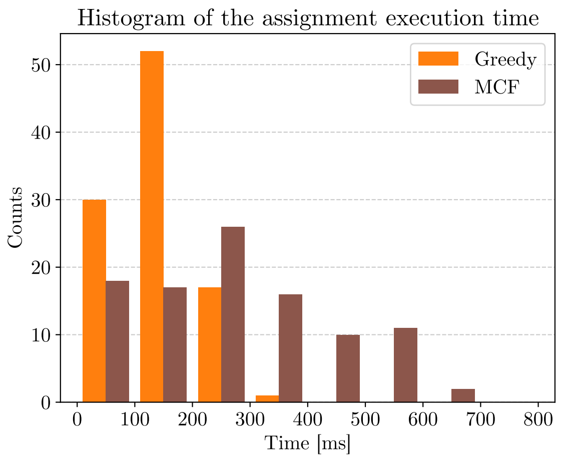
To analyze separate parts of the method, the processing times of frontier detection, goal assignment, and path planning were measured. These values were obtained during the experiments and provide a general idea about the algorithms' complexity. The results are shown in Figure 4.3. The x-axis represents the time in milliseconds, and the y-axis represents the counts of how many times each algorithm completed its objective within a certain time interval.



(a) : Frontier detection



(b) : Path planning



(c) : Goal assignment

Figure 4.3: Execution time of the algorithms in the office environment

Frontier detection's processing time (see Figure 4.3a) depends on the map. In an indoor environment, the processing time is significantly higher at the beginning of the exploration, as most of the leaf nodes to be explored are free. As the exploration progresses, the number of free leaf nodes $l \in \mathcal{M}_{\text{free}}$ is decreasing, therefore, the condition in Algorithm 1 at line 3 is satisfied fewer times. The median processing time of frontier detection was 59.64 ms.

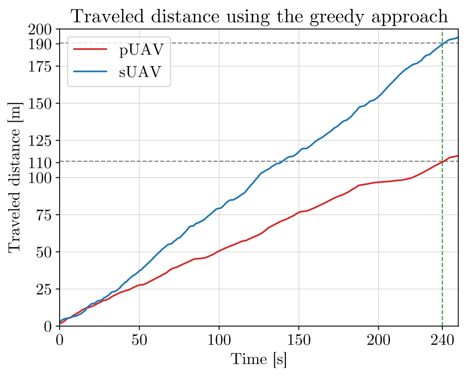
The number of found POIs directly affects the processing time of both implemented methods for goal assignment. These effects are discussed later with the comparison of the two methods.

The processing time of the path-planning algorithm differs based on the *state* variable. However, the *MONITORING* state's processing time is insignificant in comparison to the *PLANNING* and therefore was not measured. The number of UAVs waiting for new action has the main effect on the *PLANNING* state's time. Figure 4.3b shows two peaks in the histogram. This is expected as there are two UAVs, and when both of them need a new path, the processing time doubles.

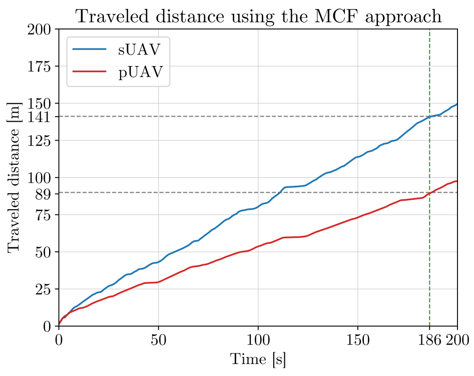
To compare the two algorithms for solving the task allocation challenge described in Section 3.1.3, experiments with identical exploration parameters (see Table 4.2) and identical UAVs' initial position were executed. The parameter N used in the MCF approach is set to $N = 5$, meaning the algorithm looks only for the first five accessible POIs for each UAV.

Both approaches successfully explored the environment and created the occupancy map visualized in Figure 4.2. However, the MCF approach significantly improves the exploration efforts. As shown in Figure 4.4c, using the greedy approach, it took $t_{greedy} = 240.68\ s$ to explore 95 % of the space, whereas the MCF algorithm was able to achieve the same result in $t_{mcf} = 186.55\ s$. The 95% boundary is marked by a red horizontal dashed line in Figure 4.4c, and the gray vertical dashed lines correspond to t_{greedy} and t_{mcf}. If the greedy approach and time t_{greedy} is considered the baseline solution, the MCF results in almost 30% improvement. This is directly linked to the total traveled distance by the UAVs. Figures 4.4a and 4.4b show the distance traveled during the exploration in both scenarios, and it is clear that thanks to the earlier finish when using the MCF approach, the traveled distance is lower. The green vertical dashed line is at value t_{greedy} in Figure 4.4a, and at t_{mcf} in Figure 4.4b. The gray vertical dashed lines show the corresponding traveled distance for the pUAV and the sUAV.

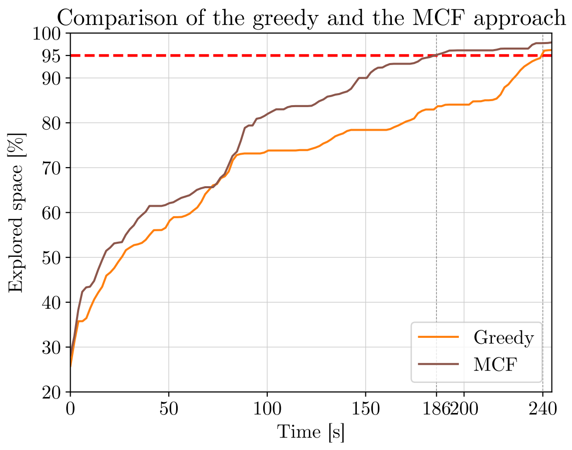
However, the drawback of the MCF lies in higher execution time. The performance of the two algorithms was evaluated by measuring their respective processing times. The results are shown in Figure 4.3c, a histogram of both algorithms' goal assignment processing time created from 100 samples for each method. It is visible that even though the MCF approach overall speeds up the exploration, the computation of the global objective is more expensive. The median processing time of the greedy approach was 144.53 ms, and the median processing time for the MCF approach was 253.76 ms.



(a) : Traveled distance using the greedy approach



(b) : Traveled distance using the MCF approach



(c) : Goal assignment

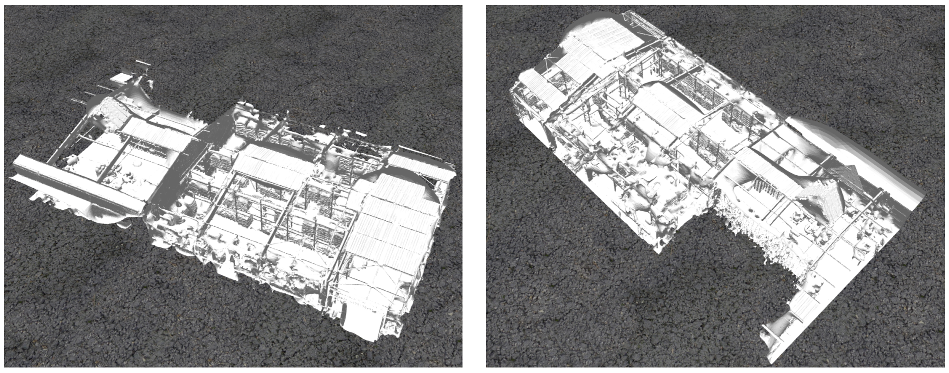
Figure 4.4: Comparison of the greedy and the MCF approach for solving the task allocation challenge in the office environment

4.1.2 Warehouse

The environment visualized in Figure 4.5 was selected as it provides a 3D model of an existing industrial warehouse, where the real-world experiments described in the following section were executed. The 3D model was captured by the Leica BLK360 laser scanner.

The transition from simulation to the real world is challenging, as the simulation creates a perfect world without measurement errors and mainly without hardware malfunction. These experiments aimed to prepare the algorithms, mainly the configuration file, as much as possible to avoid tuning the parameters in the limited time window for real-world experiments. They also provide an idea of what can be expected from the real-world test.

Figure 4.6 shows the occupancy map at the end of the 3-minute exploration mission. The aisles are wide enough to fit both UAVs, so the pUAV can precisely scan the whole space. However, the exploration efforts were divided,



(a) : Front view **(b) :** Back view

Figure 4.5: Gazebo model of a warehouse environment used for the experiments

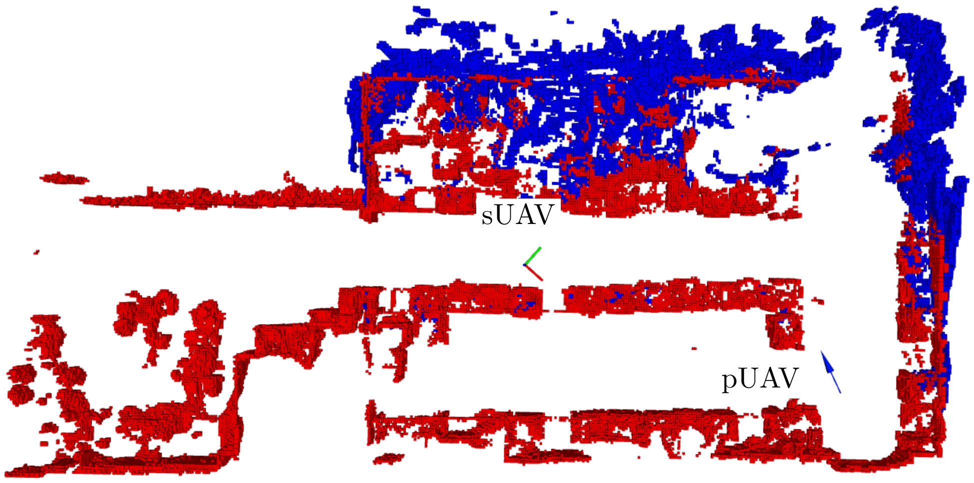


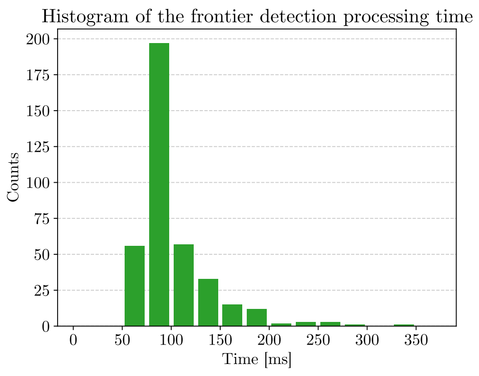
Figure 4.6: Occupancy map of the warehouse at the end of the 3-minute mission in simulation

which led to lower exploration time.

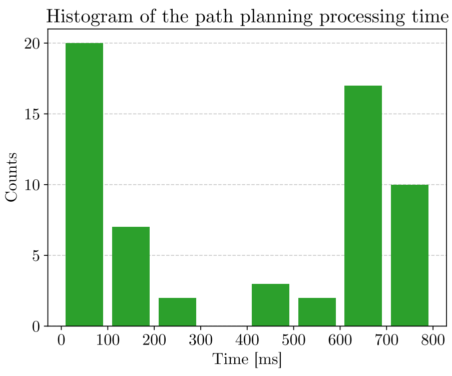
The same analysis as in the office environment was executed. The histogram of processing times can be seen in Figure 4.7. These values correspond to the expected results described in Section 4.1.1. Figure 4.8 shows the results from comparing the goal allocation methods, and as expected, the MCF method explored 95% of the space faster. However, the distance traveled by the pUAV is slightly higher. This is due to the fact that it went exploring the room on the left by itself and then traveled all the way back to help the sUAV.

4.2 Real World

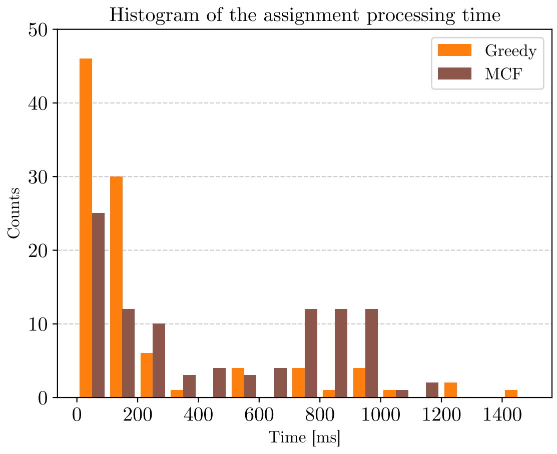
The proposed algorithms were tested in a real-world industrial warehouse environment. The task was identical to the simulation: to explore as much space as possible and generate an occupancy map. Used hardware described



(a) : Frontier detection



(b) : Path planning

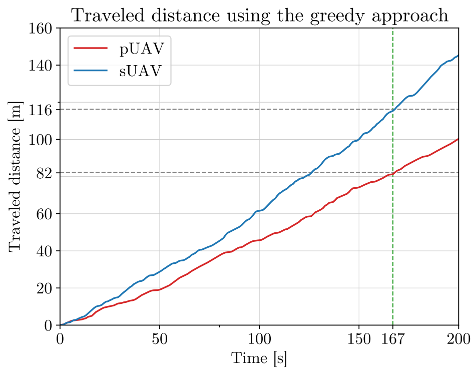


(c) : Goal assignment

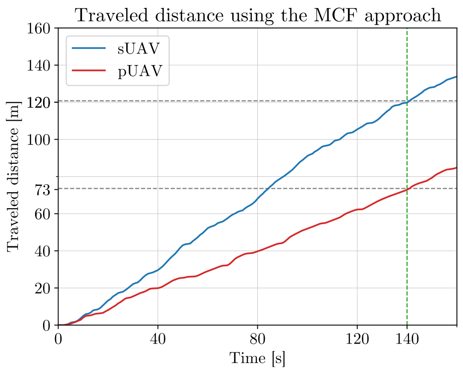
Figure 4.7: Execution time of the algorithms in the warehouse environment in simulation

in Section 2.1 provided the only available computational resources. The proposed algorithms and the ones described in Section 2.2.2 ran fully on board the pUAV. Section 2.2.3 provides an overview of algorithms executed on the sUAV.

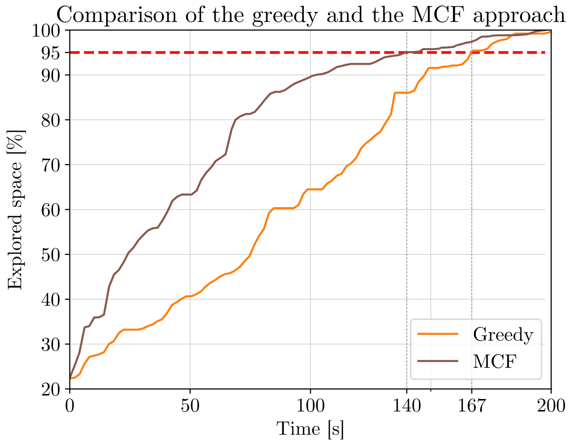
The UAVs could not explore the entire space presented in Section 4.1.2. During the experiment, the main focus was on the safety of the pUAV due to the expensive equipment it was carrying. This was ensured by significantly increasing the UAV's size s_P in the configuration file, the safety distance d_S, and the critical distance d_C. The size of the sUAV s_S was also increased (see Table 4.3). The pUAV was then placed in the main aisle to provide as much data about the environment as possible; see Figure 4.10. So, these changes in the configuration prevented the sUAV from properly exploring the surroundings as the safety measures often blocked it. Additionally, the experiments were executed during working hours, which resulted in a limited available area and prematurely finished experiments to secure the warehouse staff's safety.



(a) : Traveled distance using the greedy approach



(b) : Traveled distance using the MCF approach



(c) : Goal assignment

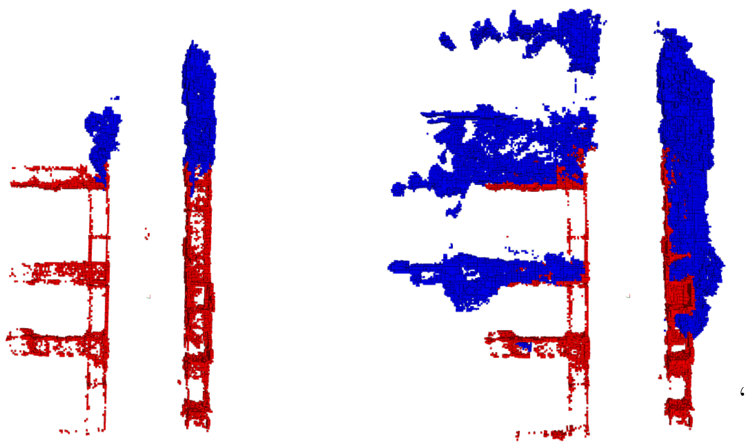
Figure 4.8: Comparison of the greedy and the MCF approach for solving the task allocation in the warehouse environment in simulation

However, the experiments provided useful data regarding the real-life execution time of the proposed approach. The environment was complex, so random samples from the frontiers cluster were selected to enlarge the POIs's set. Unfortunately, this resulted in around 300 POI due to a mistake in the algorithm at the start of the exploration, but as it progressed, the number of POIs rapidly decreased.

The assignment challenge was solved using the MCF, which outperformed the greedy method in simulation experiments. However, due to security measures, none of the POIs were reachable by the pUAV. This led to the worst-case scenario, where the accessibility of all the POIs had to be verified by computing and checking their paths at least once. Despite the large number of POIs, the maximum time required to solve the assignment problem was $t_{assign} = 1.15\ s$. This time was measured during the exploration. It indicates that the algorithm can handle large and complex environments with hundreds of POIs. The final map from one of the successful experiments is

Parameter	Notation	Value	Unit
pUAV size (radius)	s_P	1.8	m
sUAV size (radius)	s_S	0.4	m
minimum safety distance	d_S	4.0	m
minimum critical distance	d_C	3.5	m
minimum radius of spheres	r_{sph}	0.9	m
frontier detection rate	F_{front}	0.5	Hz
path planning rate	F_{path}	2	Hz
collision avoidance rate	F_{coll}	10	Hz

Table 4.3: Exploration parameters for real-world experiments



(a) : Initial position **(b) :** Final position

Figure 4.9: Global occupancy map during real-world experiments

shown in Figure 4.9b. Figure 4.9 also includes the occupancy grid at the initial position to show the exploration progress. Figures 4.10 and 4.11 show photographs taken during the real-world experiment.

During the experiment, drifts in the estimated position provided by the relative localization were noticed. The transformation ${}^{\mathrm{S}}_{\mathrm{G}}\boldsymbol{T}$ was not always correct, and, therefore, the path followed by the sUAV differed from the one planned by the pUAV. This occurred mainly in cases where the sUAV was not tracked by the pUAV for a prolonged period, and the only information available about its position was from received odometry. This behavior is dangerous as it does not ensure the safety of the sUAV, and methods to resolve this issue are part of future work.

These experiments proved that the proposed method is able to run in real time on the available hardware. Even during the worst-case scenario, where there are no accessible POIs for pUAV, the algorithms are fast enough to ensure smooth exploration. However, they also pointed out issues that must be addressed before executing additional real-world tests.

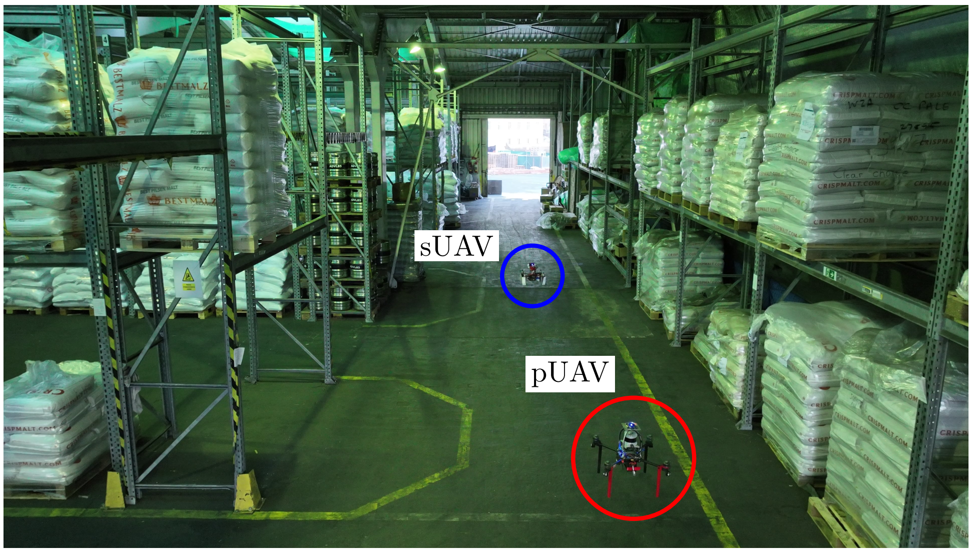


Figure 4.10: Photograph of the initial position during the real-world experiment

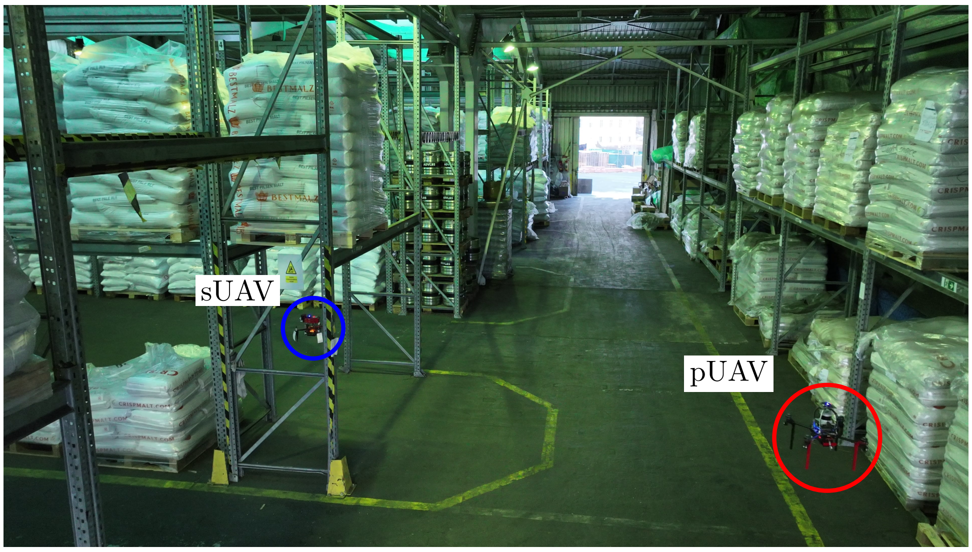


Figure 4.11: Photograph taken during the real-world experiment

Chapter 5

Conclusion

This master's thesis presented a novel method for unknown-space exploration using a heterogeneous UAV team. The method consisted of multiple algorithms that optimized different aspects of the exploration process. The frontier-based exploration strategy was the foundation for the team's collective movement and information gathering.

Two distinctive approaches were proposed to solve the task allocation problem. Firstly, the greedy approach was implemented that prioritizes selecting POIs based on Euclidean distance and individual UAV priority, ensuring low computational complexity. Secondly, a reduction to a minimum cost flow problem was employed, considering global objectives derived from mutual distances to POIs and corresponding path lengths.

A key component of the method involves determining the accessibility of POIs. This task was addressed using SphereMap. The approach enables fast path computation while assessing the distance to the nearest obstacle along the designated path.

Path planning was performed using two different methods. The first involves planning on SphereMap, characterized by its speed and emphasis on safety. The second employs the MRS SubT planning library. The method has higher computational complexity but provides the shortest path considering a given obstacle distance. The MRS SubT planning library is used during collision avoidance maneuvers, which prevents collision of the two UAVs.

The algorithms were tested and analyzed in simulation environments to validate this method. Additionally, real-world experiments were conducted, and the performance on board the UAVs was evaluated. These experiments not only showed the properties of the proposed method but also provided valuable insights into its real-world applicability and potential for further improvement.

The entire thesis assignment has been successfully fulfilled. According to the assignment, the following tasks have been completed:

1. An overview of the current state of the art in exploration strategies and multi-UAV exploration is provided in Section 1.2.

2. A multi-robot exploration method for a team of heterogeneous UAVs was designed. The description can be found in Chapter 3.
3. The method was built upon MRS UAV system described in Chapter 2.
4. The method was tested in two simulation environments and analyzed. The results of performed simulation tests are described in Chapter 4, Section 4.1.
5. The approach was tested in real-world experiments, and the results are described in Chapter 4, Section 4.2.

5.1 Future Work

The main object of future work, as mentioned in Section 3.3, could be improving the collision avoidance method. Different approaches, for example, RRT* planning, could be implemented to re-plan the part of the path provided by the SphereMap at which the collision would occur.

Additionally, all the algorithms except the collision avoidance approach were implemented to work with any number of dependent UAVs. In the future, a scalable collision avoidance approach could also be created.

The odometry drift detected during real-world experiments could be resolved by scheduling UAV meet-ups. When the sUAV would be out of the LiDAR-based detector range for a longer time period, a request for a meet-up could be made. This would trigger a special goal assignment approach that selects goals for the UAVs based on mutual visibility.

Finally, to solve the assignment problem, additional methods from the MCF library could be incorporated. Moreover, the problem could be formulated as minimum weight perfect matching by adding fake UAVs to match the number of POIs. Using this formulation, the Hungarian algorithm, with guaranteed polynomial time complexity, could be implemented to solve the problem.

5.2 Utilization of Generative AI tools

During the preparation of this work, the author used ChatGPT and Bing Copilot to improve readability and language. Also, the AI tool Perplexity was used to search for additional related work in Section 1.2. Each provided research paper was thoroughly checked. After using these services, the author reviewed and edited the content as needed and takes full responsibility for the content of this work.

Bibliography

[1] G. Li, T.-W. Wong, B. Shih, C. Guo, L. Wang, J. Liu, T. Wang, X. Liu, J. Yan, B. Wu, *et al.*, “Bioinspired soft robots for deep-sea exploration,” *Nature Communications*, vol. 14, no. 1, p. 7097, 2023.

[2] G. Muscato, F. Bonaccorso, L. Cantelli, D. Longo, and C. Melita, “Volcanic environments: Robots for exploration and measurement,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 40–49, 2012.

[3] M. Petrlik, P. Petráček, V. Krátký, T. Musil, Y. Stasinchuk, M. Vrba, T. Báča, D. Heřt, M. Pecka, T. Svoboda, and M. Saska, “Uavs beneath the surface: Cooperative autonomy for subterranean search and rescue in darpa subt,” *Field Robotics*, vol. 3, p. 1–68, Jan. 2023.

[4] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, “Multi-uav exploration with limited communication and battery,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2230–2235, 2015.

[5] G. Skorobogatov, C. Barrado, and E. Salamí, “Multiple uav systems: A survey,” *Unmanned Systems*, vol. 08, no. 02, pp. 149–169, 2020.

[6] C. I. Connolly, “The determination of next best views,” *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 432–435, 1985.

[7] 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)*, pp. 1462–1468, 2016.

[8] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, “An efficient sampling-based method for online informative path planning in unknown environments,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[9] L. Schmid, C. Ni, Y. Zhong, R. Siegwart, and O. Andersson, “Fast and compute-efficient sampling-based local exploration planning via distribution learning,” *IEEE Robotics and Automation Letters*, vol. 7, p. 7810–7817, July 2022.

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

[11] B. Yamauchi, “Frontier-based exploration using multiple robots,” in *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS ’98, (New York, NY, USA), p. 47–53, Association for Computing Machinery, 1998.

[12] M. Keidar, E. Sadeh-Or, and G. A. Kaminka, “Fast frontier detection for robot exploration,” in *Advanced Agent Technology* (F. Dechesne, H. Hattori, A. ter Mors, J. M. Such, D. Weyns, and F. Dignum, eds.), (Berlin, Heidelberg), pp. 281–294, Springer Berlin Heidelberg, 2012.

[13] M. Keidar and G. A. Kaminka, “Efficient frontier detection for robot exploration,” *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 215–236, 2014.

[14] S. S. Belavadi, R. Beri, and V. Malik, “Frontier exploration technique for 3d autonomous slam using k-means based divisive clustering,” in *2017 Asia Modelling Symposium (AMS)*, pp. 95–100, 2017.

[15] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, “Rapid exploration with multi-rotors: A frontier selection method for high speed flight,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2135–2142, 2017.

[16] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using rao-blackwellized particle filters.,” in *Robotics: Science and systems*, vol. 2, pp. 65–72, 2005.

[17] S. Ahmad, A. B. Mills, E. R. Rush, E. W. Frew, and J. S. Humbert, “3d reactive control and frontier-based exploration for unstructured environments,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2289–2296, 2021.

[18] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, “Fast frontier-based information-driven autonomous exploration with an mav,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2020.

[19] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, “Efficient autonomous exploration planning of large-scale 3-d environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[20] F. Yang, D.-H. Lee, J. Keller, and S. Scherer, "Graph-based topological exploration planning in large-scale 3d environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12730–12736, 2021.

[21] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[22] K. Masaba and A. Q. Li, "Gvgexp: Communication-constrained multi-robot exploration system based on generalized voronoi graphs," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 146–154, 2021.

[23] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413–14423, 2020.

[24] J. Faigl, M. Kulich, and L. Přeučil, "Goal assignment using distance cost in multi-robot exploration," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3741–3746, 2012.

[25] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[26] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X* (E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, eds.), (Berlin, Heidelberg), pp. 157–173, Springer Berlin Heidelberg, 2013.

[27] B. Yamauchi, "Decentralized coordination for multirobot exploration," *Robotics and Autonomous Systems*, vol. 29, no. 2, pp. 111–118, 1999.

[28] A. J. Smith and G. A. Hollinger, "Distributed inference-based multi-robot exploration," *Autonomous Robots*, vol. 42, pp. 1651–1668, 2018.

[29] J. Yu, J. Tong, Y. Xu, Z. Xu, H. Dong, T. Yang, and Y. Wang, "Smmr-explore: Submap-based multi-robot exploration system with multi-robot multi-target potential field exploration method," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8779–8785, 2021.

[30] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[31] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott,

et al., “Autonomous teamed exploration of subterranean environments using legged and aerial robots,” in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3306–3313, IEEE, 2022.

[32] A. A. et al., “Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge,” *arXiv pre-print*, 2021.

[33] N. e. a. Hudson, “Heterogeneous ground and air platforms, homogeneous sensing: Team csiro data61’s approach to the darpa subterranean challenge,” *Field Robotics*, vol. 2, p. 595–636, Mar. 2022.

[34] G. Best, R. Garg, J. Keller, G. A. Hollinger, and S. Scherer, “Resilient multi-sensor exploration of multifarious environments with a team of aerial robots,” in *Robotics: Science and Systems (RSS)*, 05 2022.

[35] B. Zhou, H. Xu, and S. Shen, “Racer: Rapid collaborative exploration with a decentralized multi-uav system,” *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1816–1835, 2023.

[36] V. Pritzl, M. Vrba, P. Štěpán, and M. Saska, “Fusion of visual-inertial odometry with lidar relative localization for cooperative guidance of a micro-scale aerial vehicle,” *arXiv pre-print*, 2023.

[37] V. Pritzl, M. Vrba, Y. Stasinchuk, V. Krátký, J. Horyna, P. Štěpán, and M. Saska, “Drones guiding drones: Cooperative navigation of a less-equipped micro aerial vehicle in cluttered environments,” *arXiv pre-print*, 2023.

[38] D. Hert, T. Baca, P. Petracek, V. Kratky, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, G. Silano, D. Bonilla Licea, P. Stibinger, R. Penicka, T. Nascimento, and M. Saska, “Mrs modular uav hardware platforms for supporting research in real-world outdoor and indoor environments,” in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1264–1273, 2022.

[39] D. Hert, T. Baca, P. Petracek, V. Kratky, R. Penicka, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, M. Sramek, A. Ahmad, G. Silano, D. B. Licea, P. Stibinger, T. Nascimento, and M. Saska, “MRS drone: A modular platform for real-world deployment of aerial multi-robot systems,” *Journal of Intelligent & Robotic Systems*, vol. 108, jul 2023.

[40] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, “The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles,” *Journal of Intelligent & Robotic Systems*, vol. 102, apr 2021.

[41] 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.

[42] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6753–6760, 2018.

[43] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, pp. 649–666, Springer, 2016.

[44] M. Burri, H. Oleynikova, , M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2015.

[45] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 5420–5425, 2010.

[46] J. Zhang and S. Singh, "Loam : Lidar odometry and mapping in real-time," *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, 01 2014.

[47] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, 04 2013.

[48] M. Vrba, Y. Stasinchuk, T. Báča, V. Spurný, M. Petrlík, D. Heřt, D. Žaitlík, and M. Saska, "Autonomous capture of agile flying objects using uavs: The mbzirc 2020 challenge," *Robotics and Autonomous Systems*, vol. 149, p. 103970, 2022.

[49] M. Vrba, V. Walter, V. Pritzl, M. Pliska, T. Báča, V. Spurný, D. Heřt, and M. Saska, "On onboard lidar-based flying object detection," *arXiv pre-print*, 2023.

[50] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, (Paris, France), 2020.

[51] T. Musil, M. Petrlík, and M. Saska, "Spheremap: Dynamic multi-layer graph structure for rapid safety-aware uav planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11007–11014, 2022.

[52] A. Frangioni and A. Manca, “A computational study of cost reoptimization for min-cost flow problems,” *INFORMS Journal on Computing*, vol. 18, no. 1, pp. 61–70, 2006.

[53] V. Krátký, 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.

List of Terms

A-LOAM	Advanced implementation of LOAM
DARPA	Defense Advanced Research Projects Agency
FFD	Fast Frontier Detector
FOV	Field of View
IMU	Inertial Measurement Unit
IR	infrared
LiDAR	Light Detection and Ranging
LOAM	LiDAR Odometry and Mapping
MCF	Minimum-cost flow
MPC	Model predictive control
MRS group	Multi-robot Systems group
MRS UAV system	Multi-robot Systems Group UAV system
MRTA	Multi-Robot Task Allocation
NBV	Next-Best-View

POI Point of Interest

pUAV primary Unmanned Aerial Vehicle

RGB Red-Green-Blue

RGBD Red-Green-Blue-Depth

ROS Robot Operating System

RRT Rapidly Exploring Random Tree

SLAM Simultaneous Localization And Mapping

sUAV secondary Unmanned Aerial Vehicle

SubT Challenge Subterranean Challenge

TSP Traveling Salesman Problem

UAV Unmanned Aerial Vehicle

VIO Visual Inertial Odometry

WFD Wavefront Frontier Detector