Spring 6-11-2023

Underwater Robot Path Planning in an Intermittent Communication System

Hunter Gallant
Dartmouth College, hunter.j.gallant.gr@dartmouth.edu

UNDERWATER ROBOT PATH PLANNING IN AN INTERMITTENT COMMUNICATION SYSTEM

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

in

Computer Science

by Hunter Gallant

Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

May 2022

Examining Committee:

Alberto Quattrini Li, Chair

Xia Zhou

Soroush Vosoughi

F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

Abstract

Sunflower, a novel cross-medium localization system between an aerial drone and an underwater robot, has not yet been implemented in a multi-robot exploration system. This project's aim was to simulate various configurations of multi-robot systems, and to create an algorithm, called AdjustPath, to improve exploration and avoid inter-robot collisions. With three, five, seven, and ten simulated underwater robots, there was significant improvement when the AdjustPath algorithm was used. Knowing this, future hardware using the Sunflower system could use this proposed algorithm to increase efficiency and avoid more collisions.

Acknowledgments

I'd like to thank my advisors, Alberto Quattrini Li and Xia Zhou, for taking the time to help refine and improve my thesis, as well as Soroush Vosoughi, for joining my thesis committee. I'd also like to thank Charlie Carver and Qijia Shao for allowing me to join their incredible project and iterate on their existing codebase. Your research is incredible, and I can't wait to see what you do next.

Thank you to my friends from Dartmouth who supported me in my journey through this master's program—particularly Corbin, Jason, Nick, Ted, and Jess. Your support and friendship helped me through the toughest parts of this program. And a massive thank you to my family for having my back and providing your endless love and support.

Contents

Abstract ii

Acknowledgments iii

1 Introduction **1**

2 Related Work **4**

3 Approach **8**

3.1 Problem Statement 8

3.2 AdjustPath 10

3.3 Better Priority Queue 11

4 Experimental Setting **13**

4.1 Metrics 13

4.2 Environment 14

4.3 Implementation 15

4.4 Additional Tests 17

5 Results **18**

6 Conclusion **23**

6.1 Future Work 24

List of Figures

1.1 An aerial drone communicates with one underwater robot at a time, in a multi-robot system where the robots can't communicate directly to each other. 2

4.1 Screenshot of the running simulation. The robots are represented by the colored pill-shaped objects; the obstacles, by the red cubes; and the drone, by the purple cube. 14

5.1 Tests on different numbers of robots. 19

5.2 Tests on varying delay times. 20

Chapter 1

Introduction

Water is an incredibly difficult environment to communicate effectively over long distances, with the two best options being radio waves and sonar. Unfortunately, radio waves travel slower underwater than in air, and both radio and sonar suffer from low bandwidth [1] while communicating underwater. This creates a problem when attempting to coordinate multi-robot exploration systems.

An alternative to direct robot-to-robot communication is presented by Amphilight [2], and the problem of localization is addressed by Sunflower [3]. Using visible light lasers, we are able to send data from an underwater robot to an aerial drone. Using Sunflower's laser localization as a foundation, we can build a "data delivery" system, where the drone communicates with each robot one-at-a-time. This consists of the drone sending the planned paths and path history of the other robots, and the robot sending back an updated planned path based off that received info.

The practical applications of a system like this for naval military use is clear, but it could also be used for civilian operations as well. In 2011, after a massive tsunami, Japan used a series of heterogeneous underwater robots to search for victims in the underwater wreckage [4]. The potential for the Sunflower system to assist with search-and-rescue operations like this is incredible, and if this thesis can improve the

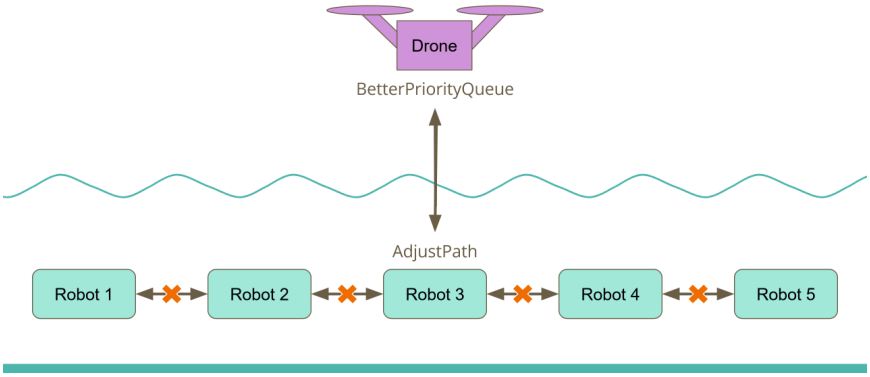

Figure 1.1: An aerial drone communicates with one underwater robot at a time, in a multi-robot system where the robots can't communicate directly to each other.

efficiency of that system even a little bit more, then the time and effort spent would not have been in vain.

The goal with this thesis is to show that a laser based system for air-water localization/communication provides a viable platform for using swarm robotics in an environment where consistent inter-robot communication can be unreliable – see Figure 1.1 for the general scenario. To do this, I built a simulation based off this system. This serves as a framework for testing an algorithm called AdjustPath, designed specifically for this method, showing how it can be implemented on hardware. AdjustPath functions as an optimization addendum to any given robot's path-finding algorithm, improving performance in areas not fully addressed in other works.

Using a simulation allows us to run experiments without spending excess time and money on hardware systems, allowing for a quick turnaround period of algorithmic edits. With future experiments, this simulation platform can be used to refine and adapt algorithms before they are tested in a real-world setting, reducing costs for the lab. The experiments showed strong indications that the proposed AdjustPath algorithm is capable of reducing the number of collisions between robots, and additionally provides a slight increase in the amount explored by the all the robots combined.

In the following, I will start by discussing the current gaps in literature around

communication between underwater robots, and how my project can make an impact in filling those spaces. After that, I'll explain the problem I'm solving in a formal problem statement, along with the structure and purpose of the AdjustPath algorithm. Then I will review the experimental setting of the simulation, and following that I'll discuss the results of the experiments. At the end, I will review the project and propose several future projects that could use this thesis as their foundations.

Chapter 2

Related Work

The field of multi-robot path planning is well-researched, showcased throughout surveys covering the broad spectrum of research projects within this ever-growing space [5]. The focus here is on discussing and experimenting with a system of localization and communication for underwater robots, particularly in an environment of intermittent communication.

The primary foundation for this thesis comes from "Sunflower: Locating Underwater Robots From the Air" [3]. The system presented in that paper provides cross-medium localization using visible light lasers. An aerial drone, hovering over the water, can use a laser mounted on its underside to search for an underwater robot. Upon contact, the robot can send data to the drone. With an additional laser, or with a slight modification of the current setup, the drone could send data to the robot as well, making this communication setup bidirectional.

In "Current Algorithms, Communication Methods and Designs for Underwater Swarm Robotics: A Review", the authors discuss various gaps in the current literature regarding underwater robotics [6]. In Section V, they discuss the aforementioned lack of reliable, high broadband communication between fully submerged robots [7, 8]. Not having this communication makes it difficult to coordinate the robots and attempt

to avoid collisions. While it is possible to have acoustic communication underwater, the bitrate is very low, and the waves can be interrupted by hard surfaces [9].

Another example of this particular bitrate issue is in an experiment by Klein et. al. [10], where they were restricted to using low-range radio frequencies to maintain communication within a system of underwater robots The robots were able to send each other data, but only when constrained to a smaller exploration area. Each robot was tasked with staying close to the other robots, which limits the system's capacity for exploration.

After seeing the innovations in localization with Sunflower and communication with Amphilight, the core concept behind this project began to form: how can I use these novel methods to solve these problems? Using the Amphlight system with Sunflower certainly has the potential to have a much higher bitrate than acoustically communicating underwater. And, while there have been developments in communicating with visible light lasers underwater [11, 12], none of them have the consistency or the reliability demonstrated in the Sunflower experiments. Moreover, those underwater lasers still have difficulty with long-distance communication, and in a swarm of robots exploring a potentially large region of water, this could severely hamper their efficiency.

Another paper [13] provided a source of inspiration for this project. This paper posited a method for improving the exploration efficiency of a swarm of robots. A key idea proposed was an iterative method of improving a robot's planned path, in a function called ImprovePath. The algorithm would loop over the current robot's path and attempt to minimize the number of interactions with other robots in the swarm, stopping after reaching a certain threshold of error. While this algorithm was useful for study, it was unable to be directly implemented here. As will be discussed later, ImprovePath was designed for a constantly updating system, and since the drone

takes time to move between robots and communicate with them, this algorithm was not directly applicable in this project.

With the drone taking time to move between robots and communicate with them individually, this leads to an intermittent communication system. The question then arises, can this system be a viable method of communicating between robots? In the paper "A scheme for robust distributed sensor fusion based on average consensus", the authors found that their own communication system was robust to unreliable connections [14]. They disrupted their robots' communication with Gaussian noise, and found that it had little effect on the final outcome. While the proposed multirobot system using Sunflower will have longer communication disruptions due to the nature of the system, this paper provided some preliminary proof that this setup might work. It is entirely possible to have a swarm of robots that cannot constantly communicate with each other, and to optimize that swarm's performance.

For additional methods that handle intermittent communication, we can take a look at how Amigoni et al. defines and classifies these situations [15]. Under their definitions, we would say that this project falls under an "event-based connectivity", where the robots leaves "traces" of their presence with the drone. The major difference between our system and the ones presented in this paper is that we use the drone as a messenger; this is similar to a mobile base station [16], but does not require distance restraints on the exploration robots.

In addition, Quattrini Li mentions more communication problems within multi-robot exploration scenarios [17]. A particularly interesting paper [18] brings up the idea of building and maintaining a graph of the system of robots, updating with new information whenever possible. That paper assumes different information about the environments being explored, but serves as a useful comparison to the experiments performed here. Perhaps an eventual combination, as a longer research project, could

bring some new insights.

Chapter 3

Approach

Section 3.1

Problem Statement

Suppose we have a system of n underwater robots. The goal of these robots is to navigate and gather information within an unknown space. They are able to retrieve a set of sensor data about their immediate environment, and they are aware of their relative positional changes due to on-board accelerometers.

This system of n robots is able to communicate with an aerial drone. This drone is restricted to only communicating with one robot at any point in time. Being over the water, the drone does not have obstacles between each robot's position, and is able to move freely between locations. My goal is to provide an algorithm to reduce inter-robot collisions in this intermittent communication system.

The assumptions for this problem derive directly from the features and limitations of Sunflower. Since the system uses visible light lasers, let us assume the robots have no opaque obstacles above them, blocking communication—for example, the open ocean. Let us also assume that the drone knows its height above the water, as well as the angle and direction of the laser beam that connects to the robot.

Algorithm 1 AdjustPath.

Output: modified planned path for this robot
Require: *thisPath* = this robot's planned path
Require: *otherPaths* = Last known planned paths for other robots
Require: i = current move index in each path

```
for each robot in otherPaths do
    Compare number of remaining moves to thisPath
    Store minimum remaining number of moves in commonMoves
    if commonMoves is zero then
        continue
    end if
    for j < commonMoves do
        Let thisMove = thisPath[i] + j
        Let otherMove = otherPaths[robot][i] + j
        if thisMove equals otherMove then
            Remove thisMove from thisPath
            Let temp = Grid.Location
            Set Grid.Location = obstacle
            Insert Search(previousMove, nextMove) into thisPath[i] + j
            Set Grid.Location = temp
        end if
    end for
end for
```

This information can easily be attained via a barometer (altitude) and precise motor control (laser angle). With this, the location of the receiving robot is updated at every communication with the drone.

The core objective in this problem statement is to propose a system in which Sunflower can be deployed across multiple robots in order to assist in the exploration of an unknown area. The AdjustPath algorithm will serve to improve the efficiency of the exploration by taking advantage of Sunflower's properties, and attempting to negate the downside of having an intermittent communication system.

Section 3.2

AdjustPath

The data we need for this algorithm is the set R. This contains lists of data for each robot r_i, which itself contains two lists and an integer. The first list in r_i is history$_i$, and this contains tuples with location coordinates and values denoting whether the robot found an obstacle at that location. The second list, path$_i$, is an ordered list of coordinates the robot is attempting to reach. The integer index$_i$ denotes where in path$_i$ the robot is currently located.

$$R = \{r_1, r_2, ..., r_n\}$$

$$r_i = \{\text{history}_i, \text{path}_i, \text{index}_i\}$$

$$\text{history}_i = \{(x_1, y_1, v_1), (x_2, y_2, v_2), ...\}$$

$$\text{path}_i = \{(x_1, y_1), (x_2, y_2), ...\}$$

When the drone makes contact with an underwater robot, it transfers the most recently updated version of R. The first action the robot takes is to update its local grid with each history$_i$, improving its knowledge of the map. The robot then calls the function AdjustPath (Algorithm 1) using this data. In this algorithm, the robot loops over every other robot's planned paths. Within each loop, it checks whether there are any moves left in each planned path, stored in the resulting number in *commonMoves*. If there are none, then the information about the other robot's planned path is so out-of-date that it is not useful to compare the two paths, and the loop continues to the next robot's data. If *commonMoves* is greater than zero, we loop over that value and check the indices of the two planned paths.

At any time in this inner loop, when the two path locations are the same, we

begin the modification of the current robot's planned path. We begin by removing the conflicting location from this robot's path. Then we store the current value of the grid location, and temporarily set the location to a designated "obstacle" value. We run the mapping algorithm from the previous move to the next move—and since we set the collision location as "obstacle", the mapping algorithm will treat that as a place that is impossible to move through. After the mapping algorithm completes, we insert the new moves into this path, restore the original grid value of the collision location, and continue to the looping to check for more collisions. With our system of n robots, assuming a maximum *commonMoves* size of m, with an average value where $m > n$, the runtime of AdjustPath is $O(n * m) = O(m)$.

This algorithm was originally intended to be a slightly modified version of the ImprovePath algorithm from "Anytime Planning" [13]. However, ImprovePath assumes there is a constantly updating communication system. Since the drone is only sporadically communicating with the robots, it is inefficient to continually run an algorithm to alter the planned path. AdjustPath only runs in two instances. First, when a robot receives new data from the drone. Second, after a robot has completed its current planned path and begins planning a new path to explore.

Section 3.3

Better Priority Queue

In the experiments, we'll compare this BetterPriorityQueue (Algorithm 2) with a naive approach to the priority queue method. The naive approach has the drone create a priority queue of the robots it has yet to visit, with priority given to the robots that will crash with each other the soonest. In contrast, BetterPriorityQueue provides a more complex system of weights, giving each robot a score based on a combination of when this robot could crash with another and the total time elapsed

Algorithm 2 BetterPriorityQueue.

Require: *unvisitedRobots* = underwater robots that haven't been visited in this cycle
Require: *GetWorstRobot*() = the robot that should be visited the soonest
Ensure: *unvisitedRobots* is not an empty list
  nextRobot = *GetWorstRobot*(*unvisitedRobots*)
  Return *nextRobot*

  Define *GetWorstRobot*(*unvisitedRobots*):
  for each *robot* in *unvisitedRobots* **do**
    obstacles = obstacles encountered by this robot in its planned path
    time = time passed since the last visit from the drone
    score = *obstacles* ∗ *time*
  end for
  Return *robot* with the highest *score*

since last communication.

Chapter 4

Experimental Setting

Section 4.1

Metrics

We have two core metrics for gauging the success of these simulations: percent explored of the unknown environment, and the number of inter-robot collisions. Since this project's goal is to increase efficiency of a swarm of exploratory robots, looking at how much of the environment gets explored is important for gauging how the algorithms are performing.

In the same thread, efficiency is the key idea behind measuring inter-robot collision. These "collisions" are not actually defined as two robots hitting each other; rather, they are the detection of other objects within the same grid cell. Each "collision", therefore, is when at least two robots occupy the same location at the same time. In a real-life scenario, the robots may or may not physically hit in these collisions situations. But if we can successfully avoid as many inter-robot collisions as possible, not only are we making the system more efficient, we can also avoid more situations where hardware could be damaged.

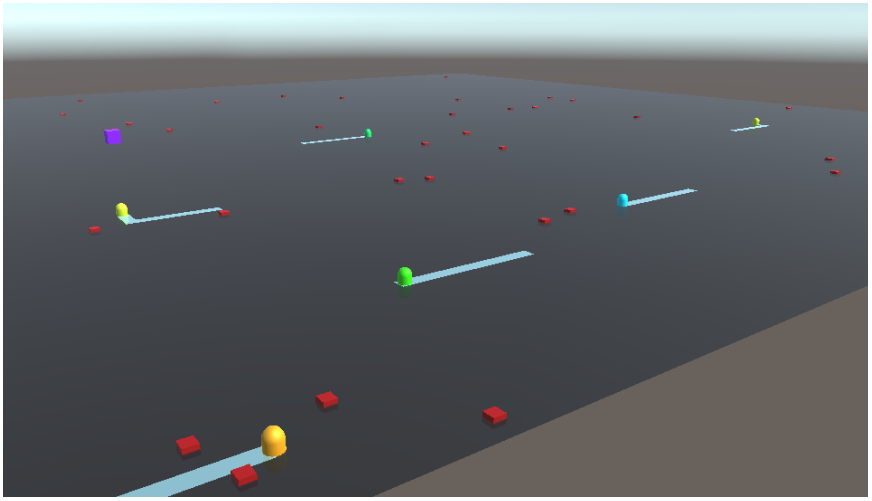

Figure 4.1: Screenshot of the running simulation. The robots are represented by the colored pill-shaped objects; the obstacles, by the red cubes; and the drone, by the purple cube.

Section 4.2

Environment

The simulations for this project were coded in Unity Engine, using C#. The primary motivation for this was visualizing the environment in a realistic, albeit abstracted, 3D view; this allowed me to spot errors and changes between iterations significantly faster. Unity also came with some additional features that happened to streamline the creation of the simulation environment. It has both global and local coordinate systems, allowing each robot to store their own coordinates relative to their starting locations. Unity also has native collision detection, allowing the robots to automatically detect when they're near obstacles or other robots. This functions are working pretty much the same as a visual or sonar-based setup would in hardware experiments.

Since we're using a software environment, a number of real-life variables have been abstracted to create a more quantifiable system. For instance, the area of exploration is a grid-based map, locally defined and stored by each robot individually. Each cell in the grid represents an area of detection, within which each robot can detect

obstacles or other robots. Figure 4.1 provides an example of the beginning of a simulation. The purple cube represents the drone, and the different colored capsules are the underwater robots. The red cubes are obstacles, and the gray plane is the exploration grid.

A* search [19] was used as an exploration mapping algorithm in this project. Each robot would pick an unexplored point on the grid and try to move there. To change A* search, typically used for minimizing the path length between two points, into a mapping algorithm, the generated paths were weighted to avoid traveling through already explored locations. There can certainly be other exploration methods tested in future experiments, but the focus of this project is on creating a system to increase efficiency within the bounds of this intermittent localization and communication system.

To increase exploration difficulty, a number of stationary obstacles were placed around the grid. "Collisions" with these don't add to the number of inter-robot collisions. A* search was additionally adapted to avoid future collisions with these obstacles by "skipping over" that location in the grid. Since these obstacles were unpredictable to each robot, their planned paths could change on the fly, leading to inaccurate predictions by the other robots. Only after the drone came back to that robot could it update the other robots about the obstacle it had encountered, and how that changed its planned path. Much like in the real world, these obstacles provided additional complexity to the system, creating more variation in the planned paths.

Section 4.3

Implementation

The primary algorithm tested was AdjustPath, with a few other settings being modified to compare and contrast results. Each setting was run on four different amounts

of underwater robots: three, five, seven, and ten. For some unknown reason, even when seeding a pseudo-random generator with a specific integer value, the system would produce different results on different runs. This is potentially due to variations in CPU clock speeds over time. It was still important to have a defined pseudo-random seed though, since it helped stabilize the simulation runs as the settings were changed. Due to the slight variability between runs, there were ten iterations on each altered parameter test, and the results are the averages of each of those ten iterations. Each of those runs took around eleven minutes to complete, and the total amount of time for this set of experiments to complete was over seven hours.

Our experiments are defined follows, with processes and communicated data being cumulative between experimental groups:

- Control: no communication between any robots.
- SendData: the drone communicates the path history of all robots to each other.
- AdjustPath: the robots run AdjustPath and send their updated planned paths to the drone.
- PrioritizeDrone: a simple priority queue implemented on the drone
- BetterPriorityQueue: the drone implements the BetterPriorityQueue instead of the previous simple priority queue.

When testing both the original priority queue and the BetterPriorityQueue algorithm, the same AdjustPath was run by each individual robot upon communicating with the drone. Additionally, though, the priority queues were calculated on the drone after the robot would update its adjusted path. This would be used by the drone in its next cycle of communications to get an estimate of how many obstacles that particular robot would encounter. The drone also recorded the time stamp

of when it ended communications with the robot, another factor that was used in BetterPriorityQueue.

Section 4.4

Additional Tests

Using the current Sunflower hardware, the system will have a communication speed of 500 bits per second. To emulate this, there was a timed delay as the drone transferred data to the robot, and as the robot sent its own data back to the drone. The speed of the overall simulation was greatly increased, so a standard delay time of three seconds was chosen for the primary experiments.

Additional experiments were performed on the seven robot system, using the amount of delay as the independent variable. This was to account for potentially different communication speeds in future versions of Sunflower. The delay times for this series of experiments were half a second, three seconds, and ten seconds.

Chapter 5

Results

The results for the primary set of experiments are presented in Figure 5.1. For all four robot swarms, the control group explored significantly less of the total area. Without the drone carrying data between the robots, only the ten robot system managed to explore over 50% of the map, with the other tests falling below that. With the communication of robot path histories, labelled as "SendData", the exploration percentage skyrocketed, with increases of 20-30% of the area explored. The change when adding AdjustPath and the drone's simplistic priority ordering ("PrioritizeDrone") was less significant, but instead had a tighter data spread, with an average standard deviation 15% less than SendData. When we added BetterPriorityQueue, there was significant improvement in the percent explored particularly with the larger robot systems. We went from 72% to 78% exploration in the seven robot system, with an improvement of 8% overall; and in our ten robot system, going from 84% to 92% gave us an improvement of 9.5%.

The biggest improvements from having any form of communication with the drone was in the number of inter-robot collisions. Having communication with the drone enabled the robots to plan future paths away from other robots. Adding the AdjustPath improved on that as well, showing that the function was able to successfully

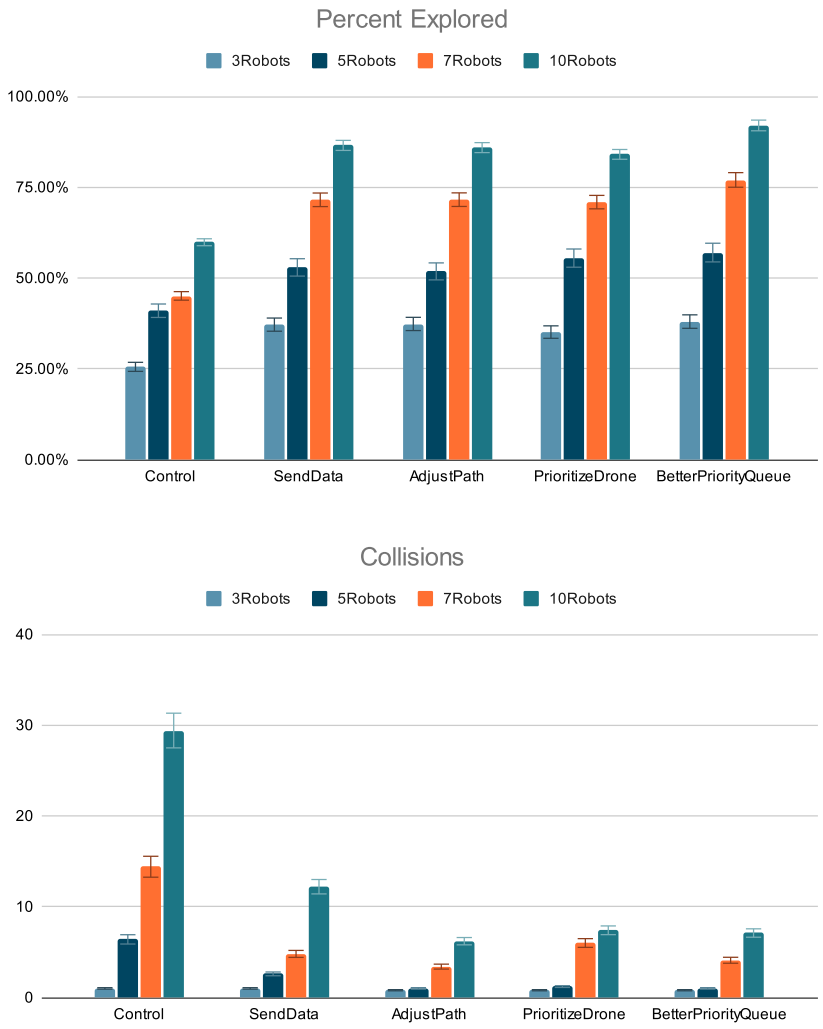

Figure 5.1: Tests on different numbers of robots.

decrease the number of times that two robots were together in the same grid cell.

An initially unexpected result was that the number of inter-robot collisions increased slightly when the drone attempted to assemble the naive priority list of robots to move towards. After some analysis, the reason for these collisions has become apparent. Suppose the drone receives data about the robots' planned paths, and orders its movement as follows.

$$DronePath = \{robot_i, \ldots, robot_j, robot_k\}$$

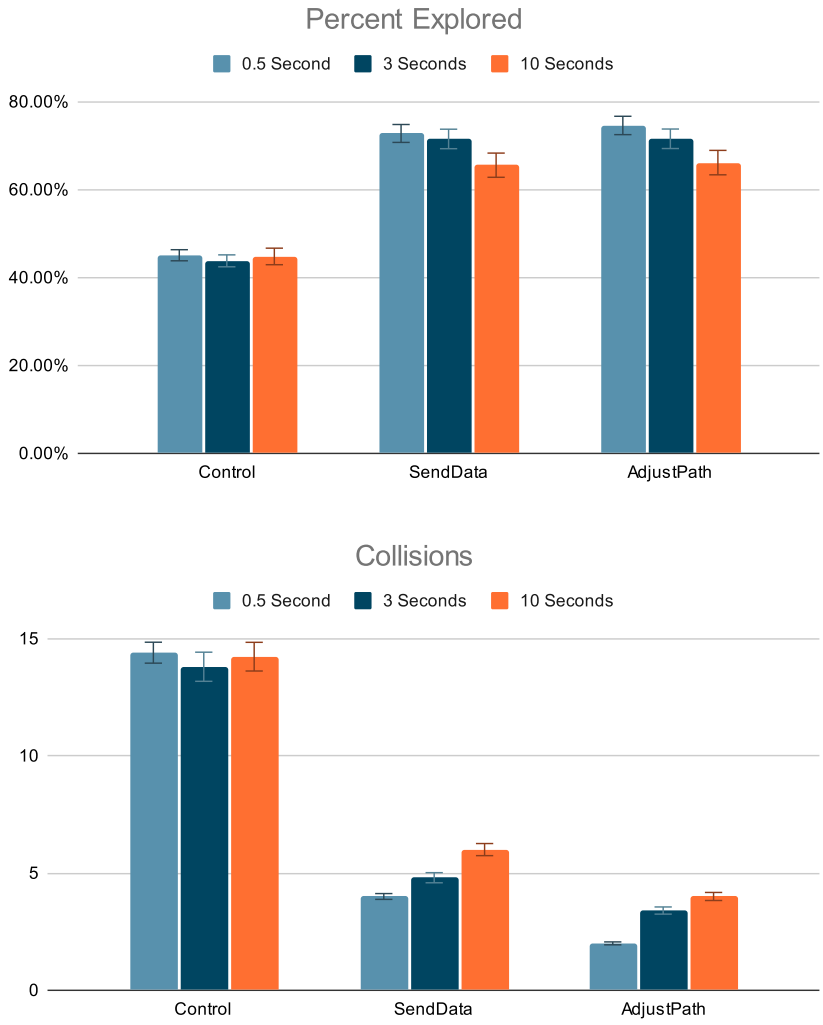

Figure 5.2: Tests on varying delay times.

The drone will fly through these robots in order, attempting to rearrange the remaining robots into better priority lists as it does. Once it reaches $robot_k$, it will create a new priority list. Since it has visited $n - 1$ robots since $robot_i$, the drone will calculate that $robot_i$ has a very low possibility of colliding with the robots it has just recently visited, such as $robot_j$. In most cases, the drone's second priority queue would look almost (if not entirely) inverted from the first priority queue.

$$DronePath = \{robot_k, robot_j, \ldots, robot_i\}$$

With this priority method, the drone ends up visiting some robots only after visiting $2(n-1)$ other robots first. This means that those robots will receive updates almost half as often as in the regular AdjustPath tests. In a system with non-continuous updates, longer delays between communications will lead to more collisions.

The key problem with using the priority queue method on the drone is the growth of planned path uncertainty. The longer the drone is out of communication with any particular robot, the higher the chance that the robot would have hit an obstacle or reached the end of its current planned path. When either of those cases occurs, the drone's data is outdated and therefore inaccurate.

This problem of "planned path uncertainty" was solved by using the BetterPriorityQueue algorithm. By taking the time between visits into account, the drone was able to decrease the amount of time that passed between any two robot visits when compared to the basic priority queue. As a result, not only did the number of collisions decrease, but the percent explored increased as well; with fewer collisions between robots, and improved updating for potential collisions, the robots could avoid wasting time backtracking and spend more time exploring unknown grid areas. As mentioned previously, the average area explored for the seven robots system was 78% when using BetterPriorityQueue, compared to 72% when only using AdjustPath.

In the secondary tests, I modified the drone's delay time as data is transferred between itself and each robot. Figure 5.2 shows how our three different delay times correlate with the amount of the map explored and the number of collisions between robots. The overall spread of this data is not very wide, which leads to an interesting analysis: the amount of time spent with each robot does not seem to greatly change the effectiveness of the drone's communication.

Clearly, there is some effect of changing the delay time. But when using only half a second of "communication time", we might expect the results to be up to six times

better than with the three seconds of delay, and possibly even twenty times better than with ten seconds. However, we can see there were only slight differences in the results. This means that the core improvements over the control group come primarily from the AdjustPath algorithm, and do not rely on having fast communication between the drone and system of robots.

Chapter 6

Conclusion

From running these simulations, we know that the system proposed by Sunflower is capable of increasing the amount of exploration performed by a system of underwater robots. On top of that, we know that the AdjustPath algorithm reliably assists in reducing the number of potential collisions between robots in this system. The goal of this project was to provide simulated tests in which an algorithm could be developed to improve the performance of a system of underwater robots using Sunflower.

To further show improvement from using Sunflower, BetterPriorityQueue builds on AdjustPath, showing improved performance in the total amount of area explored. While the number of collisions slightly increased when the drone did not visit each robot in the same order, the increase in the area explored made up for that. While a perfect system would see near complete exploration, this intermittent communication system has shown improvement with each additional algorithm.

In addition, we have some interesting results showing that the speed of communication is not a major factor in the effectiveness of this algorithm. Knowing this, future hardware implementations of Sunflower might not require a very high bitrate to increase the efficiency of a swarm of robots. This could save time and money with future hardware experiments; the high bitrate communication hardware can signifi-

cantly increase the price, and the implementation of additional code and hardware structure for that expensive equipment is a difficult and lengthy process.

Section 6.1

Future Work

The immediate future application of this project is the implementation of the AdjustPath algorithm onto hardware. The key benefit of the simulation was that AdjustPath could be altered without the costs involved with hardware tests. Now that we have some evidence showing the algorithm works, the next step would be implementing it on hardware and beginning various experiments. Adding BetterPriorityQueue to the aerial drone would further improve the efficiency of the practical system, and would be very interesting to see in a future study.

Specifically for the BetterPriorityQueue algorithm, there are still many additional changes that could be added. These would take time to run, but using a more complex simulation environment could be an improved method of studying how this simulation would work before a hardware implementation is completed. Including dynamic obstacles would also be an intriguing experiment, perhaps as an abstract representation of fish or human divers. In addition, using different depths with the underwater robots were not included in this project, and could serve as a springboard to a number of experiments for future studies.

Bibliography

[1] Lorenzo Mock-Bunting. *The Many Challenges of Underwater Communication*. Accessed: 2022-11-25. 2015. URL: https://schmidtocean.org/cruise-log-post/the-many-challenges-of-underwater-communication/.

[2] Charles J. Carver, Zhao Tian, Hongyong Zhang, Kofi M. Odame, Alberto Quattrini Li, and Xia Zhou. "AmphiLight: Direct Air-Water Communication with Laser Light". In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 373–388. ISBN: 978-1-939133-13-7. URL: https://www.usenix.org/conference/nsdi20/presentation/carver.

[3] Qijia Shao, Charles J. Carver, Samuel Lensgraf, Amy Sniffen, Maxine Perroni-Scharf, Hunter Gallant, Alberto Quattrini Li, and Xia Zhou. "Sunflower: Locating Underwater Robots From the Air". In: *MobiSys '22: Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. Vol. 3. 2. 2022, pp. 1025–1032. DOI: 10.1145/3498361.3539773.

[4] Robin R Murphy, Karen L Dreger, Sean Newsome, Jesse Rodocker, Brian Slaughter, Richard Smith, Eric Steimle, Tetsuya Kimura, Kenichi Makabe, Kazuyuki Kon, et al. "Marine heterogeneous multirobot systems at the great Eastern Japan Tsunami recovery". In: *Journal of Field Robotics* 29.5 (2012), pp. 819–831. DOI: 10.1002/rob.21435.

[5] Hang Ma. "Graph-Based Multi-Robot Path Finding and Planning". In: *Current Robotics Reports* 3.3 (June 2022), pp. 77–84. DOI: 10.1007/s43154-022-00083-8. URL: https://doi.org/10.1007.

[6] Jack Connor, Benjamin Champion, and Matthew A. Joordens. "Current Algorithms, Communication Methods and Designs for Underwater Swarm Robotics: A Review". In: *IEEE Sensors Journal* 21.1 (2021), pp. 153–169. DOI: 10.1109/JSEN.2020.3013265.

[7] M Ben-Ari and F. Mondada. "Swarm Robotics". In: *Elements of Robotics* 15 (2018), pp. 251–265.

[8] M. Joordens and M. Jamshidi. "Consensus control for a system of underwater swarm robots". In: *IEEE Syst. J.* 4.1 (2010), pp. 65–73.

[9] M. Stojanovic and J. Preisig. "Underwater acoustic communication channels: Propagation models and statistical characterization". In: *IEEE Commun. Mag.* 47.1 (2009), pp. 84–89.

[10] Daniel J Klein, Patrick K Bettale, Benjamin I Triplett, and Kristi A Morgansen. "Autonomous Underwater Multivehicle Control with Limited Communication: Theory and Experiment". In: *IFAC Proceedings Volumes* 41.1 (), pp. 113–118. DOI: 10.1109/IPSN.2005.1440896.

[11] Hassan M Oubei, Jose R Duran, Bilal Janjua, Huai-Yung Wang, Cheng-Ting Tsai, Yu-Cheih Chi, Tien Khee Ng, Hao-Chung Kuo, Jr-Hau He, Mohamed-Slim Alouini, Gong-Ru Lin, and Boon S. Ooi. "4.8 Gbit/s 16-QAM-OFDM transmission based on compact 450-nm laser for underwater wireless optical communication". In: *Opt. Express* 23.18 (2015), pp. 23302–23309.

[12] Chao Shen, Yujian Guo, Hassan M Oubei, Tien Khee Ng, Guangyu Liu, Ki-Hong Park, Kang-Ting Ho, Mohamed-Slim Alouini, and Boon S Ooi. "20-meter

underwater wireless optical communication link with 1.5 Gbps data rate". In: *Opt. Express* 24.22 (2016), pp. 25502–25509.

[13] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas. "Anytime Planning for Decentralized Multirobot Active Information Gathering". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1025–1032. DOI: 10.1109/LRA.2018.2794608.

[14] L. Xiao, S. Boyd, and S. Lall. "A scheme for robust distributed sensor fusion based on average consensus". In: (2005), pp. 63–70. DOI: 10.1109/IPSN.2005.1440896.

[15] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. "Multirobot Exploration of Communication-Restricted Environments: A Survey". In: *IEEE Intelligent Systems* 32.6 (2017), pp. 48–57. DOI: 10.1109/MIS.2017.4531226.

[16] Romit Pandey, Arun Kumar Singh, and K. Madhava Krishna. "Multi-robot exploration with communication requirement to a moving base station". In: *2012 IEEE International Conference on Automation Science and Engineering (CASE)* (2012), pp. 823–828.

[17] A. Quattrini Li. "Exploration and Mapping with Groups of Robots: Recent Trends." In: *Curr Robot Rep 1* (2020), pp. 227–237. DOI: https://doi.org/10.1007/s43154-020-00030-5.

[18] Francesco Amigoni, Jacopo Banfi, Nicola Basilico, Ioannis M. Rekleitis, and Alberto Quattrini Li. "Online Update of Communication Maps for Exploring Multirobot Systems Under Connectivity Constraints". In: *International Symposium on Distributed Autonomous Robotic Systems*. 2018.

[19] Amit Patel. *Introduction to A**. Accessed: 2023-04-18. URL: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.