Autonomous UAV-UGV Robot Collaboration for Exploration and Mapping of Unknown Environments

by

Noor Khabbaz

A thesis submitted to the School of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Master of Applied Science in Mechanical Engineering

Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada
April, 2024

THESIS EXAMINATION INFORMATION

Submitted by: **Noor Khabbaz**

M.A.Sc. in **Mechanical Engineering**

Thesis Title: Autonomous UAV-UGV Robot Collaboration for Exploration and Mapping of Unknown Environments

An oral defence of this thesis took place on April 5, 2024 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Amirkianoosh Kiani
Research Supervisor	Dr. Scott Nokleby
Examining Committee Member	Dr. Hoaxiang Lang
Thesis Examiner	Dr. Aaron Yurkewich

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

This thesis addresses the limitations of existing approaches to autonomous exploration and mapping of unknown environments that use multiple Unmanned Ground Vehicles (UGVs). An Unmanned Aerial Vehicle (UAV) is introduced into the multi-robot system to overcome the challenges of relative localization and obstacle detection. A novel method is proposed for autonomously determining the UGVs' starting poses using ArUco markers visible to the UAV, resulting in the initialization of a global merged map. A second method is developed to overcome UGV obstacle detection limitations. UAV depth camera data is processed to detect and incorporate previously unseen obstacles into the UGVs' navigation schemes, enabling avoidance. Experimental validation demonstrates the effectiveness of both methods in enhancing system autonomy. The integration of a UAV into multi-robot systems presents a promising solution to address UGVs' localization challenges and limited field of view to improve their functionality in hazardous environments.

Keywords: Unmanned Aerial Vehicle (UAV), Unmanned Ground Vehicle (UGV), UAV-UGV collaboration, robot localization, obstacle detection, autonomous exploration

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ontario Tech University to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Tech University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Statement of Contributions

Part of the work described in Chapter 3 has been published as:
N. Khabbaz and S. Nokleby, "ArUco-Based Global Map Initialization for Multi-Robot Exploration," in *Proceedings of the 2023 CCToMM Symposium on Mechanisms, Machines, and Mechatronics*, 2023.

At the time of writing, part of the work described in Chapters 5 and 6 has been accepted for publication as:
N. Khabbaz and S. Nokleby, "UAV Obstacle Mapping for Multi-UGV Exploration and Mapping," in *Proceedings of the 2024 USCToMM Symposium on Mechanical Systems and Robotics*, to appear.

Acknowledgements

I would first like to thank my supervisor, Dr. Scott Nokleby, for his support and guidance. Your willingness to make time for me, offering insightful perspectives, has been instrumental in my success.

I would also like to thank my fellow researchers in the Mechatronic and Robotic Systems (MARS) Laboratory. Chris, Troy, Cameron, and Bryce — you provided me with support and camaraderie during my drone experiments. You also made the lab a fun place to work in.

Most importantly, thank you to my parents for encouraging me through my academic journey and all that it has taken to get to this point. And to my partner Ben, for being an enthusiastic listener and my closest confidant through the ups and downs of research.

I would like to express my gratitude to the DND IDEaS program for the financial support of my research. This work was also supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Project Background

This research provides novel methods to address two fundamental problems in the field of mobile robotics: relative localization between multiple robots and obstacle avoidance. These methods are applied to the task of multi-robot exploration and mapping, building upon a novel exploration algorithm presented in [1, 2]. The previous work focused on developing a multi-robot exploration algorithm aimed at facilitating coordination among Unmanned Ground Vehicles (UGVs) for efficient exploration and mapping of unknown environments, thereby minimizing time and energy expenditures.

This thesis improves the capabilities of the previously developed method by significantly increasing the autonomy of multi-robot systems through the introduction of an Unmanned Aerial Vehicle (UAV). The UAV plays a pivotal role in autonomously initializing the shared global map among UGVs and identifying previously undetectable obstacles. By addressing key limitations of the prior work, such as the dependence on human operators for relative localization and the inability to detect certain types of potentially hazardous obstacles, this research details the development of new methods

that may be applied to a variety of mobile robotics tasks.

1.2 Thesis Problem Statement and Goals

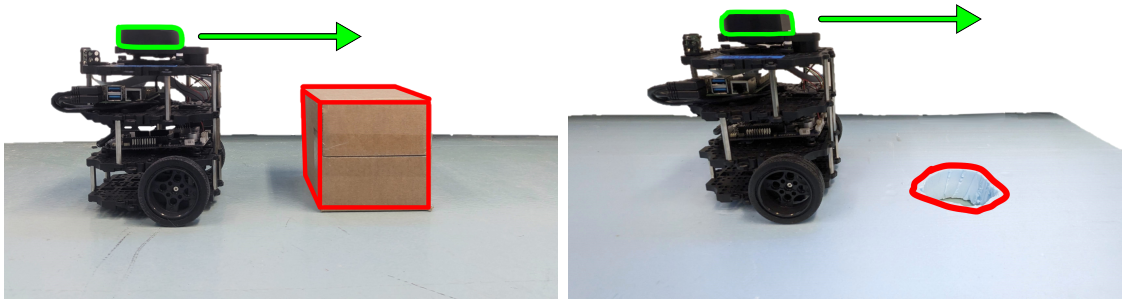
The problem addressed in this thesis is the development of new methods to increase the autonomy of a multi-UGV system in undertaking an exploration and mapping task through the introduction of a UAV. It can also be applied more generally to UGVs doing other tasks which require navigation, such as environmental data collection or mobile manipulation. This main objective was divided into smaller goals listed below, followed by their detailed explanations.

- Determine a novel method to autonomously localize multiple UGVs using UAV vision data so that they share a global map.
- Determine a novel method to use UAV imagery to detect obstacles and place them on the global map so that they may be avoided by UGVs.
- Implement the above methods through development on a programming platform.
- Design experiments to test the developed methods and execute the designed tests.

For a team of autonomous robots to be able to coordinate tasks in a mission, they must first each know where the others are located. The goal of this research is to employ a UAV in localizing multiple UGVs. Several approaches exist in the literature, however, most fail to meet the required accuracy, can only localize one UGV, or do not operate in real-time. In the new method, the UAV should be able to localize multiple UGVs at the start of the mission and, from there, UGV wheel odometry and Light Detection and Ranging (LiDAR) scans should be used to update their poses.

The desired result is that all robots should share a global map and be able to perform coordinated exploration.

The second goal relates to the use of the UAV's real-time vision data to detect obstacles that may be undetectable to the UGVs and to share the obstacle locations with the UGVs for avoidance. UGVs equipped with a two-dimensional (2D) LiDAR are not able to sense objects that are above or below the height of their LiDAR and are susceptible to collisions and falling into holes, as illustrated in Fig 1.1. Existing methods to accomplish this found in the literature had several drawbacks, such as the requirement for obstacles to be of a certain shape or colour, compatibility only with predefined obstacles, and the inability to operate in real-time. The new method should overcome these limitations.



(a) (b)

Figure 1.1: (a) UGV unable to detect obstacle. (b) UGV unable to detect hole.

Third, the two developed methods should be implemented with real-world robots through the creation of software programs. Codes are to be created using the Python and C++ programming languages to actualize the methods. Robot Operating System (ROS) was chosen as the software architecture to facilitate the implementation of the codes and the communication between the various robots.

Finally, experiments must be designed to test the functionality and performance of the developed methods. Various aspects of the methods should be evaluated through experimentation, including positional accuracy and success rate.

1.3 Summary of Contributions

The main contributions of this thesis are:

- Development of a real-time method to localize multiple UGVs labeled with unique ArUco markers using UAV vision data.
- Development of a real-time method to detect and map the locations of obstacles with UAV-collected three-dimensional (3D) pointcloud data, not reliant on predetermined obstacle features.
- Development of codes within a ROS framework to facilitate the experimentation of the new methods. These can be found at: `https://github.com/noorbot/mars-quadcopter/`
- Experimental testing of the new methods.
- Analysis of the results pertaining to the performance of the new methods.

1.4 Thesis Outline

The rest of the thesis is presented as follows: Chapter 2 is a literature review of the current state-of-the-art in UAV-UGV collaboration and existing methods. Chapters 3 and 4 discuss the development, implementation, and experimental results of the novel method created for multi-UGV localization using UAV data. Chapters 5 and 6 discuss the development, implementation, and experimental results of the novel method created for UAV obstacle detection and mapping for UGV navigation. Chapter 7 concludes the findings of this thesis and suggests future work for the development of this research.

Chapter 2

Literature Review

This chapter contains a review of the state-of-the-art concerning the central topics relating to the research work. The following sections discuss UAV/UGV collaboration, multi-robot exploration and mapping, localization of UGVs by a UAV, and aerial obstacle mapping for UGVs.

2.1 UAV/UGV Robot Collaboration

The collaboration of heterogeneous robots allows for the robots' strengths to be combined, surpassing their individual capabilities. The mix of UAVs and UGVs is particularly compelling due to their complementary strengths and weaknesses. Characteristically, UGVs are physically stable, can carry high payloads, and have long battery life. However, a UGV's ability to traverse an area in its entirety may be limited by environmental obstacles, limiting its scope for gathering sensory information. On the other hand, UAVs are able to travel through areas more freely and see from a higher vantage point, but they lack the ability to carry large payloads and fly for long durations of time. Due to the complementary nature of these robots' abilities, there has been considerable research in leveraging their combined potential over the past two

decades.

Applications of mixed UAV/UGV systems have been developed for various industries that can benefit from automation. These systems have seen use on agricultural [3], manufacturing [4], and construction sites [5]. While the majority of implementations involve data collection and site inspection, many perform tasks such as the transportation of objects [6] and even search and rescue missions [7].

2.1.1 Classifications of UAV/UGV Missions

In their proposed system for unified classification, Chen et al. [8] propose a taxonomy of mixed UAV/UAV systems along three axes as shown in Fig. 2.1. The Functional Role axis is split into Same Role and Different Role. The four functional roles are defined as: mobile sensor, mobile actuator, decision maker, and auxiliary facility. A 2021 review by Ding et al. [9] found that by far, the combination of different functional roles was more common than same functional roles. The second axis is Task Goal, which is divided into Coupled Goal and Decoupled Goal. The coupled task goal scenario occurs when both kinds of vehicles are working together and their actions must be tightly coordinated. In other cases, tasks may be decoupled if the shared task can be broken down and decoupled into different sub-goals. Finally, the Decision-Making axis is divided into three classifications: Centralized, Decentralized, and Hybrid Decision-Making. These distinctions are concerned with where the bulk of the computations take place, whether it is in one central vehicle or amongst the team of robots. Chen et al. advise that centralized decision-making is more suitable for small-scale systems and decentralized decision-making is beneficial for large-scale systems due to its increased robustness and scalability. Hybrid approaches can be employed by combining aspects of both methods.

Caska and Gayrette [10] have identified three main types of missions that collaborative UAV/UGV systems are being developed for. The first is for intelligence, surveillance,

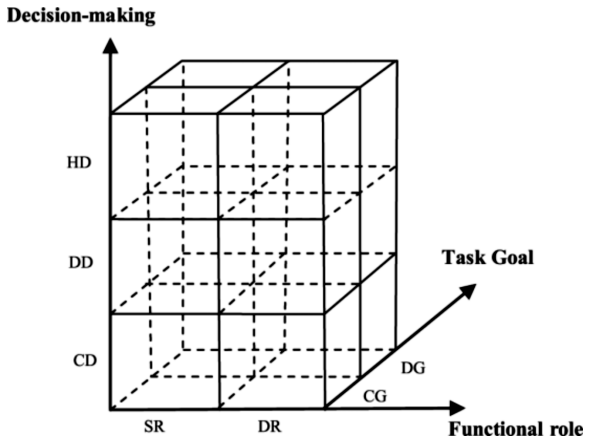


Figure 2.1: Three-axis taxonomy representation [8].

and reconnaissance. The second is object tracking, path planning, and localization. The final type of mission is formation control.

The majority of applications of UAV/UGV collaboration explored for this review use the UAV as a flying sensor to aid the UGV [5,6,11–21]. This leverages the large field of view that aerial vehicles possess. In such missions, the UAV often provides the UGV with environmental information in order for it to complete its task. Although it is often possible for UGVs to navigate unknown environments using only their own sensory information, the addition of a flying sensor UAV can bring advantages. The inclusion of a UAV to aid in the collection of data for path planning has been shown to improve the performance of the UGV [22]. For information-gathering missions, UAVs can gather data that UGVs cannot, due to their more flexible vantage point. They can also travel through cluttered areas that are inaccessible to UGVs.

Another common configuration for collaborative UAV/UGV operations is the use of the ground vehicle as a platform for the aerial vehicle. In order to conserve the limited battery lives of UAVs, some teams have used UGVs to transport UAVs to areas of interest. From there, a UAV may take off and perform its task. Later, a UGV can act as a landing pad. To accomplish this, the UAV must be able to track the position of the UGV accurately enough to land on it. This configuration strongly plays to the strengths of both robot types and has seen several successful

implementations [3, 23–26].

While many mixed UAV/UGV systems employ the UAV as a mobile sensor, it is also possible for aerial robots to interact with their surroundings. Aerial manipulation includes activities wherein a UAV carries objects, presses buttons, or otherwise interacts with its environment to reach an objective. This can expand the potential applications of UAVs to assembly, construction, agriculture, and payload transportation tasks. A study by Miki et al. [27] made innovative use of aerial manipulation to allow a UAV to aid in the transportation of a UGV up a steep cliff. This was accomplished with the use of a tether tied between the two robots. The UAV flew to the top of the cliff and anchored the tether, then the UGV used a winch to climb up, as depicted in Fig. 2.2.

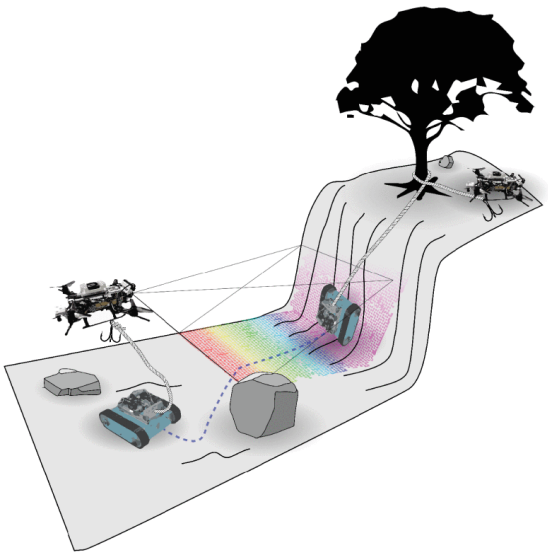


Figure 2.2: UAV/UGV collaboration by tether [27].

.

2.1.2 Real-time Collaboration

Many implementations of the flying sensor concept involve the UAV supplying the UGV with information in real-time [5, 6, 17, 26–29], while others first deploy the UAV to conduct a survey of the environment and then bring the information back to a

ground station [11, 30]. There, the data is processed for UGV path planning. The latter approach is often taken to overcome the robots' limited processing power or when reliable communication between the robots is not possible. A disadvantage of this method is that changes in the environment after the UAV survey is conducted are not captured and may affect the UGV's ability to navigate the area.

2.1.3 Control Architectures

In a multi-robot system, the flow of communication and control signals can be centralized, decentralized, or a hybrid combination of the two architectures.

Centralized approaches are systems wherein all robots report to and get instructions from one central agent. This central agent collects sensory information from each individual robot in the network and makes decisions for all members [8]. Since there is only one decision maker, it can be guaranteed to find the optimal solution [25]. With this setup, only one vehicle bears the computational load and, therefore, the remaining robots can be more simple, with reduced computational capabilities [25]. A stationary PC-based server may also be used as the central computer instead of a mobile robot. A drawback to centralized architectures is the reduced redundancy as compared to a decentralized multiple-robot system. If the central robot fails, the others are rendered unable to continue the mission. A second drawback is the limitation in scaling a centralized system to a large team of robots [8]. The greater the number of robots, the more load is put on the central agent when finding solutions and communicating them in real-time.

In a decentralized control system, each robot receives information about the current state of the other robots and calculates its own control signals [25]. Without a master robot, each vehicle is its own decision-maker and needs to facilitate its own computations. In this scheme, the robots must communicate with several robots instead of just one, resulting in the requirement for more complex communication protocols. This

architecture does not suffer from single-point-of-failure issues or scaling limitations, however, it loses the guarantee of an optimal solution [25].

Hybrid approaches combine the strengths of centralized and decentralized control architectures. The master robot can gather the states of all robots and make decisions, but each robot can complete the task independently and communicate with other robots. There are a multitude of ways this can be realized. One implementation of this is found in Hu et al.'s research [31], wherein individual robots have a certain autonomous ability to execute simple tasks in a decentralized manner, but a central server is used to deal with complex functions, such as image processing and Simultaneous Localization and Mapping (SLAM). Another example of a hybrid approach is a system wherein teams of UAVs and UGVs make their own decisions in a centralized way, then the central UAV and the central UGV share information to make a consensus decision [8].

2.2 Multi-Robot Exploration and Mapping

The goal of exploration and mapping algorithms is to allow a robot to autonomously navigate through an unmapped area and use its sensors to create an accurate map representation of the space, most often using SLAM algorithms. The creation of such maps can be beneficial for human use, but additionally, a map is a necessary precursor for most other autonomous missions involving navigation. Once a map is in place, a robot can be given a target and expected to find an optimal route to reach it.

Multiple robots may be used to reduce the time to complete exploration and to tackle the mapping of larger environments. However, the introduction of more than one robot brings challenges. First, it is desirable for the exploration to be coordinated between the robots, so that they do not all explore the same area, but rather split up to increase efficiency. Second, there must be a method to merge the robots' maps together to collect all of the information into a combined map. This introduces

another problem, which is that the robots must all share a global map, where all of their relative positions are known, so that their maps may be merged.

The basis of most modern autonomous exploration and mapping algorithms originates with the concept of frontier exploration, first developed by Yamauchi in 1997 [32]. He defined that for an autonomous robot, an area can be arranged as a grid where each cell is one of three classes: open, occupied, and unknown. This is commonly called an occupancy grid. Frontiers are defined as the boundaries between open space and unexplored space, as illustrated in Fig. 2.3. When a robot reaches a frontier, it gathers sensory information and reduces the unknown space, until eventually no frontiers exist and exploration is complete.

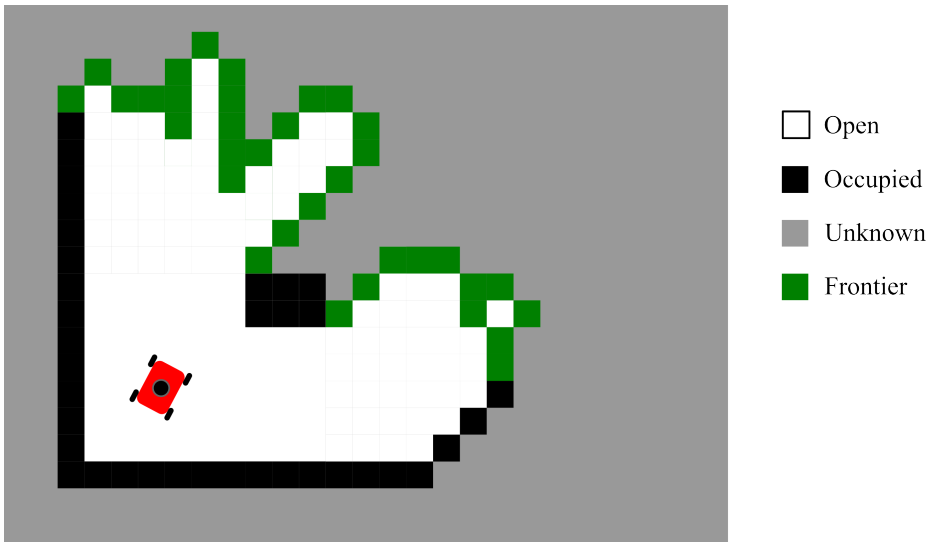


Figure 2.3: Occupancy grid representation of an environment. Frontiers exist at the boundaries between open and unknown spaces.

Shortly after the formulation of frontier exploration, Yamauchi saw the value of extending it with multi-robot exploration [33]. His method involves the robots broadcasting their maps to one another so that each robot has updated information, but decisions regarding which frontier to visit are made independently, leading to overlap. Since then, many researchers have proposed methods for more efficient multi-robot exploration. Some involve a central server that allocates the various robots to explore different areas [34, 35], while others use a distributed approach [36, 37].

It is common to assign robots to explore the frontiers nearest them, optimizing to reduce their travel distance. Some methods build on this concept by factoring frontiers' utility, to optimize for increases in information gain [38, 39]. In calculating the expected utility of a frontier, estimates are made regarding how many adjacent cells will become known by visiting it. Simmons et al. [40] established a system wherein robots place bids on frontiers, which are quantified by subtracting the travel cost from the utility. Then a central server assigns which frontier each robot should visit. Other researchers have added further complexity to the bidding concept to optimize multi-robot exploration [41, 42]. In scenarios where communication ranges limit the distances between robots, a role-based approach involving certain robots acting as explorers and others working to relay the information has been used [43].

The K-means clustering algorithm has also been seen in the literature as a method to delegate frontiers for multi-robot exploration and mapping [44–46]. Here, all frontiers are spatially grouped into clusters, with the number of clusters set equal to the number of robots. Robots are then assigned to visit a cluster, most often based on which is nearest. This process is iterated until no frontiers exist. Although [44–46] have shown improved efficiency in multi-robot exploration, their work is missing important considerations necessary to practically implement the exploration in real-time. They assume that each robot knows a global map and ignores the process of merging their maps. Goodwin [1] and Goodwin and Nokleby [2] implemented a map merging algorithm in tandem with the K-means frontier exploration approach. Fig. 2.4 demonstrates segmentation in the paths taken by three UGVs to explore and map a simulated environment using this method. A global map is established by manually entering the coordinates of each UGV into the program at the start of the mission. The work of this thesis aims to eliminate the manual coordinate entry and to increase the autonomy of the system. The poses of the UGVs will instead be found using a UAV.

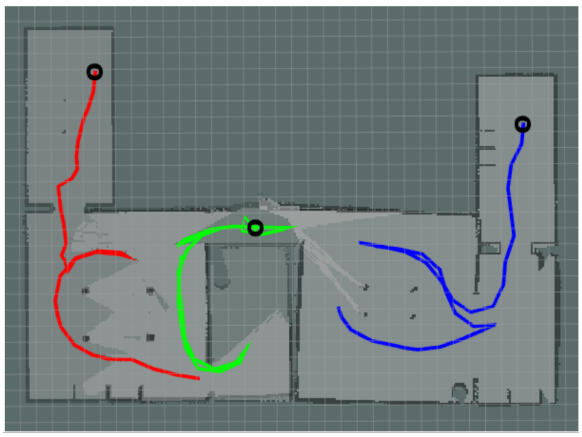


Figure 2.4: Multi-UGV frontier exploration using K-means assignment method [1].

2.3 Relative Localization for UAV/UGV Systems

A prerequisite for most autonomous multi-robot collaboration missions is the initialization of a global map, where each robot knows where the others are located. For multi-robot exploration and mapping, it is necessary for the robots to share a global map so that they can construct a merged map and coordinate exploration. A global map is also necessary for a UAV to share the locations of obstacles with UGVs. While GPS is a common tool for providing the positions of mobile robots, it may not be accessible in indoor environments and would not provide sufficient precision for most collaborative robot missions.

Various researchers have developed methods to tackle the problem of autonomously establishing a shared map between multiple robots. In systems with UAV/UGV collaboration, most of the methods involve one robot visually spotting the other and estimating their relative poses. One approach found in the literature requires UGVs to have a simple visual marker—a white rectangle with a black dot inside—that can be spotted with a UAV's camera [47]. A similar feature extraction approach was found in other works [6, 15].

Rather than creating their own unique visual markers, some researchers use popular marker libraries such as ArUco [20], AprilTag [48], and Whycon [5]. In [20], the ArUco marker mounted to the UGV must be visible to the UAV at all times for localization. In [48], the AprilTag mounted on the UGV is used to update its position each time it executes a movement in order to compensate for drift in wheel odometry. In [5], the Whycon marker is affixed to the blimp UAV and must be spotted by the UGV's camera for relative localization.

Faessler et al. [49] use a set of four infrared LEDs mounted to the UAV which can be detected from the ground by a UGV equipped with an infrared-pass filtered camera. This method boasts achieving relative position within a few centimeters of error—depending on the distance between the robots—wherein the localization is used as feedback to stabilize the UAV's flight. Another approach sees the UGV equipped with a laser pointer that projects a pattern onto the ceiling, which is spotted by a UAV to calculate the robots' relative poses [50].

Methods not reliant on the robots visually spotting one another were also found in the literature. In the work of Fankhauser et al. [12], a UAV performs 3D SLAM and then a UGV localizes itself by comparing its sensor readings to the SLAM data. A similar approach is seen in [18]. A different method involves the use of real-time kinematics (RTK) GPS technology used to find the positions of all robots relative to a base station [17, 51]. Guo et al. [52] have developed a method that uses ultra-wideband technology to determine the relative positions of multiple UAVs for formation control.

All streams of approaches have their drawbacks. Visual-based methods suffer from line-of-sight problems, vulnerability in non-ideal lighting conditions, and heavy computations. Methods using the comparison of SLAM models do not work well in feature-less hallways and open environments without nearby distinct walls. The RTK GPS and ultra-wideband methods provide a sub-meter level of precision that would not be sufficient for the proposed collaboration scenario.

Upon review of the existing methods for UAV/UGV localization, it was determined

that the use of a unique marker mounted to the UGV would be most suitable for achieving the goals of this research. This approach allows for simple implementation, only requiring a downwards-facing camera attached to the UAV and no additional sensors on the UGV. ArUco markers are two-dimensional images composed of a 5 by 5 grid of white and black squares. When viewed by a camera, the relative pose of a marker can be calculated with minimal computational costs using existing software libraries [53, 54]. The aim of this research work requires positional accuracy within 10 cm to enable collaborative mapping and the sharing of obstacle information. Researchers have implemented ArUco markers to study displacements less than 1 mm to measure the vibrations of a structure, proving their potential for high accuracy [55].

The implementation of ArUco markers will differ from that seen in [20, 48], however. These methods require the UAV to track the UGV and provide constant position feedback. This would not be possible in a single-UAV multi-UGV mission scenario, as is proposed here. Instead, the ArUco markers will be used to merely initialize the global map at the very start of the mission, providing the robots with their relative poses. Then, as the UGVs roam, their poses will be updated using a combination of wheel odometry data and the matching of their LiDAR scans.

2.4 UAV Obstacle Mapping for UGVs

A great number of approaches have been developed for the purpose of identifying obstacles and their locations from aerial image data and communicating them with UGVs so that they may be avoided. Existing methods have been found to fit within three categories: those that use aerial sensor data to create an elevation map, those that perform semantic segmentation to classify terrain, and those that capture obstacles as discrete objects.

2.4.1 Elevation Model Methods

Aerial sensor data can be used to determine elevation levels and slopes, which can in turn be used to create traversability maps for UGVs. It has been demonstrated that UAVs can be used to create accurate 3D topographic maps for all types of outdoor environments, such as steep rocky slopes [56], creeks [57], and forests [58], and indoor environments [59]. A Digital Elevation Model (DEM) is a representation of an environment that assigns a height value for each square of a 2D grid. A DEM can be used to find paths that avoid rapid changes in elevation, thus providing a UGV with an obstacle-free path.

Structure-from-motion (SFM) is a computer vision and image analysis technique that uses the known camera motion between frames and aligns these frames to make 3D estimates [60]. This technique is used in the work of Fankhauser et al. [12] to use data from a UAV's monocular camera to produce depth data, which is subsequently used to create a traversability map for a legged UGV. Another approach from Miki et al. [27] sees the use of a UAV's time-of-flight distance sensor to capture elevation information and build a DEM, demonstrated in Fig. 2.5.

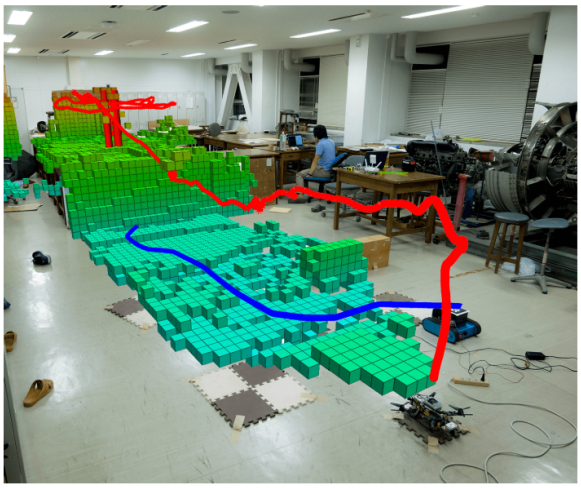


Figure 2.5: Elevation model approach for UAV obstacle detection. The elevation information is used to assess UGV traversability [27].

2.4.2 Semantic Segmentation Methods

Semantic segmentation refers to a specific type of machine learning model that uses a Convolution Neural Network (CNN) to classify each pixel within an image [61]. UGVs can use terrain classifications such as *pavement* and *grass* to compute optimal paths within an environment. An early example of this technique is found in [13], where overhead imagery from an unmanned helicopter is used to label terrain and aid a UGV's navigation through a dense environment. Semantic segmentation has also been applied to UGV image data, as in the work of Asadi et al. [5], where it was simply classified as *ground* and *not ground* to ensure safe navigation. A focus of the work was to reduce the computational load of the segmentation algorithm as to allow it to be executed in real-time. The method developed by Christie et al. [11] combines elevation modeling with semantic segmentation by considering the data from the produced DEM to inform the terrain classification algorithm, resulting in more accurate labeling. In their work, a large outdoor area was classified with labels such as *grass, vegetation, road,* and *building* from monocular aerial imagery, as shown in Fig 2.6. However, the image processing took place at a ground station in-between the UAV flight and the UGV mission rather than in real-time. Conversely, a more simple implementation was found in [51], where pixels of 2D UAV images were labeled purely based on colour, resulting in less computationally demanding obstacle detection, but with reduced robustness to unexpected objects and changing lighting conditions. After the segmented map is created, it is commonly used to generate a costmap to inform UGV navigation [11, 13, 51].

2.4.3 Discrete Obstacle Detection Methods

The final category of methods that facilitate UAV obstacle mapping for UGV obstacle avoidance requires the detection of discrete obstacles and their geo-referenced locations. Mueggler at al. [48] propose a method wherein the obstacles must be labeled

Figure 2.6: Semantic segmentation applied to aerial image data [11].

with ArUco markers for detection to take place. After an autonomous UAV survey of the premises, mission planning takes place to find the optimal path for the UGV to reach a target. In another work, obstacles of a distinct colour were recognized from UAV monocular camera images and their locations where recorded [14]. More sophisticated approaches to obstacle detection that rely on computer vision-enabled feature extraction were developed [15, 17]. However, both works rely on obstacles to be of predetermined shapes.

It was identified that there is a gap in existing approaches for discrete obstacle detection that do not require obstacles to have a certain shape, colour, or distinct marking. This presents an interesting opportunity to explore the use of depth data for the detection of individual objects as obstacles. The proposed method to use depth data for discrete obstacle detection is different from the elevation model methods, because rather than building a DEM, the elevation data is only used to identify the existence of an obstacle and not its height. This results in less information required to be communicated with the UGVs.

2.5 Summary

The literature shows the advantages of UAV/UGV collaboration to overcome individual limitations. Collaborative systems commonly employ UAVs as flying sensors to aid UGVs in data collection and path planning. Control architectures vary from

centralized, decentralized, to hybrid models, each with trade-offs in redundancy, scalability, and computational complexity. Real-time collaboration and dynamic control architectures offer promising avenues for enhancing multi-robot systems' performance and autonomy.

Autonomous exploration and mapping techniques enable robots to autonomously navigate unknown areas and generate accurate map representations using sensor data. Frontier-based exploration methods use occupancy grids to find still unexplored areas. This concept has been extended to multi-robot exploration to decrease mission times and allow for the mapping of larger areas. Various coordination strategies have been proposed to optimize exploration efficiency and avoid redundant coverage. Notably, the K-means clustering algorithm has been employed to assign frontiers to robots. However, existing methods overlook real-time implementation challenges, such as map merging. Goodwin [1] and Goodwin and Nokleby [2] addressed this limitation by integrating a map merging algorithm with K-means frontier exploration. The present thesis seeks to enhance system autonomy by automating UGV pose detection using a UAV, eliminating the need for manual coordinate input.

Various methods have been proposed for autonomous UAV/UGV localization, predominantly relying on visual data to facilitate relative pose estimation. Approaches range from feature extraction techniques to the use of marker libraries such as ArUco and AprilTag. Some methods utilize infrared LEDs or laser pointers for detection. Alternatively, non-visual techniques include 3D SLAM, RTK GPS, and ultra-wideband technology. Among these methods, the use of ArUco markers emerges as a promising solution due to its simplicity, low computational cost, and high accuracy potential. Unlike previous implementations, where constant tracking is required, ArUco markers in this research serve to only initialize the global map at mission start. Subsequent UGV pose updates are achieved through wheel odometry and LiDAR scan matching.

Methods for obstacle detection from aerial data for UGV navigation were found to fall into elevation modeling, semantic segmentation, and discrete obstacle detection

approaches. Elevation model techniques derive depth data from UAV imagery to create traversability maps for UGV path planning. Semantic segmentation employs CNNs to classify terrain types to aid UGV navigation. Discrete obstacle detection methods focus on identifying individual obstacles. However, existing methods reveal limitations in their requirement for predetermined characteristics for detection. It was identified that a new discrete obstacle detection method that uses depth data presents a promising avenue for improving UGV navigation among obstacles, regardless of obstacle features.

Chapter 3

UAV-UGV Localization Method

This chapter describes the method that was developed for providing relative localization in a multi-UGV system using UAV sensor data. First, the proposed framework is presented. Next, the robot hardware is described, as well as the open-source packages used. The UAV/UGV localization method and its software implementation are detailed. Parts of this methodology are also presented in [62].

The main concept behind the novel UAV/UGV localization method is that ArUco markers are to be mounted to the UGVs, which get detected by the UAV's camera as it surveys the area. Fig. 3.1 shows the main transformations involved. Once the UAV's camera spots the ArUco marker, the pose of the UGV is calculated and a new transform (shown in red) is published between the global reference frame, 'Map', and the UGV. The implementation involves the configuration of several existing open-source software libraries as well as a novel algorithm for UGV localization.

3.1 Framework Overview

This work uses ROS to facilitate communication between the various agents in the system over a network. ROS was first created in 2009 by Quigley et al. [63] as a

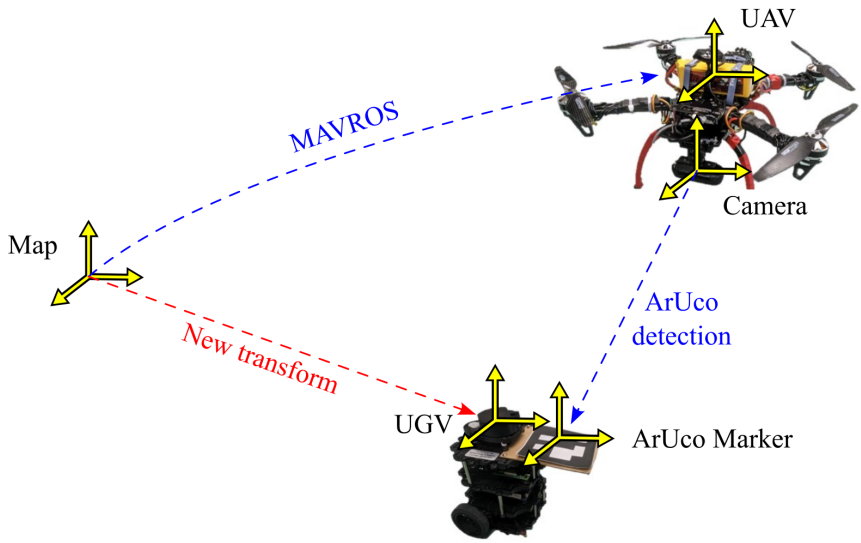


Figure 3.1: Transform diagram for robot localization.

general-use robot communication protocol that has been adopted by both industry and academia. In a ROS framework, agents publish 'nodes' to the network in order to communicate with one another. Agents can subscribe to nodes to access their data as needed.

Fig. 3.2 provides an overview of the framework developed for the UAV/UGV localization method. Under each agent, the boxes represent nodes and the arrows represent the interactions between the nodes. The method involves modifications made to existing open-source nodes, the development of a custom node for UGV localization, and the connections formed between the nodes.

A centralized approach is used, where the ROS server, run on a remote PC, handles the decision-making and more complex algorithms, while the mobile robots handle only low-level computations. The UAV manages the computations related to its own flight, which uses the MAVROS library [64], as well as running the camera. The MAVROS node takes in motion capture information from an external vision system as well as its own IMU sensors in order to form a position estimate. The position estimate is necessary to know the position of the UAV and thus, its camera. The centralized server subscribes to the camera node in order to use the data to run the

ArUco detection node. The UGV localization node was created by the author to take the ArUco information and process it to estimate the UGVs' poses and communicate them appropriately to enable multi-UGV navigation and mapping. Some nodes need a new instance for each UGV, as indicated by the 'Robot_x' prefix. The UGVs run a node that manages their basic operation and publishes sensor information, transformations, and more. This information is used by the Gmapping node, which uses the laserscan data to create an occupancy grid to represent the environment. The UGV node subscribes to the Navigation node for velocity commands and motor power. One instance of the map merging node is run by the centralized server to combine the UGVs' maps together. This map can later be used for the multi-UGV frontier exploration, which is not included here.

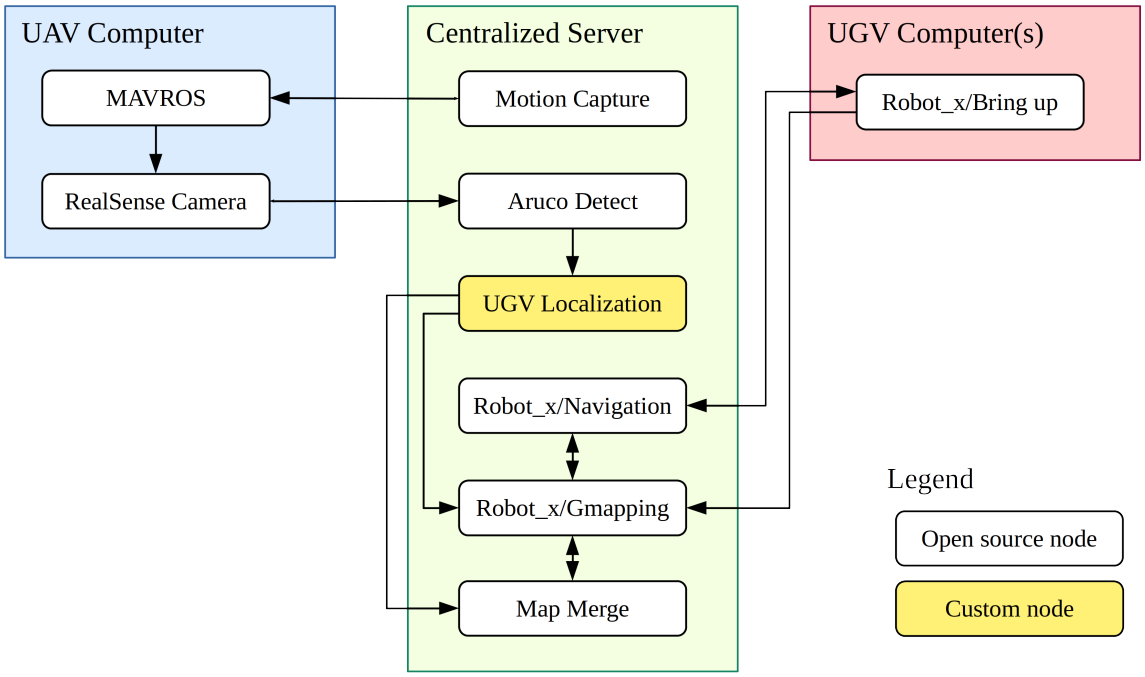


Figure 3.2: Overview of framework nodes.

3.2 The Robots

3.2.1 UAV

A quadcopter UAV was assembled as shown in Fig. 3.3. A custom-built quadcopter was used because it was important that it could be modified to meet the research needs and to communicate with the ROS network. The full parts list is shown in Table 3.1.

Figure 3.3: Custom-built quadcopter.

Table 3.1: Custom-built quadcopter parts.

Description	Model	Notes
Flight controller	Pixhawk 4	
On-board computer	UP core	4GB RAM
Camera	Intel RealSense D435i	
4x Motor	Quanum MT	Series 4108 475KV
4x Electronic speed controller	Hornet ESC	6S 40A
4x Propeller	Quanum Carbon Fiber	11in len., 5.5in pitch
Frame	Diatone Whitesheep	
Power module	Pixhawk PM07	
Voltage regulator for computer	Valefod Buck Converter	5A
Battery	Turnigy Graphene Panther	1200mAh 6S 75C
RC receiver	Frsky 8XR	2.4GHz
Handheld remote controller	Turnigy 9XR Pro	
Active markers	BlackTrax	

The quadcopter uses a Pixhawk 4 (PX4) flight controller to manage its flight, which

can be done both through remote control and autonomously. The UAV also has an onboard computer that communicates with the ROS network and enables autonomous flight. In order to safely fly the quadcopter indoors, the robot must know its position at all times. This is achieved with the use of an OptiTrack external motion capture system comprised of fourteen cameras. The quadcopter is outfitted with three Black-Trax active markers that are tracked by the cameras. This information, as well as IMU sensor data, is used by the PX4 to compute a local position estimate.

A communication bridge must be established between the ROS server and the UAV's flight controller, the PX4. This is done using an open-source package called MAVROS [64]. MAVROS is used to bridge ROS and the MAVLink protocol, which is used by the flight controller. This is what allows the onboard computer, which runs ROS, to communicate the external motion capture system data with the flight controller, and receive back a position estimate. MAVROS is also necessary so that autonomous flight commands can be sent from the ROS server to the flight controller, which then controls the UAV's motor speeds. In this work, coordinates are provided to the UAV for it to perform its autonomous flight to survey the environment. For real-world implementation, it is assumed that the UAV has a guidance, navigation, and control sub-system that enables it to autonomously survey the area.

The other software executed on the UAV's on-board computer is for its camera. Intel provides a ROS package for the RealSense camera so that its data can be published as a message over the ROS server [65]. The stream of 2D images is subscribed to by the ArUco detection node. Additionally, the pointcloud data is also published which is used by the obstacle detection method in Chapter 5.

3.2.2 UGVs

The UGVs used in this work are the TurtleBot3 Burgers, shown in Fig. 3.4. The Turtlebot3 Burger is a small, open-source mobile robot made by ROBOTIS designed

to run on ROS [66]. Its notable components include a Raspberry Pi 4 microprocessor, a motor driver board, dual motors, an IMU, and a 2D LiDAR.

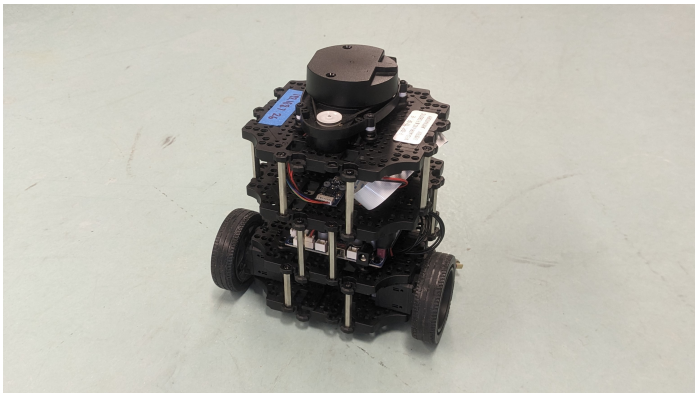


Figure 3.4: TurtleBot3 Burger.

The TurtleBot3 burger is a popular education and research tool, leading to accessibility to a vast amount of open-source software. The ROS node called 'Bring_up' initializes the robot on the ROS server and publishes information such as its battery state, odometry data, laserscan data, and transform frames.

Another key peice of software used by the UGVs is the Navigation Stack, which is a set of packages designed for motion and path planning [67]. The global path planner implements a variation of Dijikstra's algorithm [68] to find the optimal path to a goal point.

SLAM is implemented with the UGVs' laserscan data using the Gmapping software package [69]. It uses a probabilistic approach to determine the occupancy of each grid cell.

3.3 Open-Source Packages

This section details the open-source software packages that were modified and utilized in the creation of the novel framework.

3.3.1 ArUco Detection

'Aruco_detect' is a ROS package that finds ArUco markers in an image stream, locates their corner points, and estimates 3D transforms from the camera to the markers [70]. No modifications were made to this package apart from editing a parameter file to make it subscribe to the RealSense camera stream. The data gathered from the ArUco detection node gets processed to form the UGV pose estimate.

3.3.2 Transform Library (tf)

The 'tf' (transform) ROS package makes it possible to keep track of multiple coordinate frames over time and maintains relationships between coordinate frames [71]. It may be used both for a single mobile robot — to keep track of wheel locations, grippers, etc. — as well as for multiple agents. The tf library can broadcast a rigid transform, which is used to define the transform between the UAV's body and its camera. The transform broadcaster function was also used in the new UGV Localization node to describe the new transform necessary for the UGV pose.

Transform relationships can be stored in a tree structure to visualize their connections. The transform tree for the multi-robot system is explained in further detail in Section 3.4.3.

3.3.3 Map Merging

The 'multi_robot_map_merge' package created by Hörner [72] takes in multiple robots' maps and combines them into one global map. The package can be used in two ways. The first is if the initial poses of the UGVs are known, wherein it performs a simple stitching algorithm to merge the maps together based on the provided initial poses. This was used in Goodwin's [1] multi-robot exploration algorithm. The second option

is to state that the initial poses are unknown, wherein a feature-matching approach between the UGV laserscans is undertaken. If there are enough feature matches, the transform between the UGVs will be estimated to enable map merging.

The above methods for the UGV initial poses were found to be unsatisfactory. The first requires a person to manually enter the starting pose of each UGV relative to the others, while the second one is unreliable in most realistic scenarios, as it requires distinct features and the robots to all be near one another to have similar laserscans. It is for this reason that a new multi-UGV relative localization method using a UAV was created. In the proposed method, the UGV localization node sets the initial pose parameters to be used by the map merging node.

3.4 Relative Localization Method

This section details the UGV Localization node that was created specifically to take the ArUco marker pose data, calculate the UGVs' poses, and communicate the UGV poses appropriately to create a global map. An overview of the global map initialization is shown in Fig. 3.5 and a detailed flowchart of the novel UGV localization node is shown in Fig. 3.6.

3.4.1 Collection of the ArUco marker data

ArUco marker data is used to estimate the UGVs' initial poses. The ArUco detection node provides a transform showing the pose of the marker relative to the camera. To obtain the transform of the markers relative to the reference frame Map instead, known transforms can be used to make the calculation, as shown here:

$$ {}^{map}_{aruco}\mathbf{T} = {}^{map}_{UAV}\mathbf{T} \times {}^{UAV}_{camera}\mathbf{T} \times {}^{camera}_{aruco}\mathbf{T} \tag{3.1} $$

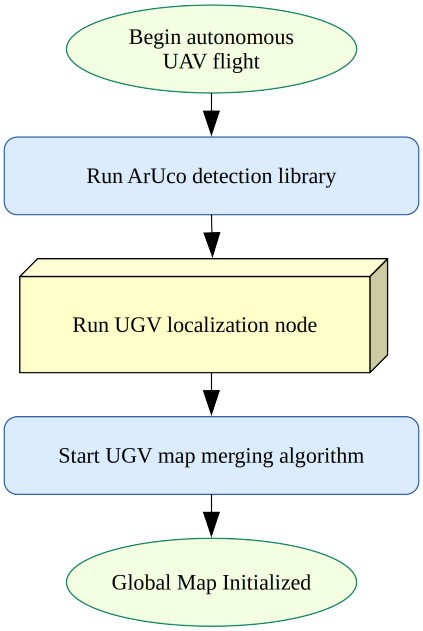


Figure 3.5: Overview flowchart of global map initialization.

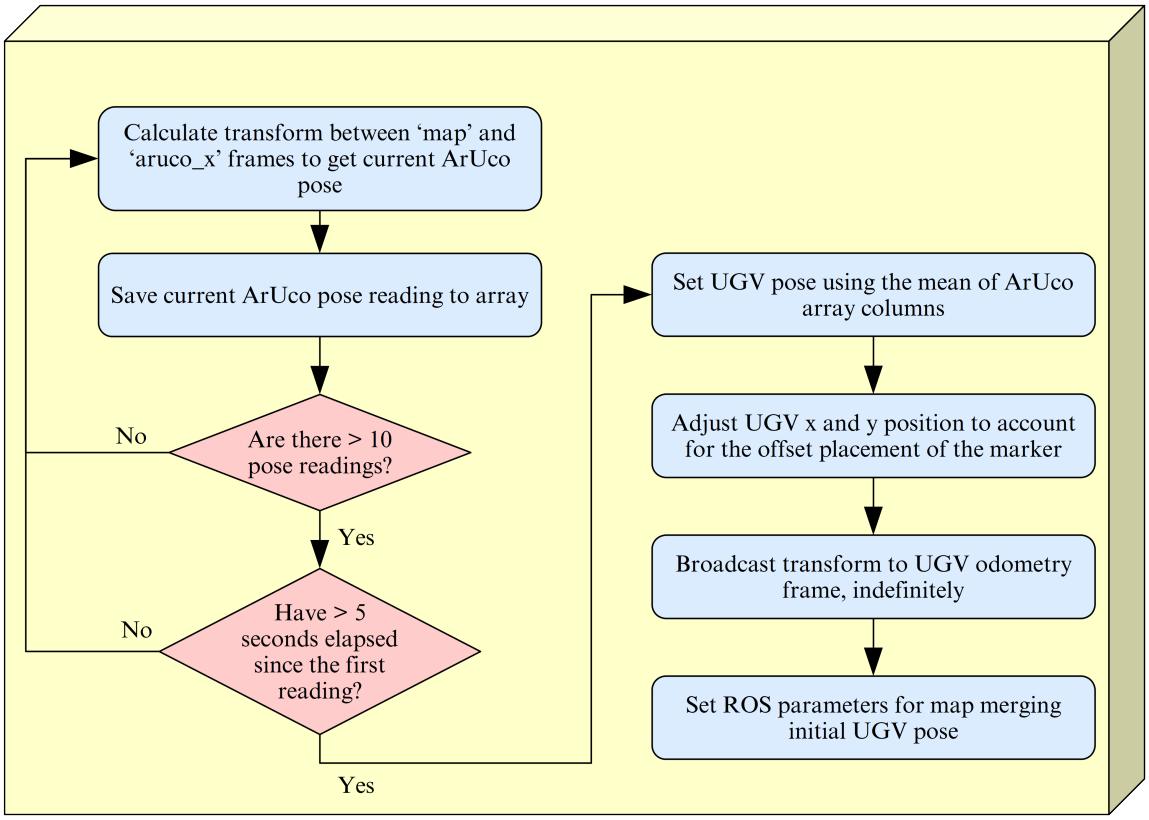


Figure 3.6: UGV localization node flowchart.

In order to increase the accuracy of the estimate, multiple ArUco pose readings are collected and the average is taken. To maximize the chances of accurate ArUco marker pose estimation in as many mission scenarios as possible, conditions have been made regarding when a sufficient number of readings have been collected for averaging. First, at least ten ArUco pose readings should be collected. Second, at least five seconds should elapse from the first detection instance. The rationale for the second condition is to ensure that once the ArUco marker enters the UAV camera's field of view, data will continue to be gathered for at least five seconds. This way, as the UAV moves in its continuous search path, readings from when it is directly over top of the ArUco marker, which are most accurate, will be factored in.

In the software implementation, the ArUco pose information is stored in an array, where columns store *x, y, z,* position and quaternion orientation data. The mean of each column is taken to estimate the ArUco marker's pose. This same process can be run simultaneously for multiple UGVs.

3.4.2 Calculation of UGV pose

Once the ArUco marker's pose has been estimated, a calculation can be made as to the UGV's pose. The offset position of the ArUco marker relative to the TurtleBot3 in the *x*-*y* plane must be accounted for. Fig. 3.7 shows the offset between the UGV and ArUco coordinate frames, which is 10 cm along its x-axis.

The UGV's pose relative to the map frame can be calculated as follows:

$$\begin{bmatrix} x \\ y \\ z \\ yaw \end{bmatrix} = \begin{bmatrix} aruco\_x \\ aruco\_y \\ aruco\_z \\ aruco\_yaw \end{bmatrix} + \begin{bmatrix} 0.1 \cdot \cos\left(aruco\_yaw\right) \\ 0.1 \cdot \sin\left(aruco\_yaw\right) \\ 0 \\ 0 \end{bmatrix} [m] \tag{3.2}$$

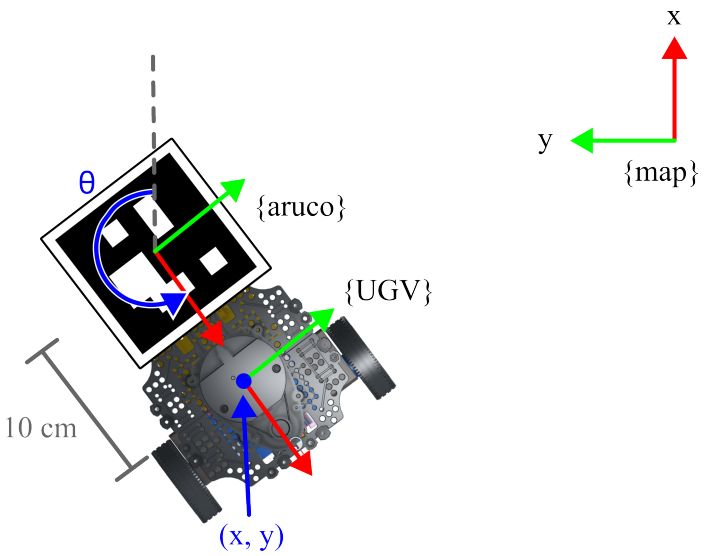


Figure 3.7: UGV pose diagram.

The psuedo code for the ArUco pose data collection, averaging, and UGV pose calculation is shown in Algorithm 1.

Algorithm 1 Process ArUco Pose Readings and Calculate UGV Pose

```
# Save ArUco marker pose readings to array
while not enoughArucoReadings do
    if currentArucoReading ≠ prevArucoReading then
        arucoArray ← currentArucoReading
        prevArucoReading ← currentArucoReading
    end if
    if length(arucoArray) > 10 and timeElapsed > 5s then
        enoughArucoReadings ← False
    end if
end while
# Take mean average of ArUco pose readings
ugvPose ← mean(arucoArray)
ugvPose.x ← ugvPose.x + 0.1 × cos(ugvPose.yaw) # Adjust for 10cm offset
ugvPose.y ← ugvPose.y + 0.1 × cos(ugvPose.yaw)
```

3.4.3 Publishing UGV Transform

In order for mobile robots to share a global map, their poses must all be connected to a shared reference frame. Previously, a rigid transform for the initial pose of each robot

was provided through manual input. In this work, this transform, which was obtained using the described method, is instead published autonomously. The transform tree, shown in Fig. 3.8, describes the relationships between the important frames in the multi-robot system. Once the camera spots an ArUco marker, the pose information is processed to provide an estimate of the UGV's initial pose. This information is then broadcast throughout the ROS server as a transform from the Robot_1/map frame to the robot's odometry frame, as shown in red. The odometry frame remains stationary, and as the UGV moves, its position relative to the odometry frame, which represents its starting position, is determined through a combination of odometry and laserscan data, as shown in 3.9. This means that the UAV must only determine the starting poses of the UGVs and thereafter their poses will always be known and connect back to the Map frame.

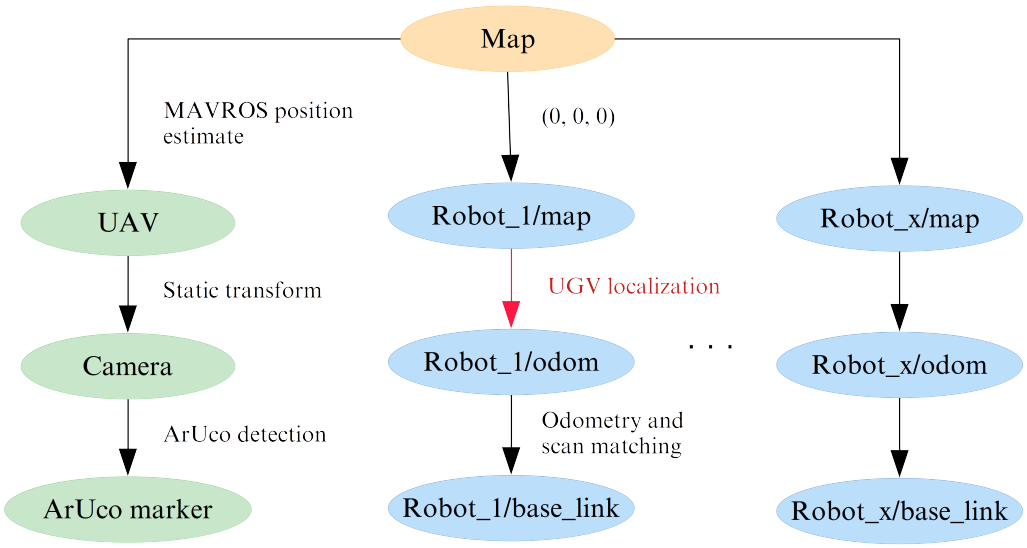


Figure 3.8: Transform tree representing the relationships between the multi-robot system coordinate frames. It should be noted that the Map and Robot_1/map frames have equivalent poses.

3.4.4 Setting Map Merging Parameters

To initialize map merging between multiple UGVs, requisite for the frontier exploration task to commence, the initial poses of each robot must be provided. This

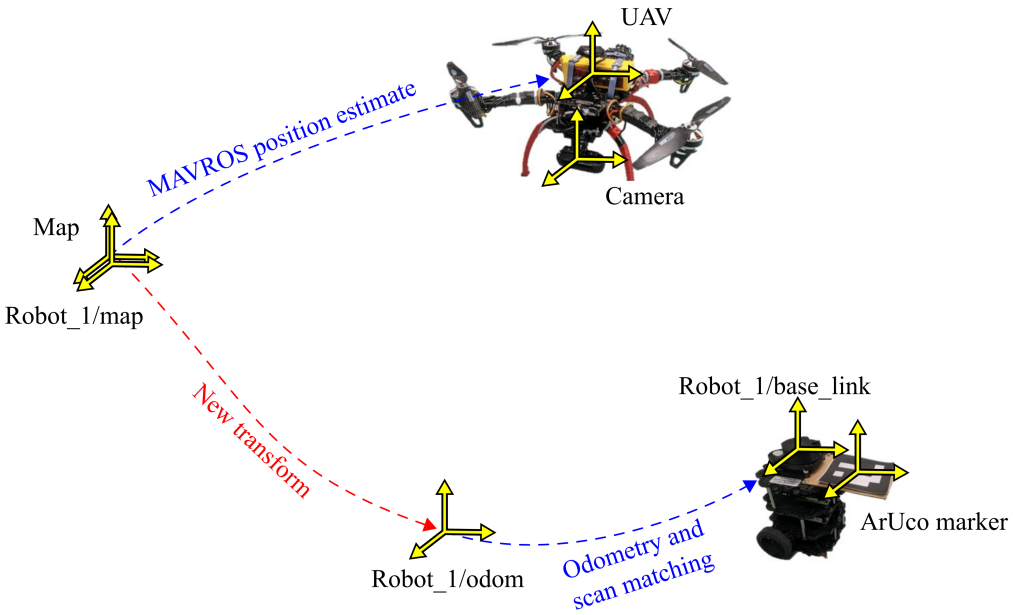


Figure 3.9: Visualization of transforms in the UAV/UGV system. The Map frame is the shared reference frame that all robots must connect to. It is shown here that a new transform is published to the UGV's odometry frame, which is the robot's initial pose. As the UGV roams, its pose is tracked relative to this starting pose using odometry and scan matching techniques.

allows the stitching algorithm to combine the maps. This was previously done by a user hard-coding the parameters [1, 2]. In this work, these parameters are set autonomously using the calculated UGV pose. This is implemented in the novel UGV Localization node by sending the updated pose parameters to the ROS server.

Chapter 4

UAV-UGV Localization Experiments

This chapter provides a review of the experiments undertaken to evaluate the new framework for UAV/UGV localization and their results. To produce maps that accurately depict the spaces they represent, it was decided that the maximum permissible position error be no more than 10 cm and the maximum permissible orientation error be no more than 5°.

4.1 UGV Pose Estimation

4.1.1 Pose Estimation Accuracy Experiments

The accuracy of the UGV pose estimates from the UGV/UGV localization method was evaluated through a series of experiments. The estimated UGV poses obtained from the UAV camera data were compared to ground-truth data, which was produced with an external motion capture system. The UAV was flown autonomously over top of the UGVs to collect pose data, as shown in Fig. 4.1. Reflective markers were placed

on the UGVs for tracking with the external vision system.

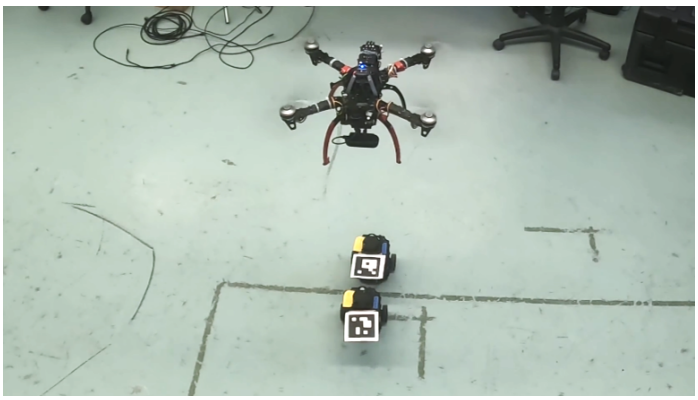


Figure 4.1: UAV flight over UGVs for pose data collection. The UAV moves autonomously in a pre-programmed flight pattern at a speed of 10 cm/s at a height of 1.25 m.

The UGV pose estimate obtained from the UAV's data takes the average of available readings across multiple frames. Fig. 4.2 shows a plot of the pose estimate of a single UGV in the x-y plane. From the twenty-five data points, the average is taken and then compared with the ground-truth estimate. The variability of the individual estimates between frames is caused in majority by the small delay in the UAV's position estimate as it moves. During this experiment, the UAV was moving along the y-axis, which resulted in significantly greater variability in the y-direction than the x-direction, as shown in Fig. 4.3. A trade-off exists between the UAV's translational speed and the accuracy of the estimates.

The accuracy of the UGV poses obtained with the new method was tested over ten runs. The results in Table 4.1 show an average planar error of 4.01 cm with a standard deviation of 1.96 cm. Planar error was calculated as the Euclidean distance between the true UGV position as measured by the motion capture system and the estimate in the x-y plane. The orientation estimate, which represents the UGV's yaw, had an average error of 1.72° with a standard deviation of 0.90°. All runs were within the defined limits to produce sufficiently accurate maps.

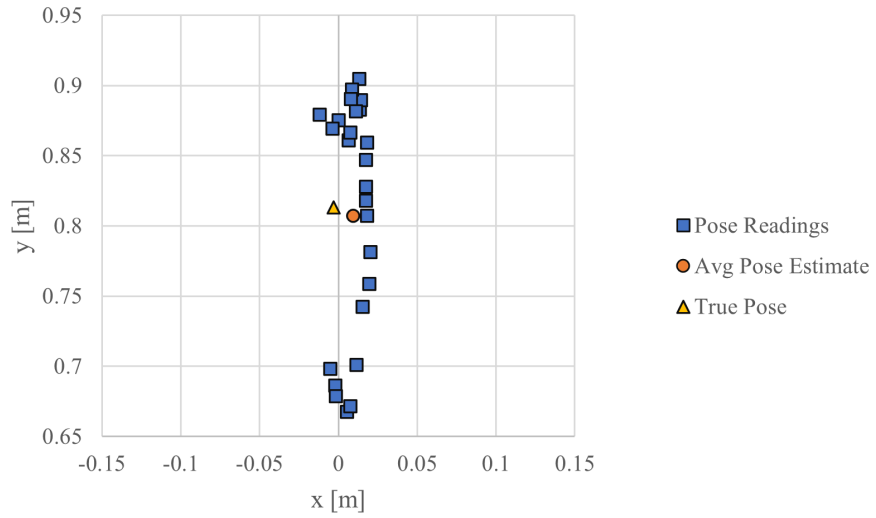


Figure 4.2: Experimental results of the estimation of a UGV's position in the x-y plane. The average of the pose estimates in the x and y directions is taken to form a final pose estimate, which can be compared to the true UGV pose. The planar error is 1.37 cm for this instance.

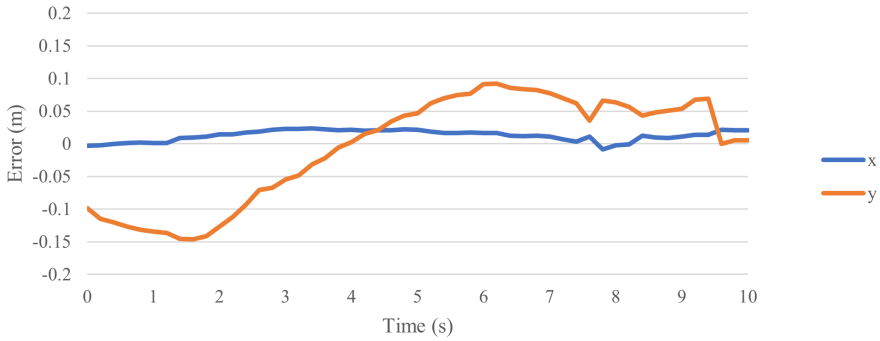


Figure 4.3: Error in UGV position readings over time. This data was collected during a UAV's flight along the y-axis, resulting in significant variability in positional error in the y-direction compared to the x-direction.

Table 4.1: Error in UGV pose estimate.

	Position estimate	Orientation estimate
Average error	4.01 cm	1.72°
Maximum error	5.45 cm	3.07°
Minimum error	1.37 cm	0.52°
Standard deviation	1.96 cm	0.90°

4.1.2 Pose Estimation Accuracy with Distance Experiment

The previous experiments all took place with the UAV flying at a height of 1.25 m above the ground. This was chosen to avoid the ground effect which causes unstable UAV flight in close proximity to rigid surfaces [73], while staying clear of the ceiling. It is, however, desirable to measure the accuracy of the UGV localization method at various distances to evaluate its accuracy in different scenarios. Thus, an experiment was conducted wherein the UGV position accuracy was measured while the UAV flew at increasing heights, from 1 m to 2.5 m. Six position estimates were taken at each height and compared to ground truth data produced by the external motion capture system. Fig. 4.4 shows the results from this experiment in a boxplot. The data shows that the further the camera is from the targets, the accuracy of the estimates decreases and the variance increases. Flight height and position estimate error are shown to have a linear relationship. Nonetheless, even at a flight height of 2.5 m, the maximum that could be achieved safely in the indoor test area, the position error was repeatedly under 10 cm and sufficient for multi-UGV map merging.

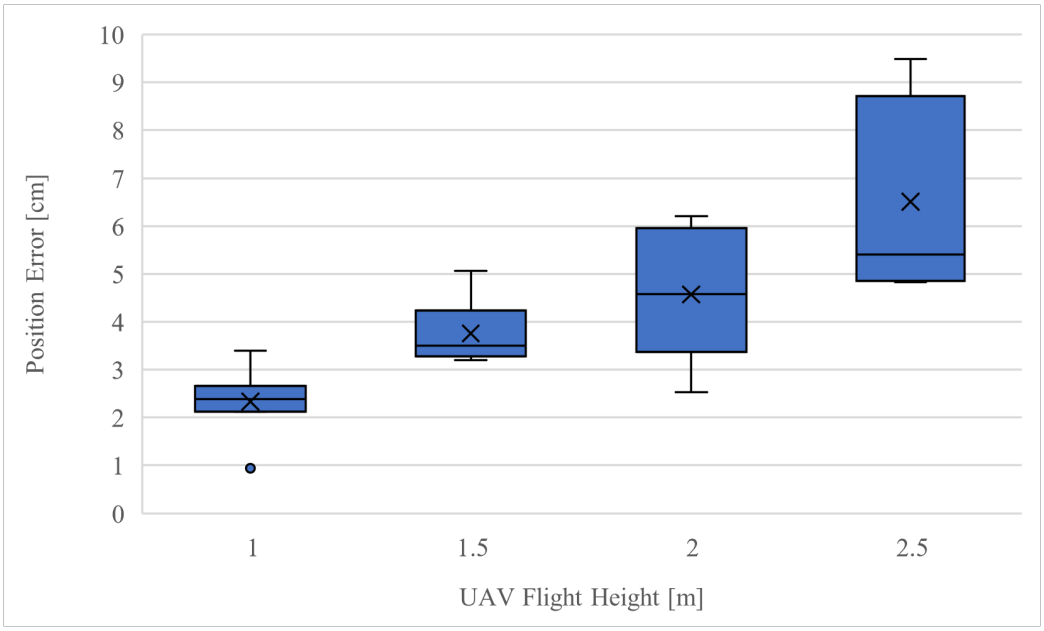


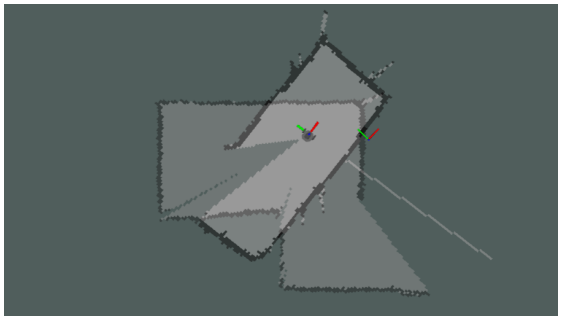
Figure 4.4: Boxplot of the position estimation error.

4.2 Global Map Initialization Experiment

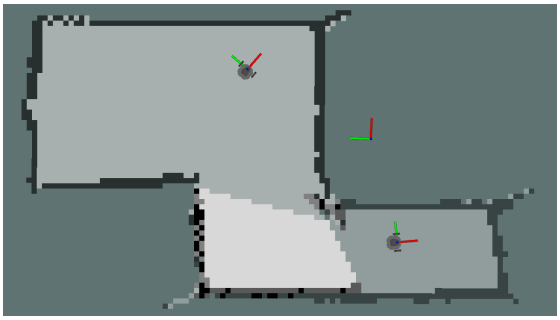
To validate the ability of the localization method to initialize a functional global map, experiments involving the merging of multiple UGV maps were undertaken. UGVs were arranged in various configurations within an environment with constructed walls, as shown in Fig. 4.5(a). At first, the UGVs were not aware of their poses and thus, their LiDAR scans showed a mismatch, as shown in Fig. 4.5(b). This is due to the fact that if the UGV poses are not set, they assume themselves to be at the (0, 0) position. Then, the UAV flew along a pre-programmed path to search the area for the UGVs. Upon the detection of their ArUco markers, the UGV Localization node sent their correct poses relative to the Map frame. The resulting visualization in Fig. 4.5(c) demonstrates successful localization of the robots, with correct positions and orientations and accurate LiDAR scan overlap that represents their environment. This created a properly merged map from which multi-robot exploration could be carried out.

The global map initialization experiment was repeated with several configurations of UGV positions and wall placement. Fig. 4.6 shows another configuration. Here, after the global map was initialized successfully, the UGVs navigated about the environment to map the rest of it. It is shown in Fig. 4.6(d) that the map initialization method allowed for the relative positions of the robots to be updated by the odometry and LiDAR scan data after their initial poses were recognized by the UAV. This allowed for map building that represented the environment with sufficient accuracy. It may be noted that the additional map data beyond the walls in Fig. 4.6(d) is caused by the ribbon-like material which the walls are made of being too narrow and the UGV LiDARs sometimes seeing above or below it.

(a)

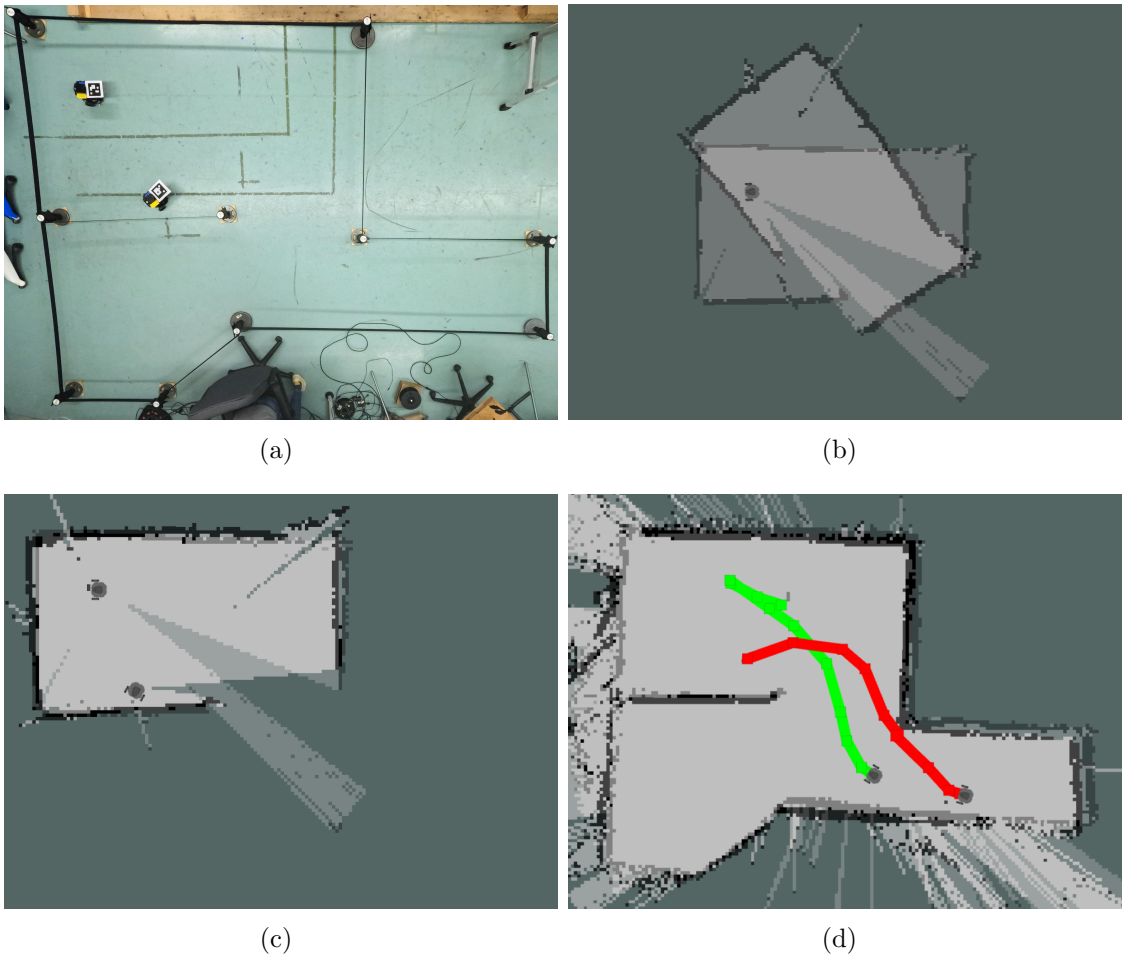


(b)



(c)

Figure 4.5: (a) Test set up for global map initialization experiment. (b) UGV LiDAR scans at mission start, showing mismatch. (c) UGV LiDAR scans after global map initialization using the proposed method.



(a) (b)

(c) (d)

Figure 4.6: (a) Experimental setup for global map initialization experiment. (b) UGV LiDAR scans at mission start, showing mismatch. (c) UGV LiDAR scans after global map initialization. (d) Map results after UGV traversal of the environment. The map adequately represents the area.

Chapter 5

UAV Obstacle Mapping Method

This chapter describes a new method created for obstacle detection via UAV data for the purpose of UGV navigation. First, an overview of the framework is provided. Next, the open-source packages used are described. The description of the method is divided into two parts; the obstacle detection method and the method for sharing obstacle information with UGVs to enable avoidance. Details regarding the software implementation of these methods are provided.

The main concept for the novel obstacle detection method is to process the 3D pointcloud data from the UAV's camera to isolate obstacles. Then, the obstacle pointcloud data is used as an input for the UGVs' navigation schemes.

5.1 Framework Overview

The processes that each agent in the multi-robot system runs are shown in Fig. 5.1. The diagram shows the addition of a new node for pointcloud processing to the existing framework. The configuration for the obstacle detection method was built on top of the UGV localization framework and, therefore, many of the same nodes are required. However, if one desired to use the obstacle detection method but not UGV

localization, it would be possible to omit the ArUco detection and UGV localization nodes.

The pointcloud data is taken from the camera and processed using the proposed method using the centralized server. It also takes information from each UGVs' navigation node which is used to access the current position of the UGVs. This information is important for the pointcloud processing to ensure that UGVs are not captured as obstacles. The resulting obstacle pointcloud is then sent to each robot's navigation node so that the obstacle data can be added to their costmaps. The costmap data is then used to inform the UGVs' path-planning, thus allowing for obstacle avoidance.

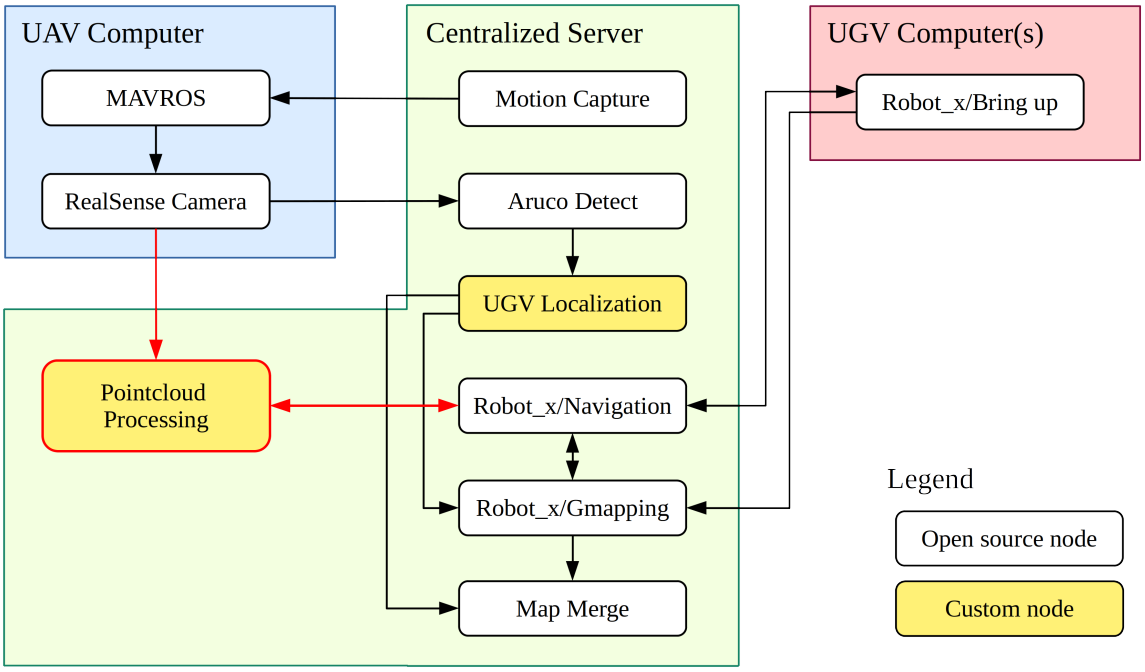


Figure 5.1: Overview of framework nodes. Updates from UAV/UGV localization framework are outlined in red.

5.2 Open-Source Packages

Software packages discussed in Chapter 3 such as MAVROS, RealSense Camera, the navigation stack, Gmapping, and the Turtlebot3 libraries are needed for the UAV obstacle detection and avoidance method, too.

5.2.1 Open3D Library

The Open3D Library is an open-source tool to simplify pointcloud manipulation [74]. The library has built-in functions for operations such as voxel-downsampling, transformations, Random Sample Consensus (RANSAC) plane segmentation, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and K-Nearest Neighbours (KNN), all of which are used for the pointcloud processing in this method.

5.3 UAV Obstacle Detection

The main goal of the new obstacle detection method is to enable a UAV to detect obstacles that may interfere with a UGV's ability to safely traverse an environment. This may include objects on the ground as well as holes. The approach developed requires the use of a downwards-facing depth camera mounted to the UAV. The pointcloud data produced by the sensor is processed so that only points belonging to obstacles remain. This processing occurs in real-time. The steps to process the pointcloud data to isolate obstacles are shown in Fig. 5.2 with a demonstration shown in Fig. 5.3.

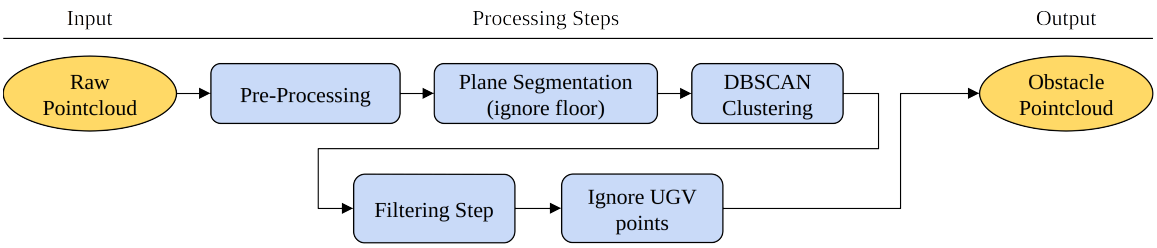


Figure 5.2: Pointcloud processing methodology to isolate obstacles.

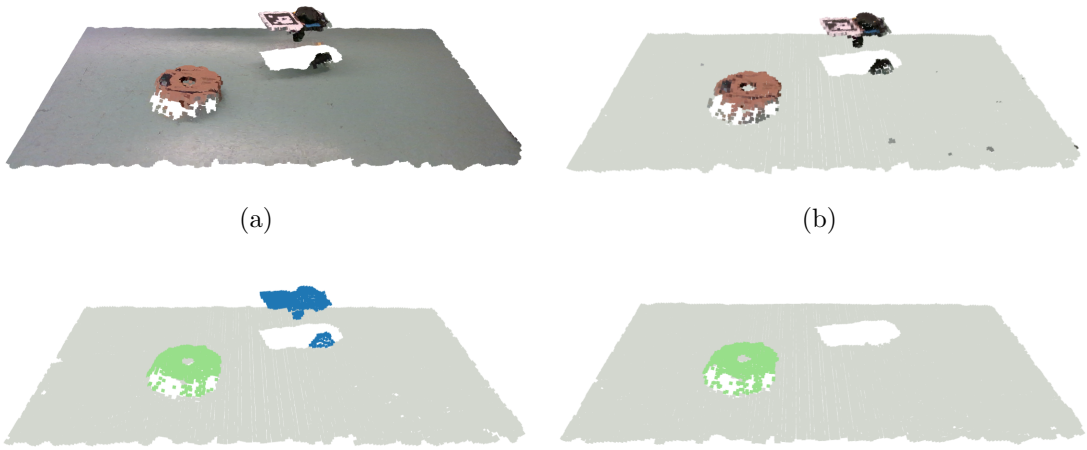


Figure 5.3: Demonstration of the novel obstacle detection method. (a) Raw pointcloud. (b) Pointcloud after plane segmentation. (c) Pointcloud after DBSCAN clustering and filtering step, showing removal of noise. (d) Final obstacle pointcloud after UGV points removal.

5.3.1 Pointcloud Pre-Processing

An Intel RealSense D435i camera is mounted to the UAV facing the ground. The RealSense Camera library is able to produce a pointcloud from the sensor's data in real-time and publish it to the ROS server. A pointcloud is a digital representation of three-dimensional space that consists of many discrete measurement points. Each point has a set of Cartesian coordinates describing its position and can also have a colour value. The method developed here takes the raw pointcloud data and processes it to isolate obstacles.

Pre-processing steps are applied to the pointcloud data to prepare it for obstacle

detection. The raw pointcloud data locates points about the Camera coordinate frame, rather than the global reference frame, Map. To be able to share obstacle positions with the UGVs, the pointcloud must be transformed to the shared global reference frame, as illustrated in Fig. 5.4. To achieve this, the current transform between the Map and Camera frames is calculated and then applied to the pointcloud.

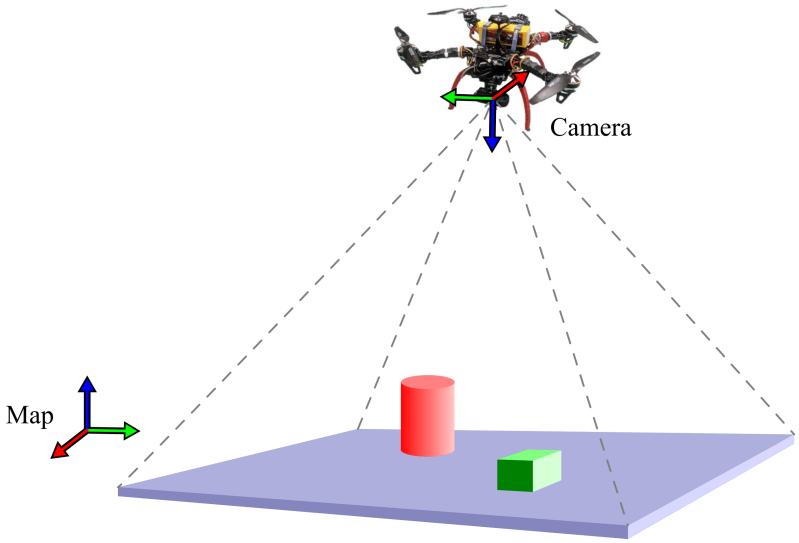


Figure 5.4: Pointcloud transformation.

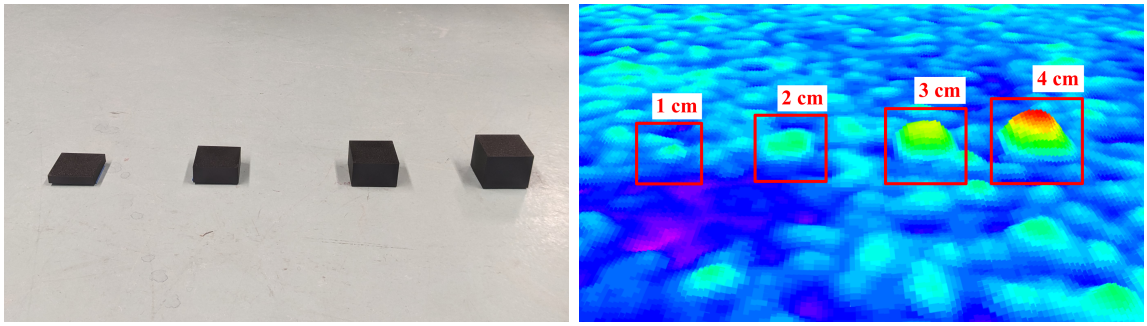
Next, the pointcloud data is put through voxel downsampling to decrease the computational costs of subsequent processing steps. Voxel downsampling divides a three-dimensional volume into cells and takes the average of all points in each cell to replace it with one point. Thus, the points are reduced uniformly into a 1 cm^3 grid.

5.3.2 Plane Segmentation

Points belonging to the ground plane can be removed from the pointcloud as they can be excluded from being considered obstacles. A RANSAC plane segmentation algorithm identifies all points corresponding with the flat ground. This algorithm finds the largest plane in a given pointcloud and classifies all points within a specified threshold as belonging to that plane. Such a threshold is required due to noise in

the pointcloud originating from sensor error. The way the threshold works is that it envelopes points within a specified range above and below the plane as part of the plane. A well-tuned threshold value optimizes for the ability to detect all objects outside of the ground plane, while not allowing for noise in ground points to be allowed to pass through.

To determine a suitable threshold value, the error in depth measurements was analyzed. With the camera pointed at the plain ground from a distance of 1.25 m, it was found that the range between the highest point and the lowest point is 6.6 cm, on average. Fig.5.5 shows the raw pointcloud output when viewing a set of short obstacles, ranging from 1 cm to 4 cm tall, coloured by intensity in the z-axis. This image demonstrates the fluctuations in the ground and it is made evident that the sensor is not precise enough to differentiate obstacles smaller than 2 cm from the ground.

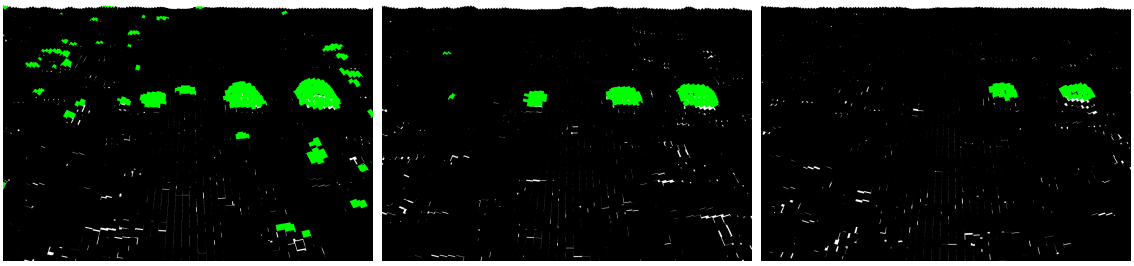


(a) (b)

Figure 5.5: (a) Obstacles ranging from 1 cm to 4 cm tall. (b) Pointcloud output with colour intensity according to the height of points. The flat ground shows height fluctuations due to sensor error, resulting in the 1 cm and 2 cm obstacles blending with the ground points.

The plane segmentation algorithm was tested with different threshold values to find an appropriate middle ground between detecting all objects and allowing noisey points to be classified as obstacles. Fig. 5.6 shows the plane segmentation results when using threshold values of 1 cm, 1.5 cm, and 2 cm. While the 1 cm threshold result allows for the detection of the most obstacles, it allows many unwanted points through. The 1.5 cm threshold result is a great improvement, but still allows enough noise through

that will collect in the costmap over time and tarnish its quality, making the UGVs believe there are obstacles where there are none. Finally, when a 2 cm threshold is applied, no noise is found to pass through. This trade-off with not being able to identify obstacles that are 2 cm or shorter is necessary due to the camera error, as it was shown that objects below this height are challenging to distinguish from the flat ground.



(a) (b) (c)

Figure 5.6: Plane segmentation applied to the same setup as in Fig. 5.5(a). Black points are considered to be part of the ground plane, green points are outlier points. (a) 1 cm threshold. (b) 1.5 cm threshold. (c) 2 cm threshold.

5.3.3 Pointcloud Cleanup

After the removal of points from the ground plane, unwanted points caused by sensor error may remain, as shown in Fig. 5.3(b). Although the 2 cm plane segmentation threshold works well when the camera is stationary, the fluctuations are greater when the UAV is in flight. However, increasing this threshold will cause important obstacles to be ignored and should not be done. Since the unwanted remaining points are few, it is possible to overcome this issue by applying filtering.

To filter out the unwanted points remaining after plane segmentation, clusters of points with fewer than 50 members are to be removed, as they are considered to be very small and likely caused by sensor error. To execute this operation, the remaining points are grouped by proximity using DBSCAN clustering. This allows for individual obstacles to be differentiated as clusters of points. The points are stored in an array

where they are assigned a cluster label. Points that are too far away from their neighbours and, thus, cannot be clustered are automatically removed. Then, clusters of points with fewer than 50 members are deleted from the obstacle pointcloud. This process is further detailed in Algorithm 2. Fig. 5.3(c) shows the pointcloud output after the filtering step.

Algorithm 2 Remove Noise from Obstacle Pointcloud

Require: *obstacle_pcl* ▷ the obstacle pointcloud

1: $labels \leftarrow$ DBSCAN($obstacle\_pcl$)
2: $obstacle\_pcl[label] \leftarrow labels$
3: $obstacle\_pcl$.remove($obstacle\_pcl[label ==$-1]) ▷ remove points which cannot be clustered
4: **for all** *labels* **do** ▷ remove clusters with fewer than 50 members
5: **if** $current\_label < 50$ **then**
6: $obstacle\_pcl$.remove($obstacle\_pcl[label == current\_label]$)
7: **end if**
8: **end for**

5.3.4 Removal of UGV Points

The final operation the pointcloud undergoes is the removal of points representing the UGVs themselves, as they should not be considered as obstacles. The DBSCAN clustering that has already been applied makes it possible to remove the cluster of points belonging to a UGV. The correct cluster label is identified by reading the current known position of the UGV and using KNN to find the nearest point in the pointcloud. Then, all points with the same label are removed, thus removing the entire cluster. This process is further detailed in Algorithm 3

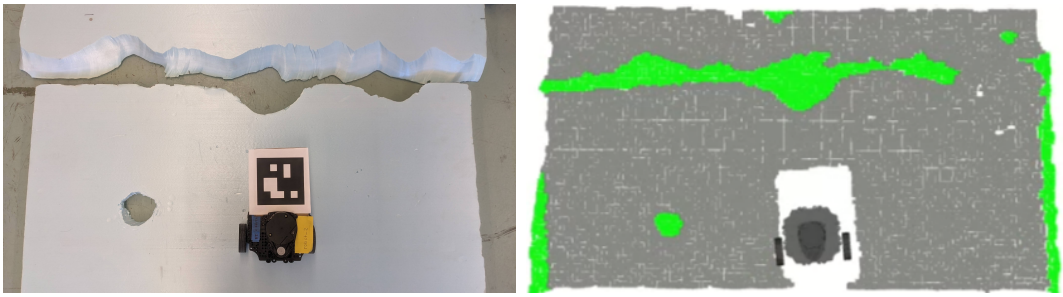
Algorithm 3 Remove UGVs from Obstacle Pointcloud

Require: $obstacle\_pcl$, UGV_coords

1: $labels \leftarrow$ DBSCAN($obstacle\_pcl$)
2: $obstacle\_pcl[label] \leftarrow labels$
3: $nearest\_point \leftarrow obstacle\_pcl$.KNN($UGV\_coords$, 1) ▷ find nearest point to UGV
4: $UGV\_label \leftarrow nearest\_point$.ID
5: $obstacle\_pcl$.remove($obstacle\_pcl[label == UGV\_label]$) ▷ remove cluster

After these steps, all points remaining in the pointcloud are considered to belong

to obstacles. The final output is shown in Fig. 5.3(d). Obstacle points are stored and the obstacle pointcloud is updated as the UAV surveys the area, building a map of all present obstacles. This obstacle pointcloud is published to the ROS server. Fig. 5.7 presents a demonstration of the obstacle detection method wherein negative obstacles, or holes, are detected.



(a) (b)

Figure 5.7: (a) UGV placed in an environment with holes. (b) Visualization output, where green points represent obstacles, including holes.

The design of the proposed method for UAV obstacle detection works on objects of any shape and does not rely on known colours, geometries, or markings. It is designed to work on objects of any size, given that they are more than 2 cm tall and have a surface area of at least 50 cm^3.

5.4 Obstacle Sharing for UGV Avoidance

Once obstacles have been isolated in the UAV's pointcloud, their locations must be shared with the UGVs to enable avoidance. This is achieved by modifying the existing ROS framework for UGV navigation with the new pointcloud data. Since all robots share a common coordinate reference frame, it is possible to relay geo-referenced obstacle information.

The existing UGV navigation scheme uses each robot's laserscan data to monitor its surroundings and add obstacles to its costmap. A costmap is a two-dimensional representation of a robot's environment with values corresponding to the difficulty,

or cost, of traversing each cell. The costmap is based on the occupancy grid, where it is identified whether cells contain free space or are occupied. Around occupied cells, an inflation radius is applied, where cost values are propagated out and decrease with distance. This makes it so that in path planning, the robot is assigned a route that avoids collisions.

In this work, in addition to the UGV laserscan data, the UAV pointcloud data is also used as an obstacle observation source. This makes it possible for the obstacles captured in the pointcloud to be added to the UGVs' costmaps and, thus, considered in path planning. To implement this in the ROS framework, the UGVs' costmap parameter files were modified to include the pointcloud topic being published over the server as a second obstacle observation source, in addition to the laserscan data. Costmap parameter files are part of the ROS Navigation Stack and work to configure the way a robot's costmap is created. In this case, the Obstacle Layer was configured to use the pointcloud data to mark the costmap. The other important settings include:

- Consider only obstacles within the TurtleBot3's height
- Consider obstacles below the ground (holes)
- Do not allow the pointcloud obstacles to be cleared, or removed, by the UGV laserscan due to it seeing free space

These parameters need only to be configured once for each UGV and from then on, its navigation scheme will look for data coming in from the pointcloud topic.

The obstacles isolated in the pointcloud are stamped directly onto the costmap and traversal costs surrounding obstacle points are inflated. This method allows for the shapes and sizes of the obstacles to be captured accurately, rather than approximating the obstacles with circles or squares at their known locations. This enables optimal path planning and ensures that safe routes can be found. Fig. 5.8 demonstrates the addition of an irregularly shaped object to a UGV's costmap.

(a) (b)

Figure 5.8: (a) A UGV and an irregularly shaped obstacle that the UGV cannot detect. (b) UGV costmap depicting the obstacle which has been added from UAV pointcloud data. The pink-coloured areas have a higher traversal cost than the deeper purple-coloured areas.

Chapter 6

UAV Obstacle Mapping Experiments

This chapter provides a review of the experiments undertaken to evaluate the new framework for UAV obstacle mapping for UGV navigation along with test results. First, the UAV obstacle detection method is tested in terms of its positional accuracy, frequency of correct detection, and speed. Next, the UGV obstacle avoidance is evaluated, followed by full system test results. Parts of the experimental results presented in this chapter have been accepted for publication in [75].

6.1 Obstacle Detection Experiments

6.1.1 Positional Accuracy of Obstacle Mapping

Experiments were conducted to measure the positional accuracy of the novel UAV obstacle detection method. The (x, y) coordinates of obstacles as observed by the UAV camera were compared to ground-truth data obtained via an external motion capture system. The centre position of each obstacle was determined by both methods

to allow for a measurable comparison. An obstacle's centre was extracted from the pointcloud data by taking the planar average of its points. For the motion capture system, a reflective marker was placed at the object's centre. Round and rectangular objects were chosen for ease of placement of the reflective marker, but this method can be applied to objects of any shape.

The UAV pointcloud method for obstacle mapping takes the average of available readings across multiple frames to form an estimate of an obstacle's centre position. Fig. 6.1 shows a scatter plot of the centre position estimate for a single obstacle. Here, the average of the ten estimates is taken and compared to the true obstacle position. The variability seen in the estimates is caused by the slight delay in the UAV's own position estimate as it moves. Were the UAV hovering in place, the estimate would show greater consistency. Additionally, the method chosen to determine an obstacle's centre position for the purpose of this experiment is only ideal when the object is directly below the camera and none of its sides are seen, as this skews the planar average of the points. However, when the average of all estimates is taken, the resulting estimate proves to be near the true obstacle position.

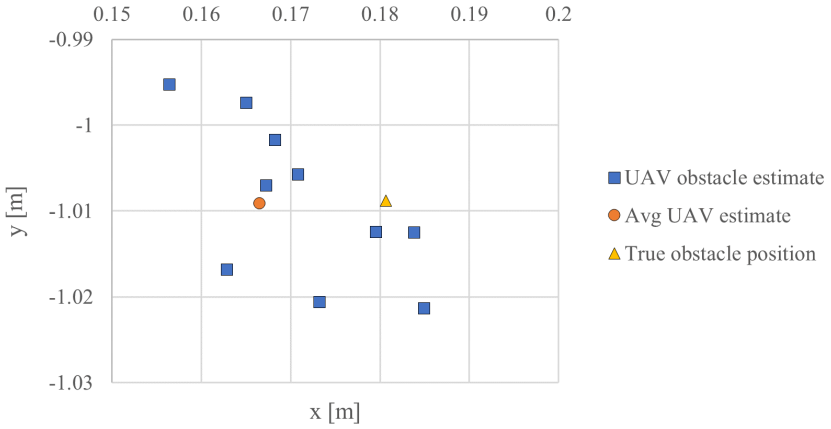


Figure 6.1: Obstacle position estimate for a single object, taking the average of ten sensor readings.

The positional accuracy of the UAV obstacle detection method was assessed through testing with various objects, encompassing a total of fifteen trials. Each trial calculated the average position from five to ten available frames. Analysis of the outcomes,

as presented in Table 6.1, reveals an average planar error of 3.06 cm, with a standard deviation of 1.33 cm. Planar error was computed as the Euclidean distance between the true coordinates of obstacles and their corresponding estimated positions in the x-y plane. Given the costmap inflation radius of 15 cm, this low positional error can ensure that obstacles are well avoided.

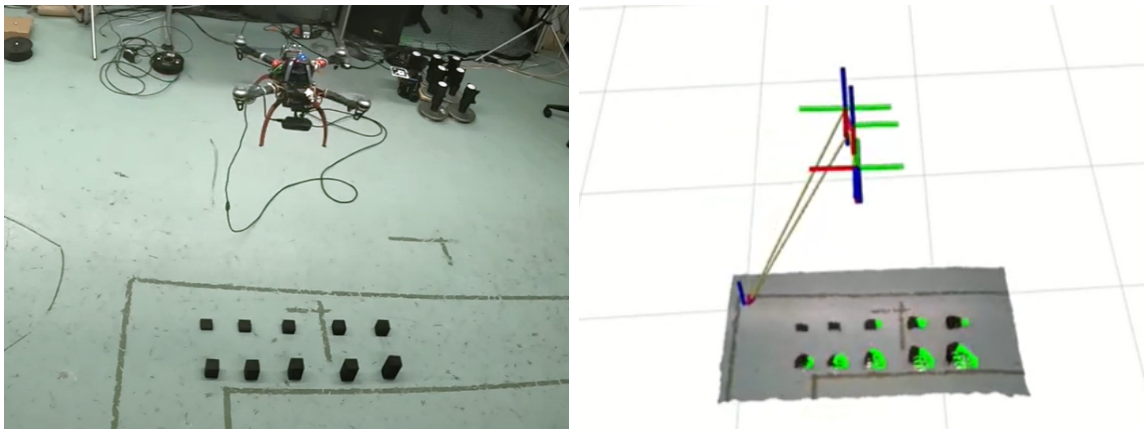
Table 6.1: UAV obstacle detection positional error for various objects over fifteen runs.

	Planar (x, y) error [cm]
Average	3.06
Minimum	1.23
Maximum	5.30
Standard Deviation	1.33

6.1.2 Frequency of Obstacle Detection

The robustness of the UAV obstacle detection technique was assessed through the use of objects of varying heights. Twenty boxes were fabricated for this experiment, each having a base measuring 5 cm by 5 cm and incrementing heights ranging from 1 cm to 20 cm. The primary objective was to ascertain whether the method could detect each box within 1 second upon its entrance into the UAV's field of view. The experimental setup and ensuing results are shown in Fig. 6.2.

The results show an obstacle detection rate of 90%, with the exceptions being the 1 cm tall and 2 cm tall boxes. It was expected that objects under 2 cm tall would not be detectable with this method, as they would get classified as part of the floor in ground plane segmentation. This is a result of noise in the camera's depth readings, as discussed in the previous chapter. Notably, no instances of false detection were recorded. Moreover, the algorithm demonstrated a high level of repeatability, characterized by consistent and quick marking of detectable objects by the camera upon their entry into its field of view.



(a) (b)

Figure 6.2: (a) UAV surveying obstacles with heights ranging from 1 cm to 10 cm. (b) Visualization output illustrating the successful detection of all obstacles taller than 2 cm, denoted by green points. No instances of false detection occurred.

6.1.3 Speed of Obstacle Detection

During the experiments described previously, measurements of image acquisition rates and processing times were taken. The measurements are shown in Table 6.2. The pointcloud processing for obstacle detection is found to run at an average frequency of 7 Hz. This high rate leads to quick detection of objects, even if they are only in the UAV's field of view for a brief moment. In addition, the UGV localization via ArUco detection was found to run at rate of 30 Hz.

Table 6.2: Frequency and time consumption for robot localization and obstacle detection processes.

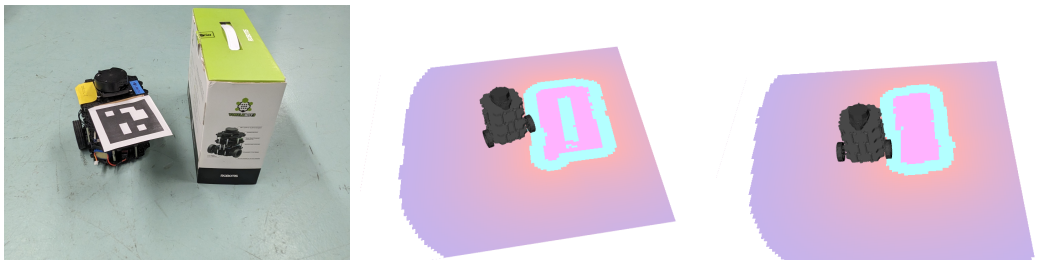
	Average Frequency (Hz)	Max. Period (s)	Min. Period (s)
2D image streaming	30	0.324	0.10
ArUco detection	30	0.048	0.020
Pointcloud streaming	30	0.039	0.029
Pointcloud processing	7	0.240	0.073

6.2 Obstacle Avoidance Experiments

6.2.1 Costmap Validation

In addition to ensuring accurate recording of obstacle locations, it was imperative to validate that the costmap appropriately represented the shapes and dimensions of obstacles. As explained in the methodology, the costmap is dynamically updated directly from the pointcloud data rather than taking the central position of obstacles and inflating them. This approach was selected to depict the true geometries of objects on the costmap, facilitating optimal path planning and avoidance.

To assess whether the shapes and sizes of obstacles are recorded accurately on the costmap, the experiment depicted in Fig. 6.3 was conducted. An obstacle tall enough to be detected by the UGV laserscan was chosen and the UGV was driven around it to capture it on its costmap. Then, the experiment was repeated with the UGV LiDAR deactivated, employing the UAV camera for obstacle detection instead. Visually similar outcomes were found between the UAV-generated and UGV costmaps, confirming that this method is sufficient for comparable obstacle avoidance. Additionally, another experiment with a different obstacle shape yielded similar satisfactory results, validating the approach.



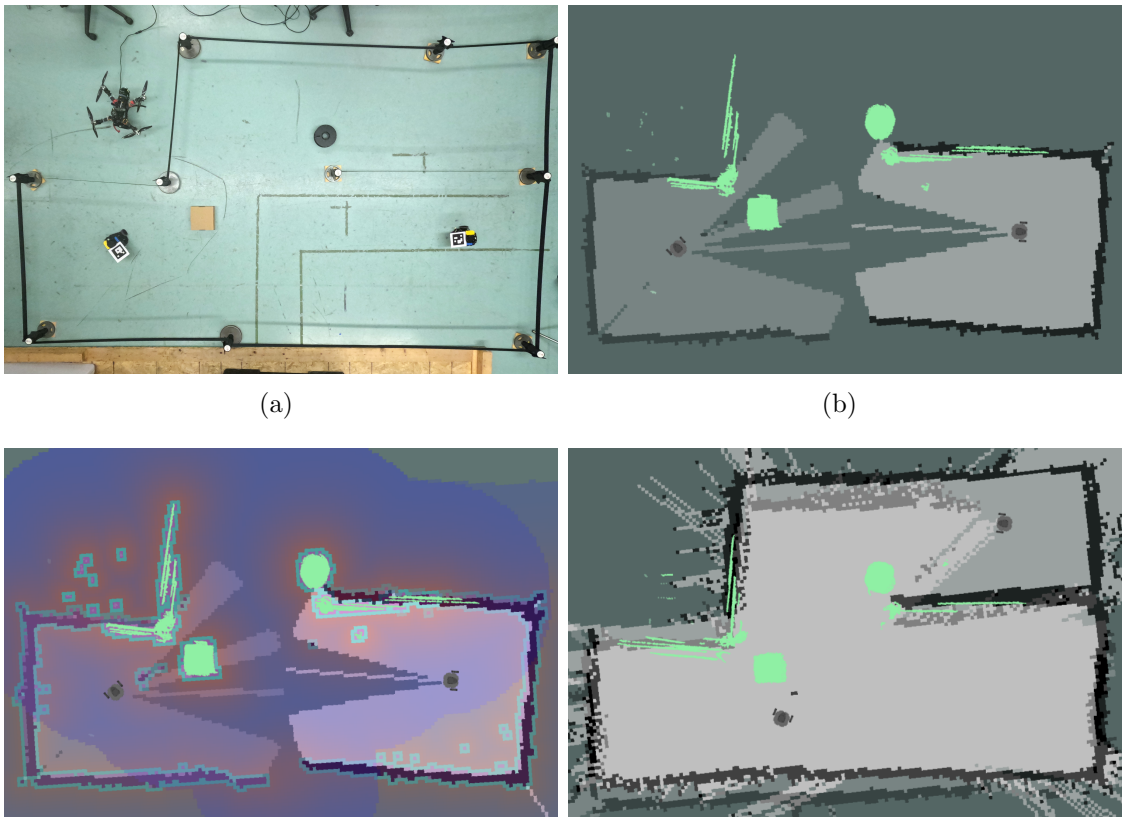
(a) (b) (c)

Figure 6.3: (a) Setup for UGV costmap experiment. (b) Costmap generated using UGV LiDAR data. (c) Costmap generated using UAV pointcloud data, demonstrating comparable outcomes.

6.2.2 UGV Navigation Among Obstacles

The UGVs were assigned the task of autonomous exploration and mapping within an environment containing obstacles short enough to only be detectable by the UAV. This comprehensive experiment validates the achievement of the research goal regarding the use of the novel obstacle detection method for such missions.

The experimental setup, depicted in Fig. 6.4(a), features two obstacles. The UAV performed a pre-programmed autonomous flight to survey the area for UGVs and obstacles. Subsequently, the initial poses of the two UGVs were recorded by the UAV upon the detection of their ArUco markers. The two obstacles were detected and then incorporated into the UGVs' costmaps, as illustrated in Fig. 6.4(c). Finally, the autonomous multi-UGV exploration and mapping algorithm was executed, resulting in the successful mapping of the area, as demonstrated in Fig. 6.4(d). The path-planning of the UGVs was notably influenced by the UAV-detected obstacles, as evident from their maintained distance during navigation.



(a) (b)

(c) (d)

Figure 6.4: (a) Experimental setup for obstacle avoidance. (b) UGV localization with UAV-detected obstacles shown in green. (c) UGV costmaps showing inclusion of UAV-detected obstacles. (d) Resultant map created by UGVs.

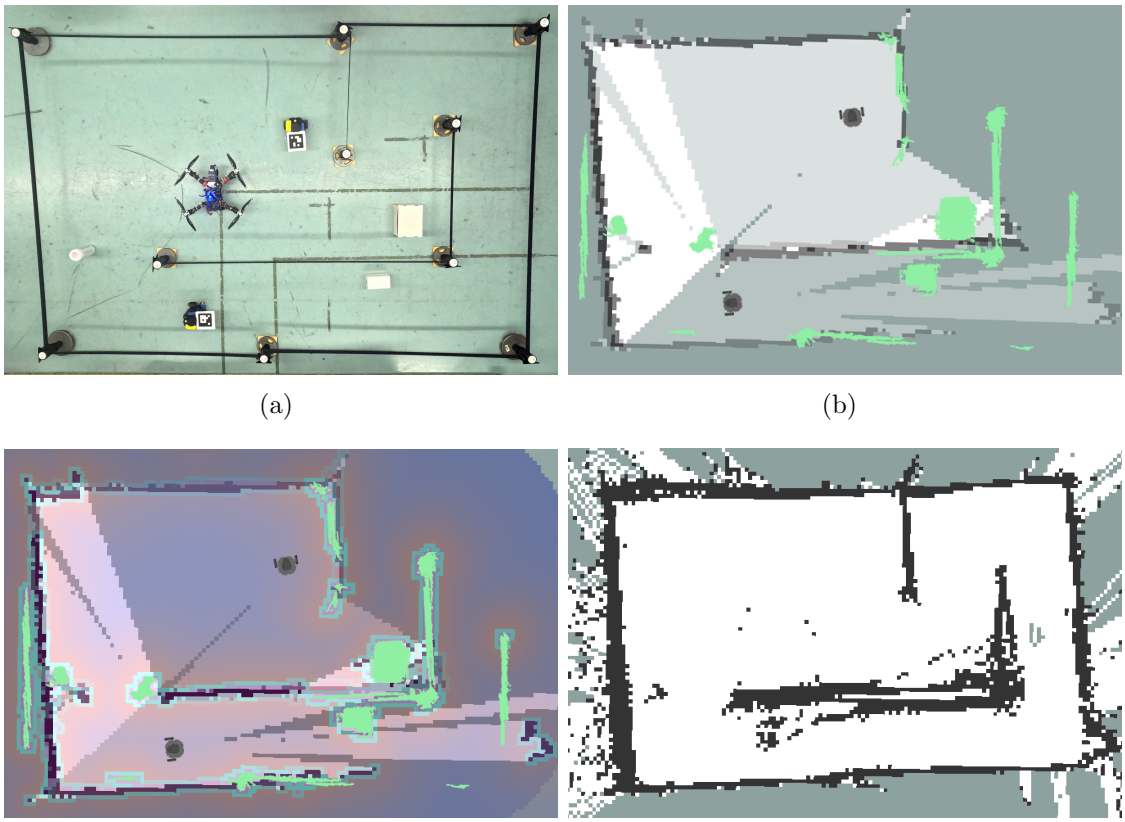
6.3 Full-System Implementation

Further testing was conducted through comprehensive experiments to validate that the various developed methods work together successfully to meet the research objectives. Several configurations of UGVs, obstacles, and the environment were used. The criteria for success are listed as follows:

- UAV survey detects all UGV poses
- UGV pose estimations support map merging
- UAV survey detects all obstacles

- Obstacles added to UGV costmaps
- UGVs safely traverse the environment and create an accurate map representing the environment

The configuration in Fig. 6.5(a) features two UGVs and three obstacles. After the UAV survey flight took place, it was confirmed that the UGV poses were accurately estimated to support map merging and that the three obstacles were detected, as demonstrated in Fig. 6.5(b). The obstacle information from the pointcloud was added to both UGV costmaps in Fig. 6.5(c). This enabled the UGVs to safely navigate the environment and to create a map of it, shown in Fig. 6.5(d).

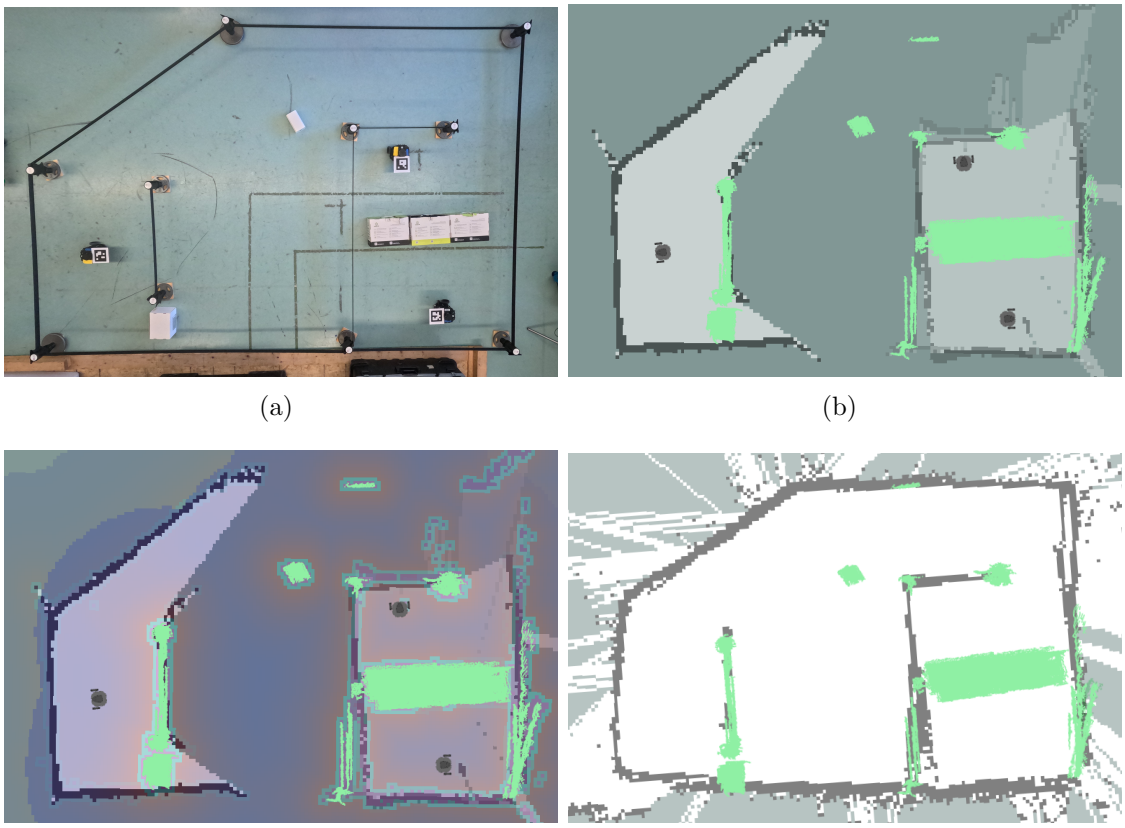


(a) (b)

(c) (d)

Figure 6.5: (a) Experiment setup with two UGVs and three short obstacles (white). (b) UGV LiDAR scans showing successful map alignment from localization. UAV-detected obstacle pointcloud shown in green. (c) UGV costmap, including UAV-detected obstacles. (d) Resultant map created by UGVs.

A second configuration is shown in Fig. 6.6(a), including three UGVs and three obstacles. Here, the placement of the obstacles demonstrates a strong effect on path planning. The UGV on the bottom right cannot sense the large obstacle beside it and it is only aware of it from the UAV pointcloud data. This obstacle keeps the UGV trapped and prevents it from performing exploration. If the UAV did not provide information about this obstacle, the UGV would have collided with it. Similarly, the path planning of the UGV on the left side of the environment is affected by the obstacle near it, causing it to take a different route. Thanks to the successful relative localization and obstacle detection, the UGVs were able to build a map of their environment (see Fig 6.6(d)).



(a) (b)

(c) (d)

Figure 6.6: (a) Experiment setup with three UGVs and three short obstacles (white). (b) UGV LiDAR scans showing successful map alignment from localization. UAV-detected obstacle pointcloud shown in green. (c) UGV costmap, including UAV-detected obstacles. (d) Resultant map created by UGVs.

A video demonstration of this experiment can be found in `https://youtu.be/X17OKxwzg4Q`. It should be noted that the vague edges of the produced maps are caused by the thin ribbon used to form the walls in the experimentation area. Sag in the ribbons causes the UGV LiDARs to see above or below the walls as they move, allowing them to see beyond the walls. A true walled environment would allow for the creation of more distinct maps. Further inaccuracies in the maps, such as thick or skewed walls, are caused by the UGVs' accumulating pose errors as they move through the environment, which is not a result of their initial pose estimations produced by the novel localization method. These pose errors originate from the existing odometry and scan matching techniques used by the UGVs.

These full-system implementation experiments demonstrate that the research objectives have been met. They exemplify a mission where the UAV surveys the area to provide relative localization and obstacle mapping for the UGVs. As a result, the multi-UGV exploration and mapping task can be accomplished with greater autonomy and safety. The developed methods can be easily applied to other UAV/UGV systems.

Chapter 7

Conclusions and Recommendations for Future Work

7.1 Conclusions

This thesis is about the cooperation between a UAV and multiple UGVs to explore unknown environments. It describes the addition of a UAV into an existing multi-UGV framework to enhance its autonomous capabilities. A review of the literature was conducted regarding existing UAV/UGV collaboration, multi-UGV exploration, robot localization, and aerial obstacle mapping. It was found that multi-UGV exploration can be improved through autonomous relative localization of the robots as opposed to manual entry. Of the methods wherein a UAV assists UGVs with localization, approaches using libraries of unique markers, such as ArUco, were found to be the most simple to implement. However, these approaches require constant position feedback and are not suitable for multi-UGV scenarios. Hence, this research set out to develop a novel localization method wherein the UAV provides UGVs with their relative poses to initialize a global map, enabling collaborative exploration. Through the literature review, an opportunity to also employ the UAV in obstacle detection to

aid UGV navigation was identified. The main approaches to UAV obstacle detection were found to be elevation modeling, semantic segmentation, and discrete obstacle detection. Within discrete obstacle detection, existing methods required obstacles with recognizable geometries, colours, or markings, and none use three-dimensional data. This revealed the opportunity to use UAV-collected depth data for the detection of individual objects as obstacles.

The main objectives of the research were established to be the development of methods to enable a UAV to localize multiple UGVs and to provide the UGVs with obstacle information. A framework was developed using ROS and open source hardware, including a quadcopter UAV and Turtlebot3 Burger UGVs. Existing ROS packages were used for SLAM, map merging, UGV navigation, autonomous flight, camera operation, and ArUco detection. New nodes were developed for UGV localization and obstacle mapping.

The new UAV/UGV relative localization method requires ArUco markers to be mounted to the UGVs. During an autonomous UAV flight at the start of a mission, detection of these markers allows for the estimation of UGV poses. The ArUco pose information is averaged, transformed relative to a shared global reference frame, and used to estimate UGV poses. The UGV pose estimates are then used to initialize map merging between UGVs and to localize the robots. Once this initial pose estimation is provided and the global map is built, subsequent pose updates during UGV navigation are handled using existing odometry and laserscan matching techniques.

The novel method for UAV obstacle mapping for UGV navigation uses three-dimensional data gathered by the UAV to build a pointcloud. This pointcloud is processed to isolate points belonging to obstacles. The main processing steps include plane segmentation to remove floor points, clean-up using DBSCAN clustering, and UGV points removal. A plane segmentation threshold of 2 cm was found to produce the best results in balancing the trade-off in claiming too many points as part of the ground and not enough. The output pointcloud is published to the ROS server and used by the

UGV navigation packages to build a costmap. The pointcloud marks the locations of obstacles on UGV costmaps so that they are considered in path planning.

Tests were conducted to evaluate the performance of the developed methods. For the UAV/UGV localization method, UGV pose estimates were compared to ground truth data to determine its accuracy. It was found that with a UAV flight height of 1.25 m, the average position error was 4.01 cm and the average orientation error was 1.72°. These results were found to be sufficient for the creation of merged maps, which are defined to require positional accuracy within 10 cm and orientation accuracy within 5°. Error was found to increase with UAV planar velocity and flight height. Global map initialization between multiple UGVs was tested and the relative localization results were found to produce maps that accurately represent an environment.

The aerial obstacle mapping method also underwent a series of tests to assess its performance. First, the accuracy of obstacle position estimation was evaluated by comparing the estimated coordinates of obstacle centres with their true values. It was determined that the average planar error was 3.06 cm. With the obstacle inflation radius on the UGV costmaps, this low positional error can ensure collision-free navigation. Through additional experiments, it was found that the novel method can repeatably identify obstacles that are over 2 cm tall. It was also found that the shapes and sizes of obstacles were recorded accurately on UGV costmaps. Once their costmaps were marked, UGVs were able to avoid the obstacles as though they were able to sense them themselves.

Full-system testing was conducted to put all of the developed methods into one cumulative mission. This started with a UAV flight wherein the area was surveyed for UGVs and obstacles. Once UGV ArUco markers were detected, their poses were estimated and they were brought into the global map. Obstacle information was stored in a pointcloud that was used to update the UGV costmaps. After the autonomous flight, the UGVs explored the area to create a map. Through the execution of this experiment in various configurations, it was proven that each method worked

as intended to allow for successful UGV mapping.

7.2 Recommendations for Future Work

The developed methods for collaborative UAV/UGV localization and obstacle mapping have been proven to enable increased autonomy in multi-robot exploration and mapping missions. A few recommendations for improvement and further extension of the methods are outlined below.

The current method for relative localization only uses the UAV to inform the UGVs of their initial poses and, thereafter, existing odometry and laserscan matching techniques are used to track their poses. However, there is an opportunity to use the UAV to update UGV poses during their exploration in order to correct dead-reckoning errors. This would result in improved map accuracy. Sensor fusion techniques may be used to factor in UGV pose data received from the UAV in their pose estimates.

Another suggestion for improvement of the relative localization method is the use of a more complex visual detection method, as seen in the work of Faessler et al. [49]. This method uses a set of infrared LEDs instead of an ArUco marker and would allow for more precise pose estimation at greater distances. Additionally, this method would not be vulnerable to changing light conditions, as is the case for ArUco markers.

A limitation of the current obstacle detection method is its requirement for flat ground surfaces. This is sufficient for most indoor environments but the process would need to be updated to function in outdoor terrain with uneven surfaces. A suggestion to overcome this issue would be to replace the plane segmentation step with another process that can ignore points with a slight height gradient to their neighbours. Alternatively, instead of isolating the obstacles in the pointcloud, elevation model methods may be more suitable for UGV navigation in rough terrain, so that the cost of traversing slopes could be considered.

References

[1] L. Goodwin, "A Robust and Efficient Autonomous Exploration Methodology of Unknown Environments for Multi-Robot Systems," Master's thesis, Ontario Tech University, Oshawa, ON, 2022.

[2] L. Goodwin and S. Nokleby, "A K-Means Clustering Approach to Segmentation of Maps for Task Allocation in Multi-robot Systems Exploration of Unknown Environments," in *Proceedings of the 2022 USCToMM Symposium on Mechanical Systems and Robotics*, P. Larochelle and J. M. McCarthy, Eds. Cham: Springer International Publishing, 2022, pp. 198–211.

[3] Q. Vu, M. Raković, V. Delic, and A. Ronzhin, "Trends in Development of UAV-UGV Cooperation Approaches in Precision Agriculture," in *Interactive Collaborative Robotics*, A. Ronzhin, G. Rigoll, and R. Meshcheryakov, Eds. Cham: Springer International Publishing, 2018, pp. 213–221.

[4] H. Miura, A. Watanabe, M. Okugawa, T. Miura, and T. Koganeya, "Plant Inspection by Using a Ground Vehicle and an Aerial Robot: Lessons Learned From Plant Disaster Prevention Challenge in World Robot Summit 2018," *Advanced Robotics*, vol. 34, pp. 104–118, 2020.

[5] K. Asadi, A. K. Suresh, A. Ender, S. Gotad, S. Maniyar, S. Anand, M. Noghabaei, K. Han, E. Lobaton, and T. Wu, "An Integrated UGV-UAV System for Construction Site Data Collection," *Automation in Construction*, vol. 112, 2020.

[6] E. H. C. Harik, F. Guérin, F. Guinand, J.-F. Brethé, and H. Pelvillain, "UAV-UGV Cooperation for Objects Transportation in an Industrial Area," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015, pp. 547–552.

[7] Z. Kashino, G. Nejat, and B. Benhabib, "Aerial Wilderness Search and Rescue with Ground Support," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 99, 2020.

[8] J. Chen, X. Zhang, B. Xin, and H. Fang, "Coordination Between Unmanned Aerial and Ground Vehicles: A Taxonomy and Optimization Perspective," *IEEE Transactions on Cybernetics*, vol. 46, 2016.

[9] Y. Ding, B. Xin, and J. Chen, "A Review of Recent Advances in Coordination between Unmanned Aerial and Ground Vehicles," *Unmanned Systems*, vol. 9, 2021.

[10] S. Çaşka and A. Gayretli, "A Survey of UAV/UGV Collaborative Systems," in *CIE44 & IMSS'14 Proceedings*, 2014.

[11] G. Christie, A. Shoemaker, K. Kochersberger, P. Tokekar, L. McLean, and A. Leonessa, "Radiation Search Operations Using Scene Understanding with Autonomous UAV and UGV," *Journal of Field Robotics*, vol. 34, 2017.

[12] P. Fankhauser, M. Bloesch, P. Krüsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, "Collaborative Navigation for Flying and Walking Robots," in *IEEE International Conference on Intelligent Robots and Systems*, 2016.

[13] B. Sofman, J. A. Bagnell, A. Stentz, and N. Vandapel, "Terrain Classification from Aerial Data to Support Ground Vehicle Navigation," *Robotics*, 2006.

[14] N. Giakoumidis, J. U. Bak, J. V. Gómez, A. Llenga, and N. Mavridis, "Pilot-Scale Development of a UAV-UGV Hybrid with Air-Based UGV Path Planning,"

in *Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012*, 2012.

[15] M. Garzón, J. Valente, D. Zapata, and A. Barrientos, "An Aerial-Ground Robotic System for Navigation and Obstacle Mapping in Large Outdoor Areas," *Sensors (Switzerland)*, vol. 13, pp. 1247–1267, 2013.

[16] B. Gilhuly and S. L. Smith, "Robotic Coverage for Continuous Mapping Ahead of a Moving Vehicle," *arXiv.org*, 2019.

[17] K. Harikumar, T. Bera, R. Bardhan, and S. Sundaram, "Autonomous Navigation and Sensorless Obstacle Avoidance for UGV with Environment Information from UAV," in *Proceedings - 2nd IEEE International Conference on Robotic Computing, IRC 2018*, vol. 2018-January. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 266–269.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza, "Air-Ground Localization and Map Augmentation using Monocular Dense Reconstruction," in *IEEE International Conference on Intelligent Robots and Systems*, 2013.

[19] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-Denied Environments," *IEEE Transactions on Vehicular Technology*, vol. 68, 2019.

[20] J. H. Kim, J. W. Kwon, and J. Seo, "Multi-uav-based stereo vision system without gps for ground obstacle mapping to assist path planning of ugv," *Electronics Letters*, vol. 50, 2014.

[21] C. Phan and H. H. Liu, "A Cooperative UAV/UGV Platform for Wildfire Detection and Fighting," in *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing, ICSC 2008*, 2008.

[22] J. Li, Y. Cheng, J. Zhou, J. Chen, Z. Liu, S. Hu, and V. C. Leung, "Energy-Efficient Ground Traversability Mapping Based on UAV-UGV Collaborative Sys-

tem," *IEEE Transactions on Green Communications and Networking*, vol. 6, 2022.

[23] S. Çaşka and A. Gayretli, "An Algorithm for Collaborative Patrolling Systems with Unmanned Air Vehicles and Unmanned Ground Vehicles," in *RAST 2015 - Proceedings of 7th International Conference on Recent Advances in Space Technologies*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 659–663.

[24] B. Arbanas, A. Ivanovic, M. Car, M. Orsag, T. Petrovic, and S. Bogdan, "Decentralized Planning and Control for UAV–UGV Cooperative Teams," *Autonomous Robots*, vol. 42, 2018.

[25] M. F. S. Rabelo, A. S. Brandao, and M. Sarcinelli-Filho, "Landing a UAV on Static or Moving Platforms Using a Formation Controller," *IEEE Systems Journal*, vol. 15, 2021.

[26] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture," *IEEE Transactions on Robotics*, vol. 32, 2016.

[27] T. Miki, P. Khrapchenkov, and K. Hori, "UAV/UGV Autonomous Cooperation: UAV Assists UGV to Climb a Cliff by Attaching a Tether," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8041–8047.

[28] S. Minaeian, J. Liu, and Y. J. Son, "Crowd Detection and Localization using a Team of Cooperative UAV/UGVs," in *IIE Annual Conference and Expo 2015*, 2015.

[29] K. Lu, R. Xu, J. Li, Y. Lv, H. Lin, and Y. Liu, "A Vision-Based Detection and Spatial Localization Scheme for Forest Fire Inspection from UAV," *Forests*, vol. 13, 2022.

[30] R. Fedorenko, A. Gabdullin, and A. Fedorenko, "Global ugv path planning on point cloud maps created by uav," in *2018 3rd IEEE International Conference on Intelligent Transportation Engineering, ICITE 2018*, 2018.

[31] C. Hu, C. Hu, D. He, and Q. Gu, "A New ROS-based Hybrid Architecture for Heterogeneous Multi-robot Systems," in *Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015*, 2015.

[32] B. Yamauchi, "A Frontier-based Exploration for Autonomous Exploration," *IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA*, 1997.

[33] ——, "Frontier-based Exploration Using Multiple Robots," in *Proceedings of the International Conference on Autonomous Agents*, 1998.

[34] C. Nieto-Granda, J. G. Rogers, and H. I. Christensen, "[coordination strategies for multi-robot exploration and mapping," *International Journal of Robotics Research*, vol. 33, 2014.

[35] Matan Keidar and Gal A. Kaminka, "Efficient Frontier Detection for Robot Exploration," *International Journal of Robotics Research*, vol. 33, 2014.

[36] P. Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems," *IEEE Robotics and Automation Letters*, vol. 9, pp. 475–482, 2024.

[37] H. Xie, D. Zhang, X. Hu, M. C. Zhou, and Z. Cao, "Autonomous Multirobot Navigation and Cooperative Mapping in Partially Unknown Environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, 2023.

[38] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed Multirobot Exploration and Mapping," *Proceedings of the IEEE*, vol. 94, 2006.

[39] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated Multi-Robot Exploration," *IEEE Transactions on Robotics*, vol. 21, 2005.

[40] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for Multi-Robot Exploration and Mapping," in *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, AAAI 2000*, 2000.

[41] R. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-Robot Exploration Controlled by a Market Economy," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, 2002.

[42] B. P. Gerkey and M. J. Matarić, "Sold!: Auction Methods for Multirobot Coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, 2002.

[43] J. D. Hoog, S. Cameron, and A. Visser, "Role-Based Autonomous Multi-Robot Exploration," in *Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, ComputationWorld 2009*, 2009.

[44] A. Solanas and M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2004.

[45] D. Puig, M. A. Garcia, and L. Wu, "A New Global Optimization Strategy for Coordinated Multi-Robot Exploration: Development and Comparative Evaluation," *Robotics and Autonomous Systems*, vol. 59, 2011.

[46] J. Faigl, M. Kulich, and L. Preucil, "Goal Assignment Using Distance Cost in Multi-Robot Exploration," in *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[47] P. Stegagno, M. Cognetti, L. Rosa, P. Peliti, and G. Oriolo, "Relative Localization and Identification in a Heterogeneous Multi-Robot System," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1857–1864.

[48] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza, "Aerial-Guided Navigation of a Ground Robot Among Movable Obstacles," in *12th IEEE Interna-*

tional Symposium on Safety, Security and Rescue Robotics, SSRR 2014 - Symposium Proceedings, 2014.

[49] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A Monocular Pose Estimation System based on Infrared LEDs," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014.

[50] H. Sun, W. Zheng, Y. Cao, and Z. Wang, "A Strategy for Heterogeneous Multi-Robot Self-Localization System," in *2011 Chinese Control and Decision Conference (CCDC)*. IEEE, 2011, pp. 4198–4203.

[51] J. Peterson, H. Chaudhry, K. Abdelatty, J. Bird, and K. Kochersberger, "Online Aerial Terrain Mapping for Ground Robot Navigation," *Sensors (Switzerland)*, vol. 18, 2018.

[52] K. Guo, X. Li, and L. Xie, "Ultra-Wideband and Odometry-Based Cooperative Relative Localization with Application to Multi-UAV Formation Control," *IEEE Transactions on Cybernetics*, vol. 50, pp. 2590–2603, 2020.

[53] A. Babinec, L. Jurišica, P. Hubinský, and F. Duchoň, "Visual Localization of Mobile Robot using Artificial Markers," in *Procedia Engineering*, vol. 96. Elsevier Ltd, 2014.

[54] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion," *Pattern Recognition*, vol. 47, pp. 2280–2292, 2014.

[55] T. Tocci, L. Capponi, and G. Rossi, "ArUco Marker-Based Displacement Measurement Technique: Uncertainty Analysis," *Engineering Research Express*, vol. 3, 2021.

[56] F. Agüera-Vega, F. Carvajal-Ramírez, P. Martínez-Carricondo, J. S.-H. López, F. J. Mesas-Carrascosa, A. García-Ferrer, and F. J. Pérez-Porras, "Reconstruction of extreme topography from uav structure from motion photogrammetry," *Measurement*, vol. 121, 2018.

[57] X. Zhang, H. Li, Z. Gong, Z. Zhou, W. Dai, L. Wang, and S. Daramola, "Method for UAV-Based 3D Topography Reconstruction of Tidal Creeks," *Journal of Geographical Sciences*, vol. 31, 2021.

[58] J. Lisein, M. Pierrot-Deseilligny, S. Bonnet, and P. Lejeune, "A Photogrammetric Workflow for the Creation of a Forest Canopy Height Model from Small Unmanned Aerial System Imagery," *Forests*, vol. 4, 2013.

[59] J. J. Ruiz, L. Diaz-Mas, F. Perez, and A. Viguria, "Evaluating the accuracy of dem geneation algorithms from uav imagery," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2013.

[60] M. A. Fonstad, J. T. Dietrich, B. C. Courville, J. L. Jensen, and P. E. Carbonneau, "Topographic Structure from Motion: A New Development in Photogrammetric Measurement," *Earth Surface Processes and Landforms*, vol. 38, 2013.

[61] V. Pradeep, R. Khemmar, L. Lecrosnier, Y. Duchemin, R. Rossi, and B. Decoux, "Self-Supervised Sidewalk Perception Using Fast Video Semantic Segmentation for Robotic Wheelchairs in Smart Mobility," *Sensors*, vol. 22, 2022.

[62] N. Khabbaz and S. Nokleby, "ArUco-Based Global Map Initialization for Multi-Robot Exploration," in *Proceedings of the 2023 CCToMM Symposium on Mechanisms, Machines, and Mechatronics*, 2023.

[63] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, 2009.

[64] V. Ermakov, "ROS Wiki: MAVROS," https://wiki.ros.org/mavros, [Online; accessed 14-February-2024].

[65] Intel, "RealSense Camera ROS Wrapper," https://dev.intelrealsense.com/docs/ros1-wrapper, [Online; accessed 14-February-2024].

[66] ROBOTIS, "ROBOTIS E-Manual: TurtleBot3 Burger," https://emanual.robotis.com/docs/en/platform/turtlebot3/features/, [Online; accessed 9-February-2024].

[67] E. Marder-Eppstein, "ROS Wiki: Navigation," http://wiki.ros.org/navigation, [Online; accessed 14-February-2024].

[68] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, pp. 269–271, 1959.

[69] B. Gerkey, "Gmapping," https://github.com/ros-perception/slam_gmapping, [Online; accessed 14-February-2024].

[70] J. Vaughan, "ROS Wiki: Aruco_Detect," http://wiki.ros.org/aruco_detect, [Online; accessed 14-February-2024].

[71] T. Foote, "tf: The transform library," in *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2013.

[72] J. Hörner, "Map-Merging for Multi-Robot System," Bachelor's Thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2016.

[73] X. Kan, J. Thomas, H. Teng, H. G. Tanner, V. Kumar, and K. Karydis, "Analysis of ground effect for small-scale uavs in forward flight," *IEEE Robotics and Automation Letters*, pp. 3860–3867, 2019.

[74] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv preprint arXiv:1801.09847*, 2018.

[75] N. Khabbaz and S. Nokleby, "UAV Obstacle Mapping for Multi-UGV Exploration and Mapping," in *Proceedings of the 2024 USCToMM Symposium on Mechanical Systems and Robotics*, to appear.